هوش مصنوعی

فصل سوم
حل مسئله از طریق جستجو

مدرس : ایمان مختاری
دانشگاه پیام نور شهرکرد

عامل های حل مسئله

- عامل حل مسئله از دسته عامل های مبتنی بر هدف است.
- اگر به الگوریتمی هیچ اطلاعاتی به جز تعریف مسئله داده نشود،آن الگوریتم را ناآگاه می گوییم.
- الگوریتم های آگاه آن هایی هستند که ایده هایی (توابع هیوریستیک) راجع به حل مسئله دارند.

چهار گام حل مسئله

3

- تدوین هدف(فرموله کردن هدف) (مشخص کردن حالت هایی از دنیا که در آنها هدف برآورده شده است.)
- تدوین مسئله (فرایند تصمیم گیری در مورد انتخاب اقدامات و حالت هاست تا به یک هدف برسیم.
- جستجو(عاملی که با چند گزینه از مقادیر ناشناخته مواجه است،ابتدا باید دنباله های ممکن از اقداماتی را که منجر به حالت هایی با مقدار شناخته شده می شوند،بررسی کند و سپس بهترین دنباله را انتخاب کند.)
- اجرای اقدامات پیشنهادی مرحله جستجو

یک عامل ساده حل مسأله

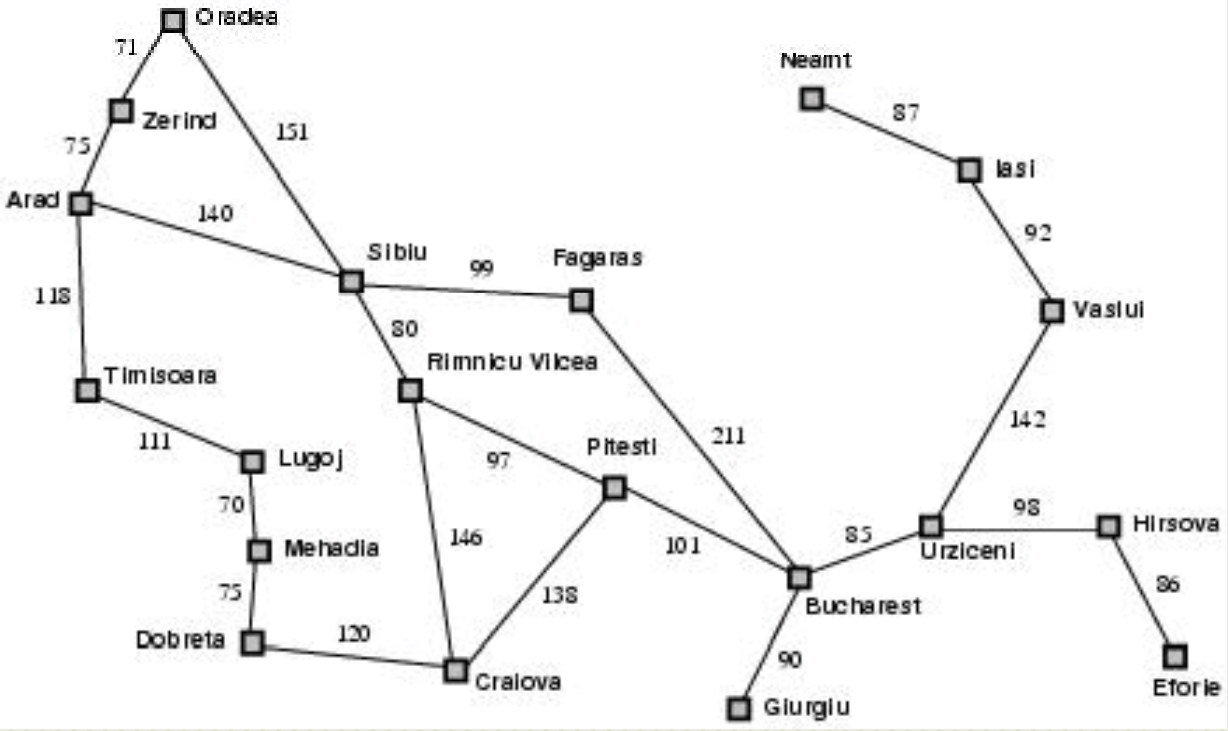
4

```
function SIMPLE-PROBLEM-SOLVING-AGENT(percept) returns an action
  inputs: percept, a percept
  static: seq, an action sequence, initially empty
          state, some description of the current world state
          goal, a goal, initially null
          problem, a problem formulation

  state ← UPDATE-STATE(state, percept)
  if seq is empty then do
      goal ← FORMULATE-GOAL(state)
      problem ← FORMULATE-PROBLEM(state, goal)
      seq ← SEARCH(problem)
  action ← FIRST(seq)
  seq ← REST(seq)
  return action
```

مثال مسئله رومانی

5



مثال مسئله رومانی

6

- عامل در شهر آراد است.
- عامل فردا باید از بخارست پرواز کند
- اکنون چهار گام برای حل مسئله رومانی را بیان می کنیم.

مثال مسئله رومانی

7

- **تدوین هدف:**
  - هدف عامل رسیدن به بخارست است
- **تدوین مسئله:**
  - **حالت ها :** شهر های مختلف
  - **اقدامات:** حرکت بین شهرها
- **جستجو:**
  - دنباله ای از شهر ها Arad > Sibiu > Fagaras > Bucharest
- **اجرا:**
  - اجرای راه حل پیدا شده در بخش جستجو

بررسی گام ۲: تدوین مسئله (مولفه های یک مساله)

8

1. **حالت ابتدایی:**
   - حالت اولیه که عامل از آن شروع میکند.
   - In(Arad)
2. **تابع پسین:**
   - تابع پسین که توصیفی از اقدامات ممکن را برای عامل مهیا می کند.
   - S(X) تابع پسین که **زوج اقدام – حالت** است.
   - S(Arad)={(Arad → Zerind, Zerind),…}

فضای حالت:

- مجموعی ای از حالت ها که از حالت اولیه می توان به آنها رسید،که به صورت گراف آنرا تشکیل می دهیم. حالت ها گره ها و فعالیت ها یال های آن را تشکیل میدهند.
- **فضای حالت** = حالت اولیه + حالاتی که با تابع پسین می توان به آنها رسید.

بررسی گام ۲: تدوین مسئله

9

3. **آزمون هدف:**
   - تعیین می کند آیا حالت خاصی ، حالت هدف است یا خیر.
   - مثلا بودن در بخارست
4. **هزینه مسیر:**
   - تابع هزینه مسیر که برای هر مسیر یک هزینه عددی در نظر می گیرد.معمولا تابع هزینه ای را انتخاب می کنیم که مقیاس کارایی عامل را مشخص کند.
   - مثال:مجموع فاصله ها ، تعداد اقدامات انجام شده
   - c(x,a,y) : هزینه گام برای اجرای اقدام a از حالت x به حالت y
   - در مسئله رومانی هزینه مسیر فاصله بین دو شهر به کیلومتر است.
   - راه حل یک مسئله ، مسیری از حالت ابتدایی تا حالت هدف است.راه حل بهینه راه حلی است که کمترین هزینه مسیر را داشته باشد.

تجرید حالات=فرایند حذف حالتهای جزیی مثلا ریزش باران از لیست حالات را تجرید حالات می گویند

تجرید اقدامات=فرایند حذف جزییات از لیست اقدامات را تجرید می گویند مثلا Carwash جزء اقدامات نیست

شرط اعتبار تجرید= درصورتی که بتوان هر راه حل مجرد را به یک راه حل مفصلتر تبدیل کرد مثلا از آراد تا سیبیو رادیو روشن و ....

