دانشگاه پیام نور
مرکز آران و بیدگل

جزوه درس

هوش مصنوعی

گردآورنده:

مصطفی قبائی آرانی

بنام آنکه جان را فکرت آموخت

پیشگفتار

در این جزوه تلاش شده است تمام سرفصل‌های درس پوشش داده شود. یادآوری این نکته مهم، ضروری است که این جزوه جهت کمک به دانشجویان در جهت کاهش یادداشت برداری، ترسیم شکل‌ها، مثال‌ها در هنگام تدریس بوده است. بنابر این در کنار این جزوه، هر دانشجو مطابق با ذوق و سلیقه خود جزوه‌ای دست نویس، حاوی یادداشت‌هایی به منظور تکمیل و تفهیم هرچه بهتر مطالب خواهد داشت.

این جزوه، ادعای جایگزینی مراجع اصلی این درس را ندارد بلکه تنها خلاصه‌ای از مطالب مهم از مراجع درس است. در گردآوری این جزوه از کتاب ***هوش مصنوعی: رهیافتی نوین*** تالیف راسل و پیتر نورویگ استفاده شده است.

در اینجا لازم است از دانشجویان محترم آقای رجب زاده، آقای محمد مومنی و آقای محمد مقنی که اینجانب را در تایپ و تدوین این جزوه یاری نموده‌اند کمال تشکر را داشته باشم. همچنین در این جزوه احتمال اشتباه وجود دارد، به این جهت در مطالعه مطالب جزوه دقت کافی را داشته باشید و وجود هرگونه ایراد و همچنین نظرات و پیشنهادات خود را از طریق پست الکترونیکی mostafaghobaye@yahoo.com به اطلاع اینجانب برسانید.

مصطفی قبائی آرانی

پاییز ۸۹

تقدیم به:

دانشجویانی که عقلشان بر احساسشان غلبه می‌کند.

دانشجویانی که برای تحمل آرای دیگران تمرین می‌کنند.

دانشجویانی که برای چهل سال آینده خود برنامه دارند.

دانشجویانی که فرق هشت و هشت و یک دقیقه را می‌دانند.

دانشجویانی که رنگهای شاد خلقت را در ظاهر خود سپاس می‌گویند.

دانشجویانی که با محاسبه حروف اضافه سخن می‌گویند.

دانشجویانی که قاعده مند فکر می‌کنند.

دانشجویانی که برای هر سوالی چندین پاسخ متفاوت قائل اند.

دانشجویانی که عصبانیت خود را به تاخیر می‌اندازند.

دانشجویانی که شان را بر قدرت مقدم می‌شمارند.

دانشجویانی که در رفتار قابل پیش بینی‌اند.

دانشجویانی که برای افزایش قدرت کشور تامل می‌کنند.

دانشجویانی که دغدغه‌های وفای به عهد، آنها را از خواب بیدار می‌کند.

(دکتر محمود سریع القلم)

فهرست

فصل اول: مقدمه **۱**

علل مطالعه هوش مصنوعی: ۲
سیستمهای هوشمند: ۲
تفاوت انسانی بودن و منطقی بودن (عقلانیت) : ۲
بررسی دقیق تر هر یک از این چهار رهیافت: ۳
مانند انسان عمل کردن (آزمون تورینگ): ۳
تست تورینگ: ۳
مانند انسان فکر کردن (مدل سازی شناختی): ۴
منطقی فکر کردن (رهیافت قوانین تفکر): ۵
منطقی عمل کردن (رهیافت عامل منطقی): ۵
زیربنای هوش مصنوعی: ۶
تاریخچه هوش مصنوعی: ۷
کاربردهای هوش مصنوعی (وضعیت فعلی هوش مصنوعی): ۹
سوالات تستی آخر فصل ۱۰

فصل دوم: عاملهای هوشمند **۱۳**

عاملها و محیطها ۱۴
عامل ۱۴
مثالهایی از عاملها: ۱۴
ادراک: ۱۴
دنباله ادراکی ۱۵
تابع عامل: ۱۵
عاملها چگونه عمل میکنند؟ ۱۶
تفاوت میان منطقی بودن و عقل کل بودن (همه چیز دانی) ۱۶
تعریف عامل منطقی ایده آل: ۱۷
خودمختاری: ۱۷

تعیین مشخصات محیط کار (PEAS): ۱۸
انواع محیطها: ۱۸
مثال‌هایی از انواع محیط و ویژگی‌های آنها ۲۰
ساختار عاملهای هوشمند ۲۰
انواع برنامه های عامل: ۲۱
برنامه های عامل: ۲۱
عاملهای مبتنی بر جدول: ۲۱
عاملهای واکنشی ساده: ۲۳
ساختار عامل واکنشی ساده ۲۳
عامل واکنشی مبتنی بر مدل: ۲۴
ساختار عامل مبتنی بر مدل ۲۵
عاملهای مبتنی بر هدف: ۲۵
تفاوت عامل واکنشی و هدف گرا: ۲۶
عاملهای مبتنی بر سودمندی: ۲۶
تفاوت معیار کارایی و تابع سودمندی: ۲۷
عاملهای یادگیرنده : ۲۷
سوالات تستی آخر فصل ۲۹

فصل سوم: حل مسئله از طریق جستجو ۳۷

عاملهای حل مسئله: ۳۸
فرموله کردن چند مسئله نمونه: ۴۰
مسئله دنیای جاروبرقی: ۴۰
مسئله معمای ۸: ۴۰
مسئله ۸ وزیر: ۴۱
گسترش : ۴۲
اندازه گیری کارایی حل مسئله : ۴۳
انواع استراتژیهای جستجو: ۴۴
جستجوی اول-سطح: ۴۵
پیچیدگی زمانی و حافظه جستجوی سطحی ۴۶
جستجوی هزینه یکنواخت: ۴۷

جستجوی عمقی: ۴۷
جستجوی عمقی محدود شده: ۴۹
جستجوی عمیق شونده تکراری: ۴۹
پیچیدگی زمانی عمیق کننده تکراری ۵۰
کارایی IDS ۵۰
جستجوی دو طرفه: ۵۱
مقایسه استراتژیهای جستجوی ناآگاهانه ۵۲
اجتناب از حالتهای تکراری : ۵۳
جستجو با اطلا عات ناقص: ۵۴
سوالهای تستی آخر فصل ۵۷

فصل چهارم: جستجو و اکتشاف آگاهانه ۶۷

جستجوهای آگاهانه : ۶۸
جستجوی اول- بهترین: ۶۸
جستجوی حریصانه: ۶۹
جستجوی A*: ۷۰
هیورستیک قابل قبول: ۷۱
هیورستیک سازگار(یکنوا): ۷۲
خصوصیات A* ۷۴
کانتور: ۷۴
جست و جوی اکتشافی با حافظه ی محدود (IDA*): ۷۴
جست و جوی اولین- بهترین بازگشتی (RBFS): ۷۵
الگوریتم SMA*: ۷۶
خواص SMA* ۷۷
توابع هیورستیک: ۷۸
اثر کیفیت تابع هیورستیک بر کارایی: ۸۱
تسلط: ۸۲
مقایسه جستجوهای (h1) A*، (h2)A*، IDS برای حل صد نمونه مسئله تصادفی معمای ۸: ۸۲
مقایسه بین هزینه جستجو و فاکتور انشعاب موثر ۸۳
مسا ئل تعدیل شده: ۸۳

الگوریتمهای جست وجوی محلی: ۸۴
سوالات تستی آخر فصل ۹۵

فصل پنجم: مسائل ارضای محدودیت(CSP) ۱۱۳

مسائل ارضای محدودیت (CSP): ۱۱۴
گراف محدودیت: ۱۱۵
مسئله رمزنگاری (معمای ریاضیات) : ۱۱۵
مسئله ۴-وزیر: ۱۱۶
مزایای بیان مسئله به صورت CSP : ۱۱۶
انواع مسائل CSP: ۱۱۷
انواع محدودیتها: ۱۱۷
محدودیتهای اولویت دار: ۱۱۸
جستجوی عقبگرد برای CSP: ۱۱۸
حل مساله ۴-وزیر به روش عقبگرد ۱۱۸
حل مساله رنگ آمیزی نقشه به روش عقبگرد ۱۱۹
ترتیب انتخاب متغیرها : ۱۱۹
بررسی پیشرو (FC) ۱۲۱
پخش (انتشار) محدودیت: ۱۲۳
سازگاری یال : ۱۲۳
سازگاری k (K- consistency) : ۱۲۴
تعریف K- consistence : ۱۲۵
ترتیب تشخیص تناقض: ۱۲۵
جستجوی محلی برای CSP: ۱۲۵
سوالات تستی آخر فصل ۱۲۶

فصل ششم: جستجوی رقابتی ۱۳۵

مقدمه: ۱۳۶
دلایل مطالعه بازیها ۱۳۶
انواع بازیها: ۱۳۷
الگوریتم Min-Max: ۱۳۸

یک نمونه بازی (tic - tac - toe): ۱۳۸
بازیهای چند نفره: ۱۴۰
هرس آلفا-بتا (α-β): ۱۴۱
تصمیمات بلادرنگ ناقص : ۱۴۴
توابع ارزیابی: ۱۴۴
قطع جست وجو: ۱۴۵
حالت ساکن: ۱۴۵
اثرافق: ۱۴۶
بازیهایی که دارای عامل شانس هستند: ۱۴۶
ارزیابی موقعیت در بازیها با عنصر شانس: ۱۴۷
سوالات تستی آخر فصل ۱۴۹

فصل هفتم: عاملهای منطقی ۱۵۷

عامل مبتنی بر دانش: ۱۵۸
دنیای وامپوز: ۱۵۹
ویژگیهای محیطی دنیای وامپوز: ۱۵۹
منطق: ۱۶۰
استلزام: ۱۶۱
صحت: ۱۶۴
کامل بودن: ۱۶۴
منطق گزاره ای: ۱۶۴
گزاره: ۱۶۴
روابط ترکیبی: ۱۶۴
جدول صحت: (معنای منطق گزاره ها) ۱۶۵
گرامر منطق گزاره ها (نحو منطق گزاره ها) ۱۶۵
روش انتفاع مقدم : ۱۶۵
روش صحت تالی: ۱۶۵
هم ارزیهای مهم: ۱۶۶
گزاره نما: ۱۶۷
سورها: ۱۶۷

ترکیب سورها: ۱۶۷
نقیض سورها: ۱۶۷
استنتاج: ۱۶۸
قوانین استنتاج: ۱۶۸
روش برهان خلف: ۱۶۹
هم ارزی: ۱۶۹
معتبر بودن: ۱۷۰
ارضا شدنی: ۱۷۰
الگوهای استدلال در منطق گزاره ای: ۱۷۰
اثبات از طریق قوانین استنتاج: ۱۷۱
خاصیت یکنوایی: ۱۷۱
اثبات با استفاده از جدول صحت: ۱۷۱
تحلیل: ۱۷۲
فرمهای نرمال: ۱۷۳
فرم نرمال عطفی (cnf): ۱۷۳
الگوریتم Resolution: ۱۷۳
عبارات هورن: ۱۷۴
زنجیره ساز رو به جلو(FC): ۱۷۵
زنجیره ساز رو به عقب(BC): ۱۷۶
مقایسه دو روش FC, BC: ۱۷۷
کاربرد زنجیرهای رو به جلو و زنجیرهای رو به عقب: ۱۷۷
سوالات تستی آخر فصل ۱۷۹

فصل هشتم: منطق مرتبه اول ۱۸۷

مروری بر ویژگیهای منطق گزاره ها: ۱۸۸
منطق مرتبه اول(FOL) ۱۸۸
انواع رابطه ها: ۱۸۸
انواع منطق: ۱۸۹
مدلهای منطق مرتبه اول: ۱۹۰
نحو منطق مرتبه اول: ۱۹۱

گرامر منطق مرتبه اول : ۱۹۲
Term: ۱۹۲
جملات اتمیک :..... ۱۹۲
جملات پیچیده:..... ۱۹۳
سورها:..... ۱۹۳
سورعمومی: ۱۹۳
سور وجودی: ۱۹۴
خصوصیات سورها: ۱۹۴
سورهای تو در تو: ۱۹۵
مفهوم تساوی: ۱۹۶
ادعاها و پرسشها در منطق مرتبه اول: ۱۹۶
دامنه ی خویشاوندی: ۱۹۶
دامنه مجموعه ها: ۱۹۷
مهندسی دانش: ۱۹۸
دامنه مدارهای الکتریکی ۱۹۹
سوالات تستی آخر فصل ۲۰۱

فصل نهم: استدلال در منطق مرتبه اول ۲۰۵

مقدمه: ۲۰۶
قانون حذف سور عمومی: ۲۰۶
نمونه سازی عمومی Universal instantiation ۲۰۶
Subst({v/g}, α): ۲۰۶
قانون حذف سور وجودی:..... ۲۰۶
نمونه سازی وجودی Existential instantiation ۲۰۷
تقلیل به استنتاج گزاره ای: ۲۰۷
تئوری Herbrond :..... ۲۰۸
تئوری چرچ و تورینگ: ۲۰۸
قانون انتزاع تعمیم یافته GMP : ۲۰۸
یکسان سازی:..... ۲۰۹
نکات تکمیلی در مورد یکسان سازی:..... ۲۱۰

فراکردهای معین منطق مرتبه اول: ۲۱۰
پایگاه دانش نمونه: ۲۱۱
الگوریتم زنجیره ساز رو به جلو: ۲۱۲
اثبات به روش زنجیره ی مستقیم ۲۱۲
پایگاه دانش Datalog: ۲۱۳
زنجیره ساز به جلو افزایشی: ۲۱۳
الگوریتم زنجیره ساز روبه عقب: ۲۱۳
برنامه نویسی پرولوگ (منطقی): ۲۱۶
نکاتی دیگر در مورد پرولوگ: ۲۱۷
دستور Cut: ۲۱۷
تحلیل: ۲۱۸
مراحل تبدیل به CNF: ۲۱۸
تبدیل به فرم CNF ۲۱۸
استاندارد سازی متغیرها: ۲۱۹
فرایند حذف سور وجودی (اسکولمایز): ۲۲۰
مثال در مورد Resolution: ۲۲۰
الگوریتم Resolution: ۲۲۱
اثبات مجرم بودن West توسط Resolution: ۲۲۲
مثالjake,Tunaبه وسیله Resolution: ۲۲۲
استراتژیهای تحلیل: ۲۲۳
سوالات تستی آخر فصل ۲۲۵

فصل اول: مقدمه

آنچه در این فصل خواهید آموخت:

- ❖ **رهیافت‌های هوش مصنوعی**
- ❖ **مبانی هوش مصنوعی**
- ❖ **تاریخچه هوش مصنوعی**

علل مطالعه هوش مصنوعی:

i. یادگیری بیشتر در مورد خودمان

ii. استفاده و بهره برداری از موجودیت‌ها و سیستم‌هایی که توسط هوش مصنوعی تولید می شوند.

سیستم‌های هوشمند:

i. **نرم افزاری:** مثل نرم افزارهای هوشمند نظیر نرم افزارهای تشخیص چهره یا تشخیص صدا

ii. **سخت افزاری:** مثل ربات

افراد مختلف، دیدگاه‌های متفاوتی نسبت به هوش مصنوعی دارند و برای آنها در ساخت سیستم‌های هوشمند، دو سوال مطرح است:

- آیا بیشتر به رفتار اهمیت می دهید یا به تفکر و استدلال؟
- آیا در ساخت سیستم‌های هوشمند، انسان را الگو قرار می‌دهید یا بر اساس یک استاندارد ایده آل هوشمندی (منطقی بودن، عقلانیت) عمل می‌کنید؟

با توجه به این دو سوال، هوش مصنوعی به چهار رهیافت تقسیم می شود:

انسان گونه فکر کردن	منطقی فکر کردن
انسان گونه عمل کردن	منطقی عمل کردن

تفاوت انسانی بودن و منطقی بودن (عقلانیت) :

منطقی بودن یعنی یک سیستم بر اساس دانش خود بهترین کار ممکن را در یک لحظه انجام دهد در حالیکه انسان در هر لحظه نمی‌تواند بهترین کار ممکن را در یک لحظه انجام دهد چون انسان‌ها کامل نیستند.

منطقی فکر کردن ۱. ۲.	مانند انسان فکر کردن ۳. ۴.
منطقی عمل کردن ۵. ۶.	مانند انسان عمل کردن ۷. ۸..

تعاریف مختلف برای هوش مصنوعی از دیدگاه دانشمندان مربوط به رهیافت‌های مختلف:

۱. مطالعه توانایی‌های ذهنی از طریق مدل‌های کامپیوتری (محاسباتی).

۲. مطالعه محاسباتی که امکان مشاهده، درک و استدلال را فراهم می‌نماید.

۳. تلاش برای ساختن کامپیوترهایی که فکر کنند، ماشین‌هایی با قدرت تفکر و حس کامل

۴. خودکار سازی فعالیت‌هایی که با تفکر انسان در ارتباط هستند مثل یادگیری، تصمیم گیری و حل مسئله

۵. هوش محاسباتی(AI)، شامل مطالعه عامل‌های هوشمند است.

۶. هوش مصنوعی با رفتار هوشمندانه مصنوعات دست بشر سر و کار دارد.
۷. هنر خلق ماشین‌هایی که توانایی عملیاتی داشته باشند که آن عملیات اگر بخواهد توسط انسان انجام شود، نیاز به هوش دارد.
۸. مطالعه چگونگی ساخت کامپیوترهایی که کارها را در هر لحظه بهتر از انسان‌ها انجام می‌دهند.

نکته: از نظر تاریخی هر چهار رهیافت دنبال شده‌اند و هرکدام طرفداران مخصوص به خود را دارد ولی تنشی بین رهیافت‌های انسانی و رهیافت‌های منطقی وجود دارد به دو دلیل:

- نگرش مبتنی بر انسانی، جزء علوم تجربی است که شامل فرضیات و تاثیر آن توسط تجربیات است.
- نگرش منطقی، ترکیبی از ریاضیات و مهندسی است که با منطق سر و کار دارد.

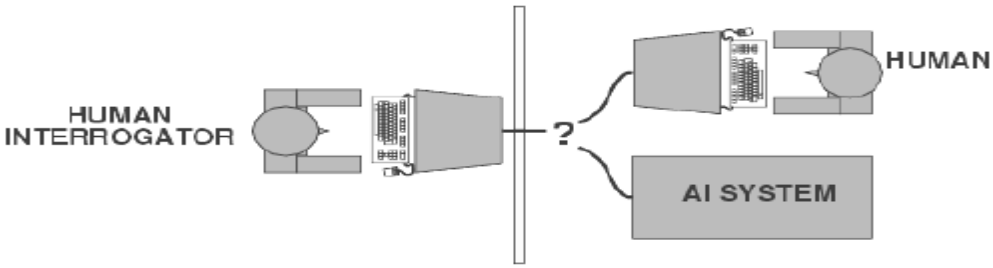
بررسی دقیق تر هر یک از این چهار رهیافت:

مانند انسان عمل کردن[1] (آزمون تورینگ):

اگر سیستمی بخواهد مانند انسان عمل کند باید از طریق تست تورینگ آزموده شود. تورینگ برای اولین بار تعریف علمی رضایت بخشی از هوش مصنوعی ارائه کرد، یعنی برای تست سیستم هوشمند، به جای پیشنهاد لیستی از پارامترها و سوالات که شاید هم بحث برانگیز باشد تست تورینگ را مطرح کرد.

تست تورینگ:

پرسش گر از طریق تله تایپ[2] یک سری سوالات را برای دو کامپیوتر مطرح می کند، تا تشخیص بدهد در کدام کامپیوتر، برنامه هوشمند و در کدام کامپیوتر، انسان معمولی قرار دارد. اگر پرسش گر فریب بخورد و متوجه نشود آن گاه برنامه هوشمندی که طراحی کرده‌ایم با موفقیت از آزمون تورینگ بیرون می‌آید.



کامپیوتری که بخواهد تست تورینگ را انجام دهد باید قابلیت‌های زیر را داشته باشد:

- **پردازش زبان طبیعی[3]:** جهت توانایی در برقراری ارتباط موفقیت آمیز به زبان انگلیسی یا زبان‌های انسانی دیگر
- **بازنمایی دانش[1]:** به منظور ذخیره سازی آنچه که از قبل می‌داند و یا در حین آزمون آن را به دست می‌آورد.

---

[1] Act like humans

[2] Teletype

[3] Natural Language Process

- **استدلال خودکار**[۲]: جهت استفاده از اطلاعات ذخیره شده در پاسخگویی به سوالات و کسب نتایج جدید
- **یادگیری ماشینی**[۳]: تا خود را با شرایط تازه وفق دهد و الگوهارا کشف و برون ریزی کند.

نکته: تست تورینگ از ارتباط فیزیکی مستقیم بین کامپیوتر و پرسش گر اجتناب می‌کند، زیرا شبیه سازی فیزیکی شرط هوشمندی نیست. تست کامل تورینگ، شامل یک سیگنال ورودی است که مصاحبه کننده از طریق آن توانایی ادراک اشیاء را دارد. به عبارت دیگر، اعمال فیزیکی ربات و انسان برای مصاحبه کننده به صورت یک سیگنال ویدئویی نشان داده می‌شود. بنابراین مشخصات فیزیکی آنها(مصاحبه شوندگان) مشخص نیست بلکه فقط عملکرد آنها در رابطه با اشیاء مورد محاسبه قرار می‌گیرد. لذا در تست جامع تورینگ به دو قابلیت زیر نیز نیاز است:

- **بینایی کامپیوتر**[٤]: برای درک اشیاء
- **رباتیک**[٥]: برای جابجایی و کنترل اشیاء

مانند انسان فکر کردن[٦] (مدل سازی شناختی):

اگر بخواهیم ادعا کنیم برنامه هوشمند ما مانند انسان فکر می‌کند ابتدا باید بررسی کنیم انسان چگونه فکر می‌کند برای این کار دو راه وجود دارد:

I. از طریق درون گرایی (سعی در به دست آوردن طرز تفکر):

اگر قادر به ایجاد تئوری دقیقی از ذهن باشیم آن گاه می‌توان این تئوری را به برنامه کامپیوتری تبدیل کرد. نویل و سیمون در سال ۱۹۶۱برنامه GPS[۷] (حل کننده مسائل عمومی) را طراحی کردند آنها تنها به فکر حل کردن مسائل نبودند بلکه سعی کردند در حل یک مسئله خاص، مراحل استدلال برنامه خود را با مراحل تحلیل و استدلال انسان برای آن مساله، مقایسه و تطبیق دهند. در همین زمان محققان دیگری مانند وانگ فقط در پی پاسخ درست بوده‌اند بدون توجه به رفتار انسانی.

II. از طریق آزمایشات روان شناسی:

علم شناخت[۸]، یک علم میان رشته‌ای است که مدل‌های کامپیوتری هوش مصنوعی و فنون تجربی روانشناسی را ترکیب می‌کند تا بتواند تئوری‌های دقیقی از کارکرد ذهن بدست آورد.

---

[۱] Knowledge representation

[۲] Automated Reasoning

[۳] Machine Learning

[٤] Vision

[٥] Robotic

[٦] Think like humans

[۷] General problem solver

[۸] Cognitive

منطقی فکر کردن[۱] (رهیافت قوانین تفکر):

ارسطو از اولین کسانی بود که تلاش‌هایی را برای تدوین تفکر درست انجام داد قیاس صوری[۲]ارسطو معروف است این قیاس می گوید که« با داشتن مقدمات درست، نتایج درست به دست می آید. »

مثال از قیاس صوری:

مقدمات (مفروضات):

- ❖ سقراط یک انسان است
- ❖ همه انسان‌ها فانی هستند.

نتیجه: سقراط فانی است

این مجموعه قوانین و یکسری قوانین دیگر باعث بوجود آمدن علم منطق شد. در سال ۱۹۶۵ برنامه‌هایی وجود داشتند که قادر بودند با وجود حافظه و زمان کافی شرحی از مسأله به زبان منطق را دریافت کنند و اگر راه حلی وجود داشته باشد آن را پیدا کنند. رهیافت منطقی فکر کردن با دو مشکل عمده روبرو بود:

- ❖ دریافت دانش غیررسمی و تبدیل آن به شکل رسمی توسط علائم منطقی کار ساده‌ای نیست مخصوصاً اگر درجه اطمینان دانش، کمتر از ٪۱۰۰ باشد
- ❖ تفاوت بزرگی بین قابل حل بودن مساله در تئوری و انجام آن در عمل وجود دارد یعنی مسائلی وجود دارند که با تعداد کمی فرضیات می‌تواند کامپیوتر را به بن بست محاسباتی بکشاند.

منطقی عمل کردن[۳] (رهیافت عامل منطقی):

اگر سیستمی بخواهد منطقی عمل کند لازمه‌اش این است که ابتدا منطقی فکر کند یعنی این رهیافت تمام رهیافت-های قبلی را تحت پوشش خود قرار می‌دهد. بنابراین این رهیافت جامع ترین است. عامل‌های کامپیوتری ویژگی‌های دیگری از قبیل عملکرد خود مختار، ایستادگی در مدت زمان طولانی، وفق پیدا کردن، ... را نیز باید داشته باشند. در نگرش قوانین تفکر یا منطقی فکر کردن تاکید عمده بر روی استنتاج‌های صحیح بوده است در حالیکه تولید استنتاج-های قسمتی از یک عامل منطقی است به عبارت دیگر عملکرد منطقی فقط شامل استنتاج درست نیست زیرا در بعضی موقعیت‌ها تصمیم صحیح و اثبات شده‌ای برای انجام وجود ندارد اما عامل باز هم باید عملی را انجام دهد حتی در بعضی موارد نمی‌توان بر اساس استنتاج‌های صحیح تصمیم گیری نمود مثلاً عقب کشیدن دست بدون تفکر از شی داغ، عکس العملی است که نسبت به عمل ناشی از تفکر انسانی موفق تر است. بنابراین ما هوش مصنوعی را از دیدگاه عامل عقلانی (رهیافت چهارم) بررسی می‌کنیم این کار دو مزیت دارد:

- ❖ اینکه رهیافت چهارم بسیار عمومی‌تر از قوانین تفکر هستند زیرا فکر کردن زیر مجموعه عمل کردن است یعنی برای منطقی عمل کردن ابتدا باید منطقی فکر کنیم(دلیل برتری بر رهیافت ۳)

---

[۱] Think rationally

[۲] Syllogism

[۳] Act rationally

- رهیافت چهارم در مقایسه با رهیافت‌های مبتنی بر تفکر و رفتار انسان بیشتر تابع پیشرفت‌های علمی می‌باشد چون پیشرفت‌های علمی خیلی قانون پذیر تر از رهیافت‌هایی است که مبتنی بر تفکر یا رفتار انسان می‌باشند علاوه بر این، رهیافت‌های مبتنی بر تفکر یا رفتار انسان پیچیده و ناشناخته و دور از دسترس ما هستند. (دلیل برتری بر رهیافت‌های ۱و۲)

نکته: در محیط‌های پیچیده رسیدن به عقلانیت کامل (رسیدن به منطق کامل و انجام اعمال صحیح) امکان پذیر نیست چون محاسبات زیادی را می‌برد بنابراین فرضیه سودمندی را می‌پذیریم و عقلانیت کامل حالت تئوریک وتحلیلی دارد عقلانیت محدود یعنی درست عمل کردن هنگامی که وقت کافی برای انجام محاسبات وجود داشته باشد.

زیربنای هوش مصنوعی:

زیربنای هوش مصنوعی یعنی رشته‌هایی که به نوعی با هوش مصنوعی در ارتباط بوده و باعث بوجود آمدن آن شده‌اند که آن رشته‌ها عبارت‌اند از:

I. فلسفه

فلاسفه هوش مصنوعی را قابل تصور و انجام پذیر ساختند آنها این کار را با این فرض انجام می‌دهند که ذهن از برخی جهات مانند ماشینی است که بر اساس دانشی که به زبان داخلی خاصی بیان گردیده عمل می‌کند و این تفکر می‌تواند برای انتخاب اقداماتی که انجام خواهد شد استفاده شود.

II. ریاضیات

ریاضیدانان، ابزارهای بکارگیری گزاره‌های منطقی قطعی و همچنین گزاره‌های غیر قطعی و احتمال را فراهم کردند این افراد همچنین اقدامات اولیه درک محاسبات و استدلال را در مورد الگوریتم‌ها انجام دادند.

III. اقتصاد

اقتصاددانان، مسئله تصمیم گیری که نتایج مورد انتظار تصمیم گیرنده را بیشینه می‌کنند را به رسمیت درآوردند.

IV. روانشناسی

روانشناسان این فرضیه را پذیرفتند که می‌توان انسان‌ها و حیوان‌ها را ماشین‌های پردازش اطلاعات در نظر گرفت. زبان شناسان نشان دادندکه استفاده از زبان با این فرضیه تطابق دارد.

V. کامپیوتر

مهندسین کامپیوتر ابزارهایی فراهم نمودند که کاربردهای هوش مصنوعی را امکان پذیر می‌نمایدحجم برنامه‌ها روبه افزایش است و بدون پیشرفت‌های عظیمی که در سرعت و حافظه صنعت کامپیوتر به وجود آمده است اجرای آنها، امکان پذیر نمی‌باشد.

VI. کنترل و Cybernetic

نظریه کنترل با طراحی دستگاه‌ها سرو کار دارد که بر اساس بازخورد[1] از محیط به صورت بهینه عمل می‌کنند. در ابتدا ابزارهای ریاضی نظریه کنترل کاملاً با هوش مصنوعی تفاوت داشت اما این رشته‌ها در حال نزدیک شدن به یکدیگر هستند.

تاریخچه هوش مصنوعی:

پیدایش هوش مصنوعی (۱۹۴۴-۱۹۵۶)

اشتیاق اولیه، آرزوهای بزرگ (۱۹۵۲-۱۹۶۹)

اندکی واقعیت (۱۹۶۶-۱۹۷۳)

سیستم‌های مبتنی بر دانش: کلید قدرت؟ (۱۹۶۹-۱۹۷۹)

تبدیل هوش مصنوعی به یک صنعت (۱۹۸۰ تاکنون)

بازگشت به شبکه‌های عصبی (۱۹۸۶ تاکنون)

تبدیل هوش مصنوعی به علم (۱۹۸۷ تاکنون)

ظهور عامل‌های هوشمند (۱۹۹۵ تاکنون)

۱۹۴۳: اولین کار جدی در حیطه‌ی هوش مصنوعی توسط مک کالو و والترپیتز انجام شد که آنها برای انجام کارهای خود از ۳ منبع استفاده کرده‌اند.

- دانش فیزیولوژی پایه و عملکرد نرون در مغز
- تحلیل رسمی منطق گزاره‌ها
- نظریه تورینگ در زمینه محاسبات

راسل و وایپ هید، یک مدل عصبی مصنوعی پیشنهاد دادند که در آن هر عصب می‌توانست دو حالت off یا on داشته باشد. بنابراین هر تابع محاسباتی می‌تواند توسط شبکه‌ای از عصب‌های متصل به هم تشکیل گردد و هر تابع را می‌توان توسط ساختارهای ساده شبکه پیاده سازی کرد.

۱۹۵۰: آلن تورینگ اولین بار دید کاملی از هوش مصنوعی را در مقاله‌ای با عنوان « محاسبات ماشینی هوشمند » ارائه داد. در این مقاله مفاهیمی مانند « آزمون تورینگ – یادگیری ماشین و الگوریتم ژنتیک » را مطرح کرد.

۱۹۵۱: هینسکی و ادموندز اولین کامپیوتر شبکه‌های عصبی را طراحی کرده‌اند، که SNARC نام داشت.

۱۹۵۲: آرتور ساموئل برنامه‌ای ساخت که یاد می‌گرفت بهتر از نویسنده‌اش بازی کند در نتیجه این فرضیه که کامپیوترها فقط کاری را انجام می‌دهند که به آن گفته می‌شود را نقض کرد چون یک کامپیوتر نتایج میانی را در خودش ذخیره می‌کرد و با عمل استدلال، کارهای خارق العاده‌ای انجام می‌داد.

۱۹۵۶: بزرگان و دانشمندان این رشته در کنفرانس دارت موث گرد هم آمدند و نام هوش مصنوعی را به جای عقلانیت محاسباتی (نام قدیمی هوش مصنوعی) انتخاب کردند.

---

[1] Feed Back

۱۹۵۸: جان مک کارتی زبان لیسپ را که به بهترین زبان هوش مصنوعی تبدیل شد را تعریف کرد این زبان، یک زبان سطح بالای قدیمی است.

۱۹۶۳: برنامه SAINT توسط جیمز اسلاگل برای حل مسائل انتگرال گیری فرم بسته ارائه شد.

۱۹۶۷: برنامه Student که مسائل جبری داستان دار را حل می‌کرد.

۱۹۶۸: برنامهAnalogy توسط تام ایوانز طراحی شد مسئله‌ای بود که برای حل مسائل قیاس هندسی در آزمون هوش مورد استفاده قرار می‌گرفت.

۱۹۷۳–۱۹۶۶: کند شدن سیر تحقیقات هوش مصنوعی به سه دلیل:

- **پیچیده تر شدن الگوریتم‌های برنامه جدید مثل برنامه ترجمه متون،** ترجمه ماشینی متون ناموفق بود چون برخی از کلمات دارای چندین معنی بودند ثانیا ترجمه باید طبق زبان مقصد باشد. بنابراین ترجمه‌ای ناکارآمد به دست می‌آمد.
- **انجام ناپذیری بسیاری از مسائلی که سعی در حل آنها بود.** در اکثر برنامه‌های هوش مصنوعی برای حل مسائل، ترکیبات مختلف راه حل‌ها را انتخاب می‌کردیم تا زمانی که راه حل مسئله پیدا شود و این روش زمانی ممکن بود که تعداد فرضیات محدود بود ولی در اثبات مسائلی که فرضیات بیشتری داشتند با شکست مواجه می‌شدیم.
- **بکارگیری بعضی محدودیت‌ها روی ساختارهای اساسی مانند محدودیت نمایش پرسپترون دو ورودی،** پرسپترون، ساده‌ترین شبکه عصبی است و چیزهایی که می‌تواند یاد بگیرد را بازنمایی می‌کند ولی پرسپترون دو ورودی را نمی‌توان چنان آموزش داد که بتواند متفاوت بودن دو ورودی را تشخیص بدهد.

۱۹۷۹–۱۹۶۹: برنامه DENDRAL که وظیفه آن حل مسئله ساختار مولکولی یک ماده بر اساس اطلاعات فراهم شده از یک طیف نگار جرمی می‌باشد که ورودی برنامه شامل فرمول اولیه مولکول و طیف جرمی آن می‌باشد مثلا تمام زیر ملکول‌های C6H13NO2 که جرم اتمی آنها برابر m=15 باشد را استخراج می‌کرد. نسخه اولیه برنامه تمام ساختارهای ممکن سازگار با فرمول را تولید می‌کرد که برای مولکول‌ها بزرگ دردسرساز بود چون باید تعداد زیر مولکول‌های زیادتری تولید می‌شد بنابراین به جای استفاده از یک نقطه بیشینه، از دو نقطه بیشینه X2,X1 استفاده کردند تا تعداد حالت‌ها کاهش یابد. برنامه DENDRAL اولین سیستم خبره متمرکز دانش بود یعنی پایگاه دانش و کنترل در یک جا متمرکز بودند و مهارت سیستم از تعداد زیادی قواعد خاص ناشی می‌شود ولی سیستم‌های خبره دیگر مانند MYCIN بخش دانش و کنترل آنها از یکدیگر جدا بودند.

روش‌های ضعیف: روش‌هایی هستند که مبتنی بر یک جستجوی همه منظوره می‌باشند که قدم‌های اولیه یادگیری را بر می‌دارند اما تلاش در جهت یافتن راه حل‌های کامل را ندارند. برنامه DENDRAL از این رهیافت استفاده می-کرد.

نکته: برنامه MYCIN در زمینه پزشکی برای تشخیص عفونت‌های خونی و برای استفاده برخی پزشکان جوان توسط ۴۵۰ قاعده طراحی شده بود.

دو تفاوت MYCIN و DENDRAL:

i. بر خلاف قواعد DENDRAL هیچ مدل نظری عمومی وجود نداشت که بتوان قواعد MYCIN را از آن استخراج کرد این قواعد باید از طریق مصاحبه مفصل با افراد خبره، استخراج می شد. که خود این افراد خبره از کتاب‌های تخصصی و از قواعد سرانگشتی استفاده می‌کردند.

ii. این قواعد باید عدم قطعیتی که در دانش پزشکی وجود داشت را منعکس می‌ساخت بنابراین MYCIN از یک حساب عدم قطعیت به عنوان عوامل قطعیت استفاده می‌کرد.

تبدیل هوش مصنوعی به صنعت(۱۹۸۰تاکنون): اولین سیستم خبره تجاری موفق شرکت DEC کهR_1XCON نام داشت شروع به تنظیم سفارش‌های سیستم کامپیوتری بر اساس نیاز مشتری می‌کرد.

کاربردهای هوش مصنوعی (وضعیت فعلی هوش مصنوعی):

i. **برنامه ریزی و زمان بندی خودمختار:** اولین برنامه خودمختار Remote Agent نام داشت که توسط ناسا برای کنترل فضاپیما طراحی شده بود و بر عملیات فضاپیما در حین اجرای طرح‌ها نظارت می‌کرد.

ii. **انجام بازی:** Deep blue نام اولین کامپیوتری بود که توانست با نتیجه ۳/۵ بر ۲/۵ بر قهرمان شطرنج جهان، گری کاسپارف غلبه کند.

iii. **کنترل خودمختار:** سیستم بینایی کامپیوتری ALVENN به منظور هدایت خودرو در طول مسیر آموزش دیده می‌شود. این سیستم بر روی یک خودرو نصب می‌شد که توسط دوربین‌های ویدئویی که روی ماشین نصب شده بود تصاویر ویدئویی را دریافت می‌کرد و حرکت مناسب را انجام می‌داد.

iv. **تشخیص پزشکی:** مانند سیستم MYCIN

v. **برنامه ریزی ترابری:** در دوران بحران خلیج فارس، امریکا از یک ابزار تحلیل و برنامه ریزی مجدد پویای DART استفاده کرد که برای انجام خودکار برنامه ریزی ترابری و زمان بندی در حمل و نقل استفاده می-شد که تعداد مسافران، خودروها و مبدا و مقصد را به عنوان ورودی دریافت می‌کرد.

vi. **رباتیک:** بسیاری از جراحان از رباتها، برای عمل جراحی استفاده می‌کردند.

vii. **درک زبان و حل مسئله:** Proverb نام برنامه کامپیوتری است که جداول کلمات متقاطع را بهتر از اکثر انسانها حل می‌کرد.

سوالات تستی آخر فصل

ترم اول ۸۷–۸۸

۱. یک عامل عقلانی چه ویژگی‌هایی باید داشته باشد؟

الف. دانش قبلی در مورد محیط ب. قدرت استدلال

ج. توانایی تعامل با محیط د. قدرت تصمیم گیری

ترم دوم ۸۷–۸۸

۲. شیوه آزمون تورینگ در کدام حیطه قرار می‌گیرد؟

الف. مانند انسان عمل کردن ب. مانند انسان فکرکردن

ج. بطور عقلانی فکرکردن د. بطور عقلانی عمل کردن

۳. برنامه GPS کدامیک از تعاریف هوش مصنوعی را یادآور می‌شود؟

الف. مانند انسان عمل کردن ب. مانند انسان فکرکردن

ج. بطور عقلانی فکرکردن د. بطور عقلانی عمل کردن

ترم تابستان ۸۸

۴. تست تورینگ مربوط به کدام رویکرد هوش مصنوعی است؟

الف. عملکرد انسان گونه ب. تفکر انسان گونه ج. تفکر عقلانی د. عملکرد عقلانی

ترم اول ۸۸–۸۹

۵. انجام عمل واکشی از قبیل " فوری عقب کشیدن دست از روی اجاق داغ " در کدام طبقه از تعاریف هوش مصنوعی مطرح می‌شود؟

الف. تفکر انسان گونه ب. عملکرد انسان گونه ج. تفکر عقلانی د. عملکرد عقلانی

ترم دوم ۸۸–۸۹

۶. کامپیوتری که می‌خواهد در آزمون تورینگ پذیرفته شود به کدام امکان نیاز ندارد؟ (منظور آزمون جامع نمی-باشد.)

الف. پذیرش زبان طبیعی ب. استدلال خودکار

ج. بینایی کامپیوتری د. یادگیری ماشین

ترم تابستان ۸۹

۷. برای اینکه یک عامل در آزمون تورینگ پذیرفته شود به کدام قابلیت نیازی ندارد؟

الف. پردازش زبان طبیعی ب. استدلال خودکار ج. الگوریتم جستجو د. علم روباتیک

ترم اول ۸۹–۹۰

۸. قیاس صوری ارسطو کدام طبقه از تعاریف هوش مصنوعی را به یاد می‌آورد؟

الف. تفکر انسان گونه ب. عملکرد انسان گونه

ج. تفکر عقلانی د. عملکرد عقلانی

ترم دوم ۸۹–۹۰

۹. برای پی بردن به طرز کار واقعی ذهن انسان کدام مورد کاربردی نیست ؟

الف. درون نگری ب.علم شناخت (رویکرد مدل سازی شناختی)

ج.آزمایشهای روانشناسی د.آزمون تورینگ

تابستان ۹۰

۱۰. آزمون تورینگ برای تست هوشمندی در کدام دسته از انواع تعاریف هوش مصنوعی بکار می‌رود؟

الف. عملکرد انسان گونه ب. تفکر انسان گونه

ج. تفکر عقلانی د. عملکرد عقلانی

پاسخ نامه تستی

سوال	گزینه صحیح
۱	الف
۲	الف
۳	ب
۴	الف
۵	د
۶	ج
۷	ج
۸	ج
۹	د
۱۰	الف

فصل دوم: عامل‌های هوشمند

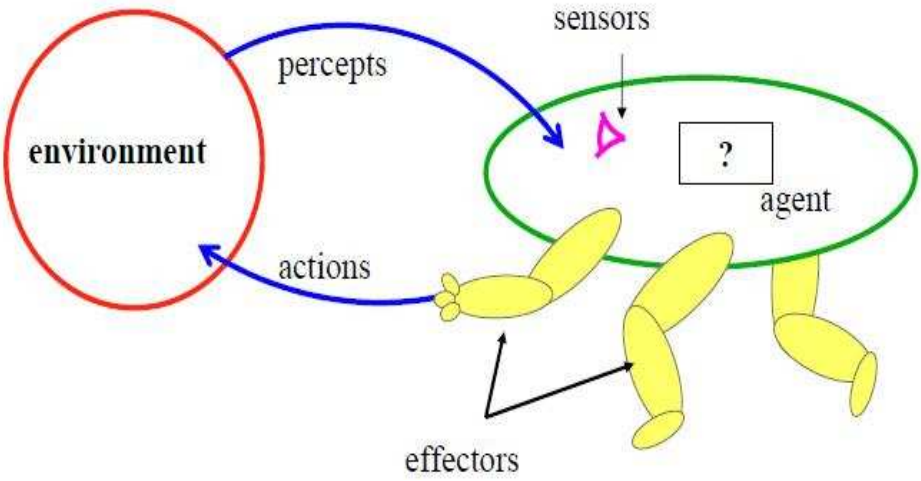
آنچه در این فصل خواهید آموخت:

- ❖ **عامل و مفاهیم مربوطه**
- ❖ **انواع محیط‌ها**
- ❖ **انواع برنامه‌های عامل**

عامل‌ها و محیط‌ها

عامل[۱]:

هر چیزی است که محیط اطراف خود را از طریق حسگرها[۲] ادراک می‌کند و از طریق محرک‌ها[۳] بر روی آن محیط عمل انجام می‌دهد.



عامل‌ها از طریق حسگرها و اقدامگرها با محیط در تعامل هستند

مثال‌هایی از عامل‌ها:

عامل انسانی

- **Sensor:** گوش، چشم، پوست، زبان، بینی،....
- **effector:** دست، پا، دهان، اندام‌های دیگر

عامل روباتیک

- **Sensor:** دوربین، یابنده‌های مادون قرمز
- **effector:** موتور،چرخ‌ها،بازوها

عامل نرم افزاری:

- **sensor:** صفحه کلید
- **effector**: صفحه نمایش

ادراک[۴]:

ورودی‌های ادراکی عامل در هر لحظه (هر آنچه که عامل از طریق سنسورهایش دریافت می‌کند.)

---

[۱] agent

[۲] sensor

[۳] effector

[۴] percept

دنباله ادراکی[۱]:

تاریخچه و سابقه کامل هر چیزی که عامل تاکنون دریافت کرده است.

نکته: انتخاب یک عمل توسط عامل در هر لحظه به کل دنباله ادراکاتی که تاکنون دریافت کرده است بستگی دارد.

تابع عامل:

رفتار یک عامل توسط یک تابع عامل که هر رشته ادراکات ممکن را به یک عمل نگاشت می‌کند، توصیف می‌شود تابع عامل را می‌توان به صورت جدولی درآورد که عامل را توصیف می‌نماید. اگر یک عامل را جهت انجام آزمایش در اختیار داشته باشیم، می‌توانیم با آزمایش، تمام رشته ادراکات ممکن را ثبت و اقداماتی که عامل در پاسخ انجام می‌دهد در قالب یک جدول بسازیم. این جدول ویژگی‌های بیرونی عامل است. از دیدگاه درونی برای یک عامل مصنوعی، تابع عامل به صورت یک برنامه عامل پیاده سازی می‌شود. تابع عامل یک توصیف انتزاعی ریاضی است ولی برنامه عامل یک پیاده سازی واقعی می‌باشد که در معماری در حال اجراست. به عبارت دیگر برنامه عامل، تابع عامل را تحقق می-بخشد. مفهوم عامل ابزاری برای تحلیل سیستم‌هاست و نه یک توصیف انتزاعی که دنیا را به دو گروه عامل‌ها و غیر عامل‌ها تقسیم کند.

مثال: دنیای جاروبرقی

محیط عامل: شامل دو فضای A,B می‌باشد که هر کدام می‌توانند تمیز یا کثیف باشند.

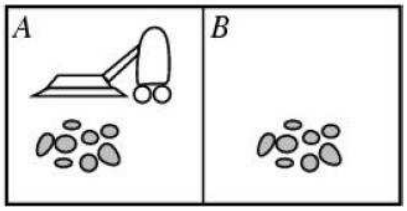
حسگرها: حسگر وضعیت، حسگر تعیین محل

ادراکات: [A,Dirty][A,Clean][B,Clean][B,Dirty]

محرک‌ها: چرخ‌ها، ابزارهای مکش

اعمال: حرکت به چپ، راست، مکش، هیچ کار

نکته: در این مثال ساده، ما می‌توانیم تمام رشته ادراکات ممکن و عملکردهای وابسته را لیست نماییم. این عامل،یک عامل مبتنی بر جدول نام دارد.



[۱] Percept sequence

Percept sequence	Action
[A,Clean]	Right
[A, Dirty]	Suck
[B, Clean]	Left
[B, Dirty]	Suck
[A, Clean],[A, Clean]	Right
[A, Clean],[A, Dirty]	Suck

عامل‌ها چگونه عمل می‌کنند؟

یک عامل منطقی، عاملی است که کار درست انجام می‌دهد. کار درست، عملی است که باعث موفق تر شدن عامل می‌شود. برای اندازه گیری میزان موفقیت یک عامل از واژه‌ای به نام معیار کارایی[۱] استفاده می‌کنیم. بدیهی است که معیار کارایی یکسانی برای همه عامل‌ها وجود ندارد. معیار کارایی باید از دید **طراح عامل** تعیین شود. به عنوان مثال برای جاروبرقی معیارهای کارایی عبارتند از: میزان انرژی الکتریکی مصرف شده، میزان زباله جمع شده در چند ساعت توسط جارو برقی و میزان سر و صدای تولید شده و ...

نکته: زمان ارزیابی معیار کارایی اهمیت دارد. باید معیار کارایی را در دراز مدت ارزیابی کرد.

تفاوت میان منطقی بودن[۲] و عقل کل بودن[۳] (همه چیز دانی)

باید بین منطقی بودن و عقل کل بودن تفاوت قائل شویم:

- یک عامل عقل کل نتیجه واقعی اقداماتش را می‌داند و بر این اساس عمل می‌کند ولی عقل کل بودن در واقعیت غیر ممکن است (داستان شخص پیاده رو و هواپیما)
- منطقی بودن، کارایی مورد انتظار را بیشینه می‌کند در حالیکه عقل کل بودن کارایی واقعی را بیشینه می‌کند و اگر ما انتظار داشته باشیم که یک عامل آنچه را که در عمل بهترین خواهد بود انجام دهد طراحی چنین عاملی غیر ممکن است.

نتیجه اینکه عقل کل بودن نیاز به یک دانش بی نهایت دارد، ولی منطقی بودن یعنی در حدی که به عامل، دانش تزریق کرده‌ایم از او انتظار داشته باشیم.

نکته: به طور خلاصه منطقی بودن یک عامل همیشه به چهارچیز وابسته است:

- معیار کارایی که درجه موفقیت عامل را مشخص می‌کند
- مشاهداتی که عامل تاکنون بوسیله حسگرهایش دریافت کرده است، این تاریخچه ادراکی کامل را دنباله ادراکی می‌نامند.

---

[۱] performance measure

[۲] Rationality

[۳] omniscience

- آنچه را که عامل درباره محیط می‌داند.
- اعمالی که عامل می‌تواند انجام دهد.

این موارد منجر به تعریف یک عامل منطقی ایده آل می‌شود.

تعریف عامل منطقی ایده آل:

« یک عامل منطقی ایده‌آل، به ازای هر رشته ادراکات ممکن باید اعمالی را بر اساس رشته ادراکی و دانش پیش زمینه‌ای که دارد، انجام دهد تا معیار کارایی‌اش را ماکزیمم کند. »

نکته: فاکتورهای لازم برای منطقی عمل کردن عبارت است از:

- **جمع آوری اطلاعات و اکتشافات:** انجام اعمالی که منجر به بدست آوردن اطلاعات مفید یا به عبارتی منجر به تغییر مشاهدات آینده می‌شود، جمع آوری اطلاعات[1] نامیده می‌شود که یک قسمت مهم از منطقی بودن است.
- **یادگیری سیستم با توجه به تجربیات خود**
- **خودمختاری که باید عامل داشته باشد**

نکته: عامل‌های موفق، محاسبه تابع عامل را به سه دوره مختلف تقسیم می‌کنند:

- وقتی عامل طراحی می‌شود مقداری محاسبات توسط طراح عامل انجام می‌شود
- وقتی عامل در حال تصمیم گیری برای انجام عمل بعدی خود می‌باشد محاسبات بیشتری انجام می‌دهد.
- وقتی عامل از تجربیاتش یاد می‌گیرد، محاسبات بیشتری برای تغییر رفتار خود انجام می‌دهد.

خودمختاری[2]:

رفتار یک عامل می‌تواند متکی بر پایه تجربه خود و دانش درونی بنا نهاده شود. اگر عامل فقط بر اساس دانش درونی (پیش زمینه) عمل کند و به ادراکات دریافت شده از محیط توجه نکند فاقد خود مختاری است. بنابراین بهتر این است که عامل خودمختار باشد، یعنی تجربیاتش را نیز در نظر بگیرد.

نکته: در عمل به ندرت از ابتدا نیاز به خود مختاری کامل است. وقتی عامل تجربه‌ای ندارد و یا تجربه کمی دارد به صورت تصادفی عمل می‌کند، چون از ابتدا هیچ ادارکی از محیط نگرفته، بنابراین همان گونه که خداوند برای تکامل حیوانات، به اندازه کافی واکنش‌های ذاتی قرار داده است تا بتوانند آنقدر زنده بمانند و سپس خودشان یاد بگیرند، به طریق مشابه، معقول خواهد بود که برای یک عامل هوش مصنوعی مقداری دانش اولیه برای عامل فراهم کنیم. پس از کسب تجربه کافی از محیط اطرافش، رفتار یک عامل عقلانی می‌تواند به طور موثر، مستقل از دانش قبلی‌اش شود. بنابراین افزودن یادگیری، طراحی یک عامل عقلانی ساده را امکان پذیر می‌کند تا بتواند در محیط‌های گوناگون موفق باشد.

---

[1] Information gathering

[2] Autonomy

تعیین مشخصات محیط کار (PEAS):

برای یک عامل، اولین مرحله تعیین مشخصات محیط کار تا حد امکان به صورت کامل می‌باشد. مواردی مانند مقیاس کارایی (Performance measure)، محیط (Environment)، اقدام گرها (Actuator) و حسگرها (Sensor) تحت عنوان محیط کار (PEAS) توصیف می‌شوند.

عامل: راننده تاکسی اتوماتیک **معیار کارایی**: امنیت، سرعت، راحتی، سود و ... **محیط:** خیابان‌ها، افراد، پیاده، مشتری‌ها و ... **اثرکننده‌ها:** فرمان، شتاب دهنده‌ها، بوق، چراغ‌ها و ... **حسگرها:** دوربین‌ها، حسگرهای صوتی (sonar)، سرعت سنج، GPS، کیلومتر شمار، حسگرهای موتور، صفحه کلید، میکروفون و ...	**عامل:** سیستم تشخیص پزشکی **معیار کارایی:** سلامتی بیمار، به حداقل رساندن هزینه بیمار و ... **محیط:** بیمار، بیمارستان، کارمندان و ... **اثر کننده‌ها:** صفحه نمایش (پرسش‌ها، آزمایش‌ها، تشخیص‌ها، مداوا) **حسگرها:** صفحه کلید (دریافت علایم، یافته‌ها و پاسخ-های بیمار)

انواع محیط‌ها:

I. کاملاً رویت پذیر[1] در مقابل نیمه رویت پذیر[2]:

اگر حسگرهای یک عامل امکان دسترسی به وضعیت کامل محیط در هر لحظه از زمان را به عامل بدهند می‌گوییم محیط کاملاً رویت پذیر است، مانند: صفحه شطرنج، محیط پازل ۸ و محیط جدول کلمات متقاطع. کار در محیط‌های کاملاً رویت پذیر آسان است، زیرا نیازی نیست که عامل، هیچ حالتی را حفظ و ذخیره کند تا بتواند اتفاقاتی که در دنیا روی می‌دهد را ثبت کند. گاهی اوقات، عدم دقت حسگرها و یا نویز باعث می‌شود قسمتهایی از حالت‌ها در داده‌های حسگر حذف شوند، اینگونه محیط‌ها، نیمه رویت پذیر هستند. محیط رانندگی تاکسی یک محیط نیمه رویت پذیر است.

II. قطعی[3] در مقابل غیر قطعی (اتفاقی):

اگر بر اساس وضعیت فعلی و اقدامی که توسط عامل انجام می‌شود وضعیت بعدی محیط به طور کامل تعیین شود گوییم که محیط قطعی است. در غیر این صورت اتفاقی است مثلاً محیط دنیای جاروبرقی قطعی است ولی محیط رانندگی تاکسی اتفاقی است (ممکن است چراغ راهنما قرمز شود یا بنزین تمام شود). اگر یک محیط بدون توجه به اقدامات دیگر عامل‌ها قطعی باشد محیط را استراتژیک یا راهبردی گوییم، مانند محیط بازی شطرنج.

III. مرحله‌ای[4] در مقابل ترتیبی:

مرحله یا Episode شامل ادراک توسط عامل و آنگاه انجام یک عمل می‌باشد. در محیط‌های مرحله‌ای انتخاب عمل در هر مرحله تنها به خود آن مرحله بستگی دارد و در مراحل بعدی تاثیری ندارد. بسیاری از کارهای دسته بندی از این

---

[1] Fully observable

[2] Partially observable

[3] deterministic

[4] Episodic

نوع هستند، مثل عاملی که قطعات معیوب را روی خط مونتاژ شناسایی می‌کند که بدون توجه به تصمیمات قبلی قطعه معیوب را شناسایی می‌کند. در محیط ترتیبی تصمیم فعلی می‌تواند بر تصمیمات بعدی تاثیر بگذارد، مانند شطرنج و رانندگی تاکسی.

نکته: محیط‌های مرحله‌ای نسبت به ترتیبی ساده تر هستند چون عامل نیاز ندارد به جلوتر فکر کند.

IV. ایستا در مقابل پویا:

اگر در فاصله زمانی که عامل تعمق می‌کند (یعنی از لحظه دریافت ادراک تا لحظه انتخاب عمل) محیط نیز تغییر کند می‌گوییم محیط برای آن عامل پویا است، در غیر این صورت محیط ایستاست. هواشناسی و رانندگی تاکسی محیط‌های پویا هستند. در هواشناسی چون در حین تجزیه و تحلیل، دما تغییر می‌کند و در رانندگی ممکن است چراغ‌های راهنمایی در حین رانندگی تغییر کنند. محیط جدول کلمات متقاطع و شطرنج، ایستا می‌باشد.

نکته: به محیطی که با گذشت زمان در حین سنجش شرایطش عوض نمی‌شود اما امتیاز کارایی عامل، تغییر می‌کند محیط نیمه پویا[1] گفته می‌شود. محیط شطرنج بدون ساعت، ایستا و محیط شطرنج با ساعت نیمه پویاست.

سوال: در کدامیک از محیط‌های زیر عامل نیازمند به حفظ وضعیت جاری نیست؟

الف. محیط کاملا قابل مشاهده ب. قطعی ج. گسسته د. پویا

V. گسسته[2] در مقابل پیوسته:

اگر تعداد ادراکات و اعمال عامل را بتوان با اعداد گسسته بیان کرد محیط گسسته، در غیر اینصورت محیط پیوسته می‌باشد. محیط بازی شطرنج وجاروبرقی یک محیط گسسته ولی رانندگی تاکسی یک محیط پیوسته است.

تعریف دیگر:

اگر ادراکات یا اعمال یک عامل مقادیر پیوسته به خود بگیرد محیط پیوسته، ولی اگر مجموعه مقادیر گسسته بگیرد محیط گسسته می‌باشد. به عنوان مثال در محیط رانندگی تاکسی زاویه‌های فرمان یا سرعت، مقادیر پیوسته به خود می‌گیرند.

VI. تک عامله در مقابل چند عامله:

محیطی تک عامله است که فقط یک عامل در آن قرار دارد مانند محیطی که یک عامل در آن جدول کلمات متقاطع را حل می‌کند یا محیط جاروبرقی. در محیط‌های چند عامله، بیش از یک عامل در محیط قرار دارد، مانند محیط بازی شطرنج، که یک محیط دو عامله است. محیط‌های چند عامله به دو دسته محیط‌های چندعامله رقابتی و چند عامله مشارکتی تقسیم می‌شوند. در محیط چندعامله رقابتی، مقیاس کارایی عامل‌ها در تناقض با یکدیگر است مانند محیط بازی شطرنج ولی در چندعامله مشارکتی، مقیاس کارایی عامل‌ها در راستای یکدیگر است مانند محیط رانندگی تاکسی با مقیاس کارایی اجتناب از تصادف. در حالیکه اگر مقیاس کارایی در این محیط پارک کردن در یک محل خاص باشد محیط چند عامله رقابتی می‌باشد.

---

[1] Semi dynamic

[2] Discrete

نکته: حالات نیمه رویت پذیر، اتفاقی، ترتیبی، پویا، پیوسته و چند عامله جزء مشکل ترین محیط‌ها می‌باشند مانند محیط رانندگی تاکسی و محیط سیستم تشخیص پزشکی.

مثال‌هایی از انواع محیط و ویژگی‌های آنها

محیط	قابل دسترسی	قطعی	اپیزودیک	ایستا	گسسته
شطرنج به همراه ساعت	YES	YES	NO	Semi	YES
شطرنج بدون ساعت	YES	YES	NO	YES	YES
پوکر	NO	NO	NO	YES	YES
تخته نرد	YES	NO	NO	YES	YES
راندن تاکسی	NO	NO	NO	NO	NO
سیستم تشخیص پزشکی	NO	NO	NO	NO	NO
سیستم تحلیل تصویر	YES	YES	YES	Semi	NO
ربات جابجا کننده اشیاء	NO	NO	YES	NO	NO
کنترل‌کننده پالایشگاه	NO	NO	NO	NO	NO
آموزش‌دهنده انگلیسی با ارتباط متقابل	NO	NO	NO	NO	YES

ساختار عامل‌های هوشمند

وظیفه طراح هوش مصنوعی، طراحی برنامه عامل است یعنی طراحی تابعی که دنباله ادراکی را به یک عمل نگاشت می‌کند. باید بین تابع عامل و برنامه عامل تفاوت قائل شویم. برنامه عامل، پیاده سازی تابع عامل می‌باشد که بر روی قسمتی از دستگاه‌های محاسباتی که معماری نامیده می‌شود، اجرا خواهد شد. در ضمن، برنامه عامل، ادراک فعلی را به عنوان ورودی دریافت می‌کند، در حالیکه تابع عامل کل تاریخچه ادراکات را به عنوان ورودی دریافت می‌کند. معماری ممکن است یک کامپیوتر یا یک سخت افزار خاص منظوره، مانند پردازشگر تصاویر دوربین یا فیلتر کننده ورودی صدا باشد. به طور کلی معماری، ابتدا مشاهدات و ادراکات را از حسگرها می‌گیرد و برای برنامه عامل قابل دسترس می‌کند، سپس برنامه عامل را اجرا نموده و اعمال انتخاب شده را به محرک‌ها می‌رساند. ارتباط بین عامل، معماری و برنامه به صورت زیر می‌باشد.

معماری + برنامه = عامل

انواع برنامه‌های عامل:

انواع گوناگون طراحی‌های ساده برنامه عامل وجود دارند که نوع اطلاعاتی را که صریحاً وجود دارند و در فرآیند تصمیم گیری استفاده می‌شوند را منعکس می‌سازند. این طراحی‌ها از نظر کارایی، فشردگی و انعطاف پذیری با یکدیگر تفاوت دارند. در حقیقت هر برنامه عامل نحوه انتخاب عمل را مشخص می‌کند. طراحی مناسب برای هر عامل به طبیعت محیط بستگی دارد.

برنامه‌های عامل:

برنامه‌های عامل که معرفی خواهیم کرد همگی طرح کلی یا اسکلت یکسانی دارند. ادراک فعلی را به عنوان ورودی از حسگرها دریافت می‌کنند و یک عمل را به محرک‌ها بر می‌گردانند.

```
AGENT-BASE-PROGRAM
Function SKELETON-AGENT
( Percept ) returns action
Static:
Memory, the agent memory of the world
Memory ← update – memory ( memory, percept )
Action ← choose – best –action ( memory )
Memory ← update – memory ( memory, percept )
Return action
```

عامل‌های مبتنی بر جدول[1]:

برنامه‌های مبتنی بر جدول برای هر ادراک جدید فراخوانی می‌شوند وهر بار یک اقدام را بر می‌گرداند و با استفاده از ساختار داده خاصی مانند لیست پیوندی تاریخچه رشته ادراکات را ذخیره می‌کند. این روش برای عامل‌هایی که تعداد ادراکات آن کم و معلوم باشد مناسب است.

معایب: رویکرد مبتنی بر جدول در ساخت عامل به دلایل زیر محکوم به شکست است:

- جدول مورد نیاز یرای عامل ساده‌ای که تنها قادر به بازی شطرنج باشد 35^{100} سطر خواهد داشت.
- زمان بسیار طولانی لازم است تا طراح قادر به ساخت جدول باشد.
- عامل مبتنی بر جدول فاقد هرگونه خودمختاری است زیرا محاسبه بهترین عمل، کاملاً درونی صورت می‌گیرد و اگر چنانچه شرایط محیط به گونه‌ای غیر قابل پیش بینی تغییر کند، عامل شکست خواهد خورد.
- حتی اگر به عامل، روش یادگیری داده شود تا درجه‌ای از خودمختاری حاصل شود برای یادگیری مقدار صحیح از بین انبوه سطرهای جدول به زمان بی نهایت نیاز خواهد بود.

---

[1] Table-driven

نکته: اگر p مجموعه ادراکات ممکن و T طول عمر عامل یعنی تعداد کل ادراکاتی که دریافت می‌کند باشد، جدول مرجع تعداد p^T داده خواهد داشت. (برای عامل جاروبرقی این فرمول برابر 4^T می‌شود).

نکته: با وجود تمام موارد فوق، عامل مبتنی بر جدول آنچه را که می‌خواهیم انجام می‌دهد، یعنی نگاشت مورد نیاز عامل را پیاده سازی می‌کند. یک نمونه از برنامه عامل مبتنی بر جدول در زیر آمده است.

Table – Driven – Agent

Function TABLE – DRIVEN – AGENT (percept) return action

Static :

Percepts, a sequence, initially empty

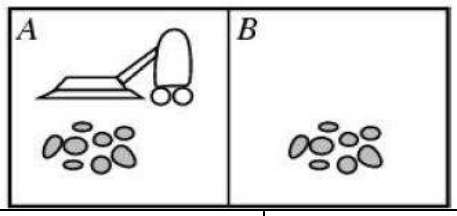
Table, a table, indexed by percept sequences, initially fully specified

Append percept to the end of percepts

Action ← lookup (percepts,table)

Return action

برنامه عامل مبتنی بر جدول برای عامل جاروبرقی در شکل زیر نشان داده شده است.



Percept sequence	Action
[A, Clean]	Right
[A, Dirty]	Suck
[B, Clean]	Left
[B, Dirty]	Suck
[A,Clean], [A, Clean]	Right
[A,Clean], [A, Dirty]	Suck

چهار نوع اصلی برنامه عامل که اصول زیربنایی تقریبا هر سیستم هوشمندی را تشکیل می‌دهند عبارتند از:

- عامل‌های واکنشی ساده[۱]
- عامل‌های واکنشی مبتنی بر مدل[۲]
- عامل‌های مبتنی برهدف[۳]
- عامل‌های مبتنی بر سودمندی[۱]

---

[۱] Simple reflex agent

[۲] Model-based

[۳] Goal-based

عامل‌های واکنشی ساده:

ساده ترین نوع عامل، عامل واکنشی ساده است. این عامل‌ها اقدامات را بر اساس ادراک فعلی انتخاب می‌کنند و تاریخچه ادراکات را نادیده می‌گیرند، یعنی حافظه ندارند. برای مثال عامل جاروبرقی یک عامل واکنشی ساده است زیرا تصمیم آن تنها بر اساس محل فعلی محیط و اینکه آیا آن مکان دارای زباله است یا نه، می‌باشد. به خاطر حذف سابقه ادراک برنامه عامل در مقایسه با جدول آن بسیار کوچک است. بعنوان مثال در برنامه عامل جاروبرقی در مقایسه با جدول آن تعداد حالات ممکن از 4^T به 4کاهش می‌یابد.

نکته: انتخاب عمل بر اساس یک سری قوانین شرط – عمل انجام می‌شوند. قوانین شرط – عمل[۲] از دو نوع اکتسابی و غریزی هستند.

- **اکتسابی:** اگرماشین جلویی چراغ زد، آنگاه توقف کن.
- **غریزی:** اگر شی‌ای به چشم شما نزدیک شود، آنگاه چشم‌های خود را ناخودآگاه می‌بندیم.

نکته: عامل‌های واکنشی ساده این ویژگی قابل ستایش را دارند که ساده هستند ولی کاربرد و هوشمندی بسیار محدودی دارند. عامل واکنشی ساده تنها در صورتی کار می‌کند که بر اساس ادراک فعلی بتواند درست تصمیم بگیرد و این یعنی اینکه محیط کاملاً رویت پذیر باشد. حتی اگر به مقدار اندکی عدم رویت پذیری وجود داشته باشد می‌تواند مشکلات بسیار جدی بروز کند. در مورد عامل‌های واکنشی ساده که در محیط‌های نیمه روئیت پذیر کار می‌کنند حلقه-های بی نهایت اغلب غیر قابل اجتناب هستند. اگر عامل بتواند اقداماتش را تصادفی کند گریز از حلقه‌های بی نهایت امکان پذیر است. برای مثال اگر عامل جاروبرقی تمیز را ادراک کند، برای انتخاب بین چپ و راست، سکه بیندازد که به طور متوسط در دو مرحله به مربع دیگر خواهد رسید. بنابراین یک عامل واکنشی ساده تصادفی شده ممکن است از یک عامل واکنشی ساده قطعی بهتر عمل کند. بنابراین در محیط‌های تک عاملی، تصادفی عمل کردن، چندان منطقی نیست ولی در محیط‌های چند عامله عاقلانه است.

ساختار عامل واکنشی ساده

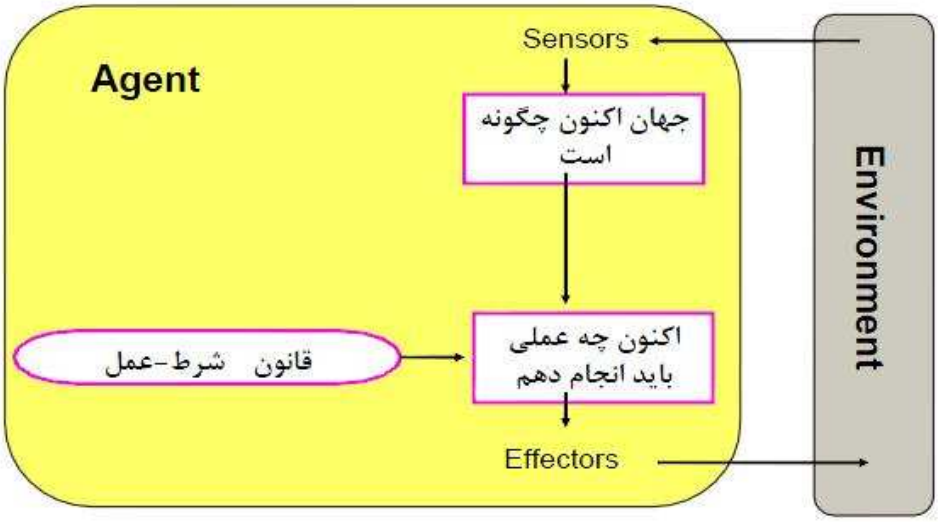
برنامه عامل واکنشی ساده و ساختار آن در شکل زیر آمده است.

```
Function SIMPLE – REFLEX – AGENT ( percept ) returns action
Static: Rules a set of condition – action – rules
State ← INTERPRET – INPUT ( percept )
Rule ← RULE – MATCH ( state,rules )
Action ← RULE – ACTION [rule]
return action
```

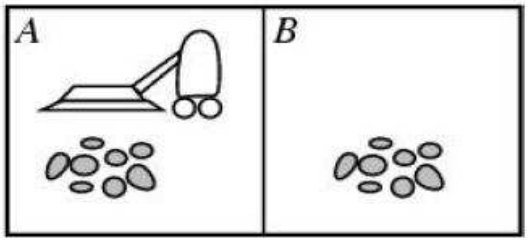
---

[۱] Utility-based

[۲] Condition-action Rule



مثالی از عامل واکنشی ساده برای عامل جاروبرقی:




```
Function REFELEX-VACUUM-AGENT ( [location,state] )
Return an action
        If status == Dirty then return Suck
        Else if location == A then return Right
        Else if location == B then return Left
```

عامل واکنشی مبتنی بر مدل:

همانطور که در روش قبل (عامل واکنشی ساده) اشاره شد گاهی اوقات محیط نیمه رویت پذیر است. موثر ترین روش برخورد عامل با محیط نیمه رویت پذیر نگه داشتن سوابق آن بخش از دنیا است که اکنون عامل نمی‌تواند آن را ببیند. این بدان معنی است که عامل باید به نوعی حالات داخلی را نگه داری کند که به تاریخچه ادراکات وابسته است. بهنگام سازی اطلاعات وضعیت داخلی همزمان با گذر زمان نیازمند دو نوع دانش کد شده در برنامه عامل است.

نیازمند آن هستیم که برخی اطلاعات در باره چگونگی تغییر جهان، مستقل از عامل را داشته باشیم.

نیازمند اطلاعات در مورد تاثیر اعمال خود عامل بر روی محیط می باشیم.

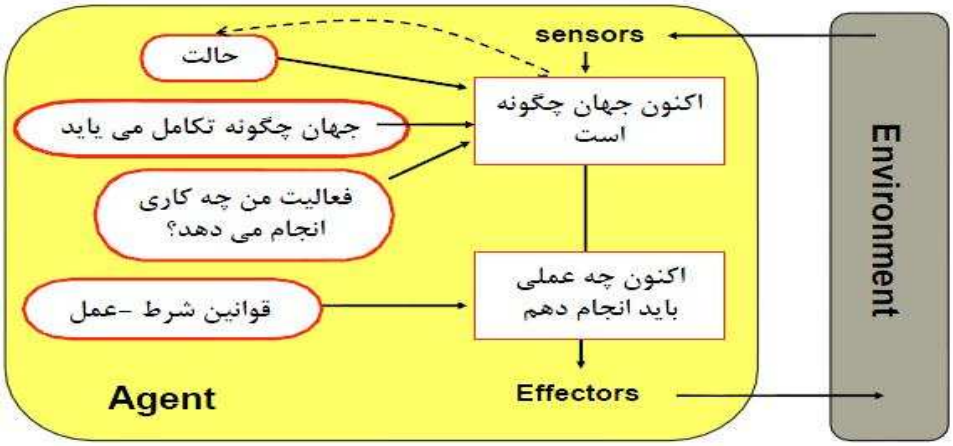
نکته: این دانش(دانش بخش دوم) که درباره چگونگی عملکرد دنیا می‌باشد مدل دنیا[1] نامیده می‌شود. و عاملی که از چنین مدلی استفاده می‌کند، **مبتنی بر مدل** نامیده می‌شود.

[1] Model of the world

ساختار عامل مبتنی بر مدل

برنامه و دیاگرام عامل مبتنی بر مدل در شکل زیر آمده است:

```
Function REFELEX_AGENT_WITH_SATAE
        ( percept ) returns action
Static:
        state , a description of the current world state
        rules , a set of condition_action rules
State ← UPDATE-STATE ( state,percept )
Rule ← RULE-MATCH ( state,rules )
action ← RULE-ACTION [rule]
State ← UPDATE-STATE ( state,action )
Return action
```



عامل‌های مبتنی بر هدف:

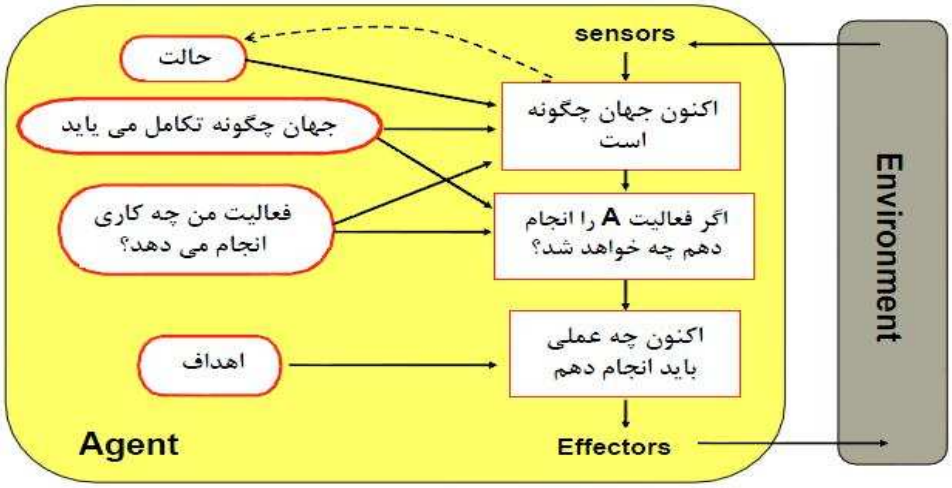
اطلاع از وضعیت فعلی محیط، همیشه برای تصمیم گیری در مورد اقدام بعدی کافی نیست. برای مثال در سر یک چهار راه، تاکسی می‌تواند به چپ، راست، یا مستقیم برود. تصمیم درست به مقصد مسافر بستگی دارد. بنابراین همانگونه که عامل نیازمند دانستن وضعیت جاری و قبلی است نیاز به یک هدف نیز خواهد داشت تا تصمیم گیری‌های وی بر مبنای آن هدف جهت گیرد. رسیدن به هدف گاهی اوقات ساده و گاهی اوقات پیچیده است. در مواقعی که هدف ساده است، ارضای هدف بلافاصله بعد از انجام یک عمل نتیجه خواهد شد ولی در مواقعی که هدف پیچیده باشد، عامل باید دنباله اقدامات طولانی را برای رسیدن به هدف طی کند. در مواقع پیچیده، جستجو[۱] و برنامه ریزی[۲] به یافتن دنباله‌ای از اعمال منجر خواهد شد. در مدل هدف گرا نمی‌توان از قوانین شرط – عمل گذشته تبعیت کرد بدین

---

[۱] Search

[۲] Planing

معنا که اگر فلان اتفاق افتاد آنگاه من چنان کنم. اگرچه بنظر می‌رسد عامل مبتنی بر هدف کمتر موثر باشد اما بسیار انعطاف پذیر است و می‌تواند خود را با شرایط محیطی وفق دهد و نیازی به دوباره نویسی قوانین شرط – عمل نیست. دیاگرام عامل مبتنی بر هدف در شکل زیر آمده است:



تفاوت عامل واکنشی و هدف گرا:

در طراحی عامل‌های واکنشی، طراح برای حالات متفاوت، عمل درست را پیش محاسبه می کند، ولی در عامل‌های هدف گرا عامل می‌تواند دانش خود را در مورد چگونگی واکنش، بهنگام سازی کنند.

بنابراین این دو نوع برنامه فوق در موارد زیر با هم تفاوت دارند:

- ❖ برای عامل واکنشی ساده مجبور به دوباره نویسی تعداد زیادی قوانین شرط – عمل خواهیم بود.
- ❖ عامل هدف‌گرا نسبت به رسیدن به مقاصد متفاوت، انعطاف پذیر است
- ❖ به سادگی با تعیین یک هدف تازه، می توان عامل هدف گرا را به رفتاری تازه برسانیم.

عامل‌های مبتنی بر سودمندی:

اهداف به تنهایی برای ایجاد رفتاری با کیفیت و سودمندی بالا کافی نخواهد بود. برای مثال در رسیدن تاکسی به مقصد ممکن است دنباله زیادی از اعمال وجود داشته باشد تا به مقصد برسیم ولی بعضی از این مسیرها سریعتر، امن تر و ارزانتر از بقیه هستند. اهداف فقط بین حالات راضی و ناراضی تفاوت قائل می‌شوند و درباره اینکه یک حالت چقدر عامل را راضی می‌کند سخن نمی‌گویند. برای مقایسه بین حالت‌ها که میزان راضی بودن را تعیین می‌کند از تابع سودمندی استفاده می‌کنیم.

تابع سودمندی[1]: یک حالت یا رشته‌ای از حالات را به یک عدد حقیقی که درجه رضایت نام دارد، نگاشت می‌کند.

تعریف کامل تابع سودمندی امکان تصمیم گیری منطقی برای دو حالتی که اهداف ناکافی هستند را دارد.

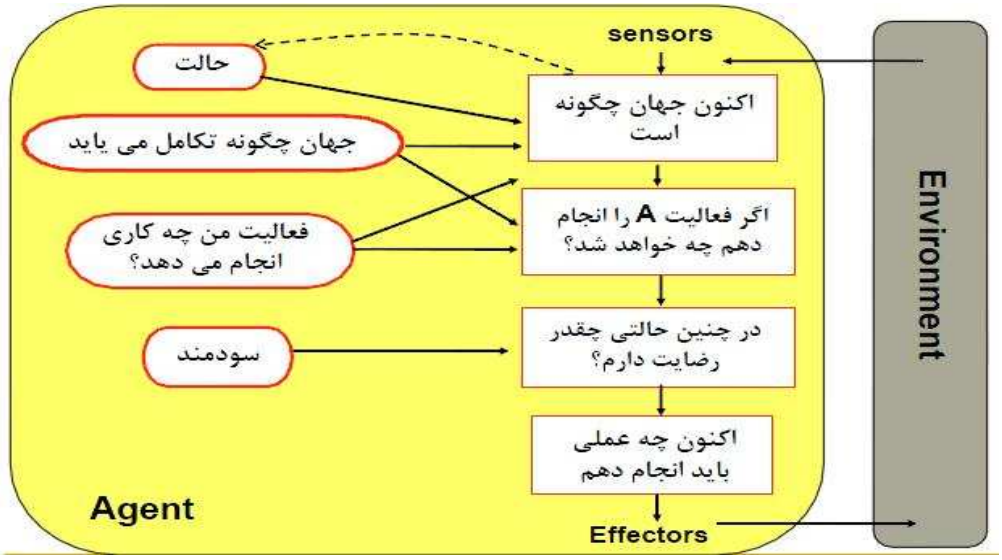
در مورد تابع سودمندی توجه به نکات زیر ضروری است:

- ❖ زمانیکه اهداف متناقض باشند تابع سودمندی می‌تواند موازنه خوبی برقرار کند. مثلاً سرعت و ایمنی با هم در تناقض هستند.

[1] utility Function

- ❖ زمانیکه چندین هدف وجود دارد عامل می‌تواند برای رسیدن به مقصد آنها را طی کند اما هیچ کدام از آنها با قطعیت قابل حصول نیست.
- ❖ سودمندی روشی را فراهم می‌کند که در آن موفقیت عامل بر اساس اهمیت اهداف وزن دهی می‌شود. مثلاً یک تابع سودمندی یک تابع خطی است که در آن متغیرها اهداف می‌باشند و مقدار ضریب آنها بر اساس اهمیت اهداف تعیین می‌شوند.

