هوش مصنوعی

نام مرجع :

Artificial Intelligence A Modern Approach

نویسنده :

استوارت راسل ، پیتر نورویگ

تهیه کننده :

مجتبی پورمحقق

هوش مصنوعي

فصل اول

مقدمه

هوش مصنوعي Artificial Intelligence

فهرست

- هوش مصنوعي چيست؟
- مباني هوش مصنوعي
- تاريخچه هوش مصنوعي

مقدمه

هوش مصنوعي چيست؟

عاقلانه فکر کردن	مانند انسان فکر کردن
عاقلانه عمل کردن	مانند انسان عمل کردن

مقدمه

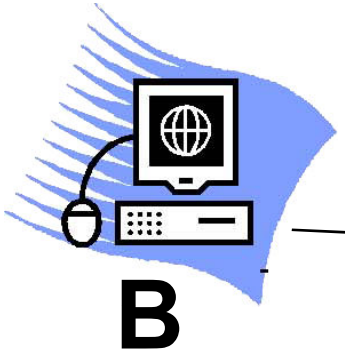
مانند انسان عمل کردن Acting humanly

✓ هنر ساخت ماشينهايي که کارهايي را انجام ميدهند که آن کارها توسط انسان با فکر کردن انجام ميشوند.

✓ مطالعه براي ساخت کامپيوترها براي انجام کارهايي که فعلاً انسان آنها را بهتر انجام ميدهد.

مقدمه (مانند انسان عمل کردن)

تست تورینگ



B

A

کدام انسان است؟
A یا B

مقدمه

مانند انسان فکر کردن Thinking humanly

- تلاش جديد و هيجان انگيز براي ساخت ماشين هايي متفکر و با حس کامل
- خودکارسازي فعاليت هاي مرتبط با تفکر انسان ، فعاليتهايي مثل تصميم گيري ، حل مسئله ، يادگيري

مقدمه

عاقلانه فکر کردن Think rationally

- ✓ مطالعه توانايي هاي ذهني از طريق مدل هاي محاسباتي (منطق گرايي)
- ✓ مطالعه محاسباتي که منجر به درک و استدلال مي شود.

مقدمه

عاقلانه عمل کردن Act rationally

طوري عمل کند که بهترين نتيجه را ارائه دهد

✓ هوش محاسباتي ، مطالعه طراحي عامل هاي هوشمند است

مقدمه

مباني هوش مصنوعي

فلسفه: منطق ، استدلال ، ناشي شدن تفكر از مغز فيزيكي ، مباني يادگيري ، زبان و عقلانيت

روان شناسي: تطبيق ، اثر طبيعي ادراک و تاثير آن بر محيط

رياضيات: نمايش رسمي الگوريتمها ، محاسبات ، تصميم پذيري و تصميم ناپذيري ، احتمال

زبان شناسي: علم ارائه ، گرامر

مقدمه

مباني هوش مصنوعي

نظريه كنترل و سيبرنتيک: تحت كنترل در آوردن محصولات مصنوعي ، ثبات و پايداري ، طراحي عامل بهينه

علوم عصبي: نحوه پردازش اطلاعات توسط مغز

اقتصاد: نظريه تصميمهاي عقلايي ، نظريه بازي

مهندسي كامپيوتر: ساخت كامپيوترهاي سريع

مقدمه

تاريخچه هوش مصنوعي

- 1943 ، مک کولوچ و والتر پيتز: ارايه مدل نرون مصنوعي بيتي(دو حالته) قابل يادگيري به منظور محاسبه هر تابع قابل محاسبه.
- 1950 ، آلن تورينگ اولين بار ديد کاملي از هوش مصنوعي را تحت عنوان "*محاسبات ماشيني و هوشمند*" ارايه نمود.
- 1951 ، هينسکي و ادموندز اولين کامپيوتر شبکه عصبي را طراحی کردند.
- 1952 ، آرتور ساموئل: برنامه اي ساخت که ياد ميگرفت بهتر از نويسنده اش بازي کند ؛ در نتيجه اين تصور را که "کامپيوتر فقط کاري را انجام ميدهد که به آن گفته شود" نقض کرد.