مثال جاروبرقی

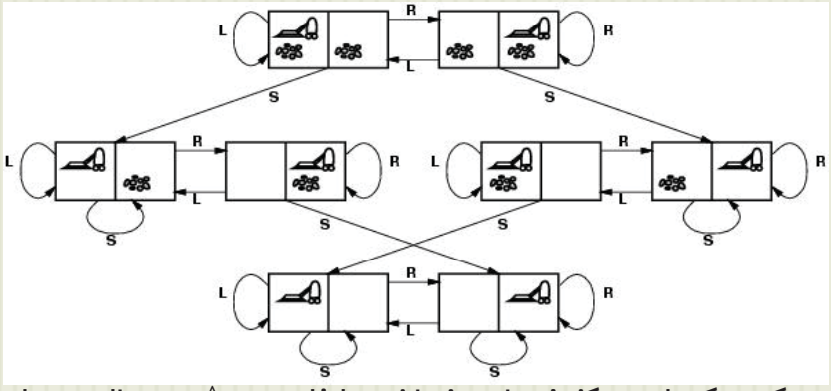
10

انواع مسائل

۱-مسائل اسباب بازی=برای نمایش و تمرین روشهای مختلف حل مساله (مقایسه کارایی الگوریتم ها)

۲-مسائل دنیای واقعی=راه حلهای آنها برای مردم اهمیت دارند

دنیای جارو برقی



حالتها: دو مکان که هر یک ممکن است کثیف یا تمیز باشند.لذا ۸ = ۲^۲ * ۲حالت در این جهان وجود دارد($n2^n$)

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

تابع پسین: حالتهای معتبر از سه عملیات: راست(R)، چپ(L)، مکش(S)

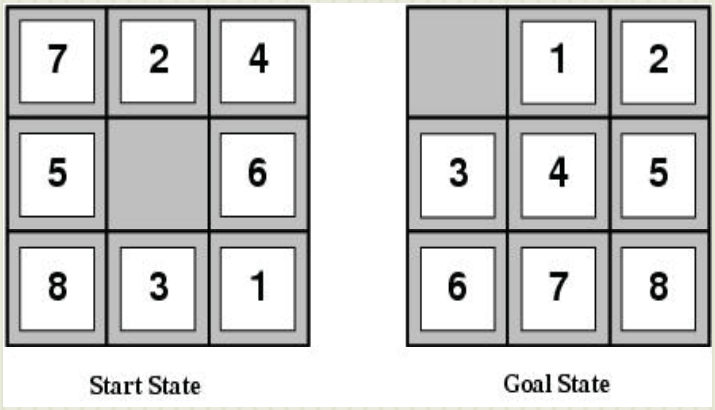
آزمون هدف: تمیزی تمام مربعها

هزینه مسیر: هر گام ۱ واحد هزینه دادرد،هزینه مسیر برابر است با تعداد گامها در مسیر

مثال معمای ۸

11

معمای ۸

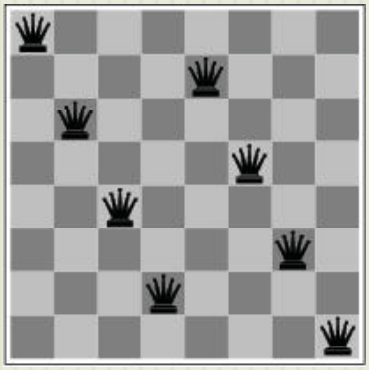


معمای ۸ نمونه‌ای است شامل یک صفحهٔ ۳*۳ با ۸ مربع شماره دار در یک صفحه خالی.
هر مربع که مجاور خانه خالی است. می‌تواند به درون آن خانه برود. هدف رسیدن به ساختاری است که در سمت راست شکل نشان داده شده است. نکته مهم این است که بجای اینکه بگوییم «مربع شماره ۴ را به داخل فضای خالی حرکت بده» بهتر است بگوییم «فضای خالی جایش را با مربع سمت چپش عوض کند.»

حالتها: مکان هر هشت خانه شماره دار و خانه خالی در یکی از ۹ خانه(9!/2)
حالت اولیه: هر حالتی را میتوان به عنوان حالت اولیه در نظر گرفت
تابع پسین: حالتهای چهار عمل برای انتقال خانه خالی(انتقال خانه خالی به چپ، راست، بالا پایین)
آزمون هدف: بررسی میکند که حالتی که اعداد به ترتیب چیده شده اند(طبق شکل روبرو) رخ داده یا نه
هزینه مسیر: هر گام 1 واحد هزینه دارد،هزینه مسیر= تعداد گامها در مسیر

مثال مسئله ۸ وزیر

12



هدف از مسئله ۸ وزیر، قرار دادن ۸ وزیر بر روی صفحه شطرنج به صورتی است که هیچ وزیری نتواند به دیگری حمله کند.دو نوع بیان وجود دارد 1بیان افزایشی(حالت اولیه خالی) جایگزین یکی یکی وزیرها و دیگری بیان 2وضعیت کامل(حالت اولیه همه هستند) که شروع با ۸ وزیر و حرکت آنها

حالتها: هر ترتیبی از ۰ تا ۸ وزیر در صفحه، یک حالت است
حالت اولیه: هیچ وزیری در صفحه نیست
تابع پسین: وزیری را به خانه خالی اضافه میکند
آزمون هدف: ۸وزیر در صفحه وجود دارند و هیچ کدام به یکدیگر گارد نمیگیرند
در این فرمول بندی باید ۳*۱۰^۱۴ دنباله ممکن بررسی میشود

حالتها: چیدمان n وزیر $(0 \le n \le 8)$، بطوریکه در هر ستون از n ستون سمت چپ، یک وزیر قرار گیرد و هیچ دو وزیری سطری بهم گارد نگیرند
حالت اولیه: با ۸ وزیر در صفحه شروع میشود
تابع پسین: وزیری را در سمت چپ ترین ستون خالی قرار میدهد، بطوری که هیچ وزیری آن را گارد ندهد
آزمون هدف: ۸وزیر در صفحه بدون گارد

مثال مسئله دنیای واقعی

13

مسائل مسیریابی:

الگوريتم‌هاي مسير يابي كاربردهاي زيادي دراند، مانند مسيريابي در شبكه‌هاي كامپيوتري، سيستم‌هاي خودكار مسافرتي و سيستم‌هاي برنامه‌نويسي مسافرتي هوايي.و تنها رسيدن به مقصد مهم است

مسائل تورینگ:

شبيه به مسائل مسيريابي هستند با اين تفاوت كه عامل در هر حالت بايست ليست تمام **حالات قبلي** را داشته باشد (مثل سفر به همه شهرها...) مثل مساله فروشنده دوره گرد(TSP)

مسئله فروشنده دوره گرد مسئله مشهوری است که در آن هر شهر حداقل یکبار باید ملاقات شود هدف یافتن کوتاهترین مسیر است. علاوه بر مکان عامل، هر حالت بايد مجموعه شهرهايي را كه عامل ملاقات كرده، نگه دارد. علاوه بر برنامه‌ريزي سفر براي فروشنده دوره‌گرد، اين الگوريتم‌ها براي اعمالي نظير برنامه‌ريزي حركات مته خوردكار سوراخ‌كننده برد مدار استفاده مي‌شود.

چیدمان: VISI ابزار طراحی کمکی کامپیوتری با هدف طراحی مداری روی تراشه است که کمترین مساحت و طول اتصالات و بیشترین سرعت را داشته باشد، قرار دادن سلول‌ها روي تراشه به گونه‌اي است كه آنها روي هم قرار نگيرند و بنابراين فضايي نيز براي سيم‌هاي ارتباطي وجود دارد كه بايد بين سلول‌ها قرار گيرند. وظايف مهم *چیدمان سلول* و *کانال یابی* است كه بعد از اينكه ارتباطات و اتصالات مدار كامل شد، اين دو قسمت انجام مي‌شوند.

هدایت ربات:

تعميم مساله مسيريابی است يک ربات مي‌تواند در يک فضاي پيوسته با يک مجموعه نامحدودي از حالات و عمليات ممكن حركت كند. ربات‌هاي واقعي بايد قابليت تصحيح اشتباهات را در خواندن حسگرها و كنترل موتور داشته باشند.

تعیین خودکار ترتیب مونتاژ:

هدف پيدا كردن ترتيب مونتاژ قطعات يك شي است (مثال مادبرد)

بررسی گام ۳: جستجو

14

- پس از تدوین مسئله آن را با جستجو در فضای حالت حل می کنیم.
- برای این کار از درخت جستجو استفاده می کنیم.
- درخت جستجو به وسیله حالت اولیه و تابع پسین ایجاد می شود که فضای حالت را تشکیل می دهد.
- ریشه درخت جستجو ، حالت اولیه است.
- ابتدا ریشه را بررسی می کنیم ببینیم حالت هدف است یا نه .اگر حالت هدف بود،مسئله حل می شود .در غیر این صورت حالت فعلی را بسط می دهیم.
- یعنی تابع پسین را به حالت فعلی اعمال کرده مجموعه جدیدی از حالت ها را تولید می کنیم.
- انتخاب کردن ، آزمون و بسط دادن را ادامه می دهیم تا راه حلی پیدا شود یا حالتی برای بسط وجود نداشته باشد.
- انتخاب حالت برای بسط توسط راهبرد جستجو تعیین می شود.