دیاگرام عامل مبتنی بر سودمندی در شکل زیر آمده است:



تفاوت معیار کارایی و تابع سودمندی:

i. معیار کارایی چگونگی و میزان موفقیت یک عامل را نشان می‌دهد، ولی تابع سودمندی میزان سودمندی یک وضعیت از دنیا را از دیدگاه عامل بر می‌گرداند.

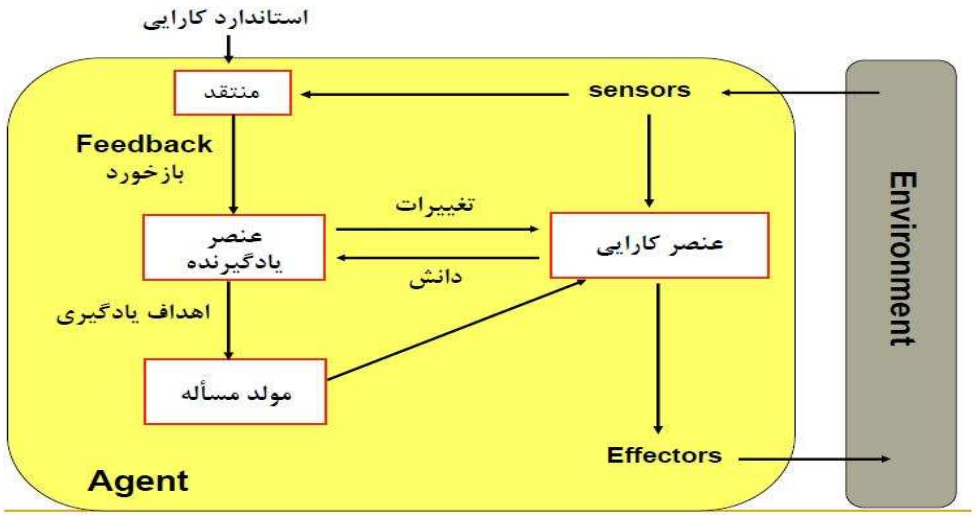
ii. معیار کارایی درجه موفقیت عامل است ولی تابع سودمندی درجه رضایت عامل است.

iii. معیار کارایی، پارامتر مقایسه بین دو عامل است ولی تابع سودمندی پارامتر مقایسه بین دو وضعیت در رسیدن به هدف است.

عامل‌های یادگیرنده :

عامل‌های یادگیرنده شامل مولفه‌های عنصر کارایی، عنصر یاد گیرنده، مولد مسئله و منتقد می‌باشد. عنصر کارایی، مسئول انتخاب فعالیت‌های خارجی است. عنصر یادگیرنده، مسئول ایجاد بهبودهاست. منتقد، مسئول تولید بازخورد با توجه به استاندارد کارایی برای عنصر یادگیرنده است. مولد مسئله، مسئول پیشنهاد فعالیت‌هایی است که منجر به تجربیات آموزنده جدید می‌شود.

نکته: طراحی عنصر یادگیری بسیار به طراحی عنصر کارایی وابسته است.



دیاگرام یک عمل یادگیرنده در شکل زیر آمده است:

مثال: این چهار مولفه را بر روی تاکسی خودکار بررسی می‌کنیم.

عنصر کارایی: عنصر کارایی شامل هر مجموعه از دانش‌ها و روال‌ها می‌باشد که تاکسی برای انتخاب اقدامات رانندگی‌اش در اختیار دارد. تاکسی با استفاده از این عنصر کارایی به جاده می‌رود و رانندگی می‌کند.

منتقد: منتقد دنیا را مشاهده می‌کند و اطلاعات را برای عنصر یادگیری می‌فرستد. برای مثال برای اینکه تاکسی بدون نگاه به عقب و راهنما زدن به سمت چپ می‌پیچد، آنگاه منتقد شاهد کلمات زشت و زننده‌ای ست که دیگر رانندگان آنرا به زبان می‌آورند.

عنصر یادگیری: بر اساس این تجربه، عنصر یادگیری می‌تواند قاعده‌ای را تنظیم کند که بیانگر این است، در هنگام سبقت، راهنما بزند. عنصر کارایی با نصب این قاعده جدید، تغییر خواهد کرد.

مولد مسئله: ممکن است زمینه‌های خاصی از رفتار را شناسایی کند که نیازمند اصلاح هستند، مانند آزمایش‌هایی از قبیل امتحان ترمزها در جاده‌هایی با سطوح خشک و خیس.

سوالات تستی آخر فصل

ترم اول ۸۷–۸۸

۱. کدامیک از موارد زیر در مورد ویژگی‌های محیط کار، درست نیست؟

الف. قطعی در مقابل اتفاقی ب. مرحله ای در مقابل ترتیبی

ج. وابسته در مقابل غیر وابسته د. تک عامله در مقابل چند عامله

۲. عامل یا Agent تلفیقی از ..... می‌باشد.

الف.هوش + تحرک ب. برنامه + معماری ج.حسگر + برنامه د. استنتاج + حسگر

۳. کدامیک از موارد زیر در مورد رویکرد جدول گرا در برنامه‌های عامل درست است؟

الف. هیچ عامل فیزیکی در این دنیا فضای ذخیره سازی این جدول را ندارد.

ب. طراح دقت کافی برای ساختن جدول را داراست.

ج. عامل می‌تواند تمام داده‌های صحیح را از طریق جدولش یاد بگیرد.

د. رویکرد جدول گرا در ساخت عامل، موجب موفقیت برنامه عامل خواهد شد.

۴. کدام مورد از انواع عامل‌ها نمی‌باشد؟

الف. عامل واکنشی ساده ب. عامل مبتنی برهدف ج. عامل مبتنی بر هوش د. عامل مبتنی برمدل

۵. برای طراحی عامل راننده تاکسی کدامیک از برنامه‌های عامل زیر مناسب تر خواهد بود؟

الف. عامل واکنشی ساده ب. عامل همراه با تثبیت تغییرات دنیا

ج.عامل هدف گرا د.عامل مبتنی بر سودمندی

۶. نحوه عمل عامل‌های واکنشی ساده در محیط‌های پیوسته چگونه است؟

الف. در بسیاری ازموارد تصمیم گیری کامل و دقیق نیست، زیرا عامل ورودی، محیط پیوسته را به کمیتی گسسته تبدیل نموده و بر اساس آن عمل خواهدکرد.

ب.کامل است، زیرا می‌توان برای هر ورودی، رفتاری را در عامل تعیین کرد.

ج. اگر محیط قابل دسترس باشد، عملکرد آن کامل خواهد بود.

د. عمکرد عامل بستگی به پیوسته بودن محیط ندارد.

۷. کدامیک از گزینه‌های زیر تفاوت معیار کارایی و تابع سودمندی عامل را بیان نمی‌کند؟

الف. معیار کارایی برای تصحیح عملکرد عامل استفاده می‌شود و تابع سودمندی در طراحی ساختار عامل دخالت دارد.

ب. معیار کارایی چگونگی و میزان موفقیت یک عامل را نشان می‌دهد و تابع سودمندی میزان سودمندی یک وضعیت را در دنیا و از دیدگاه عامل بر می‌گرداند.

ج. معیار کارایی، پارامتر مقایسه بین دو عامل است و تابع سودمندی پارامتر مقایسه دو وضعیت در رسیدن به هدف.

د. معیار کارایی، موفقیت عامل است و تابع سودمندی درجه رضایت عامل.

ترم دوم ۸۷ – ۸۸

۸. مقیاس کارایی از کدام دیدگاه قابل قبول تر است؟

الف. خودعامل ب. حریف ج. طراح و سازنده عامل د. محیط

۹. هرچه عامل از دانش داخلی‌اش بیشتر استفاده کند و به ادراکش کمتر توجه داشته باشد، خودمختاری ..... دارد و سیستمی که کاملا خود مختار باشد..... است.

الف. بیشتری – غیرهوشمندتر ب. بیشتری – غیر خودمختارتر

ج. کمتری – هوشمندتر د. کمتری – غیر هوشمندتر

۱۰. قوانین Condition-Action مربوط به کدامیک از عامل‌ها می‌باشد؟

الف. عامل‌های واکنشی ب. عامل‌های مبتنی بر سودمندی

ج. عامل‌های مبتنی بر هدف د. عامل‌های حل مسئله

۱۱. در این نوع محیط تجربه عامل به بخش‌های مجزا تقسیم می‌شود که تصمیم گیری در هر مرحله ربطی به حالت قبل ندارد.

الف. قطعی ب. مرحله‌ای ج. ایستا د. گسسته

۱۲. کدامیک از تعاریف زیر محیط نیمه پویا را توصیف می‌کند؟

الف. زمانی که محیط پویاست، اما هر چه که زمان بگذرد امتیازی از ما کم می‌شود.

ب. زمانی که محیط گسسته است و گذشت زمان را هم داشته باشیم.

ج. زمانی که محیط ایستاست و هر چه که زمان بگذرد امتیازی از ما کم می‌شود.

د. زمانی که محیط ایستاست و زمان تغییر نمی‌کند.

ترم تابستان ۸۸

۱۳. کدامیک از گزینه‌های زیر صحیح نیست ؟

الف. عقلانیت کارائی مورد انتظار را بیشینه می‌کند.

ب. کمال، کارایی واقعی را بیشینه می‌کند.

ج. عقلانیت مستلزم همه چیزدانی است.

د. انتخاب عقلانی تنها به رشته ادراکات تا آن لحظه بستگی دارد.

۱۴. کدام گزینه در مورد عاملی که فقط بر اساس دانش درونی عمل می‌کند، صحیح نیست ؟

الف. ممکن است در یک محیط قطعی ساده موفق عمل کند.

ب. کاملاً خودمختار است.

ج. بسیار آسیب پذیر است.

د. امکان یادگیری ندارد .

۱۵. کدامیک از محیط‌های کاری عامل‌های زیر، یک محیط ایستا می‌باشد؟

الف. تشخیص پزشکی ب. جدول کلمات متقاطع ج. شطرنج زمان دار د. معلم انگلیسی محاوره‌ای

۱۶. در محیط‌های نیمه رؤیت پذیر کدام نوع عامل، اغلب دچار حلقه‌های بی نهایت می‌شود؟

الف. واکنشی ساده ب. واکنشی مبتنی بر مدل ج. مبتنی بر هدف د. مبتنی برسودمندی

۱۷. کدامیک از اجزاء مفهومی یک عامل یادگیرنده، نمی‌باشد ؟

الف. عنصر یادگیری ب. عنصر دانش ج. منتقد د. مولد مسئله

ترم اول ۸۸–۸۹

۱۸. عملکرد رضایت بخش عامل (کارگزار) از کدام دیدگاه قابل قبول تر است؟

الف خودعامل ب. طراح و سازنده عامل ج. کاربر عامل د. تماشاگر عامل

۱۹. کدامیک جزو ویژگی‌های محیط کار شطرنج زمان دار است؟

الف. ایستا ب. پویا ج. نیمه پویا د. تک عاملی

ترم دوم ۸۸–۸۹

۲۰. محیط کار سیستم تشخیص پزشکی چگونه است؟

الف. کاملاً رؤیت پذیر، قطعی، نیمه پویا، گسسته

ب. کاملاً رؤیت پذیر، راهبردی، پویا، گسسته

ج.بعضاً رویت پذیر، اتفاقی، ایستا، گسسته

د. بعضاً رویت پذیر، اتفاقی، پویا، پیوسته

ترم تابستان ۸۹

۲۱. کدامیک جزء ویژگی‌های محیط عامل شطرنج زمان دار است؟

الف. قطعی ب. اتفاقی ج. راهبردی د. پیوسته

۲۲. پیچیده ترین محیط کار کدام است؟

الف. نیمه رؤیت پذیر، اتفاقی، ترتیبی، پویا، پیوسته و چند عاملی

ب. نیمه رؤیت پذیر، راهبردی، ترتیبی، پویا ،پیوسته و چند عاملی

ج. نیمه رؤیت پذیر، اتفاقی، ترتیبی، نیمه پویا ،پیوسته و چند عاملی

د. نیمه رؤیت پذیر، راهبردی، مرحله ای، پویا، پیوسته و چند عاملی

۲۳. کدامیک جزءمشکلات عامل‌های مبتنی بر جدول (table driven) نمی‌باشد؟

الف. حافظه‌ی بیش از حد مورد نیاز جدول

ب. زمان بیش از حد موردنیاز جهت پر کردن جدول توسط طراح

ج. پیچیدگی زیاد پیاده سازی

د.عدم خودمختاری

ترم اول ۸۹ – ۹۰

۲۴. کدامیک جزء ویژگی‌های محیط کاری تخته نرد نمی‌باشد ؟

الف. قطعی ب. کاملاً رویت پذیر ج. ایستا د. گسسته

۲۵. کدامیک از عامل‌های زیر درجه هوشمندی ضعیفتری دارند؟

الف. عامل‌های مبتنی بر جدول ب. عامل‌های واکنشی ساده

ج. عامل‌های مبتنی بر هدف د. عامل‌های مبتنی بر سودمندی

۲۶. کدام عامل تنها در محیط کاملاً رویت پذیر امکان تصمیم گیری صحیح دارد ؟

الف. واکنشی ساده ب. واکنشی مبتنی بر جدول

ج. مبتنی بر هدف، مبتنی بر مدل د. مبتنی بر سودمندی، مبتنی بر مدل

ترم دوم ۸۹–۹۰

۲۷. کدام مورد جزء مراحل محاسبه تابع عامل در عامل‌های موفق نمی‌باشد؟

الف. در حین انجام اقدام ب. هنگام طراحی عامل (توسط طراح)

ج. وقتی از تجربیاتش چیزی یاد می‌گیرد د. وقتی به اقدام بعدی فکر می‌کند

۲۸. از ۴ نوع عامل زیر کدام یک مبتنی بر مدل هستند ؟

۱. واکنشی ساده

۲. واکنشی مبتنی بر مدل

۳. مبتنی بر هدف

۴. یاد گیرنده

الف. ۱و۲ ب. ۲و۳ ج. فقط ۲ د. ۳و۴

تابستان ۹۰

۲۹. یک عامل هوشمند برای سیستم راننده تاکسی طراحی شده است. کیلومترشمار جزء کدامیک از ویژگی‌های عامل است؟

الف. مقیاس کارایی ب. عملیات ج. محیط د. حسگر

۳۰. کدام گزینه در رابطه با خواص محیط اطراف عامل نادرست است؟

الف. اگر وضعیت بعدی محیط تنها با توجه به وضعیت کنونی و نیز به کمک اعمالی که عامل انجام می‌دهد، تعیین شود، محیط قطعی است.

ب. اگر ابزار حس کننده عامل امکان دسترسی به وضعیت کامل محیط را بدهد، محیط قابل دسترسی است.

ج. اگر در حین سنجیدن عامل، محیط تغییر کند، محیط پویاست.

د. اگر تعداد محدود از ادراک و اعمال به وضوح تعریف شده باشد، محیط ترتیبی است.

۳۱. تابعی که نگاشت عامل از ادراک به اعمال را پیاده سازی می‌کند و وظیفه اصلی هوش مصنوعی است، چه نام دارد؟

الف. معماری عامل ب. اهداف عامل ج. برنامه عامل د. عملیات عامل

۳۲. کدام عامل زیر برای پیگیری قسمت‌هایی از دنیا که در ادراک فعلی قابل مشاهده نیستند، حالت داخلی را نگهداری می‌کند؟

(حداقل جزء لازم در عامل ذکر شود.)

الف. مبتنی بر مدل ب. مبتنی بر هدف

ج. مبتنی بر سودمندی د. ساده

۳۳. گزینه نادرست را انتخاب نمایید.

الف. عقلانیت هر عامل به دانش قبلی، رشته ادراک، اقدامات و مقیاس کارایی بستگی دارد.

ب. عقلانیت سعی در بیشینه کردن کارایی دارد. اما در عمل، ممکن است کارایی واقعی بیشینه نگردد.

ج. عقلانیت باعث خودمختاری عامل می‌شود.

د. عقلانیت با همه چیز دانی تفاوت دارد.

پاسخ نامه تستی

سوال	گزینه صحیح	سوال	گزینه صحیح
۱	ج	۱۶	الف
۲	ب	۱۷	ب
۳	الف	۱۸	ب
۴	ج	۱۹	ج
۵	د	۲۰	د
۶	الف	۲۱	ج
۷	الف	۲۲	الف
۸	ج	۲۳	ج
۹	د	۲۴	الف
۱۰	الف	۲۵	الف
۱۱	ب	۲۶	الف
۱۲	ج	۲۷	الف
۱۳	ج	۲۸	ب
۱۴	ب	۲۹	د
۱۵	ب	۳۰	د
		۳۱	ج
		۳۲	الف
		۳۳	ج

سوالات تشریحی آخر فصل

ترم اول ۸۷–۸۸

عامل‌های واکنشی مبتنی بر مدل را تعریف کنید.

ترم دوم ۸۷–۸۸

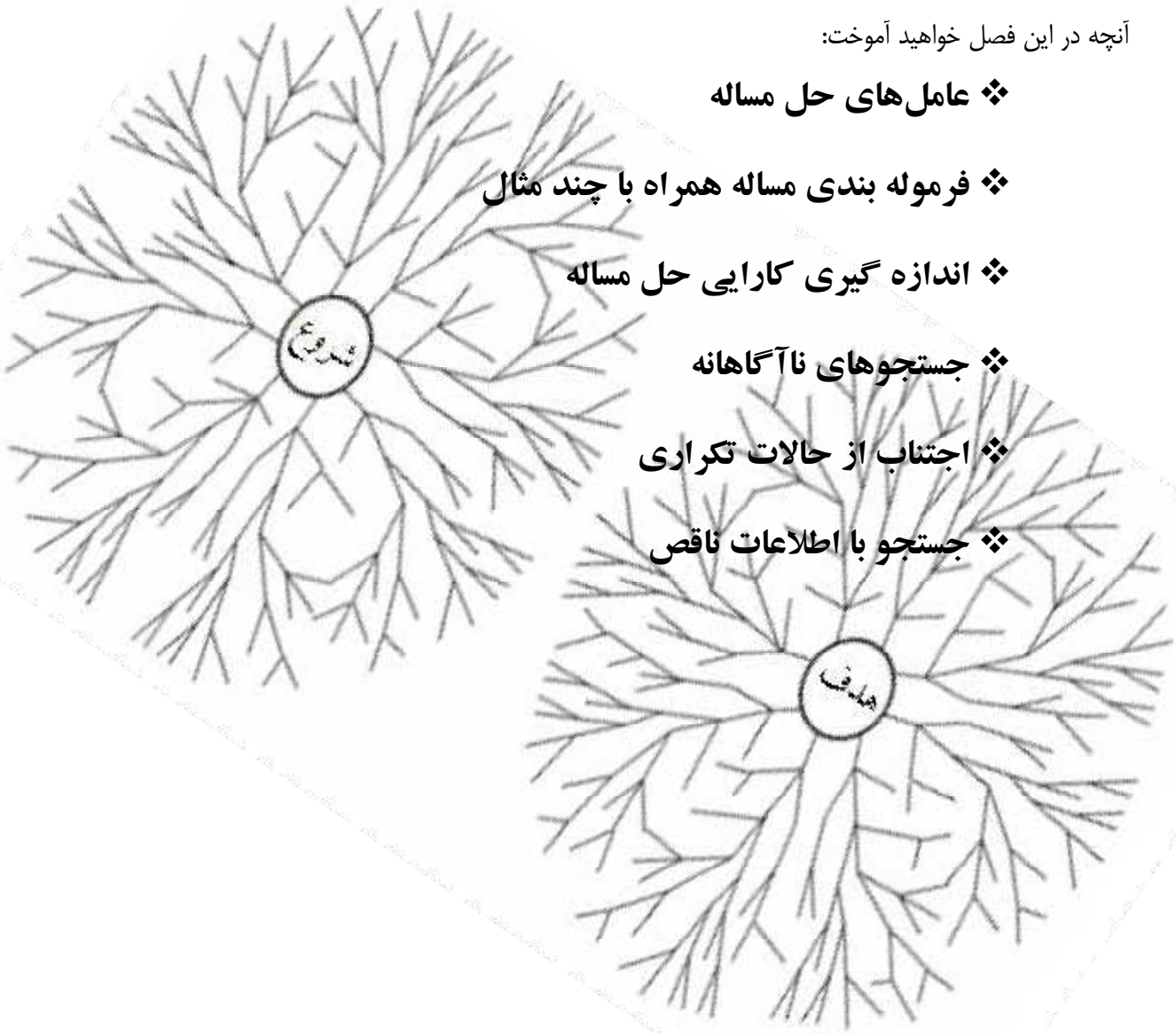
جدول زیررا کامل کنید.

محیط	مقیاس کارایی	اقدام گرها	حسگرها	نوع عامل
				سیستم تشخیص پزشکی
				ربات جابجا کننده اشیا

ترم دوم ۸۸–۸۹

عامل‌های مبتنی بر سودمندی را بارسم شکل توضیح دهید. (۱/۵ نمره)

فصل سوم: حل مسئله از طریق جستجو



آنچه در این فصل خواهید آموخت:

- **عامل‌های حل مساله**
- **فرموله بندی مساله همراه با چند مثال**
- **اندازه گیری کارایی حل مساله**
- **جستجوهای ناآگاهانه**
- **اجتناب از حالات تکراری**
- **جستجو با اطلاعات ناقص**

عامل‌های حل مسئله:

نوعی از عامل‌های هدف‌گرا هستند، که توسط یافتن ترتیب عملیات تصمیم می‌گیرند، چه نوع عملی را انجام دهند، تا به حالت مطلوب سوق پیدا کنند. مراحل زیر باید توسط یک عامل حل مسئله انجام شود:

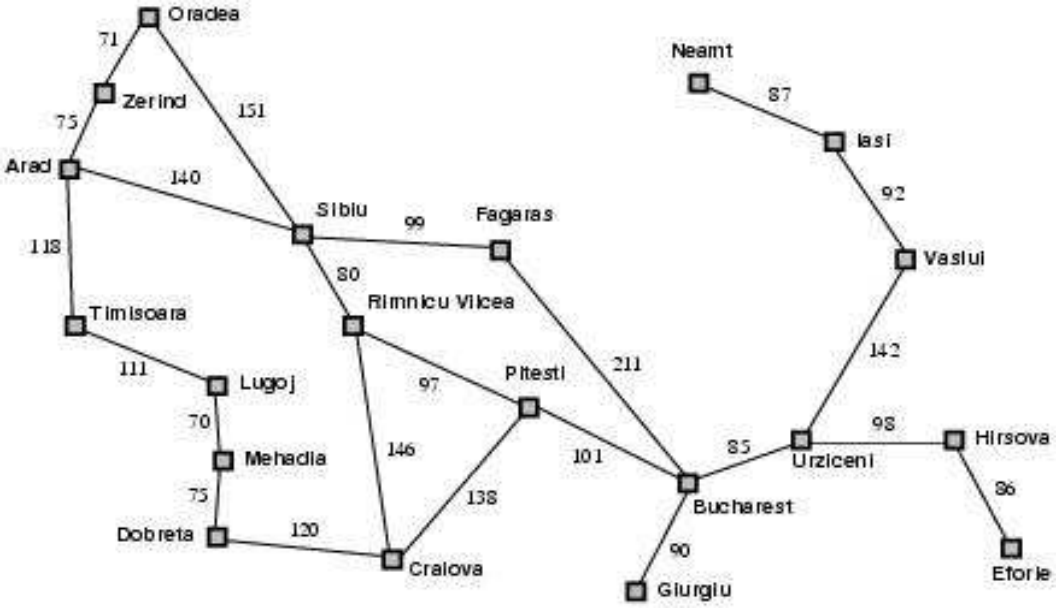
فرموله کردن (تدوین) هدف: وضعیت‌های مطلوب نهایی کدامند.

فرموله کردن مسئله: چه اقدامات و وضعیت‌هایی برای رسیدن به هدف موجود است.

جستجو: در این مرحله عامل تصمیم می‌گیرد که چه رشته اعمالی می‌تواند وی را از حالت شروع به حالت هدف برساند. مسلماً این رشته از اعمال از بین یک مجموعه اعمال ممکن انتخاب می‌شود، خروجی این مرحله یک راه-حل[1]است.

اجرا: راه‌حلی که از مرحله قبل به دست آمده اجرا می‌شود.

مثال: نقشه رومانی:



صورت مسئله: رفتن از آراد به بخارست

فرموله کردن هدف: رسیدن به بخارست.

فرموله کردن مسئله:

- **وضعیت‌ها:** شهرهای مختلف
- **فعالیت‌ها:** حرکت بین شهرها

جستجو: دنباله‌ای از شهرها مثل آراد، سیبیو، فاگارس، بخارست. (جستجو با توجه به کم‌هزینه‌ترین مسیر)

تعریف مسئله: مجموعه‌ای از اطلاعات است که عامل از آنها برای این‌که چه عملی را انجام دهد، استفاده می‌کند. یک مسئله با موارد زیر تعریف می‌شود:

---

[1] solution

حالت اولیه: حالتی که عامل از آن شروع می‌کند. در مثال رومانی شهر آراد n(Arad)

تابع جانشین: توصیفی از حالت‌های ممکن که برای عامل مهیاست.

S (Arad) = {zerind – Sibiu – Timisoara}

فضای حالت: مجموعه‌ای از حالت‌ها که از حالت اولیه می‌توان به آنها رسید. در مثال رومانی: کلیه شهرهایی که با شروع از آراد می‌توان به آنها رسید.

نکته: فضای حالت= حالت اولیه + تابع جانشین.

آزمون هدف: تعیین می‌کند که آیا حالت خاصی حالت هدف است یا خیر. دو نوع هدف داریم:

- **هدف صریح:** در مثال رومانی رسیدن به بخارست.
- **هدف انتزاعی (ضمنی):** در مثال شطرنج، رسیدن به حالت کیش و مات.

مسیر: دنباله‌ای از حالت‌ها که دنباله‌ای از فعالیت‌ها را به هم متصل می‌کند. در مثال رومانی: Arad, Sibiu, Fagaras یک مسیر است.

هزینه مسیر: برای هر مسیر یک هزینه عددی در نظر می‌گیرد. در مثال رومانی: طول مسیر بین شهرها بر حسب کیلومتر.

راه حل مسئله: مسیری از حالت اولیه به حالت هدف است. راه‌حل بهینه کمترین هزینه را دارد.

نکته: فرآیند حذف جزییات از یک توصیف، تجرید[1] یا انتزاع نام دارد. به عنوان نمونه در مثال رومانی یک سری از توصیفات در دنیای واقعی وجود دارند که به مسئله پیدا کردن یک مسیر به بخارست ربطی ندارد. مثلاً همراهان مسافر، منظره بیرون از پنجره، فاصله ایستگاه تا پمپ‌بنزین و ...

نکته: در تجرید توجه به نکات زیر ضروری است:

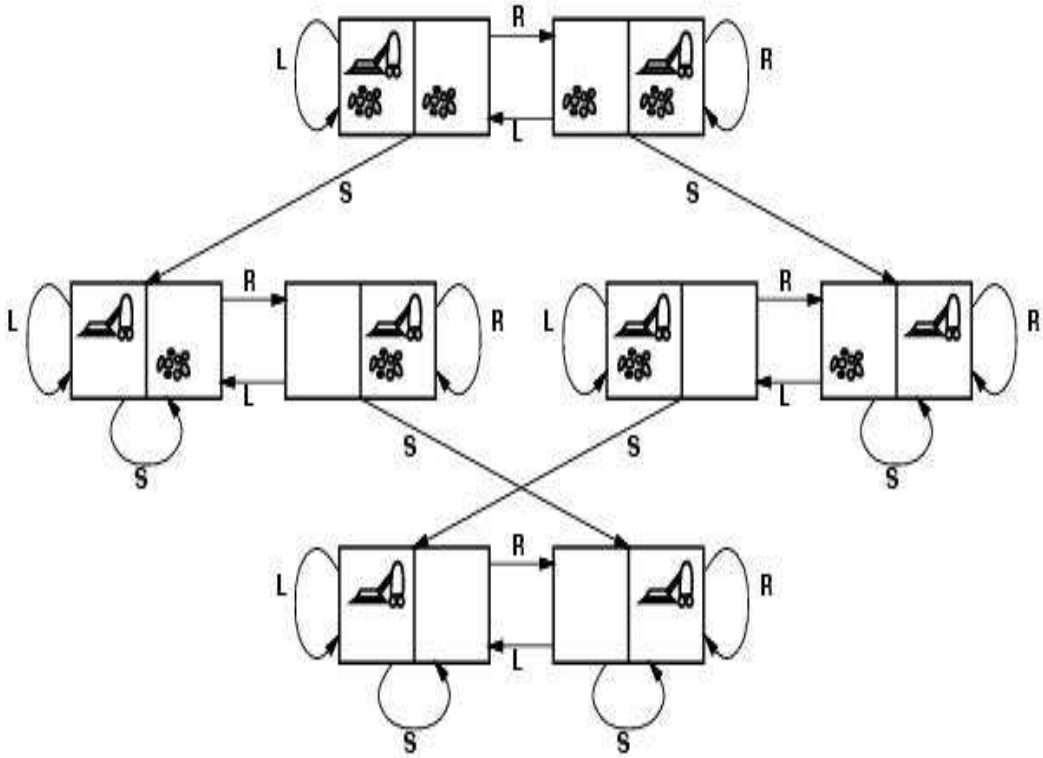
- یک تجرید معتبر است اگر بتوانیم هر راه‌حل خلاصه را به یک راه‌حل درونیابی با جزئیات بیشتر بسط دهیم.
- یک تجرید مفید است اگر هریک از اعمال نسبت به مسئله اصلی آسان‌تر انجام شوند.

---

[1] Abstraction

فرموله کردن چند مسئله نمونه:

مسئله دنیای جاروبرقی:



حالتها: دو مکان که هر یک ممکن است کثیف یا تمیز باشند. لذا $2 * 2^2 = 8$ حالت در این جهان وجود دارد.

حالت اولیه: هر حالتی می‌تواند به عنوان حالت اولیه طراحی شود.

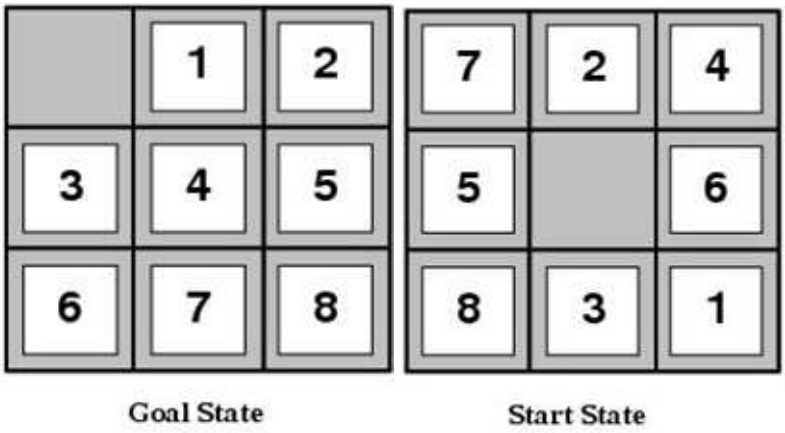
تابع جانشین: حالتهای معتبر از سه عملیات: راست، چپ، مکش.

آزمون هدف: تمیزی تمام مربع‌ها.

هزینه مسیر: تعداد مراحل در مسیر.

نکته: اگر محیطی n محل داشته باشد $n * 2^n$ حالت خواهد داشت.

مسئله معمای ۸:



حالتها: مکان هر هشت خانه شماره دار و خانه خالی در یکی از ۹ خانه

حالت اولیه: هر حالتی را میتوان به عنوان حالت اولیه در نظر گرفت.

تابع جانشین: حالت‌های معتبر از چهار عمل، انتقال خانه خالی به چپ، راست، بالا یا پایین

آزمون هدف: بررسی می‌کند که حالتی که اعداد به ترتیب چیده شده‌اند (طبق شکل روبرو) رخ داده یا نه

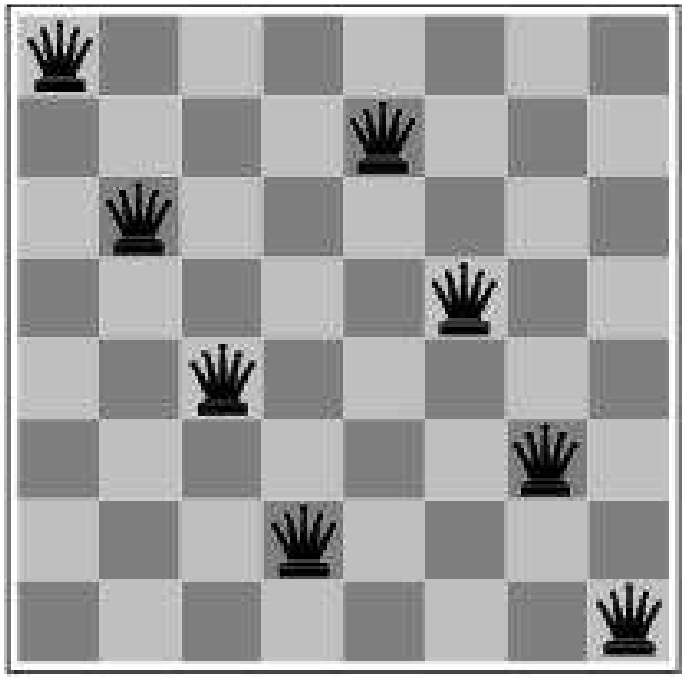
هزینه مسیر: برابر با تعداد مراحل در مسیر.

مسئله ۸ وزیر:

هدف در این مسئله قرار دادن ۸ وزیر روی صفحه شطرنج است به نحوی که هیچ وزیری به دیگری حمله نکند.

برای این مسئله دو نوع فرموله‌سازی وجود دارد:

- **فرموله سازی افزایشی**[1]: که با صفحه خالی شروع شده و در هر عمل یک وزیر به صفحه اضافه می‌شود.
- **فرموله سازی حالت کامل**[2]: با همه ۸ وزیر روی صفحه شروع می‌کند و در هر عمل یک وزیر به اطراف حرکت داده می‌شود.



فرمول بندی افزایشی

حالت‌ها: هر ترتیبی از ۰ تا ۸ وزیر در صفحه، یک حالت است

حالت اولیه: هیچ وزیری در صفحه نیست

تابع جانشین: وزیری را به خانه خالی اضافه می‌کند

آزمون هدف: ۸ وزیر در صفحه وجود دارند و هیچ کدام به یکدیگر گارد نمی‌گیرند

در این فرمول بندی باید $3 * 10^{14}$ دنباله ممکن بررسی می‌شود

---

[1] incremental formulation

[2] complete state formulation

فرمول بندی حالت کامل

حالت‌ها: چیدمان n وزیر ($0 \le n \le 8$)، بطوری که در هر ستون از n ستون سمت چپ، یک وزیر قرار گیرد و هیچ دو وزیری بهم گارد نگیرند

حالت اولیه: با ۸ وزیر در صفحه شروع می‌شود

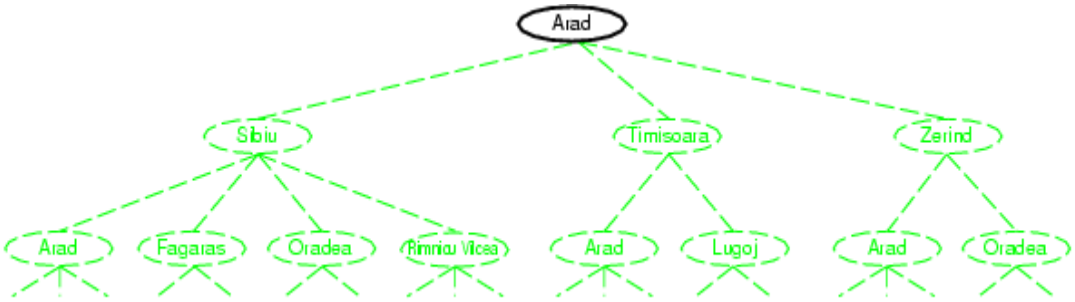
تابع جانشین: وزیری را در سمت چپ‌ترین ستون طوری جابجا می‌کند، بطوری که هیچ وزیری آن را گارد ندهد.

آزمون هدف: ۸ وزیر در صفحه وجود دارند و هیچ کدام به یکدیگر گارد نمی‌گیرند

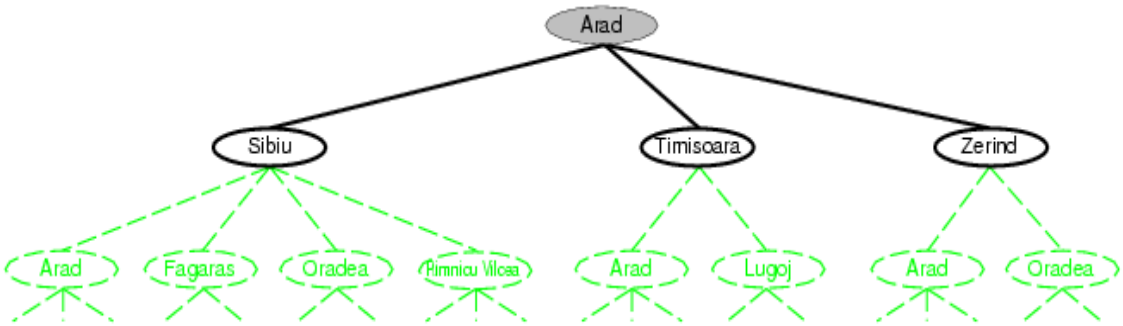
در فرموله سازی حالت کامل، فضای حالت از $3 * 10^{14}$ به 2057 کاهش می‌یابد و پیدا کردن راه‌حل در فضای حالت کامل راحت‌تر از افزایشی است. برای فرموله کردن مسئله ۱۰۰ وزیر در افزایشی 10^{400} حالت و در روش فرموله سازی حالت کامل 10^{52} برای پیدا کردن راه حل باید بررسی شود.

گسترش[1] :

به معنی بکارگیری تابع مابعد برای یک حالت و تولید یک مجموعه جدید ازحالات می‌باشد. مجموعه نودهایی که تولید شده‌اند اما هنوز گسترش نیافته‌اند مجموعه حاشیه[2] نامیده می‌شود و هر عنصر از مجموعه حاشیه یک نود برگ است یعنی نودی که هنوز هیچ مابعدی در درخت جستجو ندارد.



حالت اولیه

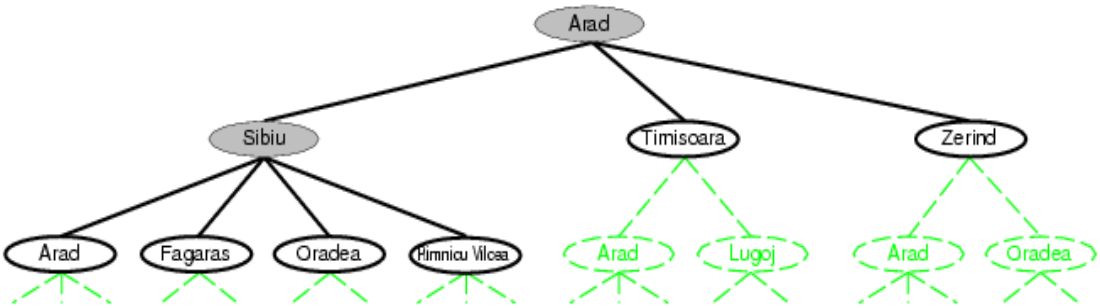


پس از گسترش آراد

---

[1] expand

[2] fringe



پس از گسترش سیبیو

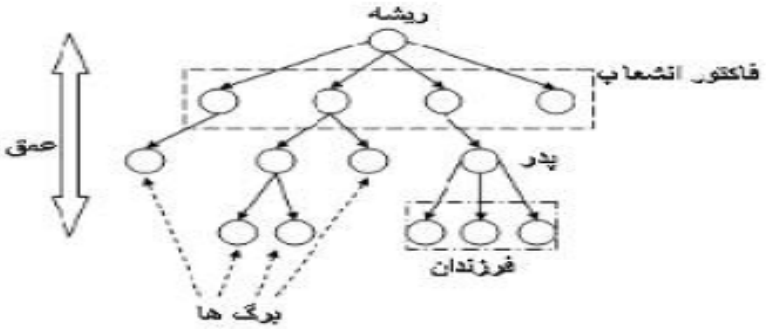
اندازه گیری کارایی حل مسئله :

خروجی الگوریتم جستجو راه حل[۱] یا شکست[۲] است. کارایی الگوریتم جستجو با چهار معیار زیر ارزیابی می شود:

- **کامل بودن[۳]:** آیا الگوریتم تضمین می‌کند که در صورت وجود راه حل، راه حل را پیدا کند؟
- **بهینگی[۴]:** آیا الگوریتم تضمین می‌کند که از بین چندین راه حل، راه حل بهینه یا کم هزینه ترین را پیدا کند؟
- **پیچیدگی زمانی[۵]:** چه مدت زمانی طول می‌کشد تا الگوریتم جستجو، راه حل راپیدا کند؟
- **پیچیدگی فضایی[۶]:**الگوریتم جستجو برای پیدا کردن راه حل، چقدر حافظه نیاز دارد؟

نکته: پیچیدگی زمانی و مکانی بر اساس سه مفهوم زیر بیان می‌شود:

- **b:** حداکثر فاکتور انشعاب درخت جستجو.
- **d:** عمق کم هزینه ترین راه حل.
- **m:** حداکثر عمق فضای حالت که می‌تواند بی نهایت نیز باشد.



---

[۱] Solution

[۲] Failure

[۳] Completeness

[۴] Optimally

[۵] Time complexity

[۶] Space complexity

نکته: معمولاً پیچیدگی زمانی بر اساس تعداد نودهای تولید شده در حین جستجو و پیچیدگی مکانی بر اساس ماکزیمم تعداد نودهای ذخیره شده در حافظه اندازه گیری می‌شود. برای ارزیابی کارایی یک الگوریتم جستجو می‌توان فقط هزینه جستجو که مجموع پیچیدگی زمانی و مکانی یا هزینه کل را در نظر گرفت، که هزینه کل، مجموع هزینه جستجو و هزینه مسیر راه حل می‌باشد.

نکته: هزینه کل = هزینه جستجو + هزینه مسیر

نکته: پس از تعریف یک مسئله و فرموله سازی آن باید راه حلی برای آن تشخیص داده شود که این راه حل بر اساس جستجو در فضای حالت پیدا خواهد شد.

انواع استراتژی‌های جستجو:

از نظر اینکه آیا از تعداد مراحل یا هزینه مسیر از حالت جاری به حالت هدف اطلاعاتی دارند یا نه، به دو دسته تقسیم می شوند:

I. جستجوهای آگاهانه[۱]:

این دسته از الگوریتم‌ها علاوه بر اطلاعات مسئله اطلاعات اضافی برای رسیدن به هدف در اختیار دارند که می توانند امید بخش تر بودن یک گره نسبت به گره دیگر را با توجه به اطلاعات اضافی تشخیص دهند که به این دسته جستجوها، جستجوهای اکتشافی[۲] نیز گفته می‌شود.

II. جستجوهای ناآگاهانه[۳]:

این دسته از الگوریتم‌های جستجو، هیچ اطلاعاتی غیر از تعریف مسئله در اختیار ندارند و فقط می‌توانند جانشین‌هایی را تولید و هدف را از غیر هدف تشخیص دهند که به آن جستجوهای کور کورانه[۴] نیز گفته می‌شود.

---

[۱] Informed search

[۲] heurestic

[۳] Uninformed search

[۴] blind

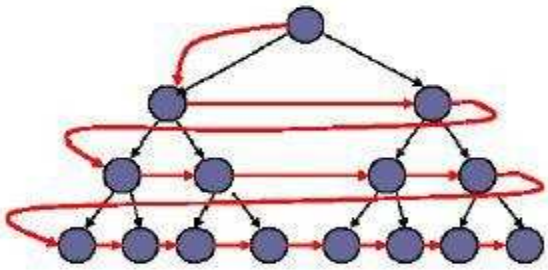
انواع جستجوهای ناآگاهانه عبارتند از:

- جستجوی اول–سطح[۱]
- جستجوی هزینه یکنواخت[۲]
- جستجوی اول–عمق[۳]
- جستجوی عمقی محدود شده[۴]
- جستجوی عمقی تکرار شونده[۵]
- جستجوی دو طرفه[۶]

جستجوی اول–سطح:

در جستجوی اول سطح (bfs)، ابتدا نود ریشه گسترش یافته، سپس همه مابعدهای نود ریشه، و بعد از آن مابعدهای آن و ... گسترش می‌یابد. بطور کلی نودهای یک سطح از درخت جستجو قبل از نودهای سطح بعدی گسترش می‌یابند.

نکته: فراخوانی (Tree-search(problem, FIFOqueue منجر به اجرای جستجوی سطح اول سطح می-شود.



جستجوی سطحی

صف FIFO، مابعدهای تولید شده جدید را در انتهای صف قرار می‌دهد. بدین ترتیب نودهای کم عمق تر قبل از نود-های عمیق تر گسترش می‌یابند. اگر کم عمق ترین نود هدف در عمق متناهی d و فاکتور انشعاب b متناهی باشد جستجوی BFS حتما هدف را بعد از گسترش همه نودهای کم عمق تر از d پیدا می‌کند. لذا این روش کامل است اگر تابع هزینه مسیر یک تابع غیر کاهشی از عمق نود باشد این الگوریتم بهینه نیز می‌باشد. به عنوان مثال اگر همه اعمال هزینه یکسان داشته باشند این جستجو هدف را پیدا می‌کند. لازم بذکر است این الگوریتم با وجود کامل بودن همیشه کارا نیست. یعنی پیچیدگی مکانی و زمانی بسیار بالایی دارد.

---

[۱] Breadth-First-Search

[۲] Uniform-Cost-Search

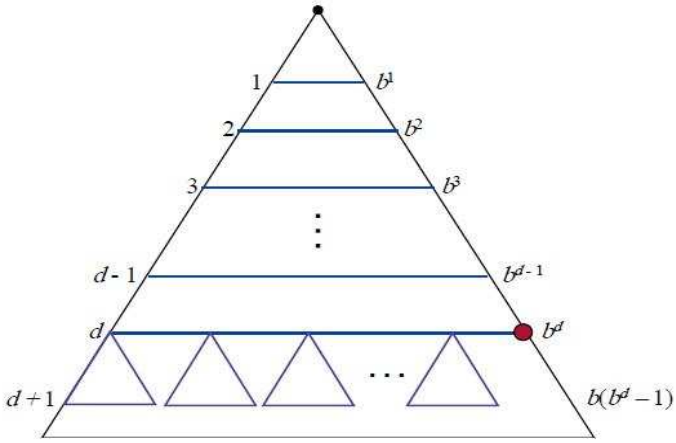
[۳] Depth-First-search

[۴] Limited-Depth-Search

[۵] Iteration-Depth-Search

[۶] Bidirectional search

پیچیدگی زمانی و حافظه جستجوی سطحی



$$1 + b + b^2 + b^3 + \cdots + b^d + b(b^d - 1) = O(b^{d+1})$$

نکته: هر نودی که تولید می‌شود باید در حافظه بماند، زیرا هر نود قسمتی از مجموعه حاشیه است یا یک جد برای نودهای برگ است. بنابراین پیچیدگی مکانی نیز مانند پیچیدگی زمانی است. ویژگی‌های الگوریتم جستجوی اول-سطح را در موارد زیر می‌توان خلاصه کرد:

کامل بودن: بله

بهینگی: بله (مشروط)، در صورتی بهینه است که هزینه مسیر تابعی غیر نزولی از عمق گره باشد (مثل وقتی که اعمال هزینه یکسانی دارند)

پیچیدگی زمانی : $O(b^{d+1})$

پیچیدگی فضا: $O(b^{d+1})$

نکته: با توجه به اطلاعات موجود در جدول زیر، نیازمندی‌های حافظه جستجوی اول-سطح بسیار بغرنج تر از پیچیدگی زمانی است. ۳۱ ساعت، زمان زیادی برای حل یک مسئله خیلی مهم با عمق هدف ۸ نیست اما تعداد کمی از کامپیو ترها دارای حافظه اصلی یک ترا بایتی هستند. علاوه بر آن نیازمندی‌های زمان هم یک فاکتور مهم می‌باشد. به عنوان مثال، اگر راه حل در عمق ۱۲ قرار داشته باشد آنگاه طبق فرضیات جدول زیر، جستجو ۳۵ سال طول خواهد کشید.

زمان و فضای لازم در جستجوی سطحی

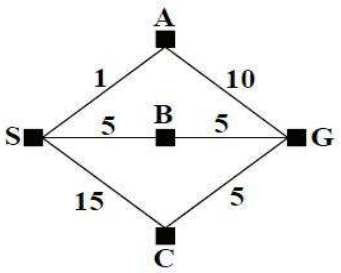
عمق	تعداد گره‌ها	زمان	حافظه
۲	۱۱۰۰	۱۱/۰ ثانیه	۱ مگابایت
۴	۱۱۱۱۰۰	۱۱ ثانیه	۱۰۶ مگابایت
۶	$۱۰^{۷}$	۱۹ دقیقه	۱۰ گیگابایت
۸	$۱۰^{۹}$	۳۱ ساعت	۱ ترابایت
۱۰	$۱۰^{۱۱}$	۱۲۹ روز	۱۰۱ ترابایت
۱۲	$۱۰^{۱۳}$	۳۵ سال	۱۰ پتابایت
۱۴	$۱۰^{۱۵}$	۳۵۲۳ سال	۱ هگزابایت

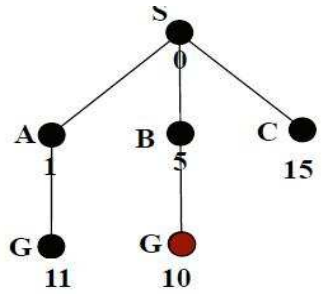
b = ۱۰، ۱۰۰۰۰ گره در هر ثانیه، هر گره ۱۰۰۰ بایت

جستجوی هزینه یکنواخت:

جستجوی اول-سطح، فقط وقتی هزینه همه اعمال یکسان است، بهینه می‌باشد، زیرا این روش جستجو کم عمق ترین نود را گسترش می‌دهد. الگوریتم جستجو با هزینه یکنواخت با هر تابع، هزینه مسیر بهینه است این الگوریتم جستجو نودی را انتخاب می‌کند که کمترین g را دارد (g(n) هزینه رسیدن از حالت شروع به نود n)یعنی نود با کمترین هزینه مسیر را انتخاب می‌کند. توجه کنید که اگر هزینه همه اعمال یکسان باشد این روش جستجو با جستجوی اول سطح یکسان می‌شود. اگر این روش نودی را گسترش دهد که انجام یک عمل در آن با هزینه صفر منجر به برگشت به همان حالت شود، مانند عمل NoOp در یک حلقه نامتناهی گیر می‌کند.

مثال: جستجوی هزینه یکنواخت





0: [S(0)]
1: [A(1), B(5), C(15)]
2: [B(5), G(11), C(15)]
3: [G(10), G(11), C(15)]
4: [G(11), C(15)]

نکته: در مورد جستجوی هزینه یکنواخت توجه به نکات زیر ضروری است:

- اگر هزینه هر عمل بزرگتر یا مساوی یک ثابت کوچک مثل ε باشد کامل بودن این جستجو تضمین می‌شود، همین شرط برای اطمینان از بهینگی نیزکافیست.
- فرض کنید C^* کل هزینه از گره آغاز تا هدف باشد و هر عمل حداقل ε تا هزینه داشته باشد، در بدترین حالت پیچیدگی مکانی وزمانی $O(b^{(C^*/\varepsilon)})$ می‌باشد.
- اگر هر عملگر هزینه غیر منفی داشته باشد هزینه یک مسیر با ادامه آن، کاهش پیدا نخواهد کرد و الگوریتم می‌تواند کم هزینه ترین مسیر را پیدا کند. اگر بعضی از عمل‌ها هزینه منفی داشته باشند، باید جستجویی از تمام گره‌ها صورت گیرد تا راه حل بهینه پیدا شود.

جستجوی عمقی[1]:

این روش جستجو در هر مرتبه، عمیق ترین گره در مجموعه حاشیه (گره‌های گسترش نیافته) درخت جستجو را گسترش می‌دهد. اگر عمیق ترین نود فرزندی نداشته باشد، جستجو به سمت عمیق ترین نودی که هنوز گسترش نیافته برمی‌گردد.این جستجو می‌تواند توسط الگوریتم Tree _ search با یک صف LIFO پیاده سازی شود.

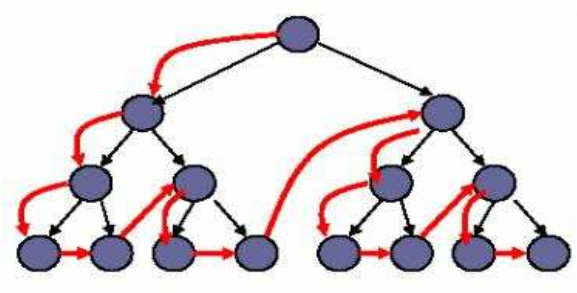
[1] Depth First Search

علاوه بر استفاده از پیاده سازی از طریق Tree _ search، معمولاً جستجوی اول عمق، با یک تابع بازگشتی پیاده سازی می‌شود.

نکته: جستجوی اول عمق به حافظه کمی نیاز دارد. این روش در هر لحظه یک مسیر از ریشه تا یک برگ را به همراه نودهای همزاد[1] هر نود موجود در مسیر را در حافظه نگه می‌دارد. یک نود گسترش یافته به محض اینکه همه نوادگانش کاملا بررسی شدند، از حافظه خارج می‌شود.

نکته: برای هر فضای حالت با فاکتور انشعاب b و عمق ماکزیمم m جستجوی اول عمق فقط به اندازه bm+1 نود حافظه نیاز دارد. نوع دیگر از جستجوی اول عمق، جستجوی عقبگرد[2] نامیده می‌شود که حافظه کمتری استفاده می-کند.در این جستجو در هر لحظه به جای تولید همه فرزندان، فقط یک فرزند تولید می‌شود. هر نود جزئی گسترش یافته به خاطر می‌سپارد که کدام فرزندانش باید بعدا گسترش یابند. در این حالت پیچیدگی حافظه O(m) می‌باشد.

معایب: اشکال جستجوی اول- عمق این است که ممکن است با یک انتخاب نادرست در یک مسیر خیلی طولانی یا حتی نامتناهی گیر کند در حالیکه یک راه حل، نزدیک به ریشه درخت وجود دارد، اگر زیر درخت چپ دارای عمق نامحدود باشد و شامل هیچ راه حلی نباشد جستجوی اول - عمق هرگز تمام نمی‌شود. لذا این روش جستجو، با این فرضیات، کامل نیست. در بدترین حالت جستجوی اول-عمق همه نودهای درخت جستجو را تولید خواهد کرد. یعنی پیچیدگی زمانی آن $O(b^m)$ می‌باشد. یکی دیگر از مشکلات جستجوی اول - عمق حلقه بی پایان می‌باشد.مثالی از جستجوی اول-عمق در شکل زیر نشان داده شده است.

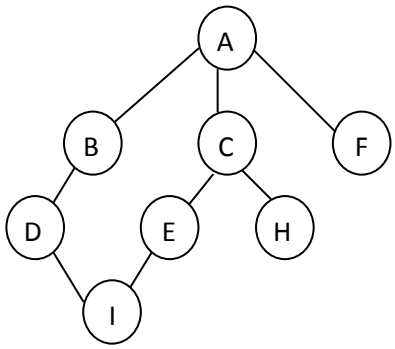


جستجوی عمقی

[1] Sibling

[2] Back-tracking search

مثال: اگر در گراف زیر جستجوی اول- عمق را از راس C شروع کنیم، ترتیب رویت شدن گره‌ها از چپ به راست چگونه خواهد بود.(فرض کنید فرزندان یک گره بر اساس ترتیب حروف الفبا انتخاب می‌شوند)

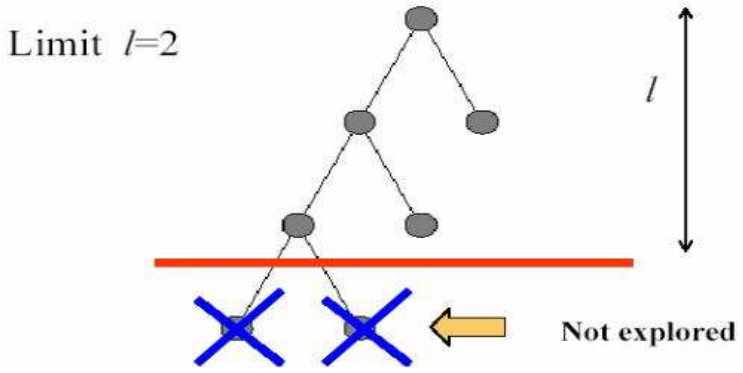


C	H	E	A	F	B	D	I

Output: C A B D I F E H

جستجوی عمقی محدود شده[1]:

مشکل درخت‌های نامحدود با مشخص کردن حدی برای عمق، حل می‌شود. وقتی درخت را به عمق L محدود کنیم یعنی فرض کرده‌ایم نودها در عمق L فرزندی ندارند این روش جستجوی عمقی محدود شده نامیده می‌شود. اگر کم عمق ترین هدف بعد از عمق برش قرار گیرد یعنی L<d باشد این روش جستجو کامل نیست. اگر L>=d باشد جستجوی عمقی محدود شده کامل است، اما همچنان بهینه نیست. پیچیدگی زمانی $O(b^l)$ و پیچیدگی مکانی O(bl) می‌باشد. جستجوی اول عمق نوع خاصی از جستجوی عمقی محدود شده با L= ∞ می‌باشد. گاهی اوقات عمق برش بر اساس دانش مسئله تعیین می‌شود به عنوان مثال در نقشه رومانی ۲۰ شهر وجود دارد. لذا اگر راه حلی وجودداشته باشد، بیشترین طولی که می‌تواند داشته باشد ۱۹ است و L= 19 یک انتخاب ممکن است.



جستجوی عمقی محدود شده

جستجوی عمیق شونده تکراری[2]:

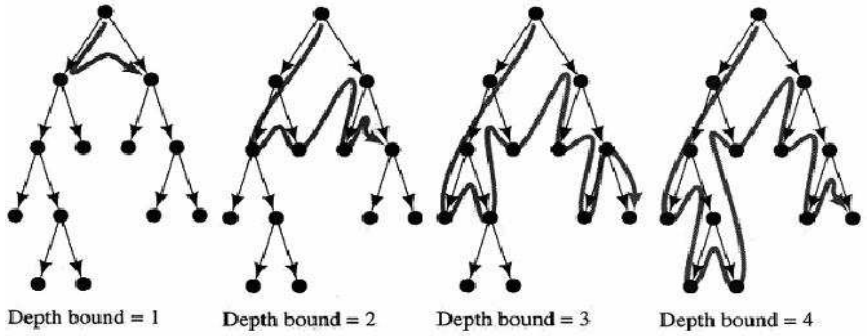
این روش در واقع همان جستجوی عمقی محدود شونده است، که برای یافتن بهترین عمق برش چندین بار اجرا می-شود. جستجوی عمقی تکرار شونده این کار را با افزایش آهسته عمق برش انجام می‌دهد. ابتدا عمق ۱ سپس ۲ و ...

---

[1] Depth Limited Search

[2] Iterative-Depth Search

یک هدف پیدا شود. عمق برش که به d می‌رسد هدف پیدا خواهد شود. این روش جستجو، جستجوی اول عمق و اول سطح را ترکیب می‌کند، مانند جستجوی اول سطح، با فاکتور انشعاب متناهی کامل است و وقتی هزینه مسیر یک تابع غیر کاهش از عمق نود باشد بهینه است و پیچیدگی مکانی این روش مانند جستجوی عمقی خطی می‌باشد. پیچیدگی مکانی آن o(bd) است. زیرا در آخرین تکرار حداکثر عمق درخت، d می‌باشد.



Stages in Iterative-Deepening Search

جستجوی عمقی تکرار شونده

نکته: در جستجوی عمقی تکرار شونده، نودها در آخرین سطح یکبار، در سطح ماقبل آن دوبار، و ... تولید می‌شوند. بنابراین تعداد کل نودهای تولید شده عبارتست از:

پیچیدگی زمانی عمیق کننده تکراری

DLS (l = 0)	0
DLS (l = 1)	b^1
DLS (l = 2)	$b^1 + b^2$
.	.
.	.
.	.
DLS (l = d - 1)	$b^1 + b^2 + b^3 + \ldots + b^{d-1}$
DLS (l = d)	$b^1 + b^2 + b^3 + \ldots + b^{d-1} + b^d$

سربار $\leq N_{DLS\ (l=d)}$

$$N_{IDS} = db^1 + (d-1)b^2 + (d-2)b^3 + \ldots + 2b^{d-1} + b^d$$

کارایی IDS

تعداد گره‌های تولید شده توسط DLS در عمق d با فاکتور انشعاب b:

$$N_{DLS} = b + b^2 + \cdots + b^{d-1} + b^d$$

تعداد گره‌های تولید شده توسط IDS در عمق d و با فاکتور انشعاب b:

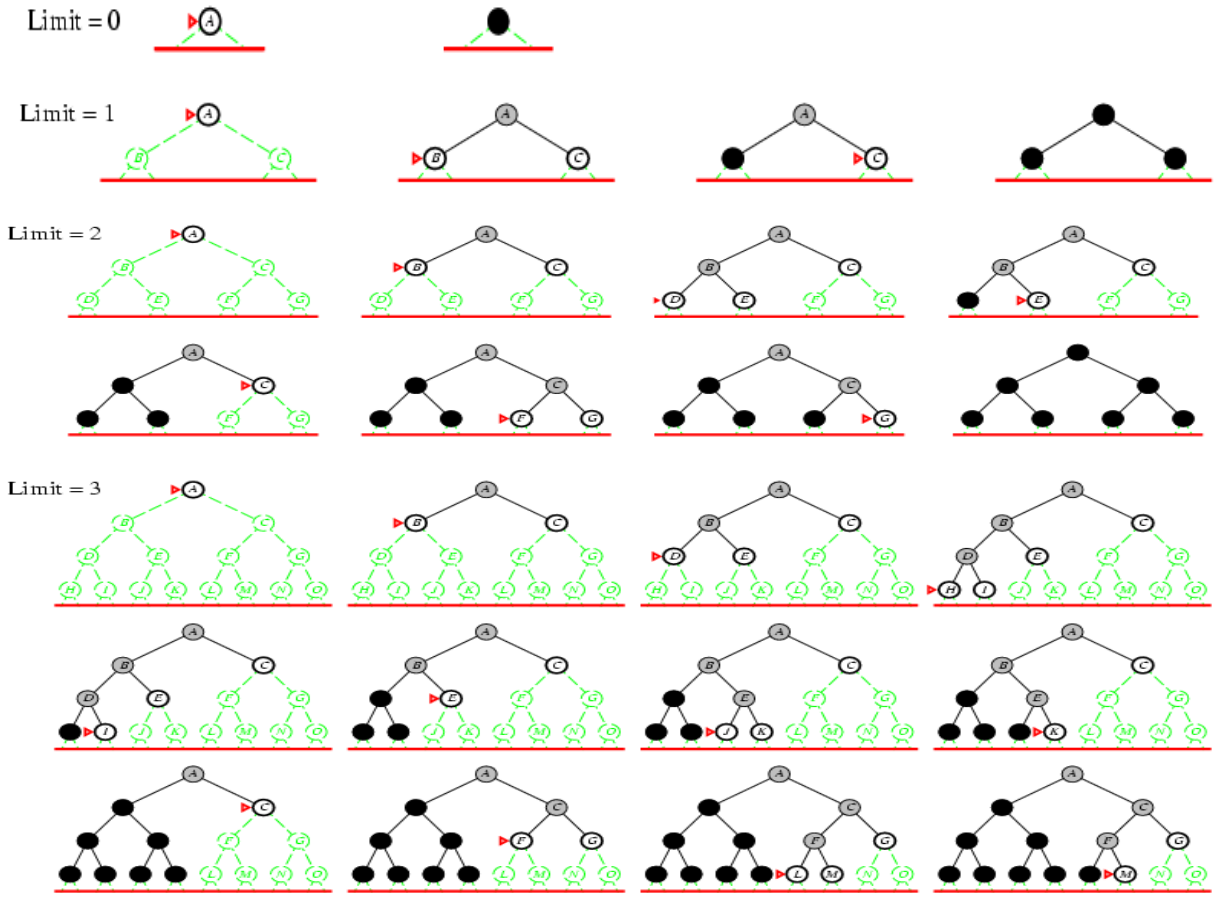
$$N_{IDS} = db^1 + (d-1)b^2 + \cdots + 2b^{d-1} + 1b^d$$

اگر b = 10 و d = 5:

N_{DLS} = 1 + 10 + 100 + 1,000 + 10,000 + 100,000 = 111,111

N_{IDS} = 6 + 50 + 400 + 3,000 + 20,000 + 100,000 = 123,456

بنابراین جستجوی عمقی تکرارشونده علی رغم تولید تکراری حالات سریع تر از جستجوی اول سطح می‌باشد. به طور کلی اگر فضای حالت بزرگ و عمق راه حل نامشخص باشد، در میان جستجوهای ناآگاهانه، جستجوی عمقی تکرار شونده ترجیح داده می‌شود. اگر خصوصیت تکرار شونده این جستجو در جستجو با هزینه یکنواخت استفاده شود منجر به کاهش نیازمندی‌های حافظه آن خواهد شد. البته در این مورد به جای محدودیت عمق از محدودیت مسیر استفاده می‌شود. مثالی از جستجوی عمقی تکرار شونده در شکل زیر بیان شده است.

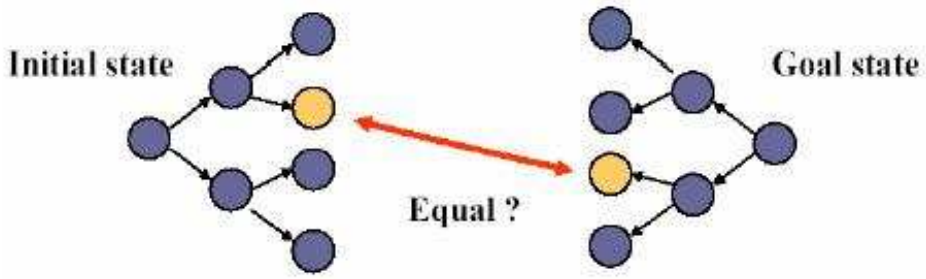


جستجوی دو طرفه[1]:

ایده این روش جستجو از حالت شروع به سمت جلو و از حالت نهایی به سمت عقب می‌باشد. بدین ترتیب که، از حالت شروع عملگرها را مستقیم اعمال کرده و از حالت هدف نیز معکوس عملگرها را اعمال می‌کنیم و وقتی به حالت مشترک رسیدیم این مسیر جواب خواهد بود.

نکته: انگیزه به کارگیری این روش این خواهد بود که: $O(b^{d/2})=b^{d/2}+b^{d/2} <= b^{d}$

[1] Bidirectional search



جستجوی دو سویه

این جستجو بدین ترتیب پیاده سازی می‌شود که یکی از جستجوهای آن یا هر دو قبل از گسترش یک نود بررسی می‌کنند که آن نود در مجموعه حاشیه دیگری وجود دارد یا نه، اگر وجود داشت راه حل پیدا شده است. بررسی وجود یک نود در مجموعه حاشیه جستجوی دیگری با استفاده از جدول درهم سازی[1] در زمان ثابتی انجام می‌شود لذا پیچیدگی زمانی جستجوی دو طرفه $O(b^{d/2})$ است و پیچیدگی مکانی جستجوی دو طرفه نیز $O(b^{d/2})$ است. این میزان حافظه مهم ترین ضعف جستجوی دوطرفه است. اگر هر دو طرف این جستجو از جستجوی اول سطح استفاده کنند، این جستجو کامل و اگر هزینه اعمال یکسان باشند، این جستجو بهینه است. اگر در دو طرف این الگوریتم، ترکیبات دیگری از جستجوها استفاده شود کامل بودن و بهینگی آن تعیین نمی‌شود.

معایب این روش (جست و جوی دو طرفه):

- در این روش برای حرکت از سمت راست به عقب نیاز به معکوس عملگر می‌باشد، که این بدان معنا است که اگر در یک گره باشیم از چه گره‌ای می‌توان به این گره رسید، این روش در مورد تمام مسئله‌ها امکان پذیر نیست، چون بعضی از عملگرها را نمی‌توان معکوس اعمال کرد.
- در بعضی از موارد حالت هدف تعریف صریح و مشخصی ندارد و ضمنی است، مانند بازی شطرنج که حالت هدف، این است که حریف مات شود لذا حالت هدف صریحی، وجود ندارد.
- نیاز به تصمیم گیری دارد که چه نوع جست و جویی در هر طرف انجام می‌شود.

مقایسه استراتژی‌های جستجوی ناآگاهانه

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening
Complete?	Yes	Yes	No	No	Yes
Time	$O(b^{d+1})$	$O(b^{\lceil C^*/\epsilon \rceil})$	$O(b^m)$	$O(b^l)$	$O(b^d)$
Space	$O(b^{d+1})$	$O(b^{\lceil C^*/\epsilon \rceil})$	$O(bm)$	$O(bl)$	$O(bd)$
Optimal?	Yes	Yes	No	No	Yes

مقایسه روش‌های جستجوی ناآگاهانه

b: فاکتور انشعاب I: محدوده عمق انتخاب شده

d: عمق کم عمق ترین مسیر راه حل m: عمیق ترین عمق مسئله

[1] Hash table

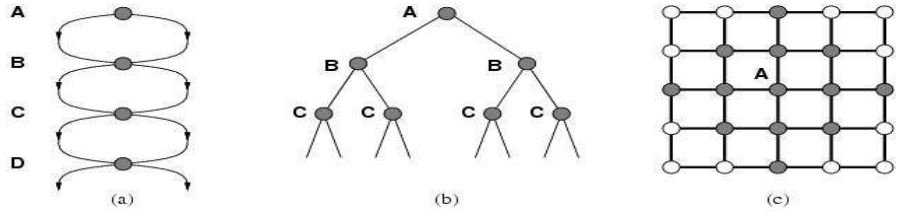
اجتناب از حالت‌های تکراری :

حالت تکراری به معنای گسترش نودهایی است که چند لحظه پیش با آنها مواجه شده‌ایم و قبلا گسترش یافته‌اند این گسترش باعث اتلاف زمان و فضا می‌شود. مشکل حالات تکراری برای حالتی که فضای حالت آن یک درخت است و برای رسیدن به هر حالت فقط یک راه وجود دارد، هرگز اتفاق نمی‌افتد. مثلاً مسئله ۸ وزیر به طوری که هر وزیر در سمت چپ ترین ستون خالی قرار گیرد، بسیار کارا است، زیرا فقط یک مسیر برای رسیدن به هر نود وجود دارد، در برخی مسائل، حالات تکراری غیر قابل اجتناب است، این موضوع شامل مسائلی است که در آنها عملگرها قابل وارون شدن باشد، مثل مسیریابی و معمای پازل ۸، که از این دسته‌اند.درخت جستجوی این مسائل،نامتناهی است، اما اگر بعضی حالات تکراری را حذف کنیم، درخت جستجو، متناهی می‌شود.وجود حالات تکراری ممکن است موجب غیر قابل حل شدن یک مسئله حل شدنی شود. سه راه حل برای مشکل راه حل تکراری وجود دارد:

- به حالتی که هم اکنون آمده‌اید برنگردید.
- از ایجاد مسیرهای دوار بپرهیزید.
- حالتی را که قبلا تولید شده مجددا تولید نکنید، چون این مسئله باعث می‌شود که هر حالت درحافظه نگهداری شود و پیچید گی فضایی $O(s)$ که s تمام حالات در درخت فضای حالت است ایجاد شود.

الگوریتم Graph_search: اگر یک الگوریتم هر حالتی را که تاکنون دیده است به خاطر آورد، آنگاه می‌تواند گراف فضای حالت را به طور مستقیم جستجو کند.اگر به الگوریتم Tree_search یک ساختمان داده به نام closed که نودهای گسترش یافته را ذخیره می‌کند اضافه کنیم الگوریتم جدیدی به دست می‌آید که Graph_search نام دارد، مجموعه نودهای گسترش نیافته در لیست open ذخیره می‌شوند، و مجموعه نودهای گسترش یافته در لیست close ذخیره می‌شوند.

مقایسه Graph_search با tree_search: اگر نود جاری با نودهای موجود در لیست close انطباق پیدا کرد، به جای گسترش حذف می‌شود، در مورد مسائلی با حالت تکراری زیاد Graph_ search کاراتر از Tree_search است. پیچیدگی زمانی و مکانی Graph_search در بدترین وضعیت تابع نمایی از سایز فضای حالت است که خیلی کوچک تر از $O(b^d)$ می‌باشد. توجه کنید که استفاده از لیست close، سبب می‌شود، که پیچیدگی مکانی جستجوی اول عمق و جستجوی عمقی تکرار شونده خطی نباشد. از آنجا که الگوریتم graphs همه نودها را در حافظه نگه می-دارد ،بنابراین در برخی مسائل به خاطر محدودیت حافظه، اجرا نشدنی هستند.



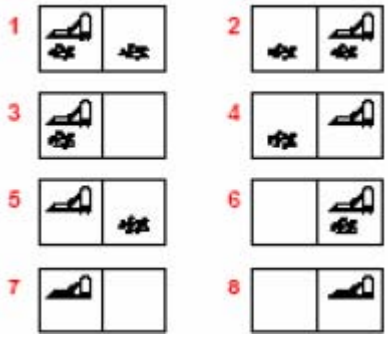
شکل بالا، فضای حالتی را نشان می‌دهد که درخت جستجوی به طور نمایی بزرگتری را تولید می‌کند.

جستجو با اطلا عات ناقص[1]:

برای جستجوهای ناآگاهانه، فرض کردیم، که محیط کاملاً مشاهده پذیر و قطعی است، و عامل نتیجه اعمالش را می-داند و عامل می‌تواند دقیقاً محاسبه کند کدام حالت از کدام رشته از اعمالش، نتیجه می‌شود، و همیشه می‌داند در کدام حالت قرار دارد. اینگونه مسائل تک حالته نامیده می‌شوند. حال اگر دانش حالات و اعمال، ناکافی باشند باعث به وجود آمدن ۳ نوع مسئله می‌شوند:

I. مسائل بدون حسگر[2]:

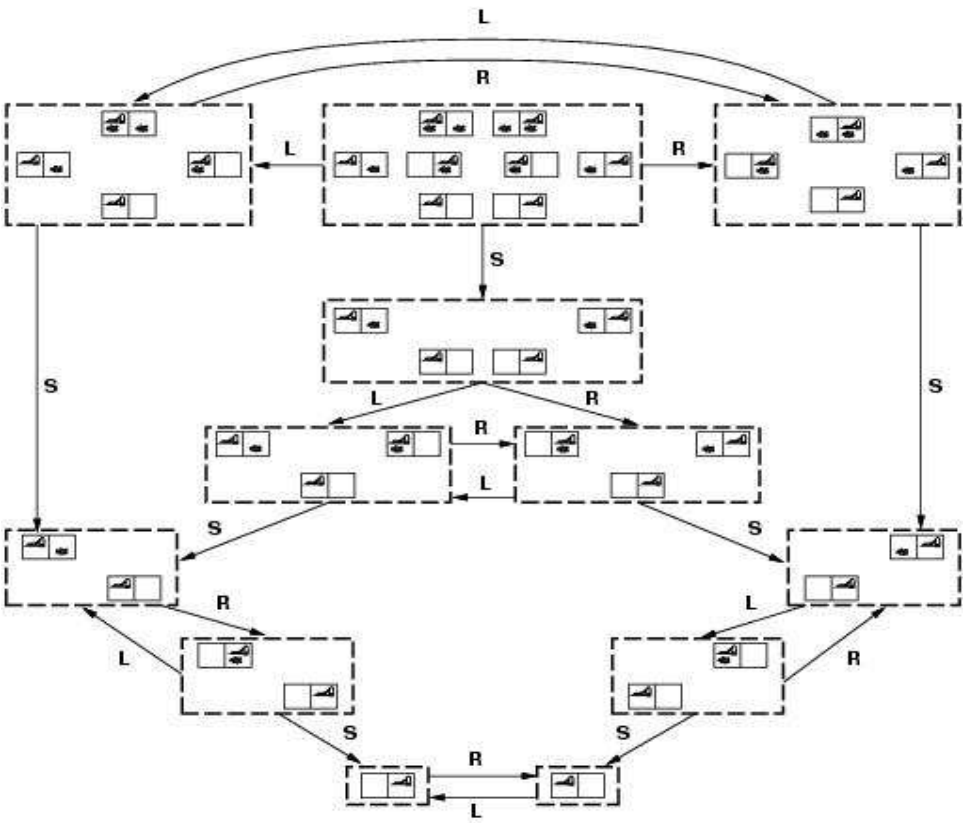
وقتی عامل حسگر نداشته باشد، فقط می‌داند در یکی از حالات اولیه ممکن است و با انجام هر عمل به یکی از چندین حالت بعدی که حالت باور نامیده می‌شود منتقل می شود، به عبارتی دیگر در اینگونه مسائل یک حالت به صورت مجموعه‌ای از حالات باور تعریف می‌شود لذا به این مسائل، چند حالته نیز گفته می‌شود.



شکل الف: فضای حالت برای دنیای معین بدون حسگر جارو برقی

[1] Searching with partial information

[2] Sensor less problem



شکل ب: فضای حالت باور برای دنیای معین بدون حسگر جارو برقی

II. مسائل اقتضائی[1] (احتمالی):

اگر محیط پاره‌ای مشاهده پذیر باشد یا اعمال غیر قطعی باشند، مشاهدات عامل بعد از هر عمل، اطلاعات جدیدی را در اختیار می‌گذارد. هر مشاهده‌ای یک احتمال وقوع تعریف می‌کند، که می‌تواند برای آن برنامه ریزی شود، اگر عدم قطعیت به دلیل فعالیت‌های عامل دیگر ایجاد شود، مسئله خصمانه نامیده می‌شود.

III. مسائل اکتشافی[2]:

اگر هم حالات و هم اعمال محیط (فضای حالت)، ناشناخته باشند عامل باید به دنبال کشف آنها باشد، در واقع مسائل اکتشافی، تعمیم یافته مسائل احتمالی است.

انواع مسئله

i. **قطعی، کاملاً مشاهده پذیر:** مسائل تک حالته

- ❖ عامل دقیقا می‌داند در چه حالتی خواهد بود، راه حل یک دنباله می‌باشد.

ii. **قطعی، مشاهده پذیر جزئی:** مسائل چند حالته

- ❖ ممکن است عامل ایده‌ای درباره اینکه کجاست نداشته باشد، راه حل یک دنباله است.

---

[1] Contingency

[2] Exploration problem

iii. **غیر قطعی و / یا مشاهده پذیر جزئی:** مسائل احتمالی

- ❖ ادراک اطلاعات جدیدی درباره حالت فعلی فراهم می‌کند.
- ❖ در حین اجرا باید از حسگرها استفاده کند.
- ❖ راه حل بصورت یک درخت.
- ❖ اغلب جستجو و اجرا بصورت یک در میان (interleave)

iv. **فضای حالت ناشناخته:** مسائل اکتشافی (online)

سوال‌های تستی آخر فصل

ترم اول ۸۷–۸۸

۱. طراحی عامل به چه شکلی است؟

الف. تدوین هدف، روال جستجو، اجرا ب. مقداردهی اولیه، جستجو، اجرا

ج. مشاهدات، استنتاج اجزا د.مشاهدات، استنتاج، ارائه راه حل

۲. کدامیک از روش‌های ارزیابی الگوریتم محسوب نمی‌شود؟

الف. کامل بودن ب.قابل درک بودن ج.پیچیدگی زمانی د.پیچیدگی فضایی

۳. فرض کنید A یک جاروبرقی اتوماتیک است. محیط این جاروبرقی، مطابق شکل زیر، از سه خانه کنارهم تشکیل شده است. این جاروبرقی می‌تواند از هر یک از این خانه‌ها با انجام یک حرکت به خانه مجاور نقل مکان نماید و زباله موجود در آن خانه را (درصورت وجود) جمع آوری کند. باتوجه به اینکه این جاروبرقی برای جمع آوری هر زباله باید درهمان خانه‌ای که زباله در آن وجود دارد، قرار بگیرد، فضای حالت این مسأله دارای چند وضعیت منحصر بفرد است؟

	A	

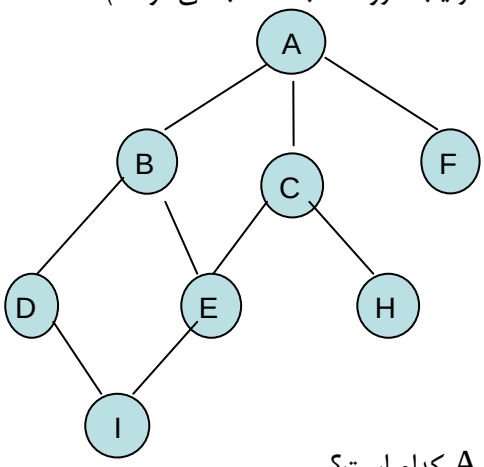
الف. ۲۴

ب. ۹

ج. ۶۳

د. ۸۱

۴. اگر در گراف زیر جستجو در عمق (Depth first search) را از راس C شروع کنیم، کدام گره‌ها به ترتیب از چپ به راست رؤیت می‌شوند؟ (فرض کنید فرزندان یک گره بر اساس ترتیب حروف الفبا انتخاب می‌شوند.)