مقدمه (تاريخچه هوش مصنوعي)

- 1956 ،نشست كارگروهي دورتموند: انتخاب نام هوش مصنوعي
- 1959 ، هربرت جلونتر: برنامه(GTP) را ساخت كه قضايا را با اصل موضوعات مشخص ثابت مي كرد.
- 1958 ، جان مک كارتي: تعريف زبان ليسپ كه بهترين زبان هوش مصنوعي شد.
- 1973-1958 ، جيمز اسلاگل: برنامه حل مسايل انتگرالگيري فرم بسته
  - تام ايوانز: برنامه حل مشابهت هاي هندسي
  - دانيل بابروز: برنامه حل مسايل جبري
  - ديويد هافمن: پروژه محدوده بينايي روبات در جهان بلوكها
  - ديويد والتز: سيستم بينايي و انتشار محدود
  - پاتريک ونيستون: نظريه يادگيري

مقدمه (تاریخچه هوش مصنوعي)

(1966-1973) کند شدن مسیر تحقیقات هوش مصنوعی

- پیچیده شدن الگوریتم برنامه های جدید
  - برنامه ترجمه متون
- انجام ناپذیری بسیاری از مسائلی که سعی در حل آنها بود
  - عدم موفقیت اثبات قضایا با مفروضات بیشتر
- بکارگیری بعضی محدودیتها روی ساختارهای اساسی
  - محدودیت نمایش پرسپترون دو ورودی

مقدمه (تاریخچه هوش مصنوعي)

(1969- 1979) سیستم های مبتنی بر دانش

- جست و جوی همه منظوره که سعی بر یادگیری داشت تا پیمودن راه حل کامل
  - مثل برنامه DENDRAL ، بوچانان و همکارانش در سال 1969
    - مزیت برنامه DENDRAL این بود که اولین سیستم پاداش غنی بود
- متدولوژی جدید سیستم خبره
  - مثل سیستم MYCIN که برای تشخیص عفونتهای خونی طراحی شد
    - استفاده از فاکتورهای قطعیت
- افزایش تقاضا برای شِمای نمایش دانش
  - استفاده از منطق در پرولوگ ، استفاده از ایده مینسکی یعنی قابها و ...

مقدمه (تاريخچه هوش مصنوعي)

1980 تا کنون: تبدیل هوش مصنوعی به یک صنعت

1986 تاکنون: برگشت به شبکه های عصبی

1987 تاکنون: هوش مصنوعی به علم تبدیل میشود

1995 تاکنون: ظهور عاملهای هوشمند

هوش مصنوعي

فصل دوم

عاملهاي هوشمند

فهرست

- عامل
- خواص محیطهای وظیفه
- برنامه های عامل

عاملهای هوشمند

دنباله ادراک

سابقه کامل هر چیزی است که عامل تاکنون درک کرده است.