راهبردهای جستجو

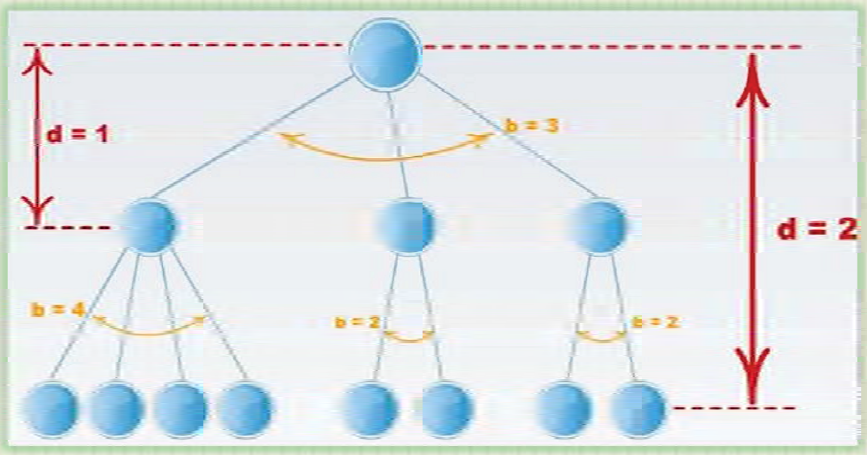
15

- یک راهبرد جستجو بر اساس ترتیب بسط گره های درخت جستجو تعیین می شود.
- اندازه گیری کارایی الگوریتم ها و راهبرد های جستجو:
  - کامل بودن: آیا الگوریتم تضمین می کند که در صورت وجود راه حل، آن را بیابد
  - بهینگی : آیا الگوریتم راه حل بهینه ، با کمترین هزینه را تولید می کند:
  - پیچیدگی زمانی :زمان مورد نیاز برای یافتن جواب
  - پیچیدگی فضایی:حافظه مورد نیاز برای یافتن جواب

راهبردهای جستجو

16

- پیچیدگی های زمانی و فضایی بر حسب سه کمیت زیر بیان می شود:
  - b :حد اکثر فاکتور انشعاب درخت جستجو(حداکثر پسینهای یک گره)
  - d : عمق راه حل با کمترین هزینه(کم عمق ترین گره هدف)
  - m: حداکثر عمق فضای حالت(طولانی ترین مسیر در فضای حالت)



الگوریتم های جستجوی ناآگاهانه(کور)

17

اگر به الگوریتمی هیچ اطلاعاتی به جز تعریف مسئله داده نشود،آن الگوریتم را ناآگاه می گوییم. و تنها قادر به تولید پسینها و تشخیص حالت هدف از حالت غیر هدفند

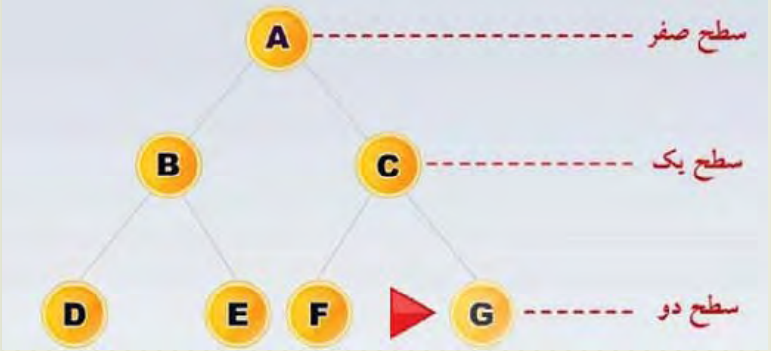
- Breadth-first search: جستجوی اول سطح
- Uniform-cost search :جستجوی هزینه یکنواخت
- Depth-first search :جستجوی اول عمق
- Depth-limited search :جستجوی عمق محدود
- Iterative deepening search : جستجوی عمیق شونده تکراری
- Bidirectional search :جستجوی دو طرفه

جستجوی اول سطح (BF-search)

18

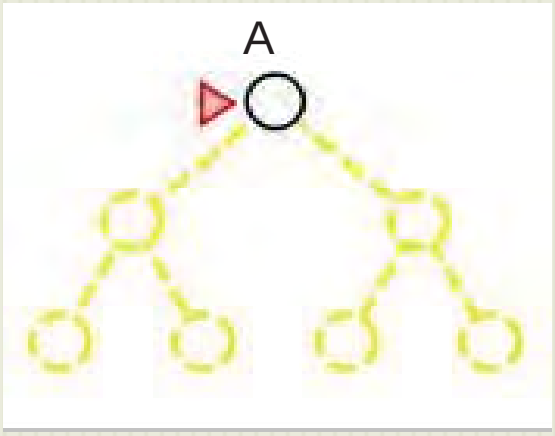
جستجوی اول سطح یا جستجوی عرضی (BF-search)

- بسط ریشه سپس تمام پسینهای ریشه (سطح بعدی) و .....
- پیاده سازی با صف FIFO، گره ای که زودتر بیاید زودتر بسط داده می شود



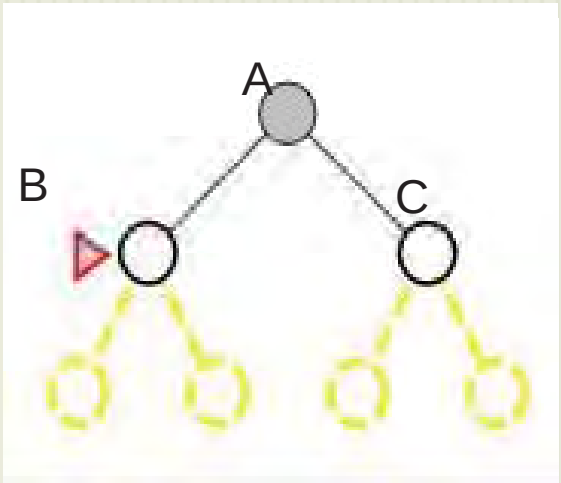
جستجوی اول سطح (BF-search)

- بسط ریشه سپس تمام پسینهای ریشه (سطح بعدی) و .....
- پیاده سازی با صف FIFO، گره ای که زودتر بیاید زودتر بسط داده می شود



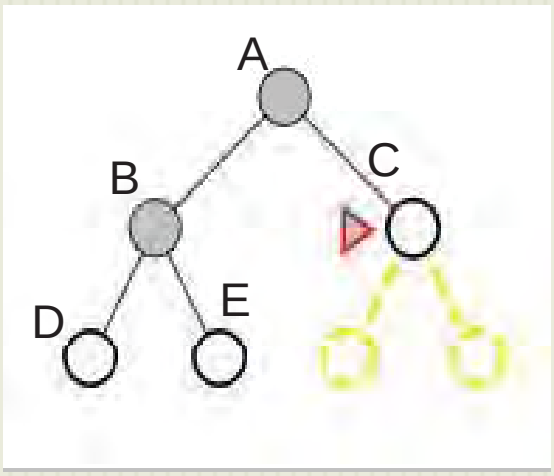
... جستجوی اول سطح (BF-search)

- بسط ریشه سپس تمام پسینهای ریشه (سطح بعدی) و .....
- پیاده سازی با صف FIFO، گره ای که زودتر بیاید زودتر بسط داده می شود



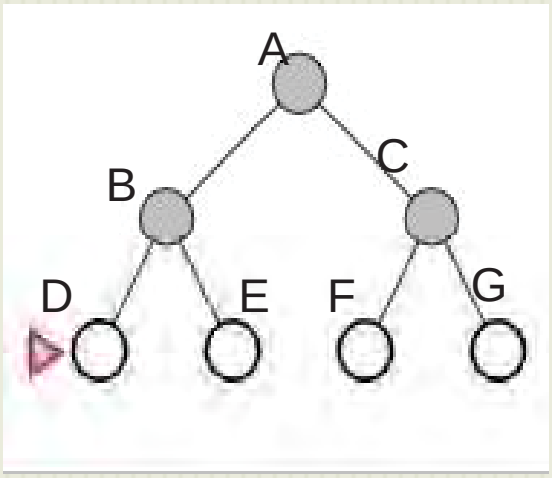
... جستجوی اول سطح (BF-search)

- بسط ریشه سپس تمام پسینهای ریشه (سطح بعدی) و .....
- پیاده سازی با صف FIFO، گره ای که زودتر بیاید زودتر بسط داده می شود



... جستجوی اول سطح (BF-search)

- بسط ریشه سپس تمام پسینهای ریشه (سطح بعدی) و .....
- پیاده سازی با صف FIFO، گره ای که زودتر بیاید زودتر بسط داده می شود



ارزیابی الگوریتم جستجوی اول سطح (BF-search)

23

- کامل بودن :
  - کامل است اگر فاکتور انشعاب آن متناهی باشد.
- پیچیدگی زمانی :
  - در سطح n ام b^n گره تولید می شود در سطح ۲ با فاکتور انشعاب ۲ = ۴ گره
  - تعداد کل گره های تولید شده تا عمق d +1

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

هدف گسترش نمی یابد

- پیچیدگی فضایی :
  - مانند پیچیدگی زمانی است زیرا هر گرهی که تولید می شود باید در حافظه باقی بماند.