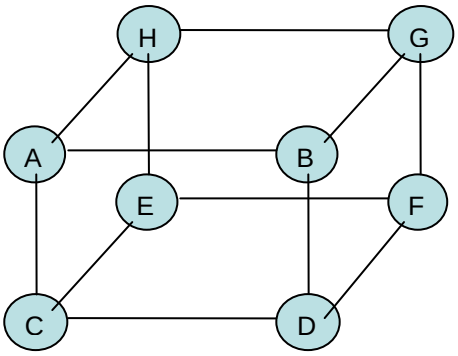
الف. ABCDEFHI

ب. CABDIEFH

ج. CAEHBFID

د. CABDEHIF

۵. پیمایش اول عمق (Depth first) برای گراف مقابل با شروع از گره A کدام است؟



الف. ABDCEFHG

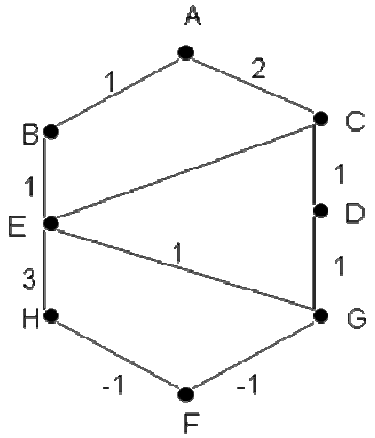
ب. ACEFDBHG

ج. ACDFBEGH

د. AHGBDCEF

۶. کدامیک از الگوریتم‌های جستجو زیر از لحاظ زمان و حافظه برروی گراف فوق بهتر عمل می‌کند؟

(گره A شروع وگره G هدف می‌باشد. هزینه عملگرها برای رفتن از یک گره به گره دیگر روی یال‌ها نوشته شده است).



الف. جستجوی اول سطح

ب. جستجوی اول عمق

ج. جستجوی با هزینه یکسان

د. جستجوی عمقی محدود شده با حداکثر عمق ۲

۷. کدامیک از الگوریتم‌های زیر کامل نمی‌باشد؟

الف.دوطرفه ب.اول عمق ج.اول سطح د.عمیق شونده تکراری

۸. کدامیک از الگوریتم‌های زیر از لحاظ فضا مناسب تر است؟

الف.دوطرفه ب. اول سطح ج.با عمق محدود د.اول عمق

ترم دوم ۸۷–۸۸

۹. عامل‌های حل مسئله نوعی از.....می‌باشد.

الف. عامل‌های معقول ب. عامل‌های واکنشی ج. عامل‌های مبتنی بر هدف د. عامل‌های یاد گیرنده

۱۰. انواع مختلف نقص (کامل نبودن) منجر به کدام نوع مسئله نمی‌شود؟

الف. مسئله بدون حسگر (منطبق) ب. مسائل اقتضایی

ج. مسائل هیوریستیک د. مسائل اکتشافی

۱۱. کدام جستجو از لحاظ زمانی ارجح است؟

الف. جستجوی دوطرفه ب. جستجوی عمیق شونده تکراری

ج. جستجوی عمق محدود د. جستجوی هزینه یکنواخت

۱۲. بزرگترین مشکل الگوریتم جستجوی اول سطح کدام گزینه است؟

الف. زمان اجرا ب. حافظه ج. کامل بودن د. ناقص بودن راه حل

۱۳. کدام الگوریتم جستجو برای مسائلی که عمق آنها زیاد است، مناسب نیست؟

(به دلیل اینکه اگر در مسیر اشتباهی بیفتد، مدت زمان زیادی طول خواهد کشید که متوجه شود مسیرش غلط بوده و به حالت قبلش برگردد.)

الف. اول سطح ب. با هزینه یکنواخت ج. اول عمق د. عمیق شونده تکراری

۱۴. الگوریتمی که از لحاظ زمانی از مرتبه جستجوی اول سطح است ولی از لحاظ پیچیدگی حافظه از رتبه جستجوی اول عمق می باشد، کدام گزینه است؟

الف. جستجوی تپه نوردی ب. جستجوی هزینه یکنواخت

ج. جستجوی عمق محدود د. جستجوی عمیق شونده تکراری

ترم تابستان ۸۸

۱۵. در تدوین مسئله « حذف جزئیات از یک بازنمایی » را چه می‌نامند؟

الف. تجرید (Relaxation) ب. تعدیل شده (Abstraction)

ج. به اشتراک گذاری (Validation) د.معتبر سازی (Sharing)

۱۶. کدام گزینه در رابطه با الگوریتم‌های جستجو صحیح است؟

الف. اگر فاکتور انشعاب محدود باشد، جستجوی اول عمق کامل است.

ب. جستجوی هزینه یکنواخت سربار قابل توجهی را در مقایسه با جستجوی طولانی کننده تکراری ایجاد می‌کند.

ج. میزان حافظه مورد استفاده در جستجوی اول عمق نسبت به جستجوی اول سطح کمتر است.

د. اگر فاکتور انشعاب، متناهی باشد، آنگاه جستجوی هزینه یکنواخت کامل است، ولی بهینه نیست.

ترم اول ۸۸–۸۹

۱۷. با تغییر راهبرد جستجو (search strategy) در (Tree search)، کدام جستجو را نمی‌توان دید؟

الف. اول سطح ب. هزینه یکنواخت ج. اول عمق د. جستجوی دوطرفه

۱۸. بزرگترین مشکل جستجوی اول سطح مثلاً با (b=10) کدام است؟

الف. کامل نبودن ب. عدم بهینگی ج. پیچیدگی زمانی د. پیچیدگی حافظه

۱۹. با در نظر گرفتن لیست شرایط زیر، روش جستجوی هزینه یکنواخت در چه شرایطی کامل است؟

۱. در هر شرایطی

۲. در شرایطی که هزینه اقدامات در یک سطح برابر باشد.

۳. به شرطی که فاکتور انشعاب متناهی باشد.

۴. در هر دوجهت از جستجوی اول سطح استفاده شود.

الف. ۲ ب. ۳ ج. ۳و۲ د. ۳و۴

۲۰. استفاده از Graph search در چه مسائلی بسیار مؤثرتر از Tree search است؟

الف. مسائل با حالت‌های تکراری زیاد در درخت ب. در مسائل با پیچیدگی زمانی بالا

ج. در مسائل با پیچیدگی زمانی حافظه بالا د. در مسائل اکتشافی

۲۱. در محیط کاملاً رویت پذیر و قطعی برای عامل (کارگزار) جاروبرقی " بدون حسگر" کدام گزینه صحیح نیست؟

الف.حالت اولیه = مجموعه حالت باور شامل ۸ حالت ممکن

ب. حالت هدف = دو حالت هدف مجزا وجود دارد (دو مجموعه حالت باور هدف هر کدام یک حالت هدف را در بر دارد.)

ج. به دلیل نداشتن حسگر عامل گاهی هدف را نخواهد یافت.

د. تنها ۱۲ حالت باور دسترس پذیر وجود دارد.

ترم دوم ۸۸–۸۹

۲۲. پیچیدگی زمانی جستجوی عمیق شونده تکراری به کدام عامل زیر بستگی دارد؟

الف. بیشترین عمق درخت ب. اندازه فضای حالت

ج. تابع هیوریستیک (اکتشافی) د. عمق کم عمق ترین گره درخت

۲۳. کدام گزینه در مورد الگوریتم جستجوی گراف (Graph-search) صحیح نیست؟

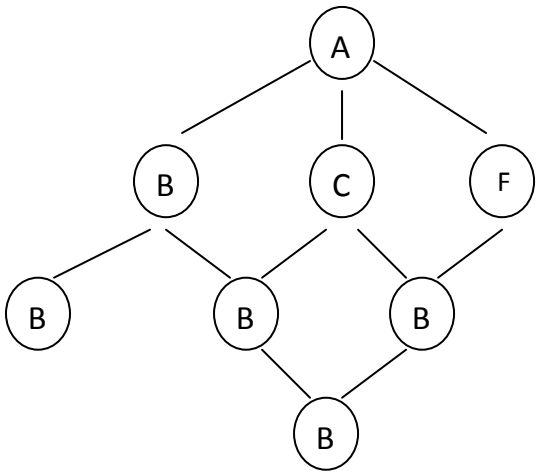
الف. درمسائلی که حالت‌های تکراری زیادی دارد مؤثرتر از Tree-search عمل می‌کند.

ب. خطر از دست دادن بهینگی را دارد.

ج. تمام گره‌های گسترش یافته را در حافظه ذخیره می‌کند.

د. جستجوی هزینه یکنواخت با الگوریتم Graph-search بهینه نیست.

۲۴. در گراف زیر با انجام جستجوی اول عمق و شروع از راس C، کدام گره‌ها به ترتیب از چپ به راست گسترش می‌یابند؟ (فرزندان یک گره بر اساس ترتیب حروف الفبا نوشته می‌شوند.) (با استفاده از Graph_search)



الف. C, A, H, B, F, D, E, G

ب. C, A, E, H, B, F, G, D

ج. C, A, B, D, F, E, G, H

د. C, A, B, D, E, G, H, F

۲۵. پاسخ سؤال قبل با جستجوی اول سطح کدام گزینه است؟ (با استفاده از Graph_search)

الف. C,A,E,H,B,F,G,D ب. C,A,H,B,F,D,E,G

ج. C,A,B,D,F,G,H,E د. C,A,B,D,F,E,G,H

۲۶. در مورد جستجوی دوطرفه کدام گزینه درست نیست؟

الف. پیچیدگی زمانی آن $O(d^{d/2})$ است.

ب. بزرگترین نقطه ضعف آن، میزان حافظه بری آن است.

ج. اگر هر دو جستجو از نوع اول عمق باشند، الگوریتم کامل و بهینه است.

د. انجام جستجوی دوطرفه همیشه عملی نیست.

ترم تابستان ۸۹

۲۷. کدامیک از عامل‌ها به صورت " تدوین، جستجو، اجرا " طراحی می‌شوند؟

الف. واکنشی ساده ب. حل مسئله ج. مبتنی بر دانش د. مبتنی بر جدول

۲۸. فضای حالت مسئله به طور ضمنی توسط کدام گزینه قابل تعریف است؟

الف. حالت‌ها واقدامات ب. حالت ابتدایی و اقدامات

ج. حالت شروع و حالت هدف د. حالت ابتدایی و تابع پسین

۲۹. کدام گزینه تعریفی از تجرید (abstraction) را ارائه می‌دهد؟

الف. در نظر گرفتن یک هدف واحد، که قصد رسیدن به آن را داریم.

ب. کنار گذاشتن حالت‌هایی که برای جستجوی هدف فعلی به ما کمکی نمی‌کنند.

ج. حذف جزئیات از یک بازنمایی

د. کنار گذاشتن مسیرهای انحرافی که مارا از هدف دور می‌کند.

**** با در نظر گرفتن شرایط زیر به سؤالات ۳۰ و ۳۱ پاسخ دهید.

در هر شرایطی

در شرایطی که هزینه‌ی اقدامات در یک سطح برابر باشد.

به شرطی که فاکتور انشعاب متناهی باشد.

هزینه ی هر اقدام از ε بزرگتر باشد.

در هر دو جهت از جستجوی اول سطح استفاده شود.

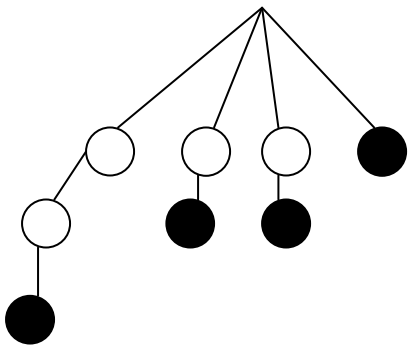
۳۰. جستجوی اول سطح در چه شرایطی بهینه است؟

الف. ۱ ب. ۲ ج. ۳ د. ۴

۳۱. در چه شرایطی جستجوی دوطرفه بهینه است؟

الف.۲و۴ ب. ۲و۳ ج. ۳و۴ د. ۲و۵

۳۲. در حین انجام یک جستجو، درخت جستجوی حاصل به شکل زیر رشد یافته است. راس‌هایی که نامزد بسط داده شدن هستند به رنگ سیاه مشخص شده‌اند. این جستجو چه روشی می‌تواند باشد؟



الف. اول عمق (Depth first)

ب. اول سطح (Breadth first)

ج. جستجوی هزینه یکنواخت (Uniform cost)

د. عمیق شوند تکراری (Iterative deepening)

۳۳. اگر در مسئله n وزیر، هر وزیر بتواند در هر خانه‌ایی که وزیری در آن نباشد قرار گیرد در آن صورت یک حالت که شامل n وزیر باشد در درخت فضای حالت در چند گره درخت تکرار می‌شود؟

الف. n ب. !n ج. تنها یک گره د. log n

۳۴. در عامل جاروبرقی اگر اقدام Suck به این صورت تعریف شود که در صورت کثیف بودن مکش و در صورت تمیز بودن، گاهی آشغال‌ها را روی فرش خالی می‌کند. (محیط مورفی) در این صورت عامل در کدام محیط می‌تواند هدف را بیابد.

الف. محیط رویت ناپذیر (مسئل بدون حسگر)

ب. محیط نیمه رویت پذیر (مسائل اقتضایی)

ج. محیط کاملاً رویت پذیر

د. در محیط مورفی عامل جاروبرقی حتی با محیط کاملاً رویت پذیر گاهی ناموفق خواهد بود.

۳۵. نقطه ضعف اصلی A* چیست؟

الف. کامل نبودن ب. نیمه بهینگی ج. پیچیدگی زمانی د. پیچیدگی حافظه

ترم اول ۸۹–۹۰

** با در نظر گرفتن شرایط زیر به سوالات ۳۶ و ۳۷ پاسخ دهید:

در هر شرایطی

در شرایطی که هزینه اقدامات در یک سطح برابر باشد.

به شرطی که فاکتور انشعاب متناهی باشد.

هزینه ی هر اقدام از ε بزرگتر باشد.

در هر دو جهت از جستجوی اول سطح استفاده شود.

۳۶. روش حل عمیق شونده تکراری در چه شرایطی بهینه است؟

الف. ۴ ب. ۳ ج. ۲ د. ۱

۳۷. در چه شرایطی جستجوی دو طرفه کامل است؟

الف. ۳ و۵ ب. ۵ ج. ۳ و۴ د. ۱

۳۸. کدام جستجو از لحاظ پیچیدگی زمانی ارجح است ؟

الف. اول سطح ب. اول عمق ج. عمیق شونده تکراری د. دو طرفه

۳۹. در کدام نوع از مسائل حالت های تکراری غیر قابل اجتناب هستند؟

الف. مسائل دارای اقدامات معکوس پذیر ب. مسائل اقتضایی

ج. مسائل بدون حسگر د. مسائل اکتشافی

۴۰. در مورد Graph search با جستجوی هزینه یکنواخت کدام گزینه صحیح است؟

الف. کامل و غیر بهینه است. ب. نه کامل و نه بهینه است.

ج. کامل نیست و بهینه است. د.کامل و بهینه است.

۴۱. در محیط کاملا رویت پذیر و قطعی برای عامل جاروبرقی "بدون حسگر" در همان محیط ۲ مکانه با عمل L,R,S کدام گزینه صحیح نیست؟

الف. حالت اولیه: مجموعه حالت باور شامل ۸ حالت ممکن

ب. حالت هدف: دو حالت هدف مجزا وجود دارد (دو مجموعه حالت باور هدف هر کدام یک حالت هدف را در بر دارند)

ج. به دلیل نداشتن حسگر عمل گاهی هدف را نخواهد یافت.

د. تنها ۱۲ حالت باور دسترس پذیر وجود دارند.

ترم دوم ۸۹ –۹۰

۴۲. اگر در جستجوی هزینه یکنواخت گرهی گسترش یابد که دارای اقدامی با هزینه صفر بوده و با آن به همان حالت برگردد چه شرایطی پیش می‌آید؟

الف. جستجو متوقف می‌شود.

ب. جستجو در یک حلقه بینهایت گرفتار می‌شود.

ج. بعد از انتخاب این اقدام با انتخاب اقدام‌های دیگرجستجو ادامه می‌یابد.

د. این اقدام انتخاب نمی‌شود زیرا هزینه گره ایجاد شده آن بیشتر از بعضی گره ها است و حالت جدیدی را نیز بر نمی-گرداند

۴۳. جستجوی دو طرفه برا ی کدامیک از مسائل زیر قابل استفاده و سودمند است؟

الف. مسئله‌ای که همه اقدامات آن معکوس پذیر باشد.

ب. تنها برای مسائلی که یک حالت هدف و اقدامات معکوس پذیر دارند قابل استفاده می‌باشند.

ج. مسئله‌ای که در آن یک و دو یا تعداد حالات کمی هدف وجود دارد و اقدامات معکوس پذیر دارد.

د. مسئله‌ای که در آن تعداد حالاتی که هدف هستند زیاد است (مانند شطرنج)

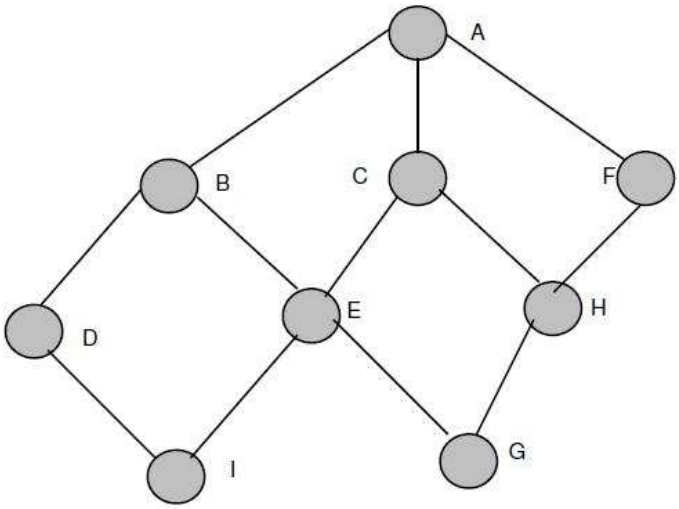
تابستان ۹۰

۴۴. پیچیدگی زمان استراتژی‌های جستجوی دو طرفه و عمیق کننده تکراری، به ترتیب از راست به چپ کدام است؟

(b: فاکتور انشعاب d: عمق پاسخ l: محدودیت عمق)

الف. $b^{d/2}$ و b^{d} ب. b^{d} و b^{l} ج. $b^{d/2}$ و $b^{d/2}$ د. b^{d} و b^{d}

۴۵. اگر در گراف زیر، جستجوی عمقی را از گره c شروع کنیم، ترتیب رویت گره‌ها کدام گزینه است؟ (اولویت گره‌ها بر اساس ترتیب حروف الفباست.)



الف. CABDIEGHF ب. CAEHBFIGD

ج. CEIDBAFHG د. CABDEFGHI

۴۶. اگر تمامی گره‌های موجود در گراف جستجو، دارای هزینه یکسانی باشند، کدام جفت از الگوریتم‌های جستجوی زیر، در این شرایط، یکسان عمل می‌کنند؟

الف. اول سطح و اول عمق ب. اول سطح و هزینه یکنواخت

ج. اول عمق و دوطرفه د. عمیق شونده تکراری و دوطرفه

۴۷. جواب سوال زیر را کدام معیار ارزیابی یک الگوریتم پاسخگو است؟

" آیا در صورت وجود جواب، این الگوریتم قادر به یافتن جواب است "

الف. تابع ارزیابی ب. بهینگی ج. پیجیدگی زمانی د. کامل بودن

پاسخ نامه تستی

سوال	گزینه صحیح	سوال	گزینه صحیح	سوال	گزینه صحیح
۱	الف	۱۶	ج	۳۱	د
۲	ب	۱۷	د	۳۲	ج
۳	الف	۱۸	د	۳۳	ب
۴	ب	۱۹	د	۳۴	الف
۵	د	۲۰	الف	۳۵	د
۶	ب	۲۱	ج	۳۶	ج
۷	ب	۲۲	د	۳۷	الف
۸	ج	۲۳	د	۳۸	د
۹	ج	۲۴	ج	۳۹	الف
۱۰	ج	۲۵	ب	۴۰	د
۱۱	الف	۲۶	ج	۴۱	ج
۱۲	ب	۲۷	ب	۴۲	ب
۱۳	ج	۲۸	د	۴۳	الف
۱۴	د	۲۹	ج	۴۴	الف
۱۵	الف	۳۰	ب	۴۵	الف
				۴۶	ب
				۴۷	د

سوالات تشریحی آخر فصل

ترم اول ۸۷–۸۸

جستجوی اول سطح را تشریح کنید.

جستجوی هزینه یکنواخت را تعریف کنید.

۴ مورد از راهبردهای جستجوی ناآگاهانه را نام ببرید.

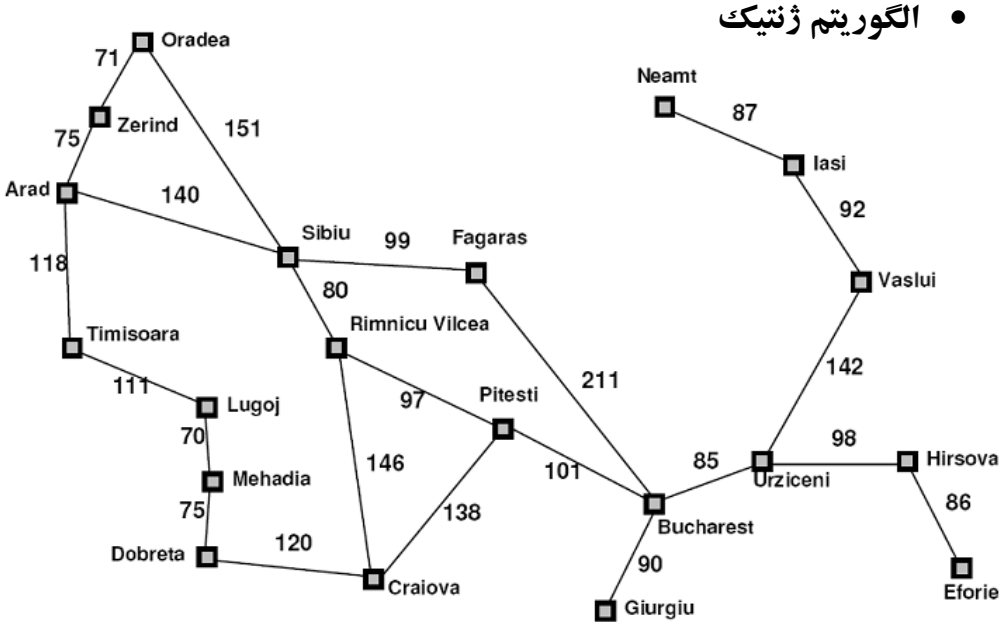
تابستان ۹۰

برای تعریف یک مسئله در هوش مصنوعی از چه مولفه‌هایی استفاده می‌شود؟ برای تعریف مساله‌ی پازل هشت تایی، مقدار هر یک از این مولفه‌ها را مشخص نمایید؟ (۱ نمره)

فصل چهارم: جستجو و اکتشاف آگاهانه

آنچه در این فصل خواهید آموخت:

- ❖ **جستجوی اول-بهترین**
  - **جستجوی حریصانه**
  - **جستجوی *A و خصوصیات آن**
- ❖ **جستجوهای حافظه محدود شده**
  - **جستجوی عمیق کننده تکراری *A (*IDA)**
  - **جستجوی *SMA**
  - **جستجوی اول بهترین بازگشتی (RBFS)**
- ❖ **توابع هیورستیک و مفاهیم مربوطه**
- ❖ **الگوریتم‌های جستجوهای محلی**
  - **جستجوی تپه نوردی**
  - **جستجوی شبیه سازی حرارت**
  - **جستجوی پرتو محلی**
  - **الگوریتم ژنتیک**



مقدمه:

همانطور که در فصل قبل دیدیم جستجوهای ناآگاهانه در بیشتر موارد ناکارا هستند، زیرا در این الگوریتم‌ها، معیار انتخاب گره بعدی برای گسترش، تنها به شماره سطح آن بستگی دارد و از ساختار مسئله بهره نمی‌برند. الگوریتم‌های ناآگاهانه، درخت جستجو را به یک روش از پیش تعریف شده گسترش می‌دهند، یعنی قابلیت تطبیق پذیری با آنچه که تاکنون در مسیر جستجو دریافت کرده‌اند و نیز حرکتی که می‌تواند خوب باشد را ندارند. در این فصل نشان می‌دهیم که چگونه یک الگوریتم جستجوی آگاهانه می‌تواند با کارایی بیشتر، راه حل‌ها را پیدا کند.

جستجوهای آگاهانه :

جستجوی اول– بهترین[۱]:

شیوه‌ای که بر آن متمرکز می‌شویم جستجوی اول–بهترین نامیده می‌شود. جستجوی اول–بهترین مثالی از الگوریتم-های Tree_search و Graph_search است که در آن ترتیب نودها برای گسترش بر اساس یک تابع ارزیابی[۲] f(n) می‌باشد. یعنی در هر مرحله نودی که کمترین مقدار ارزیابی را دارد برای گسترش انتخاب می‌شود. برای پیاده سازی این جستجو از طریق الگوریتم‌های عمومی Tree_search و Graph_search باید از صف اولویت که در آن نودها بر اساس مقدار تابع f(n) به ترتیب صعودی مرتب هستند، استفاده شود. الگوریتم اول بهترین یک حالت کلی دارد و خانواده‌ی الگوریتم‌های اول بهترین در توابع ارزیابی با هم تفاوت دارند. می‌توان نشان داد جستجوی اول سطح با f(n) = Depth(n) و جستجوی هزینه‌ی یکسان با f(n) = g(n) حالت خاصی از جستجوی اول_بهترین هستند. جزء کلیدی در این الگوریتم‌ها تابع هیوریستیک[۳] یا اکتشافی می‌باشد که با h(n) نمایش داده می-شود. تابع h(n) رایج ترین فرمی است که در آن دانش اضافی در مورد مسئله به الگوریتم جستجو اضافه می‌شود. تنها محدودیت تابع h(n) این است که اگر n گره هدف باشد آنگاه h(n) = 0 می باشد.

نکته: هزینه تخمینی از گره n تا گره هدف = h(n)

```
function Best-First-Search(problem,Eval-FN)
                                  returns solution sequence
 nodes := Make-Queue(Make-Node(Initial-State(problem))
loop do
  if nodes is empty then return failure
   node := Remove-Front(nodes)
  if Goal-Test[problem] applied to State(node) succeeds
     then return node
   new-nodes := Expand(node, Operarors[problem],
                          Eval-FN)))
   nodes := Insert-by-Cost(new-nodes,Eval-FN(new-node))
end
```

---

[۱] Best-First Search

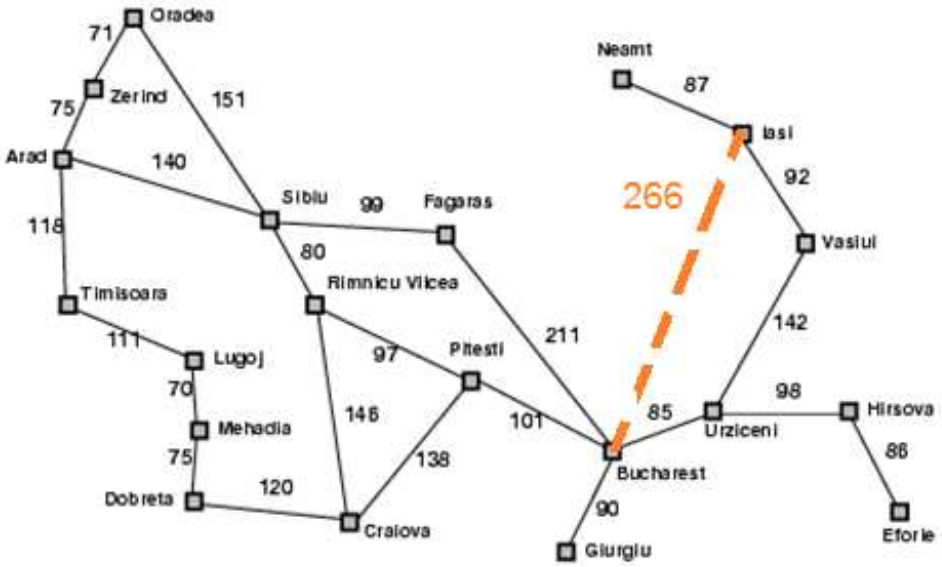
[۲] Evaluation Function

[۳] heuristic

جستجوی حریصانه[1]:

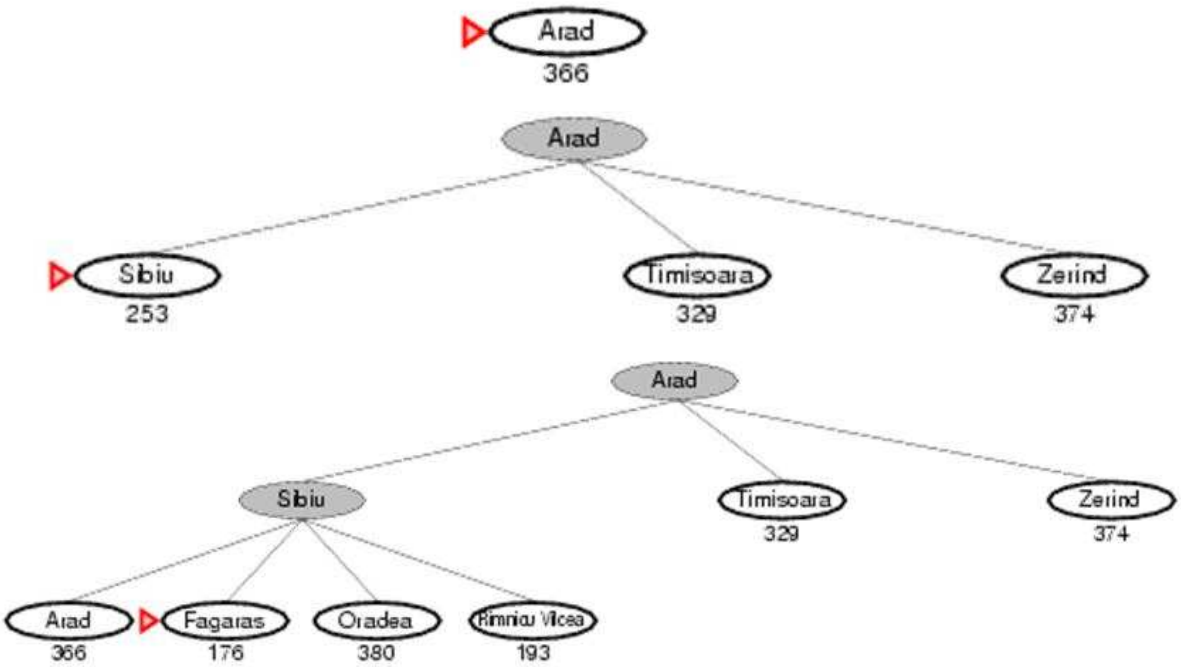
ایده اصلی در این روش به حداقل رساندن هزینه‌ی تخمین زده شده برای رسیدن به هدف می‌باشد، بدین ترتیب که، گره‌ای که به هدف نزدیک تر باشد، ابتدا گسترش می‌یابد. تابعی که هزینه رسیدن از یک حالت به حالت هدف را تخمین می‌زند تابع اکتشافی نامیده می‌شود، و باحرف h نشان داده می‌شود. اکنون این روش جستجو را در مساله مسیریابی در رومانی با استفاده از هیورستیک خط مستقیم بررسی می‌کنیم.

نکته: $h_{SLD}(n)$ = فاصله مستقیم از n تا بخارست

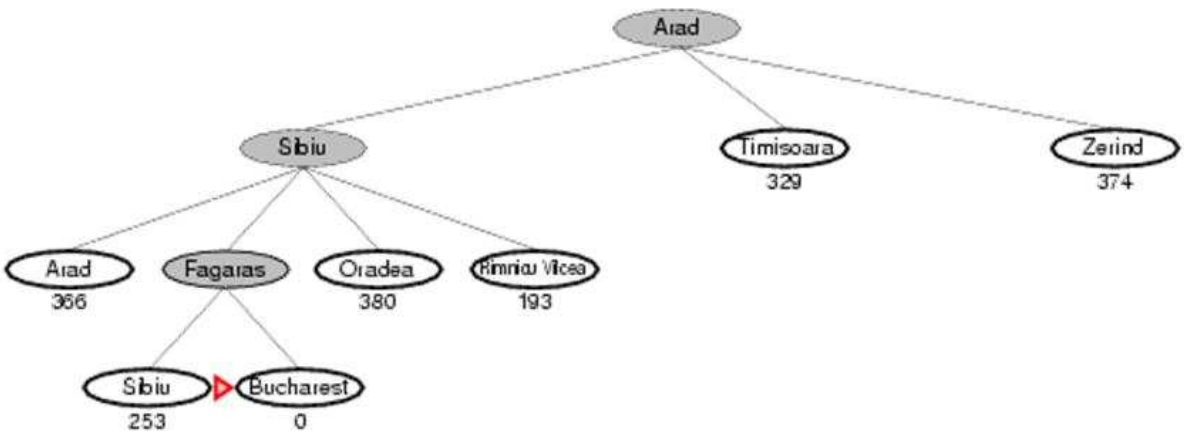


Straight-line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374



[1] Greedy Search



نکته: جستجوی حریصانه ازلحاظ دنبال کردن یک مسیر ویژه در تمام طول راه به طرف هدف، مانند جستجوی اول عمق می‌باشد اما زمانی که به بن بست می‌رسد به سمت بالا بر می‌گردد. این جستجو بهینه و کامل نیست زیرا ممکن است مانند اول عمق در یک مسیر نامتناهی به سمت پایین شروع کند و هرگز برای بررسی بقیه نودها برنگردد. در بدترین وضعیت پیچیدگی زمانی و مکانی برای جستجوی حریصانه $O(b^m)$ است که m حداکثرعمق فضای جستجو است. جستجوی حریصانه تمام گره‌ها را در حافظه نگهداری می‌کند، لذا پیچیدگی فضایی آن مشابه پیچیدگی زمانی آن می‌باشد. البته این پیچیدگی زمانی و مکانی با انتخاب یک تابع هیورستیک خوب به شدت کاهش می‌یابد.

جستجوی A*:

تعریف: جستجوی حریصانه هزینه تخمین زده شده‌ی h(n) به سمت هدف را حداقل می‌کند و هزینه جستجو را کاهش می‌دهد، اما نه کامل است نه بهینه، از طرف دیگر جستجو با هزینه یکسان، هزینه مسیر یعنی g(n) راحداقل می‌کند که هم کامل است هم بهینه، اما در مواردی می‌تواند بی فایده باشد. برای دست یافتن به مزایای هر دو جستجو از ترکیب دو روش تحت عنوان A* استفاده می‌کنیم.

ساختار:

ایده: از گسترش مسیرهایی که تاکنون مشخص شده پر هزینه می‌باشند، اجتناب کن.

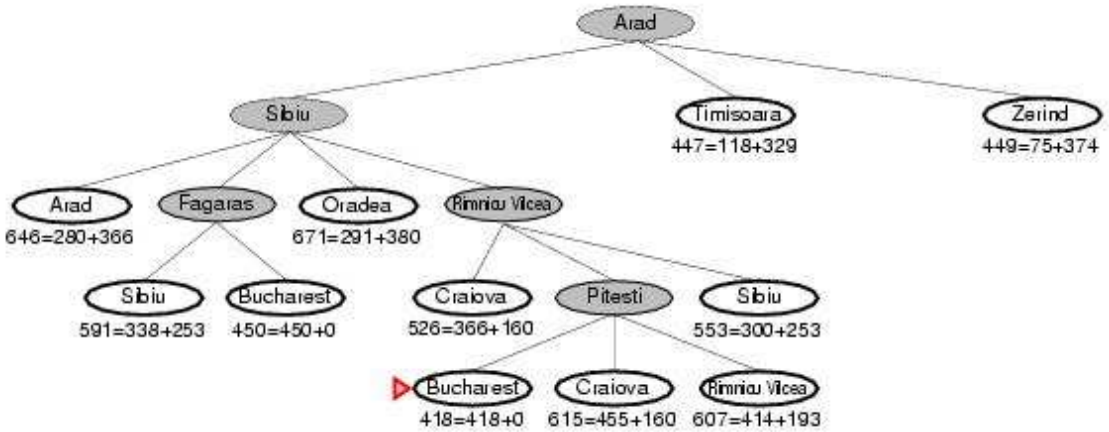
ترکیب مزایای UCS و جستجوی حریصانه:

- جستجوی حریصانه h(n) را حداقل می‌کند، نه کامل و نه بهینه
- جستجوی UCS، هزینه مسیر را حداقل می‌کند، کامل و بهینه است، می‌تواند بسیار زمانبر باشد.

تابع ارزیابی

$$f(n) = g(n)+h(n)$$

- **g(n):** هزینه مسیر پیموده شده تا n.
- **h(n):** هزینه تخمینی ارزانترین مسیر راه حل از n تا هدف.
- **f(n):** هزینه تخمینی ارزانترین راه حل که از n می‌گذرد.



شکل بالا جستجوی A* را برای مساله مسیریابی در کشور رومانی را نشان می‌دهد.

نکته: اگر تابع هیوریستیک شرایط لازم را داشته باشد، A* کامل وبهینه است. A* که از Graph_search استفاده می‌کند به شرطی بهینه است که h(n) سازگار[1] یا یکنواخت باشد و A* که از Tree_search استفاده می‌کند به شرطی بهینه است که h(n) قابل قبول[2] باشد.

هیورستیک قابل قبول:

تابع هیوریستیکی قابل قبول است که هرگز هزینه رسیدن به هدف را بیشتر از هزینه واقعی تخمین نزند. به عبارت دیگر، یک هیورستیک مانند h(n) قابل قبول است اگر برای هر گره n داشته باشیم: $h(n) \leq h^*(n)$ ، که در این رابطه، h*(n) هزینه واقعی برای رسیدن به هدف از گره n می‌باشد. از آنجایی که g(n) هزینه واقعی رسیدن به n است اگر h(n) قابل قبول باشد آنگاه f(n) نیز قابل قبول خواهد بود، یعنی f(n) هرگز بیشتر از هزینه واقعی راه حل از طریق n را تخمین نمی‌زند.

$$\left.\begin{array}{c} f(n) = h(n) + g(n) \\ h(n) \leq h^*(n) \\ f^*(n) = h^*(n) + g^*(n) \end{array}\right\} \xrightarrow{g^*(n)=g(n)} f(n) \leq f^*(n)$$

نکته: به طور خلاصه برای هیورستیک قابل قبول داریم:

- یک هیورستیک مانند h(n) قابل قبول است اگر برای هر گره n داشته باشیم: $h(n) \leq h^*(n)$. که $h^*(n)$ هزینه واقعی برای رسیدن به هدف از گره n می‌باشد.
- یک هیورستیک قابل قبول هرگز هزینه رسیدن را بیش از حد تخمین نمی‌زند، یعنی خوش بینانه است. مثال هیورستیک $h_{SLD}(n)$ (هیچگاه فاصله واقعی را بیش از حد تخمین نمی‌زند.)
- اگر h(n) قابل قبول باشد، A* با استفاده از Tree_search بهینه است.

---

[1] Consistency

[2] admissible

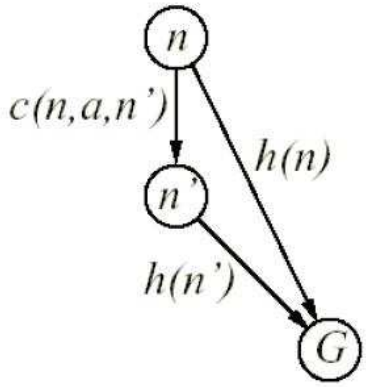
هیورستیک سازگار(یکنوا):

تابع h(n) سازگار است اگر رابطه h(n) < C(n, a, n') + h(n') برقرار باشد که در این رابطه، n' با استفاده از عمل a از حالت n تولید شده است و C(n, a, n') هزینه عمل a را نشان می‌دهد، این نامساوی فرمی از نامساوی مثلثی است که هر ضلع مثلث باید کوچکتر یا مساوی مجموع دو ضلع دیگر باشد.

نکته: به طور خلاصه برای هیورستیک سازگار داریم:

- ❖ یک هیورستیک سازگار است اگر:
- ❖ اگر h سازگار باشد، داریم:

$$\begin{aligned} f(\mathrm{n}') &= \mathrm{g}(\mathrm{n}') + \mathrm{h}(\mathrm{n}') \\ &= g(n) + c(n, a, \mathrm{n}') + \mathrm{h}(\mathrm{n}) \\ &\geq g(n) + h(n) \\ &\geq f(n) \end{aligned}$$



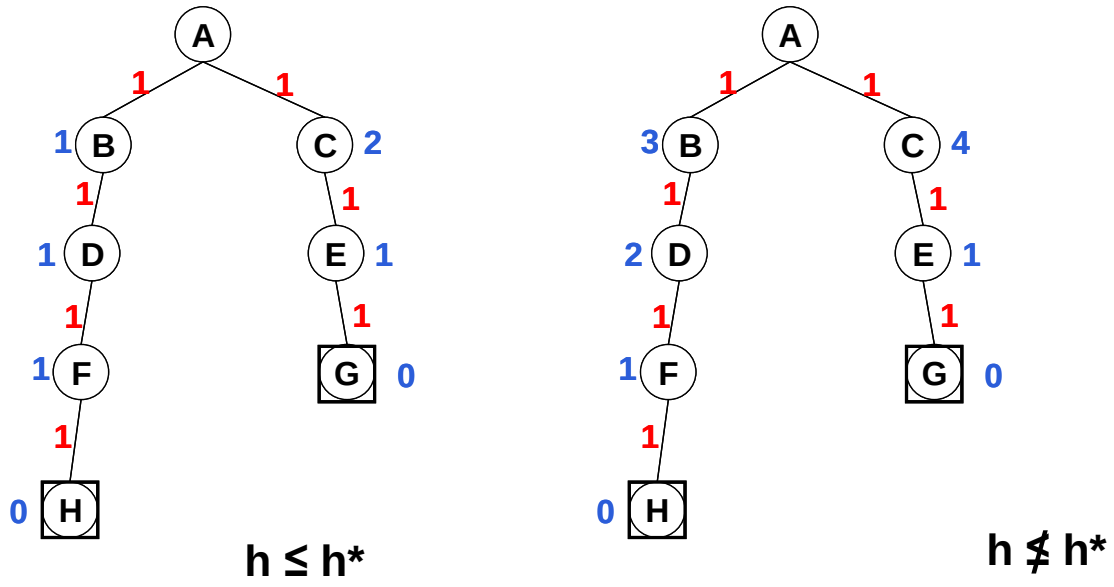
- ❖ یعنی، f(n) در طول هر مسیری غیر کاهشی می‌باشد. (یکنوایی، monotonicity)
- ❖ اگر h(n) سازگار باشد، A* با استفاده از Graph_search بهینه است.

نکته: اگر تابع هیوریستیک یکنوا نباشد می‌توان آن را به نحوی اصلاح نمود تا یکنوا گردد. بدین صورت که هر گره جدیدی که تولید می‌شود باید کنترل شود که هزینه f این گره از هزینه f پدرش کمتر است یا نه؟ اگر کمتر باشد هزینه f پدر به جای فرزند می‌نشیند. این معادله سازی، معادله pathmax نامیده می‌شود که در این معادله f(n') = max{f(n), g(n') + h(n')}، n' فرزند n می‌باشد.

نکته: اگر c* هزینه مسیر راه حل بهینه باشد و h(n) قابل قبول باشد آنگاه:

- ❖ A* تمام گره‌ها با f(n) < c* را گسترش می‌دهد.
- ❖ A* بعضی از گره‌ها با f(n) = c* را گسترش می‌دهد.
- ❖ A* هیچ گره‌ای با f(n) > c* را گسترش نمی‌دهد.

نکته: اگر تابع h(n) قابل قبول باشد آنگاه A* بهینه خواهد بود، یعنی هدف کم هزینه تر را پیدا خواهد کرد. برای درک این مطلب به مثال زیر توجه کنید:



شرایط کامل نبودن A*:

هر الگوریتمی که تمامی گره‌ها را در نواحی بین ریشه وهدف بسط ندهد در معرض خطرگم کردن راه حل بهینه است اما A* برای هر تابع اکتشافی بهینگی دارد. A* در صورتی کامل نخواهد بود که گره‌های نامحدود زیادی با $f(n) < c^*$ وجود داشته باشد. تنها راهی که ممکن است گره‌های نا محدودی وجود داشته باشد این است که:

- گره‌ای با فاکتور انشعاب نامحدود وجود داشته باشد.
- مسیری با هزینه ی مسیر متناهی اما تعداد زیادی گره ی نامحدود در طول مسیر وجود داشته باشد.

شرایط بهینگی A*:

در میان الگوریتم‌های بهینه، A* با هر تابع اکتشافی از نظر بهینگی کاراست، یعنی هیچ الگوریتم دیگری تضمین نمی‌کند که تعداد کمتری گره نسبت به A* گسترش دهد. علت این است که هر الگوریتمی که تمام نودها با $f(n) < c^*$ را گسترش دهد در معرض خطراز دست دادن راه حل بهینه می‌باشد. اگرچه جستجو A* کامل و بهینه و از نظر پیچیدگی کاراست، اما تعداد نودهای موجود در کانتور هدف یک تابع نمایی از طول راه حل است.

شرط غیر نمایی بودن تعداد نودها عبارتنداز: $| h(n) - h^*(n) | <= O(\log(h^*(n)))$

نکته: تنها اشکال عمده‌ی A* زمان نمایی آن نیست از آنجائیکه این الگوریتم، تمام نودهای تولید شده را در حافظه نگهداری می‌کند. بنابراین حافظه بسیار زیادی را مصرف خواهد کرد. به همین دلیل A* برای مسائل با فضای حالت بزرگ کارایی ندارد. الگوریتم‌هایی معرفی خواهیم کرد که بدون ازدست دادن کارایی و بهینگی بر مشکل فضای حالت بزرگ غلبه کنند، مثل: RBFS, IDA*, SMA*.

خصوصیات A*

i. **کامل؟** بله، مگر اینکه تعداد نامحدودی گره با $f \leq f(G)$ وجود داشته باشد. A* در گراف‌های متناهی محلی (با فاکتور انشعاب محدود) است به شرط آنکه هزینه تمام عملگرها مثبت می‌باشد.

نکته: A* در صورتی کامل نیست:

- گره‌ای با فاکتور انشعاب نامحود داشته باشد
- مسیری با هزینه محدود اما ا تعداد گره‌های نامحدود وجود داشته باشد.

ii. **پیچیدگی زمانی؟** نمایی بر حسب خطای نسبی + طول راه حل می‌باشد مگر اینکه خطا در تابع کشف کننده رشدی سریعتر از لگاریتم هزینه مسیر واقعی نداشته باشد، به زبان ریاضی:

$$|h(n) - h^*(n)| \leq O(log\, h^*(n))$$

iii. **پیچیدگی فضا؟** تمام گره‌ها را در حافظه نگه می‌دارد.

iv. **بهینه؟** بله، نمی‌تواند f_{i+1} را گسترش دهد مگر f_i تمام شده باشد.

- A* تمام گره‌ها با $f(n) < c^*$ را گسترش می‌دهد.
- A* بعضی از گره‌ها با $f(n) = c^*$ را گسترش می‌دهد.
- A* هیچ گره‌ای با $f(n) > c^*$ را گسترش نمی‌دهد

کانتور[1]:

یک کانتور با برچسب L، فضای بسته‌ای شامل تمام نودهایی است که مقدار f(n) آن‌ها کمتر یا مساوی L است. کانتورها متحدالمرکزاند. کانتورها در جست وجو با هزینه یکسان در اطراف حالت اولیه، دایره‌ای شکل‌اند و با توابع اکتشافی دقیقتر نواحی به سمت حالت هدف کشیده می‌شوند ودر اطراف مسیر بهینه بیضی شکل می‌شوند.

جست و جوی اکتشافی با حافظه‌ی محدود (IDA*):

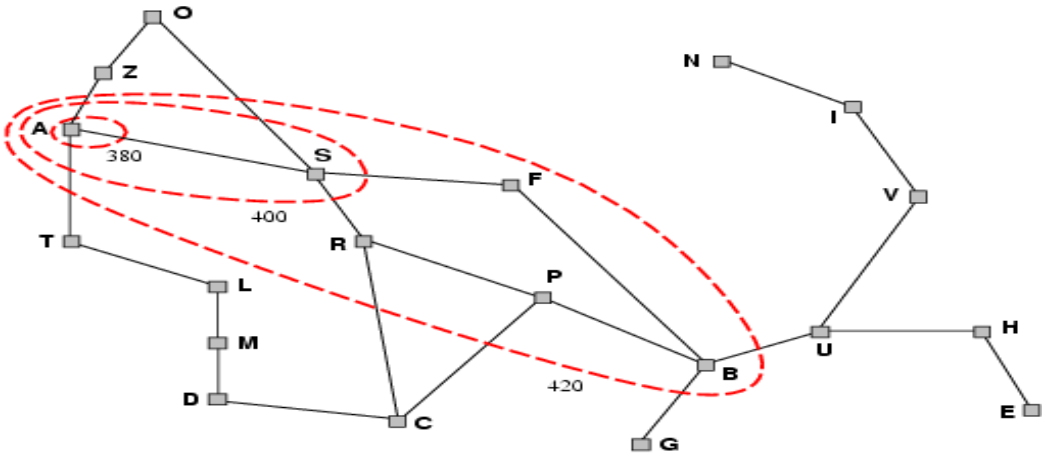
ساده ترین راه برای کاهش حافظه مورد نیاز A*، استفاده از ایده‌ی موجود در جست و جوی عمقی تکرار شونده است. الگوریتم حاصل از اعمال ایده‌ی تکرار شونده‌ی عمقی در A*، IDA* نام دارد. تفاوت IDA* با عمقی تکرار شونده در این است که در این الگوریتم به جای عمق، محدودیت روی هزینه f قرار داده می‌شود. مقدار اولیه‌ی f_limit برابر مقدار f ریشه است. در هر تکرار گره‌هایی که f آنها کمتر از f_limit آن تکرار است، گسترش می-یابند. اگر در این تکرار هدف پیدا شد که کار تمام است، در غیر این صورت کمترین مقدار f گره‌های گسترش نیافته در این تکرار، جایگزین مقدار f_limit می‌شود و دوباره الگوریتم با مقدار جدید f_limit اجرا می‌گردد. این تکرارها ادامه می‌یابد تا زمانی که مقدار f_limit به گونه‌ای باشد که نود هدف نیز برای گسترش انتخاب شود. نود هدف در تکرار با f_limit = c* پیدا می‌شود.

---

[1] contours

خواص IDA*:

این الگوریتم کامل و بهینه است و در هر مرحله فقط گره‌هایی که f آنها کمتر از f_limit است در حافظه نگهداری می‌شوند. بنابراین از نظر پیچیدگی مکانی مانند جست و جوی عمقی، خطی است. البته در بدترین حالت O(bc*/δ) می باشد که δ کمترین هزینه ی اعمال می‌باشد. IDA* برای مسا ئل با هزینه‌ی مرحله‌ای واحد مناسب است و نیاز به نگهداری صف مرتبی از گره‌ها ندارد. اما متأسفانه مانند نسخه‌ی تکرار شونده‌ی جست و جو با هزینه ی یکنواخت منجر به افزایش محاسباتش می‌شود.

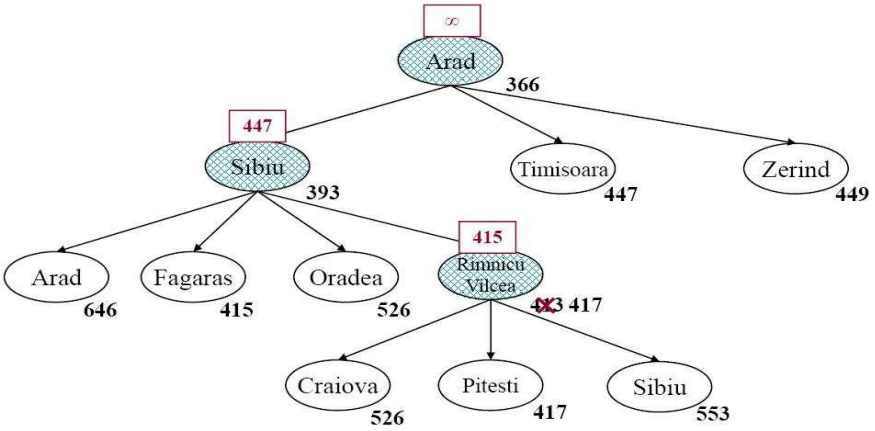


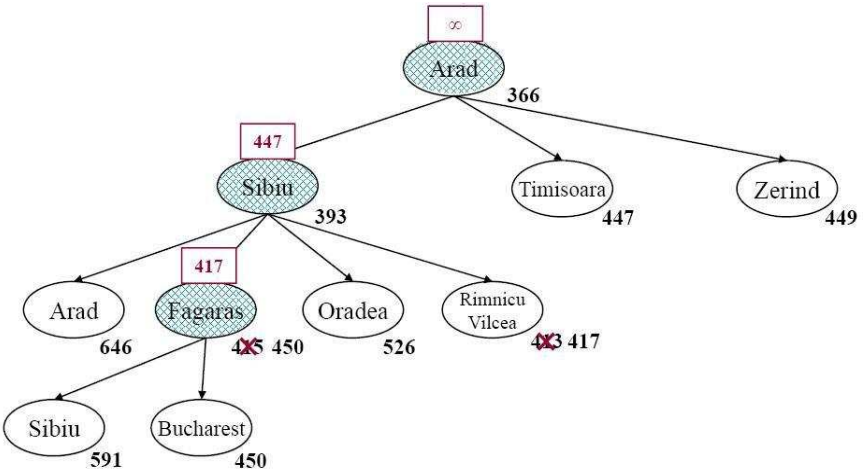
جست و جوی اولین– بهترین بازگشتی (RBFS):

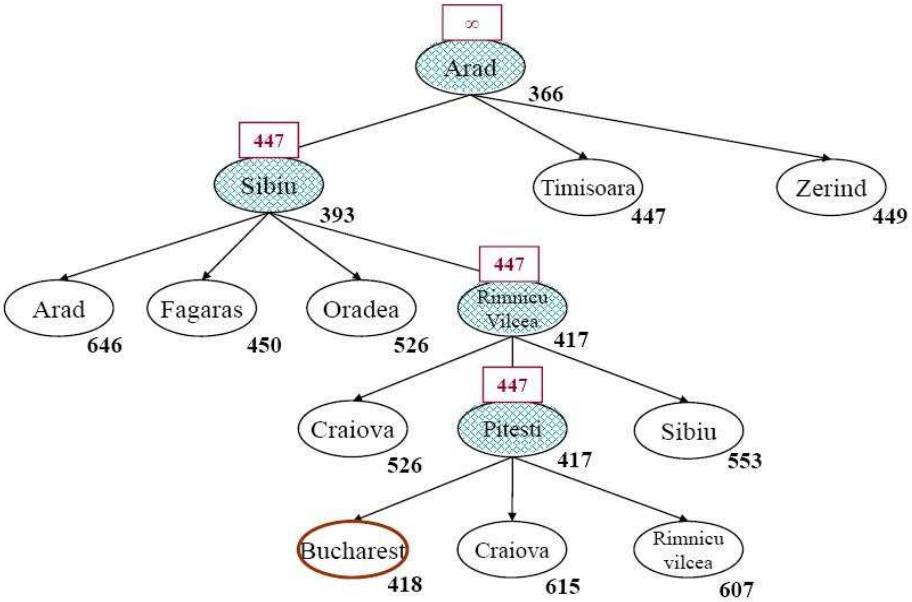
این جست و جو یک الگوریتم بازگشتی ساده است که از جست و جوی اول بهترین تقلید می‌کند، با این تفاوت که پیچیدگی مکانی آن خطی است. ساختار آن شبیه جست و جوی عمقی بازگشتی است. اما به جای آنکه دائماً مسیر فعلی را به سمت پایین ادامه دهد، مقدار f بهترین مسیر جانشین را از طریق اجداد گره فعلی نگهداری می‌کند. اگر f گره فعلی از این حد تجاوز کند، الگوریتم به عقب بر می‌گردد، تا مسیر جانشین را انتخاب نماید. در بازگشت به عقب این الگوریتم مقدار f مربوط به بهترین برگ از زیر درخت فراموش شده را به یاد می‌آورد و می‌تواند تصمیم بگیرد آیا این زیر درخت باید بعداً گسترش یابد یا خیر.

خواص RBFS:

RBFSکمی از IDA* کاراتر است اما این الگوریتم نیز مشابه IDA* گره‌های تکراری تولید می‌کند. اگر تابع اکتشافی h(n) قابل قبول باشد، RBFS همانندA* بهینه است. پیچیدگی مکانی آن تابع خطی O(bd) است. تعیین پیچیدگی زمانی آن دشوار است و به دقت تابع هیوریستیک و میزان تغییر بهترین مسیر در اثر گسترش گره‌ها بستگی دارد. IDA* و RBFS از حافظه اندکی استفاده می‌کنند که این مسأله می‌تواند به آنها آسیب برساند. IDA*در هر تکرار فقط یک عدد را نگهداری می‌کند که هزینه فعلی f است، ولی RBFS اطلاعات بیشتری را نسبت به IDA* درحافظه نگهداری می‌کند. اگر حافظه‌ی زیادتری مهیا داشته باشد، این دو الگوریتم راهی برای استفاده از آن ندارند، بنابراین بهتر است الگوریتمی داشته باشیم که از کل حافظه موجود استفاده کند (مانند الگوریتم SMA*). شکل زیر جستجوی RBFS را روی مسئله مسیریابی در رومانی را نشان می‌دهد.







الگوریتم*SMA:

*SMA مانند *A بهترین برگ را گسترش می‌دهد تا حافظه پر شود. با پر شدن حافظه، بدون از بین بردن گره‌های قبلی نمی‌توان گره جدیدی اضافه کرد. زمانی که نیاز به تولید فرزند باشد و حافظه‌ای در اختیار الگوریتم نباشد، نیاز به نوشتن مجدد بر روی حافظه است. برای انجام این امر، *SMA، یک گره را حذف می‌کند و فرزند جدید از حافظه‌ی

آن استفاده خواهد کرد. گره‌هایی که به این طریق حذف می‌شوند، گره‌های فراموش شده[۱] نام دارند. در این حالت هایی برای حذف شدن انتخاب می‌شوند که هزینه ی آنها بالاست. برای جلوگیری از جست وجوی مجدد زیردرخت-هایی که از حافظه حذف شده‌اند، در گره‌ی پدر آن‌ها اطلاعاتی درباره‌ی کیفیت بهترین مسیر، درزیردرخت فراموش شده نگهداری می‌شود. بنابراین زمانی این زیردرخت‌ها دوباره تولید خواهند شد که ثابت شود سایر مسیرهای دیگر بدتر از مسیر فراموش شده هستند.

خواص SMA*

i. می‌تواند از تمام حافظه قابل دسترس استفاده ببرد.

ii. از حالات تکراری تا جایی که حافظه اجازه می‌دهد، جلوگیری می‌کند.

iii. این الگوریتم کامل است به شرط آنکه حافظه کافی، برای ذخیره ی کم عمق ترین مسیر راه حل وجود داشته باشد.

iv. این الگوریتم بهینه است به شرط آنکه حافظه کافی برای ذخیره ی کم هزینه ترین مسیر (هدف با کمترین f) وجود داشته باشد.

v. زمانی که حافظه‌ی موجود برای جست وجوی درخت کافی باشد، جست وجوی SMA* بهینه‌ی کاراست.

نکته: SMA* بهترین الگوریتم همه منظوره برای یافتن راه حل‌های بهینه است. اگر مقدار f تمام برگ‌ها یکسان باشد، باید الگوریتم یک گره را هم برای گسترش و هم برای حذف انتخاب کند. SMA* این مسأله را با گسترش بهترین برگ جدید و حذف بهترین برگ قدیمی حل می‌کند. گاهی اوقات ممکن است SMA* مجبور شود دائماً بین مجموعه‌ای از مسیرهای حل کاندید، تغییر وضع دهد، در حالی که بخش کوچکی از هر کدام در حافظه جای می‌شود.

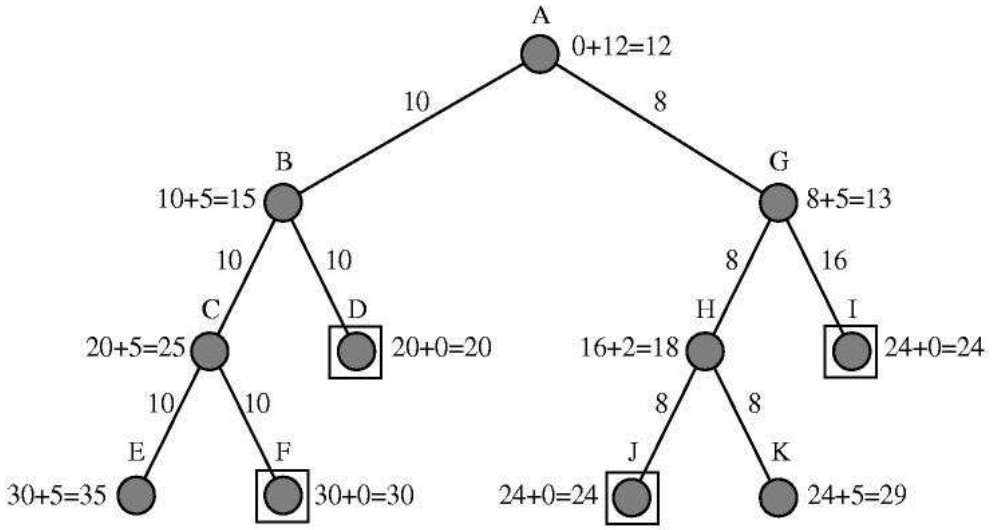
نکته: ترتیب الگوریتم‌های حافظه محدود شده از نظر پیچیدگی حافظه:

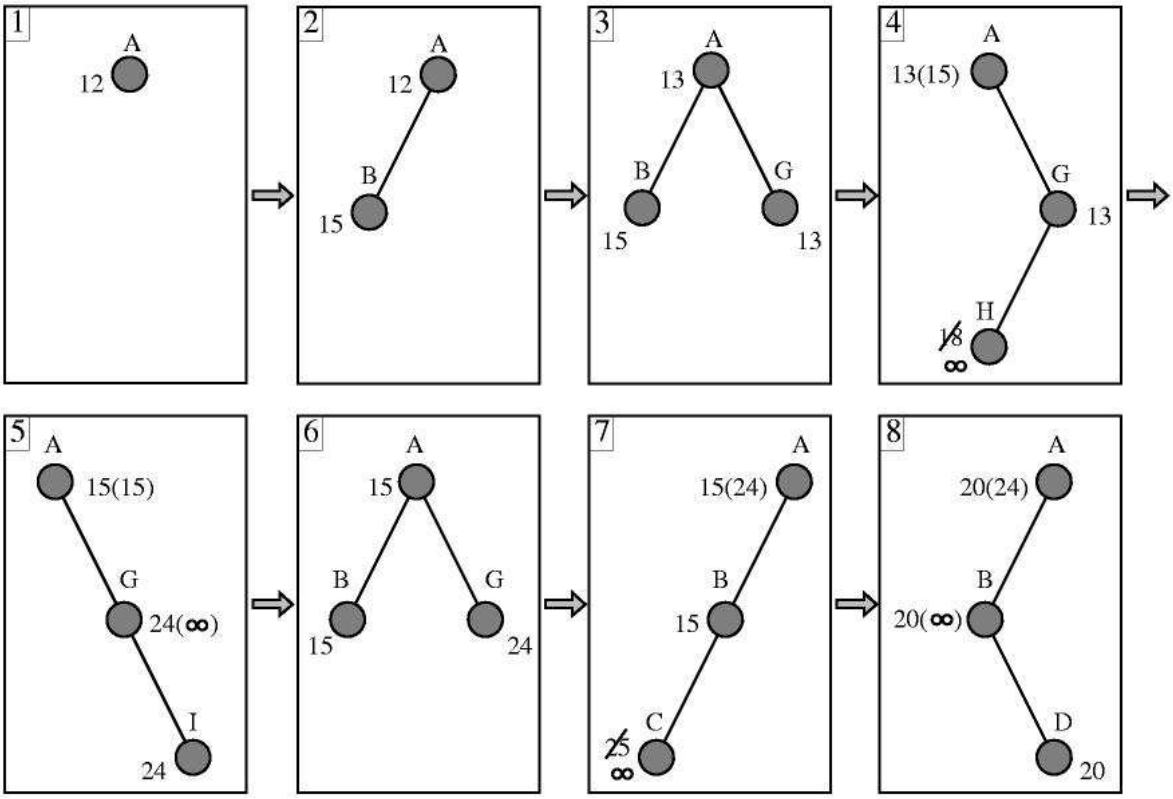
IDA* < SMA* < RBFS

---

[۱] Forgotten node

مثال: شکل زیر مثالی از الگوریتم *SMA با ۳ خانه حافظه را نشان می‌دهد





توابع هیورستیک:

در این بخش، توابع اکتشافی معمای ۸ بررسی می‌شود تا با ماهیت این توابع آشنا شوید. معمای ۸ یکی از اولین مسائل جستجوی اکتشافی بود. میانگین هزینه راه حل برای یک نمونه تصادفی معمای۸، ۲۲ مرحله می‌باشدکه متوسط فاکتور انشعاب برابر ۳ می‌باشد. بنابراین جستجوی ۲۲ مرحله‌ای تقریباً b^d که در اینجا برابر $۳^{۲۲}$ حالت دارد که تعداد حالات بسیار زیادی است و با انتخاب یک تابع اکتشافی مناسب می‌توان مراحل جستجو را کاهش داد. اگر بخواهیم با *A

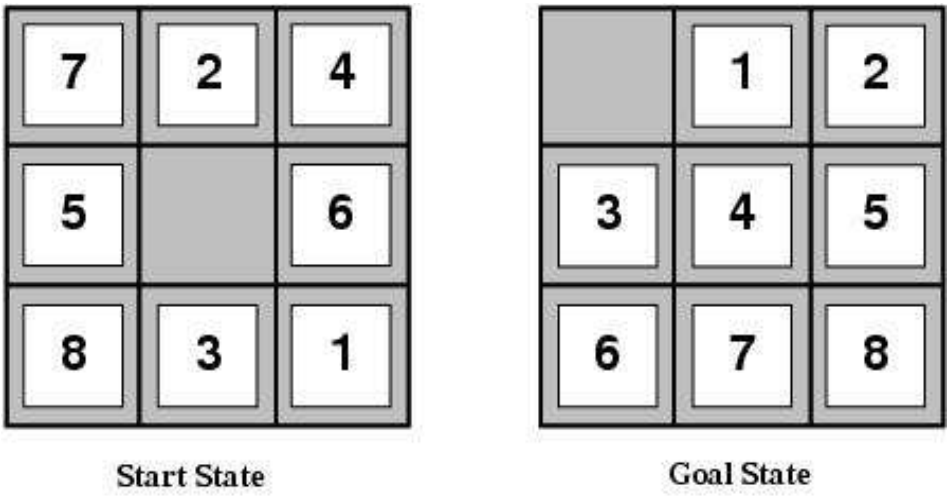
کوتاهترین راه حل‌ها را بیابیم، به تابعی اکتشافی نیار داریم که تعداد مراحل را اضافه تخمین نزند یعنی قابل قبول باشد. دو تابع هیورستیک رایج برای معمای ۸ عبارتند از:

- **h1 =** تعداد خانه‌هایی که در مکان‌های نادرست قرار دارند. این تابع،هیورستیک قابل قبولی است زیرا بدیهی است هر خانه که در جای نامناسبی قرار دارد، حداقل یکبار باید جابجا شود.
- **h2 =** مجموع فواصل خانه‌ها از مکان‌های صحیح آنها. آز آنجائیکه خانه‌ها در امتداد قطر جابجا نمی‌شوند، فاصله‌ای که محاسبه می‌شود، مجموع فواصل افقی و عمودی است. این فاصله گاهی فاصله بلوک شهر یا فاصله مانهاتان[1] نامیده می‌شود. این تابع نیز قابل قبول است، زیرا با هر جابجایی یک خانه، یک مرحله به هدف نزدیک تر می‌شود.

مثال: شکل زیر مثالی از معمای ۸ را نشان می‌دهد که مقدار توابع هیورستیک h1 و h2 برای آن‌ها محاسبه شده است:

h1=8

h2=3+1+2+2+2+3+3+2=18

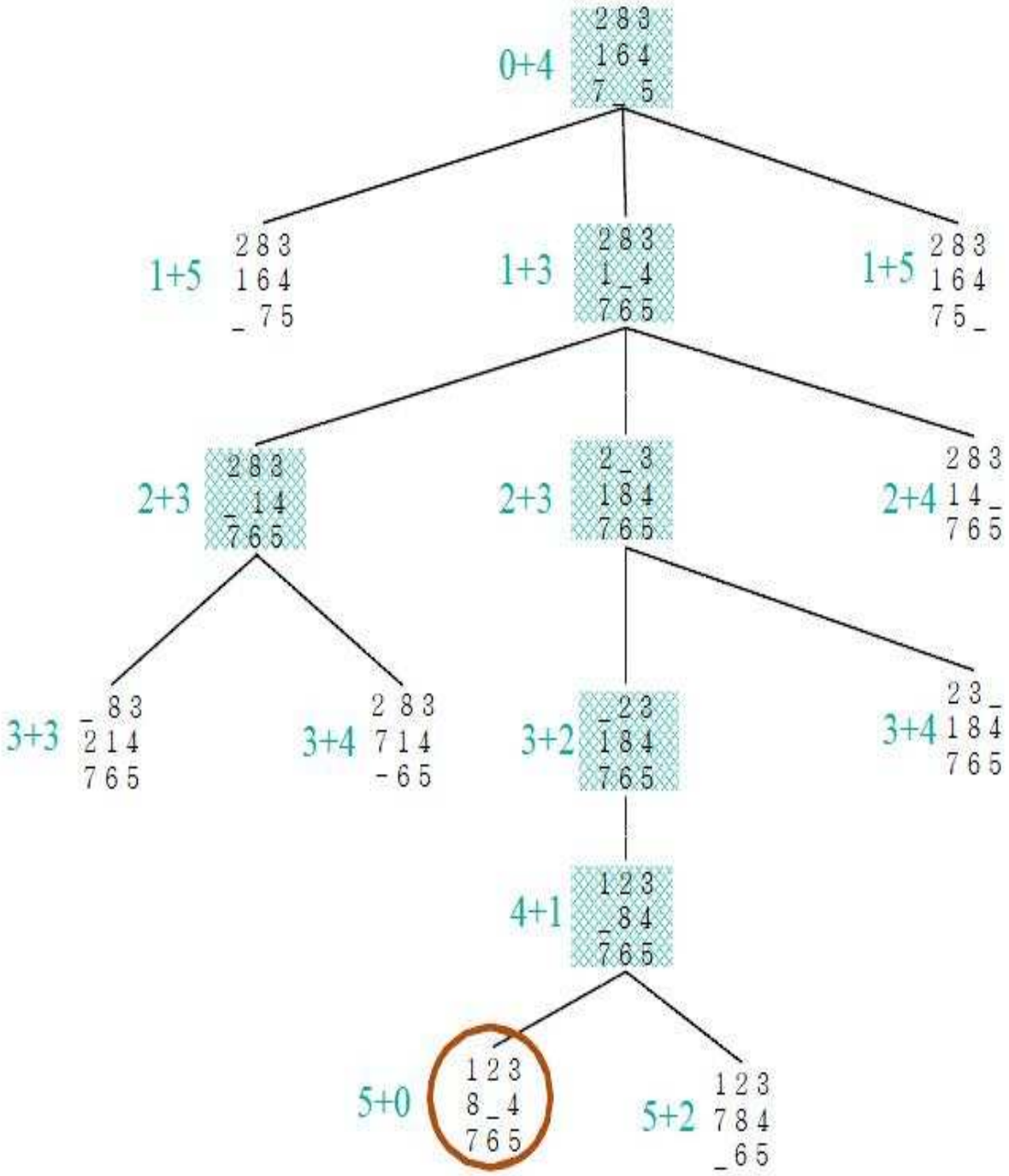


همانطور که انتظار داریم هیچ کدام از این برآوردها، هزینه واقعی راه حل نیست، بلکه هزینه واقعی ۲۶ است.

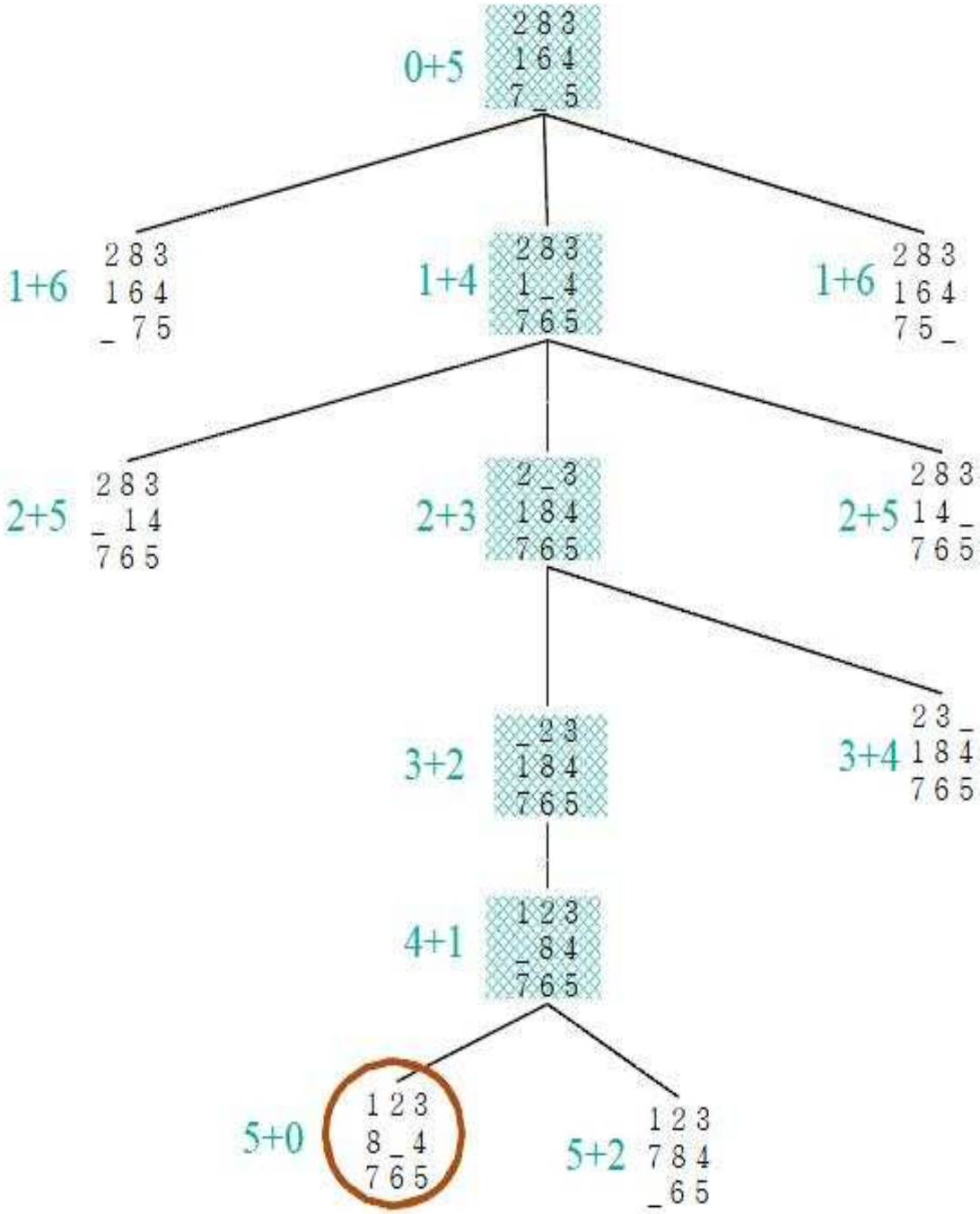
در شکل صفحه بعد نمونه‌ای از مسئله معمای ۸ یکبار با تابع هیورستیک h1 و یکبار با h2 حل شده است.

[1] Manhattan

مثال: جستجوی A* با h_1



مثال: جستجوی A* با h_2



اثر کیفیت تابع هیورستیک بر کارایی:

یک روش تعیین کیفیت تابع هیورستیک، فاکتور انشعاب مؤثر[1] است که با b* نشان داده می‌شود. اگر تعداد گره‌های گسترش یافته توسط روال جستجو N باشد و عمق راه حل d باشد، آنگاه b* به صورت زیر محاسبه می شود:

$$N + 1 = 1 + b^* + (b^*)^2 + \cdots + (b^*)^d$$

[1] Effective Branch Factor

مثال: اگر A* راه حلی را در عمق ۵ با استفاده از ۵۲ گره پیدا کند، فاکتور انشعاب موثر را پیدا کنید.

پاسخ:

$$53 = 1 + b^* + (b^*)^2 + \cdots + (b^*)^5 \rightarrow b^* = 1.92$$

نکته: بنابراین اندازه گیری b* روی مجموعه‌ای کوچک می‌تواند مفید بودن یا نبودن یک تابع هیوریستیک را در حالت کلی مشخص کند. مقدار b* برای تابع اکتشافی خوب طراحی شده، نزدیک به 1 است. فاکتور انشعاب مؤثر برای مسائل بزرگ وسخت معمولاً ثابت است. بنابراین در یک هیوریستیک هر چه فاکتور انشعاب مؤثر به 1 نزدیکتر باشد آن هیوریستیک کیفیت کشف کنندگی بیشتری دارد.

تسلط[1]:

اگر به ازای هر n، h2(n) >= h1(n) (با فرض اینکه h1, h2 قابل قبول باشند) آنگاه h2 بر h1 تسلط دارد; یعنی کاراتر می‌باشد و برای جست و جو مناسب تر است. تعداد گره‌هایی که A* با به کارگیری h2 گسترش می‌دهد هرگز بیشتر از تعداد گره‌هایی که A* با بکارگیری h1 گسترش می‌دهد نخواهد بود، یعنی هر گره‌ای که به وسیله‌ی A* با h2 گسترش می‌یابد قطعا به وسیله ی A* با h1 نیز گسترش می‌یابد. علاوه بر این A* با h1 ممکن است منجر به گسترش گره‌های دیگری بشود. لذا همیشه بهتراست از تابع اکتشافی قابل قبول با مقادیر بزرگتر استفاده کرد؛ به شرطی که زمان محاسبه‌ی مقدار آن تابع هیوریستیک در هر نود خیلی زیاد نباشد.

مقایسه جستجوهای A* (h1)، A*(h2)، IDS برای حل صد نمونه مسئله تصادفی معمای ۸:

برای تست توابع اکتشافی h2 و h1 صد نمونه مسئله تصادفی پازل ۸ با عمق‌های مختلف از ۲ تا ۲۴ تولید شده است. و به کمک جست و جوی عمقی تکرار شونده و جست وجوی درختی A* با استفاده از h1, h2 حل شده‌اند. شکل زیر مقایسه‌ای بین هزینه‌ی جست وجو یعنی تعداد گره‌های گسترش یافته و فاکتور انشعاب مؤثر در این جستجوها را نشان می‌دهد. با توجه به نتایج حاصل از این جدول، می‌توان نتیجه گرفت که h2 بهتر از h1 است و A* آگاهانه خیلی بهتر از IDS ناآگاهانه است.

[1] Dominance

مقایسه بین هزینه جستجو و فاکتور انشعاب موثر

	Search Cost			Effective Branching Factor		
d	IDS	$A^*(h_1)$	$A^*(h_2)$	IDS	$A^*(h_1)$	$A^*(h_2)$
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45
6	680	20	18	2.73	1.34	1.30
8	6,384	39	25	2.80	1.33	1.24
10	47,127	93	39	2.79	1.38	1.22
12	364,404	227	73	2.78	1.42	1.24
14	3,473,941	539	113	2.83	1.44	1.23
16	-	1,301	211	-	1.45	1.25
18	-	3,056	363	-	1.46	1.26
20	-	7,276	679	-	1.47	1.27
22	-	18,094	1,219	-	1.48	1.28
24	-	39,135	1,641	-	1.48	1.26

مسا ئل تعدیل شده[1]:

مسا ئل تعدیل شده مسا ئلی هستند که بعضی از محدودیت‌های عملگرهای آن حذف شده باشد، به عبارت دیگر مسئله ساده شده باشد. این مسائل با محدودیت کمتری بر روی عملگرها مواجه هستند. اگر قوانین معمای ۸ طوری تغییر کند که هر خانه بتواند در هر جایی قرار گیرد، به جای اینکه فقط به خانه ی همجوار خالی برود، آنگاه h1 کوتاه ترین راه حل را پیدا می‌کند. و اگر هر خانه بتواند به یک مربع در هر جهت منتقل شود حتی در مربع اشغال شده، آنگاه h2 نیزکوتاهترین راه حل را پیدا می‌کند.

معمای ۸: اصل معمای۸ را در نظر بگیرید:

« خانه A می‌تواند به خانه B برود، اگر A هم جوار افقی یا عمودی B باشد و B خالی باشد. »

با حذف یک یا دو شرط از اصل فوق، می توان سه مسئله راحت تولید کرد:

- خانه A می‌تواند به خانه‌ی B برود، اگرA هم جوار B باشد.
- خانه A می‌تواند به خانه‌ی B برود، اگرB خالی باشد.
- خانه A می‌تواند به خانه B منتقل شود.

نکته: معمولاً حل یک مسئله تعدیل شده یک تخمین یا تابع کشف کننده‌ی خوبی برای حل مسئله اصلی خواهد بود. هزینه راه حل بهینه یک مساله راحت، بیشتر از هزینه راه حل بهینه در مسئله اصلی نخواهد بود. برنامه‌ای به نام ABSOLVER با استفاده از روش مسئله راحت و تکنیک‌هایی دیگر، توابع اکتشافی را به طور خودکار تولید می-کند. این برنامه توابع هیوریستیک جدیدی را برای معمای ۸ تولید کرد، که از توابع قبلی مفیدتر است. این برنامه همچنین، اولین تابع اکتشافی مفید را برای معمای روبیک پیدا کرده است.

[1] Relaxed problem

نکته: یکی از مشکلات تولید توابع اکتشافی جدید، یافتن بهترین تابع هیوریستیک است، اگر مجموعه‌ای از توابع اکتشافی قابل قبول $h_1, h_2, \ldots, h_n$ را داشته باشیم و هیچکدام بر دیگری تسلط نداشته باشند، آنگاه بهترین تابع هیوریستیک به صورت روبرو است: $h(n) = \max(h_1(n), \ldots, h_k(n))$

این تابع اکتشافی مرکب از توابع اکتشافی $h_1, h_2, \ldots, h_n$ با کیفیت تر می‌باشند. از آنجائیکه همه‌ی توابع قابل قبول هستند h نیز قابل قبول است. در این حالت h بر تمام توابع مذکور تسلط دارد وکاراتر است. روش دیگر برای ابداع یک کشف کننده‌ی خوب استفاده از اطلاعات آماری است، که این اطلاعات می‌تواند توسط اجرای جستجو روی تعدادی از مسائل، جمع آوری شود. مانند ۱۰۰ مسئله که ساختار معمای۸ را داشته باشند و به طور تصادفی انتخاب شوند و در نهایت روی نتایج اجرای نمونه‌های مختلف تحلیل‌های آماری مثل میانگین، میانه، واریانس و... گرفته شود. ایده‌ی بانک اطلاعاتی الگو، ذخیره‌ی هزینه‌ای واقعی راه حل‌های هر ترکیب ممکن از زیر مسئله را ممکن می‌سازد. گاهی اوقات می‌توان برای ابداع یک تابع اکتشافی جدید، از ترکیب خطی چند تابع استفاده کرد. به عنوان مثال تابع خطی $H(n) = C_1 X_1(n) + C_2 X_2(n)$ که در آن $X_1(n)$ تعداد خانه‌ها با مکان نادرست و $X_2(n)$ تعداد جفت‌هایی از خانه‌های همجوار که درحالت هدف نیز همجوار یکدیگرند. مقادیر ثابت C_1, C_2 برای تطابق با هزینه‌ی واقعی تنظیم خواهند شد.

الگوریتم‌های جست وجوی محلی:

تاکنون الگوریتم‌های جستجویی که بررسی کردیم، طوری طراحی شده بودند که فضای جست وجو را به طور سیستماتیک (قدم به قدم) مورد بررسی قرار می‌دادند و در آنها تا رسیدن به هدف یک یا چند مسیر نگه داری می‌شد و مسیر رسیدن به هدف راه حل مسئله را تشکیل می‌داد.

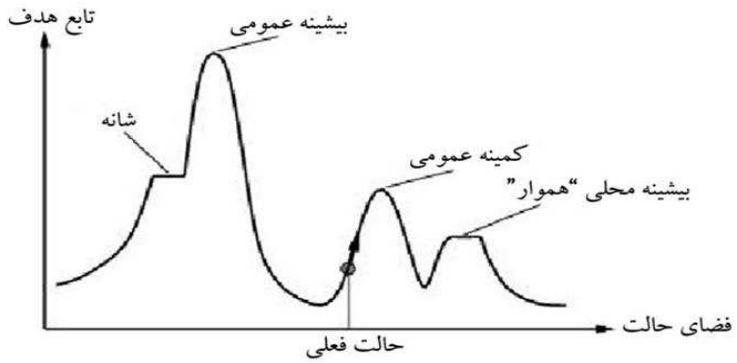
نکته: در الگوریتم‌های جست و جوی محلی مسیر رسیدن به هدف مهم نیست. به عنوان مثال در مسئله‌ی ۸ وزیر پیکر بندی نهایی مهم است و ترتیب اضافه شدن وزرا مهم نمی‌باشد. این نوع مسائل شامل کاربردهای زیادی از قبیل: طراحی مدارهای مجتمع VLSI، زمان بندی کارها، چیدمان دستگاه‌ها در کف کارخانه و غیره می‌باشد. اگر چه الگوریتم‌های جست وجوی محلی سیستمانیک نیستند، ولی دو مزیت عمده دارند:

- اینکه از حافظه کمی استفاده می کنند.
- در فضاهای حالت بزرگ ونامتناهی که الگوریتم‌های سیستماتیک مناسب نیستند، راه حل‌هایی خوبی پیدا می-کنند.

خواص الگوریتم‌های جست وجوی محلی:

جست وجوی محلی علاوه بر یافتن هدف، برای حل مسائل بهینه سازی نیز مفید هستند. در این مسا ئل مقصود یافتن بهترین حالت بر اساس یک تابع هدف[1] می‌باشد. برای درک جست جوی محلی به سطح فضای حالت زیر توجه کنید:

[1] Objective function



سطح فضای حالت فوق، هم دارای مکان (که با حالت مشخص می‌شود) و هم دارای ارزیابی (که با مقدار تابع هدف تعریف می‌شود) می‌باشد. اگر ارزیابی متناظر با هزینه باشد هدف رسیدن به پایین ترین دره یا مینیمم مطلق است. اگر ارتفاع، متناظر با تابع هدف باشد، هدف یافتن بلندترین قله یا ماکزیمم مطلق است. الگوریتم‌های جستجوی محلی این سطح را کاوش می‌کنند. یک الگوریتم جستجوی محلی کامل در صورت وجود هدف آن را می‌یابد و یک الگوریتم جستجوی محلی بهینه، همواره ماکزیمم یا مینیمم مطلق را می‌یابد.

ایده الگوریتم‌های جستجوی محلی:

ایده کلی الگوریتم‌های جست وجوی محلی این است که وقتی از یک حالت قبلی به حالت فعلی برسیم، دیگر حالت قبلی را فراموش می‌کنیم. لذا در این مسائل درخت نداریم بلکه تنها یک وضعیت فعلی داریم که خودش شامل تمام اطلاعات مورد نیاز مسئله است. در این الگو ریتم‌ها از یک حالت قانونی شروع کرده و سپس با انجام تغییراتی در آن سعی در اصلاح کیفیت آن ساختار داریم.

انواع الگوریتم‌های جستجوی محلی:

الگوریتم‌های جستجوی محلی که در این بخش بررسی خواهیم کرد عبارتند از:

۱- تپه نوردی ۲- شبیه سازی حرارت ۳- جستجوی پرتو محلی ۴- الگوریتم ژنتیک

I. جستجوی تپه نوردی[1]:

این الگوریتم درجهت افزایش مقدارارزش تابع ارزیاب حرکت می‌کند، یعنی به طرف بالای تپه. وقتی به قله رسید، یعنی جایی که هیچ همسایه‌ای ازآن بلندتر نیست، خاتمه می‌یابد. الگوریتم تپه نوردی فقط به همسایگان حالت جاری نگاه می‌کند. تپه نوردی گاهی جستجوی محلی حریصانه نیز نامیده می‌شود، زیرا بدون اینکه به آینده فکر کند بهترین همسایه راانتخاب می‌کند. اگردرالگوریتم تپه نوردی به دنبال مینیمم محلی یا کاهش هزینه باشیم الگوریتم، کاهش گرادیان نامیده می‌شود.

[1] Hill climbing

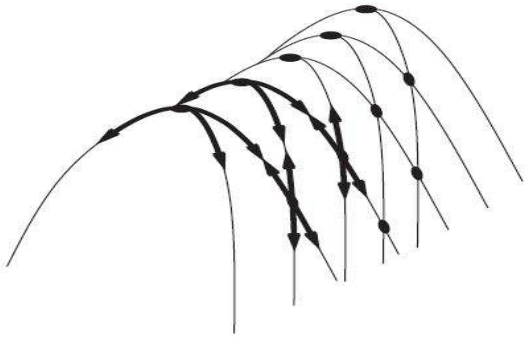
```
function HILL-CLIMBING( problem) returns a state that is a local maximum
   inputs: problem, a problem
   local variables: current, a node
                    neighbor, a node

   current ← MAKE-NODE(INITIAL-STATE[problem])
   loop do
       neighbor ← a highest-valued successor of current
       if VALUE[neighbor] ≤ VALUE[current] then return STATE[current]
       current ← neighbor
```

دلایل شکست الگوریتم تپه نوردی:

الگوریتم تپه نوردی به دلایل زیرمتوقف می‌شود:

- **ماکزیمم محلی[۱]:** قله‌ای است که ازهمه همسایگانش بلندترمی‌باشد، اما از ماکزیمم مطلق کوتاهتر است. (درمسئله ۸ وزیر، ۸ وزیر بدون برخورد ماکزیمم مطلق است ولی ۷ وزیر بدون برخورد و یکی برخوردی ماکزیمم محلی است.)
- **دماغه‌ها[۲]:** دماغه (تیغه) منجر به رشته‌ای ازماکزیمم‌های محلی می‌شود، که عبور از آن توسط الگوریتم‌های حریصانه کار دشواری است. (در مساله ۸ وزیر، چندحالت با۷ وزیر بدون برخورد و یکی برخوردی ایجاد می-شود.)



- **فلات[۳]:** ناحیه از سطح فضای حالت است که در آن مقدار تابع ارزیابی یکسان است. دونوع فلات وجود دارد: ماکزیمم محلی صاف که در آن هیچ راهی به سمت بالا وجود ندارد و شانه[٤] که از طریق آن می‌توان به بالاتر رفت و پیشروی کرد. جستجوی تپه نوردی ممکن است نتواندر فلات راهش را پیدا کند.

---

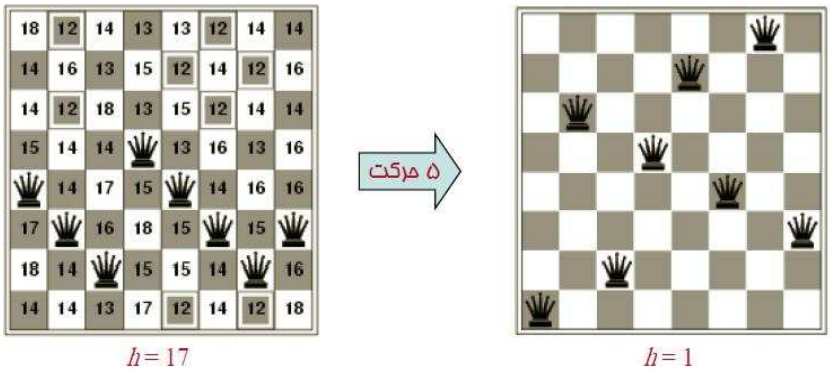
[۱] Local maxima

[۲] Ridge

[۳] plateux

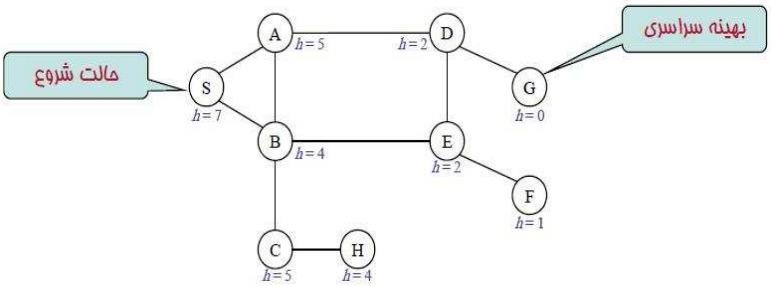
[٤] shoulder

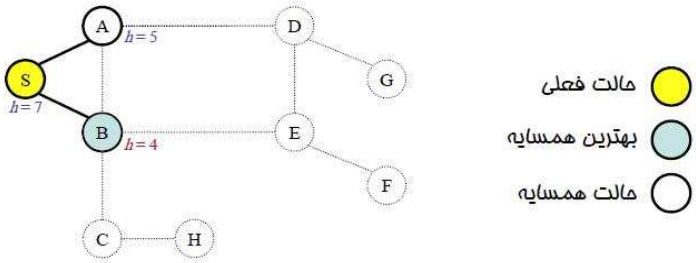
مثال: جستجوی تپه نوردی در ۸ وزیر

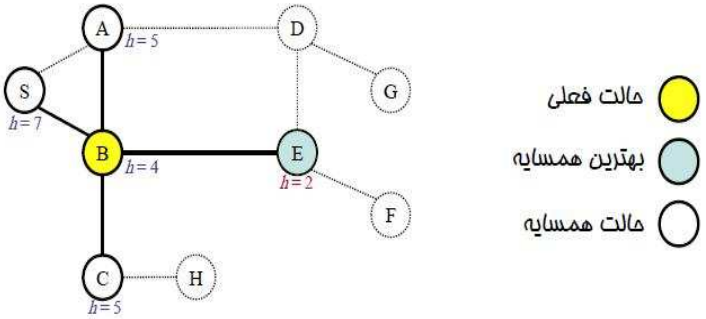


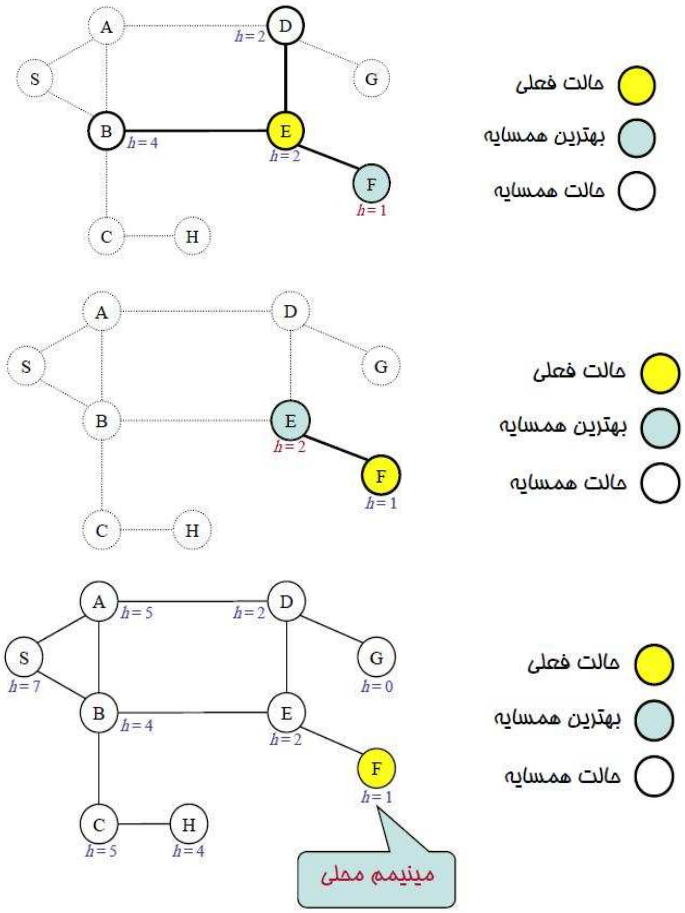
h=تعداد جفت وزیرهایی که بطور مستقیم و یا بطور غیر مستقیم یکدیگر را تهدید می‌کنند.

مثال: الگوریتم تپه نوردی معمولی در شکل زیر ارائه شده است.







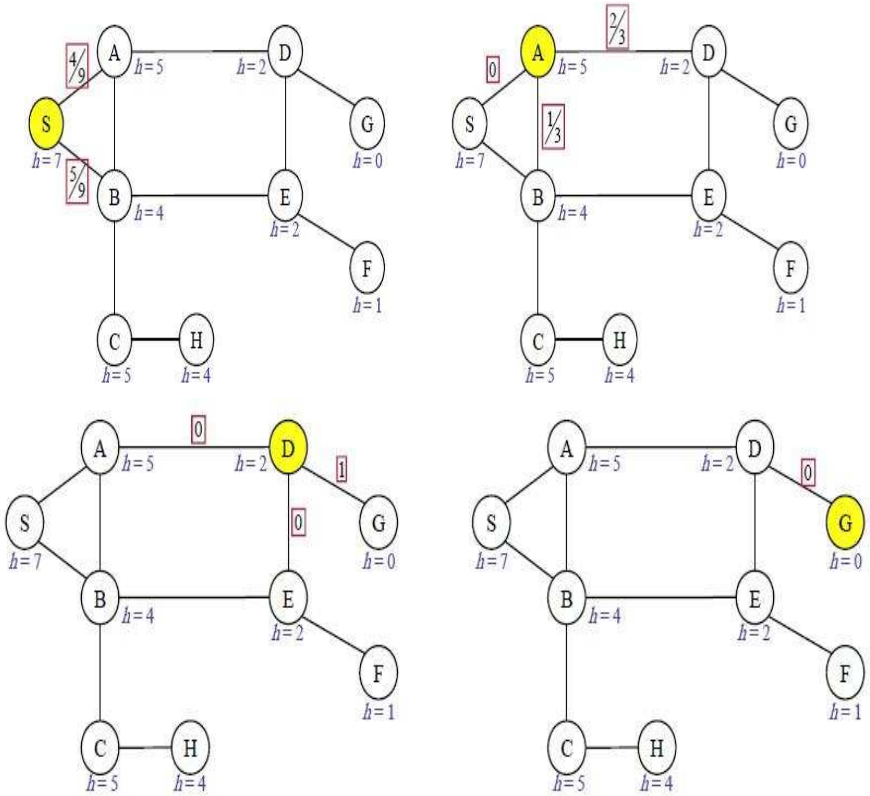


فرم‌های مختلفی از تپه نوردی مطرح شده است. الگوریتم تپه نوردی که مطرح شد تپه نوردی با تندترین شیب نیز نامیده می‌شود. این الگوریتم بسیار سریع عمل می‌کند. درمواردی که موفق است به طورمتوسط بعد از ۴ مرحله به جواب می‌رسد و وقتی ناموفق است بعداز ۳ مرحله متوقف می‌شود. و برای فضای حالتی با ۱۷ میلیون حالت مناسب است. انواع دیگر تپه نوردی عبارتند از:

i. تپه نوردی تصادفی[1]:

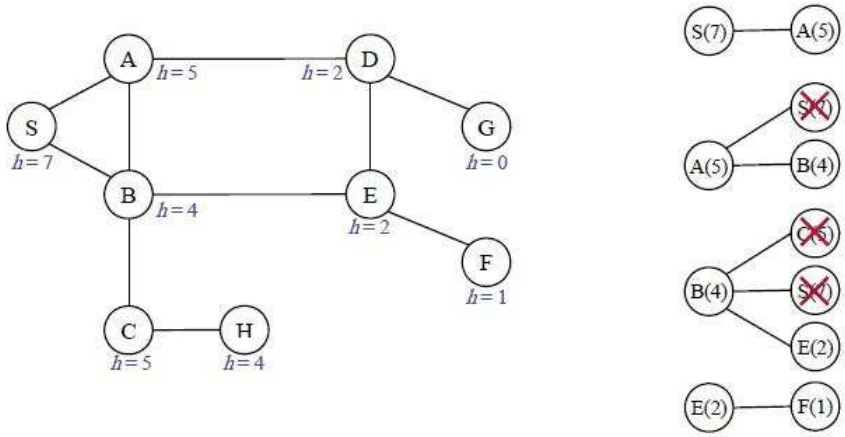
درهرمرحله از میان حرکت‌های رو به بالای تپه یکی را به طور تصادفی انتخاب می‌کند. احتمال انتخاب با میزان شیب حرکت رو به بالا تغییر می‌کند. این روش نسبت به روش تپه نوردی با تندترین شیب آهسته تر همگرا می‌شود، اما در بعضی فضاهای حالت، راه حل بهتری را می‌یابد.

[1] Stochastic hill climbing



ii. تپه نوردی اولین انتخاب [1]:

تپه نوردی تصادفی رابه این صورت پیاده سازی می‌کند که بطور تصادفی مابعدها را تولید می‌کند تا بالاخره یکی ازآنها بهتر از حالت جاری باشد. این روش برای مسائلی که دارای مابعدهای زیادی هستند مناسب است.



نکته: الگوریتم‌های تپه نوردی اشاره شده فوق (تصادفی – اولین انتخاب – تندترین شیب) کامل نیستند، چون در ماکزیمم محلی متوقف می‌شوند ودر ماکزیمم مطلق متوقف نمی‌شوند.

iii. تپه نوردی با شروع مجدد تصادفی[2]:

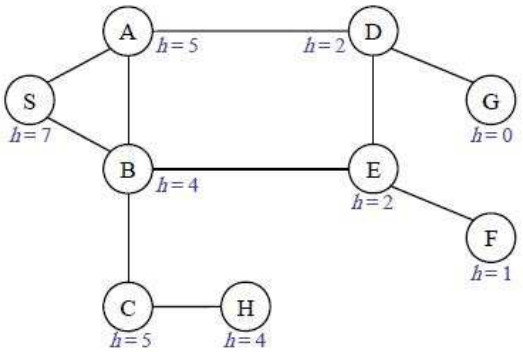
این روش معتقد است اگر موفق نشدید دوباره سعی کنید. این روش یک سری ازجستجوهای تپه نوردی راشامل می-شود که ازحالت اولیه تصادفی شروع و با رسیدن به حالت هدف متوقف می‌شود. با احتمال نزدیک به یک این روش

---

[1] First-choice hill climbing

[2] Random restart hill climbing

کامل است، زیراسرانجام حالت هدف به عنوان حالت اولیه تولید می‌شود. روش تپه نوردی باشروع تصادفی مجدد برای مسئله ۸ – وزیر، مناسبترین می‌باشد. موفقیت تپه نوردی به شکل سطح حالت بستگی دارد. اگرتعداد ماکزیمم‌های محلی و فلات‌ها کم باشد تپه نوردی باشروع مجدد تصادفی سریعاً راه حل خوبی پیدا می‌کند.



S → B → E → F ✗

C → B → E → F ✗

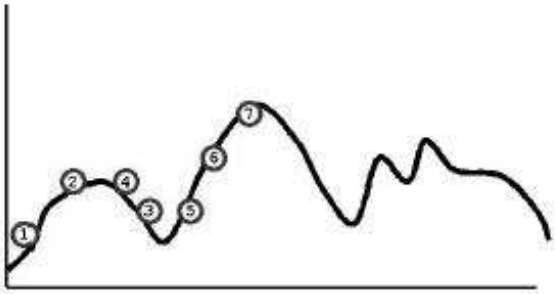
H ✗

C → H ✗

A → D → G

II. الگوریتم جست و جوی شبیه سازی حرارت:

الگوریتم تپه نوردی که اجازه‌ی حرکت رو به پایین، یعنی به طرف حالات با ارتفاع کمتر یا هزینه‌ی بالاتر را نمی‌دهد کامل نیست، زیرا ممکن است در یک ماکزیمم محلی متوقف شود. در طرف مقابل، یک حرکت کاملاً تصادفی یعنی انتخاب مابعدها به صورت تصادفی کامل است، اما کارا نیست. بنابراین تپه نوردی را با حرکت تصادفی ترکیب می‌کنیم تا هر دو ویژگی کامل بودن و کارایی را داشته باشد. شبیه سازی حرارت چنین الگوریتمی است. برای درک این الگوریتم، الگوریتم کاهش گرادیان را در نظر بگیرید و رفتن توپ پینگ پنگ به عمیق ترین شکاف در یک سطح ناصاف را در نظر بگیرید. اگر توپ بچرخد فقط در مینیمم محلی قرار می‌گیرد، اگر سطح را تکان بدهیم می‌توانیم توپ را از مینیمم محلی خارج کنیم، راه حل این است که سطح به اندازه‌ی کافی تکان داده شود تا توپ از مینیمم محلی خارج شود، اما نباید طوری تکان داده شود که از مینیمم مطلق خارج شود. الگوریتم شبیه سازی حرارت با تکان شدید (دمای زیاد) شروع می‌کند و به تدریج شدت تکان دادن را کاهش می‌دهد (به سمت دمای پایین می‌رود). تفاوت تپه نوردی و شبیه سازی حرارت در این است که در تپه نوردی بهترین مابعد انتخاب می‌شود، ولی در شبیه سازی حرارت مابعدها به طور تصادفی انتخاب می‌شوند. الگوریتم شبیه سازی حرارت در طراحی مدارات VLSI ،زمان بندی کارخانه‌ها و مسائل بهینه سازی در مقیاس بزرگ، نسبت به تپه نوردی مؤثرتر است.



III. جست و جوی پرتو محلی[۱]:

در روش‌های قبلی مانند تپه نوردی و شبیه سازی حرارت فقط یک گره در حافظه نگه داشته می‌شد ولی در الگوریتم جستجوی پرتو محلی به جای یک گره، k گره در حافظه نگه داری می‌شود. این الگوریتم با k حالت که به طور تصادفی تولید شده‌اند، شروع می‌کند. در هر مرحله تمام ما بعدهای k حالت را تولید می‌کند. اگر یکی از آنها هدف باشد الگوریتم متوقف می‌شود، در غیر اینصورت k حالت از بهترین ما بعدها انتخاب شده و این عمل تکرار می‌شود.

تفاوت جست و جوی پرتو محلی با تپه نوردی شروع تصادفی مجدد در این است که در تپه نوردی شروع تصادفی مجدد هر فرایند جستجو مستقل از بقیه اجرا می‌شود. در حالیکه در الگوریتم‌های جستجوی پرتو محلی اطلاعات مفید بین k فرایند جستجوی موازی مبادله می‌شود و وابستگی بین آنها وجود دارد جستجوی پرتو تصادفی به جای انتخاب k حالت از بهترین مابعدها، k حالت را به طور تصادفی از میان مابعدها انتخاب می‌کند، به طوریکه انتخاب یک مابعد یک تابع تصادفی از میزان ارزش آن مابعد می‌باشد.

گام‌های جستجوی پرتوی محلی

- شروع جستجو با k حالت تصادفی
- ایجاد حالت‌های بعدی k حالت
- نگهداری k تا بهترین از حالت‌های ایجاد شده
- بازگشت به گام دوم

IV. الگوریتم ژنتیک[۲]:

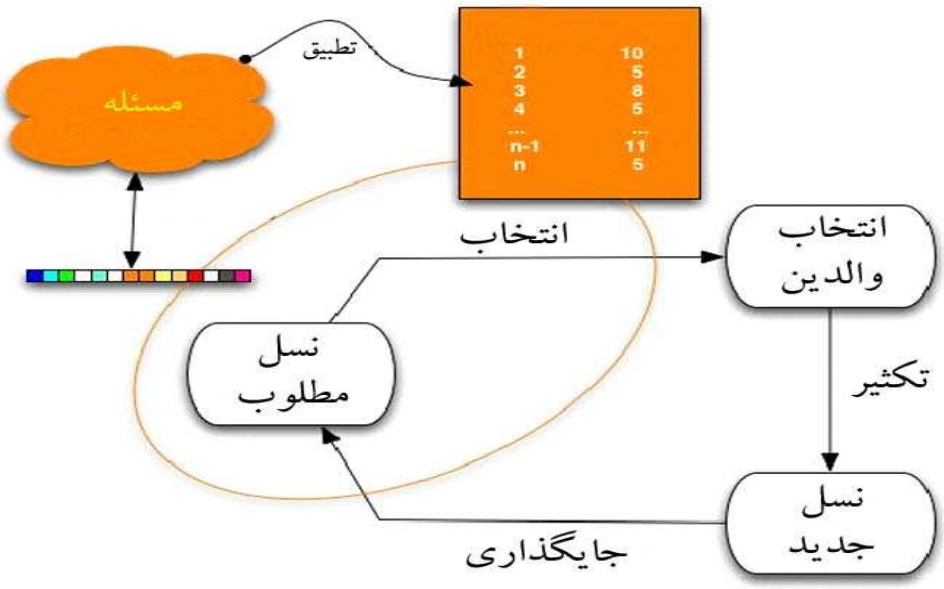
الگوریتم ژنتیک نوعی از جستجوی پرتو اتفاقی است که در آن ما بعدها از ترکیب دو حالت والد (به جای تغییر یک حالت) تولید می‌شوند. الگوریتم ژنتیک مانند جستجوی پرتو محلی با مجموعه‌ای از k حالت تصادفی شروع می‌کند که این مجموعه جمعیت[۳] نام دارد. هر حالت یا فرد، به صورت رشته‌ای از تعداد متناهی الفبا (یا مجوعه‌ای از صفر و یک‌ها) نمایش داده می‌شود. در مساله ۸ – وزیر حالت باید مکان ۸ وزیر را مشخص کند و هر وزیر باید در ستونی شامل ۸ مربع قرار گیرد. حالت‌ها در الگوریتم ژنتیک، کروموزوم نیز نامیده می‌شوند و هر کروموزوم از چندین ژن تشکیل شده است. الگوریتم ژنتیک یک الگوریتم تکاملی برای جست و جو در فضای وسیع است. رویه‌ی کلی الگوریتم ژنتیک به شرح ذیل می‌باشد:

---

[۱] Local beam search

[۲] Genetic Algorithm

[۳] population



I. تولید جمعیت:

یک جمعیت مجموعه‌ای از کروموزوم‌هاست که هر کروموزوم نمایانگر یک راه حل ممکن است. جمعیت اولیه می‌تواند توسط الگوریتم‌های مکاشفه‌ای به دست می‌آید.

II. ارزیابی کروموزوم‌ها:

به هر کروموزوم یک ارزش انتساب داده می‌شود. هدف جست و جوی ژنتیک، پیدا کردن ارزش بهینه‌ی کروموزوم می-باشد.

III. عملیات تبادل[1] و جهش[2]:

عملیات تبادل یک جفت از کروموزوم‌ها را به طور تصادفی انتخاب کرده سپس یک نقطه ی تصادفی از کروموزوم اول را انتخاب می‌کند، بعد در هر دو کروموزوم از آن نقطه تا انتهای کروموزوم را با هم جابه جا می‌کند. عملیات جهش به طور تصادفی یک کروموزوم را انتخاب کرده و سپس یک ژن که معادل انتقال وزیر به ستون دیگر در ۸ وزیر است را در درون آن کروموزوم به طور تصادفی تغییر می‌دهد.

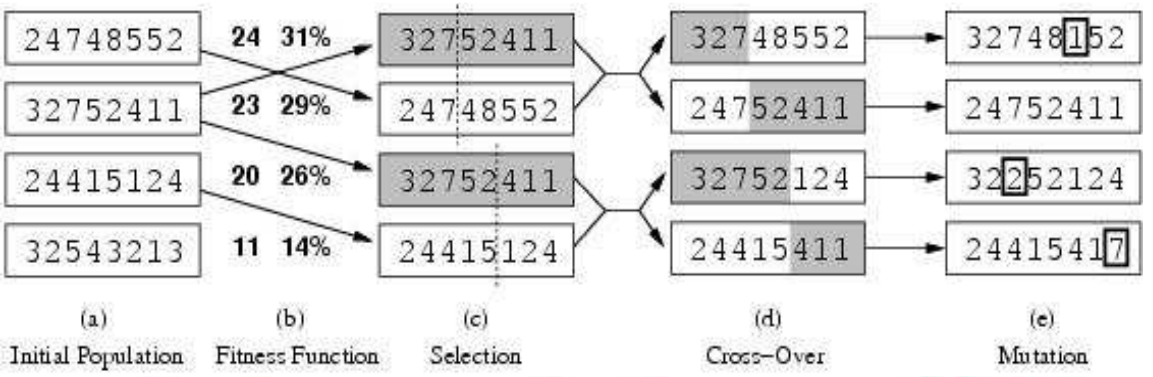
در نهایت کروموزوم‌های جدید به دست آمده از جمعیت دوباره ارزیابی می‌شوند. تا اینجا یک مرحله از الگوریتم ژنتیک به پایان می‌رسد. این مرحله از الگوریتم یا به هدف از قبل تعیین شده می‌رسد یا دوباره از بین این جمعیت، جمعیت تصادفی از کروموزوم‌ها انتخاب و الگوریتم تکرار می‌شود.

---

[1] Cross-over

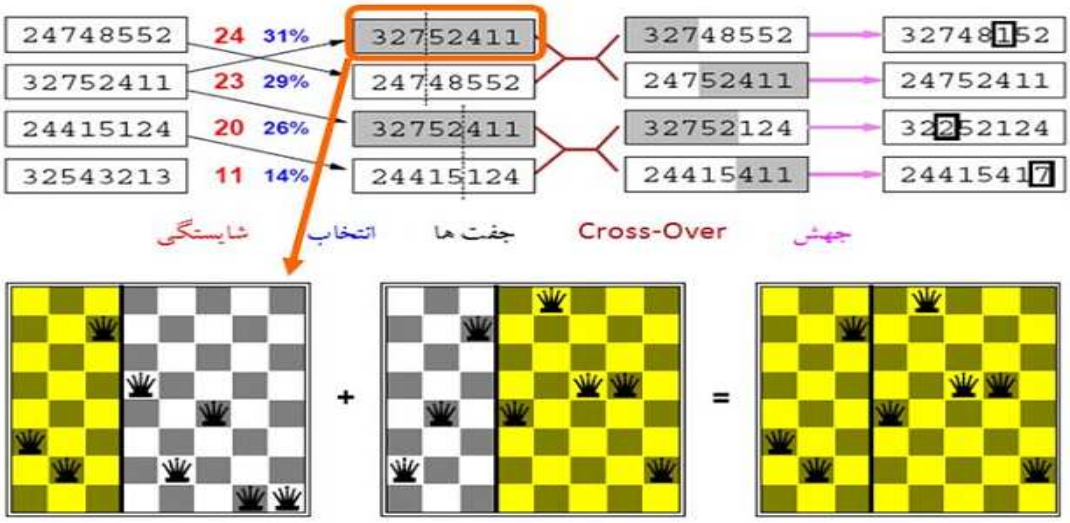
[2] mutation

مثال: شکل زیر مثالی از الگوریتم ژنتیک برای حل مساله ۸ – وزیر را نشان می‌دهد.

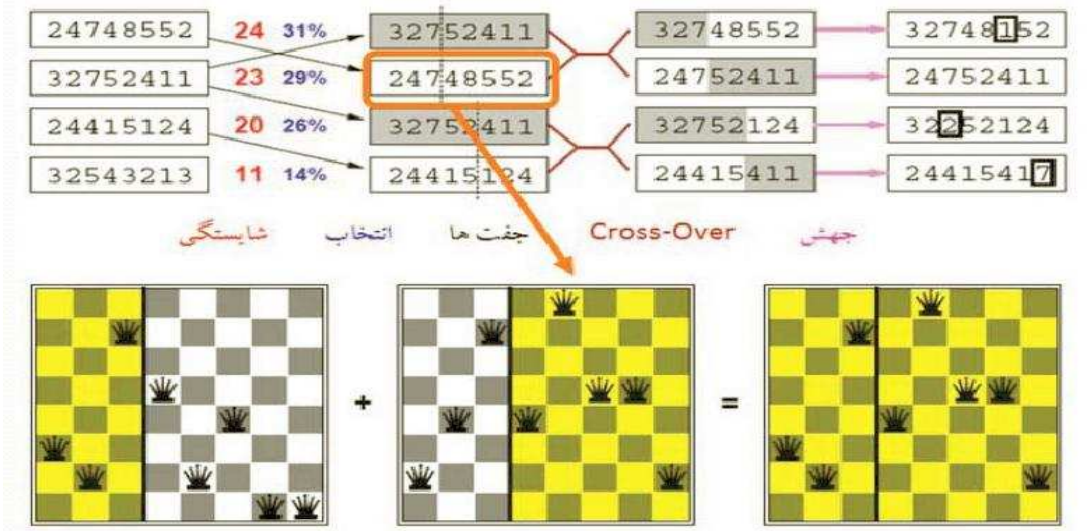


(3, 2, 7, 5, 2, 4, 1, 1) + (2, 4, 7, 4, 8, 5, 5, 2) = (3, 2, 7, 4, 8, 5, 5, 2)

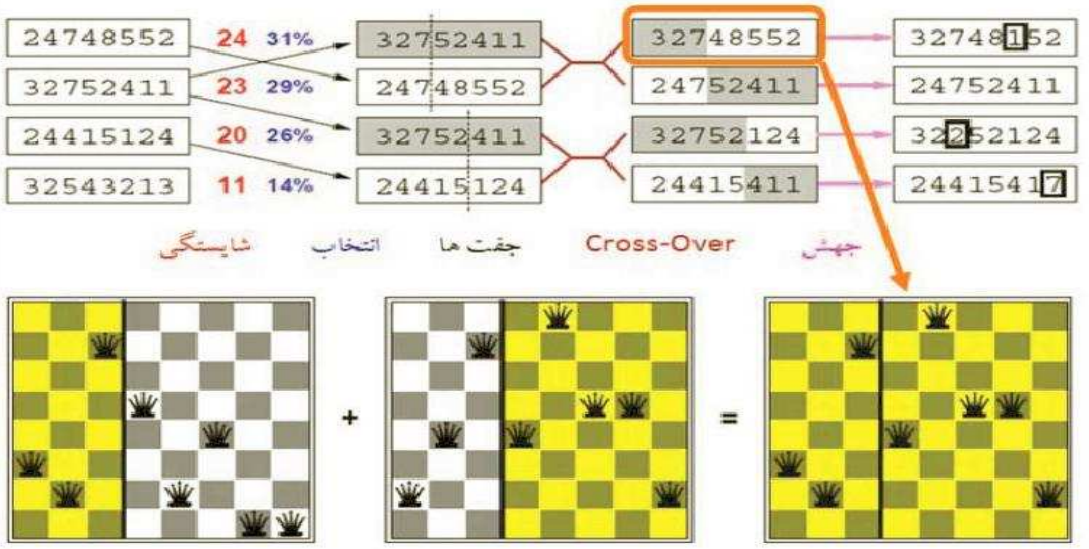
گام بعدی :



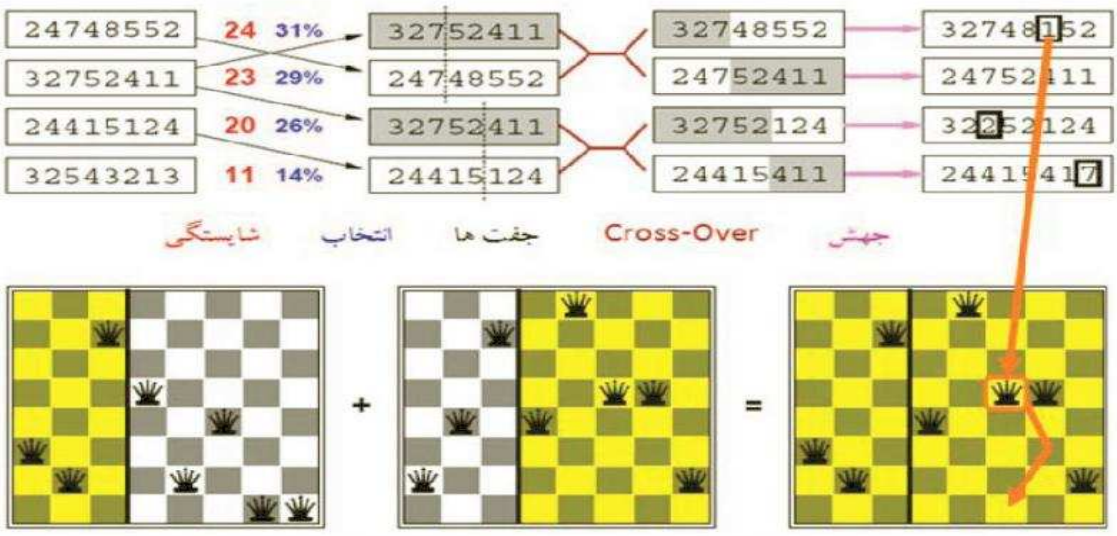
گام بعدی :



گام بعدی :



گام بعدی :



سوالات تستی آخر فصل

ترم اول ۸۷–۸۸

۱. نقطه ضعف روشن IDA* (Iterative Deeping A*) درچیست؟

الف. کامل نبودن ب. دوباره کاری ج. با عمق محدود د. مصرف حافظه زیاد

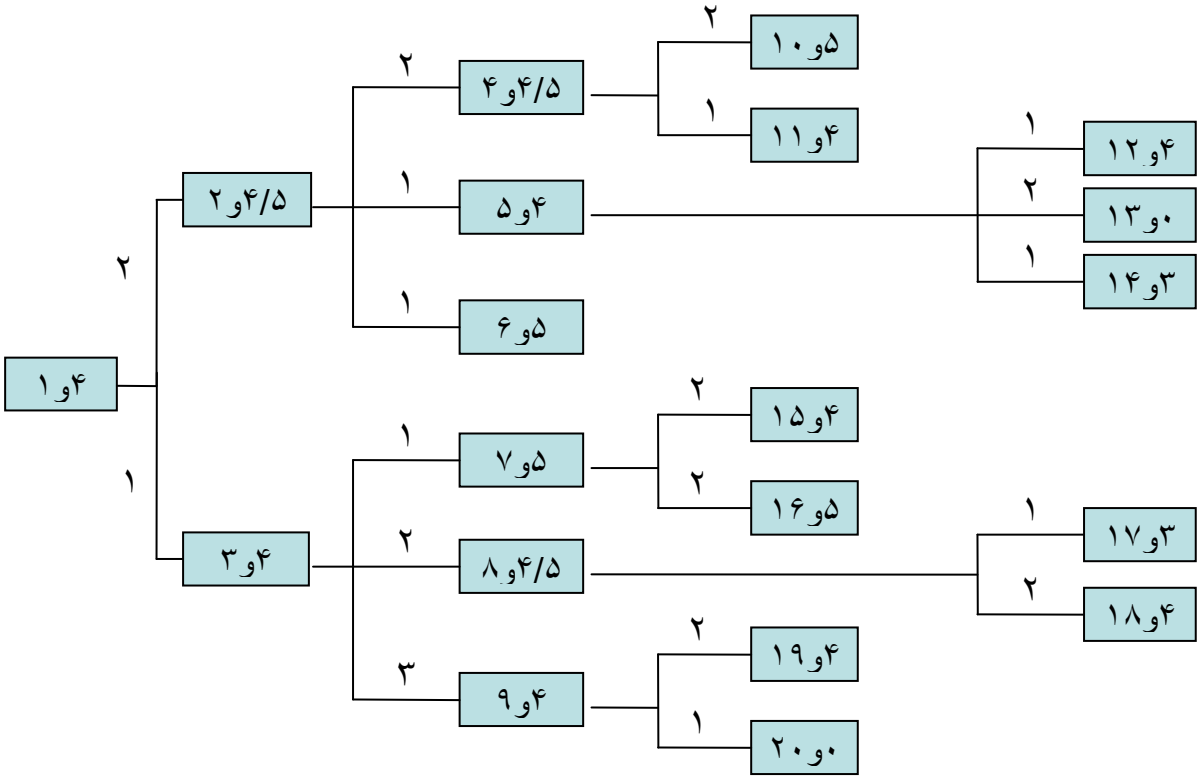
۲. در درخت تصمیم گیری زیر با استفاده از جستجوی A* کدام گزینه شماره گره‌های مورد بررسی را مشخص می-کند؟ توجه کنید هزینه هر گره در کنار شماره آن و هزینه هر شاخه روی آن نوشته شده است (درهرگره اولین عدد شماره گره ودومین عدد هزینه می‌باشد.)

الف. ۱ و ۲ و ۳ و ۴ و ۵ و ۶ و ۱۲ و ۱۳ و ۱۴

ب. ۱ و ۲ و ۳ و ۷ و ۸ و ۹ و ۱۹ و ۲۰

ج. ۱ و ۲ و ۳ و ۷ و ۸ و ۹ و ۱۷ و ۱۸ و ۱۹ و ۲۰

د. ۱ و ۲ و ۳ و ۷ و ۸ و ۹ و ۴ و ۵ و ۶ و ۱۲ و ۱۳ و ۱۴



۳. کدامیک از گزینه‌های زیر درست است؟

الف. به طور کلی در مواردی که حافظه در اختیار داریم بهتر است از روش پیمایش اول سطح برای جستجو استفاده کنیم.

ب. شرایط لازم برای آنکه A*، رسیدن به جواب بهینه را تضمین کند، به دامنه مسئله وابسته است

ج. الگوریتم پیمایش اول عمق همواره رسیدن به جواب را تضمین نمی‌کند.

د. الگوریتم A* در بدترین حالت عملکردی مشابه Best first دارد.

۴. کدامیک از موارد زیر دلایل گیرکردن جستجوی تپه نوردی می باشند.

الف. بیشینه‌های محلی

ب. فلات‌ها

ج. دماغه‌ها

د. همه موارد

۵. کدامیک از موارد زیر در خصوص روش جستجوی *LRTA در مقایسه با روش *A صحیح تر است؟

الف. *LRTA اغلب تمایل بیشتری به ادامه مسیر جاری دارد.

ب. *LRTA همواره مسیرهای کوتاهتری را می‌یابد.

ج. *LRTA اغلب تمایل کمتری به ادامه مسیر جاری دارد.

د. *LRTA همواره مسیرهای طولانی تری دارد.

۶. کدامیک از موارد زیر صحیح نیست؟

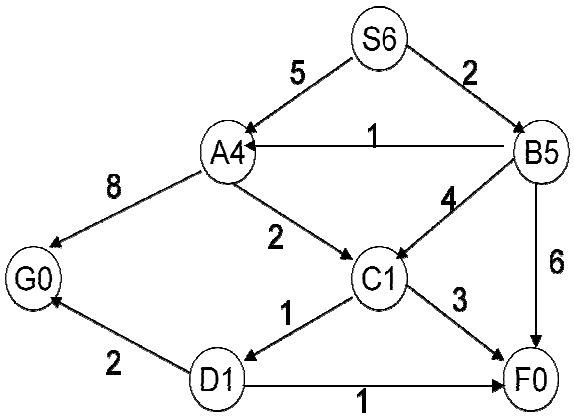
الف. الگوریتم جستجوی پرتوی محلی، اطلاعات K حالت را به جای یک حالت نگه می‌دارد.

ب. الگوریتم ژنتیک از پیوندو جهش تصادفی روی جمعیت برای تولیدنسل بعد کمک می‌گیرد.

ج. Online - Dfs - Agent قابلیت بازگشت به عقب را ندارد.

د. RBFS و *IDA از حافظه خیلی کم استفاده می‌کنند برای دسترسی فضای بیشتری از حافظه از *MA (*A مقید به حافظه) و *MA (*SMA ساده شده) استفاده می‌شود.

۷. در گراف مقابل حاصل جستجو با روش*A چیست؟ (Sنقطه شروع است و اعداد روی یال‌ها هزینه واقعی و اعداد داخل دایره‌ها مقدار h گره موردنظر است.)



الف. SBF

ب. SBAG

ج. SBSCDG

د. SBACDF

ترم دوم ۸۷–۸۸

۸. در کدام روش جستجو، بهترین جواب بدست می‌آید و اگر H(n) = 0 شود جستجو تبدیل به جستجوی هزینه یکنواخت می‌شود؟

الف. جستجوی حریصانه ب. جستجوی اول سطح

ج. جستجوی تپه نوردی د. جستجوی اول عمق *A

۹. اگر برای حل یک مسئله سه تابع هیوریستیک به نام‌های (H1, H2, H3) طراحی کنیم و هرسه تابع قابل پذیرش باشند و برای تمام وضعیت‌ها داشته باشیم: H2 > H1 > H3 آنگاه کدامیک از این توابع هیوریستیک برای حل مسئله بهتر است؟

الف. H1 ب. H3 ج. H2 د. H3 یا H2

۱۰. کدامیک از روش‌های زیر برای رفع مشکل حافظه در *A طراحی نشده‌اند؟

الف. *MA ب. RBFS ج. *IDA د. *LRTA

۱۱. توسط کدام استراتژی جستجو، مسائلی قابل حل هستند که در آنها مسیر رسیدن به جواب مهم نباشد و فقط رسیدن به جواب اهمیت دارد؟

الف. الگوریتم‌های ارضای محدودیت ب. الگوریتم‌های*A

ج. الگوریتم‌های جستجوی محلی د. الگوریتم‌های ناآگاهانه

۱۲. کدام الگوریتم را گاهی جستجوی حریصانه محلی می‌نامند؟

الف. جستجوی ارضای محدودیت ب. جستجوی *A

ج. جستجوی تپه نوردی د. جستجوی سخت سازی شبیه سازی شده

۱۳. کدام روش تپه نوردی نسبت به سایرین برتری دارد؟

الف. با شروع مجدد اتفاقی ب. اولین گزینه ج. اتفاقی د. معمولی

۱۴. کدام گزینه در مورد *LRTA صحیح است؟

الف. در محیط‌های متناهی با اقدامات برگشت پذیر کامل است.

ب. برخلاف *A در فضاهای نامتناهی کامل نیست.

ج. با استفاده از جدول Result نقشه محیط را تهیه می‌کند.

د. اکتشاف محیط دارای N حالت به این روش از مرتبه نمایی است.

ترم تابستان ۸۸

۱۵. الگوریتم ..... به جای یک حالت، اطلاعات k حالت را در حافظه نگهداری می‌کند.

الف. جستجوی تپه نوردی ب. الگوریتم تپه نوردی اتفاقی

ج. جستجوی پرتوی محلی د. جستجوی سخت سازی شبیه سازی شده

۱۶. کدام گزینه صحیح است؟

الف. به جستجوهایی که از تابع ارزیاب (f) برای انتخاب بهترین گره استفاده می‌کنند، آگاهانه می‌گوییم.

ب. جستجوی حریصانه بهینه و کامل است.

ج. هر هیوریستیک قابل قبولی سازگار است.

د. به جستجویی که h(n) بخشی از تابع ارزیاب (f) آن باشد، آگاهانه می‌گوییم.

۱۷. کدام یک از شرایط زیر، شرط سازگاری یک تابع هیوریستیک را بیان می کند؟

(n گره جاری، 'n گره پسین گره n است که با اقدام a تولید شده است.)

(تابع c هزینه گام رسیدن به 'n از n با اقدام a می‌باشد.)

الف. $h(n) < h(n') + c(n, a, n')$ ب. $h(n) \leq h(n') + c(n, a, n')$

ج. $h(n) > c(n, a, n') + h(n)$ د. $h(n) \geq c(n, a, n') + h(n)$

۱۸. RBFS برای غلبه بر چه مشکلی در A^* مطرح می‌شود؟

الف. کامل بودن ب. عدم بهینگی ج. پیچیدگی حافظه د. پیچیدگی زمانی

۱۹. کدام گزینه صحیح نیست ؟

الف. برای مکعب روبیک، بانک‌های پراکنده الگو بسیار موثر بوده است.

ب. اگر برای یک مسئله هیوریستیک‌های $h_1,\ldots,h_m$ موجود باشد، $h(n)=\max\{h_1(n),\ldots,h_m(n)\}$ بر همه برتری دارد.

ج. استفاده از بانک‌های اطلاعاتی پراکنده الگو سرعت پازل ۲۴ تایی را به میزان قابل ملاحظه‌ای افزایش می‌دهد.

د. هزینه یک راه حل بهینه برای یک مسئله تعدیل شده، یک هیوریستیک قابل قبول برای مسئله اصلی است.

۲۰. برای حالتی که تعداد کمی بیشینه محلی و فلات وجود داشته باشد، کدام نوع از الگوریتم‌های تپه نوردی می‌تواند سریع تر یک راه حل خوب پیداکند؟

الف. تپه نوردی اولین گزینه ب. تپه نوردی اتفاقی

ج. تپه نوردی با شروع مجدد تصادفی د. الف و ب

۲۱. کدام گزینه صحیح نیست ؟

الف. جستجوی سخت سازی شبیه سازی شده با تکان‌های شدید آغاز شده و تدریجاً از شدت تکان‌ها کاسته می‌شود.

ب. جستجوی پرتوی محلی مانند k جستجوی محلی می‌باشد که به صورت موازی اجرا می شوند.

ج. در جستجوی پرتوی محلی بهترین k پسین جایگزین حالت قبلی می‌شود.

د. جستجوی سخت سازی شبیه سازی شده می تواند برای مسائلی همچون چیدمان VLSI مورد استفاده قرار گیرد.

۲۲. کدام گزینه در مورد جستجوی برخط صحیح نیست ؟

الف این جستجو برای یک مسئله اکتشافی (حالت‌ها و اقدامات ناشناخته) ضروری می‌باشد.

ب. در بعضی موارد بهترین نسبت روابطی دست یافتنی بینهایت است.

ج. فضای حالتی که اقدامات قابل برگشت دارند مطمئنا قابل اکتشاف هستند.

د. پیشروی ONLINE.DFS.AGENT در هر فضای حالتی قابل استفاده است.

۲۳. کدام گزینه در مورد $LRTA^*$ صحیح نیست ؟

الف. جستجوی محلی بر خطی است که محیط دارای n حالت را حداکثر $O(n^2)$ گام اکتشاف می‌کند.

ب. هزینه تخمینی رسیدن به هدف از طریق همسایه‌ای مانند s' = هزینه رسیدن به $H(s') + s'$

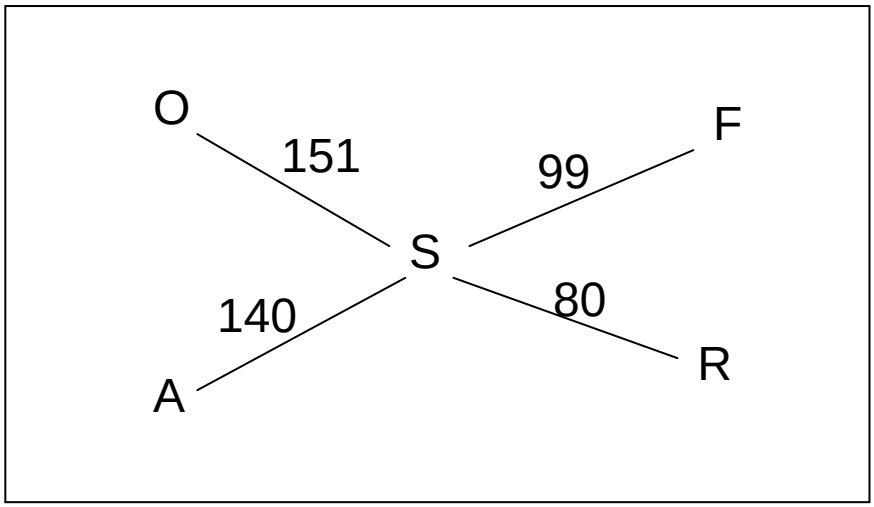
ج. این الگوریتم، برای فضاهای حالت نامتناهی، کامل است.

د. همانندONLINE.DFS.AGENT با استفاده از جدول result نقشه‌ای از محیط تهیه می‌کند.

ترم اول ۸۸–۸۹

۲۴. باتوجه به اشکال مقابل (با هدف رسیدن به بخارست(B) از سیبیو (S))، الگوریتم*A بعد از S کدام گره را اول بسط می‌دهد؟

N	H(n)
S	253
O	380
A	366
F	176
R	193



الف. O ب. F ج. R د. A

۲۵. به کارگیری*A با Graph_search به شرطی که h(n) قابل قبول باشد ولی سازگار نباشد چگونه است؟

الف. کامل ونیمه بهینه ب. کامل و بهینه ج. غیرکامل و غیربهینه د غیرکامل و بهینه

۲۶. کدام گزینه در مورد RBFS صحیح است؟

الف. پیچیدگی حافظه آن O(b+d) است

ب. تاحدودی از *IDA موثرتر است.

ج. از تولید مجدد افراطی گره‌ها سود می‌برد.

د. اگر حافظه بیشتری در دسترس باشد از ب،عملکرد بهتری خواهد داشت.

۲۷. اگر ضریب موثر انشعاب(*b) برای ۴ تابع هیوریستیک (آروینی) h1,h2,h3,h4 به ترتیب 1.2, 1.3, 1.4, 1.5 باشد، کدامیک بر بقیه ارجحیت دارد؟

الف.h1 ب. h2 ج. h3 د. h4

۲۸. در محاسبه هیورستیک از هزینه راه حل زیر مسائل، بانک‌های اطلاعاتی الگو چگونه ساخته می شود؟

الف. با جستجوی رو به عقب از حالت هدف و ثبت هزینه هر الگوی جدید

ب. حل هر زیر مسئله و به دست آوردن و ثبت هزینه آن هنگامی که با آن روبرو می شویم.

ج. تخمین هزینه زیر مسائل توسط مسئله تعدیل شده

د. با استفاده از یک الگوریتم فراگیری استقرایی

۲۹. الگوریتم جستجوی محلی برای حل کدام مسئله مناسب نمی باشد؟

الف. مسئله n وزیر ب. چیدمان دستگاه‌های کارخانه

ج. به دست آوردن مسیر بهینه برای رسیدن به شهر بخارست د. مدارهای مجتمع

۳۰. برای حل مسئله n وزیر کدام روش مؤثرتر است؟

الف. تپه نوردی ب. تپه نوردی با حرکات کناره

ج. تپه نوردی اولین گزینه د. تپه نوردی با شروع مجدد تصادفی

۳۱. از دو فرد (کروموزوم) مقابل با عمل Cross-over و یک جهش (mutation) کدام فرد نمی‌تواند ایجاد شود؟

۳ ۲ ۷	۵ ۲ ۴ ۱ ۱

۲ ۴ ۷	۴ ۸ ۵ ۵ ۲

الف. ۳۲۷۴۸۱۵۲ ب. ۲۴۷۵۲۴۱۵ ج. ۳۴۷۴۸۱۵۲ د. ۲۱۷۵۲۴۱۱

۳۲. کدام گزینه درمورد جستجوی سخت سازی شبیه سازی شده (شبه تابکاری) صحیح نیست؟

الف. در اکثر موارد قادر به فرار از اکسترم‌های محلی است.

ب. نتیجه تلاش برای ترکیب تپه نوردی با یک راهپیمایی تصادفی می‌باشد.

ج. یک حرکت بد با احتمال $e^{\Delta E/T}$ امکان وقوع خواهد داشت

د. این الگوریتم همواره بهینه سراسری را خواهد یافت.

۳۳. در Online-DFS-Agent عامل (کارگزار) در چه صورت جستجویش کامل شده است؟

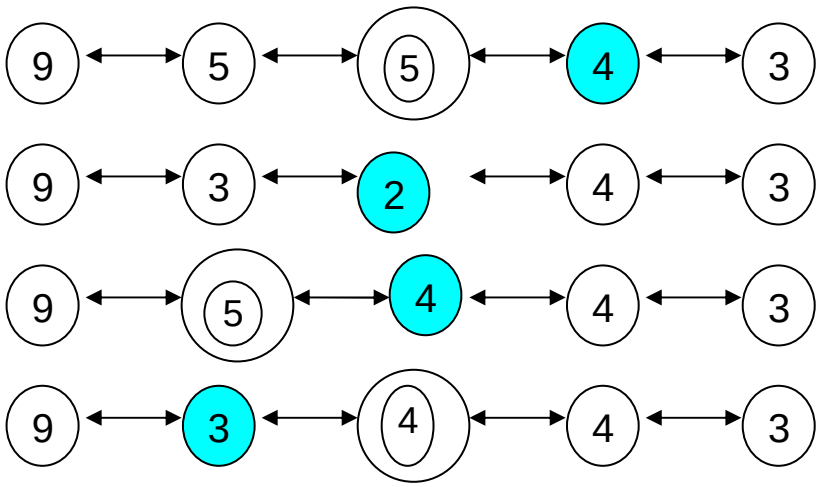
الف. در صورتی که از وضعیت فعلی اقدام اکتشاف نشده‌ایی وجود نداشته باشد (Unexplored[s] خالی باشد.)

ب. در صورتی که حالتی برای عقبگرد عامل وجود نداشته باشد. (Unbacktracked[s] خالی باشد.)

ج. در حالی که هدف یافته شود (Goal-test[s] برابر True باشد.)

د. در صوتی که شرایط ب یا ج برقرار باشد.

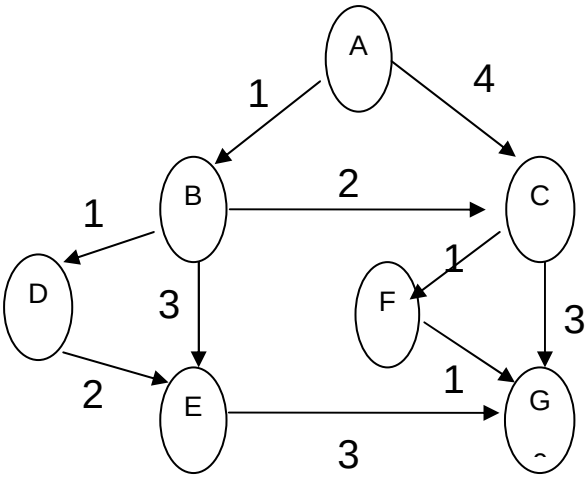
۳۴. اگر الگوریتم*LRTA در فضای حالت یک بعدی ۴ تکرار انجام داده و شکل مقابل فضای حالت را بعد از تکرارها نمایش دهدترتیب درست تکرارها را بیان کنید؟



الف. ۲و۴و۳و۱ ب. ۲و۳و۴و۱ ج. ۴و۳و۲و۱ د. ۴و۲و۳و۱

ترم دوم ۸۸–۸۹

۳۵. در گراف زیر مسیر پیداشده توسط الگوریتم جستجوی A* کدام گزینه است؟ (A گره شروع، اعداد روی یال‌ها هزینه واقعی و اعداد داخل دایره‌ها مقدار h گره مورد نظر است.) (فرض کنید A* به قابل قبول نبودن h توجهی نمی‌کند.)



الف. A,B,C,F,G

ب. A,C,G

ج. A,B,C,G

د. A,B,E,G

۳۶. در گراف سؤال قبل، مسیر پیداشده توسط الگوریتم جستجوی اول بهترین حریصانه کدام است؟

الف. A, B, C, F, G ب. A, C, G ج. A, B, C, G د. A, B, D, E, G

۳۷. نقطه ضعف الگوریتم جستجوی عمیق شونده تکراری A* (IDA*) چیست؟

الف. کامل نبودن ب. دوباره کاری

ج. مصرف زیاد حافظه د. کارایی پایین

۳۸. سه تابع هیوریستیک قابل قبول h_1 , h_2 , h_3 برای حل مسئله‌ای به روش A^* پیشنهاد شده است. بهترین انتخاب برای ادامه مسیر از گره n استفاده از کدام هیوریستیک است؟

الف. $h_1 \times h_2 \times h_3$ ب. $h_1 + h_2 + h_3$

ج. $\max\{h1, h2, h3\}$ د. انتخاب دلخواه یکی از آن‌ها

ترم تابستان ۸۹

۳۹. اگر به ازای هر گره درخت، رابطه ی $h_1 < h_2 < h_3 < h_4$ برای ۴ هیوریستیک h_1تا h_4 برقرار باشد بین ضرایب موثر انشعاب مربوط به هر هیوریستیک (یعنی b_1^* تا b_4^*) چه رابطه ای برقرار است؟

الف. $b_1^* < b_2^* < b_3^* < b_4^*$ ب. $b_4^* < b_3^* < b_2^* < b_1^*$

ج. رابطه مشخصی بین ضرایب برقرار نمی‌باشد. د. به ازای dهای مختلف روابط متفاوت خواهد بود.

۴۰. برای مسئله معمای ۸ برای وضعیت start مقدار h_1 و h_2 (فاصله منتهن) به ترتیب کدام است؟

الف. ۷ و ۹ ب. ۹ و ۷ ج. ۶ و ۱۰ د. ۷ و ۱۰

۳		۲
۴	۱	۸
۵	۶	۷

start state

	۱	۲
۳	۴	۵
۶	۷	۸

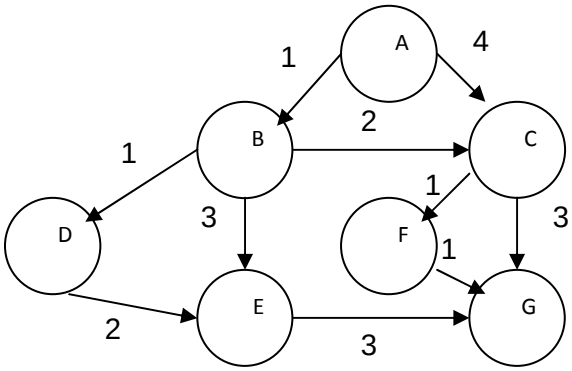
Goal state

۴۱. اگر با فضای حالت نامتناهی طرف هستید و مسیر رسیدن به هدف برای شما اهمیتی ندارد از کدام جستجو استفاده می‌کنید ؟

الف. جستجوی عمق ب. جستجوی سطحی ج. جستجوی A^* د. جستجوی محلی

۴۲. مسیر یافت شده توسط الگوریتم *A برای گراف مقابل چیست؟

الف. ABCFG ب. ACG ج. ABCG د. ABEG



۴۳. مقدار تابع برازش برای فرد (کروموزوم) 32543213 کدام است؟(در مسئله ۸ وزیر)

(راهنمایی: تابع برازش = تعداد جفت وزیرهایی که به هم حمله نمی‌کنند.)

الف. ۲۰ ب. ۱۱ ج. ۲۳ د. ۲۴

۴۴. اگر جمعیت اولیه در الگوریتم ژنتیک حاوی ۴ فرد (کروموزوم) باشد و مقدار تابع برازش برای آنها به ترتیب ۲ و ۸ و ۶ و ۴ باشد احتمال انتخاب افراد به ترتیب کدام است؟

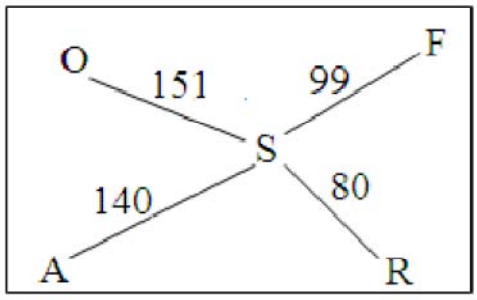
الف. ۱۰٪، ۴۰٪، ۳۰٪، ۲۰٪ ب. ۲۰٪، ۳۰٪، ۴۰٪، ۱۰٪

ج. ۲۵٪، ۲۵٪، ۲۵٪، ۲۵٪ د. ۱۰٪، ۱۰٪، ۱۰٪، ۱۰٪

ترم اول ۸۹–۹۰

** با توجه به اشکال مقابل (با هدف رسیدن به بخارست (B) از سیبو (S)) به سوالات ۴۵ و ۴۶ پاسخ دهید:

n	H(n)
S	253
O	380
A	366
F	176
R	193



۴۵. الگوریتم جستجوی حریصانه بعد از S، کدام گره را اول بسط می‌دهد؟

الف. O ب. F ج. R د. A

۴۶. جستجو با هزینه یکسان بعد از S، کدام گره را اول بسط می‌دهد؟

الف. O ب. F ج. R د. A

۴۷. کدامیک جزء جستجوهای محلی نمی‌باشند؟

الف. تپه نوردی ب. سخت سازی شبیه سازی شده (Annealing Simulated)

ج. الگوریتم ژنتیک د. RBFS

۴۸. با کدام شرایط در Graph_search با جستجوی A* مسیر بهینه به هر حالت تکراری همیشه اولین مسیری است که دنبال می‌شود؟

الف. قابل قبول بودن H(n) ب. سازگاری H(n)

ج. هیچگاه امکان پذیر نیست. د. بدون شرط همواره امکان پذیر است.

۴۹. کدام گزینه درمورد SMA* صحیح نیست؟

الف. تا هنگامی که حافظه پر نشده همانند A* عمل می‌کند.

ب. اگر حافظه پر باشد گره با بیشترین f را حذف می‌کند.

ج. جد یک زیر درخت از کیفیت بهترین مسیر در آن زیر درخت آگاه است.

د. اگر تمام گره‌های برگی دارای مقدار f یکسانی باشند با شکست روبرو می‌شوند.

۵۰. با فرض اینکه مسئله‌ای حاوی سه شرط محدود کننده باشد، حداکثر چند مسئله‌ی تعدیل شده (Relaxed) برای آن می‌توان تولید نمود؟

الف. ۶ ب. ۴ ج. ۸ د. ۷

۵۱. کدامیک جزء روش‌های مرسوم ایجاد توابع هیوریستیک نمی‌باشند؟

الف. به دست آورن هزینه یک راه حل بهینه برای یک مسئله ی تعدیل شده (Relaxed)

ب. مجموع یا ترکیب هزینه راه حل‌های زیر مسائل (با استفاده از بانک‌های اطلاعاتی الگو)

ج. فراگیری هیوریستیک از تجارب قبلی و یا تعمیم آن برای حالات مشابه

د. در نظر گرفتن مقدارهای تصادفی برای h(n) و ثبت مقادیری که جستجو را به سمت هدف هدایت می‌کند.

۵۲. در حل مسئله ۸ وزیر توسط الگوریتم ژنتیک، مقدار تابع برازش برای یک راه حل کدام است؟ (راهنمایی: تابع برازش = تعداد جفت وزیرهایی که به هم حمله نمی‌کنند.)

الف. ۲۴ ب. ۲۸ ج. ۵۶ د. ۲۵

ترم دوم ۸۹ – ۹۰

۵۳. کدام گزاره‌ها عموماً در مورد الگوریتم‌های جستجوی محلی صحیح است ؟

۱) مسیرهایی که توسط جستجو دنبال می‌شود نگهداری می‌شود.

۲) نظام مند هستند

۳) از حافظه کمی استفاده می‌کنند

۴) در فضاهای حالت بزرگ یا نامتناهی اغلب می‌توانند راه حل معقولی بیابند

۵) آزمون هدف در آن‌ها وجود ندارد

الف. ۳ و ۴ ب. ۴ و ۵ ج. ۳ و ۴ و ۵ د. همگی

۵۴. در جستجوی محلی حرکات کناره برای عبور از کدام مورد می‌تواند مفید باشد؟

الف. دماغه‌ها ب. فلات ج. شانه د. ماکزیمم محلی

۵۵. اگر تابع برازش برای جمعیتی با چهار فرد به ترتیب ۲۴ و ۲۳ و ۲۰ و ۱۱ باشد احتمال انتخاب این ۴ فرد به ترتیب چقدر است؟

الف. ۴۰٪ ۲۵٪ ۲۲٪ ۲۱٪ ب. ۳۱٪ ۲۹٪ ۲۶٪ ۱۴٪

ج.۲۱٪ ۲۲٪ ۲۵٪ ۳۱٪ د.۱۴٪ ۲۶٪ ۲۹٪ ۴۱٪

۵۶. برای مسئله‌ای که ابتدا حالات محیط و اقدامات ناشناخته هستند (مسئله اکتشافی) کدام گزینه صحیح است؟

الف.ضرورتاً باید از جستجوی بر خط (آنلاین) استفاده کرد

ب. در صورتی که اقدامات بعداً شناخته شوند می‌تواند به صورت offline جستجو کند.

ج. می‌توان همواره جستجو و سپس اقدامات را به ترتیب انجام داد (به صورت offline)

د. بهتر است ابتدا از جستجوی offline و سپس از جستجوی بر خط استفاده شود و نتیجه این دو ترکیب گردد.

۵۷. الگوریتم ONLINE-DFS-AGENT چه زمانی متوقف می‌شود؟

۱) وقتی که تمام اقدامات حالت فعلی اکتشاف شده‌اند.

۲) وقتی که برای عامل از حالت فیزیکی فعلی حالتی وجود نداشته باشد که به آن عقب گرد کند.

۳) وقتی که هدف را بیابد.

۴) هنگامی که نتیجه عقبگرد حالت شروع باشد.

الف.۱.۲.۳.۴ ب. ۳.۴ ج. ۲.۳.۴ د. ۲.۳

تابستان ۹۰

۵۸. کدام گزینه بصورت جستجوی برخط قابل انجام نیست؟

الف. تپه نوردی ب. اول عمق ج. A^* د. همه گزینه‌ها قابل انجامند.

۵۹. کدام یک از فضاهای زیر در الگوریتم‌های تپه نوردی، فرآیند جستجو را با مشکل مواجه می‌کند؟

الف. دماغه ب. فلات

ج. بیشینه محلی د. همه گزینه‌ها

۶۰. روش جستجوی A^*، تحت چه شرایطی یافتن پاسخ بهینه را تضمین می‌کند؟

الف. در صورتی که تابع هیورستیک دارای خطای تخمین در شعاع همگرایی محدود باشد.

ب. شرایط لازم برای اینکه روش A^* ج.اب بهینه را تضمین کند، به دامنه مسئله بستگی دارد.

ج. در صورتی که تابع هیورستیک مورد استفاده، فاصله وضعیت‌های مختلف تا هدف را هرگز کمتر از مقدار واقعی تخمین نزند.

د. در صورتی که تابع هیورستیک مورد استفاده، فاصله وضعیت‌های مختلف تا هدف را هرگز بیش از مقدار واقعی تخمین نزند.

۶۱. کدامیک از الگوریتم‌های زیر، جستجوی تپه نوردی اتفاقی است که از جهش و پیوند برای ایجاد حالت‌های جدید استفاده می‌کند؟

الف. RBFS ب. SMA^* ج. الگوریتم ژنتیک د. حریصانه

۶۲. کدامیک از الگوریتم‌های زیر جزء الگوریتم‌های جستجوی آگاهانه می‌باشد؟

الف. جستجوی دو طرفه ب. اول عمق ج. عمق محدود د. کامل بودن

پاسخ نامه تستی

سوال	گزینه صحیح	سوال	گزینه صحیح	سوال	گزینه صحیح	سوال	گزینه صحیح
۱	ب	۲۰	ج	۳۹	ج	۵۸	ج
۲	د	۲۱	ب	۴۰	الف	۵۹	د
۳	ج	۲۲	د	۴۱	د	۶۰	د
۴	د	۲۳	ج	۴۲	الف	۶۱	ج
۵	الف	۲۴	ج	۴۳	ب	۶۲	د
۶	ج	۲۵	الف	۴۴	الف		
۷	د	۲۶	ب	۴۵	ب		
۸	د	۲۷	الف	۴۶	ج		
۹	ج	۲۸	الف	۴۷	د		
۱۰	د	۲۹	ج	۴۸	ب		
۱۱	ج	۳۰	د	۴۹	د		
۱۲	ج	۳۱	ج	۵۰	د		
۱۳	الف	۳۲	د	۵۱	د		
۱۴	ب	۳۳	ج	۵۲	الف		
۱۵	ج	۳۴	د	۵۳	ج		
۱۶	الف	۳۵	الف	۵۴	ب		
۱۷	ب	۳۶	ب	۵۵	ب		
۱۸	ج	۳۷	ب	۵۶	ب		
۱۹	الف	۳۸	ج	۵۷			

سوالات تشریحی آخر فصل

ترم اول ۸۷–۸۸

الگوریتمی برای جستجوی تپه نوردی ارائه دهید.

ترم دوم ۸۷–۸۸

الف. روش تولید سه مسئله تعدیل شده برای مسئله معمای هشت را بیان کنید.

ب. مقدار هیوریستیک‌های بدست آمده از این مسائل تعدیل شده را از وضعیت مشخص شده در شکل الف برای رسیدن به هدف در شکل ب را مشخص کنید.

ج. اگر تعدادی هیوریستیک قابل قبول داشته باشیم بهترین هیوریستیک ممکن را چگونه می‌توان از آنها ایجاد کرد؟ مشکل این هیوریستیک چیست؟

۴	۸	
۷	۳	۲
۶	۱	۵

شکل الف. وضعیت شروع

	۱	۲
۳	۴	۵
۶	۷	۸

شکل ب. وضعیت هدف

ترم تابستان ۸۸

مسئله معمای ۸ را در نظر بگیرید.

۱	۲	۳
۸		۴
۷	۶	۵

حالت شروع

۲	۸	۳
۱	۶	۴
۷		۵

حالت هدف

الف. دو تابع هیوریستیک قابل قبول برای این معما، طراحی کنید. (چگونگی طراحی را بیان کنید.)(۵/. نمره)

ب. الگوریتم جستجوی A^* را روی هریک از این هیوریستیک‌ها اعمال کنید. (از حالت شروع، درخت جستجو را تا رسیدن به هدف مرحله به مرحله رسم نمایید.) (۵/. نمره)

ج. نتایج این دو جستجو را مقایسه کنید. (با ذکر دلیل) (۵/. نمره)

فرض کنید می‌خواهیم مسئله ۸ وزیر را بااستفاده از الگوریتم ژنتیک حل کنید.

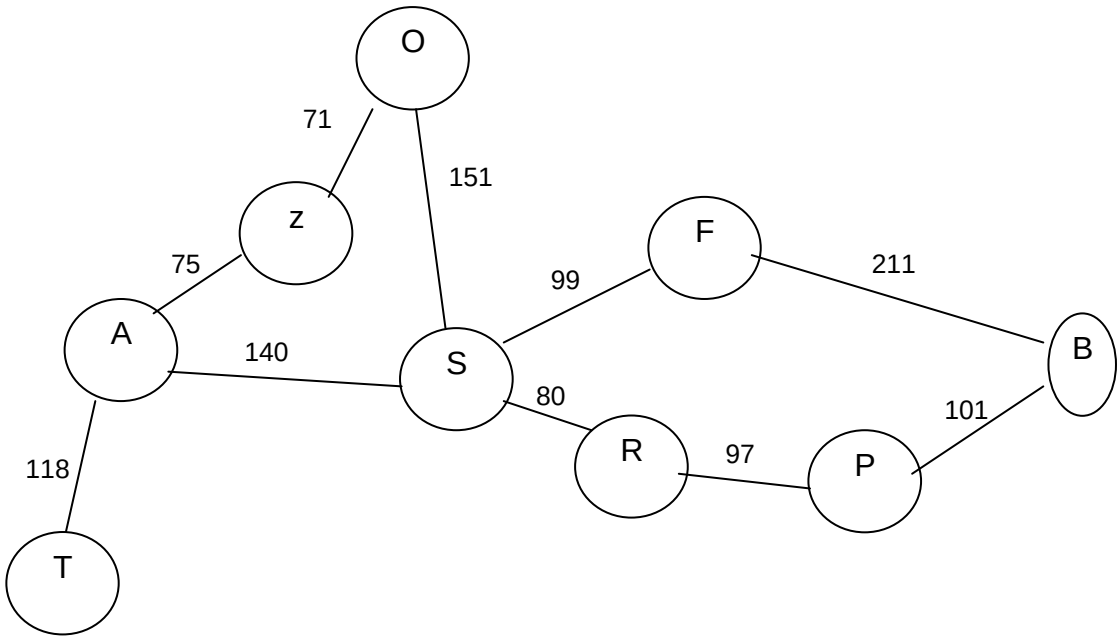
الف. مسئله را برای حل با الگوریتم ژنتیک، چگونه کد می‌کنید؟ (۵/. نمره)

ب. افرادی (کروموزوم‌هایی) که در هر جمعیت، در این مسئله وجود دارند به چه شکلی هستند؟ (۵/. نمره)

ج. به صورت مثال، چگونگی اعمال عملگر ژنتیکی crossover را، در این مسئله که روی دو فرد اعمال شده، نشان دهید. (۵/. نمره)

ترم اول ۸۸–۸۹

الگوریتم جستجو توسط A* با شروع از راس Z، درخت جستجو را مرحله به مرحله جستجو دهید تا هدف B بدست آید. هزینه مسیر چقدر است واز چه شهرهایی عبور می‌کند؟ (مقدار داخل هر گره برابر هزینه فاصله مستقیم تا هدف B می‌باشد.) (۱/۵ نمره)



ترم دوم ۸۸–۸۹

برای مسئله پازل ۸ تایی:

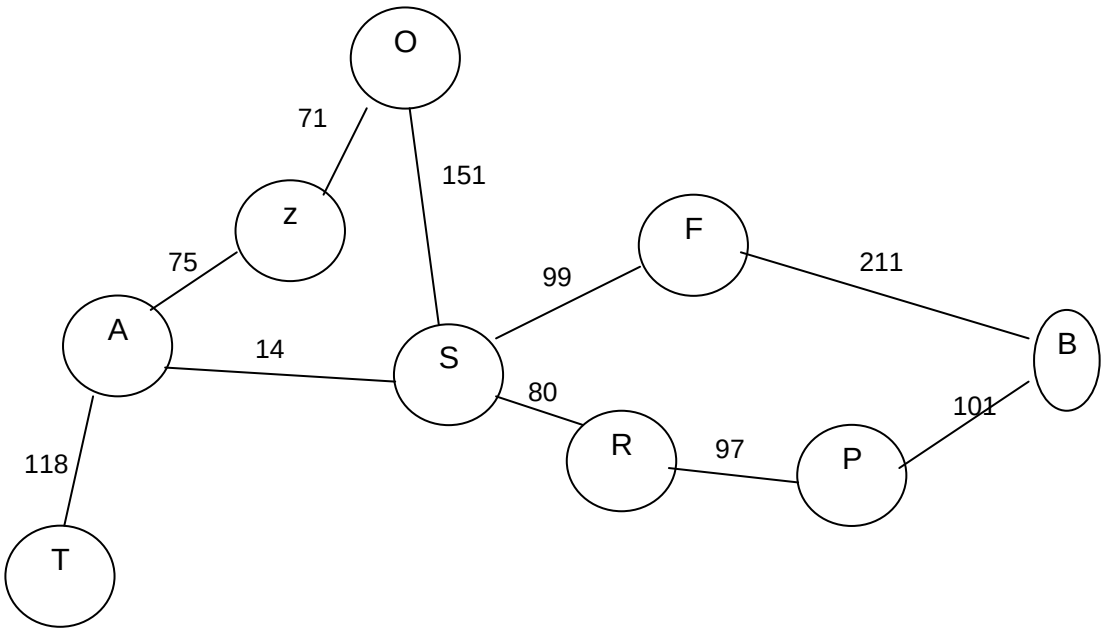
الف. سه مسئله تعدیل شده (با محدودیت‌های کمتر) تعریف کنید. (۰/۷۵ نمره)

با توجه به قسمت (الف)، سه هیوریستیک قابل قبول برای هر مسئله، استخراج کنید. (۰/۷۵ نمره)

الگوریتم تابکاری شبیه سازی شده حرارت (simulated annealing) را شرح دهید. (۰/۷۵ نمره)

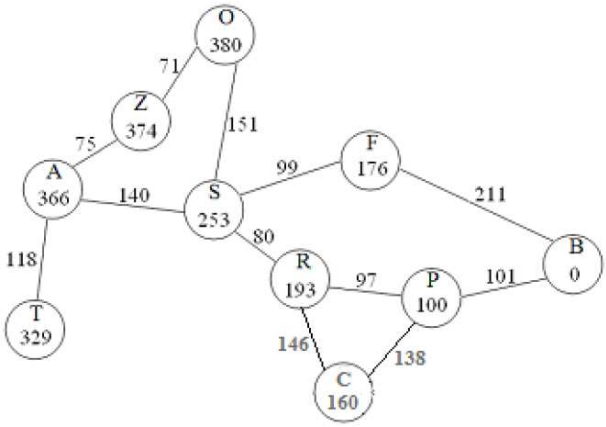
ترم تابستان ۸۹

توسط RBFS باشروع از راس A، درخت جستجو را مرحله به مرحله توسعه دهید تا هدف B بدست آید. (مقدار best و alternative و f.limit رادر هر مرحله مشخص کنید و دلیل تغییر هر یک را در صورت نیاز به تشریح، مختصراً بیان کنید. ضمناً مقدار داخل هر گره برابر هزینه فاصله مستقیم تا هدف B می‌باشد.)