$O(b^{d+1})$

- بهینگی :
  - بهینه است اگر هزینه های مراحل یکسان باشد.

روش جستجوی هزینه یکنواخت (Uniform-cost search)

24

- توسعه جستجوی اول سطح(حل مشکل بهینگی فقط زمان هزینه یکسان)
- بسط گرهی که برای رسیدن به آن کمترین مقدار هزینه شده باشد
- پیاده سازی با صف FIFO اولویت بر حسب هزینه مسیر
- اگر تمامی هزینه های مراحل یکسان باشد ، مانند جستجوی اول سطح خواهد بود.

... جستجوی هزینه یکنواخت (Uniform-cost search)

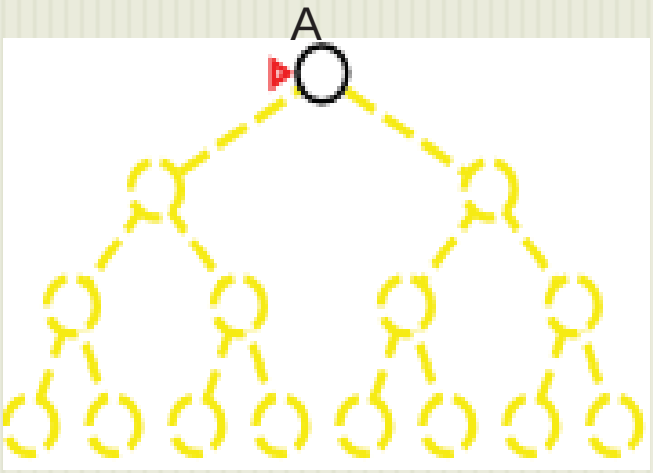
- کامل بودن:
  - کامل است اگر هزینه هر مرحله ، بزرگتر یا مساوی یک مقدار ثابت کوچک مثبت مانند ε باشد.چون اگر هزینه صفر باشد ، جستجو در یک حلقه بی نهایت گیر می کند
- پیچیدگی زمانی و فضایی:
  - C^*: هزینه راه حل بهینه
  - ε : حداقل هزینه هر فعالیت

$$O(b^{C^*/\varepsilon})$$

- بهینگی :
  - بهینه است.

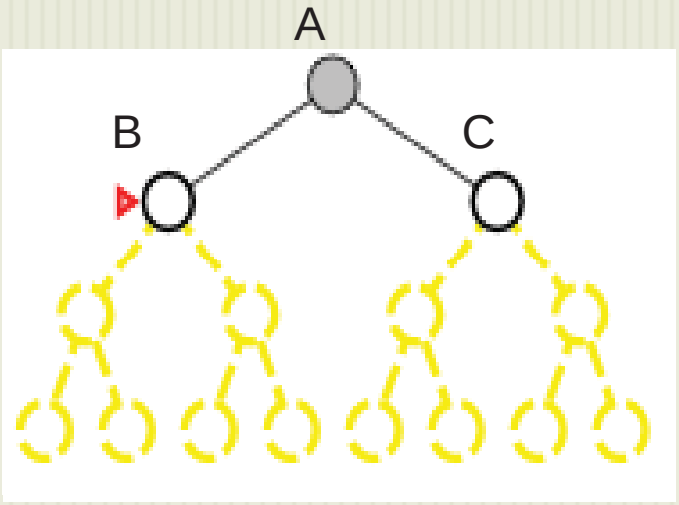
جستجوی اول عمق (DF-search)

- بسط عمیق ترین گره بسط داده نشده
- پیاده سازی با پشته LIFO، گره ای که دیرتربیاید زودتر بسط داده می شود



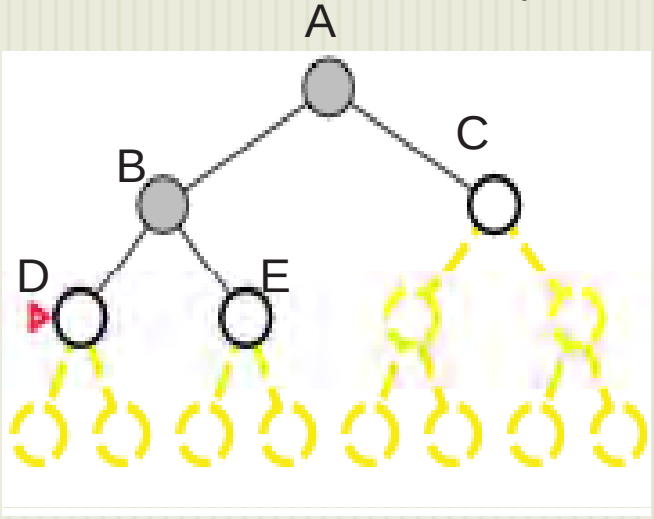
... جستجوی اول عمق (DF-search)

- بسط عمیق ترین گره بسط داده نشده
- پیاده سازی با پشته LIFO، گره ای که دیرتربیاید زودتر بسط داده می شود



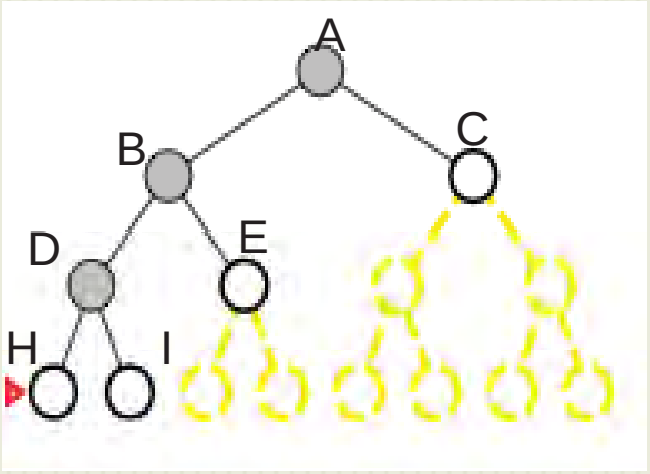
... جستجوی اول عمق (DF-search)

- بسط عمیق ترین گره بسط داده نشده
- پیاده سازی با پشته LIFO، گره ای که دیرتربیاید زودتر بسط داده می شود



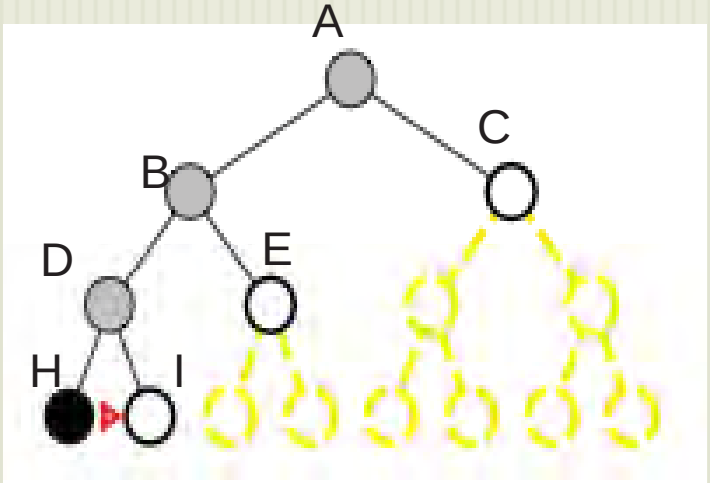
جستجوی اول عمق (DF-search) ...

- بسط عمیق ترین گره بسط داده نشده
- پیاده سازی با پشته LIFO، گره ای که دیرتربیاید زودتر بسط داده می شود



جستجوی اول عمق (DF-search) ...

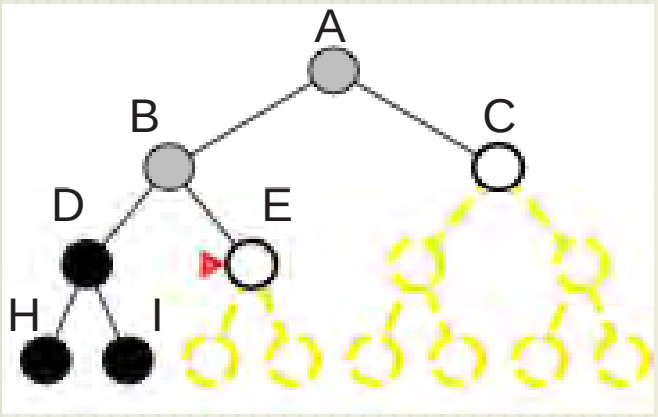
- بسط عمیق ترین گره بسط داده نشده
- پیاده سازی با پشته LIFO، گره ای که دیرتربیاید زودتر بسط داده می شود



جستجوی اول عمق (DF-search) ...