ترم اول ۸۹–۹۰

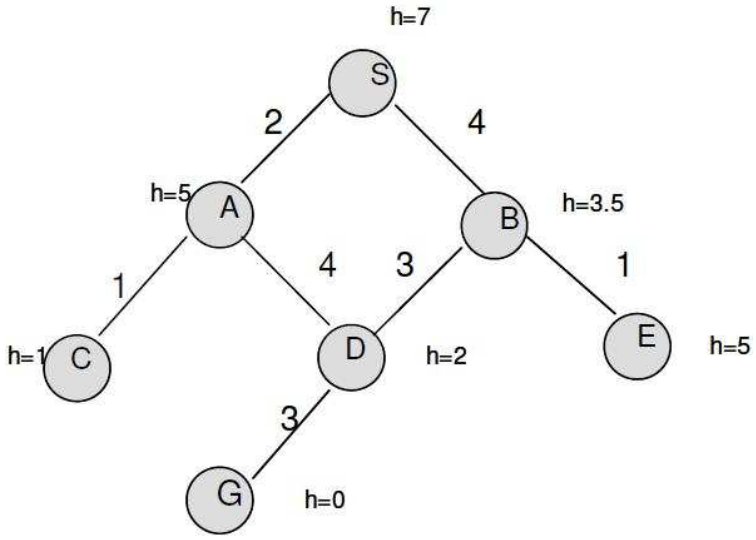
توسط الگوریتم *A با شروع از راس A، درخت جستجو را مرحله به مرحله توسعه دهید تا هدف B بدست آید (مقدار داخل هر گره برابر هزینه فاصله مستقیم تا هدف B می‌باشد.)



الف. روش *LRTA را در قالب مثالی شرح دهید. ب. به نظر شما این روش در کدام نوع از جستجوها قابل طبقه-بندی است و آیا عاملی که ازاین روش استفاده می‌کند قابلیت یادگیری دارد؟ توضیح دهید.

تابستان ۹۰

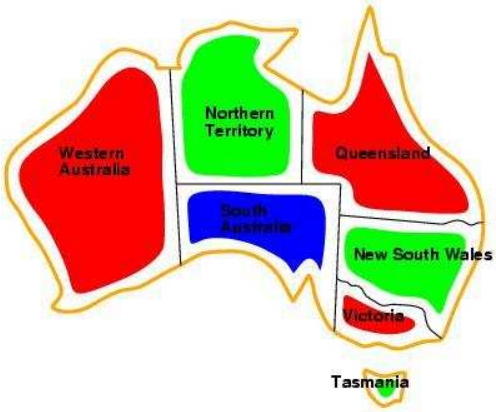
در گراف زیر، S گره شروع و G گره هدف است. عدد روی هر یال، هزینه یال را نشان می‌دهد. ارزانترین مسیر و هزینه آن در صورت استفاده از الگوریتم A^* را مشخص نمایید؟ درخت جستجو را ترسیم نمایید. (۱/۵ نمره)

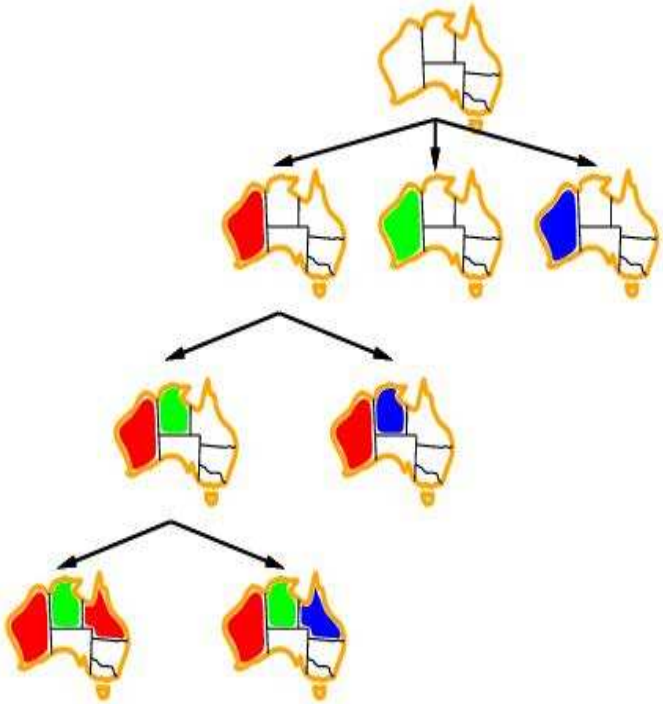


فصل پنجم: مسائل ارضای محدودیت(CSP)

آنچه در این فصل خواهید آموخت :

- ❖ **تعریف مسائل ارضای محدودیت و چند مثال**
- ❖ **جستجوی عقبگرد برای CSP**
- ❖ **کنترل روبه جلو(Forward Checking)**
- ❖ **استفاده از هیورستیک‌ها برای حل مسائل CSP**
- ❖ **جستجوی محلی برای مسائل ارضای محدودیت**





مسائل ارضای محدودیت (CSP[1]):

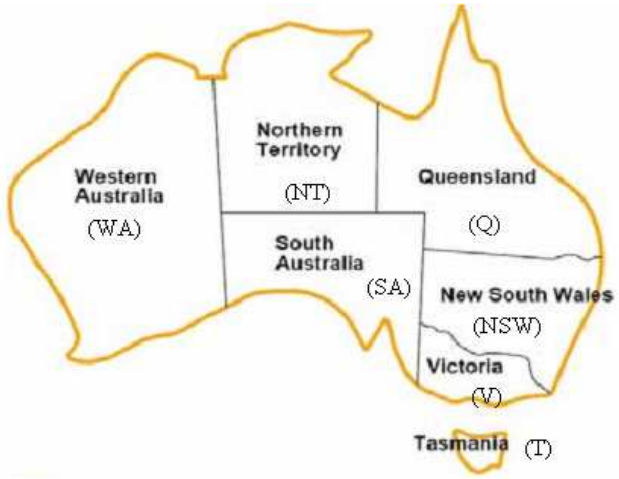
مسائل ارضای محدودیت یا **CSP** توسط مجموعه‌ای از متغیرهای $x_1, x_2, x_3, \ldots, x_n$ و مجموعه‌ای از محدودیت‌های $c_1, c_2, c_3, \ldots, c_n$ تعریف می‌شوند هر متغیر x_i دارای دامنه‌ی ناتهی D_i از مقادیر ممکن می‌باشد هر محدودیت c_i شامل تعدادی زیر مجموعه از متغیرهاست به طوری که آن محدودیت مقادیر ترکیبات مجاز این زیر مجموعه‌ها را مشخص می‌کند. هر حالت با انتساب مقادیری به چند متغیر یا تمام آن‌ها تعریف می‌شود بنابراین در حالت اولیه هیچ یک از متغیرها مقدار ندارند.

انتساب سازگار: انتسابی است که هیچ محدودیتی را نقض نمی‌کند.

انتساب کامل: انتسابی است که هر متغیری در آن باشد.

راه حل: راه حل‌ها در مسئله CSP انتساب‌های سازگار و کامل می‌باشند، یعنی یک راه حل شامل تمام متغیرهاست علاوه بر آن تمام محدودیت‌ها را نیز برآورده می‌کند. بعضی از CSPها به راه حل‌هایی نیاز دارند که تابع هدف[2] را ماکزیمم کند.

مثال:



مجموعه ای از متغیرها: {WA, NT, SA, Q, NSW, V, T}

مجموعه‌ای از مقادیر (دامنه): { آبی (B)، سبز (G)، قرمز (R)}

محدودیت‌ها: دو استان همجوار همرنگ نباشند

$$c_1: WA \neq NT, c_2: WA \neq SA, c_3: NSW \neq V$$

WA,NT={(R,B)(R,G),(B,R),(B,G),(G,R),(G,B)}

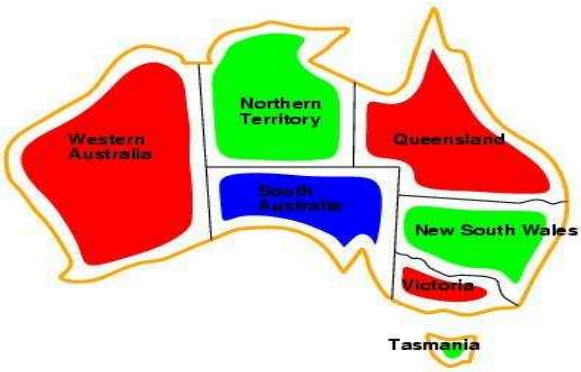
حالت: {WA = قرمز, NT = سبز, T = سبز, V = قرمز, NSW = سبز, Q = قرمز, SA = آبی}

این حالت هم سازگار و هم کامل است پس یک راه حل است.

---

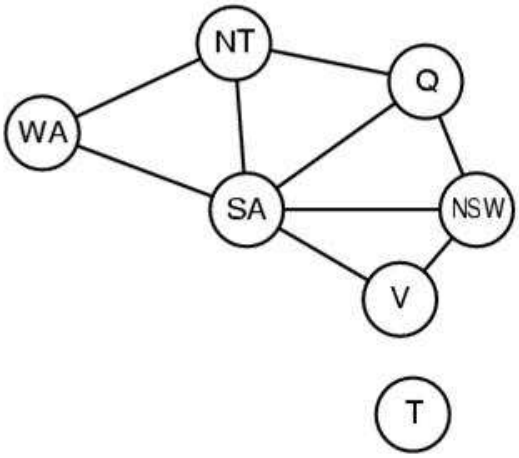
[1] constraint satisfaction problem

[2] Objective function



گراف محدودیت:

یالها: محدودیت‌ها **گره‌ها:** متغیرها



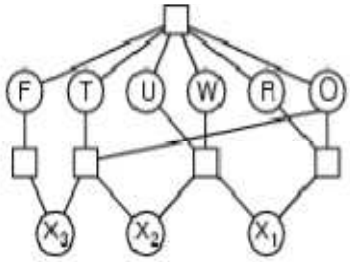
مسئله رمزنگاری[1] (معمای ریاضیات) :

هدف در مسئله معمای ریاضیات این است که در آن هر حرف یک رقم متفاوت از سایرین به خود بگیرد به طوریکه جمع ارقام انتساب یافته صحیح می‌باشد

{F = 2, O = 9, R = 7, T = 8, Y = 6, E = 5, N = 0, S = 3, I = 1, X = 4}

FORTY	29786
+ TEN	+ 850
+ TEN	+ 850
SIXTY	31486

```
  T W O
+ T W O
-------
F O U R
```



مثال:

متغیرها: $\{T, W, O, F, U, R, x_1, x_2, x_3\}$ **دامنه:** $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

محدودیت‌ها: متغیرها مقادیر متفاوت داشته باشند. $T \neq F \neq W \neq O \neq U \neq R$

All different $\{T, W, O, F, U, R\}$

---

[1] Cryptography

نکته: محدودیت All different را می‌توان به محدودیت‌های دو دویی تبدیل کرد به طور کلی اگر متغیرهای کافی برای یک مسئله CSP در نظر گرفته شود هر محدودیت مرتبه ی بالا می‌تواند به محدودیت‌های دودویی تبدیل شود.

مثال: مسئله رمزنگاری را می‌توان به صورت زیر فرموله سازی کرد:

- **حالات:** یک معمای رمزنگاری با چند حروف جایگزین شده توسط ارقام.
- **عملگرها:** وقوع هر حرف را با یک رقم جایگزین کنید که قبلاً در معما ظاهر نشده است.
- **آزمون هدف:** معما فقط شامل ارقام است ویک مجموعه از اعداد صحیح برمی‌گرداند.

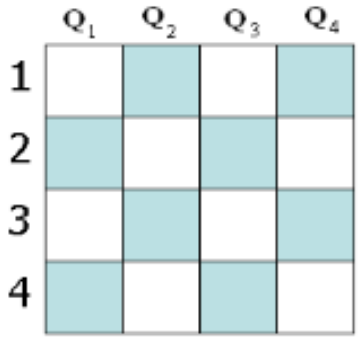
هزینه‌ی مسیر: صفر.

مسئله ۴–وزیر:

متغیرها: Q_1, Q_2, Q_3, Q_4 دامنه: {۱، ۲، ۳، ۴}

محدودیت‌ها: وزیرها نمی‌توانند در یک سطر یا ستون یا قطر قرار گیرند.

برای Q_1, Q_2: {(1,3) (1,4) (2,4) (3,1) (4,1) (4,2)}



مزایای بیان مسئله به صورت CSP :

- به دلیل نمایش استاندارد حالت‌ها (مجموعه‌ی متغیرهایی با مقادیرشان) می‌توان تابع Successor و آزمون هدف را به شکل کلی نوشت به طوریکه برای هر CSP قابل اعمال باشد.
- می‌توان هیوریستیک‌های کلی و کارایی ایجاد نمود که نیاز به تخصص اضافی در دامنه‌ی خاص مسئله نداشته باشند.

نکته:یک مسئله CSP می‌تواند با استفاده از فرموله سازی افزایشی مانند یک مسئله‌ی جستجوی استاندارد ارائه شود.

- **حالت اولیه:** انتساب خالی که در آن هیچ متغیری مقدار ندارد.
- **تابع ما بعد:** یک مقدار می تواند به هر متغیر فاقد مقدار نسبت داده شود اگر با متغیرهایی که قبلاً مقدار گرفته اند تضاد نداشته باشد.
- **آزمون هدف:** آیا انتساب فعلی کامل است یا نه ؟
- **هزینه ی مسیر:** یک هزینه ی ثابت برای هر مرحله.

نکته: هر راه حل یک انتساب کامل است لذا اگر در مسئله n متغیر وجود داشته باشد راه حل در عمق n خواهد بود و درخت جستجو دارای عمق n می‌باشد. بنابراین در بین جستجوها، جستجوی عمقی، مناسب ترین برای حل یک مسئله

CSP می‌باشد. البته اگر از فرموله سازی حالت کامل استفاده کنیم، الگوریتم‌های جستجوی محلی نیز می‌توانند برای مسائل CSP مفید باشند.

انواع مسائل CSP:

I. گسسته و متناهی:

ساده ترین نوع مسائل CSP شامل متغیرهای گسسته با دامنه‌های متناهی می‌باشد، مانند مسئله رنگ آمیزی نقشه، مسئله ۸ وزیر، مسئله رمزنگاری ریاضی، CSPهای بولین. اگر در یک مسئله CSP حداکثر اندازه دامنه هر متغیر برابر d باشد و N تعداد متغیرها باشد آنگاه تعداد انتساب‌های کامل $O(d^N)$ می‌باشد.

II. گسسته ونامتناهی :

متغیرهای گسسته ممکن است دامنه‌های نامتناهی نیز داشته باشند، مانند مجموعه‌ای از اعداد صحیح یا ر شته‌ها. به عنوان مثال در زمانبندی کارها، تاریخ شروع هر کار یک متغیر است و مقادیر ممکن آنها اعداد صحیح می‌باشند. در مسائل با دامنه‌های نامتناهی نمی‌توان محدودیت‌هارا با شمارش تمام ترکیبات مجاز مقادیر تعریف کرد به جای آن باید از زبان محدودیت استفاده شود، به عنوان مثال اگر Job_1 که ۵ روز طول می‌کشد باید قبل از Job_2 انجام شود، به یک زبان محدودیت از نامساوی‌های جبری مثل $starjob_1 + 5 \leq starjob_2$ نیاز است. علاوه براین، چنین محدودیت‌هایی را نمی‌توان با شمارش تمام انتساب‌های ممکن حل کرد، زیرا تعداد آنها نامتناهی است.

III. پیوسته :

در دنیای واقعی مسائل ارضای محدودیت با دامنه‌های پیوسته بسیار متداول هستند مثلاً زمانبندی آزمایشات روی تلسکوپ فضایی‌هابل به مشاهدات زمانی دقیقی نیاز دارد. شروع و پایان هر مشاهده متغیرهای پیوسته ای هستند که باید از قوانین نجومی تبعیت کنند. معروفترین دسته از CSPهای با دامنه‌ی پیوسته، مسائل برنامه نویسی خطی هستند که در آنها محدودیت‌ها باید به صورت نامساوی‌های خطی باشند که یک ناحیه‌ی محدب ایجاد کنند. مسائل برنامه نویسی خطی می‌تواند در زمان چند جمله‌ای بر اساس تعداد متغیرها حل شود.

انواع محدودیت‌ها:

i. **یکانی**[۱]**:** روی یک متغیر باشد، $NT \neq Red$

ii. **دوگانی**[۲]**:** محدودیت روی دو متغیر باشد، $WA \neq NT$

iii. **مرتبه بالاتر**[۳]**:** محدودیت روی سه یا بیشتر متغیر باشد، مانند مسئله رمزنگاری ریاضی

$T \neq F \neq W \neq O \neq U \neq R$

---

[۱] unary

[۲] binary

[۳] High order

نکته: محدودیت‌هایی که ذکر شد محدودیت‌های کامل یا مطلق[۱] نامیده می‌شوند که نقض یک محدودیت مطلق به معنای حذف یک راه حل بالقوه می‌باشد.

محدودیت‌های اولویت دار[۲]:

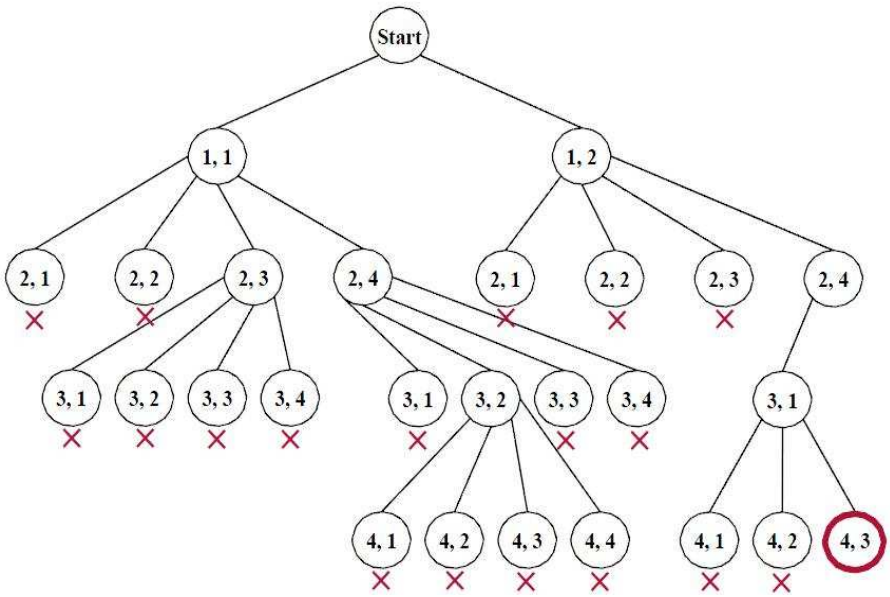
در این محدودیت‌ها یک راه حل بر سایر راه حل‌ها ترجیح داده می‌شود. به عنوان مثال در برنامه زمانی دانشگاه پروفسور **x** ترجیح می‌دهد صبح تدریس کند، در حالیکه پروفسور **y** ترجیح می‌دهد، بعد ازظهر تدریس کند. اگر جدول زمانی به گونه‌ای تنظیم شود که پروفسور **x** در ساعت ۲ بعد از ظهر تدریس کند یک راه حل محسوب می‌شود اما راه حل بهینه نیست. محدودیت‌های اولویت دار می‌توانند به صورت هزینه در انتساب هر یک از متغیرها اعمال شوند.

نکته: مسائل زیر برای حل به روش CSP مناسبند: مسئله زمانبندی، ۸ وزیر، حل جدول کلمات متقاطع، رنگ آمیزی نقشه، معمای ریاضی (رمزنگاری)، زمانبندی امتحانات، چینش ماژولها روی یک تراشه و مسائل دنیای واقعی از قبیل: مسائل انتسابی، مانند اینکه چه کسی چه کلاسی را درس می‌دهد، مسائل زمانبندی حمل ونقل و زمانبندی کارخانه.

جستجوی عقبگرد[۳] برای CSP:

اصطلاح جستجوی عقبگرد برای جستجوی عمقی به کار می‌رود، که در هر سطح یک متغیر را مقدار می‌دهد و وقتی مقدار معتبری برای انتساب به یک متغیر وجود نداشته باشد به عقب برمی‌گردد. جستجوی عقبگرد یک الگوریتم ناآگاهانه است که برای حل مسائل بزرگ ناکارامد است. به مثال زیر توجه کنید.

حل مساله ۴–وزیر به روش عقبگرد



---

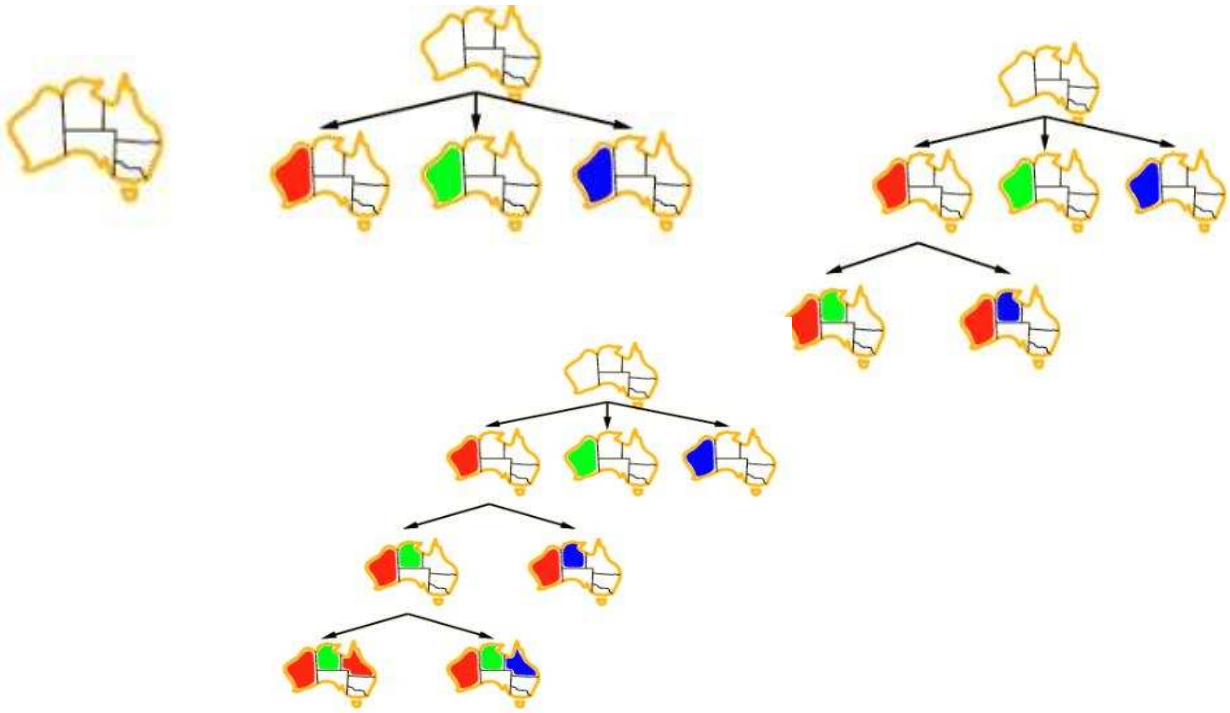
[۱] Absolute

[۲] preference constraints

[۳] Back Tracking Search

راه حل: {(1,2) (2,4) (3,1) (4,3)}

حل مساله رنگ آمیزی نقشه به روش عقبگرد



ترتیب انتخاب متغیرها :

I. **مقادیر باقیمانده کمینه[۱] (MRV) یا محدودیت ترین متغیر[۲] یا اولین شکست[۳]**

ایده: متغیری را انتخاب کن که کمترین مقادیر معتبر را دارد.

II. **اکتشاف درجه‌ای :**

ایده: متغیر با بیشترین درجه اول انتخاب می‌شود.

III. **اکتشاف مقدار با کمترین محدودیت[۴]**

ایده: متغیری انتخاب می‌شود که کمترین مقادیر را از متغیرهای باقیمانده حذف می‌کند.

---

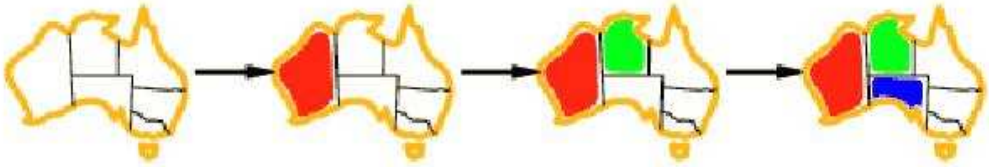
[۱] Minimum Remaining Value

[۲] Most Constraint Value

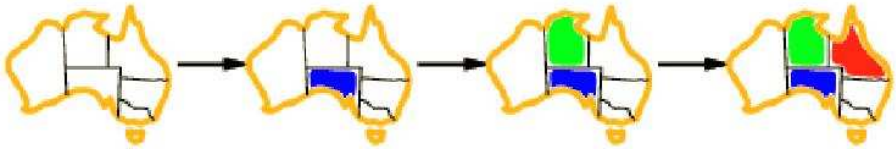
[۳] first failure

[۴] least constraint value

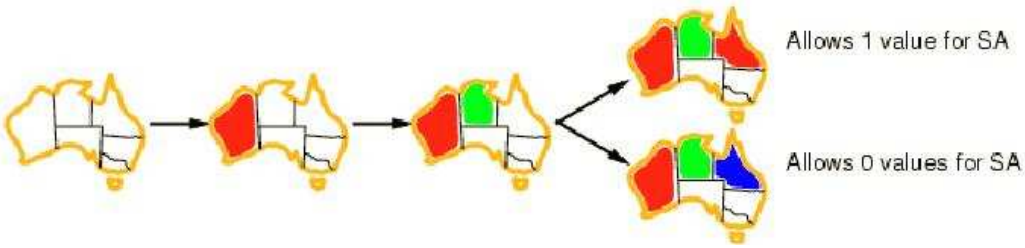
❖ **MRV:** ایده این روش این است که متغیری را انتخاب می‌کند که کمترین مقادیر معتبر را دارد یا به عبارتی محدودترین متغیر را انتخاب می‌کند. محدودترین متغیر، متغیری است که کوچکترین دامنه را دارد در این صورت متغیری انتخاب می‌شود که به احتمال زیاد با شکست مواجه شده و درخت جستجو با هرس مواجه خواهد شد.



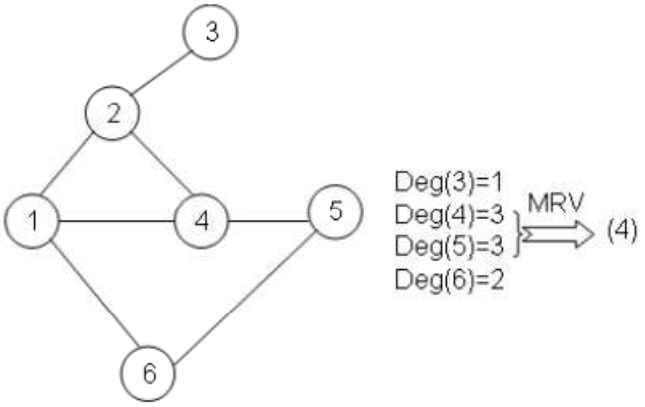
❖ **اکتشاف درجه ای:** سعی می‌کند فاکتور انشعاب را در انتخاب آینده کم کند. متغیری انتخاب می‌شود که بیشترین محدودیت را روی متغیرهای انتساب نیافته اعمال می‌کند. این هیوریستیک برای انتخاب اولین متغیر جهت انتساب مقدار مناسب است.



❖ **اکتشاف مقدار با کمترین محدودیت:** این روش مقداری را ترجیح می‌دهد که در گراف محدودیت، متغیرهای همسایه به ندرت آنرا انتخاب می‌کنند و سعی در ایجاد بیشترین قابلیت انعطاف برای انتساب بعدی متغیرها دارد. ایده‌ی این روش برای یک متغیر این است که مقداری را که کمترین محدودیت را ایجاد می‌کند انتخاب می‌کند یا مقداری را که کمترین مقادیر را از متغیرهای باقیمانده حذف می‌کند.



مثال: در صورتی که بخواهیم با استفاده از روش CSP گراف مقابل را با سه رنگ **R,G,B** رنگ آمیزی کنیم بعد از رنگ آمیزی رئوس ۱و ۲ کدام راس رنگ آمیزی خواهد شد؟ (مهندسی کامپیوتر-۸۷)



بررسی پیشرو[1] (FC)

- **ایده:** وقتی متغیری مقدار گرفت آن مقدار از همسایگانش حذف می‌شود.
- **شرط خاتمه:** الگوریتم هنگامی خاتمه می‌یابد که برای یک متغیر هیچ مقدار مجاز باقی نمانده باشد.

مثال:

- **گام اول:**

WA			NT			Q			NSW			V			SA			T		
R	G	B	R	G	B	R	G	B	R	G	B	R	G	B	R	G	B	R	G	B

- **گام دوم:**

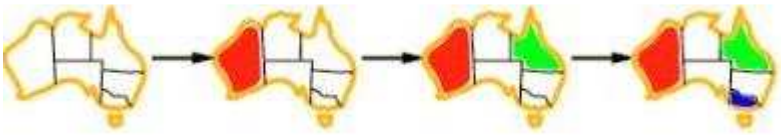
WA			NT			Q			NSW			V			SA			T		
R	G	B	R	G	B	R	G	B	R	G	B	R	G	B	R	G	B	R	G	B
R				G	B	R	G	B	R	G	B	R	G	B		G	B	R	G	B

[1] Forward Checking

❖ **گام سوم:**

WA			NT			Q			NSW			V			SA			T		
R	G	B	R	G	B	R	G	B	R	G	B	R	G	B	R	G	B	R	G	B
R				G	B	R	G	B	R	G	B	R	G	B		G	B	R	G	B
R					B	G			R		B	R	G	B			B	R	G	B

❖ **گام چهارم:**



WA			NT			Q			NSW			V			SA			T		
R	G	B	R	G	B	R	G	B	R	G	B	R	G	B	R	G	B	R	G	B
R				G	B	R	G	B	R	G	B	R	G	B		G	B	R	G	B
R					B	G			R		B	R	G	B			B	R	G	B
R					B	G			R					B				R	G	B

به متغیری رسیدیم که هیچ مقدار مجازی برایش باقی نمانده(متغیر **V**) پس الگوریتم خاتمه می یابد.

جدول FC:

	WA	NT	Q	NSW	V	SA	T
Initial domains	R G B	R G B	R G B	R G B	R G B	R G B	R G B
After WA=red	R	G B	R G B	R G B	R G B	G B	R G B
After Q=green	R	B	G	R B	R G B	B	R G B
After V=blue	R	B	G	R	B		R G B

وقتی انتساب به X صورت می‌گیرد فرایند بررسی پیشرو متغیرهای بدون انتساب مثل y را در نظر می‌گیرد که از طریق یک محدودیت به X متصل است و هر مقداری را که با مقدار انتخاب شده برای **x** برابر است از دامنه **y** حذف می‌کنیم. زمانی که یک متغیر هیچ مقدار مجاز باقیمانده ندارد جستجو خاتمه می‌یابد پس از انتساب آبی=V، دامنه SA خالی می‌شود. لذا بررسی پیشرو تشخیص می‌دهد که انتساب « **WA = R, Q = G, V = B** » با محدودیت‌های مساله سازگار نیست و الگوریتم به سرعت به عقب بر می‌گردد. این روش جستجو، تناقض را سریع تر از جستجوی عقب گرد ساده پیدا می‌کند.

پخش (انتشار) محدودیت:

اگر چه بررسی پیشرو بسیاری از ناسازگاری‌ها را کشف می‌کند، اما همه آن‌ها را نمی‌تواند تشخیص بدهد. برای مثال در جدول FC سطر سوم وقتی WA = R و Q = G است. SA, NT فقط می‌توانند آبی باشد. اما این دو متغیر همجوارند و نمی‌توانند رنگ یکسانی داشته باشند. جستجوی بررسی پیشرو این مورد را به عنوان ناسازگاری تشخیص نمی‌دهد. بنابراین بررسی پیشرو محدودیت‌ها را از متغیرهای انتساب یافته به متغیرهای انتساب نیافته منتشر می‌کند اما تمام شکست‌ها را نمی‌تواند در زودترین زمان ممکن تشخیص بدهد. پخش محدودیت یعنی تاثیر محدودیت روی یک متغیر بر سایر متغیرها. انتشار محدودیت به طور مکرر بر محدودیت‌ها به طور محلی تاکید دارد. در مثال قبل ابتدا اطلاعات محدودیت از **G, WA** به ترتیب به **NT, SA** پخش شد.

WA			NT			Q			NSW			V			SA			T		
R	G	B	R	G	B	R	G	B	R	G	B	R	G	B	R	G	B	R	G	B
R				G	B	R	G	B	R	G	B	R	G	B		G	B	R	G	B
R					B	G			R		B	R	G	B			B	R	G	B

سازگاری یال[1] :

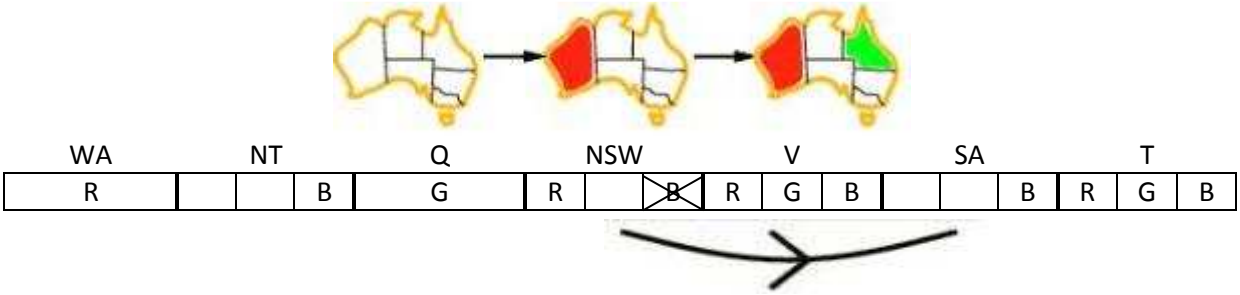
ایده سازگاری یال یک روش صحیح برای پخش محدودیت‌ها فراهم می‌کند و در نتیجه بسیار قوی تر از جستجوی بررسی پیشرو است. منظور از ARC یال جهت دار در گراف محدودیت است. یال y→x سازگار است اگر و فقط اگر به ازای هر مقدار **X**، بعضی مقادیر مجاز برای **y** موجود باشد.

مثال ۱: SA → NSW سازگار است اگر SA = blue و NSW = red.

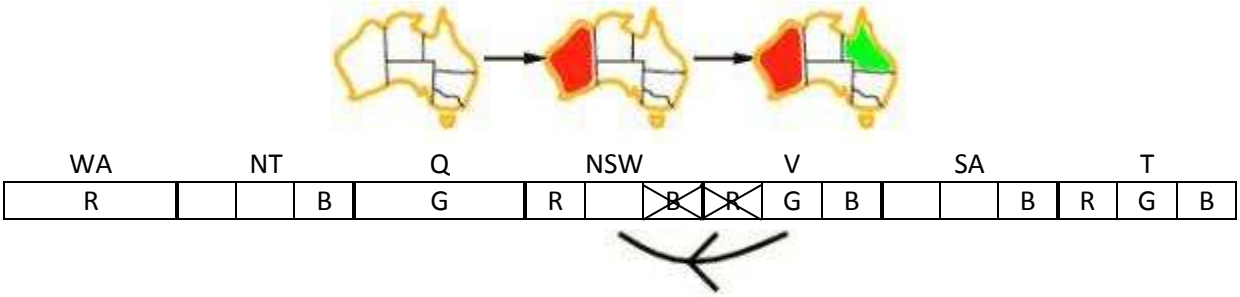
WA			NT			Q			NSW			V			SA			T		
R					B	G			R		B	R	G	B			B	R	G	B

[1] Arc consistency

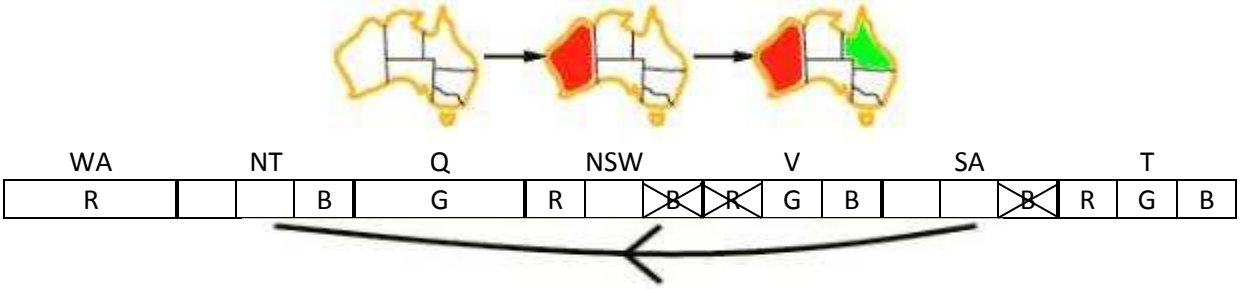
مثال ۲: NSW → SA سازگار می‌شود با حذف blue از NSW.



مثال ۳: V → NSW سازگار می‌شود با حذف blue از NSW و red از V.



مثال ۴: SA→NT سازگار می‌شود با حذف blue از NSW,SA و red از V.



نکته: بررسی Arc consistency می‌تواند به صورت یک مرحله پیش پردازش قبل از شروع جستجو انجام شود یا به صورت یک مرحله پخش مانند بررسی پیشرو پس از هر انتساب در حین جستجو که به آن روش MAC[1] گفته می‌شود، انجام شود. در هر موقع این فرایند باید به صورت مکرر انجام گیرد تا هیچ ناسازگاری باقی نماند، زیرا هر زمان که به منظور رفع ناسازگاری یال یک مقدار از دامنه متغیری حذف می‌شود، ممکن است در یال‌هایی که به آن متغیر اشاره می‌کنند ناسازگاری جدید ایجاد شود. پیچیدگی زمانی Arc consistency در بدترین حالت $O(n^2d^3)$ می‌باشد که در آن d تعداد مقادیر دامنه و n تعداد متغیرهاست. اگر چه روش سازگاری یال نسبت به بررسی پیشرو هزینه ی بیشتری دارد اما مفیدتر است. زیرا تناقض (عدم موفقیت) را زودتر از روش بررسی پیشرو تشخیص می‌دهد.

سازگاری k (K- consistency) :

سازگاری یال، تمام ناسازگاری‌های ممکن را نمی‌تواند تشخیص دهد. مثلاً انتساب NSW = R, WA = R ناسازگار است، ولی سازگاری یال نمی‌تواند آن را تشخیص دهد. بنابراین به کمک سازگاری K شکل قوی تری از پخش محدودیت را می‌تواند تعریف کرد.

---

[1] Maintaining Arc Consistency

تعریف K- consistence :

یک مساله ارضای محدودیت K- consistence است اگر برای هر k-1 متغیر و برای هر انتساب سازگار آن متغیرها، همیشه یک مقدار سازگار برای هر **K**امین متغیر وجود داشته باشد. به عنوان مثال معنای 1_consistence این است که هر متغیر با خودش سازگار است این سازگاری Node consistence نامیده می‌شود. 2_consistence همان سازگاری یال است. معنای 3_consistence این است که هر جفت از متغیرهای همجوار می‌توانند به سومین متغیر همسایه گسترش یابند. این سازگاری، سازگاری مسیر یا Path consistence نامیده می‌شوند. یک گراف قویاً K- consistence است هرگاه 1-consistence, 2-consistence و ..... k-2consistence و k-1consistence باشد. اگر گراف قویاً K_consistence باشد می‌توان مساله را بدون عقب گرد حل کرد، که پیچیدگی زمانی آن O(nd) است

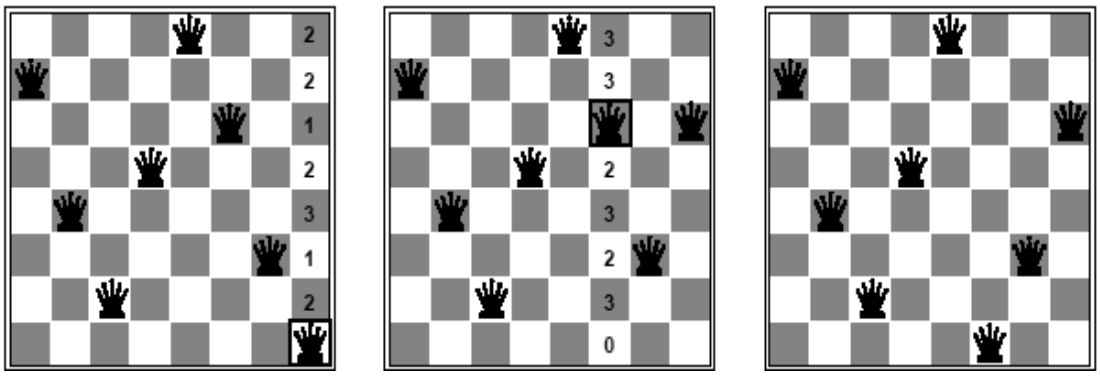
ترتیب تشخیص تناقض:

عقب گرد < FC < ARC consistence < K- consistence < k- consistence قویاً

جستجوی محلی برای CSP:

الگوریتم‌های جستجوی محلی مانند تپه نوردی و simulated annealing، بسیاری ازمسائل CSP را به طور کارایی حل می‌کنند. آنها از فرموله سازی حالت کامل استفاده می‌کنند، یعنی تمام متغیرها باید دارای مقدار باشد. حالت اولیه به همه متغیرها مقداری را نسبت می‌دهد و تابع مابعد در هر مرحله مقدار یک متغیر را تغییر می‌دهد. به عنوان مثال در مسئله۸- وزیر حالت اولیه آن یک ترکیب تصادفی از مکان هشت وزیر در هشت ستون است و تابع مابعد یک وزیر را انتخاب نموده و به جای دیگری در همان ستون انتقال می‌دهد. در انتخاب مقدار جدید برای یک متغیر، **هیوریستیک مینیمم تناقضات**[1]، بهترین هیوریستیک است. این تابع مقداری را انتخاب می‌کند که کمترین برخورد را با سایر متغیرها ایجاد می‌کند، یعنی مقداری را انتخاب کن که کمترین تعداد محدودیت‌ها را نقض کند. زمان اجرای **هیورستیک مینیمم تناقضات**، مستقل از اندازه مسئله خواهد بود.

نکته: هیورستیک مینیمم تناقضات برای حل مسائل سخت مانند زمان بندی مشاهدات تلسکوپ‌هابل مورد استفاده قرار می گیرد. مزیت دیگر جستجوی محلی این است که می‌تواند در صورت تغییر در مسئله تنظیمات را به صورت آنلاین انجام دهد.



[1] Min-conflicts Heuristic

سوالات تستی آخر فصل

ترم اول ۸۷–۸۸

۱. در مسأله ارضا محدودیت‌ها Constraint Satisfaction Problem (CSP) کدام مورد صحیح نیست؟

الف. انتخاب متغیری با بزرگترین دامنه مقادیر مجاز

ب. انتخاب متغیری با کوچکترین دامنه مقادیر مجاز

ج. انتخاب متغیری که کمترین میزان مقادیر مجاز را از دامنه سایر متغیرها حذف کند.

د. انتخاب متغیری که بیشترین میزان مقادیر مجاز را از دامنه سایر متغیرها حذف کند.

ترم دوم ۸۷–۸۸

۲. اگر بخواهیم پاسخ یک مسئله ارضا محدودیت را بیابیم کدامیک از روش‌های جستجوی زیر مناسب ترین است؟

الف. تپه نوردی ب. جستجوی اول عمق

ج. A* د. جستجوی هزینه یکسان

۳. می‌خواهیم حداکثر با ۴رنگ، یک نقشه را رنگ امیزی کنیم بطوریکه نواحی همسایه همرنگ نباشند.این مسئله را جزو کدامیک از دسته مسئله‌های زیر می‌توان در نظر گرفت؟

الف. مسئله تعدیل شده (relaxed) ب. مسائل هیوریستیک

ج. مسائل احتمالی د. مسائل ارضای محدودیت

۴. کدام گزینه در مورد یک گراف با سازگاری شدید مرتبه K صحیح نیست ؟

الف. دارای سازگاری مرتبه 1, ……., K-1, k می باشد.

ب. در این گراف بدون انجام پس گرد می‌توان مسئله را حل کرد.

ج. می‌توان تضمین کرد که راه حل مسئله حداکثر با مرتبه زمانی O(nd) پیدا می‌شود.

د. الگوریتمی که سازگاری مرتبه n را بررسی کند از مرتبه خطی است.

ترم تابستان ۸۸

۵. کدام گزینه در مورد CSP ها صحیح است؟

الف. اکثر CSPها، جابجایی پذیر هستند.

ب. یک الگوریتم استاندارد جستجو ممکن است به یک مسئله CSP قابل اعمال نباشد.

ج. ساده ترین نوع CSP شامل متغیرهای گسسته با دامنه‌های نامحدود می‌باشد.

د. یک راه حل برای CSP، می‌تواند یک انتساب ناکامل باشد.

۶. در جستجوی پسگرد برای CSP ها ،کدام گزینه برای انتخاب اولین متغیر جهت انتساب مقدار ، مؤثرتر می باشد؟

الف. هیوریستیک MRV ب.سازگاری کمان ج. انتشار محدودیت د. هیوریستیک درجه

۷. برای مسئله با گرافی که دارای سازگاری شدید مرتبه k باشد، کدام گزینه نادرست است؟

الف. دارای سازگاری مرتبه $1,\ldots,k-1,k$ می‌باشد.

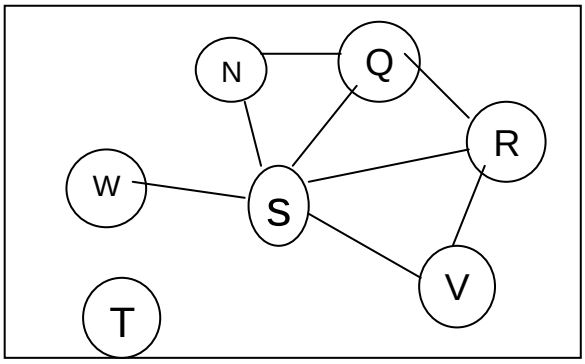
ب. بدون انجام پسگرد قابل حل است.

ج. الگوریتمی که بخواهد سازگاری مرتبه n را بررسی کند، حداکثر به مرتبه زمانی $O(n^2)$ نیاز دارد.

د. می‌توان راه حل مسئله را در حداکثر زمان $O(nd)$ محاسبه نمود.

ترم اول ۸۸–۸۹

۸. در مسئله رنگ آمیزی گراف مقابل (با سه رنگ)، با فرض اینکه در سطح اول درخت جستجوی BT + MRV، به متغیر W رنگ قرمز انتساب یافته باشد، اولین متغیری که در سطح بعد جهت رنگ امیزی انتخاب خواهد شد کدام است؟



الف. S　　ب. N یا V　　ج. Q یا R　　د. T

۹. برای گراف سؤال ۸، در صورتی که در جستجوی پیشرو مقادیر B, G, R به معنای رنگ‌های قرمز، سبز و آبی باشندو ® به معنای انتساب رنگ قرمز به متغیر باشد، با توجه به گراف و جدول زیر، کدام کمان سازگار است؟

W	N	Q	R	S	V	T
®	R B	G	R B	B	R G B	R G B

الف. R به S　　ب. S به R　　ج. N به S　　د. V به S

۱۰. اگر حداکثر ظرفیت‌های پروازهای شماره ۲۷۱ و ۲۷۲ به ترتیب ۱۶۵ و۳۸۵ نفر باشند (یعنی Flight272 ∈ [0,385] و Flight 271 ∈ [0,165]) با شرط اینکه مجموع تعداد مسافرینی که به وسیله هردو پرواز انتقال می‌یابند باید ۴۲۰ نفر باشد، کران‌های سازگار برای دو پرواز کدامند؟

الف. Flight 272 ∈ [0,255]
Flight 271 ∈ [0,165]

ب. Flight 272 ∈ [0,255]
Flight 271 ∈ [0,165]

ج. Flight 272 ∈ [165,255]
Flight 271 ∈ [0,165]

د. Flight 272 ∈ [255,385]
Flight 271 ∈ [35,165]

ترم دوم ۸۸–۸۹

۱۱. برای حل مسئله ۸ وزیر به روش جستجوی تپه نوردی کدام روش مناسب است؟ (منظور از فرمول بندی افزایشی این است که در حالت شروع همه وزیرها در صفحه قرار ندارند و به تدریج اضافه می‌شوند.)

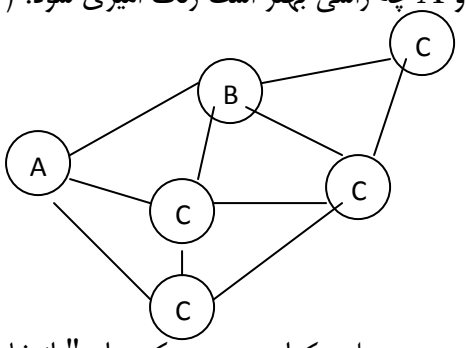
الف. فرمول بندی حالت کامل با هیوریستیک تعدادجفت وزیرهایی که به هم حمله می‌کنند.

ب. فرمول بندی حالت کامل با هیوریستیک تعدادجفت وزیرهایی که به هم حمله نمی‌کنند.

ج. فرمول بندی افزایشی با هیوریستیک تعدادجفت وزیرهایی که به هم حمله می‌کنند.

د. فرمول بندی افزایشی با هیوریستیک تعدادجفت وزیرهایی که به هم حمله نمی‌کنند.

۱۲. در مساله رنگ آمیزی گراف زیر، بعد از رنگ امیزی رئوسB و A چه راسی بهتر است رنگ امیزی شود؟ (باتوجه به هیوریستیکMRV)



الف. C

ب. D

ج. E

د. F

۱۳. در حل یک مسئله ارضای محدودیت (CSP) به روش جستجوی محلی، کدام هیوریستیک برای" انتخاب یک مقدار برای متغیر انتخاب شده" استفاده می‌شود.؟

الف. هیوریستیک حداقل مقادیر باقی مانده (MRV)

ب. هیوریستیک حداقل تناقضات

ج. هیوریستیک متغیر با حداکثر محدودیت

د. هیوریستیک درجه

۱۴. در حل یک CSP که با روش فرمول بندی افزایشی تعریف شده است کدام جستجو مناسب عمل می‌کند؟

الف. اول عمق ب. اول سطح ج.جستجوی محلی د.*A

۱۵. کدامیک از مسائل زیر برای حل توسط الگوریتم MIN_CONFLICT مناسب نیست؟

الف. مسائل زمان بندی ب. n وزیر

ج. زمان بندی رصدهای فضایی د. مساله فروشنده دوره گرد

ترم تابستان ۸۹

۱۶. تدوین حالت کامل (که هر حالت یک انتساب کامل است) در مسائل CSP در کدام روش‌های جستجو می‌توانند مفید باشند؟

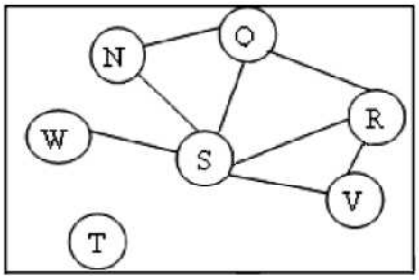
الف. جستجوهای پس رو ب. جستجوهای پیش رو ج. پس رو + MRV د. جستجوهای محلی

۱۷. در محدودیت منبع (محدودیت از درجه بالا) برای Almost(10,PA1,PA2 ,PA3,PA4) (یعنی حداکثر ۱۰ نفر برای ۴ کار نیاز داریم) کدام یک از مجموعه‌های زیر اگر به عنوان دامنه برای هر یک از ۴ متغیر در نظر گرفته شود سازگار خواهند بود؟

الف. {۶ و ۵ و ۴ و ۳} ب. {۶ و ۵ و ۴ و ۳ و ۲} ج. {۴ و ۳ و ۲} د.{۵ و ۴ و ۳ و ۲}

ترم اول ۸۹–۹۰

۱۸. اگر در گراف زیر، از هیوریستیک مقدار با حداقل محدودیت در جستجوی پس رو استفاده شود و به ترتیب انتساب-های W = red, N = green را انجام داده‌ایم، در انتساب بعدی به Q چه مقداری را تخصیص خواهیم داد؟ (CSP)



الف. Blue ب. Green ج. Red د. مقدار قابل تخصیصی وجود ندارد.

۱۹. اگر در پس گرد هوشمندانه مقدار دهی به متغیرها به ترتیب Q, R, V, T, S, W, N, T (از چپ به راست) صورت گیرد و انتساب مقابل صورت گرفته باشد:{Q = red, R = green, V = blue, T = red} مجموع تناقض برای S کدام است؟

الف.{Q, R, V} ب.{Q, R} ج.{R, V} د.{Q, V}

۲۰. در سوال قبل پس از پرش رو به عقب، به دنبال مقدار جدیدی برای کدام متغیر خواهیم بود؟

الف. Q ب. V ج. S د. T

ترم دوم ۸۹–۹۰

۲۱. در مورد سازگاری کدام گزینه صحیح نیست؟

الف. وارسی سازگاری کمان می‌تواند پس از هر انتساب مقدار و در خلال جستجو به صورت یک مرحله انتشار محدودیت اعمال شود.

ب. وقتی که یکی مقادیر مرتبط به دامنه یک متغیر جهت رفع ناسازگاری حذف می‌شود، ممکن است باعث ناسازگاری-های جدیدی در کمان‌های متصل به متغیر ایجاد شود.

ج.پس از اعمال الگوریتم AC-3 (سازگاری کمان) تمام کمان‌ها سازگار می‌شوند

د.در بدترین شرایط پیچیدگی زمانی الگوریتم AC-3 $O(n^2 d^3)$ خواهد بود.

۲۲. در حل مسئله حساب رمزی TOW + TOW = FOUR چنانچه ابر گراف محدودیت را ترسیم نمایید. تعداد متغیرهای کمکی لازم و تعداد ابر یالها چند است؟

الف. ۴ و ۵ ب. ۵ و ۳ ج. ۴ و ۴ د. ۳ و ۵

۲۳. در حل مسائل CSP برای انتخاب متغیر و انتخاب مقدار کدام مورد هیوریستیک درستی است؟

الف. بیش از همه محدوده شده است. بیش از همه محدود کننده است

ب.بیش از همه محدوده شده است. کمتر از همه محدود کننده است

ج. کمتر از همه محدوده شده است. بیش از همه محدود کننده است

د. کمتر از همه محدوده شده است. کمتر از همه محدود کننده است

۲۴. در مسائل CSP با استفاده از ..... متغیر کمکی می‌توان یک محدودیت ۳ گانه مثل $A + B = C$ را به ..... محدودیت دوگانه تبدیل کرد.

الف. ۱ و ۳ ب. ۱ و ۲ ج. ۲ و ۲ د. ۲ و ۳

تابستان ۹۰

۲۵. در کدام روش، متغیری با بالاترین محدودیت در مقایسه با سایر متغیرهای انتساب داده نشده، انتخاب می‌شود؟

الف. هیورستیک درجه ب. سازگاری یال

ج. عقبگرد د. هیورستیک حداقل مقادیر باقیمانده (MRV)

پاسخ نامه تستی

سوال	گزینه صحیح	سوال	گزینه صحیح
۱	ب	۱۵	د
۲	ب	۱۶	د
۳	د	۱۷	ج
۴	د	۱۸	ج
۵	ب	۱۹	الف
۶	د	۲۰	ب
۷	ج	۲۱	
۸	الف	۲۲	
۹	ب	۲۳	
۱۰	د	۲۴	
۱۱	الف	۲۵	الف
۱۲	ب	۲۶	
۱۳	ب	۲۷	
۱۴	الف	۲۸	

سوالات تشریحی آخر فصل

ترم اول ۸۷–۸۸

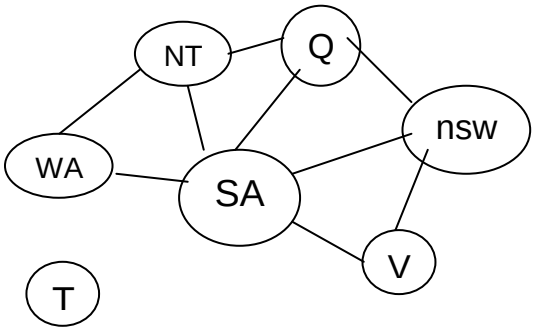
الگوریتم پس گرد ساده را برای مسائل ارضای محدودیت شرح دهید

ترم اول ۸۸–۸۹

برای مسئله رنگ آمیزی گراف (باسه رنگ) در انتخاب دو متغیر اول از هیوریستیک (آروینی) درجه وبرای بقیه متغیرها از هیوریستیک (آروینی) MRV انتخاب استفاده نمایید و جدول زیر را برای وارسی پیشرو تکمیل کنید. (۱/۵ نمره)

نکته ۱:در هر مرحله با انتساب مقدار به یک متغیر، یکی از سطرهای جدول تکمیل می‌شود. برای انتساب رنگ قرمز به یک متغیر در سلول مربوط به آن ® را بنویسید.

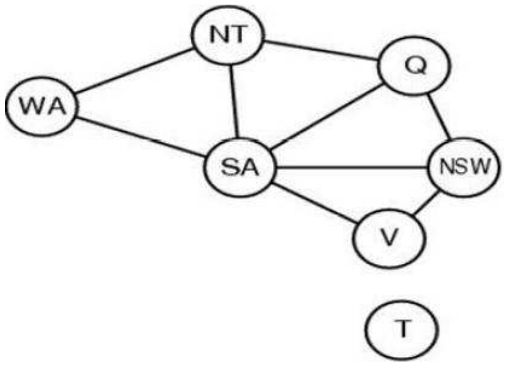
نکته ۲:در صورتی که برای دو متغیر مقادیر هیوریستیک‌ها (آروینها) یکسان است یکی را به صورت اتفاقی انتخاب کنید.



	WA	NT	Q	NSW	V	SA	T
Initial domains	RGB	RGB	RGB	RGB	RGB	RGB	RGB

تابستان ۹۰

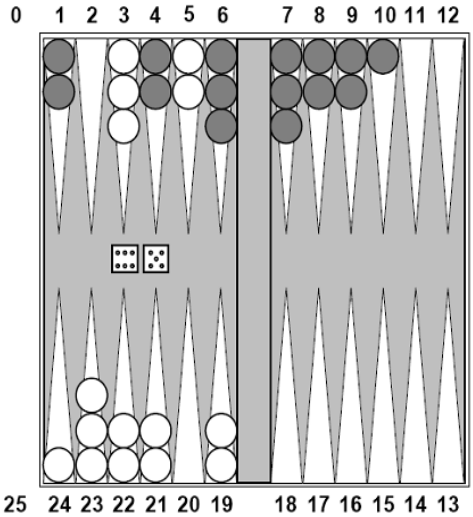
رنگ آمیزی نقشه زیر را بعنوان یک مسئله ارضای محدودیت در نظر بگیرید. با هدف انتساب رنگ‌ها بصورتی که دو ناحیه مجاور همرنگ نباشند و با استفاده از رنگ‌های قرمز، سبز، آبی و الگوریتم پیش رو، رنگ آمیزی نقشه را انجام دهید. (۱/۵ نمره)

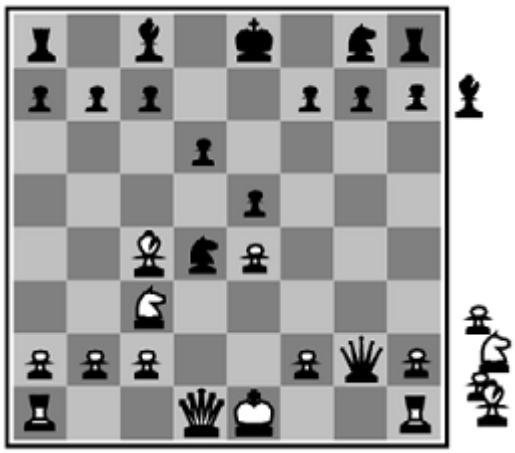


فصل ششم: جستجوی رقابتی

آنچه در این فصل خواهید آموخت:

- **بازی‌ها و دلایل مطالعه آنها**
- **الگوریتم Mini-Max**
- **بازی‌های چند نفره**
- **هرس آلفا – بتا**
- **تصمیمات بلادرنگ ناقص**
- **بازیهایی که حاوی عنصر شانس هستند**





مقدمه:

بازی‌ها حالتی از محیط‌های چند عامله هستند. در محیط‌های چند عامله، هر عامل باید اعمال سایر عامل‌ها و چگونگی تاثیر آن‌ها را مورد توجه قرار دهد. به تفاوت بین محیط‌های چند عامله رقابتی و چند عامله همکار توجه داشته باشید، محیط‌های رقابتی که در آن اهداف عامل در تناقض با یکدیگر است، منجر به مسایل جست و جوی خصمانه می‌شود، این مسایل اغلب به عنوان بازی‌ها شناخته می‌شوند. نظریه ریاضیات بازی یک شاخه از علم اقتصاد است، در اکثر بازی‌ها دو عامل وجود دارند که اعمال آن‌ها به صورت یک در میان انجام می‌شوند. در اکثر بازی‌ها دو عامل وجود دارند که اعمال آن‌ها به صورت یک در میان انجام می‌شوند و مقادیر سودمندی در حالات انتهای بازی، مساوی و مخالف هم می‌باشند. برای مثال اگر بازیکن یک بازی شطرنج را ببرد ۱+ و اگر ببازد ۱− به آن تخصیص داده می‌شود.

دلایل مطالعه بازی‌ها عبارتند از:

i. قابلیت هوشمندی انسان‌ها را به کار می‌گیرند.

ii. به دلیل ماهیت انتزاعی بازی‌ها

iii. حالت بازی‌ها را به راحتی می‌توان نشان داد و عامل‌ها معمولاً به مجموعه‌ای کوچکی از اعمال محدود هستند که نتایج آن‌ها با قوانین دقیق تعریف شده‌اند.

انواع بازی‌ها:

i. **شطرنج:** بازی‌ای است که بر روی یک صفحه انجام می‌شود و برای دو بازیگر است که شانزده مهره را با توجه به قوانینی معین حرکت می‌دهند. هدف مات کردن شاه طرف مقابل می‌باشد.

ii. **بازی چکرز:** صفحه‌ی بازی چکرز برای دو نفر می‌باشد که هر نفر دارای دوازده مهره می‌باشد، هدف پریدن و دستگیر نمودن مهره‌های حریف می‌باشد.

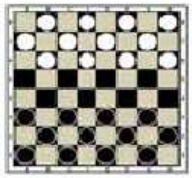
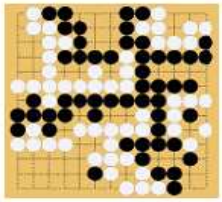
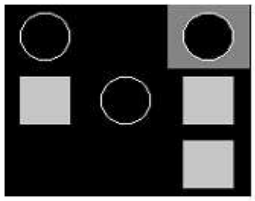
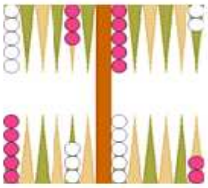
iii. **بازی go:** بازی‌ای بر روی صفحه برای دو بازیگر می‌باشد که شمارنده‌هایی را روی یک شبکه قرار می‌دهند، هدف محاصره کردن و سپس دستگیر کردن شمارنده‌های حریف می‌باشد.

iv. **تیک – تاک – تو کور:** بازی تیک – تاک – تو دارای دو بازیگر می‌باشد. برای هر دو بازیگر هدف این است که اولین کسی باشند که سه شئ همانند را در یک ردیف، ستون یا قطر قرار می‌دهند. صفحه این بازی دارای یک شبکه‌ی سه در سه می‌باشد. بنابراین دارای نه خانه می‌باشد.

v. **تخته نرد:** بازی‌ای بر روی صفحه است و دو بازیگر دارد. هر بازیگر دارای پانزده مهره می‌باشد که آن‌ها را در بیست و چهار خانه‌ی مثلثی شکل با توجه به عدد ظاهر شده روی دو تاس حرکت می‌دهد.

vi. **پل:** یک بازی کارتی حقه‌ای برای چهار نفر می‌باشد که این چهار نفر به صورت دو گروه دو نفری با هم بازی می‌کنند و در هر طرف هر گروه مقابل هم می‌نشینند. بازی پل دارای دو مرحله می‌باشد: پیشنهاد و بازی.

vii. **پوکر:** بازی‌ای کارتی است، محبوب ترین بازی از یک دسته از بازی‌ها به نام بازی‌های همچشمی می‌باشد که در آن بازیگران با کارت‌هایی کاملاً پنهان یا اندکی پنهان روی یک چیزی شرط بندی می‌کنند، سپس چیزی که شرط بندی روی آن انجام شده است به بازیگر یا بازیگران باقی مانده‌ای که دارای بهترین ترکیب کارت‌ها هستند جایزه داده می‌شود. در زیر تصویر صفحه‌ی چند بازی‌ای که هم اکنون معرفی نمودیم مشاهده می‌نمایید:

نام بازی	چکرز	go	تیک – تاک – تو کور	تخته نرد
تصویر صفحه				

	قطعی	شانسی
اطلاعات کامل	شطرنج، بازی چکرز، go	تخته نرد
اطلاعات ناقص	تیک تا تو کور	پل، پوکر

الگوریتم Min-Max:

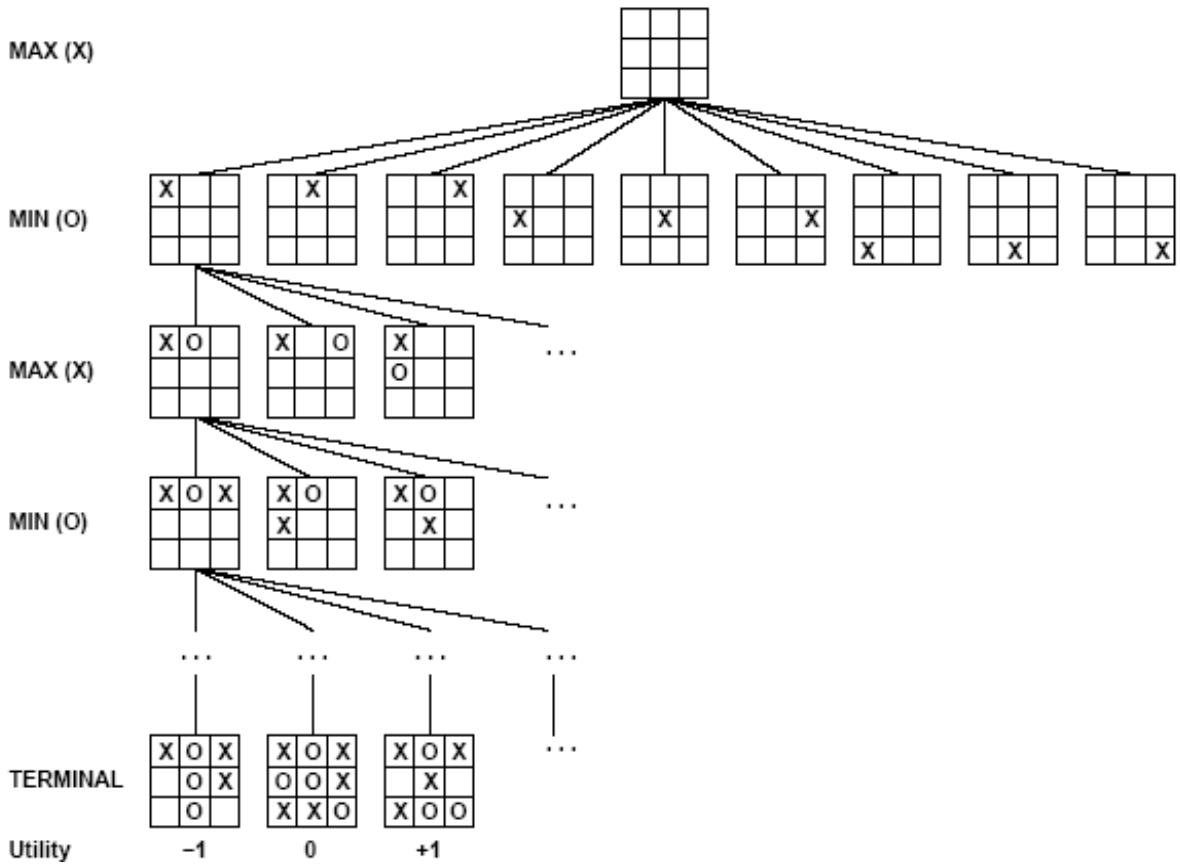
در این قسمت بازی‌های دو نفره را با دو بازیکن min و max در نظر می‌گیریم. ابتدا max حرکت می‌کند و سپس نوبت به حریف یعنی min می‌رسد، به همین گونه ادامه می‌یابد تا بازی تمام شود. در انتهای بازی امتیازات به بازیکن تخصیص داده می‌شود. یک بازی می‌تواند به صورت نوعی مسئله جست وجو به صورت زیر فرموله سازی شود:

- **حالت اولیه:** شامل موقعیت صفحه بازی و مشخص می‌شود که نوبت کدامیک از بازیکنان است.
- **تابع جانشین:** لیستی از (Move, State) را برمی‌گرداند که هر کدام نشانگر یک حرکت قانونی و حالت نتیجه می‌باشد.
- **آزمون پایانی:** این آزمون مشخص می‌کند که آیا بازی تمام شده است یا خیر. حالاتی که بازی در آن‌ها پایانی نامیده می‌شود.
- **تابع سودمندی[1]:** این تابع به حالت پایانی یک عدد نسبت می‌دهد، مثلاً در شطرنج برد عدد ۱+ و برای باخت ۱− و برای مساوی صفر است. در بعضی بازی‌ها نتایج تنوع وسیعی دارند، مثلاً در بازی تخته نرد، نتایج از ۱۹۲+ تا ۱۹۲− تغییر می‌کند. این تابع، تابع هدف یا تابع امتیاز نیز نامیده می‌شود.

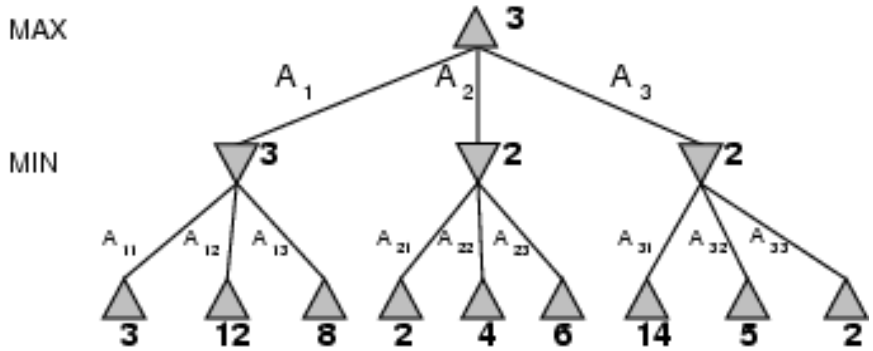
یک نمونه بازی (tic - tac - toe):

حالت اولیه وحرکات قانونی بازیکنان، درخت بازی را تشکیل می‌دهند. در شکل زیر قسمتی از درخت بازی tic - tac - toe را نشان می‌دهد. در حالت اولیه بازیکن max اولین حرکت ممکن را می‌تواند انجام دهد. مهره x متعلق به max و o متعلق به min است. بازی به نوبت بین max و min ادامه می‌یابد. اگر بازیکن سه مهره‌ی خود را در ردیف عمودی، افقی و قطری قرار داده باشد یا اینکه صفحه پر شده باشد، بازی تمام می‌شود. عدد نسبت داده شده به هر برگ در درخت بازی میزان سودمندی آن حالت پایانی را از دیدگاه max برمی‌گرداند. مقادیر بزرگ برای max خوب و برای بازیکن min بد است.

[1] Utility function



مثال:



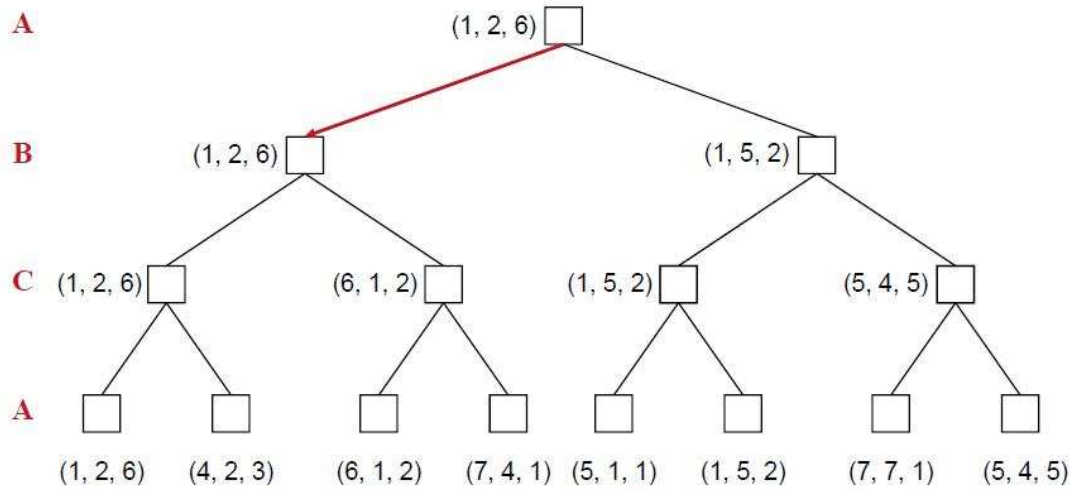
$$MINMAX-VALUE(n)=$$

$$\begin{cases} UTILTY(n) & if\ n\ is\ a\ terminalstate \\ Max_{s\in Successors(n)} MINMAX-VALUE(s) & if\ n\ is\ a\ Max\ node \\ Min_{s\in Successors(n)} MINMAX-VALUE(s) & if\ n\ is\ a\ Min\ node \end{cases}$$

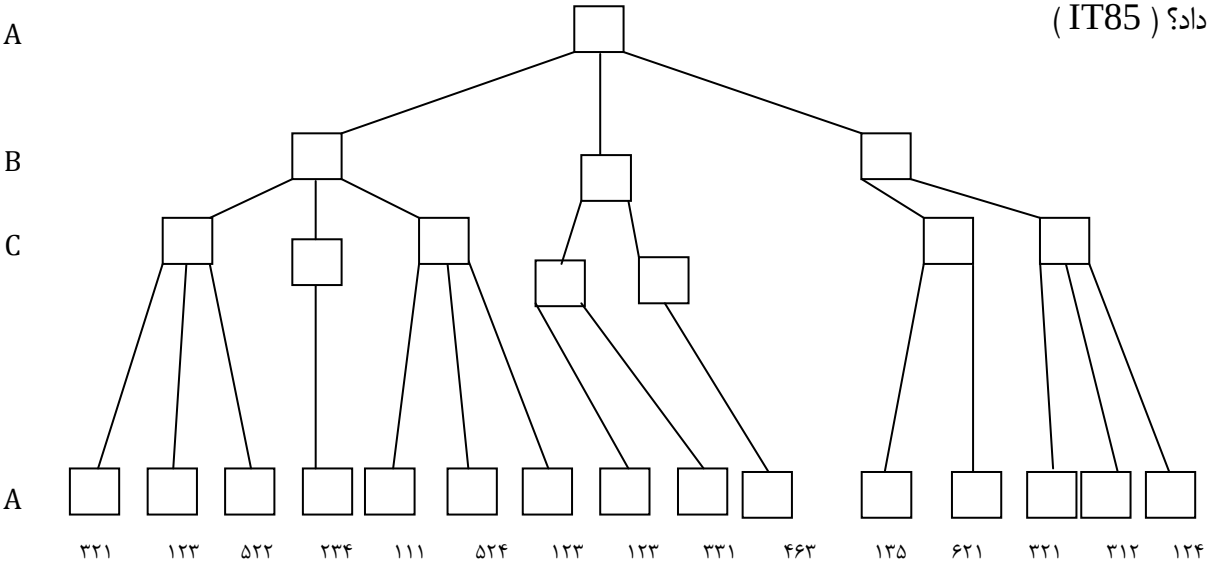
نکته: الگوریتم minimax در واقع، یک جست و جوی اول عمق در درخت بازی است، اگر ماکزیمم عمق درخت m و تعداد b حرکات قانونی در هر حالت وجود داشته باشد پیچیدگی زمانی $O(b^m)$ خواهد بود. اگر الگوریتم تمام فرزندان را با هم ایجاد کند، پیچیدگی مکانی $O(bm)$ خواهد بود و اگر در هر لحظه فقط یک فرزند ایجاد کند، پیچیدگی مکانی $O(m)$ خواهد بود. الگوریتم minimax در صورت محدود بودن درخت، کامل و بهینه است.

بازی‌های چند نفره:

اکنون می‌خواهیم الگوریتم min - max را به بازی‌های چند نفره گسترش دهیم. در ابتدا باید به جای مقدار برای هر برگ، یک بردار از مقادیر برای هر برگ در نظر بگیریم. برای مثال در یک بازی سه نفره با سه بازیکن A و B و C، بردار (v_A v_B v_C) به هر نود نسبت داده می‌شود. در یک حالت پایانی، این بردار، سودمندی آن حالت را برای هر بازیکن نشان می‌دهد، ساده ترین روش پیاده سازی تابع سودمندی به فرمی است که به جای یک مقدار، یک بردار برمی‌گرداند. بازی‌های چند نفره معمولاً با موضوع اتحاد[1] بین بازیکنان روبرو هستند. اتحاد در ابتدای بازی ایجاد می‌شود و با پیشرفت بازی شکسته می‌شود. به عنوان مثال فرض کنید احتمال پیروزی A و B در مقابل C ضعیف است بنابراین بهتر است این دو علیه C متحد شده و به جای حمله به یکدیگر، به C حمله کنند، در غیر این صورت C به تنهایی آنها را شکست می‌دهد. البته به تدریج که C ضعیف می شود، اتحاد بین A و B شکسته می‌شود.