31

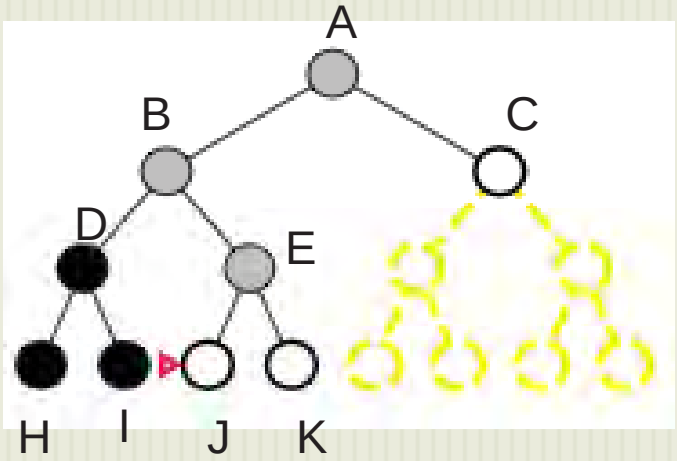
- بسط عمیق ترین گره بسط داده نشده
- پیاده سازی با پشته LIFO، گره ای که دیرتربیاید زودتر بسط داده می شود



جستجوی اول عمق (DF-search) ...

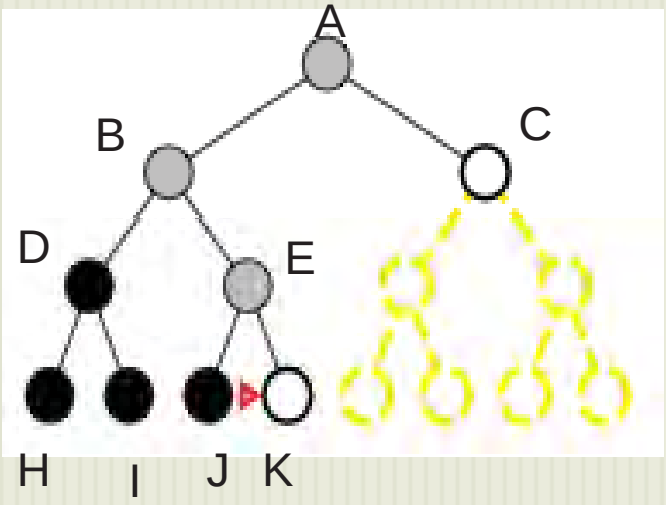
32

- بسط عمیق ترین گره بسط داده نشده
- پیاده سازی با پشته LIFO، گره ای که دیرتربیاید زودتر بسط داده می شود



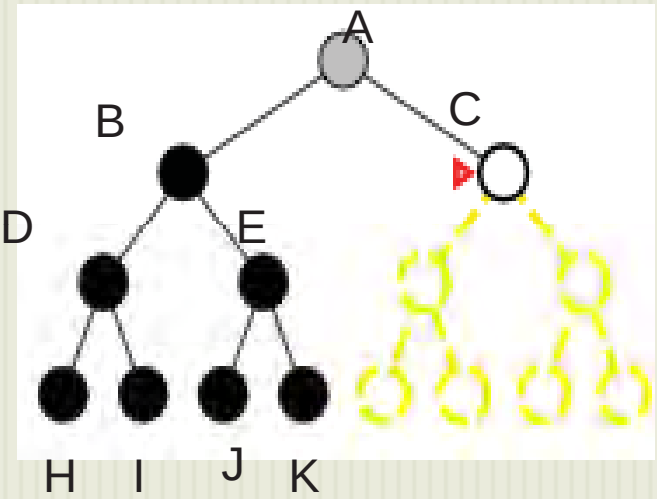
جستجوی اول عمق (DF-search) ...

- بسط عمیق ترین گره بسط داده نشده
- پیاده سازی با پشته LIFO، گره ای که دیرتربیاید زودتر بسط داده می شود



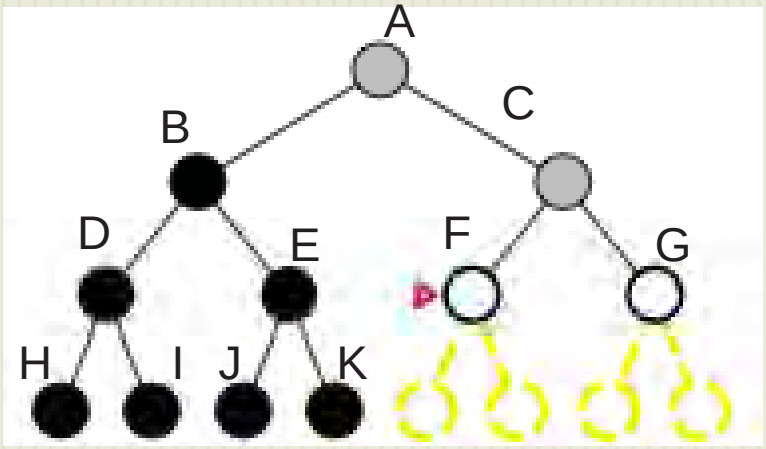
جستجوی اول عمق (DF-search) ...

- بسط عمیق ترین گره بسط داده نشده
- پیاده سازی با پشته LIFO، گره ای که دیرتربیاید زودتر بسط داده می شود



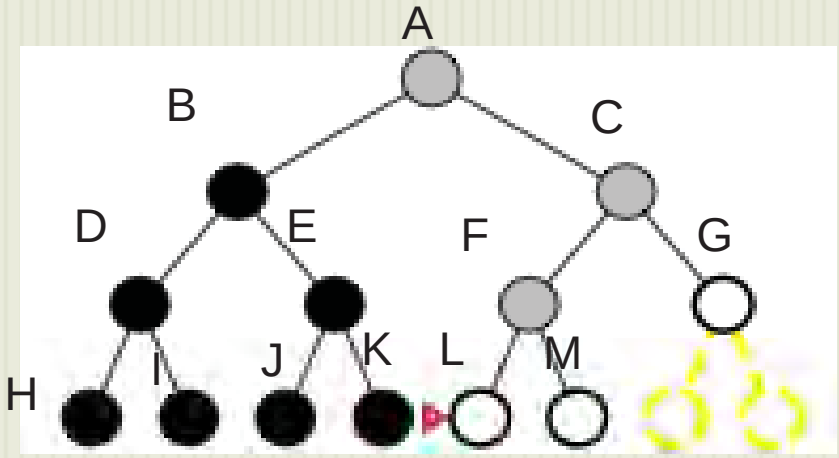
جستجوی اول عمق (DF-search) ...

- بسط عمیق ترین گره بسط داده نشده
- پیاده سازی با پشته LIFO، گره ای که دیرتربیاید زودتر بسط داده می شود



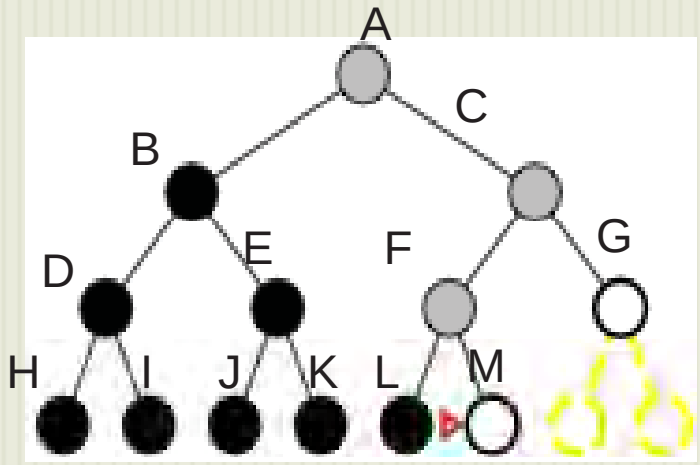
جستجوی اول عمق (DF-search) ...