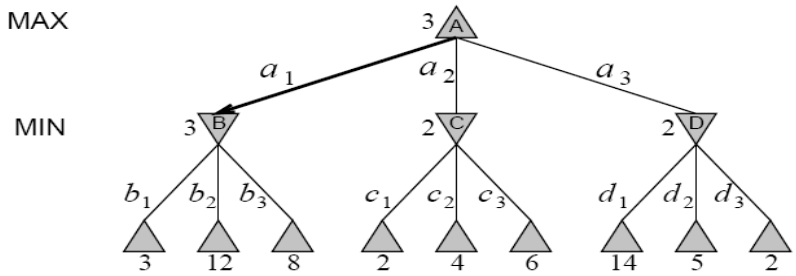
مثال: بازی سه نفره زیر را در نظر بگیرید، بازیکن A با استفاده از الگوریتم minimax کدام عمل را انجام خواهد داد؟ (IT85)



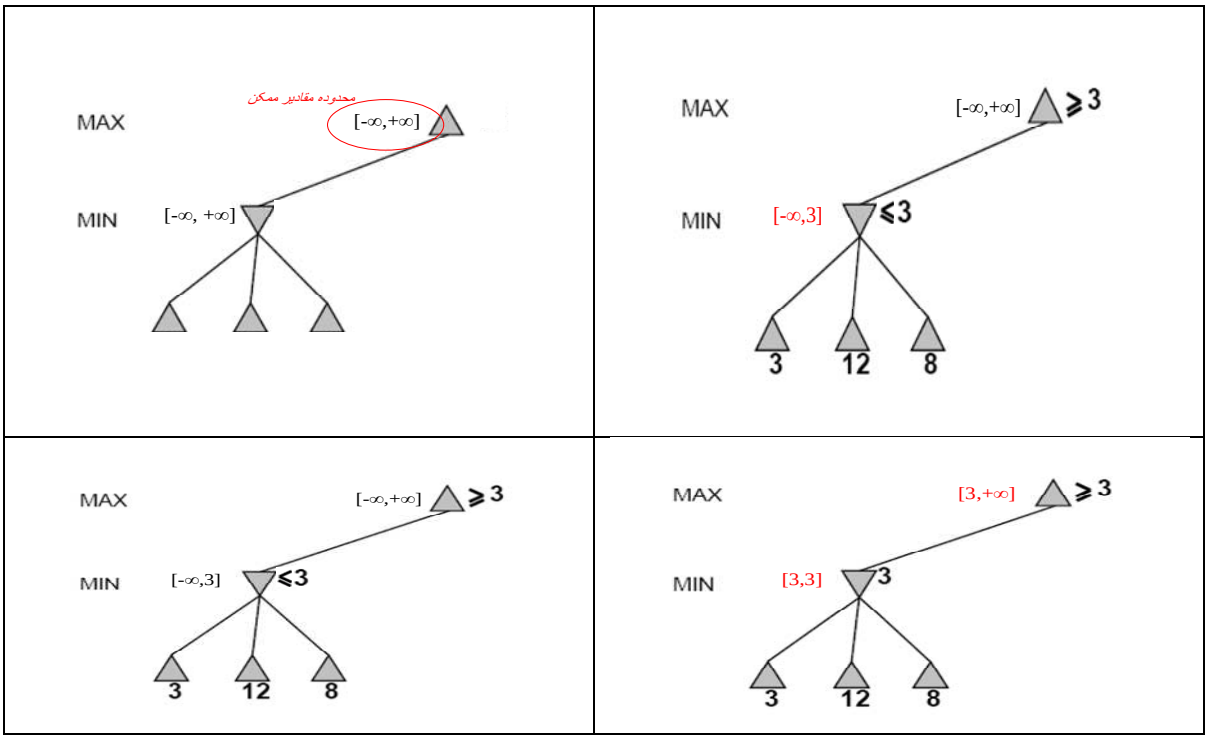
[1] Alliance

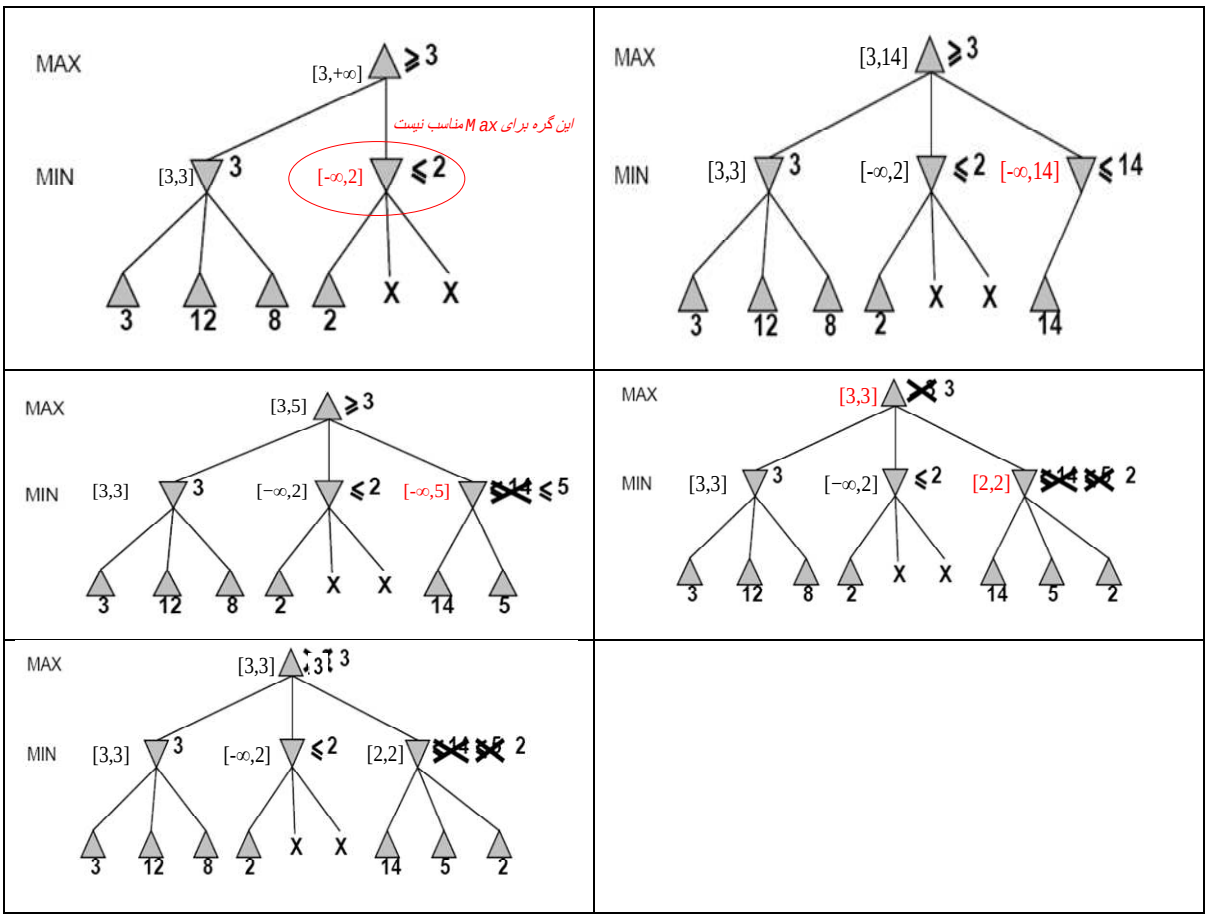
هرس آلفا–بتا (α–β):

مشکل جستجوی minimax این است که تعداد حالات بازی که این جستجو باید بررسی کند، بر حسب تعداد حرکات، نمایی است. متأسفانه نمی‌توان این رشد نمایی را کاهش داد بلکه درخت را می‌توان به طور کارایی به نصف کاهش داد، زیرا تصمیم صحیح، بدون بررسی همه گره‌های درخت جستجو امکان پذیر است. پردازش حذف شاخه‌ای از درخت جستجو بدون ایجاد آن شاخه، هرس نامیده می‌شود. هرس استفاده شده در الگوریتم minimax هرس α_β نامیده می‌شود. الگوریتم minimax استاندارد و هرس α_β حرکت یکسانی را به عنوان نتیجه برمی‌گردانند، زیرا هرس آلفا–بتا شاخه‌هایی که در نتیجه نهایی دخالت ندارند را هرس می‌کند. α وβ مقادیر موقتی هستند. α مقداری است که بازیکن max تاکنون دریافت کرده و کمتر از این نخواهد شد و β مقداری است که بازیکن min دریافت کرده و بیشتر از این نخواهد شد. به خاطر داشته باشید که الگوریتم minimax یک الگوریتم عمقی است و در نتیجه ما مجبوریم در هر زمان فقط نودهای یک مسیر از درخت را در نظر بگیریم. تأثیر هرس α_β به ترتیب بررسی فرزندان (مابعدها) بستگی دارد.



درخت بازی A





مراحل اتخاذ تصمیم بهینه در مورد درخت بازی A

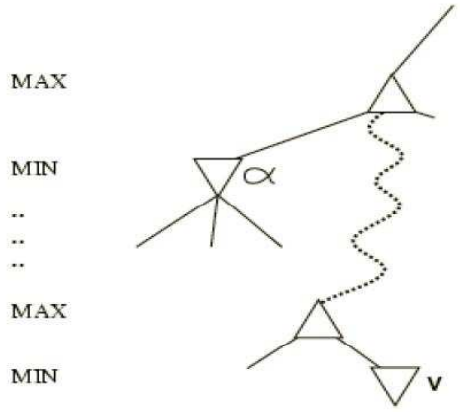
نکته: بنابراین اگر در ابتدا فرزندی که بهتر است، مورد بررسی قرار گیرد مطلوب تر است. اگر فرض کنیم چنین کاری می‌تواند انجام شود، یعنی همیشه بهترین فرزند، اول بررسی شود این برنامه به جای بررسی $O(b^m)$ کافی است $O(b^{m/2})$ را بررسی کند. اگر به جای مرتب سازی فرزندان آنها را به طور تصادفی انتخاب کنیم تعداد نودهای مورد بررسی $O(b^{3m/4})$ خواهد بود. در بازی‌ها نیز نودهای تکراری ممکن است بوجود بیایند، بنابراین بهتر است مقدار ارزیابی نودها را در یک جدول درهم ساز ذخیره کنیم تا در صورت دوباره مواجه شدن با این نود مقدار ارزیابی آن محاسبه نشود. جدول درهم سازی موقعیت‌های مشاهده شده، جدول جابجایی نیز نامیده می‌شود. این جدول معادل لیست closed در الگوریتم Graph_search است.

نکته: بنابراین به طور خلاصه در هرس α_β داریم:

- ممکن است یک گره یا زیردرخت به طور کامل هرس شود.
- ممکن است درخت به طور کامل بررسی شود و هرسی صورت نگیرد.
- الگوریتم α_β در یک بازه زمانی ثابت، تقریباً دو برابر minimax کارایی دارد.
- هنگامی ارزش α یا β به پدرش اختصاص می‌یابد، که فرزند دیگری نداشته باشد. در سایر حالات این مقدار موقتی است.

❖ موثر بودن هرس بستگی به ترتیب مقادیر فرزندان یک گره بستگی دارد. اگر در Min بودیم ترتیب صعودی بهتر است و اگر درMax بودیم ترتیب نزولی بهتر است.

وجه تسمیه:



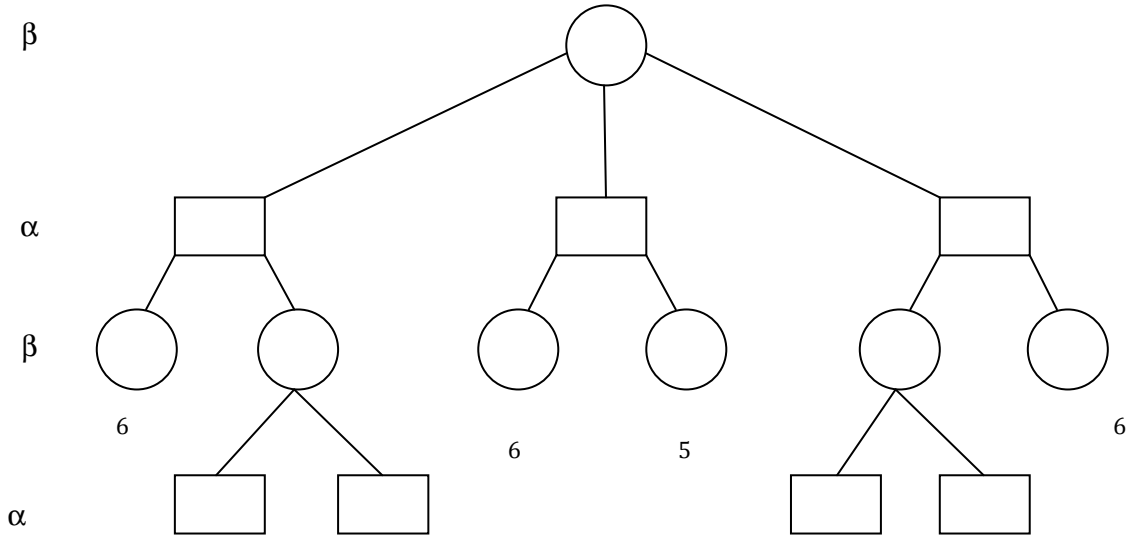
❖ آلفا مقدار بهترین انتخاب (یعنی بالاترین مقدار) یافته شده تاکنون در هر نقطه انتخاب در طول مسیر برایMAX می‌باشد.

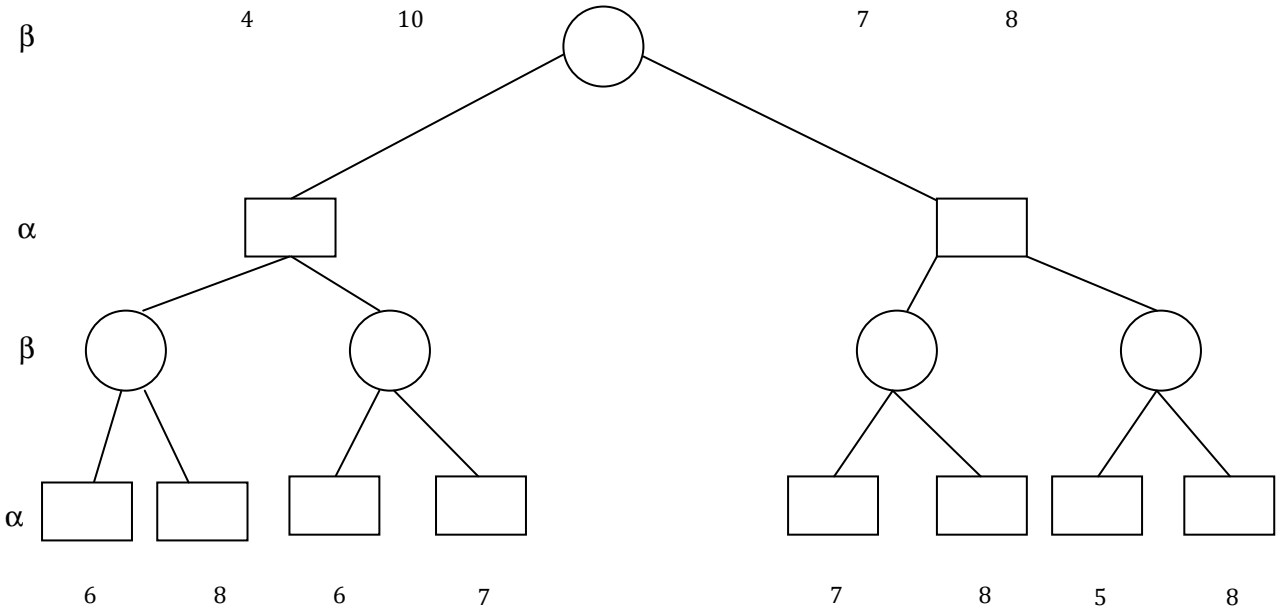
❖ اگر V بدتر از آلفا باشد، MAX از آن اجتناب می‌کند.

← آن شاخه را هرس می‌کند

❖ بتا نیز برای MIN مانند آلفا تعریف می‌شود.

مثال: اگر در درخت‌های زیر با هرسα_β پیمایش شوند کدام گره‌ها ملاقات نخواهند شد؟





تصمیمات بلادرنگ ناقص :

الگوریتم minmax کل درخت بازی را جستجومی‌کند، درحالیکه هرسα_β قسمت وسیعی از درخت را هرس می-کند. با این وجود الگوریتم هرسα_β باز هم باید تا رسیدن به حالت پایانی ادامه داده و در نتیجه قسمت وسیعی از درخت کامل را بررسی کند. شانون پیشنهاد کرد عمق درخت بازی محدود شود و به جای تابع سودمندی ازتابع ارزیابی[۱] اکتشافی برای نودها استفاده شود. بعبارت دیگر در الگوریتم minmax وهرسα_β باید تغییرات زیرصورت بگیرد:

- ❖ تابع سودمندی با تابع ارزیابی اکتشافی جایگزین شود. این تابع ارزیابی اکتشافی تخمینی ازسودمندی یک واقعیت را بر می‌گرداند.
- ❖ آزمون پایانی را با آزمون برش[۲] جایگزین کنیم. این آزمون مشخص می‌کندکه چه موقع تابع ارزیابی فراخوانی شود.

توابع ارزیابی:

توابع ارزیابی، تخمینی از سودمندی مورد انتظار بازی از موقعیت داده شده را بر می‌گردانند. کارایی یک برنامه بازی به کیفیت این تابع ارزیابی بستگی دارد. تابع ارزیابی باید به سه دلیل مناسب باشد:

- ❖ مقادیری که تابع ارزیابی به حالات پایانی نسبت می‌دهدبا مقادیری که تابع سودمندی به آنها نسبت می‌دهد متناسب باشد
- ❖ محاسبات تابع ارزیابی نباید زمان زیادی مصرف کند.
- ❖ در نودهای غیر پایانی این تابع باید به درستی شانس‌های واقعی برد را منعکس کند.

بسیاری از توابع ارزیابی برای هر خصوصیت در یک حالت ترکیبات عددی متفاوتی را در نظر می‌گیرند وسپس آن‌ها را با هم جمع نموده و بعنوان یک مقدار برمی‌گردانند. برای مثال در بازی شطرنج سرباز ارزش ۱,اسب یا فیل ۲ و قلعه ۵ است. این مقادیر جمع شده ومقدار ارزیابی یک موقعیت را مشخص می‌کند. از نظر ریاضیاتی این نوع تابع، تابع خطی وزن دار نامیده می‌شود.

$$Eval(s) = w_1f_1(s) + w_2f_2(s) + ... + w_nf_n(s) = \sum f_i(s)$$

که در این تابع، f_iها خصوصیات موجود در یک موقعیت و w_i وزن‌ها را نشان می‌دهد. مثلاً "در مورد شطرنج f_i می-تواند تعدادمهره‌های روی صفحه وw_i ها ارزش هرکدام باشد.

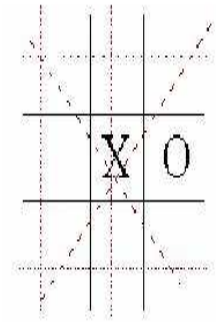
---

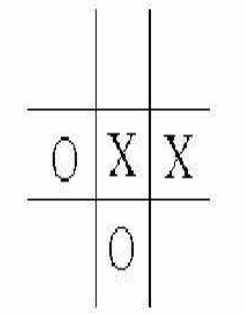
[۱] Evaluation function

[۲] Cut-off test

مثال: تابع ارزیابی برای بازی دوز

$$e(n) =$$
$$if\ n\ is\ win\ for\ Max, +\infty, if\ n\ is\ win\ for\ Min, -\infty$$
$$Else$$
$$(\ Possible\ number\ of\ rows, columns\ and\ diagonals\ available\ to\ Max\ )$$
$$-$$
$$(\ Possible\ number\ of\ rows, columns\ and\ diagonals\ available\ to\ Min\ )$$





$$e(n) = 6 - 4 = 2 \qquad e(n) = 4 - 3 = 1$$

قطع جست وجو:

راحت ترین راه برای کنترل میزان جستجو، قرار دادن محدودیت روی عمق است. میزان عمق موردنظر با استفاده از محدودیت‌های زمانی بازی مشخص می‌شود. بنابراین تست قطع برای تمام گره‌ها در زیر عمق d موفق خواهد بود. عمق طوری انتخاب می‌شود که میزان زمان استفاده شده ازآنچه قوانین بازی اجازه می‌دهد، تجاوز نکند. زمانی که وقت بازی تمام می‌شود برنامه حرکت انتخابی توسط عمیق ترین جستجوی کامل شده را بر می‌گرداند. به دلیل تخمینی بودن توابع ارزیابی، این رهیافت، نتایج ناخوشایندی را به دنبال خواهد داشت. تابع ارزیابی باید فقط برای حالاتی بکار برده شوند که ساکن هستند.

حالت ساکن[1]:

فرض کنید که حالت x در عمق برش d قرار دارد و Eval(x) = n باشد. اگر عمق برش درخت را افزایش دهیم آنگاه x دیگر در عمق برش نیست و دارای مقدار minimax است. در این صورت اگر مقدار minimax و m تفاوت چشمگیری نداشته باشند، حالت x ساکن است. موقعیت‌های غیر ساکن باید بسط داده شوند تا به موقعیت ساکن برسیم. این جستجوی اضافی جستجوی ساکن نامیده می‌شوند.

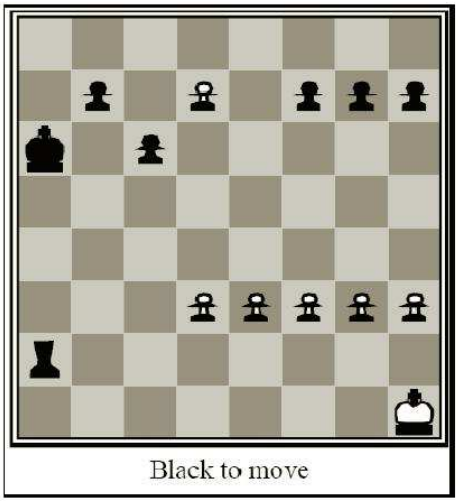
علاوه بر روش هرسα_β روش‌های زیر نیز برای بهبود الگوریتم minmax بکار برده می‌شود:

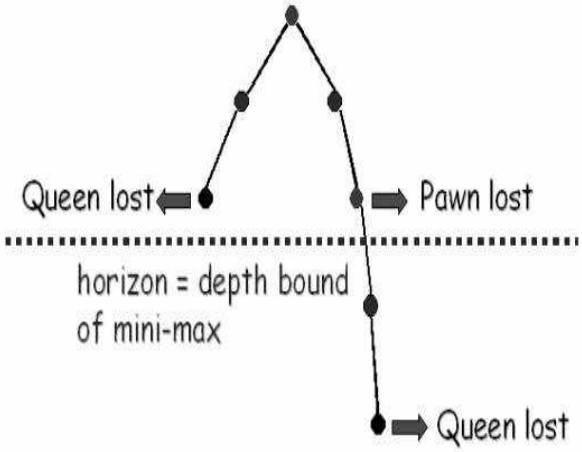
۱) انتظار برای سکون ۲) جستجوی ثانویه ۳) هرس رو به جلو ۴) عمیق شونده تکراری

[1] Quiescent

اثرافق[1]:

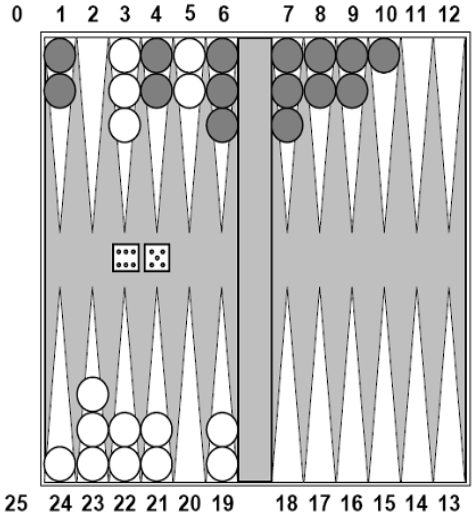
حالتی است که در آن انجام حرکتی را که برای حریف بسیار سودمند است مدتی به تعویق می‌اندازد. در حالیکه حریف بالاخره آن حرکت را انجام خواهد داد. به عنوان مثال در یک بازی شطرنج یک نفر با کیش دادن‌های متوالی، خوردن مهره خود را به تعویق می‌اندازد.

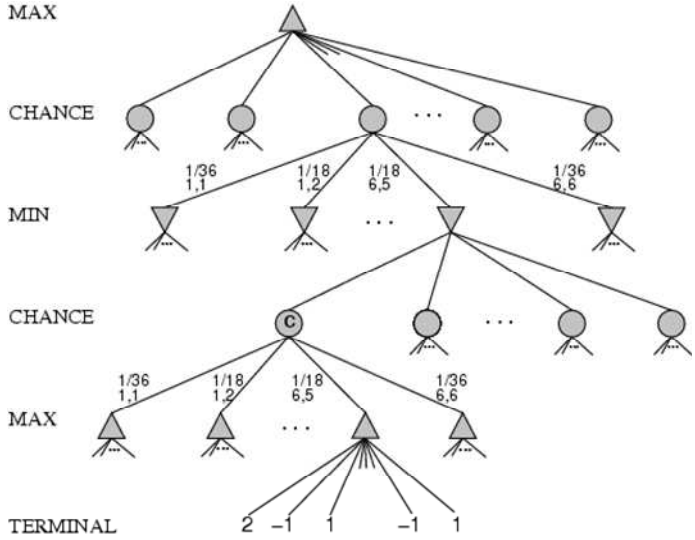




بازی‌هایی که دارای عامل شانس هستند:

درزندگی واقعی بر خلاف شطرنج حوادث غیر قابل پیش بینی زیادی وجود دارند که ما را در شرایط غافلگیرانه‌ای قرار می‌دهند. بازی‌های زیادی غیر قابل پیش بینی بودن را توسط یک عنصر تصادفی مانند پرتاپ تاس یا سکه نشان می-دهند. تخته نرد یک بازی است که شانس ومهارت را ترکیب می‌کند. تاس‌ها در ابتدای بازی، توسط بازیکنی که نوبتش است ریخته می‌شود تا مجموعه حرکاتی که قابل انجام هستند مشخص شوند. درخت بازی تخته نرد علاوه بر گره-های min و max باید شامل گره‌های شانس نیز باشد.

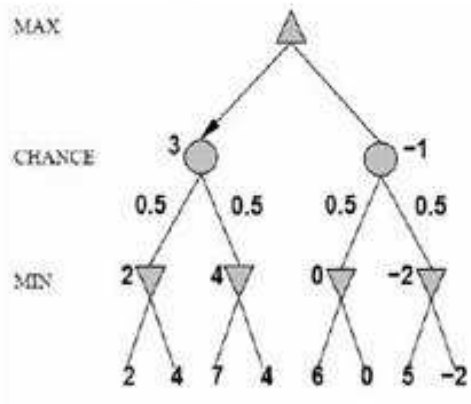




[1] Horizon Effect

دوایر نشان دهنده‌ی گره‌های شانس هستند. شاخه‌هایی که از هر گره شانس خارج شده‌اند، مقادیر مختلف تاس‌ها را مشخص می‌کنند. هر کدام با شانسی که دارند برچسب می‌خورند. برای دو تاس ۳۶ حالت وجود دارد که در آن ۲۱ حالت متفاوت قابل استخراج است، مثلاً (۱,۳) و (۳,۱) را یک حالت در نظر می‌گیریم واحتمال آن ۲/۳۶ است.

مرحله بعدی این است که چگونه بفهمیم تصمیم گیری صحیحی داشته باشیم؟ اگرچه در اینجا مقدار minmax قطعی وجود ندارد، در عوض می‌توانیم میانگین یا مقدار مورد انتظار[1] رامحاسبه کنیم. این مقدار مورد انتظار، کلیه ترکیبات تاس‌ها را در نظر می‌گیرد. بنابراین مقدار minmax در بازی‌های قطعی رابه مقدار مورد انتظار در بازی‌های شامل عنصر شانس عمومیت می‌دهیم. نودهای max و min و پایانی مانند قبل محاسبه می‌شود ولی مقدار عنصر شانس به صورت زیر محاسبه می‌شود:



$$v = \sum_{chance\ nodes} P(n) * Minimax(n) = 0.5 * 4 + 0.5 * 2 = 3$$

پیچیدگی زمانی:

n: تعداد رویدادهای متفاوت $O(b^m n^m)$

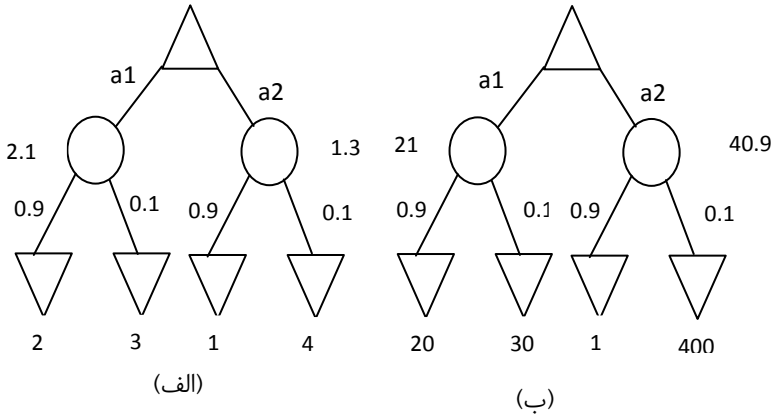
پیچیدگی زمانی الگوریتم Expect_ minmax(x) $O(b^m n^m)$ می‌باشدکه در آن، n تعداد پرتاب‌های موجود، m حداکثر عمق درخت و b فاکتور انشعاب می‌باشد.

ارزیابی موقعیت در بازیها با عنصر شانس:

در الگوریتم minmax هر انتقال با حفظ مرتبه ارزش برگ‌ها، تاثیری در انتخاب حرکت ندارد. یعنی می‌توان مقادیر ۴,۳,۲,۱ یا مقادیر ۴۰۰,۳۰,۲۰,۱ را استفاده نمود و تصمیم مشابهی گرفت. اما با وجود گره‌های شانس، این خاصیت حفظ نمی‌شود. در شکل الف با مقادیر برگی ۴,۳,۲,۱ حرکت a1 بهترین است در حالیکه در شکل ب با مقادیر برگی ۲۰,۱ ۴۰۰,۳۰ حرکت a2 بهترین است بنابراین در الگوریتم minmax استاندارد، انتقال با حفظ ارزش برگ‌ها تاثیری در انتخاب حرکت ندارد ولی اگرعنصر شانس چاشنی کار شود انتقال با حفظ مرتبه کارساز نیست.

---

[1] Expected

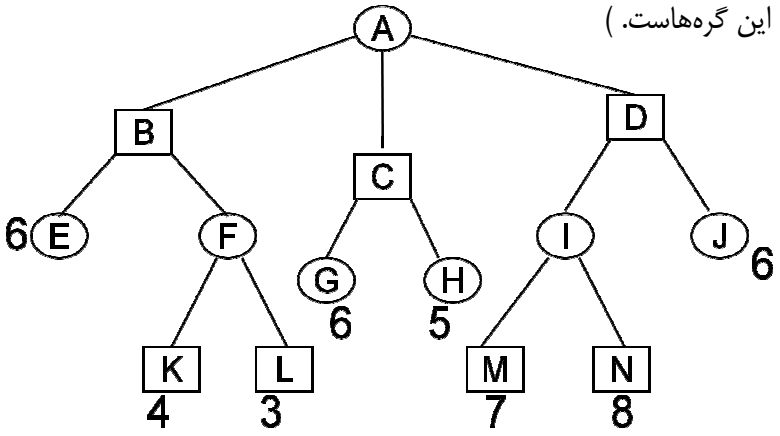


(الف) (ب)

سوالات تستی آخر فصل

ترم اول ۸۷–۸۸

۱. اگر با استفاده از روش جستجوی Minimax درخت جستجوی زیر پیمایش شود، با استفاده از روش هرس آلفا بتا و اعداد کنار گره‌های Max، مربع‌ها معرف Min کدامیک از گره‌های این درخت ملاقات نخواهد شد؟ (دوایر معرف گره برگ و اعداد معرف ارزش این گره‌هاست.)



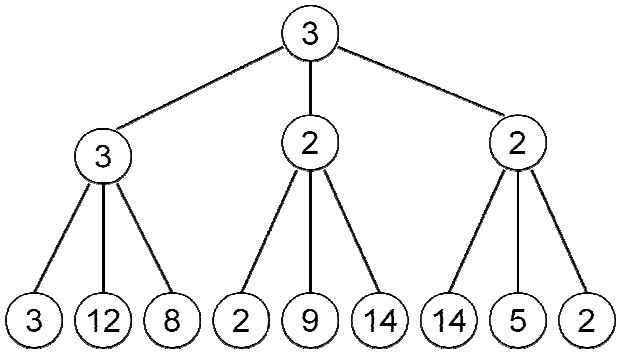
الف. {L, H, N}

ب. {L, N, J}

ج. {L, H, J}

د. {K, N, H}

۲. دردرخت Minimax زیر را در نظر بگیرید. اگر هرس آلفا بتا استفاده کنیم چند اتصال هرس می‌شود؟



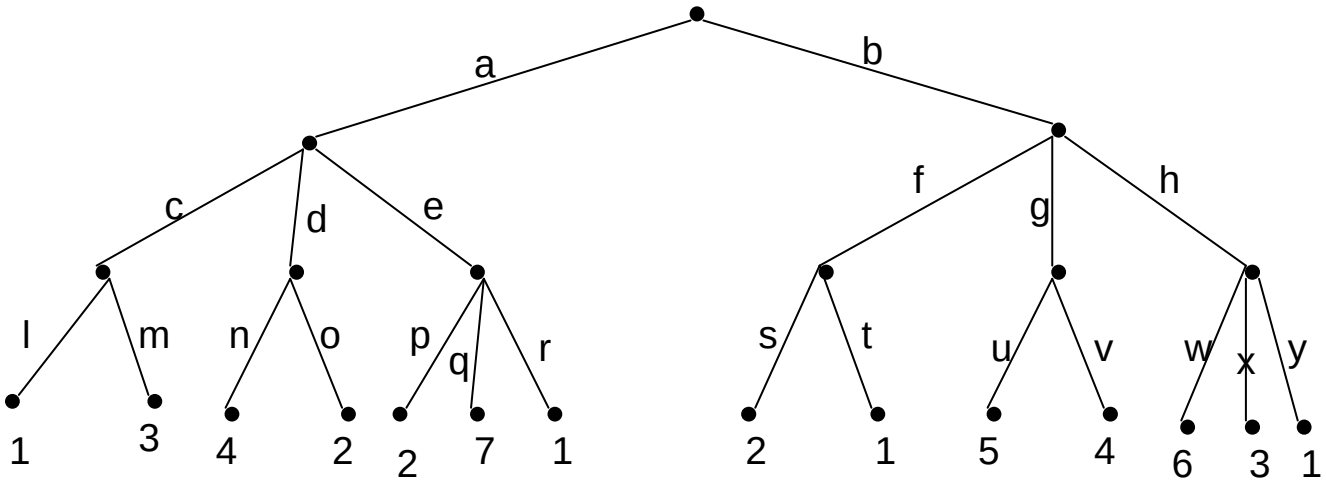
الف. ۲ اتصال

ب. ۴ اتصال

ج. ۶ اتصال

د. هیچ اتصالی هرس نمی‌شود.

۳. در درخت بازی زیر اگر از هرس آلفا بتا استفاده شود کدام شاخه‌ها حذف خواهند شد؟ (فرض می‌شود حذف شاخه غیرانتهایی به طور ضمنی حذف تمام زیر درخت تحت آن را به همراه دارد و ذکر شاخه‌های زیر درخت لازم نیست.)



الف. c-g-h

ب. o-r-v-h

ج. o-p-r-v-n-y

د. o-r-g-h

ترم دوم ۸۷–۸۸

۴. در فضای بازی‌ها، مقدار تابع سودمندی (utility function) نشانگر چیست؟

الف. ارزش بازی در هنگام خاتمه ب. مقدار خروجی بازی در گره‌های پایانی

ج. میزان سودمندی د. میزان سودمندی از دیدگاه Max

۵. در مورد بازی‌های دارای عامل شانس کدام گزینه صحیح نیست؟

الف. هرس کردن شاخه‌ها مشکل تر می‌شود. (باید برای تابع سودمندی کران‌هایی قائل شویم)

ب. دقت تابع ارزیاب باید بیشتر شود. (اینکه به موقعیت‌های بهتر امتیاز بالاتر داده شود کافی نیست.)

ج. پیچیدگی به $O(b^m n^m)$ افزایش می‌یابد. (N تعداد حالات مختلف پرتاب تاس می‌باشد)

د. ارزش گره شانس برابر ارزش بهترین پسین آن گره می‌باشد.

ترم تابستان ۸۸

۶. در مورد تصمیمات بلا درنگ ناقص در بازی‌ها کدام گزینه صحیح نیست ؟

الف. هنگامیکه زمان کافی برای جستجوی با تصمیمات بهینه نداریم مورد استفاده قرار می‌گیرد.

ب. تابع ارزیاب مقدار دقیق ارزش گره‌ها را محاسبه می‌کند.

ج. آزمون قطع جایگزین آزمون پایانی در جستجوی با تصمیمات بهینه می‌گردد.

د. تابع ارزیاب جایگزین تابع سودمندی در جستجوی با تصمیمات بهینه می‌گردد.

۷. کدام گزینه در مورد جستجوی بیشینه کمینه با هرس آلفا بتا صحیح نیست ؟

الف. از نوع اول عمق است.

ب. حالات تکراری در درخت می‌تواند هزینه جستجو را به طور نمایی افزایش دهد.

ج. با بررسی بهترین پسین‌ها فاکتور انشعاب مؤثر $\sqrt{b}$ خواهد بود.

د. با بررسی تصادفی پسین‌ها پیچیدگی در حدود $O(b^d)$ می باشد.

۸. در مورد تابع ارزیاب در بازی‌های دارای گره‌های شانس کدام مورد دقیق تر است ؟

الف. اینکه به موقعیت‌های بهتر امتیاز بالاتری بدهد کافی است (دقت زیاد تخمین مؤثر نیست.)

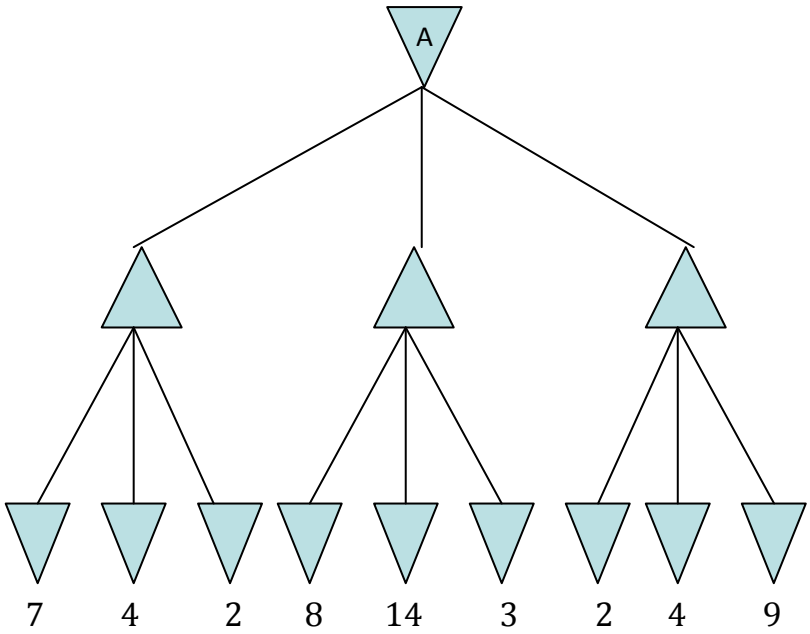
ب. اگر برای مقادیر سودمندی کران‌هایی قائل شویم، هرس آلفا بتا به شیوه مشابهی قابل اعمال است.

ج. ارزش یک گره شانس برابر ارزش بهترین پسین آن گره خواهد بود.

د. مانند روش بیشینه کمینه می‌توان از هرس آلفا بتا استفاده کرد.

ترم اول ۸۸–۸۹

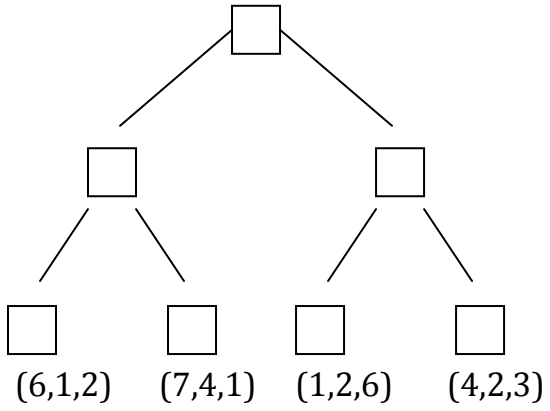
۹. با فرض اینکه Δ به معنی Max و ∇ به معنی Min باشد اگر هرس آلفا بتا استفاده شود کدام گره‌ها حذف نمی-شوند؟



الف. ۹و۴ ب. ۳و۱۴ ج. ۲و۴ د. ۹و۴و۳و۱۴

۱۰. دردرخت بازی چندنفره زیر، مقدار بردار مربوط به گره B کدام است؟

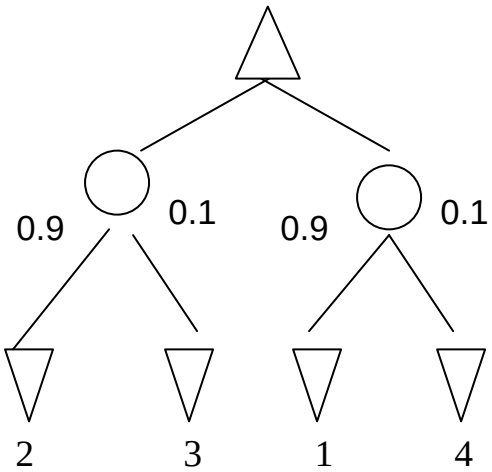
مقادیر بردارها به این ترتیب چیده شده‌اند: (A, B, C)



الف. (۶، ۲، ۳)　　ب. (۴، ۲، ۶)　　ج. (۱، ۲، ۲)　　د. (۱، ۲، ۶)

۱۱. اگر در بازی‌های دارای گره شانس درخت به صورت مقابل باشد مقدار گره Δ کدام است؟

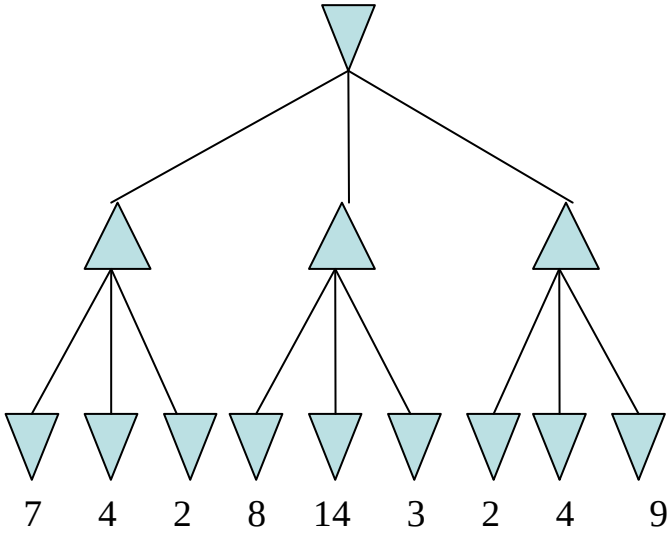
(Min = ∇ , change = O , Max = Δ)



الف. ۲.۱　　ب. ۱.۳　　ج. ۴　　د. ۳

ترم دوم ۸۸–۸۹

۱۲. در درخت زیر اعمال آلفا- بتا، منجر به حذف چه شاخه‌هایی می‌شود؟



الف. ۹و۴ ب. ۳و۱۴

ج. ۲و۴ د. ۹و۴و۳و۱۴

ترم تابستان ۸۹

۱۳. کدامیک از روش‌های زیر در بازی شطرنج می‌توانند پیچیدگی زمانی هرس آلفا و بتا را با تصمیمات بهینه بطور قابل توجهی کاهش دهند؟

۱) مرتب سازی پسین‌ها از بهترین به بدتری

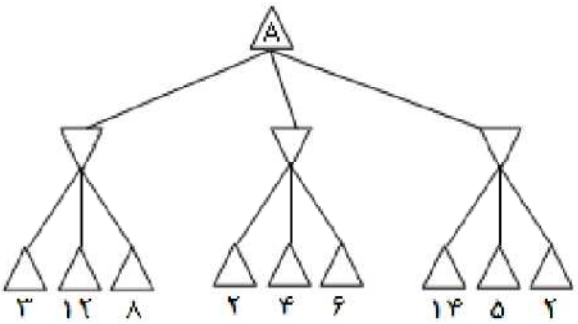
۲) ذخیره مقدار سودمندی برای هر گره در یک جدول هش واستفاده از آنها در حالات مشابه (تکراری) بدون محاسبه مجدد.

۳) جایگزینی Terminal-test با Cutoff-test و تابع سودمندی با تابع ارزیاب

الف. ۲،۱و۳ ب. ۱ ج. ۱و۲ د. ۱و۳

ترم اول ۸۹–۹۰

۱۴. اگر Δ به معنی Max و ∇ به معنی Min باشد روش Minimax چه مقداری را برای در نظر خواهد گرفت؟



الف. ۱۲ ب. ۱۴ ج. ۳ د. ۲

۱۵. اگر در سوال قبل از روش هرس آلفا و بتا استفاده شود، گره‌ها با چه مقادیری بررسی نمی‌شود؟

الف. ۸ و۱۲ ب. ۴ و۶ ج. ۲ و۵ د. ۲ و۵ و۱۴

۱۶. در بازی‌های چند نفره در صورتی که بازیکنان A و B نسبت به بازیکن C وضعیت ضعیف تری دارند معمولاً چه رفتاری صورت می‌گیرد؟

الف. معمولا A و B به جای حمله به یکدیگر به C حمله می‌کنند.

ب. هر کس بدنبال برد خود به بقیه حمله می‌کند.

ج. ممکن است یکی از آنها با C در جهت پیشرفت خود همکاری کند.

د. A و B تا انتهای بازی متحد می‌شوند.

ترم دوم ۸۹–۹۰

۱۷. کدام گزینه در مورد تصمیمات بلادرنگ ناقص در بازی‌ها صحیح نیست؟

الف. اثر افق دید به دلیل استفاده از تابع ارزیاب در موقعیت‌های غیر ساکن رخ می‌دهد.

ب. در موقعیت‌های غیر ساکن می‌توان به بسط ادامه داد تا به موقعیت‌های ساکن رسید. (جستجوی ساکن)

ج. به جای سودمندی utility از تابع ارزیاب در نقاط قطع (cut - off) در صورت امکان استفاده می‌شود.

د. تنها در موقعیت‌های ساکن استفاده از تابع ارزیاب صحیح است.

۱۸. با فرض این که تابع ارزیاب در نقاط قطع با توجه به شانس‌های برد محاسبه می‌شود و بدانیم که در موقعیت‌های شانس برد ۴۵٪ (با سودمندی ۱+) شانس باخت ۲۰٪ (با مقدار سودمندی ۱–) و شانس تساوی ۳۵٪ (با مقدار سودمندی ۰) است. مقدار برگشتی از یک تابع ارزیاب معقول چقدر است؟

الف. ۰/۵ ب. ۰/۳۵ ج. ۰/۷۵ د. ۰/۲۵

۱۹. در مورد هرس پیشرو در بازی‌ها کدام درست است؟ (Forward pruning)

الف. حذف شاخه‌های که محاسبه آن‌ها تاثیری در نتیجه ندارد.

ب. حذف شاخه‌هایی که قبلاً محاسبه شده‌اند توسط ذخیره سازی مقدار آن‌ها (در حرکات بعدی)

ج. حذف یکسری حرکات از یک گره مفروض که حتی ممکن است منجر به حذف حرکت بهینه شود.

د. حذف حرکاتی که امکان انجام آن‌ها وجود ندارد.

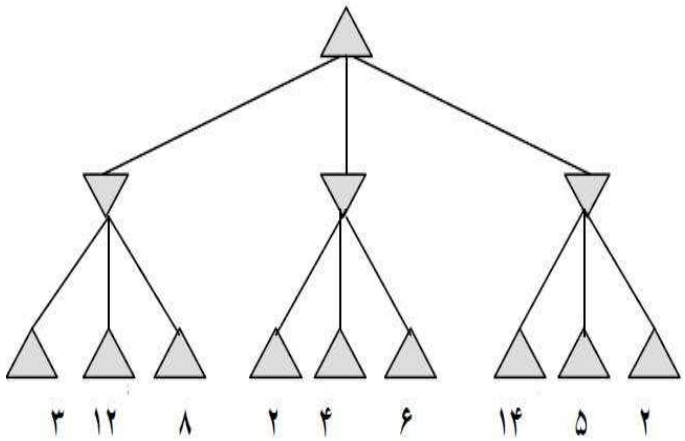
۲۰. در درخت بازی زیر اگر از هرس آلفا – بتا استفاده شود، کدام شاخه‌ها حذف خواهند شد؟

الف. ۴ و ۶

ب. ۱۴ و ۵ و ۲

ج. ۵ و ۲

د. ۴و ۶ و ۲



پاسخ نامه تستی

سوال	گزینه صحیح
۱	ج
۲	الف
۳	د
۴	ب
۵	د
۶	ب
۷	د
۸	ب
۹	ب
۱۰	د
۱۱	الف
۱۲	ب
۱۳	الف
۱۴	ج
۱۵	ب
۱۶	الف
۱۷	
۱۸	
۱۹	
۲۰	الف

سوالات تشریحی آخر فصل

ترم اول ۸۷–۸۸

الگوریتمی برای روش Minimax ارائه دهید.

ترم تابستان ۸۹

نحوه هرس شدن درخت در هرس آلفا بتا را با مثالی تشریح نمایید. چگونه می‌توان از هرس آلفا و بتا در بازی‌های با گره شانس بهره برد؟

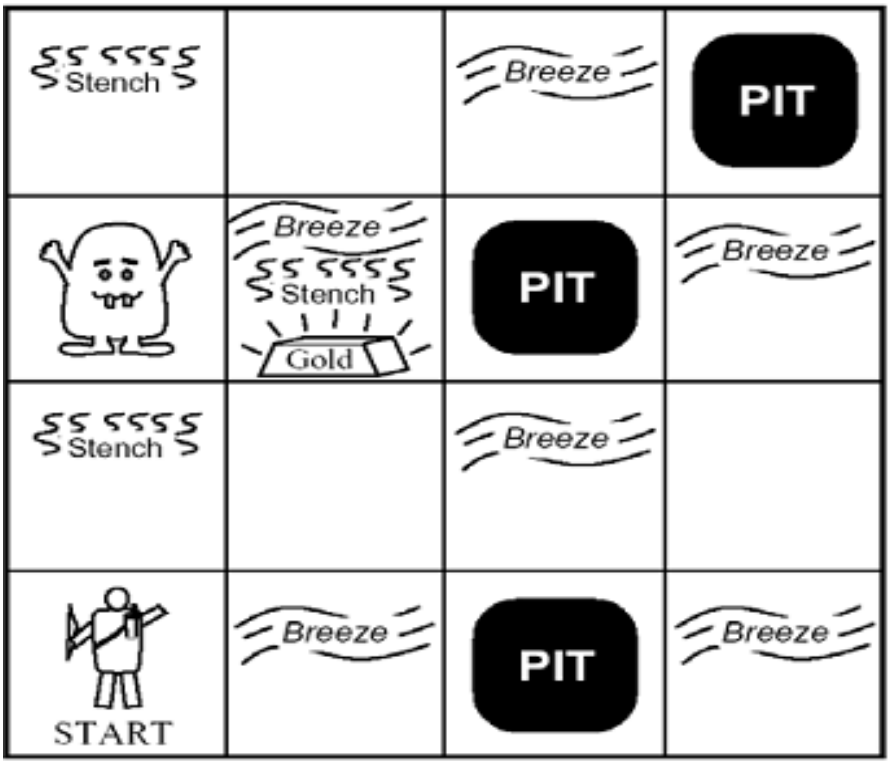
تابستان ۹۰

نحوه عملکرد الگوریتم MiniMax را شرح دهید. (۱ نمره)

فصل هفتم: عامل‌های منطقی

آنچه در این فصل خواهید آموخت:

- ❖ **عامل‌های مبتنی بر دانش**
- ❖ **محیط Wumpus**
- ❖ **منطق-مدل‌ها و استلزام‌ها**
- ❖ **منطق گزاره‌ای**
- ❖ **هم ارزی، اعتبار و صدق پذیری**
- ❖ **الگوهای استدلال در منطق گزاره‌ای**
- ❖ **الگوریتم Resolution**
- ❖ **الگوریتم زنجیره رو به جلو و زنجیره رو به عقب**



عامل مبتنی بر دانش:

جزء اصلی عامل مبتنی بر دانش پایگاه دانش[۱](KB) است. یک پایگاه دانش مجموعه‌ای از جملات در یک زبان رسمی است. هرجمله در یک زبان به نام زبان بازنمایی دانش[۲] (مثلاً منطق) بیان می‌شود و ادعایی را در مورد دنیا باز نمایی می‌کند. باید راهی برای اضافه کردن جملات به پایگاه دانش (TELL) و راهی برای پاسخ به سؤال پرسیده شده از پایگاه دانش (ASK) وجود داشته باشد که هر دوی این کار ممکن است شامل عمل استنتاج نیز باشد. استنتاج یعنی جملات جدیدی از جملات قدیمی بدست می‌آیند. نتایج حاصل، باید از KB پیروی کنند. پیروی[۳] یعنی انجام فرایند استنتاج تحت مقررات خاص. هر وقت برنامه عامل فراخوانی می‌شود ۳ کار انجام می‌شود:

- به پایگاه دانش می‌گوید چه چیزی را از محیط توسط سنسورهایش دریافت کرده است.

Tell(KB, Make - Percept - Sentence(percept, t))

- از پایگاه دانش سؤال می‌کند چه عملی را باید انجام دهد.

action ← Ask(KB, Make - Action - Query(t))

- عامل عمل انتخاب شده را با Tell ذخیره کرده وآن را اجرا می‌کند.

Tell(KB, Make - Action - Sentence(action, t))

```
Function KB - AGENT( percept ) returns an action
    static:   KB, a knowledge base
              t, a counter, initially 0, indicating time
    TELL( KB, Make - Percept - Sentence( percept, t ) )
    action ← Ask( KB, Make - Action - Query(t) )
    t ← t + 1
    return action
```

یک عامل عمومی مبتنی بر دانش

عامل مبتنی بر دانش خیلی شبیه به عاملهایی با حالت درونی می‌باشد. عاملها دردو سطح متفاوت تعریف می‌شوند:

- **سطح دانش:** در این سطح مشخص می‌شود عامل چه چیزی می‌داند و چه اهدافی دارد بدون توجه به جزئیات پیاده سازی
- **سطح پیاده سازی:** در این سطح، ساختمان داده ی اطلاعات پایگاه دانش و الگوریتم‌هایی که بر روی آن کار می‌کنند نشان داده می‌شود.

---

[۱] Knowledge Base

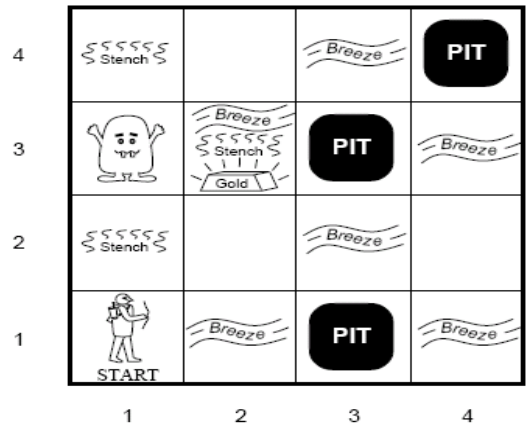
[۲] Knowledge Representation

[۳] follow

دنیای وامپوز:

قبل از اینکه به مفاهیم و الگوریتم‌های این فصل بپردازیم بهتر است مثالی را معرفی کرده و سپس مباحث خود را روی آن بیان کنیم. همان طور که در فصل ۲ اشاره کردیم برای هر عامل باید مشخصات محیط کار (PEAS) مشخص شود. مشخصات محیط کار دنیای وامپوز به صورت زیر است:

- **معیار کارایی:** ۱۰۰۰+: انتخاب طلا، ۱۰۰۰-: افتادن در گودال یا خورده شدن، ۱-: هر مرحله، ۱۰-: استفاده از تیر.
- **محیط:** بوی تعفن در مربع‌های همجوار wumpus، نسیم در مربع‌های همجوار گودال، درخشش در مربع حاوی طلا، کشته شدن wumpus با شلیک در صورت مقابله، تیر فقط مستقیم عمل می‌کند، برداشتن و انداختن طلا.
- **حسگرها:** بوی تعفن، نسیم، تابش، ضربه، جیغ زدن
- **محرکها:** گردش به چپ، گردش به راست، جلو رفتن، برداشتن، انداختن، شلیک کردن



ویژگیهای محیطی دنیای وامپوز:

i. **قابل مشاهده کامل:** خیر، فقط ادراک محلی امکان پذیر است.

ii. **قطعی:** بله، وضعیت بعدی دقیقا مشخص است.

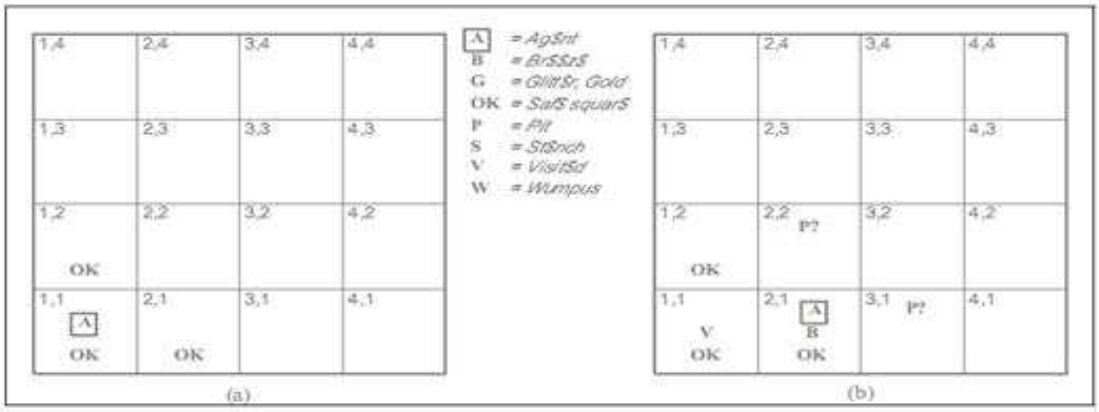
iii. **رویدادی:** خیر، ترتیبی از فعالیت‌های مرتبط با هم است.

iv. **ایستا:** بله، wumpus و گودال‌ها حرکت ندارند

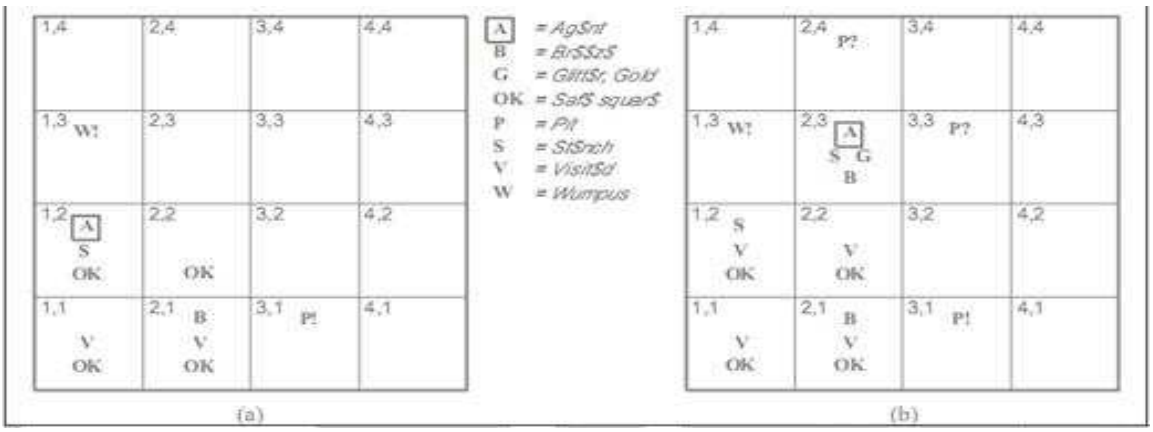
v. **تک عامله:** بله wumpus در اصل یک خصوصیت طبیعی محیط است و یک عامل مستقل محسوب نمی-شود.

vi. **گسسته:** بله

فرایند اکتشاف در دنیای وامپوز:



اولین گام برداشته شده توسط عامل در دنیای wampus (a) وضعیت اولیه، بعد از دریافت [None, None, None, None] (b) بعد از یک حرکت و دریافت [None, Breeze, None, None]



دو گام بعدی حرکت عامل (a) بعد از حرکت سوم و دریافت [Stench, None, None, None] (b) بعد از حرکت پنجم و دریافت [Stench, breeze, Glitter, None, None]

منطق[1]:

منطق، یک زبان رسمی برای بازنمایی دانش است بطوریکه می‌توان از آن نتیجه گیری‌هایی نمود. قبلاً اشاره کردیم که پایگاه دانش شامل مجموعه‌ای از جملات است. این جملات بر اساس نحو[2] زبان بازنمایی دانش، نشان داده می‌شوند. این زبان همه جملاتی را که خوش فرم[3] هستند را نشان می‌دهد. یک منطق باید معانی[4] زبان را نیز تعریف کند. آنچه معانی باید انجام دهند مشخص کردن معنای هر جمله است. معانی زبان، درستی یک جمله را در هر دنیای ممکن مشخص می‌کنند. در منطق گزاره‌ای و منطق مرتبه اول هر جمله در هر دنیای ممکن درست یا نادرست است و حالت مابینی نداریم ولی در منطق فازی یک حالت مابین وجود دارد. زبان‌های محدودیت‌ها، منطق‌ها هستند و حل

---

[1] Logic

[2] Syntax

[3] well form

[4] Semantic

محدودیت‌ها فرمی از استنتاج منطقی است. به جای دنیای ممکن از اصطلاح مدل[۱] استفاده می‌کنیم. مدل تعریف دیگری نیز دارد. وقتی می‌گوییم m مدل جمله α است معنایش این است که جمله α در مدل m درست است. مدل‌ها تجریدهای ریاضی هستند و هر کدام درستی یا نادرستی جمله مرتبط را مشخص می‌کنند.

مثال:

α: X + Y = 4 m_1: X = 0, Y = 4 m_2: X = 4, Y = 0

m_3: X = 3, Y = 1 m_4: X = 1, Y = 3 m_5: X = 2, Y = 2

در این مثال مدل‌ها ترکیبات ممکن اعدادی است که به x, y نسبت داده می‌شوند پس مدل ۲ مفهوم دارد:

- ❖ هریک از ترکیبات ممکن بر اساس تعداد گزاره‌های موجود، یک مدل نامیده می‌شود.
- ❖ به دنیایی که جمله α در آن درست باشد مدل جمله‌ی α گفته می‌شود.

نکته: برای n تا گزاره 2^n تا مدل خواهیم داشت.

	p	q
M_1	F	F
M_2	F	T
M_3	T	F
M_4	T	T

استلزام[۲]:

گاهی اوقات ممکن است بخواهیم جملات جدید و تازه‌ای را که الزاماً صحیح هستند تولید و یا استفاده کنیم در حالیکه جملات قدیمی نیز صحیح هستند. چنین ارتباطی بین جملات استلزام نامیده می‌شود.

$\alpha \models \beta$: α مستلزم β است اگر و فقط اگر در هر مدلی که α درست باشد β نیز درست باشد.

(مدل‌های β ⊆ مدل‌های α)

$a \models b$:

- ❖ جمله a استلزام جمله b است.
- ❖ جمله a جمله b را ایجاد می‌کند اگر و فقط اگر، در هر مدلی که a درست است، b نیز درست است
- ❖ اگر a درست باشد، b نیز درست است
- ❖ درستی b در درستی a نهفته است

نکته: پایگاه دانش می‌تواند به عنوان یک جمله در نظر گرفته شود و می‌گوییم یک جمله مستلزم پایگاه دانش است اگر و فقط اگر α در تمام دنیاهایی که در آن KB درست است، درست باشد.

$KB \models \alpha$: در هر مدلی که KB درست است α نیز درست است (مدل‌های α ⊆ مدل‌های KB)

---

[۱] Model

[۲] Entaiment

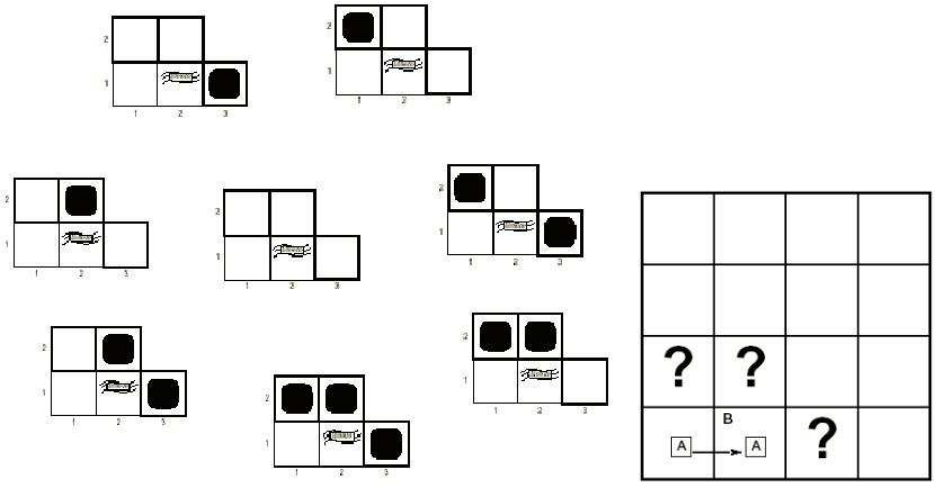
نکته: به طور خلاصه برای استلزام داریم:

- ❖ استلزام بدین معناست که چیزی از چیز دیگری پیروی می‌کند:

$KB \models \alpha$

- ❖ پایگاه دانش KB مستلزم جمله α است، اگر و فقط اگر α در تمام دنیاهایی که در آن KB درست است، درست است.
- ❖ به طور مثال $x + y = 4$ مستلزم $4 = x + y$ می‌باشد.
- ❖ استلزام رابطه‌ای است که بین ساختار جملات (syntax) و بر مبنای معنای جملات تعریف می‌شود.

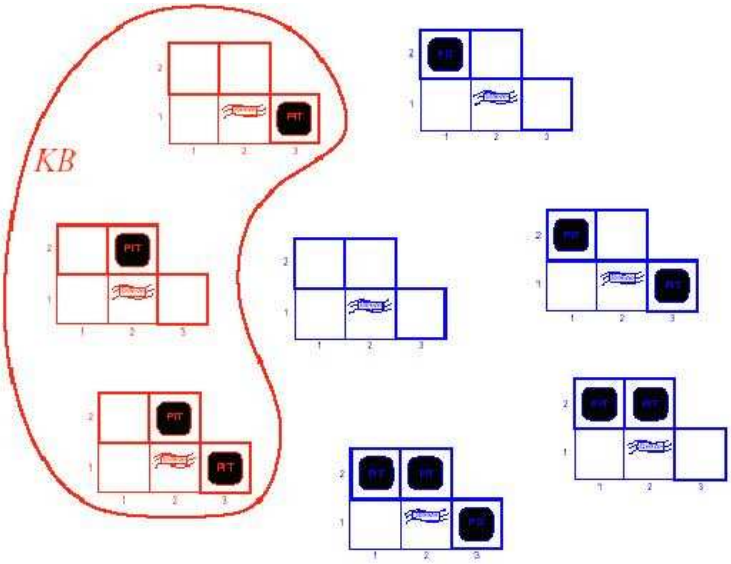
مثال: برای درک بهتر مفهوم استلزام به مثالی از دنیای وامپوز توجه فرمائید:



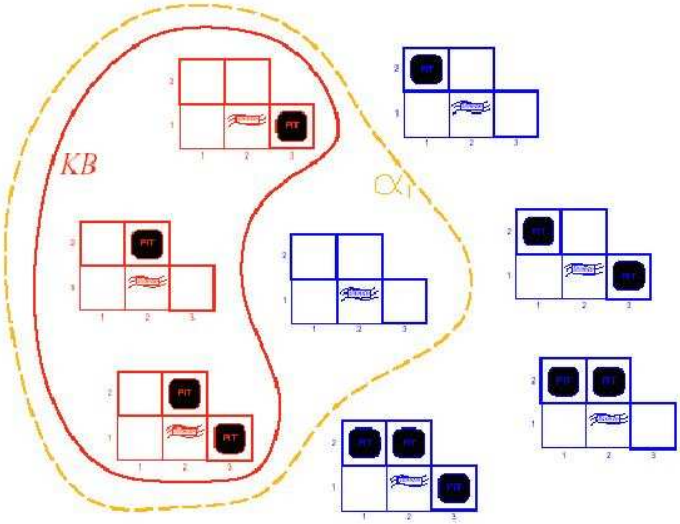
عامل در خانه [1,1] چیزی دریافت نکرده است و در خانه [2,1] نسیم دریافت کرده است. این مشاهدات با دانش عامل در مورد قوانین بازی وامپوز ترکیب شده و به KB اضافه می‌شوند. عامل می‌داند که در یکی از خانه‌های [1,2], [2,2], [3,1] چاله وجود دارد. هرکدام از این ۳ خانه ممکن است چاله داشته باشد یا نداشته باشد بنابراین 2^3 یعنی ۸ مدل ممکن وجود دارد. روش بررسی (شمارش) مدل[1] تمام مدل‌های ممکن را شمارش می‌کند و بررسی می‌کند که آیا در تمام مدل‌هایی که KB درست است α نیز درست می‌باشد؟

[1] Model checking

مدل‌های وامپوس

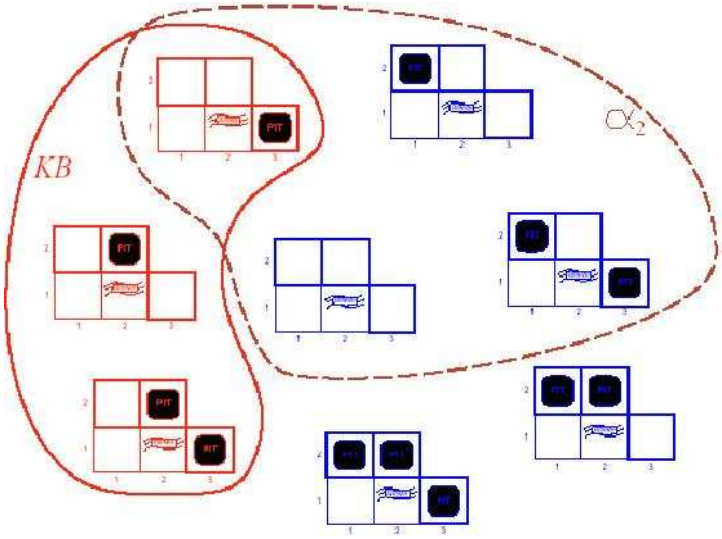


KB = Wumpus - World rules + Observations



KB = Wumpus - World rules + Observations

a_1 = “ [1, 2] is safe “,KB $\models a_1$, proved by model checking



KB = Wampus - World rules + observations

a_2 = “ [2, 2] is safe “, KB $\not\models a_2$

نکته: برای درک هرچه بهتر و تمایز مفهوم استنتاج و استلزام می‌توان مجموعه همه جملات KB را بعنوان انبار کاه و α را به عنوان سوزن در نظر گرفت. استلزام مثل سوزن در این انبار کاه و استنتاج مانند یافتن سوزن در این انبار کاه است. بعبارت دیگر الگوریتم استنتاج i بتواند α را از پایگاه دانش بدست آورد (مشتق کند) می‌نویسیم: $KB \vdash_i \alpha$

صحت[1]:

اگر یک الگوریتم استنتاج فقط جملات استلزام یافته را بدست آورد (مشتق کند) الگوریتم استنتاج صحیح گفته می‌شود. صحت یک خصوصیت مطلوب است یک الگوریتم استنتاج غیر صحیح از کشف سوزن‌هایی که وجود ندارند خبر می‌دهد لذا روش شمارش مدل یک رویه‌ی استنتاج صحیح است. شمارش مدل هنگامی کار می‌کند که فضای مدل متناهی باشد و در ریاضیات که معنای مدل نامتناهی است کار نمی‌کند.

کامل بودن[2]:

یک الگوریتم استنتاج کامل است اگر بتواند همه جملاتی که استلزام می‌شود را بدست آورد. برای انبار کاه وقتی که متناهی است به نظر می‌رسد با بررسی سیستماتیک می‌توان تصمیم گرفت که آیا سوزن در انبار کاه وجود دارد یا خیر. کامل بودن نیز یک خصوصیت مطلوب است ولی بیشتر پایگاه‌های دانش انبار نتایج نامتناهی دارند. در منطق مرتبه اول یک رویه استنتاج کامل و صحیح وجود دارد.

منطق گزاره‌ای:

منطق، مجموعه قواعدی است که به کمک آن می‌توان معتبر بودن یک استدلال را تشخیص داد.

گزاره[3]:

جمله خبری است که می‌تواند ارزش درست یا نادرست داشته باشد. گزاره‌هارا با نماد p, q, r,... نشان می‌دهیم.

گزاره‌ها دو نوع هستند: ساده و مرکب. گزاره ساده قابل تجزیه به گزاره‌های کوچکتر نیست ولی گزاره مرکب از چند گزاره ساده به کمک روابط ترکیبی تشکیل شده است.

روابط ترکیبی:

i. **نقیض (not, ~):** عملگری است که ارزش گزاره را نقض می‌کند.

ii. **ترکیب عطفی (and, ^):** این عملگر دو عملوندی است و فقط زمانی True است که هر دو عملوند آن True باشند.

iii. **ترکیب فصل (or, ∨):** این عملگر دو عملوندی است و فقط زمانی True است که حداقل یکی از گزاره‌ها True باشد.

iv. **ترکیب شرطی (⇒):** زمانی False است که مقدم آن True و تالی آن False باشد.

---

[1] Soundness

[2] completeness

[3] Propositional

v. **ترکیب دو شرطی(⇔):** برروی دو گزاره عمل می‌کند و ارزش آن هنگامی که هر دو گزاره هم ارزش باشند درست است.

جدول صحت: (معنای منطق گزاره‌ها)

P	Q	¬P	P∧Q	P ∨ Q	P ⇒ Q	P ⇔ Q
false	*false*	*true*	*false*	*false*	*true*	*true*
false	*true*	*true*	*false*	*true*	*true*	*false*
true	*false*	*false*	*false*	*true*	*false*	*false*
true	*true*	*false*	*true*	*true*	*true*	*true*

گرامر منطق گزاره‌ها (نحو منطق گزاره‌ها)

sentence →Atomic sentence | Complex sentence

Atomic sentence →True | false | Symbol

Symbol → p | q | r |...

Complex sentence → ~sentence | (sentence ∧ sentence)

|(sentence ∨ sentence)

| (sentence ⇒ sentence)

| (sentence ⇔ sentence)

روش انتفاع مقدم :

هرگاه در یک گزاره شرطی طرف اول نادرست باشد بدون توجه به طرف دوم، ارزش کل گزاره درست خواهد بود.

مثال:

- $\emptyset \subseteq A$
  $x \in \emptyset \rightarrow x \in A$
- اگر 5 زوج باشد ← سام باهوش است.

روش صحت تالی:

هرگاه در یک گزاره شرطی طرف اول (تالی) صحیح باشد طرف دوم ترکیب شرطی صحیح خواهد بود.

یک گزاره مرکب را همیشه درست[1] می‌گوییم هرگاه مستقل از ارزش گزاره‌های تشکیل دهنده آن، همیشه درست باشد.

مثال: $p \rightarrow p$ یا $p \vee \sim p$

	1	1
F	T	T
T	F	T

[1] tatulogy

یک گزاره مرکب را همیشه نادرست می‌گوییم هرگاه مستقل از ارزش گزاره‌های تشکیل دهنده آن، همیشه نادرست باشد.

مثال: $p \wedge \sim p$

p		
F	T	F
T	F	F

نکته: دو گزاره مرکب را هم ارز می‌گوییم، هرگاه ارزش آنها در کلیه حالت‌ها(مدل‌ها) یکسان باشد. $P \equiv Q$

مثال: $\neg(p \wedge q) \equiv \sim p \vee \sim q$

F	F	F	T
F	T	F	T
T	F	F	T
T	T	T	F

p				
F	F	T	T	T
F	T	T	F	T
T	F	F	T	T
T	T	F	F	F

هم ارزی‌های مهم:

♠ جابجایی: $\left.\begin{array}{l} 1)\, p \vee q \equiv q \vee p \\ 2)\, p \wedge q \equiv q \wedge p \end{array}\right\}$

♠ شرکت پذیری: $\left.\begin{array}{l} 3)\, p \vee (q \vee r) \equiv (p \vee q) \vee r \\ 4)\, p \wedge (q \wedge r) \equiv (p \wedge q) \wedge r \end{array}\right\}$

♠ توزیع پذیری: $\left.\begin{array}{l} 5)\, p \vee (q \wedge r) \equiv (p \vee q) \vee (p \vee r) \\ 6)\, p \wedge (q \vee r) \equiv (p \wedge q) \wedge (p \wedge r) \end{array}\right\}$

♠ جذب: $\left.\begin{array}{l} 7)\, p \wedge (p \vee q) \equiv p \\ 8)\, p \vee (p \wedge q) \equiv p \end{array}\right\}$

♠ شبه جذب: $\left.\begin{array}{l} 9)\, p \wedge (\sim p \vee q) \equiv p \wedge q \\ 10)\, p \vee (\sim p \wedge q) \equiv p \vee q \end{array}\right\}$

♠ دمورگان: $\left.\begin{array}{l} 11)\, \sim (p \vee q) \equiv \sim p \wedge \sim q \\ 12)\, \sim (p \wedge q) \equiv \sim p \vee \sim q \end{array}\right\}$

$13)\, p \vee F \equiv p \quad 14)\, p \vee T \equiv T \quad 15)\, p \wedge F \equiv F \quad 16)\, p \wedge T \equiv p \quad 17)\, p \vee \sim p \equiv T$

$18)\, p \wedge \sim p \equiv F \quad 19)\, p \to q \equiv \sim p \vee q \quad 20)\, p \to q \equiv \sim q \to \sim p \quad 21)\, \sim (p \to q) \equiv p \wedge \sim q$

$22)\, p \leftrightarrow q \equiv (p \to q) \wedge (q \to p) \quad 23)\, \sim (p \leftrightarrow q) \equiv \sim p \leftrightarrow q \equiv p \leftrightarrow \sim q$

مثال. درستی هم ارزی زیر را بررسی کنید.

جواب: $[(p \to q) \wedge r] \to p \equiv r \to p$

$$[(\sim p \vee q) \wedge r] \to p \equiv \sim [(\sim p \vee q) \wedge r] \vee p \equiv [(p \wedge \sim q) \vee \sim r] \vee p \equiv [(p \wedge \sim q \vee p] \vee \sim r$$
$$\equiv p \vee \sim r \equiv \sim r \vee p \equiv r \to p$$

گزاره نما:

عبارتی شامل یک یا چند متغیر که متغیرها از مجموعه خاصی عضو می‌پذیرند را گزاره نما می‌گویند (به اعضای این مجموعه، مجموعه‌ی جهانی می‌گوییم). گزاره نما را با $p_x, q_x, \ldots$ نمایش می‌دهیم.

تک متغیره: $p_x : x+4>7 \quad x \in R$

دو متغیره: $p_{x,y} : x^2 + y^2 = 16 \quad x, y \in R$

سورها:

علائمی هستند که یک گزاره نما را به یک گزاره تبدیل می‌کنند.

- $\forall$ **(سور عمومی):** به ازای همه ی مقادر باید درست باشد.
- $\exists$ **(سور و جودی):** به ازای بعضی از مقادیر باید درست باشد.
- $\nexists$ **(سور صفر):** به ازای هیچ مقدار باید درست باشد.
- $\exists!$ **(سور انحصاری):** به ازای یک مقدار باید درست باشد.

$\forall x \in R,\ x^2 > 0 \qquad \forall x \in R,\ x^2 \geq 0 \qquad \forall z \in \not\subset,\ z^2 \geq 0 \qquad \forall x \in z,\ x^2 - 9 = 0 \Rightarrow x = \pm 3$

$\exists!\ x \in z,\ x - 1 = 2 \Rightarrow x = 3$

ترکیب سورها:

$\forall x, \forall y\ \ p(x,y) \qquad \forall x, \exists y\ \ p(x,y) \qquad \exists x, \forall y\ \ p(x,y) \qquad \exists x, \exists y\ \ p(x,y)$

نقیض سورها:

$\sim (\forall x, px) \equiv \exists x, \sim px \qquad \sim (\exists x, px) \equiv \forall x, px$

مثال.