- بسط عمیق ترین گره بسط داده نشده
- پیاده سازی با پشته LIFO، گره ای که دیرتربیاید زودتر بسط داده می شود



... جستجوی اول عمق (DF-search)

- بسط عمیق ترین گره بسط داده نشده
- پیاده سازی با پشته LIFO، گره ای که دیرتربیاید زودتر بسط داده می شود

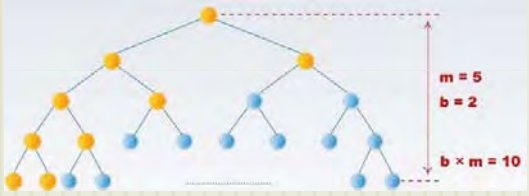


ارزیابی جستجوی اول عمق (DF-search)

- کامل بودن :
  - کامل نیست مگر اینکه فضای حالت متناهی باشد.در حالت نامتناهی گیر می کند
- پیچیدگی زمانی:

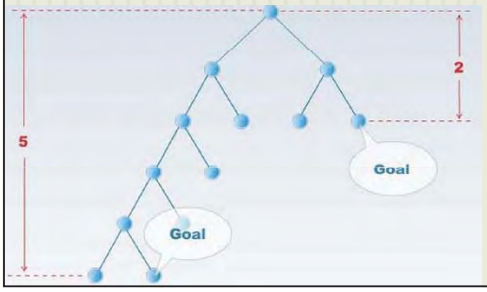
$$O(b^m)$$

- پیچیدگی فضایی:
  - چون گره های ملاقات شده بلافاصله بسط داده می شوند نیازی به حافظه بزرگ نیست $O(bm+1)$



- در جستجوی پس گرد به خاطر حفظ گره نیمه گسترش یافته شده حافظه کمتری o(bm) نیاز است

- بهینگی:
  - بهینه نیست.چون ممکن است در مسیر نزدیکتری موجود باشد



جستجوی عمق محدود (Depth-limited search)

39

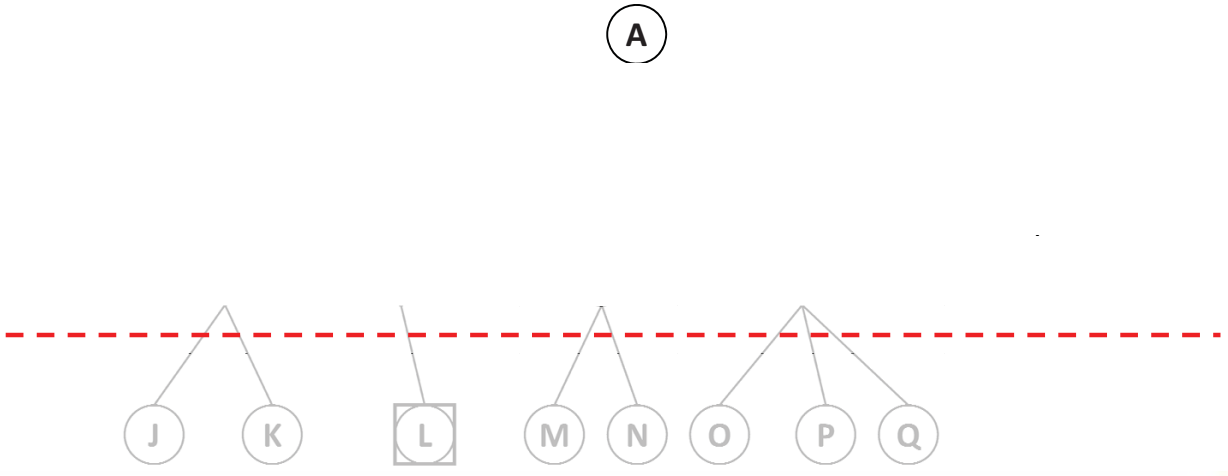
- جستجوی اول عمق با محدودیت عمق L
  - این استراتژی، برای رهایی از دامی که جستجوی عمقی در آن گرفتار می‌شد، از یک برش استفاده می‌کند.
  - با گره های سطح L طوری رفتار می شود که گویی فاقد پسین (فرزند)هستند.
- If L < d آنگاه کامل نیست.
- If L > d آنگاه بهینه نیست.(جستجوی بیشتر)
- پیچیدگی زمانی: $O(b^l)$
- پیچیدگی فضایی: $O(bl)$

مثال جستجوی عمق محدود (Depth-limited search)

40

جستجوی عمقی محدود

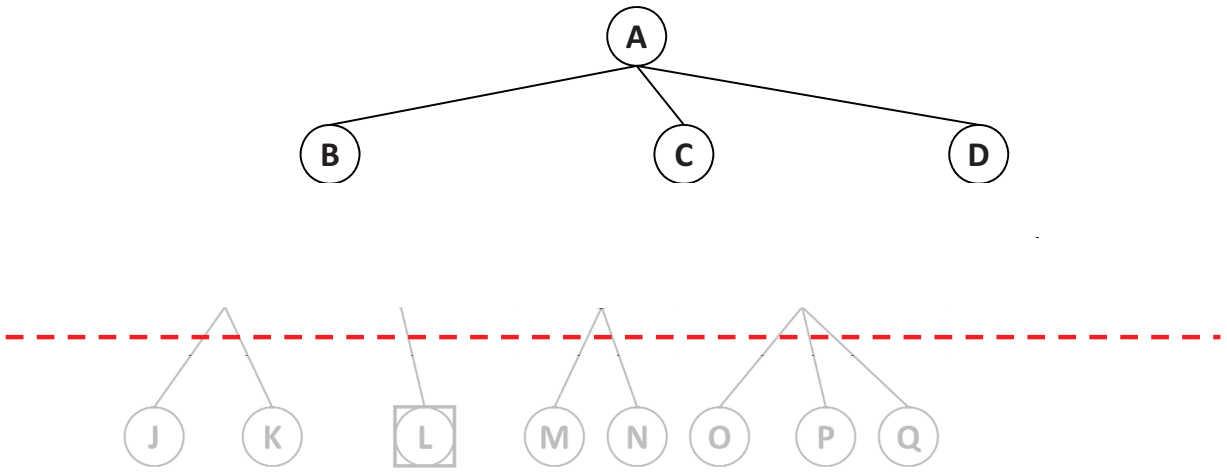
مسئله درختهای نامحدود میتواند به وسیله جست و جوی عمقی با عمق بهبود یابد Lمحدود



مثال جستجوی عمق محدود (Depth-limited search)

جستجوی عمقی محدود

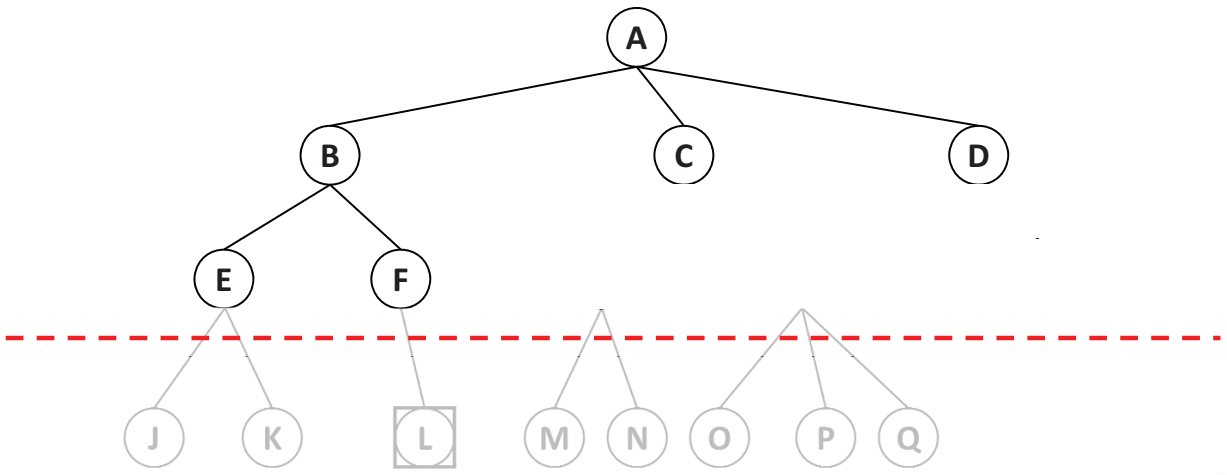
مسئله درختهای نامحدود میتواند به وسیله جست و جوی عمقی با عمق بهبود یابد نامحدود



مثال جستجوی عمق محدود (Depth-limited search)