$(\forall x \in R,\ \exists y \in N : xy > 4) \equiv \exists x \in R,\ \forall y \in N : xy \leq 4$

$(\forall x, x \in A, \to x \in) \equiv \exists x, x \in A \wedge x \notin B$

استنتاج[1]:

فرض کنید که گزاره‌های $p_1 \wedge p_2 \wedge p_3 \cdots p_n$ گزاره‌های درست باشند. چنانچه از درستی این گزا ره‌ها، درستی گزاره‌ی q نتیجه شود در این صورت q را یک استلزام منطقی از گزاره‌های $p_1 \wedge p_2 \wedge p_3 \cdots p_n$ گویند و این استنتاج به صورت زیر نشان داده می‌شود.

$$\left.\begin{array}{l} p_1 \\ p_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ p_n \end{array}\right\} \over \therefore q$$

قوانین استنتاج:

الف. قانون انتزاع: $\dfrac{\begin{array}{l} p \to q \\ p \end{array}}{\therefore q}$

ب. قانون نقیض انتزاع: $\dfrac{\begin{array}{l} p \to q \\ \sim q \end{array}}{\therefore \sim p}$

ج. قیاس: $\dfrac{\begin{array}{l} p \to q \\ q \to r \end{array}}{\therefore p \to r}$

د. رفع مولفه: $\dfrac{\begin{array}{l} p \vee q \\ \sim p \end{array}}{\therefore q}$

ه. حذف عاطف: $\dfrac{p \wedge q}{\therefore q}$ یا $\dfrac{p \wedge q}{\therefore p}$

و. ادخال فاصل: $\dfrac{q}{\therefore p \vee q}$ یا $\dfrac{p}{\therefore p \vee q}$

مثال: درستی استنتاج‌های زیر را بررسی کنید:

مثال۱:

$$\begin{array}{l} q \to r \\ p \vee q \\ \sim r \\ \hline \therefore p \end{array} \qquad \begin{array}{l} q \to r \\ \sim r \\ \hline \therefore \sim q \end{array} \qquad \begin{array}{l} p \vee q \\ \sim q \\ \hline \therefore p \end{array}$$

جواب:

مثال۲:

$$\begin{array}{l} s \to q \\ \sim r \to \sim q \\ r \to \sim p \\ s \\ \hline \therefore \sim p \end{array}$$

جواب:

$$\begin{array}{l} s \to q \\ s \\ \hline \therefore q \end{array} \Rightarrow \begin{array}{l} \sim r \to \sim q \\ q \\ \hline \therefore \sim(\sim r) \equiv r \end{array} \Rightarrow \begin{array}{l} r \to \sim p \\ r \\ \hline \therefore \sim p \end{array}$$

مثال ۳:

$$\begin{array}{l} p \vee q \\ r \to \sim p \\ s \to \sim p \\ \sim q \\ \hline \therefore \sim s \end{array}$$

جواب:

$$\begin{array}{l} r \to \sim p \\ \sim p \to s \\ \hline \therefore r \to s \end{array} \Rightarrow \begin{array}{l} p \vee q \\ \sim q \\ \hline \therefore p \end{array} \Rightarrow \begin{array}{l} s \to \sim p \\ p \\ \hline \therefore s \end{array}$$

---

[1] Inference

روش برهان خلف:

در این روش فرض می‌کنیم که حکم برقرار نیست و آن را در KB قرار می‌دهیم. عمل استنتاج را انجام می‌دهیم اگر به تناقض رسیدیم پس نقیض حکم نادرست بوده و خود حکم درست است.

$$\begin{array}{l} \sim p \to q \\ q \to r \\ \underline{\sim r} \\ \therefore p \end{array}$$

مثال: پایگاه دانش زیر را با منطق گزاره‌ها بازنمایی کنید سپس عمل استنتاج را انجام دهید.

« اگر گروه نوازندگان نمی‌توانست موسیقی را اجرا کند یا از حضار به موقع پذیرایی نمی‌شد آنگاه ضیافت سال نو لغو می‌گردید و آقای احمدی عصبانی می‌شد. اگر این ضیافت لغو می‌گردید آنگاه می‌بایستی مبالغ پرداخت شده تحویل داده می‌شد. هیچ تحویلی انجام نشد بنابراین گروه نوازندگان توانست موسیقی را اجرا کند. »

- **P:** گروه نوازندگان موسیقی را اجرا کند
- **r:** لغو شدن ضیافت سال نو
- **q:** از حضار به موقع پذیرایی شود
- **s:** عصبانی شدن آقای احمدی
- **t:** مبالغ پرداخت شده تحویل داده شوند.

$$\begin{array}{l} (\sim p \vee \sim q) \to (r \wedge s) \\ r \to t \\ \underline{\sim t} \\ \therefore p \end{array} \qquad \begin{array}{l} r \to t \\ \underline{\sim t} \\ \therefore \sim r \end{array} \Rightarrow \begin{array}{l} \sim (p \wedge q) \to (r \wedge s) \\ \underline{\sim r} \\ \sim r \vee \sim s \end{array} \Rightarrow \begin{array}{l} \sim p \vee \sim s \\ \underline{} \\ \therefore p \wedge q \end{array} \Rightarrow \begin{array}{l} p \wedge q \\ \underline{} \\ \therefore p \end{array}$$

هم ارزی[1]:

دو جمله هم ارز منطقی می‌باشند اگر و فقط اگر در مدل‌های یکسانی نتایج مشابه داشته باشند.

$$\alpha \equiv \beta \quad If \quad \alpha \models \beta \quad and \quad \beta \models \alpha$$

$(\alpha \wedge \beta)$	$\equiv$	$(\beta \wedge \alpha)$	commutativity of $\wedge$
$(\alpha \vee \beta)$	$\equiv$	$(\beta \vee \alpha)$	commutativity of $\vee$
$((\alpha \wedge \beta) \wedge \gamma)$	$\equiv$	$(\alpha \wedge (\beta \wedge \gamma))$	associativity of $\wedge$
$((\alpha \vee \beta) \vee \gamma)$	$\equiv$	$(\alpha \vee (\beta \vee \gamma))$	associativity of $\vee$
$\neg(\neg\alpha)$	$\equiv$	α	double-negation elimination
$(\alpha \Rightarrow \beta)$	$\equiv$	$(\neg\beta \Rightarrow \neg\alpha)$	contraposition
$(\alpha \Rightarrow \beta)$	$\equiv$	$(\neg\alpha \vee \beta)$	implication elimination
$(\alpha \Leftrightarrow \beta)$	$\equiv$	$((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$	biconditional elimination
$\neg(\alpha \wedge \beta)$	$\equiv$	$(\neg\alpha \vee \neg\beta)$	de Morgan
$\neg(\alpha \vee \beta)$	$\equiv$	$(\neg\alpha \wedge \neg\beta)$	de Morgan
$(\alpha \wedge (\beta \vee \gamma))$	$\equiv$	$((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$	distributivity of $\wedge$ over $\vee$
$(\alpha \vee (\beta \wedge \gamma))$	$\equiv$	$((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$	distributivity of $\vee$ over $\wedge$

[1] Equivalence

معتبر بودن[1]:

یک جمله معتبر است اگر در تمام مدل‌ها درست باشد (جمله همیشه راستگو)

ارضا شدنی[2]:

یک جمله ارضا شدنی است اگر در بعضی (حداقل یک) مدل‌ها درست باشد. یک جمله ارضا نشدنی است اگر در هیچ مدلی درست نباشد.

اگر جمله α در مدل m درست باشد گوییم مدل m جمله α را ارضا می‌کند یا m مدلی از α است. ارضا شدن می-تواند به صورت شمارش مدل‌های ممکن برای یافتن مدلی که جمله را ارضا می‌کند انجام شود. تعیین ارضا شدنی یک جمله در منطق گزاره‌ای اولین مسئله‌ای بود که ثابت شد.

نکته: اعتبار و ارضا شدن به یکدیگر وابسته هستند. α معتبر است اگر و فقط اگر ~α ارضا نشدنی باشد α ارضا شدنی است اگر و فقط اگر ~α معتبر نباشد.

الگوهای استدلال در منطق گزاره‌ای:

در این بخش به معرفی الگوهای استاندارد استنتاج می‌پردازیم. این الگوها برای مشتق کردن زنجیره‌ای از نتایج برای رسیدن به هدف استفاده می‌شوند. به الگوهای استنتاج، قوانین استنتاج نیز گفته می‌شود.

- **قیاس استثنایی (Modus Ponenes):** $\frac{\alpha \Rightarrow \beta, \alpha}{\beta}$
- **حذف and:** $\frac{\alpha \wedge \beta}{\alpha}$ یا $\frac{\alpha \wedge \beta}{\beta}$

این قوانین در هر مثال خاص مورد استفاده قرار می‌گیرند و استنتاج‌های صحیحی را بدون نیاز به شمارش مدل‌ها تولید می‌کنند. علاوه بر این دو قانون، تمام هم ارزی‌های منطقی گفته شده می‌توانند بعنوان قوانین استنتاج به کار گرفته شوند. روش‌های اثبات به دو دسته تقسیم می شوند. اثبات از طریق قوانین استنتاج و اثبات از طریق جدول صحت. این دو روش را روی مثال دنیای وامپوز بررسی می‌کنیم.

جملات دنیای وامپوس

i. اجازه دهید Pij درست باشد، اگر و فقط اگر در خانه [i, j] چاله باشد..

ii. اجازه دهید Bij درست باشد، اگر و فقط اگر در خانه [I, j نسیم باشد.

$\sim P_{1,1}$

$\sim B_{1,1}$

$B_{2,1}$

---

[1] Validity

[2] Satisfaibility

iii. چاله‌ها باعث وزش نسیم در خانه‌های مجاور می‌شوند.

$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

$B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

iv. در یک خانه نسیم می–وزد اگر و فقط اگر چاله‌ای مجاور آن باشد.

$$\begin{cases} R_1 \sim P_{1,1} \\ R_2 : \beta_{1,1} \Leftrightarrow (P_{1,1} \vee P_{2,1}) \\ R_3 : \beta_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1}) \\ R_4 \sim \beta_{1,1} \\ \underline{R_5 \sim \beta_{2,1}} \\ \therefore \sim P_{1,2} \end{cases}$$

اثبات از طریق قوانین استنتاج:

جواب:

اثبات: به این اشتقاق‌ها یعنی بکار گرفتن دنباله‌ای از قوانین استنتاج، اثبات گفته می‌شود. پیدا کردن اثبات‌ها شبیه یافتن راه حل در مسئله جستجو است. در حقیقت علمگرها (تابع تولید کننده بعدی) همان قوانین استنتاج هستند و می‌توان در الگوریتم‌های جستجو از آن‌ها استفاده کرد.

خاصیت یکنوایی[1]:

معنای یکنوایی این است که وقتی مقدم‌های مناسبی مانند β در پایگاه دانش پیدا شوند قوانین استنتاج به کار گرفته می‌شوند و نتایج جدیدی صرف نظر از آنچه در پایگاه دانش قرار دارد به KB اضافه می‌شود. (اگر $KB \models \alpha$ آنگاه $(KB, \beta \models \alpha$

اثبات با استفاده از جدول صحت:

می‌دانیم که هدف استنتاج این است که برای جمله‌ای مثل α تصمیم بگیرد آیا $KB \models \alpha$ یا خیر. اولین الگوریتم استنتاج، پیاده سازی مستقیم تعریف استلزام است. این الگوریتم عبارتست از شمارش مدل‌ها و بررسی اینکه آیا در هر مدلی که KB درست است α نیز درست است. در منطق گزاره‌ای مدل‌ها شامل انتساب True یا False به هر سمبل گزاره‌ای است الگوریتم TT - Entails یک الگوریتم کامل و صحیح است. کامل است زیر برای هر KB، α عمل می‌کند و همیشه خاتمه می‌یابد چون تعداد مدل‌ها متناهی است. این الگوریتم صحیح است زیرا مستقیماً تعریف استلزام را پیاده سازی می‌کند. اگر KB و α حاوی n سمبل گزاره‌ای باشند آنگاه 2^n مدل ممکن وجود دارد. لذا

[1] Monotonocity

بررسی این مدل‌ها پیچیدگی نمایی $O(2^n)$ را خواهد داشت. این الگوریتم پیچیدگی فضایی $O(n)$ دارد زیرا به صورت عمقی شمارش می‌کند.

استفاده از جدول درستی برای استنتاج:

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	KB	α_1
false	*false*	*false*	*false*	*false*	*false*	*false*	*false*	*true*
false	*false*	*false*	*false*	*false*	*false*	*true*	*false*	*true*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	*true*	*false*	*false*	*false*	*false*	*false*	*false*	*true*
false	*true*	*false*	*false*	*false*	*false*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*false*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*true*	*false*	*false*	*false*	*true*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*	*false*

استنتاج به وسیله شمارش:

شمارش تمام مدل‌ها به روش اول عمق صحیح و کامل است.

```
function TT-ENTAILS?(KB, α) returns true or false
   symbols ← a list of the proposition symbols in KB and α
   return TT-CHECK-ALL(KB, α, symbols, [])

function TT-CHECK-ALL(KB, α, symbols, model) returns true or false
   if EMPTY?( symbols) then
      if PL-TRUE(KB, model) then return PL-TRUE(α, model)
      else return true
   else do
      P ← FIRST( symbols); rest ← REST( symbols)
      return TT-CHECK-ALL(KB, α, rest, EXTEND(P, true, model)) and
            TT-CHECK-ALL(KB, α, rest, EXTEND(P, false, model))
```

برای n سیمبول $O(2^n)$

تحلیل[1]:

ایده این روش بر اساس برهان خلف استوار است. به جای اینکه ثابت کند گزاره‌ای مانند q درست است ثابت می‌کند ~q نمی‌تواند درست باشد. بدین منظور، فرضیات اولیه را با نقیض هدف ترکیب می‌کند و عبارات جدیدی تولید می‌کند این عمل را آنقدر انجام می‌دهد تا به تناقض برسد وقتی به تناقض رسید می‌گوید این نقیض هدف بود که باعث شد به هدف نرسیم بنابراین نقیض هدف نادرست و خود هدف درست است. قانون استنتاج Resolution در ترکیب با هر الگوریتم جستجوی کامل، منجر به الگوریتم استنتاج کامل می‌شود.

قانون Resolution واحد، یک عبارت و یک لیترال را گرفته، عبارت دیگری تولید می‌کند.

$$\frac{l_1 \vee \ldots \vee l_k, m}{l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k}$$

[1] Resolution

قانون Resolution واحد را می‌توان به قانون Resolution کامل، تعمیم داد:

$$\frac{l_1 \vee \ldots \vee l_k, m_1 \vee \ldots \vee m_n}{l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n}$$

فرم‌های نرمال:

از آنجائیکه الگوریتم‌های استنتاج روی فرم خاصی از جملات،عمل استنتاج را انجام می‌دهند لذا قبل از شرح هر الگوریتم استنتاج، فرم مربوط به آن الگوریتم را معرفی می‌کنیم. روش تحلیل بر روی جملات به فرم CNF و روش-های زنجیره رو به جلو و زنجیره رو به عقب بر روی جملات به فرم Horn عمل استنتاج را انجام می‌دهند.

فرم نرمال عطفی (cnf):

قانون Resolution فقط به لیترال‌های ترکیب فصلی اعمال می‌شود لذا در بکارگیری آن فقط به پایگاه دانش و پرسش‌هایی شامل این ترکیبات فصلی نیاز است هر جمله درمنطق گزاره‌ها با یک ترکیب عطفی از ترکیبات فصلی لیترال هم ارز است. جمله‌ای که به صورت یک ترکیب عطفی از ترکیبات فصلی لیترال‌ها بیان می‌شود فرم نرمال عطفی یا CNF گفته می‌شود.

روش‌های بدست آوردن CNF:

i. حذف ترکیبات شرطی
ii. کاهش دامنه نفی توسط دمورگان
iii. توزیع پذیری

به مثال زیر در دنیای وامپوز دقت کنید:

$$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$$

۱۱ $[B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})] \wedge [(P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}]$
$[\neg B_{1,1} \vee (P_{1,2} \vee P_{2,1})] \wedge [\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1}]$

۲۱ $[\sim B_{1,1} \vee (P_{1,2} \vee P_{2,1})] \wedge [(\sim P_{1,2} \wedge \sim P_{2,1}) \vee B_{1,1}]$

۳۱ $[\sim B_{1,1} \vee P_{1,2} \vee P_{2,1}] \wedge [(B_{1,1} \vee P_{1,2}) \wedge (B_{1,1} \vee \sim P_{2,1})]$
$\equiv [(\sim B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (B_{1,1} \vee \sim P_{1,2}) \wedge (B_{1,1} \vee \sim P_{2,1})]$

الگوریتم Resolution:

برای اینکه نشان دهیم KB|=α، مشخص می‌کنیم (KB |≠ α) ارضا نشدنی است.

- ابتدا (KB |≠ α) را به CNF تبدیل می‌کنیم.
- سپس قانون Resolution به عبارات کوچک حاصل اعمال می‌شود.
- هر جفتی که شامل لیترال‌های مکمل باشد، Resolution می‌شود تا عبارت جدیدی ایجاد گردد.
- اگر این عبارت قبلاً در مجموعه نباشد، به آن اضافه می‌شود.

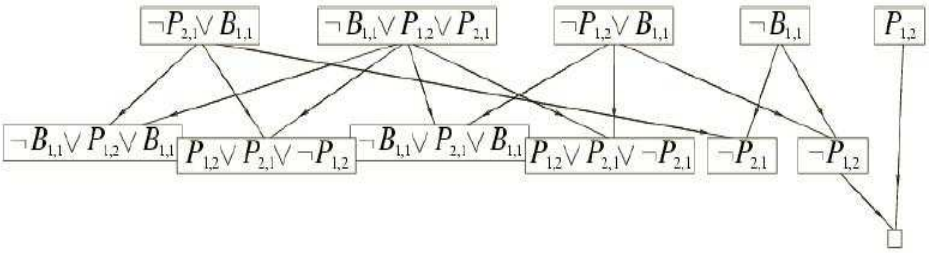
❖ این فرایند تا محقق شدن یکی از شروط زیر ادامه می‌یابد:

- هیچ عبارت دیگری وجود نداشته باشد که بتواند اضافه شود. در این صورت، KB جمله α را نتیجه نمی‌دهد
- کاربرد قانون Resolution، عبارت تهی را ایجاد می‌کند. در این صورت، KB جمله α را نتیجه می‌دهد.

مثال:

$$\boldsymbol{KB} = \left(B_{1,1} \Leftrightarrow \left(P_{1,2} \vee P_{2,1}\right)\right) \wedge \sim B_{1,1}$$

$$\boldsymbol{a} = \sim \boldsymbol{P_{1,2}}$$



عبارات هورن[1]:

ترکیب فصلی لیترال‌هایی که حداکثر یکی از آن‌ها مثبت باشد را عبارات هورن می‌گوئیم. محدود شدن به یک لیترال مثبت ممکن است جالب نباشد اما به سه دلیل زیر مهم است:

هر عبارت هورن را می‌توان به صورت یک استلزام (ترکیب شرطی) نوشت که مقدم آن ترکیب عطفی لیترال‌های مثبت و تالی آن یک لیترال مثبت است. به آن نوع از عبارات هورن که فقط یک لیترال مثبت دارند، عبارات معین[2] گفته می‌شود. لیترال مثبت را رأس و لیترال‌های منفی را بدنه عبارت گویند. عبارت معینی که فاقد لیترال‌های منفی باشد حقیقت[3] نام دارد. عبارات معین اساس برنامه نویسی منطقی را می‌سازند. برخی جملات هورن، لیترال مثبت ندارند این نوع جملات در دنیای بانک اطلاعاتی محدودیت‌های جامعیتی نامیده می‌شود. در الگوریتم‌های زنجیره رو به جلو و زنجیره رو به عقب برای سهولت فرض می‌شود که پایگاه دانش فقط شامل فراکردهای معین می‌باشد و محدودیت‌های جامعیتی وجود ندارد.

---

[1] Horn clause

[2] Define clause

[3] fact

عبارات هورن و ترکیب شرطی متناظر با آن‌ها در مثال زیر آورده شده است:

$$\sim A \vee \sim B \vee \sim C \vee D \rightarrow A \wedge B \wedge C \Rightarrow D$$
$$\sim A \vee \sim B \rightarrow A \wedge B \Rightarrow False$$

مثال: نمونه‌ای از محدودیت جامعیتی و ترکیب شرطی متناظر با آن در زیر آورده شده است:

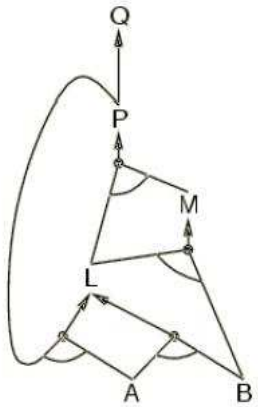
$$\sim W_{1,1} \vee W_{1,2} \qquad \sim W_{1,1} \vee W_{1,2} \vee false \equiv W_{1,1} \vee W_{1,2} \Rightarrow false$$

نکته: الگوریتم زنجیره ساز رو به جلو و زنجیره ساز رو به عقب، عمل استنتاج را فقط روی جملات هورن انجام می-دهند. تصمیم گیری در مورد استلزام با جملات هورن در زمان خطی بر حسب اندازه KB انجام می‌شود.

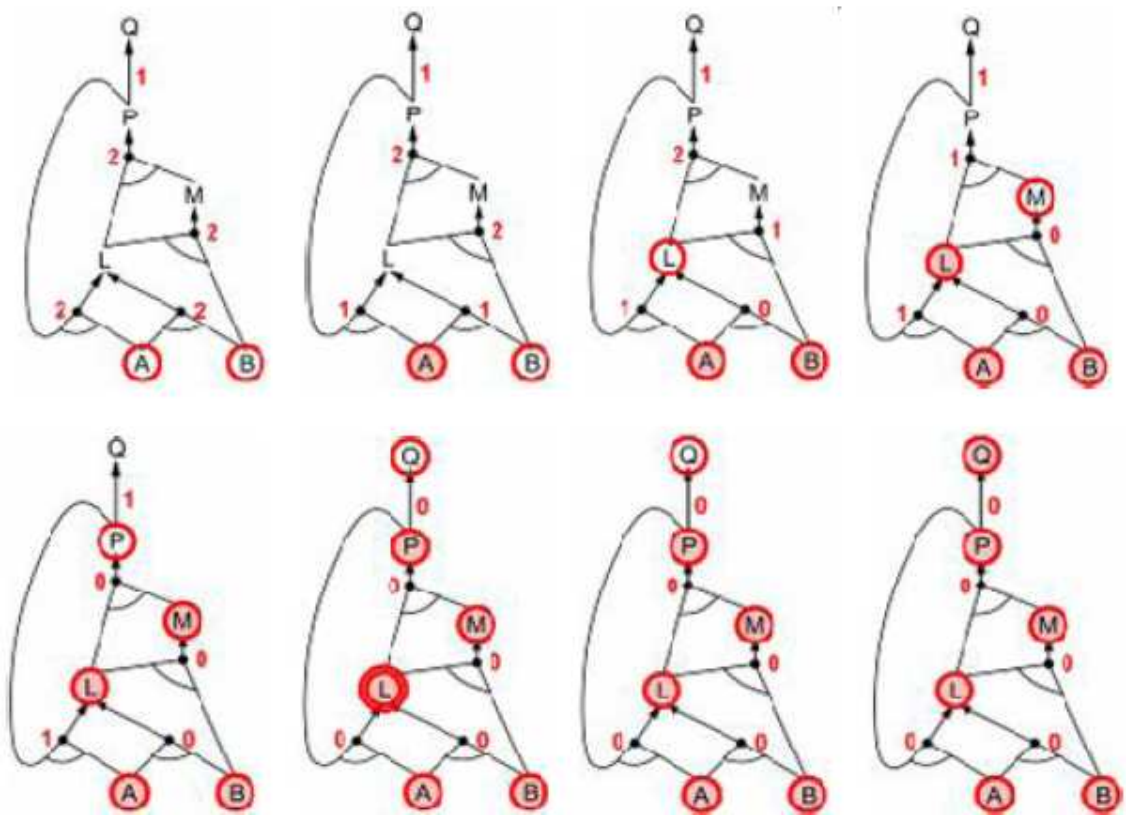
زنجیره ساز رو به جلو[1] (FC):

برای اینکه ثابت کنیم گزاره‌ای مانند q از KB مشتق می‌شود (بدست می‌آید) باید از فرضیات اولیه شروع کنیم و هر ترکیب شرطی که مقدم آن درست باشد تالی آن به KB اضافه می‌شود تا هنگامیکه یا نتیجه بدست آید یا امکان استنتاج وجود نداشته باشد.

$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$



[1] Forward chaining



الگوریتم زنجیره ساز رو به جلو از فرضیات شروع و به هدف می‌رسد و مبتنی بر داده[1] می‌باشد الگوریتم زنجیره ساز رو به جلو تعیین می‌کند آیا پرسش گزارهٔ q توسط پایگاه دانش جملات هورن مشتق می‌شود یا خیر؟ پیچیدگی زمانی الگوریتم زنجیره ساز رو به جلو، خطی است. FC یگ الگوریتم صحیح است زیرا استنتاج‌های انجام شده در آن کاربردی از قانون انتزاع است. زنجیره ساز رو به جلو در KB به فرم هورن کامل است زیرا تمام جملات اتمی را مشتق می‌کند. زنجیره ساز رو به جلو نمونه‌ای از مفهوم کلی داده گراست یعنی استدلالی که در آن داده‌ها مشخص و ثابت هستند.

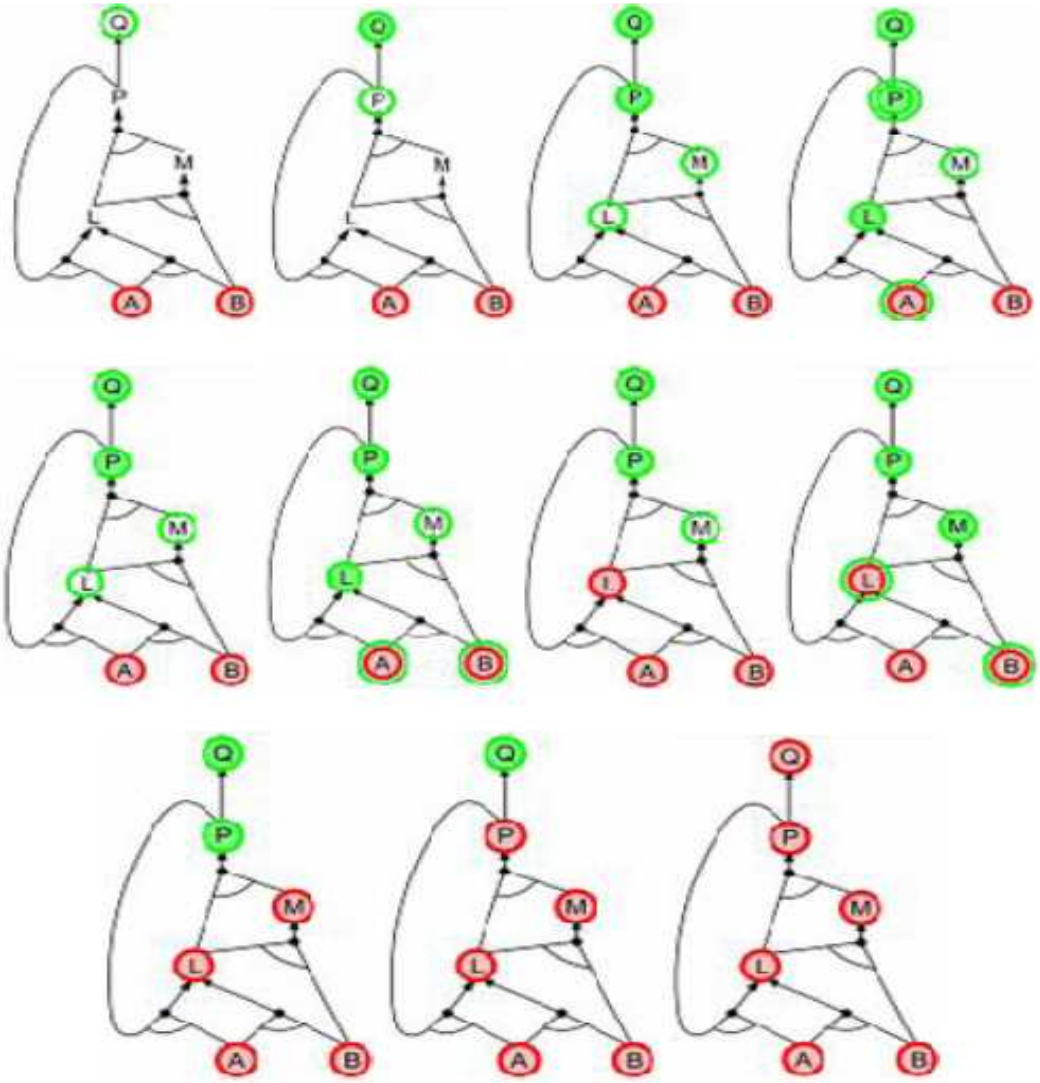
زنجیره ساز رو به عقب[2] (BC):

این الگوریتم از جمله‌ی پرسش شروع می‌کند و به عقب بر می‌گردد اگر پرسش q درست باشد کار تمام است و گر نه الگوریتم، جملات شرطی را در پایگاه دانش می‌یابد که تالی آن‌ها q باشد. اگر بتوان ثابت کرد تمام مقدم‌های یکی از آن جملات شرطی درست است آن گاه می‌توان نتیجه گرفت q درست است. این الگوریتم نوعی استدلال هدف گراست یعنی برای پاسخ به پرسش‌هایی مثل اکنون به چه چیزی باید جواب بدهم؟ کلیدهایم کجاهستند؟ و... مفیداست. اغلب هزینه این الگوریتم بر حسب اندازه KB کمتر از هزینه خطی است چون زنجیره ساز رو به عقب فقط با حقایق مرتبط برخورد می‌کند.

---

[1] Data-driven

[2] Backward chaining



مقایسه دو روش FC, BC:

FC بر مبنای داده است ولی BC بر مبنای هدف است در FC ممکن است کارهای بسیاری را انجام دهیم که به هدف مربوط نمی‌شود ولی در BC فقط با حقایق مرتبط با هدف، برخورد می‌شود و به همین دلیل پیچیدگی BC می‌تواند بسیار بهتر از خطی نسبت اندازه‌ی KB باشد چون موارد بیهوده را بسط نمی‌دهد.

کاربرد زنجیره‌ای رو به جلو و زنجیره‌ای رو به عقب:

i. در سیستم‌های انتگرال گیری از راه تجزیه به انتگرال‌های ساده تر، از استنتاج رو به جلو استفاده می‌شود چون نقطهٔ آغازین، مشخص و ثابت است.

ii. در سیستم‌های خودکار اثبات قضایا بهتر است از استدلال رو به عقب استفاده شود چون فاکتور انشعاب آن‌ها از طرف مقدم می‌تواند بسیار زیاد باشد.

iii. در سیستم‌های تشخیص پزشکی از استنتاج رو به عقب استفاده می‌شود چون با کاربری سروکار داریم که از آن پرسش می‌کنیم.

iv. برای سیستم‌های چیدمان اشیا بهتر است از زنجیره ساز رو به جلو استفاده شود چرا که هدف معینی ندارد.

به طور خلاصه زنجیره ساز رو به جلو در پیشگیری، طراحی و کنترل مورد استفاده قرار می‌گیرد و زنجیره ساز رو به عقب درتشخیص مورد استفاده قرار می‌گیرد.

نکته: به طور خلاصه برای مقایسه دو روش داریم:

❖ FC

- بر مبنای داده (data driven)
- ممکن است کارهای بسیاری انجام دهد که به هدف مربوط نمی‌شوند.

❖ BC:

- بر مبنای هدف (goal driven)
- پیچیدگی BC می‌تواند بسیار بهتر از خطی نسبت به اندازه KB باشد.

سوالات تستی آخر فصل

ترم اول ۸۷–۸۸

۱. فرض کنید مجموعه گزاره $\{q \rightarrow False\ ,\ q \leftrightarrow (p_2 \vee p_3)\}$ درست است آنگاه کدامیک از عبارات زیر را می‌توان از مجموعه گزاره بالا نتیجه گرفت؟

الف. q ب. $\sim p_2, \sim p_3$ ج. $p_2 \vee p_3$ د. $q \wedge (p_2 \vee p_3)$

ترم تابستان ۸۸

۲. کدام گزینه صحیح نیست؟

الف. جمله $(A \rightarrow B) \rightarrow C$ با جمله $A \rightarrow (B \rightarrow C)$ تفاوت معنایی ندارد.

ب. الگوریتم استنتاجی که فقط جملات ایجابی را بدست می آورد، صحیح نامیده می‌شود.

ج. ترکیب $A \leftrightarrow B \leftrightarrow C$ به پرانتز نیازی ندارد.

د. یک الگوریتم استنتاج کامل است در صورتی که بتواند هر جمله ایجاب شدنی را بدست آورد.

۳. کدام گزینه در مورد ایجاب (=|) صحیح نیست؟

الف. $\alpha \models \beta$ اگر و فقط اگر در هر مدلی که α در آن درست باشد، β نیز درست باشد.

ب. $\alpha \equiv \beta$ اگر و فقط اگر $\alpha \models \beta$ و $\beta \models \alpha$.

ج. $\alpha \models \beta$ اگر و فقط اگر $\alpha \wedge \neg\beta$ ارضاپذیر باشد.

د. $\alpha \models \beta$ اگر و فقط اگر $\alpha \rightarrow \beta$ معتبر باشد.

۴. اگر KB تهی باشد و $TELL(KB, S_1), \ldots, TELL(KB, S_n)$ را انجام دهیم آنگاه KB معادل کدام گزینه می‌باشد؟

الف. $S_1 \wedge \ldots \wedge S_n$ ب. $S_1 \vee \ldots \vee S_n$ ج. $S_1 \Rightarrow \ldots \Rightarrow S_n$ د. $S_1 \oplus \ldots \oplus S_n$

۵. کدام قاعده استنتاج به همراه هر الگوریتم جستجوی کامل می‌تواند به تنهایی یک الگوریتم استنتاج کامل ایجاد کند؟

الف. قیاس استثنایی (Modus ponens) ب. حذف عطف (AND Elimination)

ج. تحلیل (Resolution) د. دمورگان (de Morgan)

۶. شرط خروج الگوریتم تحلیل برای $KB \models \alpha$ چیست؟

الف. هیچ بند جدیدی که بتواند اضافه شود وجود نداشته باشد که در این صورت α از KB نتیجه نمی‌شود.

ب. بندی تهی ایجاد شود که در اینصورت α از KB نتیجه می‌شود.

ج. بندی تهی ایجاد شود که به معنی عدم نتیجه گیری α از KB است.

د. گزینه الف و ب

۷. کدام گزینه یک بند هورن نمی‌باشد؟

الف. $\neg P \vee \neg Q \vee R$ ب. $\neg P \wedge Q \Rightarrow R$ ج. $\neg P \vee Q \vee \neg R$ د. $P \wedge Q \Rightarrow R$

۸. الگوریتم زنجیره‌ای پس رو شکلی از..... است.

الف. منطق مرتبه اول ب. منطق فازی ج. استدلال منطقی د. استدلال هدف گرا

ترم اول ۸۸–۸۹

۹. الگوریتم زنجیره‌ای پیشرو (PL - FC - Entails) با چه جملاتی قادر به تصمیم گیری است؟

الف. جملات گزاره‌ای دلخواه ب. بندهای CNF

ج. بندهای هورن د. بندهای معین

۱۰. کدامیک شکلی از استدلال هدفگرا (Goal - directed) می‌باشد؟

الف. الگوریتم زنجیره‌ای پس رو ب. الگوریتم زنجیره‌ای پیش رو

ج. الگوریتم تحلیل (Resolution) د. TT - Entails

۱۱. در الگوریتم DPLL اگر مدل دارای C = True و D = True، A = False باشد، کدامیک از بندهای سؤال زیر، بند واحد (unit - clause) است؟

1) $A \vee \neg B$

2) $\neg A \vee C \vee D$

3) $\neg C \vee \neg B \vee \neg D$

الف. ۱ ب. ۲

ج. ۳ د. ۱و۳

۱۲. نتیجه حل دوبند زیر کدام بند است؟(تحلیلResolution)

1) $\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}$

2) $\neg P_{2,1} \vee B_{1,1}$

الف. $\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}$ ب. $P_{1,2} \vee P_{2,1} \vee \neg P_{1,2}$ ج. $\neg B_{1,1} \vee P_{1,2} \vee B_{1,1}$ د. الف یا ب

ترم دوم ۸۸–۸۹

۱۳. الگوریتم استنتاجی که فقط جملات ایجابی را بدست آورد..... نامیده می‌شود.

الف. کامل ب. صحیح ج. ارضاپذیر د. تحلیل

۱۴. کدام عبارت یک بند هورن است؟

الف. $\neg L_{1,1} \vee \neg Breeze \vee B_{1,1}$ ب. $\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}$

ج. $\neg L_{1,1} \wedge \neg Breeze \wedge B_{1,1}$ د. $\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}$

ترم تابستان ۸۹

۱۵. یک مدل (model) در منطق چیست؟

الف. مجموعه‌ایی از قواعد استنتاجی Sound

ب. دنباله‌ای از اعمال روال‌های استنتاجی برای اثبات یک جمله

ج. دنیایی است که در آن یک جمله تحت تفسیر خاصی معتبر است.

د. مجموعه جملاتی که از روی آن‌ها می‌توان قابل نتیجه گیری بودن یک جمله خاص را اثبات نمود.

۱۶. اگر G مجموعه جملات یک پایگاه دانش به زبان منطق باشد و P یک جمله به زبان منطق، گوییم P نتیجه منطقی (Entailment)G است اگر وفقط اگر:

الف. مدلی وجود داشته باشد که هم جملات G و هم P را ارضا (Satisfy) کند.

ب. هرمدلی که P را ارضا می‌کند، همه جملات P را هم ارضا می‌کند.

ج. هر مدلی که حداقل یکی از جملات G را ارضا می‌کند، P را هم ارضا می‌کند.

د. هر مدلی که همه جملات G را ارضا می‌کند، P را هم ارضا می‌کند.

۱۷. الگوریتم TT - Entails برای تصمیم گیری در مورد ایجاب گزاره‌ای از چه روشی استفاده می‌کند؟

الف. جدول درستی ب. تحلیل (Resolution)ج. زنجیره‌ای پیش رو د. زنجیره‌ای پیش رو

۱۸. جمله $P \vee \neg P$ کدام است؟

الف. معتبر (Valid) ب. ارضاپذیر(Satisfible)

ج. ارضاناپذیر (Unsatisfible) د. نامعتبر (Invalid)

۱۹. استدلال با زنجیره‌ای پیش رو (Forward Chaining):

الف. همیشه کامل است. ب. روی بندهای معین (Definite Clause) کامل است.

ج. اصلا کامل نیست. د. روی CNF (Conjunctive Normal Form) کامل است.

۲۰. در الگوریتم DPLL، در سه بند زیر کدام نماد محض (pure) می‌باشد؟

1)$A \vee \neg B$

2)$\neg A \vee C \vee D$

3)$\neg C \vee \neg B \vee \neg D$

الف. A ب. B ج. C د. D

۲۱. کدام گزینه در مورد الگوریتم WALKSAT صحیح نیست؟

الف. اگر مدلی را برگرداند، جمله ورودی واقعاً ارضاپذیر است.

ب. اگر failure برگرداند، جمله ورودی واقعاً ارضاپذیر نیست.

ج. در هر مرحله یک بند ارضا نشده را انتخاب کرده و مقدار نمادی از آن را در مدل عوض می‌نماید.

د. به روش حداقل تناقضات در CSPها شباهت زیادی دارد.

ترم اول ۸۹–۹۰

۲۲. الگوریتم TT - Entails برای تصمیم گیری در مورد ایجاب گزاره‌ای از چه روشی استفاده می‌کند؟

الف. جدول درستی ب. تحلیل (Resolution)

ج. زنجیره‌ای پیشرو د. زنجیره‌ای پس رو

۲۳. دانش در کدام عامل غیر انعطاف تر می‌باشد؟

الف. حل مسئله – جستجوگر – (هدف گرا) ب. مبتنی بر منطق

ج. عامل‌های سودمند د. مبتنی بر دانش

۲۴. یک جمله ارضا شدنی (Satisfiable) است اگر و فقط اگر:

الف. در هر مدلی از جهان صحیح باشد.

ب. با قوانین نحوی یک زبان منطقی ساخته شده باشد.

ج. بتواند توسط یک روال استنتاجی اثبات شود.

د. تفسیری از جهان وجود داشته باشد که جمله تحت آن صحیح باشد.

۲۵. کدامیک از جملات زیر به صورت هورن (horn) نوشته شده است؟

الف. $P_1 \wedge P_2 \wedge P_3 \wedge Q_1$ ب. $P_1 \wedge P_2 \wedge P_3 \Rightarrow Q_1 \wedge Q_2$

ج. $P_1 \wedge P_2 \wedge P_3 \Rightarrow Q_1$ د. $P_1 \vee P_2 \vee P_3 \Rightarrow Q_1 \wedge Q_2$

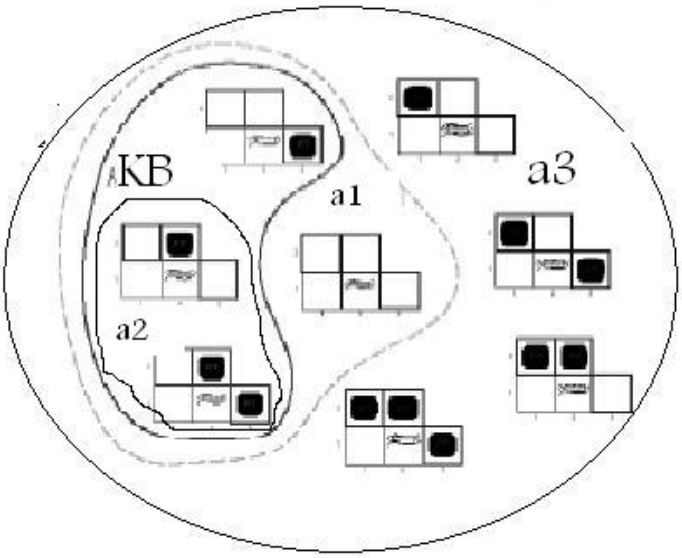
۲۶. در سه بند زیر کدامیک محض (pure) می‌باشد؟

$$\begin{cases} 1) \quad A \vee \neg B \\ 2) \quad \neg A \vee C \vee D \\ 3) \quad \neg C \vee \neg B \vee \neg D \end{cases}$$

الف. A ب. B ج. C د. D

ترم دوم ۸۹–۹۰

۲۷. با توجه به مدل‌های KB, a1, a2, a3 کدام گزینه درست است؟



الف. a2 |= KB

ب. KB |= a2

ج. KB |= a1

د. KB |= a1 v a3

۲۸. اگر عامل در دنیای ومپوز از مربع (۳.۲) وارد مربع (۳.۳) شود و نسیم را حس کند در این وضعیت برای مکان گودال چند مدل وجود دارد؟ (**راهنمایی:** واضح است که عامل می‌داند در مربع (۳.۲) گودالی وجود ندارد.)

		نسیم	■

الف.۸ ب. ۶

ج. ۷ د. ۵

۲۹. از اعمال یک مرحله حل روی زوج بندهای زیر کدام بندهای جدید حاصل می‌شوند؟

(مربوط به الگوریتم تحلیل Resolution)

$\neg P_{2,1} \vee B_{1,1}$ $B_{1,1} \vee P_{1,2} \vee P_{2,1}$ $\neg P_{1,2} \vee B_{1,1}$ $P_{1,2}$

۱) $B_{1.1}$

۲) $B_{1,1} \vee P_{2,1}$

۳) $B_{1,1} \vee P_{1,2}$

۴) $B_{1,1} \vee \sim P_{2,1}$

الف. ۴و۳و۱ ب. ۳و۲ ج. ۱و۲و۴ د. ۱و۲و۳

۳۰. کدام گزینه جزء محدودیت‌های یکپارچگی (Integrity) به شمار می‌آید؟

الف. $\neg w_{1,1} \vee \neg w_{1,2}$ ب. $w_{1,1} \wedge w_{1,2}$

ج. $\neg w_{1,1} \vee w_{1,2}$ د. $w_{1,1} \vee w_{1,2}$

تابستان ۹۰

۳۱. گزینه درست را انتخاب نمایید.

الف. بند هورن، ترکیب عطفی لیترال‌هایی است که حداکثر یکی از آنها مثبت باشد.

ب. الگوریتم زنجیره‌ای پس رو نمونه‌ای از استدلال داده گراست.

ج. زمان تصمیم گیری در مورد ایجاب با بندهای هورن توسط الگوریتم زنجیره پیش رو بر حسب اندازه پایگاه دانش خطی می‌باشد.

د. در بند معین، بیش از یک لیترال مثبت وجود دارد.

۳۲. شکل نرمال عطفی (CNF) عبارت زیر کدام است؟

$B_{1,1} \leftrightarrow (P_{1,2} \vee P_{2,1})$

الف. $(\sim B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (P_{1,2} \vee P_{2,1}) \wedge (\sim P_{1,2} \vee B_{1,1})$

ب. $(\sim B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\sim P_{1,2} \vee B_{1,1}) \wedge (\sim P_{2,1} \vee B_{1,1})$

ج. $(\sim B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\sim P_{1,2} \vee P_{2,1}) \wedge (\sim P_{1,2} \vee P_{2,1})$

د. $(\sim B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (P_{1,2} \vee P_{2,1}) \wedge (P_{2,1} \vee \sim B_{1,1}) (P_{1,2} \vee \sim B_{1,1})$

پاسخ نامه تستی

سوال	گزینه صحیح	سوال	گزینه صحیح	سوال	گزینه صحیح
۱	ب	۱۶	د	۳۱	ج
۲	الف	۱۷	الف	۳۲	ب
۳	ج	۱۸	الف		
۴	الف	۱۹	ب		
۵	ج	۲۰	ب		
۶	د	۲۱	ب		
۷	ب	۲۲	الف		
۸	د	۲۳	الف		
۹	د	۲۴	د		
۱۰	الف	۲۵	ج		
۱۱	د	۲۶	ب		
۱۲	ج	۲۷			
۱۳	الف	۲۸			
۱۴	الف	۲۹			
۱۵	ج	۳۰	الف		

سوالات تشریحی آخر فصل

ترم دوم ۸۷–۸۸

دنیای wumpus زیر را در نظر بگیرید. (عامل: AG، طلا:GLD، چاله:PIT، وامپوز:WU)

برای نشان دادن این هدف که در خانه [۲و۱] چاله‌ای وجود ندارد یعنی ($\alpha=-P_{1,2}$)

الف. ادراک عامل به ازای حضور عامل در خانه‌های (۱،۱)، (۲،۱)، (۲،۱)، (۲،۲)، (۲،۳) را به صورت نمادهای گزاره‌ای بنویسید.

ب. جملاتی از پایگاه دانش (KB) را بنویسید که برای رسیدن به هدف ضروری هستند.

ج. توسط الگوریتم "PL .Resolution"، هدف را بدست آورید. (با ذکر قوانین استفاده شده)

	2,4		PIT
PIT		PIT	GLD
1,2			
AG		WU	

ترم تابستان ۸۸

الف. در دنیای وامپوز زیر ،جمله زیر را با استفاده از منطق گزاره‌ها به ازای خانه y و x بنویسید:

"در یک خانه نسیم می‌وزد اگر و فقط اگرچاله‌ای مجاور آن باشد."

جمله فوق را برای خانه‌های (۱و۱) و (۱و۲) بازنویسی کنید. (۵/. نمره)

Stench		Breeze	PIT
Wumpus	Breeze Gold Stench	PIT	Breeze
Stench		Breeze	
Start	Breeze	PIT	Breeze

ب. جمله زیر را در مسئله دنیای وامپوز، به شکل نرمال عطفی (CNF) تبدیل کنید. (۵/. نمره)

$$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$$

ترم دوم ۸۸–۸۹

شکل نرمال عطفی(CNF) عبارت زیر را بدست آورید. (۱نمره)

$R \Leftrightarrow (P \vee Q)$

ترم تابستان ۸۹

در الگوریتم DPLL روش برخورد با نماد محض (Pure symbol) و بند واحد (Unit clause) را با مثالی تشریح نمایید و مشخص کنید که چرا این رویه باعث افزایش سرعت الگوریتم شده است؟

تابستان ۹۰

با استفاده از الگوریتم زنجیره‌ای پیشرو، آیا نماد گزاره‌ای Q~ توسط پایگاه دانش داده شده در زیر قابل استنتاج است؟ مراحل را بنویسید. گراف And - Or مربوط را ترسیم نمایید. (۱ نمره)

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

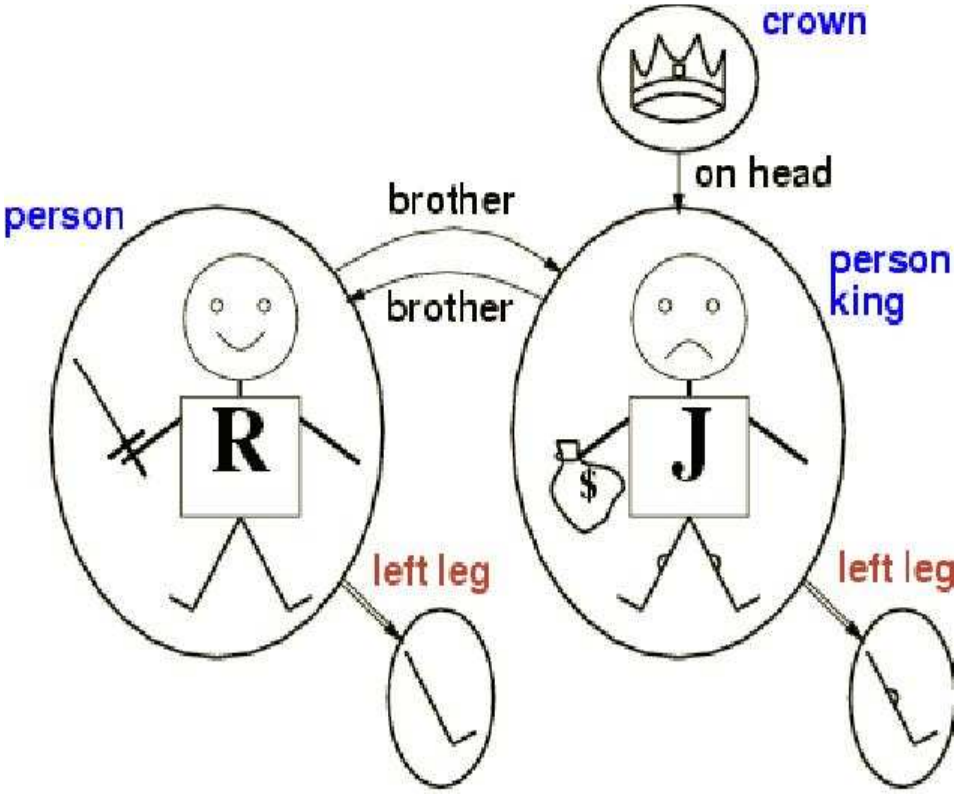
A

B

فصل هشتم: منطق مرتبه اول

آنچه در این فصل خواهید آموخت:

- ❖ **چرا منطق مرتبه اول(FOL) ؟**
- ❖ **انواع منطق**
- ❖ **بررسی ساختار(نحو) و معانی جملات در FOL**
- ❖ **معرفی چند دامنه نمونه با استفاده از FOL**
- ❖ **مهندسی دانش**



مروری بر ویژگی‌های منطق گزاره‌ها:

i. منطق گزاره‌ها یک شیوه توصیفی است که در آن دانش و استنتاج از یکدیگر مجزا هستند و استنتاج کاملاً مستقل از دامنه است.

ii. منطق گزاره‌ها یک زبان اعلانی است زیرا معنای آن بر پایه یک رابطه صحیح بین جملات و دنیاهای ممکن قرار دارد. علاوه بر این، منطق گزاره‌ها دارای قدرت بیان کافی برای اداره کردن اطلاعات جزئی با استفاده از ترکیب‌های فصلی و نقیض است.

iii. یکی دیگر از ویژگی‌های منطق گزاره‌ای، قابلیت ترکیب است که در زبان‌های بازنمایی دانش مفید می‌باشد. در یک زبان ترکیبی معنای یک جمله تابعی از معنای اجزای آن است.

iv. در منطق گزاره‌ای، معنا مستقل از متن است، برخلاف زبان‌های طبیعی که معنای جملات وابسته به متن است.

v. منطق گزاره‌ای دارای معنای ترکیبی – اعلانی است که مستقل از محتوا و غیر مبهم است و قدرت بیان بسیار محدودی در نمایش دانش دارد.

منطق مرتبه اول[۱] (FOL)

این منطق، اساس منطق گزاره‌ها را پذیرفته ولی می‌خواهد بر اساس منطق گزاره‌ها، منطق گویاتری بسازد که ایده‌های بازنمائی را از زبان‌های طبیعی می‌گیرد و از اشکالات آن اجتناب می‌کند. وقتی به نحو زبان طبیعی توجه می‌کنیم در آن موارد زیر را خواهیم داشت:

اشیاء[۲]: برخی از عناصر، اسم‌ها هستند که نشان دهنده‌ی اشیاء می‌باشند. دنیا از یکسری اشیایی تشکیل شده که توسط خصوصیاتشان از یکدیگر متمایز می‌شوند، مثل رنگ‌ها، اعداد، کشورها و ...

رابطه‌ها[۳]: عناصر دیگر، فعل‌ها یا عبارات فعلی هستند که رابطه بین اشیاء را نمایش می‌دهند. مثل رابطه زوج بودن، اول بودن برای اشیاء از نوع اعداد و ...

انواع رابطه‌ها:

i. **رابطه‌های یکانی:** prim(5)

ii. **رابطه دوتایی:** brother(richard, jack)

iii. **توابع:** plus (2,3) = 5 (دو شیء می‌گیرد و خروجی آن یک شیء دیگر است.)

مثال: برای جملات داده شده، اشیا، روابط و توابع را مشخص کنید:

۱) One plus two equals three

(یک بعلاوه دو برابر سه)

---

[۱] First Order Logic

[۲] Object

[۳] Relation

Object: one, two , three (اشیاء)

Relation: equals (رابطه)

Function: plus (تابع)

۲) Square niegboring the wampus smelly

(مربع‌های مجاور وامپوز بو می‌دهند.)

Object: square, wampus (اشیاء)

Relation: niegboring (رابطه)

Function: smelly (تابع)

انواع منطق:

انواع منطق را از دو دیدگاه هستی شناسی و حقیقت شناسی بررسی می‌کنیم. بنابراین بهتر است ابتدا این دو مفهوم بررسی شوند.

I. هستی شناسی[۱]

تفاوت عمده‌ی بین منطق مرتبه اول و منطق گزاره‌ها به هستی شناسی برمی‌گردد. هستی شناسی یک زبان یعنی آنچه این زبان در مورد یک حقیقت (مسئله) فرض می‌کند. به عنوان مثال منطق گزاره‌ای فرض می‌کند حقایقی وجود دارند که ممکن است رخ بدهند یا ندهند. منطق مرتبه اول فرض می‌کند که دنیا شامل اشیاء و روابط بین آنهاست که ممکن است این رابطه‌ها برقرار باشد یا نباشد. منطق‌های خاص منظوره، هستی شناسی بیشتری دارد. به عنوان مثال منطق موقتی فرض می‌کند که حقایق در زمان‌های خاص رخ می‌دهد.

II. حقیقت شناسی[۲]

حقیقت شناسی یعنی حالات ممکن دانشی که منطق در مورد یک حقیقت فرض می‌کند. در منطق گزاره‌ای و منطق مرتبه اول هر جمله یک حقیقت را بازنمایی می‌کند و عامل می‌تواند آن حقیقت را درست یا نادرست بداند یا نظری نداشته باشد. بنابراین، این دو منطق در مورد هر جمله سه حالت را در نظر می‌گیرند. سیستمی که از نظریه احتمال استفاده می‌کند درجه‌ای از اعتقاد را در نظر می‌گیرد که از صفر تا یک تغییر می‌کند. حقایق در منطق فازی درجه‌ای از درستی را نشان می‌دهند. درجه‌ای از اعتقاد در منطق احتمالی با درجه‌ای از درستی در منطق فازی متفاوت است.

---

[۱] Ontology

[۲] Epistemology

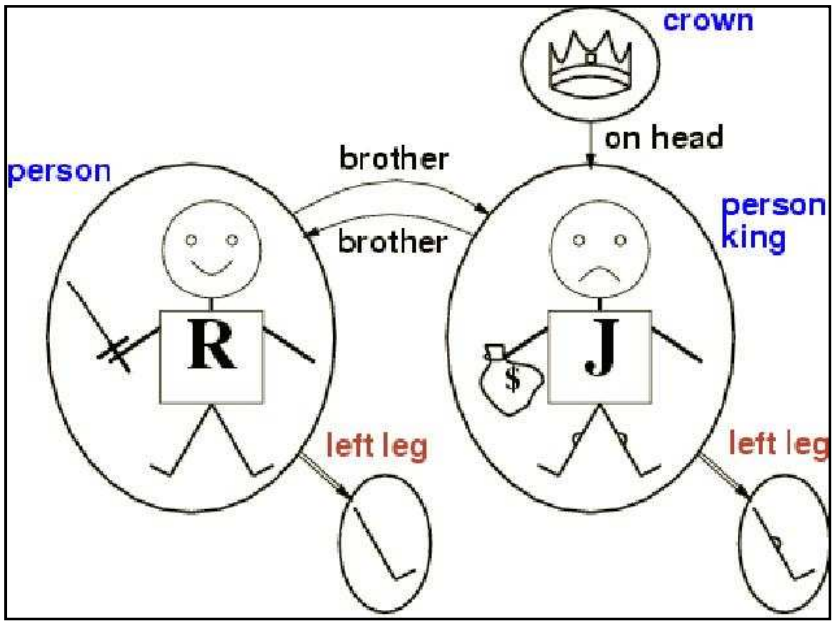
Language	Ontology	Epistemology
Propositional Logic	facts	true/false/unknown
First Order Logic	facts, objects, relations	true/false/unknown
Temporal Logic	facts, objects, relations, times	true/false/unknown
Probability theory	facts	degree of belief∈[0, 1]
Fuzzy logic	degree of truth ∈[0, 1]	known interval value

زبان	هستی شناسی (آنچه در جهان است)	حقیقت شناسی (اعتقادات عامل راجع به یک حقیقت)
منطق گزاره‌ها	حقایق	درست / نادرست / نامشخص
FOL	حقایق / اشیاء / روابط	درست / نادرست / نامشخص
منطق موقتی	حقایق / اشیاء / روابط / زمان	درست / نادرست / نامشخص
نظریه احتمال	حقایق	درجه‌ای از اعتقادات بین [۱و۰]
منطق فازی	حقایق با درجه درستی متعلق به [۱و۰]	در فاصله‌ای معین

جدول مربوط به زبان‌های رسمی و تعهدات هستی شناسی و حقیقت شناسی آنها

مدل‌های منطق مرتبه اول:

در منطق مرتبه اول، مدل‌ها دارای اشیاء هستند. در حقیقت، مدل‌ها شامل اشیاء و روابط میان آنها می‌باشند. دامنه یک مدل، مجموعه‌ای از اشیایی است که در آن مدل وجود دارند. این اشیاء عناصر دامنه نامیده می‌شوند. مدلی از منطق مرتبه اول را در شکل زیر مشاهده می‌کنید. مدل شکل زیر ۵ شیء دارد. اشیاء موجود در این مدل عبارتند از: ریچارد، جان، تاج، پای چپ جان، پای چپ ریچارد. اشیاء موجود در یک مدل ممکن است به شکل‌های مختلفی با یکدیگر ارتباط داشته باشند. رابطه مجموعه‌ای از چندتایی‌هایی از اشیاء مرتبط با هم است. نمونه‌هایی از روابط و توابع موجود در این مدل را در زیر مشاهده می‌کنید:



Brother: { <Richard, john> و <John, Richard>} (رابطه برادری)

On head :{<Crown, Khing John>} (رابطه بر سر بودن تاج)

رابطه‌های دودوئی در این مدل:

- Brother(John, Richard)
- Brother(Richard, john)
- On - head(Crown, Khing John)

رابطه‌های یکانی در این مدل:

- Person(John)
- Person(Richard)
- King(John)
- King(Richard)
- Be - crown (Crown)

توابع موجود در این مدل:

- Left_leg_of(John)
- Left_leg_of(Richard)
- Length(Left_leg_of(John))

نحو منطق مرتبه اول:

عناصر اصلی نحو منطق مرتبه اول، سمبل‌هایی است که نشان دهنده اشیاء، رابطه‌ها و توابع هستند. سمبل‌ها در منطق مرتبه اول به سه دسته تقسیم می‌شوند. این سمبل‌ها با حروف بزرگ شروع می‌شوند:

- **سمبل‌های ثابت:** برای اشاره به اشیاء به کار می‌روند. (مثل John, Richard)
- **سمبل‌های مسند:** رابطه‌ها را نشان می‌دهند. (مثل Brother, Person, King)

❖ **سمبل‌های تابع:** توابع را نشان می‌دهند. (مثل Left_leg_of)

نکته: در منطق مرتبه اول تعداد مدل‌ها و تعداد متغیرها ممکن است بی نهایت باشد. روش بررسی (شمارش) مدل، که در منطق گزاره‌ها به درستی کار می‌کند در منطق مرتبه اول قابل استفاده نیست زیرا امکان شمارش تمام مدل‌های ممکن وجود ندارد.

گرامر منطق مرتبه اول :

Sentence → AtomicSentence
| Sentence Connective Sentence
| Quantifier Variable, … Sentence
| ~ Sentence
| (Sentence)

AtomicSentence → Predicate(Term, …) | Term = Term

Term → Function(Term, …)
| Constant
| Variable

Connective → ⇒ | ∧ | ∨ | ⇔
Quantifier → ∀ | ∃
Constant → A | X_1 | John | …
Variable → a | x | …
Predicate → Before | HasColor | …
Function → MotherOf | LeftLegOf

:Term

هر ترم یک عبارت منطقی است که به یک شیء اشاره می‌کند. ترم‌ها سه نوع هستند:

❖ **ثابت‌ها:** (مثل A, B, John, …) که با حروف بزرگ شروع می‌شوند.

❖ **متغیرها:** (مثل a , x , b , …) که با حروف کوچک شروع می‌شوند.

❖ **توابع:** هر ترم پیچیده نیز شامل یک سمبل تابع و چند ترم در داخل پرانتز به عنوان آرگومان‌های آن تابع است. این نکته را به خاطر داشته باشید که خروجی یک تابع، یک شیء است ولی خروجی یک رابطه T یا F است. در زیر به چند نمونه از توابع اشاره شده است.

Brother(John), Length (Left_Leg_of(John)), Plus(2, 3)

جملات اتمیک :

با ترکیب ترم‌ها که به اشیاء اشاره می‌کنند و مسندها که به روابط اشاره می‌کنند، جملات اتمیک به دست می‌آیند. جملات اتمیک حقایقی را بیان می‌کنند که شامل مسند (فعل جمله) همراه با ترم‌هایی داخل پرانتز هستند و نتیجه آن T یا F است. در زیر به چند نمونه از جملات اتمیک اشاره شده است.

Brother(John, Richard)

Morried [Father(Richard) , Mother(John)]

Brother [John, Brother(John)]

Larg_than [Length(Left_Leg_Of(John)), Length(Left_Leg_Of(Richard))]

جملات پیچیده:

با ترکیب جملات اتمیک و روابط منطقی می‌توان جملات پیچیده تری ساخت. روابط منطقی عبارتند از:

$\wedge$, $\vee$, $\Rightarrow$, $\sim$, $\Leftrightarrow$. در زیر به چند نمونه از جملات پیچیده اشاره شده است.

Brother(John, Richard) $\wedge$ Brother(Richard, John)

King(Richard) $\vee$ king(John)

~ King(Richard) $\rightarrow$ King(John)

Older(John, 30) $\rightarrow$ ~Younger(John, 30)

~ Brother(John, peter)

سورها[1]:

در منطق مرتبه اول به جای شمارش اشیاء با استفاده از نام آن‌ها، می‌توان خواص همه یا برخی از اشیاء را بازنمایی کرد. این امر با استفاده از سورها امکان پذیر است. منطق مرتبه اول دارای دو سور استاندارد به نام‌های سور وجودی و سور عمومی می‌باشد. سورها کمک می‌کنند تا به جای شمارش اشیاء از طریق نام آن‌ها، خواص کلکسیونی از اشیاء را بیان کنیم.

سورعمومی[2]:

بیان قوانین کلی در منطق گزاره‌ها دشوار است ولی در منطق مرتبه اول با استفاده از سور عمومی‌این کار به آسانی امکان پذیر است.

سورهای عمومی مطابق شکل روبروتعریف می‌شوند: <جمله> <متغیر> $\forall$

مثال: $\forall$x, King (x) $\Rightarrow$ person(x) (همه پادشاهان شخص هستند)

$\forall$x, P(x): این جمله در یک مدل معادل جمله $P(k_1) \cap p(k_2) \cap \ldots p(k_n)$ است که kها عناصر دامنه مدل هستند و در آن p(x) یک عبارت منطقی است که بیان می‌کند p برای هر شیء x درست است.

King (John) $\Rightarrow$ Person (John)

King (Richard) $\Rightarrow$ Person (Richard)

King (Left_Leg_Of(John)) $\Rightarrow$ Person (Left_Leg_Of(John))

King (Left_Leg_Of(k)) $\Rightarrow$ Person (Left_Leg_Of(R))

King (Crown) $\Rightarrow$ Person (Crown)

---

[1] Quantifier

[2] Universal Quantifier

ارزش نهایی تمام جملات فوق T (صحیح) است. (با استفاده از روش انتفاع مقدم یعنی اگر مقدم نادرست باشد، بدون توجه به طرف دوم، طرف دوم درست است، درستی جملات دوم تا چهارم مشخص می‌شود.)

سور وجودی[1]:

یک جمله با سور عمومی حقیقتی را در مورد همه اشیاء بیان می‌کند در حالی که سور وجودی حقیقتی را در مورد بعضی از اشیاء بیان می‌کند.

سورهای وجودی مطابق شکل روبروتعریف می شوند: $\exists$ <متغیر> <جمله>

مثال: $\exists x$, Crown(x) $\wedge$ On Head(x, john)

$\exists x, p(x)$: به این معناست که p برای حداقل یک شیء درست باشد. به عبارت دیگر، این جمله، معادل جمله فصلی $p(k_1) \vee p(k_2) \vee \ldots p(k_n)$ است.

خصوصیات سورها:

به نظر می‌رسد در استفاده از سور عمومی، رابط مورد استفاده ترکیب شرطی و در استفاده از سور وجودی، رابط مورد استفاده ترکیب عطفی مناسب باشد. بکارگیری ترکیب عطفی به عنوان رابط اصلی همراه با سور عمومی منجر به ادعایی بسیار قوی و بکارگیری ترکیب شرطی به عنوان رابط اصلی همراه سور وجودی منجر به ادعایی بسیار ضعیف می‌شود. به مثال‌های زیر توجه کنید:

- هر کسی پادشاه است آنگاه یک شخص است: $\forall x$, King(x) $\Rightarrow$ Person(x)
- (ادعایی قوی) همه افراد پادشاه هستند: $\forall x$, King(x) $\wedge$ Person(x)
- وجود دارد شخصی که پادشاه است: $\exists x$, King(x) $\wedge$ Person(x)
- (ادعایی ضعیف) اگر وجود داشته باشد پادشاهی، آنگاه شخص خواهد بود: $\exists x$, King(x) $\Rightarrow$ Person(x)

یک اشتباه متداول:

سور عمومی اغلب با ترکیب شرطی و سور وجودی اغلب با ترکیب عطفی بکار می رود

$$\forall x\ Human(x) \Rightarrow Mortal(x)$$

$$\exists x\ Sister(x, Spot) \wedge Cat(x)$$

مثال : جمله زیر به معنای " هر کسی در کلاس است و هر کسی باهوش است " می باشد

$$\forall x\ at(x, Class) \wedge Smart(x)$$

ویا جمله زیر:

$$\exists x\ at(x, Class) \Rightarrow Smart(x)$$

[1] Existential Quantifier

مثال: جملات زیر را با توجه به نکته‌ای که در بالا به آن اشاره شد در فرم منطق مرتبه اول بازنمایی کنید. (فرض کنید که $p(x)$ معرف سیاستمدار بودن x و $q(x)$ معرف نادرست بودن x باشد.)

- برخی سیاستمداران نادرست هستند: $\exists x, p(x) \wedge q(x)$
- هیچ سیاستمداری نادرست نیست: $\forall x, p(x) \Rightarrow \sim q(x)$
- تمامی سیاستمداران نادرست نیستند: $\exists x, p(x) \wedge \sim q(x)$
- تمامی سیاستمداران نادرست هستند: $\forall x, p(x) \Rightarrow q(x)$

سورهای تو در تو[1]:

با استفاده از چند سور می‌توانیم جملات پیچیده‌تری بسازیم. در ساده ترین حالت، سورها از یک نوع هستند. به مثال-های زیر توجه کنید.

$\forall x\ \forall y\ Brother(x, y) \Rightarrow Sibling(x, y)$

$\forall x, y\ Sibling(x, y) \leftrightarrow Sibling(y, x)$

(Sibling: همزاد بودن)

در دو جمله فوق، چون سورها از یک نوع هستند جابجایی سورها تاثیری در معنای جمله ندارد. ولی اگر سورهای تودرتو بایکدیگر متفاوت باشند معنای جمله با جابجایی سورها تحت تاثیر قرار خواهد گرفت.

- $\forall x\ \exists y \neq \exists y\ \forall x$
- $\exists x\ \exists y = \exists y\ \exists x$
- $\forall x\ \forall y \equiv \forall y\ \forall x$

مثال: حداقل یک شخص وجود دارد که همه اشخاص را دوست دارد: $\exists x\ \forall y\ Loves(x, y)$

همه افراد، حداقل یک نفر را دوست دارند: $\forall y\ \exists x\ Loves(x, y)$

به مثال دیگری در زمینه ریاضیات توجه فرمائید.

$\forall x \in \mathbb{N}, \exists y \in \mathbb{Z}, x + y = 0$ T

$\exists y \in \mathbb{Z}, \forall x \in \mathbb{N}, x + y = 0$ F

ارتباط بین سورها: سور عمومی و سور وجودی از طریق نقیض با یکدیگر ارتباط دارند.

$\sim(\forall x\ p) \equiv \exists x, \sim p$ $\qquad$ $\sim(\exists x\ p) \equiv \forall x, \sim p$

$\forall x\ p \equiv \sim\exists x, \sim p$ $\qquad$ $\exists x\ p \equiv \sim\forall x, \sim p$

به مثال‌های زیر در این زمینه توجه فرمائید:

- همه افراد، بستنی را دوست دارند: $\forall x\ Likes(x, Icecream)$
- وجود ندارد کسی که بستنی دوست نداشته باشد : $\sim\exists x, \sim Likes(x, Icecream)$
- افرادی وجود دارند که گل کلم دوست دارند: $\exists x\ Likes(x, Broccoli)$
- هیچ کس نیست که گل کلم دوست نداشته باشد: $\sim\forall x \sim Likes(x, Broccoli)$

---

[1] nested quantifiers

مفهوم تساوی[۱]:

در منطق مرتبه اول به غیر از به کارگیری مسندها برای ساختن جملات اتمی، راه دیگری نیز وجود دارد. ما می‌توانیم از علامت تساوی (=) برای ساختن جملاتی که در آن دو ترم به شیء یکسانی اشاره می‌کنند، استفاده کنیم. به عنوان مثال در جمله Father(John) = peter، شیءای که Father(John) به آن اشاره می‌کند با شیءای که peter به آن اشاره می‌کند یکسان است.

همچنین، از علامت =، می‌توان همراه با سمبل نقیض برای تاکید بر مساوی نبودن دو شیء استفاده کرد.

$$\exists x,y\ Brother(x, Richard) \wedge Brother(y, Richard) \wedge \sim(x = y)$$

" ریچارد حداقل دو برادر دارد "

ادعاها و پرسش‌ها در منطق مرتبه اول:

جملات با استفاده از Tell به پایگاه دانش اضافه می‌شوند. چنین جملاتی را **ادعا**[۲] می‌گوییم.

- Tell (KB, King(John))
- Tell (KB, $\forall x$ King(x) $\Rightarrow$ Person(x))

با استفاده از Ask سوالاتی از پایگاه دانش در مورد درستی جملات پرسیده می‌شود. سوالاتی که توسط Ask از پایگاه دانش پرسیده می‌شود، **درخواست**[۳] یا **هدف**[۴] نام دارد.

- Ask(KB, King(Richard))
- Ask(KB, Person(John))

دامنه ی خویشاوندی:

i. **اشیاء:** افراد

ii. **رابطه یکانی:** Male, Female

iii. **رابطه دودوئی:** Parent, Husband, Married, Brother

iv. **تابع:** Father, mother

به جملات زیر در دامنه خویشاوندی توجه فرمائید:

$$\forall\ m,c\ Mother(c) \Leftrightarrow Female(m) \wedge Parent(m, c)$$

مادر هر شخصی، والد مونث آن شخص است.

$$\forall\ w,h\ Husband(h, w) \Leftrightarrow Male(h) \wedge Spouse(h,w)$$

شوهر هر فرد، همسر مذکر آن فرد است.

$$\forall x\ Male(x) \Leftrightarrow \sim Female(x)$$

---

[۱] equality

[۲] assertion

[۳] Query

[۴] Goal

مذکر و مونث بودن مولفه‌های متضادند.

$$\forall\ p,c\ Parent(\ p,\ c\ ) \Leftrightarrow Child(\ c,\ p\ )$$

والدین و فرزند رابطه معکوس دارند.

$$\forall\ g,c\ Grandparent(\ g,\ c\ ) \Leftrightarrow \exists p\ parent(\ g,\ p\ ) \wedge parent(\ p,\ c\ )$$

مادربزرگ یا پدربزرگ هر فرد، والدِ والد آن فرد است.

$$\forall\ x,y\ Sibling(\ x,\ y\ ) \Leftrightarrow \exists p\ parent(\ p,\ x\ ) \wedge parent(\ p,\ y\ ) \wedge (\ x \neq y\ )$$

همزاد[1] یک فرد، فرزند دیگر والدین آن فرد است.

دامنه مجموعه‌ها:

i. **مسند دودوئی:** $x \in S$, $S_1 \subseteq S_2$

ii. **مسند یکانی:** $Set(s)$ (یک موجودیت را می‌گیرد و می‌گوید مجموعه هست یا نه؟)

iii. **توابع دودوئی:** $S_1 \cup S_2$، $S_1 \cap S_2$

iv. **ثابت:** $\{\ \}$ (مجموعه تهی)

v. $\{x \mid s\}$**:** مجموعه حاصل از افزودن x به s

به جملات زیر در دامنه مجموعه‌ها توجه فرمائید:

$$\forall\ S_1,S_2\quad S_1 \subseteq S_2 \Leftrightarrow (\ \forall x\quad x \in S_1 \rightarrow x \in S_2\ )$$

یک مجموعه، زیرمجموعه دیگری است اگر و فقط اگر تمام اعضای مجموعه اول عضو مجموعه دوم هم باشند.

$$\forall\ S_1,S_2\quad (\ S_1 = S_2\ ) \Leftrightarrow (\ S_1 \subseteq S_2 \wedge S_2 \subseteq S_1\ )$$

دو مجموعه مساوی هستند اگر و فقط اگر هر کدام زیر مجموعه دیگری باشد.

$$\forall\ x,S_1,S_2\quad x \in (\ S_1 \cap S_2\ ) \Leftrightarrow (\ x \in S_1 \wedge x \in S_2\ )$$

یک شی عضو اشتراک دو مجموعه است اگر و فقط اگر عضو هر دو مجموعه باشد.

$$\forall\ x,S_1,S_2\quad x \in (\ S_1 \cup S_2) \Leftrightarrow (\ x \in S_1 \vee x \in S_2\ )$$

یک شی عضو اجتماع دو مجموعه است اگر و فقط اگر عضو یکی از دو مجموعه باشد.

$$\forall\ S,Set(\ s\ ) \Leftrightarrow (\ S = \{\}\ ) \vee (\ \exists\ x,S_2\quad Set(s_2) \wedge (s= \{x \mid s_2\}\ )\ )$$

مجموعه‌ها یا یک مجوعه تهی است یا با اضافه کردن یک عنصر به یک مجموعه دیگر ساخته می‌شوند.

$$\forall\ x,S\quad x \in S \Leftrightarrow S = \{\ x \mid S\ \}$$

افزودن یک عنصر که قبلاً در مجموعه وجود داشته باشد، تاثیری ندارد.

---

[1] Sibling

مهندسی دانش[1]:

به فرآیند کلی ساخت پایگاه دانش، فرآیند مهندسی دانش گفته می‌شود. یک مهندس دانش کسی است که حوزه یا دامنه خاصی را بررسی می‌کند و یاد می‌گیرد چه مفاهیمی در آن دامنه مهم هستند. یک فرآیند مهندسی شامل مراحل زیر است:

I. مشخص نمودن وظیفه

مهندس دانش باید بازه‌ای از پرسش‌ها را که برای پایگاه دانش قابل دسترس است توصیف کند و انواع حقایقی که برای یک مسئله خاص وجود دارند را تعیین نماید.

II. جمع آوری دانش مربوطه

مهندس دانش ممکن است در دامنه یا حوزه‌ای خاص خبره باشد یا از افراد خبره برای استخراج اطلاعات استفاده کند. این فرایند کسب دانش[2] نامیده می‌شود. در این مرحله دانش به طور رسمی بازنمایی نمی‌شود و ایده این مرحله درک حوزه پایگاه دانش و درک اینکه دامنه چگونه کار می‌کند است.

III. تصمیم گیری در مورد واژه نامه مسندها، توابع و ثابت‌ها

یعنی ترجمه مفاهیم مهم در سطح دامنه به اسامی در سطح منطق. به عنوان مثال آیا چاله‌ها باید به وسیله اشیا یا به وسیله مسندهای یکانی بازنمایی شوند

IV. کدگذاری دانش عمومی در مورد دامنه

مهندس دانش باید اصول مربوط به تمام ترم‌های واژه نامه را بنویسد. بدین ترتیب معنای ترم‌ها مشخص خواهند شد.

V. کدگذاری توصیف یک نمونه مسئله خاص

این مرحله شامل نوشتن جملات اتمی در مورد نمونه‌هایی از مفاهیمی است که به عنوان بخشی از هستی شناسی هستند.

VI. اعمال پرسش‌ها به رویه استنتاج و دریافت پاسخ

VII. اشکال زدایی پایگاه دانش

اگر یک اصل در یک پایگاه دانش نباشد آنگاه جواب برخی از پردازش‌ها وجود نخواهد داشت که باید اشکال زدایی صورت گیرد. فقدان اصول و یا وجود اصول ضعیف را می‌توان با توجه به مکان‌هایی که زنجیره استدلال به طور غیر منتظره متوقف می‌شود مشخص کرد.

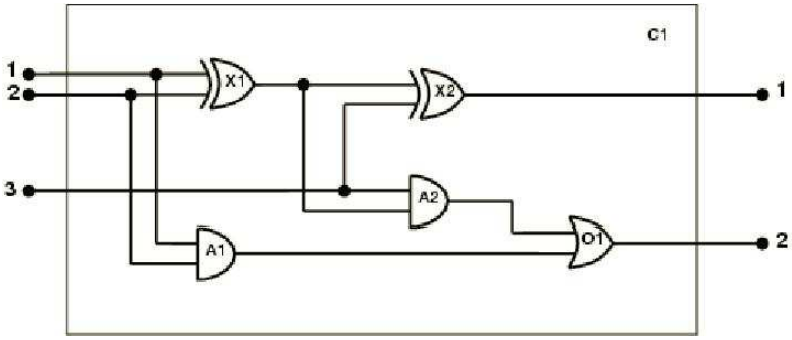
برای درک بهتر مراحل فوق، این هفت مرحله را در یک مثال (حوزه مدارهای الکترونیکی) بسط می‌دهیم.

---

[1] Knowledge Engineering

[2] Knowledge acquisition

دامنه مدارهای الکتریکی



۱. مشخص نمودن وظیفه

- آیا مدار واقعا به درستی جمع می کند؟ (وارسی مدار)

۲. جمع آوری دانش مربوطه

- ترکیب سیم ها و گیت ها؛ انواع گیت ها (AND, OR, XOR, NOT)
- دانش نامربوط: اندازه، شکل، رنگ، قیمت گیت ها

۳. تصمیم گیری در مورد لغات

- راه های مختلف:

$Type(X_1) = XOR$

$Type(X_1, XOR)$
$XOR(X_1)$

۴. کد نمودن دانش عمومی دامنه

$-\forall t_1,t_2\ Connected(t_1, t_2) \Rightarrow Signal(t_1) = Signal(t_2)$
$-\forall t\ Signal(t) = 1 \vee Signal(t) = 0$

$-1 \neq 0$

$-\forall t_1,t_2\ Connected(t_1, t_2) \Rightarrow Connected(t_2, t_1)$

$-\forall g\ Type(g) = OR \Rightarrow Signal(Out(1,g)) = 1 \Leftrightarrow \exists n\ Signal(In(n,g)) = 1$

$-\forall g\ Type(g) = AND \Rightarrow Signal(Out(1,g)) = 0 \Leftrightarrow \exists n\ Signal(In(n,g)) = 0$

$-\forall g\ Type(g) = XOR \Rightarrow Signal(Out(1,g)) = 1 \Leftrightarrow Signal(In(1,g)) \neq Signal(In(2,g))$

۵. کد نمودن دانش مربوط به یک نمونه مساله خاص

$Type(X_1) = XOR$	$Type(X_2) = XOR$
$Type(A_1) = AND$	$Type(A_2) = AND$
$Type(O_1) = OR$	

$Connected(Out(1,X_1),In(1,X_2))$	$Connected(In(1,C_1),In(1,X_1))$
$Connected(Out(1,X_1),In(2,A_2))$	$Connected(In(1,C_1),In(1,A_1))$
$Connected(Out(1,A_2),In(1,O_1))$	$Connected(In(2,C_1),In(2,X_1))$
$Connected(Out(1,A_1),In(2,O_1))$	$Connected(In(2,C_1),In(2,A_1))$
$Connected(Out(1,X_2),Out(1,C_1))$	$Connected(In(3,C_1),In(2,X_2))$
$Connected(Out(1,O_1),Out(2,C_1))$	$Connected(In(3,C_1),In(1,A_2))$

۶. اعمال پرس و جو بر رویه استنتاج

$$\exists i_1,i_2,i_3,o_1,\ o_2\ \ Signal(In(1,C_1)) = i_1 \wedge Signal(In(2,C_1)) = i_2 \wedge Signal(In(3,C_1)) = i_3 \wedge Signal(Out(1,C_1)) = o^1 \wedge Signal(Out(2,C_1)) = o_2$$

پاسخ: جدول ورودی/خروجی کامل که می تواند برای بررسی مدار استفاده شود.