جستجوی عمقی محدود

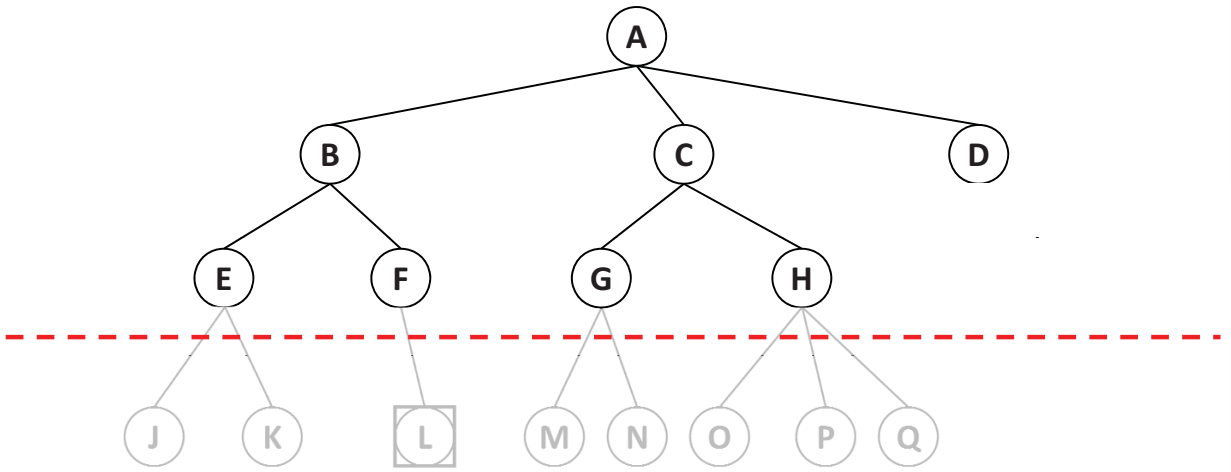
مسئله درختهای نامحدود میتواند به وسیله جست و جوی عمقی با عمق بهبود یابد نامحدود



مثال جستجوی عمق محدود (Depth-limited search)

جستجوی عمقی محدود

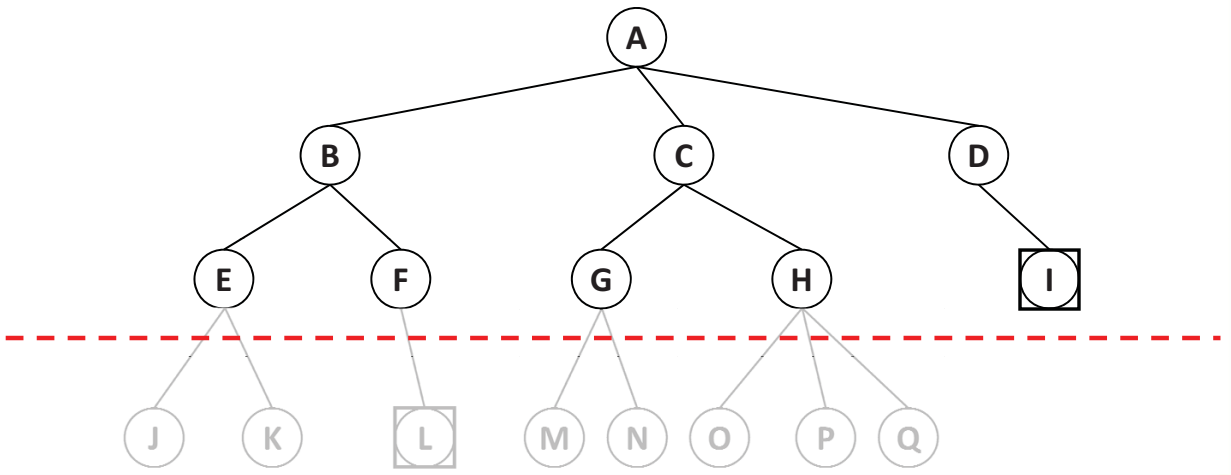
مسئله درختهای نامحدود میتواند به وسیله جست و جوی عمقی با عمق بهبود یابد ـامحدود



مثال جستجوی عمق محدود (Depth-limited search)

جستجوی عمقی محدود

مسئله درختهای نامحدود میتواند به وسیله جست و جوی عمقی با عمق بهبود یابد ـامحدود



جستجوی عمیق‌کننده تکراری:

جستجوی عمیق‌کننده تکراری:

قسمت دشوار جستجوی عمقی محدود شده، انتخاب یک محدودهٔ خوب است.

اگر محدودهٔ عمق بهتری را پیدا کنیم، این محدوده، ما را به سوی جستجوی کاراتری سوق می‌دهد.

اما برای بیشتر مسائل، محدودهٔ عمقی مناسب را تا زمانی که مسئله حل نشده است، نمی‌شناسیم.

جستجوی عمیق‌کنندهٔ تکراری استراتژی است که نظریه انتخاب بهترین محدودهٔ عمقی، توسط امتحان تمام محدودهٔ مسیرهای ممکن را یادآوری می‌کند.

و مزایای جستجوی اول سطح و اول عمق را یکجا دارد

این جستجو مانند جستجوی سطحی کامل و بهینه است، اما فقط مزیت درخواست حافظه اندک را از جستجوی عمقی دارد.

جستجوی عمیق شونده تکراری (ID search)

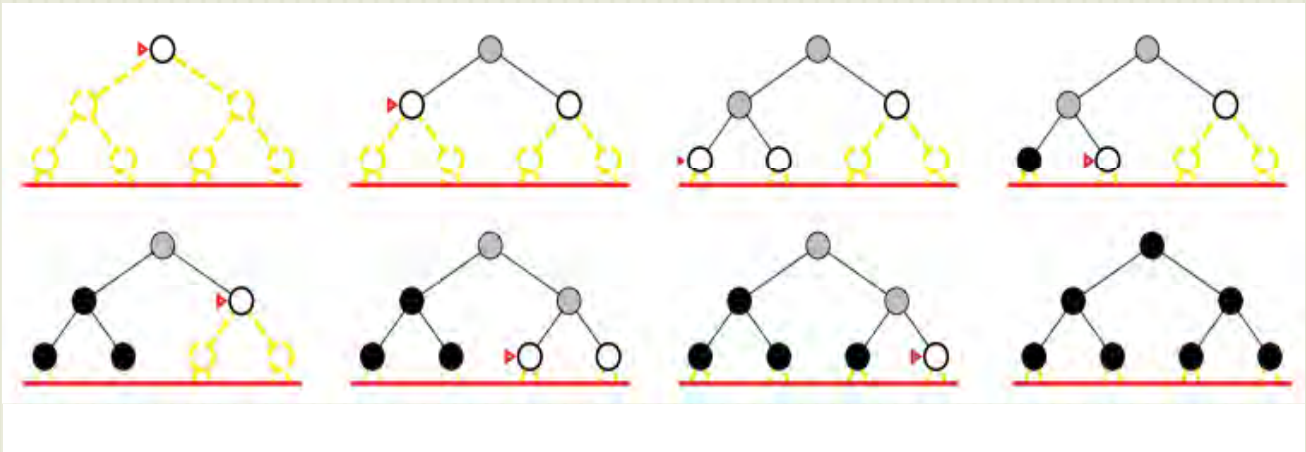
- Limit=0

... جستجوی عمیق شونده تکراری (ID search)

- Limit=1

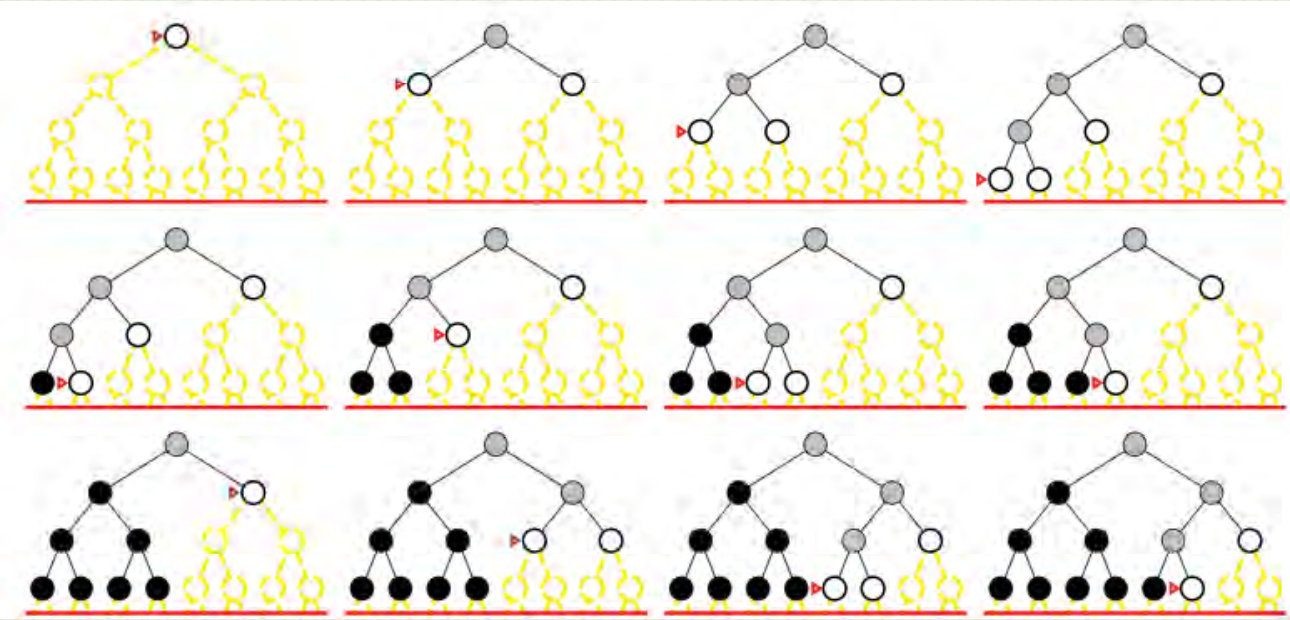
... جستجوی عمیق شونده تکراری (ID search)

- Limit=2



... جستجوی عمیق شونده تکراری (ID search)

- Limit=3



ارزیابی الگوریتم جستجوی عمیق شونده تکراری (ID search)

در جستجوی عمیق‌کننده تکراری، گره‌های سطوح پائینی یک بار بسط داده می‌شوند، آنهایی که یک سطح بالاتر قرار دارند دوبار بسط داده می‌شوند و الی‌آخر تا به ریشه درخت جستجو برسد، که d+1 بار بسط داده می‌شوند.

بنابراین مجموع دفعات بسط در این جستجو عبارتست از:

(d+1)1+(d)b^1+(d-1)b^2+...+3b^d-2+2b^d-1+1b^d

پیچیدگی زمانی این جستجو هنوز O(b^d) است، و پیچیدگی فضا O(bd) است.

در حالت کلی، عمیق‌کننده تکراری، روش جستجوی برتری است؛ زمانی که فضای جستجوی بزرگی وجود دارد و عمق راه حل نیز مجهول است.

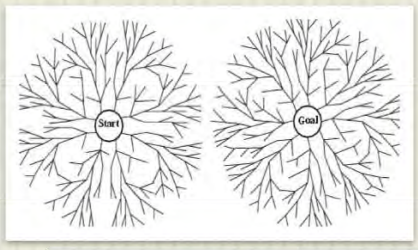
- کامل بودن :
  - YES (no infinite paths)
- پیچیدگی زمانی : $O(b^d)$
- پیچیدگی فضایی : $O(bd)$
- بهینگی :
  - YES if step cost is 1

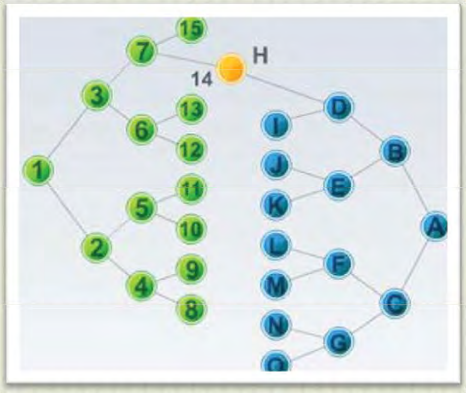
جستجوی دوطرفه

51

انجام دو جست و جوی همزمان، یکی از حالت اولیه به هدف(جلو) و دیگری از هدف به حالت اولیه (عقب) تا زمانی که دو جست و جو به هم برسند(Pred(n)=Succ(n))

در صورت وجود چند هدف کار جستجو سخت می شود





کامل بودن: بله
اگر هر دو جستجو، عرضی باشند و هزینه تمام مراحل یکسان باشد

بهینگی: بله
اگر هر دو جستجو، عرضی باشند و هزینه تمام مراحل یکسان باشد

پیچیدگی زمانی: $O(b^{d/2})$

پیچیدگی فضا: $O(b^{d/2})$

مقایسه راهبردهای جستوی ناآگاهانه

52

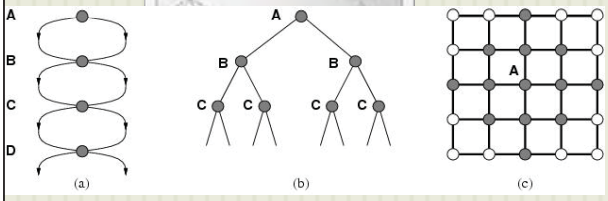
Criterion	Breadth-First	Uniform-cost	Depth-First	Depth-limited	Iterative deepening	Bidirectional search
Complete ?	YES*	YES*	NO	YES, if $l \geq d$	YES	YES*
Time	b^{d+1}	$b^{C^*/e}$	b^m	b^l	b^d	$b^{d/2}$
Space	b^{d+1}	$b^{C^*/e}$	bm	bl	bd	$b^{d/2}$
Optimal?	YES*	YES*	NO	NO	YES	YES

اجتناب از حالتهای تکراری

53

برای مسائل زیادی، حالات تکراری غیرقابل اجتناب هستند. این شامل تمام مسائلی می‌شود که عملگرها قابل وارونه شدن باشند، مانند مسائل مسیریابی و کشیش‌ها و آدمخوارها.

وجود حالتهای تکراری در یک مسئله قابل حل، میتواند آن را به مسئله غیر قابل حل تبدیل کند



سه راه برای حل مشکل حالات تکراری برای مقابله با افزایش مرتبه و سرریزی فشار کار کامپیوتر وجود دارد:

1. به حالتی که هم اکنون از آن آمده‌اید، برنگردید.
2. از ایجاد مسیرهای دوار بپرهیزید.
3. حالتی را که قبلاً تولید شده است، مجدداً تولید نکنید. این مسئله باعث می‌شود که هر حالت در حافظه نگهداری شود، پیچیدگی فضایی O(bd) داشته باشد. بهتر است که به O(s) توجه کنید که s تعداد کل حالات در فضای حالت ورودی است.الگوریتمهایی که گذشته خود را فراموش می کنند محکوم به شکستند.

جستجو با اطلاعات ناقص

54

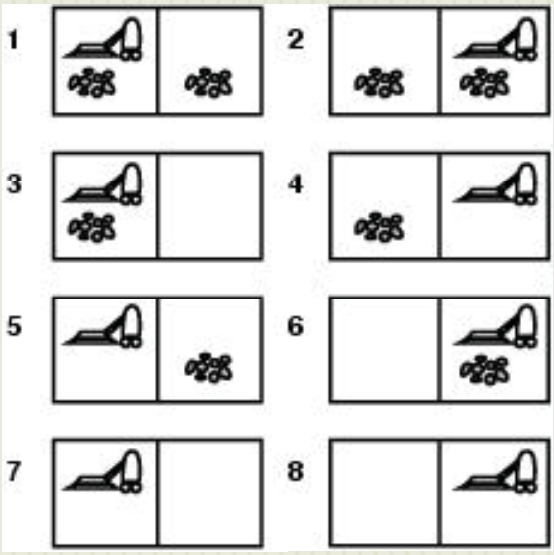
مسئله های فاقد حسگر: اگر عامل فاقد حسگر باشد، میتواند در یکی از چند حالت اولیه باشد و هر فعالیت میتواند آن را به یکی از چند حالت جانشین ببرد، لذا از فضای حالت فرضی(حالت باور) استدلال می شود و جستجو ها نیز در فضای حالت باور است نه فضای حالت فیزیکی

مسئله های اقتضایی(عدم قطعیت): اگر محیط به طور جزئی قابل مشاهده باشد یا اگر فعالیتها قطعی نباشد، ادراکات عامل، پس از هر عمل، اطلاعات جدیدی را تهیه میکنند. هر ادراک ممکن، اقتضایی را تعریف میکند که باید برای آن برنامه ریزی شود

- **مسائل خصمانه**: اگرعدم قطعیت در اثر فعالیتهای عامل دیگری بوجود آید، مسئله را خصمانه گویند

مسئله های اکتشافی(عدم شناخت محیط): وقتی حالتها و فعالیتهای محیط ناشناخته باشند، عامل باید سعی کند آنها را کشف کند. مسئله های اکتشافی را میتوان شکل نهایی مسئله های اقتضایی دانست

مثال : جاروبرقی فاقد حسگر



- عامل جارو تمام اثرات فعالیتهایش را میداند اما فاقد حسگر است.
- حالت اولیه آن یکی از اعضای مجموعه $\{1،2،3،4،5،6،7،8\}$ میباشد
- فعالیت (Right) $\{2،4،6،8\}$
- فعالیت (Right,Suck) $\{4،8\}$
- فعالیت (Right,Suck,Left,Suck) تضمین میکند که صرف نظر از حالت اولیه، به حالت هدف، یعنی ۷ برسد