۶. اعمال پرس و جو بر رویه استنتاج

چه ترکیبی از ورودی ها باعث می شوک که اولین خروجی مدار C_1 (بیت جمع) به صفر و خروجی دوم مدار C_1 (بیت نقلی) به یک تبدیل شود؟

$$\exists i_1,i_2,i_3\ Signal(In(1,C_1)) = i_1 \wedge Signal(In(2,C_1)) = i_2 \wedge Signal(In(3,C_1)) = i_3 \wedge Signal(Out(1,C_1)) = 0 \wedge Signal(Out(2,C_1)) = 1$$

پاسخ های ممکن:

$\{i_1/1,\ i_2/1,\ i_3/0\}$,
$\{i_1/1,\ i_2/0,\ i_3/1\}$,
$\{i_1/0,\ i_2/1,\ i_3/1\}$

سوالات تستی آخر فصل

ترم اول ۸۷–۸۸

۱. با این فرض که متغیر X در Q به صورت آزاد (free) ظاهر نشده است مقدار کدام یک از عبارات زیر در منطق (PREDICATE LOGIC) مسندات نادرست است؟

الف. $(\exists x(P(x) \to Q)) \to (\forall x P(x) \to Q)$

ب. $(\forall x P(x) \to Q) \to (\forall x(P(x) \to Q)$

ج. $(\exists x P(x) \to Q) \to (\forall x P(x) \to Q)$

د. $(\exists x P(x) \to Q) \to (\exists x(P(x) \to Q))$

۲. ترجمه جمله " هرکس یک و فقط یک مادر دارد." به منطق مرتبه اول چیست؟

الف. $\forall x, \exists y mother(x, y) \wedge (\forall z mother)(m, z) \Rightarrow y = z)$

ب. $\forall x, \exists y mother(x, y) \wedge (\forall z \neg mother)(m, z)$

ج. $\forall x, y(mother(x, y) \Rightarrow \neg(\exists z mother(m, z))$

د. $\forall x, y, z(mother(m, y) \wedge mother(x, z)) \Rightarrow y = z$

ترم دوم ۸۷–۸۸

۳. ایراد کدامیک از منطق‌های زیر اینست که نمی‌تواند راجع به یک سری اشیاء، کلی صحبت کند و باید برای هر کدام از اشیاء یکی یکی، قانون ایجاد کند؟

الف. منطق مرتبه دوم ب. منطق مرتبه اول ج. منطق گزاره ها د. منطق فازی

۴. حاصل تبدیل جمله "بعضی دانش آموزان برای هیچ کس نامه نمی‌نویسند مگر دانش آموزانی که دوستشان دارند."به منطبق مرتبه اول چیست؟

الف. $\exists x \forall y student(x) \wedge ((student(y) \wedge Like(x, y)) \Leftrightarrow WletterFor(x, y)$

ب. $\forall x \exists y student(x) \wedge ((student(y) \wedge Like(x, y)) \Rightarrow WletterFor(x, y)$

ج. $\exists x student(x) \wedge (\exists y student(y) \wedge WletterFor(x, y)) \Rightarrow Like(x, y)$

د. $\exists x student(x) \Rightarrow \forall y student(y) \wedge Like(x, y)) \Leftrightarrow WletterFor(x, y)$

ترم دوم ۸۸–۸۹

۵. معنی جمله $\neg\exists x \neg Likes(x, Icecream)$ در زبان طبیعی کدام است؟

الف. هیچ کس بستنی دوست ندارد.

ب. همه بستنی دوست دارند.

ج. هیچ کس وجود ندارد که بستنی دوست داشته باشد.

د. کسی وجود دارد که بستنی دوست ندارد.

۶. بازنمایی جمله " پادشاه جان تاجی روی سرش داشت."در منطق مرتبه اول کدام گزینه است؟

الف. $\exists x Crown(x) \wedge OnHead(x, john)$

ب. $\exists x Crown(x) \Rightarrow OnHead(x, john)$

ج. $\forall x Crown(x) \wedge OnHead(x, john)$

د. $\forall x Crown(x) \vee OnHead(x, john)$

ترم دوم ۸۹–۹۰

۷. در مورد پرس و جوی $ASK(KB, \exists x Person(x))$ کدام مورد درست است؟

۱) در صورت وجود یک جایگزین پاسخ آن TRUE است

۲) یک جایگزین به شکل مثلاً {X / JOHN} را در صورت وجود بر می‌گرداند.

۳) فهرستی از جایگزین‌های موجود را بر می‌گرداند.

۴) در صورت عدم وجود جایگزین پاسخ آن False است

الف. ۱و۴ ب. ۲

ج. ۳ د. ۱و ۴ میتواند درست باشد اما پاسخ استاندارد گزاره ۲ یا ۳ می‌باشد.(وابسته به حالت)

تابستان ۹۰

۸. گزاره ~∃x ~Like(x, IceCream) معادل است با:

الف. ∃x Like(~x, IceCream) ب. ∀x ~Like(x, IceCream)

ج. ∃x ~Like(x, IceCream) د. ∀x Like(x, IceCream)

۹. در کدام مرحله از فرآیند مهندسی دانش، محدوده سوالاتی که پایگاه دانش به آنها پاسخگو است، مشخص می-گردد؟

الف. گردآوری دانش مرتبط ب. طرح پرس و جو از رویه استنتاج و گرفتن پاسخ‌ها

ج. شناسایی وظیفه د. رمز کردن دانش کلی دامنه

۱۰. با توجه به پایگاه دانش زیر، مقدار ASK(KB, Person(Ali)) چیست؟

TELL(KB, King(John)) TELL(KB, ∀x King(x)) → person(x)

الف. Ali ب. True ج. False د. John

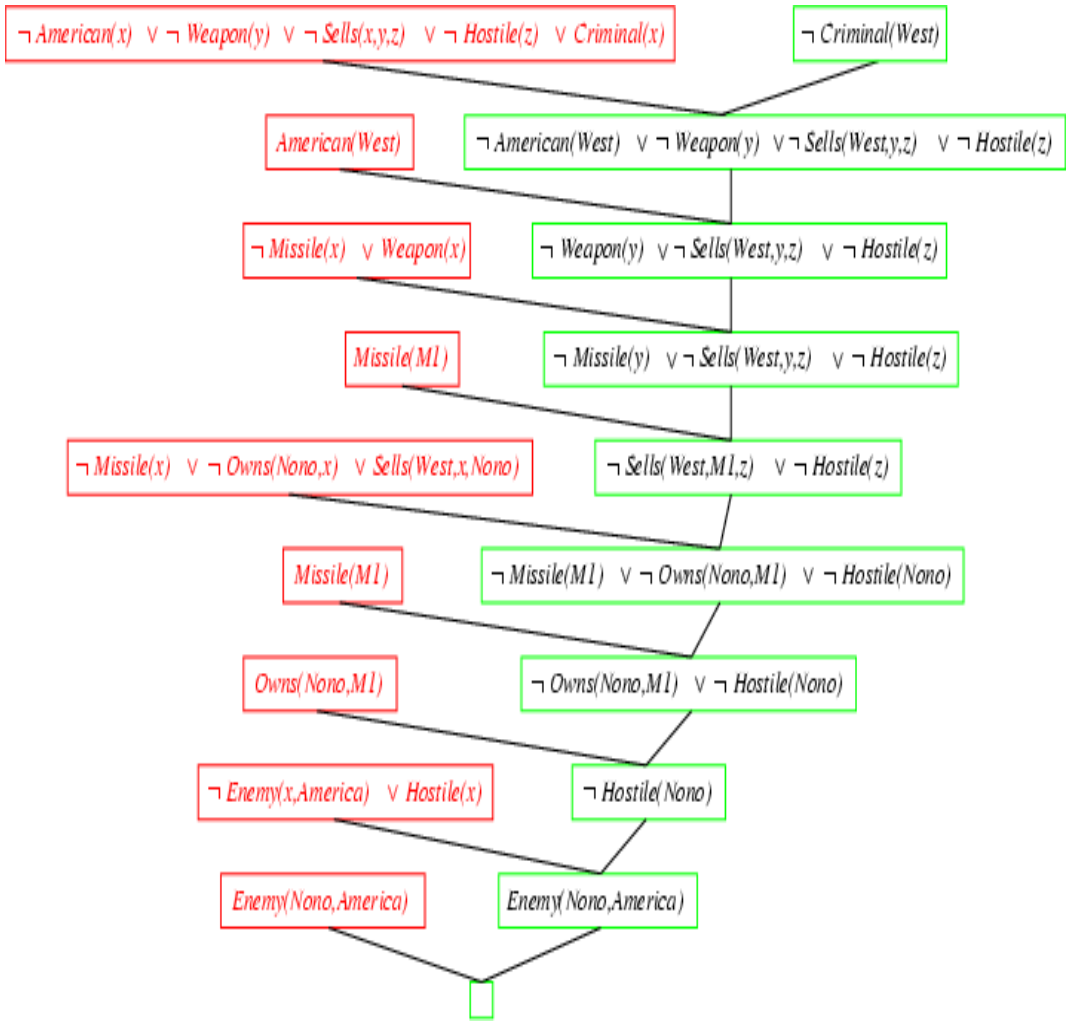
پاسخ نامه تستی

سوال	گزینه صحیح
۱	ب
۲	الف
۳	ج
۴	الف
۵	ب
۶	الف
۷	د
۸	د
۹	ج
۱۰	ج

فصل نهم: استدلال در منطق مرتبه اول

آنچه در این فصل خواهید آموخت:

- ❖ تقلیل استنتاج مرتبه اول به استنتاج گزاره‌ای
- ❖ یکسان سازی (Unification)
- ❖ قانون انتزاع تعمیم یافته (GMP)
- ❖ زنجیره استنتاج رو به جلو
- ❖ زنجیره استنتاج رو به عقب
- ❖ Resolution



مقدمه:

در این فصل ایده‌های مربوط به سیستم‌های استنتاجی منطقی مدرن را معرفی می‌کنیم. از قوانین استنتاج ساده شروع می‌کنیم که این قوانین می‌توانند به جملاتی با انواع سورها اعمال شوند و آنها را به شکل جملات بدون سور تبدیل کنند. این قوانین منجر به این ایده می‌شود که استنتاج مرتبه اول می‌تواند با تبدیل پایگاه دانش به جملات منطق گزاره‌ای و استفاده از منطق گزاره‌ای انجام شود. (تقلیل به استنتاج گزاره‌ای) معمولاً استدلال با جملات مرتبه‌ی اول با استفاده از Resolution راه حل ناکارآمدتری نسبت به استدلال با عبارات متناهی با استفاده از رو به جلو و زنجیره روبه عقب می‌باشد.

قانون حذف سور عمومی:

این قانون می‌گوید که ما می‌توانیم هر جمله حاصل از جایگزینی متغیر با یک ترم زمینی را استنتاج کنیم. برای نوشتن این قانون استنتاج به تابع جانشینی نیاز داریم. مقدار تابع subst(θ, α) برابر با نتیجه جانشینی θدرجمله α است. لذا قانون حذف سور عمومی به صورت زیر می‌باشد: به ازای هر متغیر v و جمله‌ی α، در جمله‌ی α به جای متغیرv، term زمینی g را جایگزین کن.

نمونه سازی عمومی Universal instantiation

هر نمونه از یک جمله دارای سور عمومی، توسط آن جمله قابل استلزام است.

$$\frac{\forall v \alpha}{subset(\left\{\frac{v}{g}\right\}, \alpha)}$$

مثال: $\forall x\ King(x) \wedge Greedy(x) \Longrightarrow Evil(x)$

$King(John) \wedge Greedy(John) \Longrightarrow Evil(John)$
$King(Richard) \wedge Greedy(Richard) \Longrightarrow Evil(Richard)$
$King(Father(John)) \wedge Greedy(Father(John)) \Longrightarrow Evil(Father(John))$

:Subst({v/g}, α)

اگر در جمله‌ای که به فرم منطق مرتبه‌ی اول است برای متغیرهای آن جمله، جایگزینی پیدا کنیم به این عمل جایگزینی یا Substition گفته می‌شود که در آن v یک متغیر و α جمله‌ای به فرم منطق مرتبه اول و g یک ترم زمینی[1] می‌باشد.

قانون حذف سور وجودی:

قانون حذف سور وجودی کمی پیچیده تر است. برای هر جمله α و متغیر v و سمبل k که تاکنون در پایگاه دانش ظاهر نشده داریم: به ازای برخی متغیر v و جمله‌ی α، در جمله‌ی α به جای متغیر v، ثابت k را که تاکنون در پایگاه دانش ظاهر نشده جایگزین کن.

[1] ground term

نمونه سازی وجودی Existential instantiation

برای هر جمله α، متغیر v و سیمبول ثابت k که در جای دیگری از پایگاه دانش ظاهر نشده باشد:

$$\frac{\exists v \alpha}{subst(\{\frac{v}{k}\}, \alpha)}$$

مثال: $\exists x\ Crown(x) \wedge Onhead(x, John)$

$$Crown(C_1) \wedge Onhead(C_1, John)$$

نکته: به شرطی که c1 یک سیمبول جدید باشد، به آن ثابت اسکولم می‌گویند.

جمله با سور وجودی می‌گوید که یک یا چند شیء وجود دارد که شرطی را برآورده می‌کند و فرآیند نمونه سازی (حذف) فقط نامی را برای آن اشیاء انتخاب می‌کند که این نام جدید یک ثابت اسکولم نامیده می‌شود. حذف سور وجودی حالت خاصی از یک فرایند کلی به نام skolemization است.

نکته: نمونه سازی یا حذف سورعمومی می‌تواند چندین بار اعمال شود تا نتایج محتلفی تولید کند ولی نمونه سازی یا حذف سور وجودی فقط یک بار می‌تواند اعمال شود و سپس جمله دارای سور وجودی حذف می‌گردد.

تقلیل به استنتاج گزاره‌ای:

وقتی قوانین برای استنتاج جملات غیر سوری از جملات سوری در اختیار داریم می‌توانیم استنتاج مرتبه اول را به استنتاج گزاره‌ای تقلیل دهیم. ایده این است که سور وجودی می‌تواند با یک بار نمونه سازی جایگزین شود و یک سور عمومی می‌تواند توسط مجموعه‌ای از تمام نمونه سازهای ممکن جایگزین شود. به مثال زیر توجه کنید:

فرض کنید KB فقط شامل جملات زیر باشد

$\forall x$ King(x) $\wedge$ Greedy(x) $\Rightarrow$ Evil(x)

King(John)

Greedy(John)

Brother(Richard, John)

با نمونه سازی جمله دارای سور عمومی به تمام طرق ممکن، داریم:

King(John) $\wedge$ Greedy(John) $\Rightarrow$ Evil(John)

King(Richard) $\wedge$ Greedy(Richard) $\Rightarrow$ Evil(Richard)

King(John)

Greedy(John)

Brother(Richard, John)

KB جدید به فرم گزاره‌ای درآمده است (گزاره‌ای سازی). سمبول‌های گزاره‌ای عبارتند از:

King(John), Greedy(John), Evil(John), King(Richard), etc.

این تکنیک گزاره سازی نام دارد. بنابراین هر پایگاه دانش در منطق مرتبه اول[۱] می‌تواند به گونه‌ای گزاره سازی شود که استلزام را حفظ کند، یعنی یک جمله توسط KB جدید قابل استلزام است اگر و فقط اگر توسط KB اصلی قابل استلزام باشد. بنابراین برای تقلیل به استنتاج گزاره‌ای کافیست پایگاه دانش و پرسش را گزاره سازی کنیم و با استفاده از الگوریتم‌های استنتاج منطق گزاره‌ای مانند Resolution و زنجیره رو به جلو یا عقب نتیجه را برگردانیم. یک مسئله در این تکنیک گزاره سازی رخ می‌دهد و آن این است که وقتی KB شامل سمبل‌های تابع باشد مجموعه جانشین‌های ترم زمینی نامتناهی است. به عنوان مثال اگر پایگاه دانش دارای تابع Father باشد، ممکن است جملات تو در توی نامتناهی مثل Father(Father(Father(John))) ساخته شود. الگوریتم‌های گزاره‌ای ما با مجموعه‌های بزرگ و نامتناهی جملات دچار مشکل می‌شوند. برای رفع این مشکل از تئوری Herbrond استفاده می‌کنیم.

تئوری Herbrond :

اگر جمله α توسط پایگاه دانش منطق مرتبه اول قابل استلزام باشد آنگاه این جمله توسط زیر مجموعه‌ی محدودی از پایگاه دانش گزاره‌ای سازی شده قابل استلزام است.

تئوری چرچ و تورینگ:

استلزام در منطق مرتبه اول نیمه تصمیم پذیر است. یعنی الگوریتم‌هایی وجود دارند که جملات استنتاج شده را اثبات می‌کنند اما الگوریتم‌هایی وجود ندارند که در مورد جملات غیر استنتاج شده با قطعیت اعلام کنند که آنها واقعاً اثبات نمی‌شوند.

نکته: یکی دیگر از مشکلات گزاره‌ای سازی نمودن این است که جملات نامربوط زیادی تولید می‌گردد، به مثال زیر توجه کنید:

$\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$

$King(John)$

$\forall y\ Greedy(y)$

$Brother(Richard, John)$

بدیهی است که Evil(John) اما گزاره‌ای سازی کردن حقایق زیادی مانند Greedy(Richard) تولید می-کند که نامربوط می‌باشند

قانون انتزاع تعمیم یافته[۲] GMP :

به ازای جملات p_i, p'_i, q یک مجموعه جانشینی θ که $subst(\theta, p_i) = subst(\theta, p'_i)$. برای هر i رابطه‌ی زیر برقرار است:

---

[۱] First Order Logical

[۲] General Modus Ponens

$$\frac{p_1', \quad p_2', \quad \ldots, \quad p_n', \quad (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)}{q\theta} \qquad \text{where } p_i'\theta = p_i\theta \text{ for all } i$$

p_1' is $King(John)$ $\quad$ p_1 is $King(x)$
p_2' is $Greedy(y)$ $\quad$ p_2 is $Greedy(x)$
θ is $\{x/John, y/John\}$ $\quad$ q is $Evil(x)$
$q\theta$ is $Evil(John)$

- GMP با پایگاه دانشی از clauseهای معین (دقیقاً یک لیترال مثبت) کار می‌کند
- فرض می‌شود که تمام متغیرها دارای سور عمومی هستند.

به راحتی می‌توان نشان داد که GMP یک قانون استنتاج صحیح است. GMP شکل ارتقاء یافته قانون انتزاع است. زیرا قانون MP در منطق گزاره‌ای به منطق مرتبه اول ارتقاء می‌یابد. امتیاز مهم قانون استنتاج GMP نسبت به گزاره سازی این است که این قوانین فقط جانشین‌هایی را که لازم است انجام می‌دهند. GMP با پایگاه دانشی از فراکردهای معین یعنی عباراتی که دقیقا یک لیترال مثبت دارند کار می‌کند.

یکسان سازی[1]:

به عمل جایگزینی متغیرها به جای متغیرها، توابع به جای متغیرها و ثابت‌ها به جای متغیرها بطوریکه گزاره‌های یکسان تولید شود، یکسان سازی گفته می‌شود. به عبارت دیگر داریم: اگر

- Unify(α,β) = θ if αθ = βθ

p	q	θ
Knows(John,x)	Knows(John,Jane)	{x/Jane}}
Knows(John,x)	Knows(y,OJ)	{x/OJ,y/John}}
Knows(John,x)	Knows(y,Mother(y))	{y/John,x/Mother(John)}}
Knows(John,x)	Knows(x,OJ)	{fail}

استاندارد سازی متغیرها: مثلاً Knows(z, OJ)

قوانین استنتاج ارتقا یافته به یافتن جانشین‌هایی نیاز دارند که باعث می‌شوند عبارات منطقی مختلف یکسان به نظر برسند. این فرآیند یکسان سازی نام دارد و جزء کلیدی الگوریتم‌های استنتاج مرتبه اول می‌باشد. الگوریتم unify دو جمله را دریافت کرده و یک یکسان ساز θ را در صورت وجود بر می‌گرداند. مشکل Fail هنگامی اتفاق می‌افتد که دو جمله از متغیر یکسانی استفاده کنند که با استاندارد سازی یعنی تغییر نام متغیر در یکی از جملات مشکل Fail برطرف خواهد شد.

[1] Unification

نکته: برای یکسازی سازی Knows(John,x) و Knows(y,z) دو جانشینی وجود دارد یا

- ♠ $\theta = \{y / John, x / z\}$
- ♠ $\theta = \{y / John, x / John, z / John\}$

- ❖ اولین یکسان ساز عمومی تر از دومی می‌باشد
- ❖ فقط یک عمومی ترین یکسان ساز (MGU) most general unifier وجود دارد که نسبت به تغییر نام متغیرها منحصر به فرد می‌باشد.

$$MGU = \{ y/John, x/z \}$$

نکته: تعداد یکسان سازها برای دو جمله می‌تواند صفر، یک یا بیش از یکی باشد.

نکات تکمیلی در مورد یکسان سازی:

i. مسندهای غیر همنام قابل یکسان سازی نخواهند بود. $f(x, y) \neq g(A, B)$

ii. دو ثابت غیر همنام قابل یکسان سازی نیستند. $f(A, x) \neq f(B, x)$

در یکسان سازی روی زوج عبارات شامل یک متغیر با همان متغیر یکسان سازی نمی‌تواند انجام بگیرد. به عنوان مثال در عبارت $\{f(x, x), f(g(x), g(x))\}$ یا $\{P(g(x), x), P(x, g(x))\}$ اگر بخواهیم جایگزینی $\{x / g(x)\}$ را بپذیریم باید طبق الگوریتم $g(x)$ را جایگزین x کنیم که این جایگزینی شامل x موجود در خود $g(x)$ نیز می‌باشد و یک نوع فراخوانی بازگشتی بی نهایت به وجود خواهدآمد. $x / g(g(g(g\ldots\ldots))))$

مثال ۱: کدام زوج از عبارات زیر قابل یکسان سازی هستند؟

الف. $\begin{cases} P(f(x), y) \\ P(y, f(x)) \end{cases}$ ب) $\begin{cases} P(f(x), x) \\ P(x, x) \end{cases}$

ج) $\begin{cases} P(x, y) \\ P(y, f(x)) \end{cases}$ د) $\begin{cases} P(f(x), x) \\ P(y, f(y)) \end{cases}$

مثال ۲: کدام زوج از عبارات زیر قابل یکسان سازی هستند؟

الف) $\begin{cases} Q(x, g(w, t)) \\ Q(g(A, B), y) \end{cases}$ ب) $\begin{cases} Q(x, g(y, y)) \\ Q(A, g(B, C)) \end{cases}$

ج) $\begin{cases} Q(x, y) \\ Q(g(x, y), B) \end{cases}$ د) $\begin{cases} Q(B, g(x, y) \\ Q(g(x, y), B) \end{cases}$

فراکردهای[1] معین منطق مرتبه اول:

بندهای معین، ترکیبات فصلی لیترال‌هایی است که دقیقاً یکی از آن‌ها مثبت باشد. یک فراکرد معین می‌تواند به صورت یک جمله اتمی یا یک جمله شرطی که مقدم آن ترکیب عطفی لیترال‌های مثبت و تالی آن یک لیترال مثبت است نشان داده شود. برخلاف لیترال‌های گزاره‌ای، لیترال‌های مرتبه اول می‌تواند شامل متغیر نیز باشد. در این مورد متغیرها دارای سور عمومی هستند و سورهای عمومی در فراکردهای معین مرتبه اول حذف می‌شوند. فراکردهای معین، فرم متعارف برای استفاده از قانون GMP است. به چند نمونه از بندهای معین توجه فرمائید.

---

[1] Define clause

$King(x) \wedge Greedy(x) \Rightarrow Evil(x)$

$King(John)$

$Greedy(y)$

پایگاه دانش نمونه:

قبل از اینکه به معرفی الگوریتم‌های استنتاج در منطق مرتبه اول بپردازیم لازم است با یک پایگاه دانش نمونه آشنا شویم و جملات موجود در آن را به منطق مرتبه اول تبدیل کنیم سپس الگوریتم‌های خود را روی این پایگاه دانش اجرا خواهیم کرد.

«قانون می‌گوید این یک جنایت است که یک آمریکایی به ملت‌های متخاصم اسلحه بفروشد. کشور Nono دشمن آمریکا می‌باشد و دارای تعدادی موشک می‌باشد و تمام این موشک‌ها توسط کلنل وست که یک سرهنگ آمریکایی است به این کشور فروخته شده است. »

با توجه به جملات موجود در پایگاه دانش، ثابت کنید که West مجرم است.

The law says that it is a crime for an American to sell weapons to hostile nations.

The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.

Prove that Colonel West is a criminal

یک آمریکایی که به ملت‌های متخاصم اسلحه بفروشد متخاصم است.

$American(x) \wedge Weapon(y) \wedge Sells(x,y,z) \wedge Hostile(z) \Rightarrow Criminal(x)$

کشور Nono دارای تعدادی موشک می باشد.

$\exists x\ Owns(Nono,x) \wedge Missile(x)$: $Owns(Nono,M_1)$ and $Missile(M_1)$

تمام این موشک‌ها توسط کلنل وست فروخته شده اند.

$Missile(x) \wedge Owns(Nono,x) \Rightarrow Sells(West,x,Nono)$

موشک، نوعی اسلحه است.

$Missile(x) \Rightarrow Weapon(x)$

هر کشوری که دشمن آمریکا باشد متخاصم است.

$Enemy(x,America) \Rightarrow Hostile(x)$

West یک آمریکایی است.

$American(West)$

کشور Nonoدشمن آمریکا است.

$Enemy(Nono,America)$

... it is a crime for an American to sell weapons to hostile nations:

American(x) ∧ Weapon(y) ∧ Sells(x,y,z) ∧ Hostile(z) ⇒ Criminal(x)

Nono … has some missiles, i.e., ∃x Owns(Nono,x) ∧ Missile(x):

Owns(Nono,M1) and Missile(M1)

… all of its missiles were sold to it by Colonel West

Missile(x) ∧ Owns(Nono,x) ⇒ Sells(West,x,Nono)

Missiles are weapons:

Missile(x) ⇒ Weapon(x)

An enemy of America counts as "hostile“:

Enemy(x,America) ⇒ Hostile(x)

West, who is American …

American(West)

The country Nono, an enemy of America …

Enemy(Nono,America)

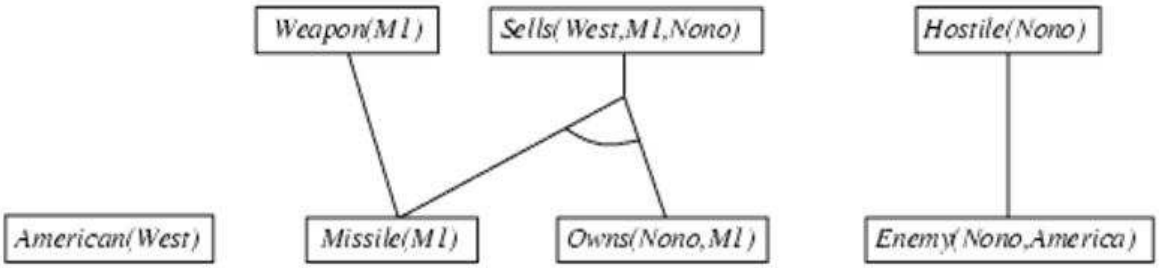
الگوریتم زنجیره ساز رو به جلو:

این الگوریتم برای بندهای معین (کلاز هورن) کار می‌کند. در این الگوریتم ابتدا، از جملات اتمی موجود در پایگاه دانش شروع می‌کنیم. اگر تمام مقدمات یک جمله شرطی درست باشد نتیجه آن جمله شرطی به حقایق موجود در KB اضافه می‌شود. این فرایند ادامه می‌یابد تا وقتی که پرسش q به KB اضافه شود یا استنتاج دیگری ممکن نباشد این الگوریتم استنتاج، برای سیستم‌هایی مفید است که استنتاج‌هایی در پاسخ به اطلاعات دریافتی جدید انجام می‌دهند. بنابراین ساخت پایگاه دانش فقط با استفاده از فراکردهای معین ارزشمند است زیرا می‌توان از هزینه Resolution اجتناب کرد.

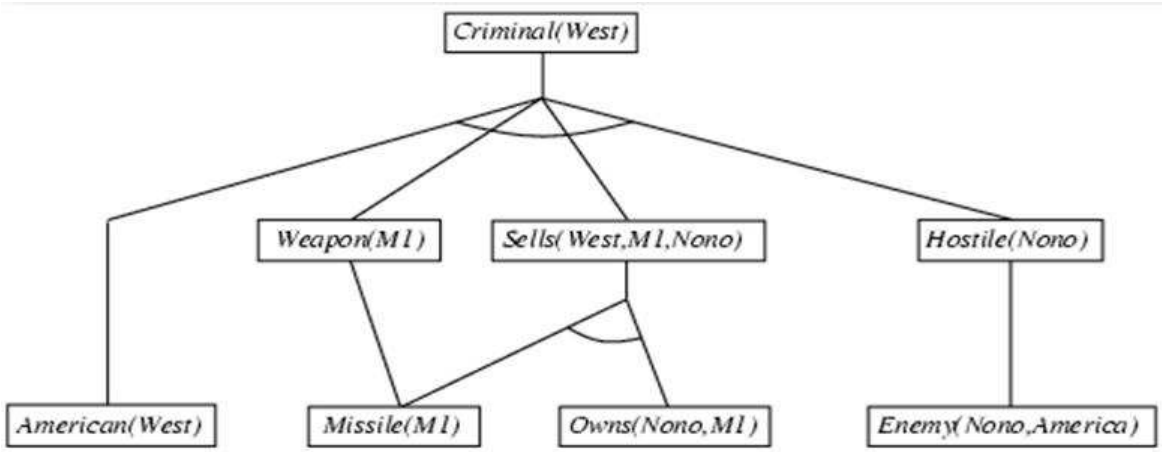
اثبات به روش زنجیره‌ی مستقیم

گام اول

گام دوم:



گام سوم:



پایگاه دانش Datalog:

پایگاه دانشی که شامل مجموعه‌ای از فراکردهای معین مرتبه اول فاقد سمبل تابع باشد را پایگاه دانش log Data می‌گوییم. (مانند پایگاه دانش معرفی شده فوق). عدم وجود سمبل تابع، عمل استنتاج را آسان تر می‌کند. الگوریتم زنجیره سازی به جلو برای فراکردهای معین مرتبه اول کامل و صحیح است. همچنین الگوریتم زنجیره ساز به جلو برای Datalog با تعداد تکرارهای محدود متوقف می‌شود ولی در حالت کلی اگر پایگاه دانش مستلزم جمله α نباشد ممکن است متوقف نشود و این مشکل اجتناب ناپذیر است.

زنجیره ساز به جلو افزایشی:

در این الگوریتم در تکرار t، فقط قوانینی را چک می‌کنیم که مقدم آن شامل یک عطف مثل P_i باشد که با حقیقت P'_i که در تکرار t-1 استنتاج شده است یکسان سازی شود سپس در مرحله تطبیق قانون، P_i را با P'_i جایگزین کرده و اجازه می‌دهد که سایر عطف‌های قانون با حقایقی از تکرارهای قبلی تطبیق یابد که خود عمل تطبیق می‌تواند پر هزینه باشد. بوسیله شاخص بندی پایگاه دانش اجازه داده می‌شود که حقایق شناخته شده در زمان O(1) بازیابی شوند. استنتاج روبه جلو به طور گسترده‌ای در پایگاه داده‌های استنتاجی بکار گرفته می‌شود. پس به طور خلاصه زنجیره سازی روبه جلو افزایشی یعنی اینکه در تکرار Kام نیاز به تطبیق قانونی که هیچ کدام از شرایطش در تکرار K-1 اضافه شده است نیست.

الگوریتم زنجیره ساز روبه عقب:

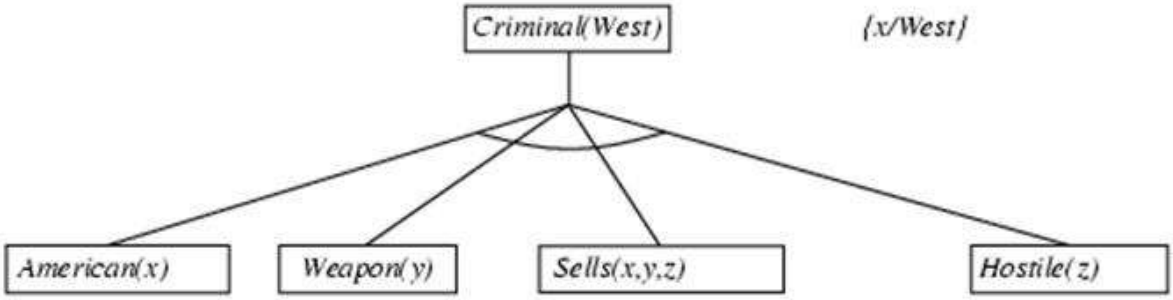
این الگوریتم از هدف به سمت عقب برمی گردد تا به حقایق شناخته شده (جملات اتمیKB) که اثبات جمله هدف را پشتیبانی می‌کنند برسد. این الگوریتم با لیستی به نام Goals که اهداف در خود نگه می‌دارد و در ابتدا فقط شامل پرسش یا هدف است فراخوانی می‌شود. لیست اهداف را می‌توان به عنوان یک پشته در نظر گرفت که تمام عناصر موجود در لیست Goals باید ارضا شوند تا شاخه جاری با موفقیت اثبات شود. الگوریتم زنجیره ساز روبه عقب، یک الگوریتم اول عمق بازگشتی است و نیازمندی‌های حافظه برای این الگوریتم بر حسب سایز اثبات جمله خطی است (O(n)).

معایب: زنجیره ساز روبه عقب برخلاف زنجیره سازی روبه جلو از دو مشکل زیر رنج می‌برد:

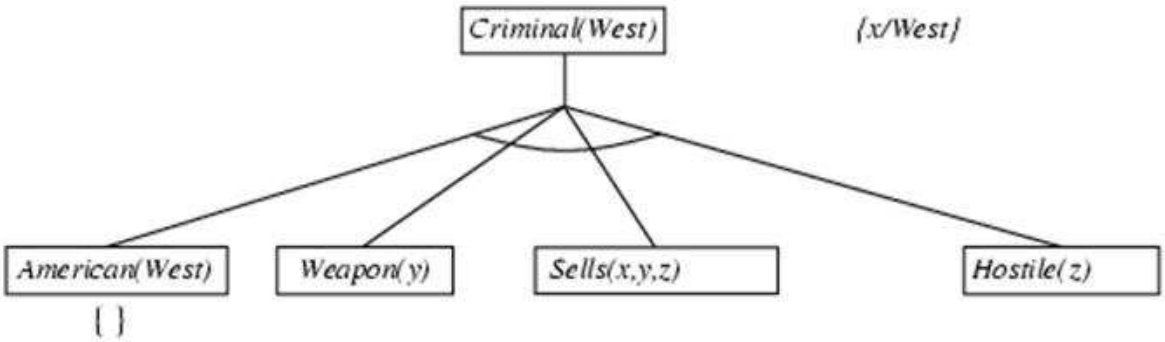
- **ناکامل بودن:** به دلیل وجود حلقه‌ها.
  - **راه حل:** مقایسه هدف فعلی با تمام اهداف موجود در پشته
- **ناکارا بودن:** به دلیل زیر هدف‌های تکراری.
  - **راه حل:** ذخیره نتایج قبلی با هدف حافظه بیشتر که نوعی Memorization است انجام می‌گیرد.

Memorization در سیستم زنجیره سازی رو به عقب یعنی اینکه راه حل‌های اهداف فرعی[1] (هدف اصلی همان پرسش است و اهداف فرعی جملاتی هستند که در لیستGoals قرار می‌گیرد و باید اثبات شوند) در حافظه کش نگهداری می‌شوند و با رسیدن مجدد به آن اهداف فرعی به جای تکراری محاسبات قبلی، از همان راه حل‌های ذخیره شده در کش استفاده می‌شود. الگوریتم زنجیره ساز روبه عقب، نوعی استدلال هدف گراست که به طور گسترده‌ای در برنامه نویسی منطقی و زبان پرولوگ بکار گرفته می‌شود.

گام اول

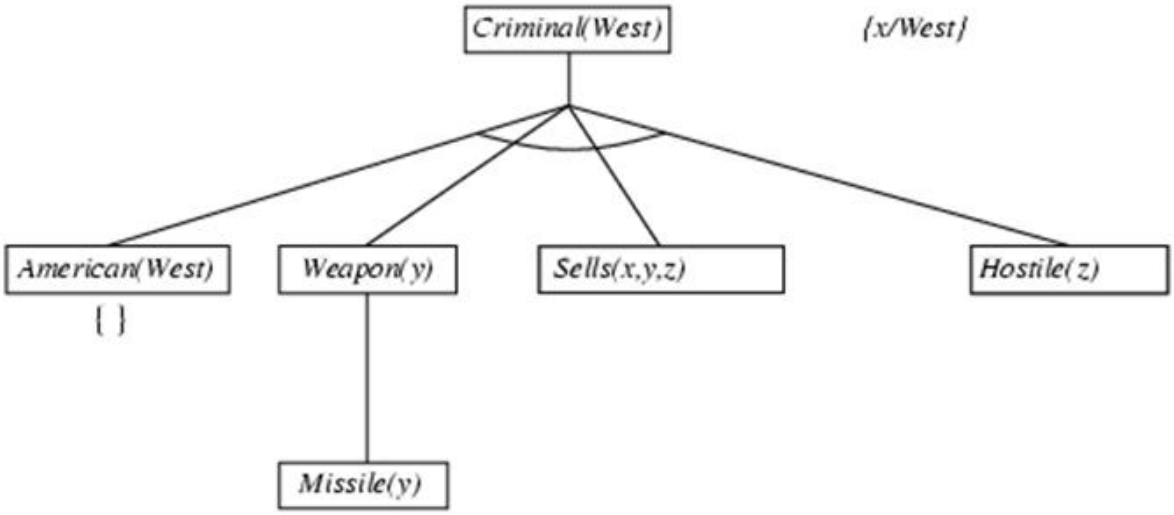


گام دوم

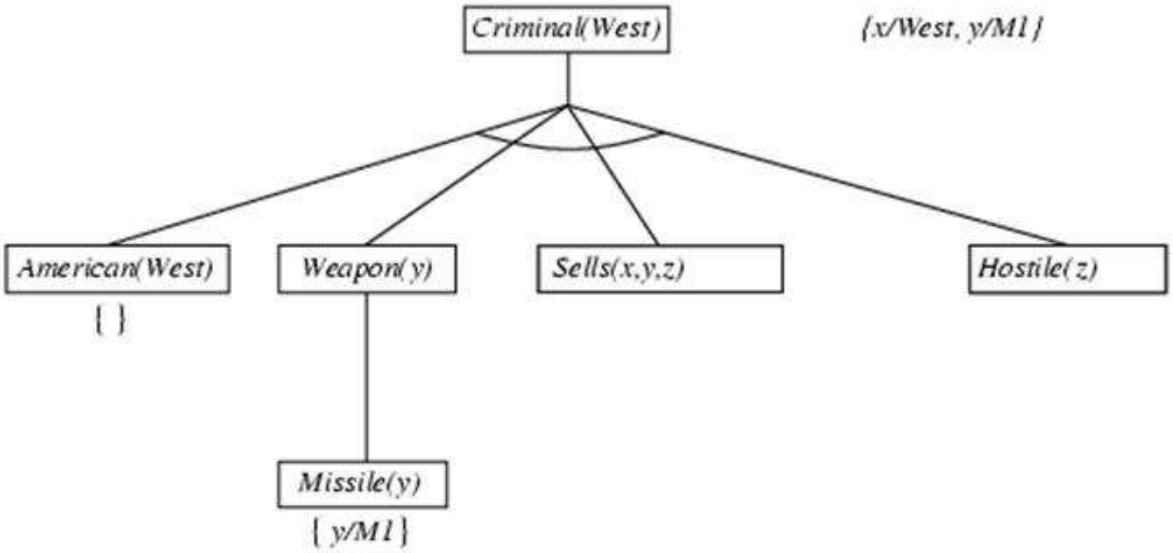


[1] Sub-goal

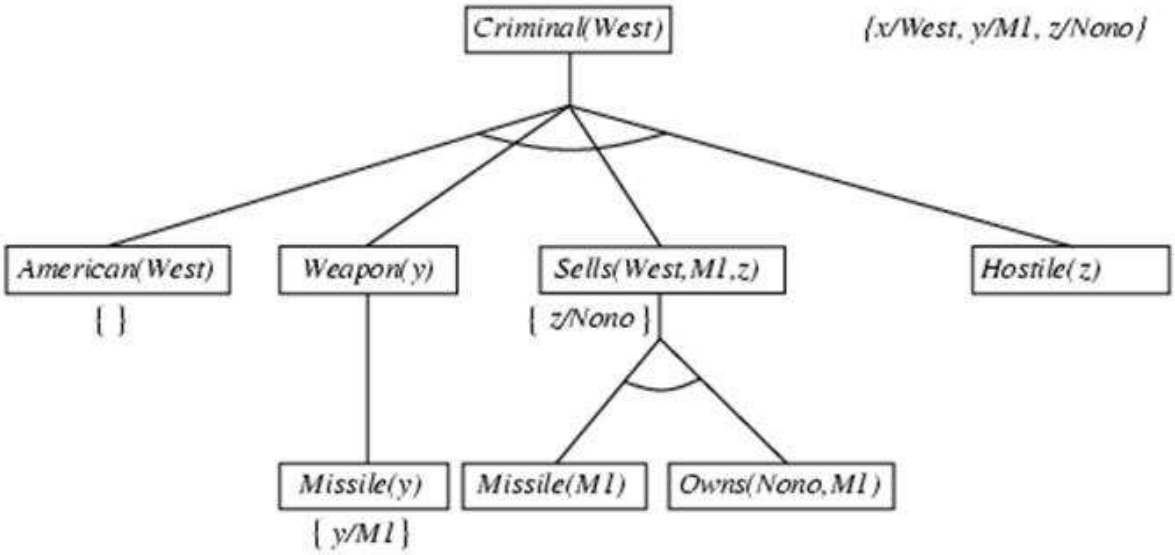
گام سوم



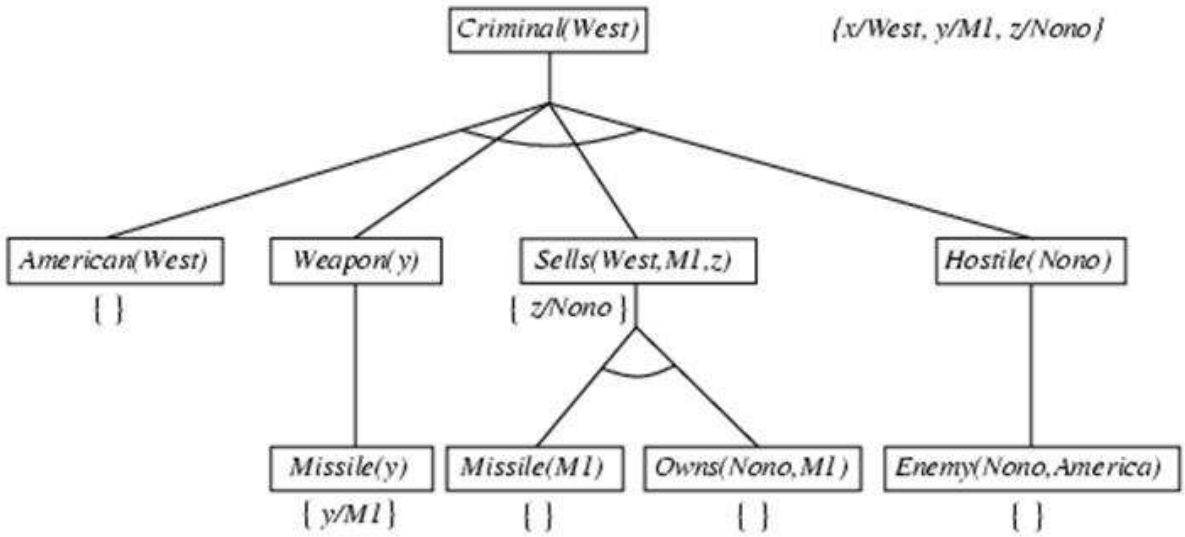
گام چهارم



گام پنجم



گام ششم



برنامه نویسی پرولوگ (منطقی):

زبان پرولوگ[1] متعلق به گروهی از زبان‌های برنامه نویسی است که زبان برنامه نویسی منطقی نام دارد. این زبان بر مبنای منطق مرتبه اول بنا نهاده شده است. برنامه نویسی منطقی به بازنمایی ایده اعلانی بسیار نزدیک است. ایده اعلانی عبارتست از اینکه سیستم باید با نمایش دانش در یکم زبان رسمی شناخته شود و مسایل آن باید با اجرای فرایندهای استنتاج بر روی آن دانش، حل شوند. این ایده در معادله Robert Kawalsi به صورت زیر است.در این زبان رابطه زیر برقرار است:

Algorithm = Logic + Control

- ❖ برنامه‌های پرولوگ مجموعه‌ای از فراکردهای معین هستند که با فرم منطق مرتبه اول استاندارد، کمی متفاوت است.
- ❖ پرولوگ از حروف بزرگ برای متغیرها و حروف کوچک برای ثابت‌ها استفاده می‌کند.
- ❖ برای نوشتن یک فراکرد ابتدا رأس و بعد بدنه‌ی آن نوشته می‌شود.
- ❖ به جای ⇒ از :- استفاده می‌شود.
- ❖ لیترال‌های موجود در بدنه با، از هم جدا می‌شوند.
- ❖ انتهای جمله با . مشخص می‌شود.

مثال:

American (x) ^ weapons(y) ^ sells(x, y, z) ^ Hostile(z) ⇒ Criminal

Prolog: Criminal(x): - american (x), weapons (y), sells(x), Hostile(z).

ویژگی‌ها: پرولوگ برای استنتاج، از روش زنجیره سازی روبه عقب روی فراکردهای Horn استفاده می‌کند و به طور گسترده‌ای در ژاپن و امریکا استفاده شده است مخصوصاً در پروژه نسل پنجم کامپیوترها. یکی از قابلیت‌های قابل توجه

[1] Programming in logic

در زبان پرولوگ پردازش لیست‌هاست. لیست، دنباله‌ی مرتبی از عناصر با هر طول متناهی است عناصر می‌توانند ثابت، متغیر یا زیر لیست باشند. هر لیست در زبان پرولوگ بین دو علامت [ ] قرار می‌گیرد. عنصر اول هر لیست را head و باقیمانده را tail می‌گویند.

نکته: پس به طور خلاصه برای برنامه نویسی منطقی داریم:

- برنامه = مجموعه‌ای از عبارات. head:- $literal_1$, … $literal_n$
- **مثال:** criminal(X):- american(X), weapon(Y), sells(X, Y, Z), hostile(Z).
- اضافه کردن یک لیست به انتهای لیست دیگر برای تولید یک لیست جدید
  - append([ ], Y, Y) الحاق با لیست خالی برابر با خود لیست
  - append([X | L], Y, [X | Z]): - append(L, Y, Z).

- پرس و جو:
  - append(A, B, [1,2])?
- پاسخ ها:
  - A = [ ] B = [1,2]
  - A = [1] B = [2]
  - A = [1,2] B = [ ]

نکاتی دیگر در مورد پرولوگ:

i. هر برنامه شامل دنباله‌ای از جملات است که به طور ضمنی به هم عطف شده‌اند.

ii. تمامی متغیرها به طور ضمنی دارای سور عمومی هستند.

iii. تمام استنتاج‌ها به طور زنجیره سازی روبه عقب با الگوریتم جستجوی اول عمق صورت می‌گیرد یعنی در صورت رسیدن به بن بست برای اثبات یک جمله پرولوگ، به آخرین مرحله ما قبل برگشته و تلاش می‌کند راه حل‌های دیگر را آزمایش کند به این کار Back traking گفته می‌شود.

دستور Cut:

یکی از دستورات بسیار مهم در زبان پرولوگ دستور cut یا برش است. این دستور به زبان پرولوگ اجازه می‌دهد که مکانیزم Back traking را از محل وقوع برش متوقف کرده تا زمان جستجو کاهش پیدا کند.

دو دلیل اصلی استفاده از دستور برش عبارتند از:

- برنامه شما سریع تر عمل خواهد کرد چون زیر هدف‌های متعددی را بررسی نخواهد کرد.
- برنامه شما حافظه کمتری مصرف خواهد کرد.

موارد استفاده از دستور Cut:

i. زمانی که می‌خواهیم به پرولوگ بگوییم که قانون درست را برای هدف معین یافته است. بنابراین cut می‌گوید که اگر این مسیر را ادامه بدهی به هدف درست می‌رسی و نیاز به بازگشت به عقب نمی‌باشد.

ii. زمانیکه می‌خواهیم به پرولوگ بگوییم ادامه دادن برای تطبیق با هدف بی فایده است و دیگر این مسیر را ادامه ندهد، از این دستور استفاده می‌شود..

iii. زمانی که می‌خواهیم پاسخ‌های بعدی ممکن را تولید نکنیم دستور cut به پرولوگ می‌گوید که دیگر به دنبال پاسخ بعدی نباشد.

به مثال زیر توجه فرمایید:

a^b → P	P:-a, !b.
c → P	P: -c.

تحلیل[1]:

تحلیل برای جملات منطق مرتبه اول، نسخه ارتقا یافته تحلیل در منطق گزاره‌ای است. در منطق گزاره‌ای دو لیترال مکمل یکدیگر هستند، اگر یکی نقیض دیگری باشد اما در منطق مرتبه اول دو لیترال مکمل یکدیگرند، اگر یکی از آنها نقیض دیگری یکسان ساز شود. قانون Resolution در منطق مرتبه اول به شکل زیر می‌باشد:

رزولوشن در مرتبه اول

$$\frac{l_1 \vee \cdots \vee l_k, \qquad m_1 \vee \cdots \vee m_n}{(l_1 \vee \cdots \vee l_{i-1} \vee l_{i+1} \vee \cdots \vee l_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n)\theta}$$

زمانی که $\text{Unify}(l_i, \neg m_j) = \theta$.

$$\frac{\begin{array}{c}\neg Rich(x) \vee Unhappy(x) \\ Rich(Ken)\end{array}}{Unhappy(Ken)} \qquad \theta = \{x/Ken\}$$

برای استفاده از Resolution در منطق مرتبه اول لازم است جملات به فرم نرمال عطفی CNF تبدیل شوند. فرم نرمال عطفی، شامل ترکیبات فصلی از لیترال‌هاست.

مراحل تبدیل به CNF:

i. حذف ترکیب شرطی

ii. انتقال ~ به داخل عبارات

iii. استاندارد کردن متغیرها (تغییر نام)

iv. فرایندحذف سور وجودی[2]

v. حذف سور عمومی و انتقال سورها به سمت چپ ترین

vi. ۶-توزیع ∨ بر ∧

تبدیل به فرم CNF

مثال: هر کس همه حیوانات را دوست داشته باشد، توسط یک نفردوست داشته خواهد شد.

---

[1] Resolution

[2] Skolemization

$\forall x\ [\forall y\ Animal(y) \Rightarrow Loves(x, y)] \Rightarrow [\exists y\ Loves(y, x)]$

❖ استلزام‌ها را حذف کنید

$\forall x\ [\neg\forall y\ \neg Animal(y) \vee Loves(x, y)] \vee [\exists y\ Loves(y, x)]$

❖ را به درون حرکت دهید.

$\neg\forall x\ p \equiv \exists x\ \neg p,\ \ \neg\exists x\ p \equiv \forall x\ \neg p$

$\forall x\ [\exists y\ \neg(\neg Animal(y) \vee Loves(x, y))] \vee [\exists y\ Loves(y, x)]$

$\forall x\ [\exists y\ \neg\neg Animal(y) \wedge \neg Loves(x, y)] \vee [\exists y\ Loves(y, x)]$

$\forall x\ [\exists y\ Animal(y) \wedge \neg Loves(x, y)] \vee [\exists y\ Loves(y, x)]$

❖ استاندارد ساز متغیرها

$\forall x\ [\exists y\ Animal(y) \wedge \neg Loves(x, y)] \vee [\exists z\ Loves(z, x)]$

❖ اسکولم سازی

$\forall x\ [Animal(F(x)) \wedge \neg Loves(x, F(x))] \vee Loves(G(x), x)$

❖ حذف سورها (عمومی)

$[Animal(F(x)) \wedge \neg Loves(x, F(x))] \vee Loves(G(x), x)$

❖ توزیع $\vee$ بر $\wedge$

$[Animal(F(x)) \vee Loves(G(x), x)] \wedge [\neg Loves(x, F(x)) \vee Loves(G(x), x)]$

استاندارد سازی متغیرها:

استاندارد سازی به این معناست که هر سور باید از یک متغیر استفاده کند و یک سور نمی‌تواند بیش از یک متغیر استفاده کنند. بنابراین برای جملاتی مثل $[\forall x, p(x)]\ v\ [\exists x, p(x)]$ که دو سور از متغیرهای همنام استفاده می کنند باید نام یکی از متغیرها را تغییر نام بدهیم. بدین ترتیب وقتی بعداً سورها حذف می‌شوند تناقضی پیش نخواهد آمد. تغییر نام‌های مشابه با توجه به حوزه عملکرد سورها صورت می‌گیرد هر سور یک حوزه عملکرد دارد که با) یا] شروع و با (یا [پایان می‌یابد.

$\forall x\ [\exists y\ Animal(y) \wedge \neg Loves(x, y)] \vee [\exists z\ Loves(z, x)]$

نکته: اگر متغیری در حوزه سور عمومی یا وجودی باشد محدود شده و اگر درحوزه هیچ سوری نباشد متغیر آزاد[1] نام دارد.

مثال: x موجود در q آزاد و x موجود در p محدود شده است.

$\exists x\ [p(x,y)] \qquad q(x)$

[1] free

فرایند حذف سور وجودی (اسکولمایز):

فرض کنید $\exists x, w(x)$ یک عبارت خوش فرم یا بخش بزرگتری از عبارت خوش فرم باشد.

اگر $\exists x$ در حوزه هیچ سور عمومی نباشد مقدار ثابتی مثل C را انتخاب و به جای x در $w(x)$ قرار می‌دهیم.

اگر x داخل حوزه سورهای عمومی $\exists[\forall_{x1}, \forall_{x2}, \dots, \forall_{xn}]$ باشد تابعی همانند F را انتخاب کرده و حاصل به صورت $w\ F(x_1, \dots\dots x_n)$ خواهد بود.

به چند مثال در این زمینه توجه فرمائید.

$\exists x, p(x, y)$

$\forall x, \exists y\ p(x, y)$

$\exists x \forall y \forall z \exists u \forall v \exists w\ p(x,y,z,u,v,w)$

به چند نمونه تست در این زمینه توجه فرمائید:

کدام گزینه بیانگر فرم clause (CNF) جمله مقابل است؟ (مهندسی کامپیوتر- ۸۱)

$[\forall x, P(x) \rightarrow Q(x)] \rightarrow R(a)$

کدام گزینه نقیض عبارت مقابل است ؟(مهندسی کامپیوتر- ۸۱)

$\exists y \forall x \forall z [P(x) \wedge Q(x,y) \rightarrow R(x,y,z)]$

کدام یک از موارد زیر فرم clause جمله مقابل است؟ (IT-۸۴)

$[\forall x, S(x,R) \rightarrow L(x,G)] \rightarrow H(R)$

مثال در مورد Resolution:

فرض کنید KB زیر را داریم:

الف. هر سگی یا انسانی را دوست دارد یا از گربه متنفر است.

ب. Rover یک سگ است.

ج. Rover گربه‌ها را دوست دارد.

حکم: اثبات کنید بعضی از سگ‌ها هستند که گربه‌ها را دوست دارند.

- **H(x):** متنفر بودن از گربه‌ها توسط x.
- **D(x):** معرف سگ بودن x.
- **L(x):** معرف دوست داشتن فردی توسط x.

الف) $\forall x\ D(x) \rightarrow L(x) \vee H(x)$

ب) $D(\text{Rover})$

ج) $\sim H(\text{Rover})$

حکم: $\exists x\ (D(x) \wedge L(x))$

CNF موارد بالا:

الف) $\sim D(x) \vee L(x) \vee H(x)$

ب) $D(\text{Rover})$

ج) $\sim H(\text{Rover})$

حکم: $D(y) \wedge L(y)$

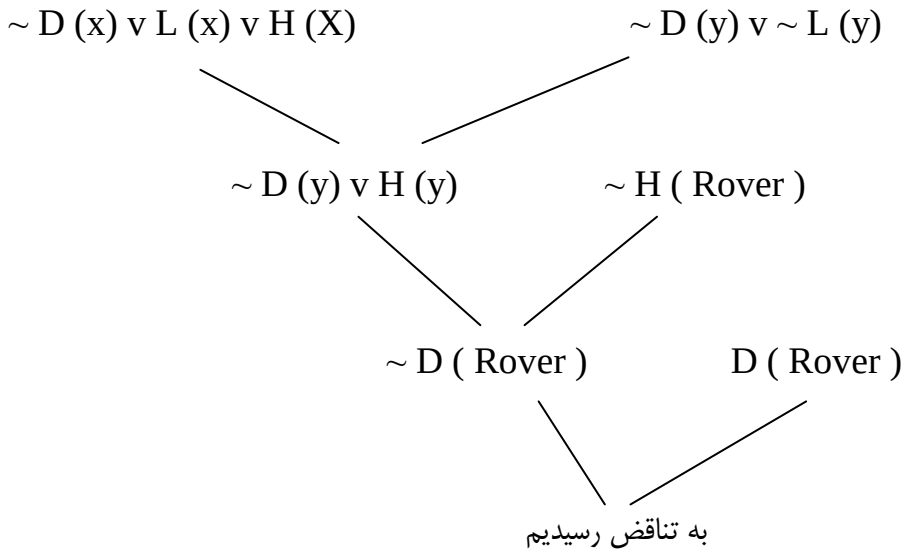
نقیض حکم: $\sim D(y) \vee \sim L(y)$

الگوریتم Resolution:

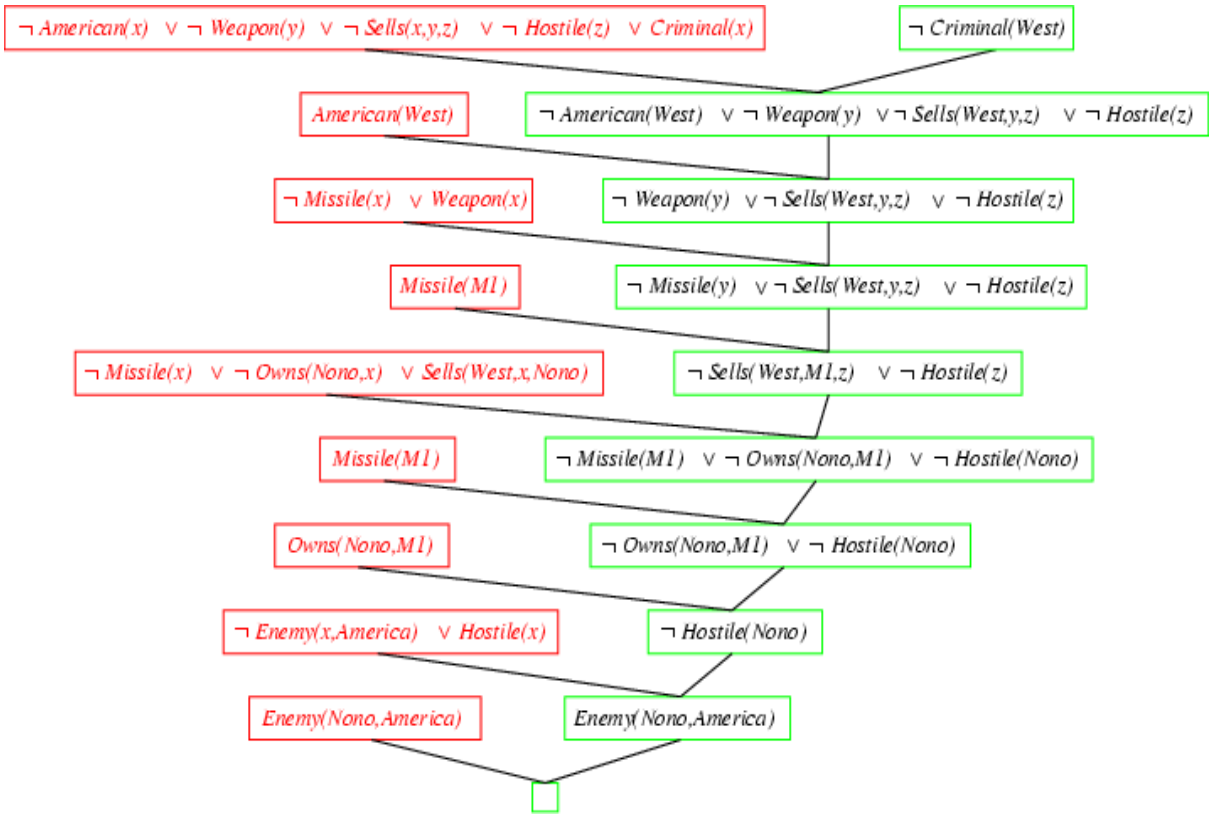
i. ابتدا فرضیات و نقیض حکم را به صورت گزاره‌های منطق مرتبه اول بیان می‌کنیم.

ii. فرضیات و نقیض حکم را با روش تبدیل گزاره به CNF، به CNFهای معادل تبدیل می‌کنیم.

iii. قانون Resolution را به این مجموعه تا زمانی اعمال می‌کنیم تا به تناقض برسیم.



اثبات مجرم بودن West توسط Resolution:

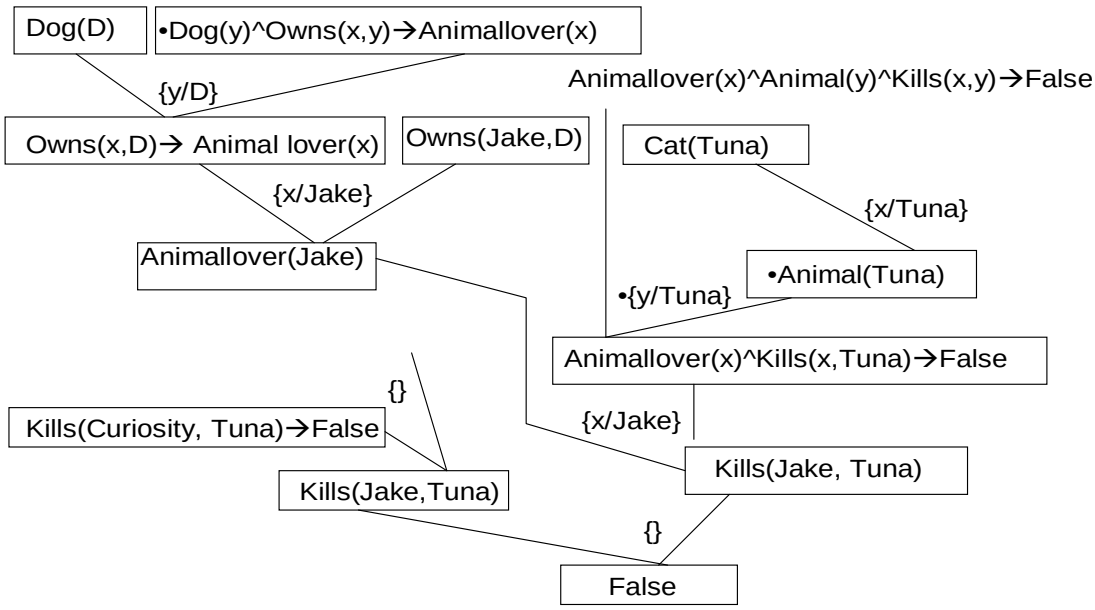


مثال jake,Tuna به وسیله Resolution:

- ♠ Jake owns a dog
- ♠ Every dog owner is an animal lover
- ♠ No animal lover kills an animal
- ♠ Either Jake or curiosity killed the cat, who named Tuna
- ♠ Did curiosity kill the cat?

- A. ∃x Dog (x) ^ Owns (Jake ,x)
- B. ∀x [∃y Dog(y) ^ Owns(x,y)→Animal lover(x)]
- C. ∀x Animallover(x)→∀x Animal(y)→~Kills(x,y)
- D. Kills(Jake, Tuna) v Kills(Curiosity, Tuna)
- E. Cat (Tuna)
- F. ∀x Cat(x) → Animal(x)

- A1. Dog(D)
- A2. Owns(Jake,D)
- B. Dog(y) ^ Owns(x,y)→ Animal lover(x)
- C. Animal lover(x) ^ Animal(y) ^ Kills(x,y)→False
- D. Kills(Jake, Tuna) v Kills(Curiosity, Tuna)
- E. Cat(Tuna)
- F. Cat(x) → Animal(x)



استراتژی‌های تحلیل:

کاربرد پشت سرهم قانون Resolution در صورت وجود اثباتی را خواهد یافت اما کارایی آن ممکن است خوب نباشد. برای بهبودی کارایی این روش، استراتژی‌های زیر پیشنهاد می‌گردد:

I. اولویت (ترجیح) واحد[1]:

این روش ترجیح می‌دهد هنگامی از قانون Resolution استفاده شود که یکی از جملات فقط یک لیترال (جملاتی که فقط یک لیترال دارند Unit clause نامیده می‌شوند) باشد. ایده موجود در این روش این است که ما سعی داریم بند خالی ایجاد کنیم لذا بهتر است به استنتاج‌هایی اولویت داده شود که عبارات کوچکتری دارند. بکارگیری Resolution، برای یک جمله مانند P و هر جمله دیگری مانند $P \vee Q \vee R$ همیشه منجر به تولید بند کوچکتری می‌شود. اولویت واحد، فرم محدودی از Resolution است که در آن، هر مرحله از Resolution بایدحاوی یک عبارت واحد (Unit clause) باشد. این استراتژی در حالت کلی کامل نیست اما برای پایگاه دانش به فرم هورن کامل است.

[1] unit preference

II. مجموعه پشتیبان[1]:

تقدم‌هایی که در استفاده از Resolution ترتیب قایل می‌شوند، مفید هستند اما در حالت کلی بهتر است بعضی از Resolutionها حذف شوند. روش مجموعه پشتیبان این کار را انجام می‌دهد. این روش با شناسایی زیر مجموعه‌ای از جملات به نام مجموعه پشتیبان شروع می‌کند. هر Resolution، جمله‌ای از مجموعه پشتیبان را با جمله دیگر آن، ترکیب می‌کند و نتیجه را به مجموعه پشتیبان اضافه می‌کند اگر مجموعه پشتیبان نسبت به کل پایگاه دانش کوچک باشد فضای جستجو بسیار کوچک خواهد شد. در این شیوه، یک انتخاب نامناسب برای مجموعه پشتیبان، الگوریتم را ناکامل خواهد کرد. استراتژی مجموعه پشتیبان دارای این مزیت است که درخت‌های اثباتی را تولید می‌کند که اغلب برای درک افراد آسان می‌باشد زیرا این روش هدف گرا می‌باشد.

III. تحلیل ورودی[2]:

در این استراتژی، هر Resolution، یکی از جملات ورودی (از KB یا پرسش) را با یک جمله دیگر ترکیب می-کند. اثبات مجرم بودن West که دارای شکل محوری است از این روش استفاده می‌کند. در پایگاه‌های دانش هورن، قانون Modus Ponens (انتزاع) نوعی از استراتژی تحلیل ورودی است زیرا یک ترکیب شرطی از KB اصلی را با دیگر جملات ترکیب می‌کند. بنابراین تعجبی ندارد که تحلیل ورودی برای پایگاه‌های دانشی که به صورت هورن هستند کامل باشد اما در حالت کلی ناکامل است.

IV. شمول[3]:

این روش تمام جملاتی را که حالت خاصی از یک جمله موجود در KB هستند، از KB حذف می‌کند. به عنوان مثال، اگر جمله P(x) در KB موجود باشد، نیاز به افزودن P(A) نیست. این روش به نگهداری KB، به صورت کوچک کمک می‌کند و در نتیجه فضای جستجو را کاهش می‌دهد.

---

[1] set of support

[2] Input Resolution

[3] subsumption

سوالات تستی آخر فصل

ترم اول ۸۷–۸۸

۱. در یک پایگاه دانش، مجموعه دانش زیر موجود است. کدامیک از گزینه‌ها از این مجموعه قابل استنتاج منطقی است؟

$\neg K(x) \vee M(x)$
$\neg K(x) \wedge Q(x)$
$L(N(x) \wedge M(g) \wedge P(x,y))$
$\neg Q(ali)$
$K(amir)$
$L(ali)$

الف. P(amir, ali)
ب. Q(ali) $\wedge$ M(amir)
ج. $N(amir) \vee \neg K(ali)$
د. $K(amir) \Rightarrow P(ali, amir)$

۲. کدامیک از گزینه‌های زیر نتیجه منطقی جملات مقابل است؟

$\exists x cat(x) \wedge owns(Hamid, x)$
$\forall x(\exists y, cat(y) \wedge owns(x,y) \Rightarrow animal-lover(x)$
$\forall x \forall y(Animal-lover(x) \wedge animal(y) \Rightarrow \neg Kills(x,y))$
$kills(Hamid, Puya) \vee kills(Behzad, Puya)$
$Fish(pupu)$
$\forall x(Fish(x) \Rightarrow Animal(x)$
$\neg kills(Hamid, Behzad)$

الف. بهزاد قاتل ماهی است.
ب. حمید دوستدار گربه است.
ج. بهزاد قاتل پویا است یا حمید قاتل بهزاد است.
د. حمید قاتل پویا است یا گربه قاتل پویا است.

ترم دوم ۸۷–۸۸

۳. کدام زوج از عبارات زیرقابل یکسان سازی (unification) هستند؟

الف. $\begin{cases} P(f(x), y) \\ P(y, f(x)) \end{cases}$
ب. $\begin{cases} P(f(x), f(x)) \\ P(x, x) \end{cases}$
ج. $\begin{cases} P(x, y) \\ P(y, f(x)) \end{cases}$
د. $\begin{cases} P(f(x), x) \\ P(y, f(y)) \end{cases}$

۴. در صورتی که پایگاه دانش زیر را داشته باشیم و از الگوریتم زنجیره سازی پیشرو(forward - chaining) استفاده نماییم چه نتایجی قابل دستیابی می‌باشد؟

$\forall x Shiny(x) \rightarrow Niceweather(x)$
$\forall x \forall y Healty(y) \wedge Niceweather(y) \rightarrow Gotoswim(x, y)$
$Shiny(Saturday)$
$Healty(A\min)$
$\forall x Gotoswim(x, Friday) \rightarrow Healty(x)$
$Gotoswim(Ali, Friday)$

الف. Healty (Ali) ب. Shiny (Friday)

ج. Gotoswim(Amin.Friday) د. Niceweather(Friday)

۵. هنگامی که می‌خواهیم سئوالی را از یک پایگاه دانش که قوانین آن به صورت منطبق مرتبه اول بیان شده‌اند بپرسیم، کدامیک از روش‌های زیر مناسب تر است؟

الف. جستجوی اول سطح

ب. استدلال پیشرو (Forward Reasoning)

ج. استدلال پسرو (Backward Reasoning)

د. جستجوی تولید و آزمون (Generate & Test)

ترم اول ۸۸–۸۹

۶. کدامیک از عبارات زیر با عبارت R(x), F(A), F(z) قابل یکسان سازی (Unification) است.
(x و yمتغیر و A و Bمقادیر ثابت هستند.)

الف. R(F(z), F(y), F(x)) ب. R(F(z), x, F(B))

ج. R(z, F(z), F(B)) د. R(F(y), y, x)

ترم دوم ۸۸–۸۹

۷. کدامیک از گزینه های زیر، عمومی ترین یکسان ساز دو عبارت زیر است؟

Unify (knows(Ali,x),knows(y,t))

الف. {y / ali, t / f(x)} ب. {y / ali, x / t}

ج. {y / ali, x / ali, t / ali } د. این عبارات قابل یکسان سازی نیستند.

۸. الگوریتم " زنجیره‌ای پیشرو در منطق مرتبه اول " بر روی کدامیک از پایگاه‌های دانش زیر مناسب است؟

الف.
$P(x) \wedge \neg Q(x) \Rightarrow R(x)$
$P(A)$
$\neg Q(x)$

ب.
$P(x) \vee Q(x) \Rightarrow R(x)$
$P(A)$
$Q(x)$

ج.
$P(x) \wedge Q(x) \Rightarrow R(x) \vee A$
$P(A)$
$\neg Q(x)$

د.
$P(x) \wedge Q(x) \Rightarrow R(x)$
$P(A)$
$Q(x)$

۹. در زبان پرولوگ (prolog) کدامیک از روش‌های استنتاج زیر استفاده می‌شود؟

الف. استدلال زنجیره پیشرو ب. استدلال زنجیره پسرو

ج. اثبات با برهان خلف د.استدلال پس گرد (backtracking)

۱۰. الگوریتم ریتی جهت بهبود کدامیک از معایب الگوریتم زنجیره‌ای پیشرو در منطق مرتبه اول ایجاد شده است؟

الف. کاهش هزینه مرحله انطباق الگو

ب. جلوگیری از انطباق‌های جزیی فراوانی که در هر تکرار ساخته و دور ریخته می‌شود.

ج. جلوگیری از تولید واقعیت‌های زیادی که با هدف مرتبط نیستند.

د.کامل نبودن آن

ترم تابستان ۸۹

۱۱. کدام زوج از عبارات زیر قابل یکسان سازی (Unification) هستند؟

الف. P(f(x), y) ب. P(f(x), f(x)) ج. P(x, y) د. P(f(x), x)

P(y, f(x)) P(x, x) P(y, f(x)) P(y, f(y))

ترم اول ۸۹–۹۰

۱۲. کدام ترتیب از جملات زیر باعث جستجوی نامحدود با حالت‌های تکراری در پرولوگ می‌شود؟

الف. $path(X,Z) :- path(X,Y), link(Y,Z).$ $path(X,Z) :- link(X,Z).$

ب. $path(X,Z) :- link(X,Z).$ $path(X,Z) :- path(X,Y), link(Y,Z).$

ج. $append([],Y,Y).$ $append([A|X],Y,[A|Z]) :- append(X,Y,Z).$

د. $p(X,[X|Y]).$ $P(X,[Y|Z]) :- p(X,Z).$

ترم دوم ۸۹–۹۰

۱۳. مساله ایجاب در منطق مرتبه اول چگونه است؟

(راهنمایی: تصمیم پذیر به معنی تولید پاسخ TRUE برای هر جمله ایجاب پذیر و FALSE برای ایجاب ناپذیر می‌باشد)

الف. نیمه تصمیم پذیر ب. تصمیم پذیر است

ج. نمی‌توان اظهار نظر کرد د. تصمیم ناپذیر است

تابستان ۹۰

۱۴. کدام گزینه، عبارت زیر پس از حذف سورهای عمومی و وجودی است؟

$\forall x[\exists y\ \ Animal(y) \wedge \sim Loves(x, y)] \vee [\exists z\ \ Loves(z, x)]$

الف. $[Animal(F(x)) \wedge \sim Loves(x, F(x))] \vee [Loves(B, x)]$

ب. $[Animal(F(x)) \wedge Loves(G(x), x)] \vee [\sim Loves(x, F(x)) \vee Loves(G(x), x)]$

ج. $[Animal(A) \wedge \sim Loves(x, A)] \vee [Loves(B, x)]$

د. $[Animal(F(x)) \wedge Loves(F(x), x)] \vee [\sim Loves(x, G(x)) \vee Loves(F(x), x)]$

۱۵. اگر عبارت مقابل درست باشند، آنگاه:

$$\begin{cases} Loves(Father(Bill), Bill) \\ Loves(x, y) \wedge \sim Loves(y, x) \end{cases}$$

الف. $\sim Loves(Bill, Father(Bill))$

ب. $Loves(Father(Bill), Father(Bill))$

ج. $Loves(Bill, Bill)$

د. $Loves(x, Bill)$

۱۶. عام‌ترین یکسان کننده (Most General Unifier) در عبارت زیر کدام است؟

$P(bill, F(bill), Y)$ $\quad$ $P(X, F(X), M(bill))$

الف. $\{X / bill\}$

ب. $\{X / bill, Y / M(bill)\}$

ج. $\{X / Y\}$

د. $\{F(bill) / Y, X / Y\}$

پاسخ نامه تستی

سوال	گزینه صحیح
۱	د
۲	ج
۳	الف
۴	الف
۵	ج
۶	د
۷	ب
۸	د
۹	ب
۱۰	ب
۱۱	الف
۱۲	ب
۱۳	الف
۱۴	ب
۱۵	الف
۱۶	ب

سوالات تشریحی آخر فصل

ترم دوم ۸۷–۸۸

عبارت زیر را به شکل نرمال عطفی (CNF) درمنطق مرتبه اول تبدیل کنید.

$$\forall x[\forall y Animal(y) \Rightarrow Loves(x,y)] \Rightarrow [\exists y Love(y,x)]$$

که به معنای زیراست:

Everyone who loves all animals is loved by someone

ترم تابستان ۸۸

با استفاده از قاعده استنتاج تحلیل (Resolution) هدف زیر را نتیجه بگیرید.

هدف: Kills(Curiosity , tuna)

$$\forall x[\forall y Animal(y) \Rightarrow Loves(x,y)] \Rightarrow [\exists y Loves(y,x)]$$
$$\forall x[\exists y Animal(y) \wedge Kills(x,y)] \Rightarrow [\forall z \neg Loves(z,x)]$$
$$\forall x Animal(x) \Rightarrow Loves(jack,x)$$
$$Kills(jack,Tuna) \vee Kills(Curiosity,Tuna)$$
$$Cat(Tuna)$$
$$\forall x Cat(x) \Rightarrow Animal(x)$$

ترم اول ۸۸–۸۹

در صورتی که جملات در منطق مرتبه اول به صورت زیر باشند آیا می‌توان Criminal(West) رانتیجه گرفت؟

الف. با الگوریتم زنجیره‌ای پیشرو نظریه خود را اثبات کنید. (۱/۵ نمره)

ب. با تحلیل اثبات (Resolution) نمایید. (۲نمره)

$$1)\forall x,y,z American(x) \wedge Weapon(y) \wedge Sells(x,y,z) \wedge Hostils(z) \Rightarrow Cra\min al$$
$$2)\exists x Owns(Nono,x) \wedge Missle(x)$$
$$3)\forall x Missle(x) \wedge Owns(Nono,x) \Rightarrow Sells(West,x,nono)$$
$$4)\forall x Missle(x) \Rightarrow Weapon(x)$$
$$5)\forall x Enemy(x,America) \Rightarrow Hostils(x)$$
$$6)American(West)$$
$$7)Enemy(Nono,America)$$

ترم دوم ۸۸–۸۹

مسئله زیر را در نظر بگیرید:

" برای یک آمریکایی، فروش تسلیحات به ملل متخاصم جرم است. کشور نونو که دشمن آمریکاست، تعدادی موشک دارد و تمامی موشک‌هایش را از سرهنگ وست که یک آمریکایی است، خریده است."

الف. مسئله فوق را به صورت بندهای معین مرتبه اول بازنمایی کنید. (۱ نمره)

ب. درخت اثبات بوسیله الگوریتم زنجیره‌ای پسرو، برای اثبات اینکه "وست یک مجرم است."را رسم کنید. (۰/۷۵ نمره)

ترم تابستان ۸۹

حذف سور عمومی، اسکولم سازی در سور وجودی، یکسان سازی و عمومی ترین یکسان ساز و جداسازی استاندارد (standardizing apart) و وارسی وقوع (occur check) و شبکه شمول (subsumptin lattice) را هر یک با مثالی مختصراً توضیح دهید.

ترم اول ۸۹–۹۰

جمله مقابل را در نظر بگیرید: " اسب‌ها حیوان هستند." در نتیجه "سر یک اسب سر یک حیوان است."

الف. مقدم و تالی جمله فوق را به زبان منطق مرتبه اول بنویسید. از سه مسند Headof(h, x) (به معنی h سر x است)، Horse(x) و Animal(x) استفاده نمایید.

ب. جمله را به شکل نرمال عطفی بنویسید.

ج. توسط تحلیل نشان دهید که تالی از مقدم نتیجه می‌شود.