تابع عامل

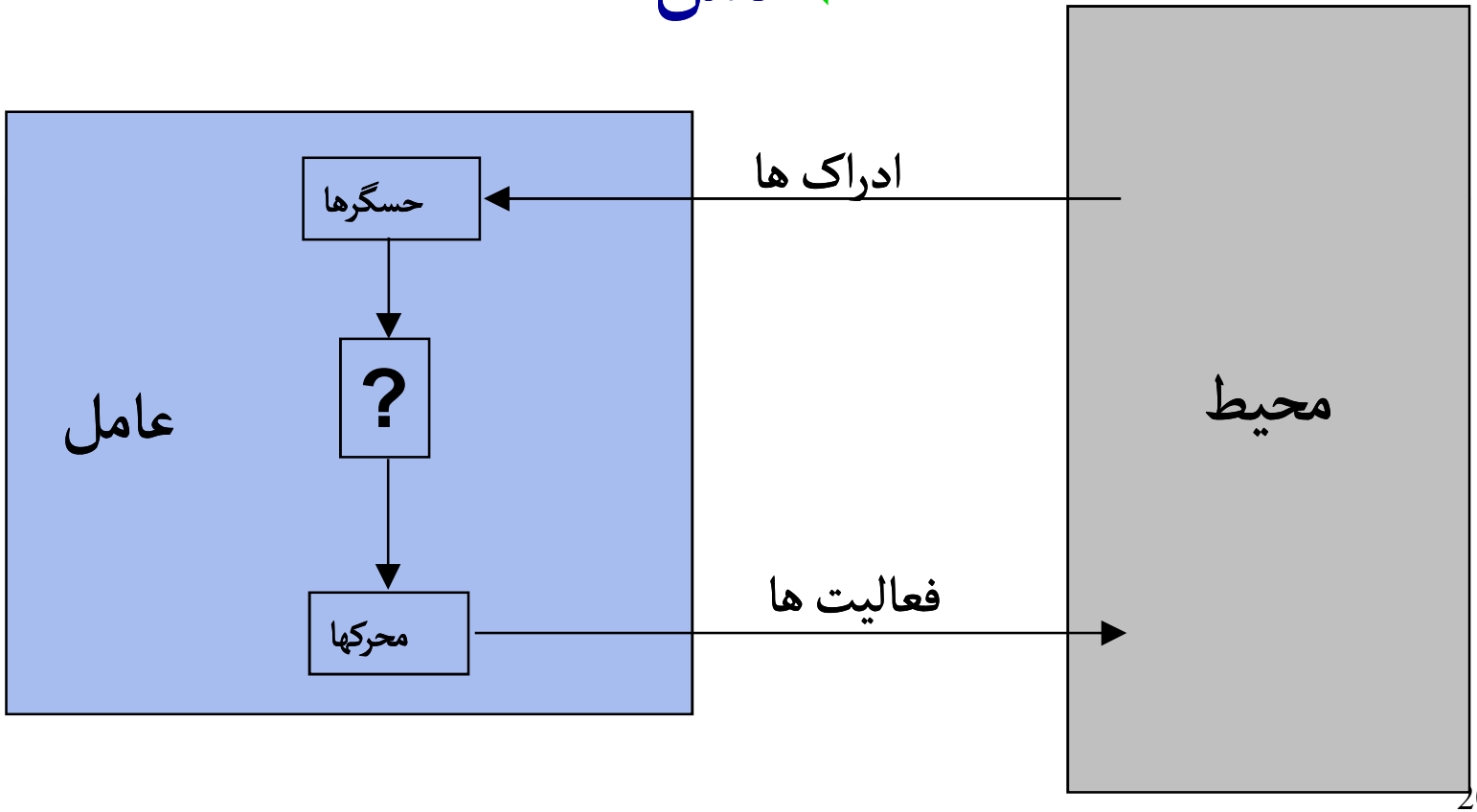
رفتار عامل توسط تابع عامل توصیف میشود که هر دنباله ادراک را به یک فعالیت نقش میکند.

$$f : P^* \rightarrow A$$

دنباله ادراک : تابع عامل ← فعالیت

عاملهای هوشمند

عامل



عاملهای هوشمند

معیارهای کارایي

- معیار کارایي ، معیاری برای موفقیت رفتار عامل است.
  - بر اساس خواسته های فرد در محیط انتخاب میشود

رفتار عقلایي

- معیار کارایي که ملاکهای موفقیت را تعریف میکند
- دانش قبلي عامل نسبت به محیط
- فعالیتهایي که عامل میتواند انجام دهد
- دنباله ادراک عامل در این زمان

عاملهای هوشمند

عامل عالِم (Omni science)

خروجی واقعی فعالیت خود را میداند و میتواند بر اساس آن عمل کند

عامل خردمند (Rational agent)

فعالیتی را انتخاب میکند که معیار کارایي اش را حداکثر میکند

- جمع آوری اطلاعات ، اکتشاف ، یادگیری

عامل خود مختار

نقص دانش قبلی خود را میتواند جبران کند

عاملهای هوشمند

- کاملاً قابل مشاهده درمقابل قابلیت مشاهده جزئی
- قطعي درمقابل غیر قطعی
  - راهبردی
- رویدادی درمقابل ترتیبي
- ایستا درمقابل پویا
- گسسته درمقابل پیوسته
- تک عاملي درمقابل چند عاملي
  - چند عاملي رقابتي درمقابل چندعاملي همیاری

خواص

محیط های وظیفه

عاملهای هوشمند

ساختار عاملها

برنامه + معماری = عامل

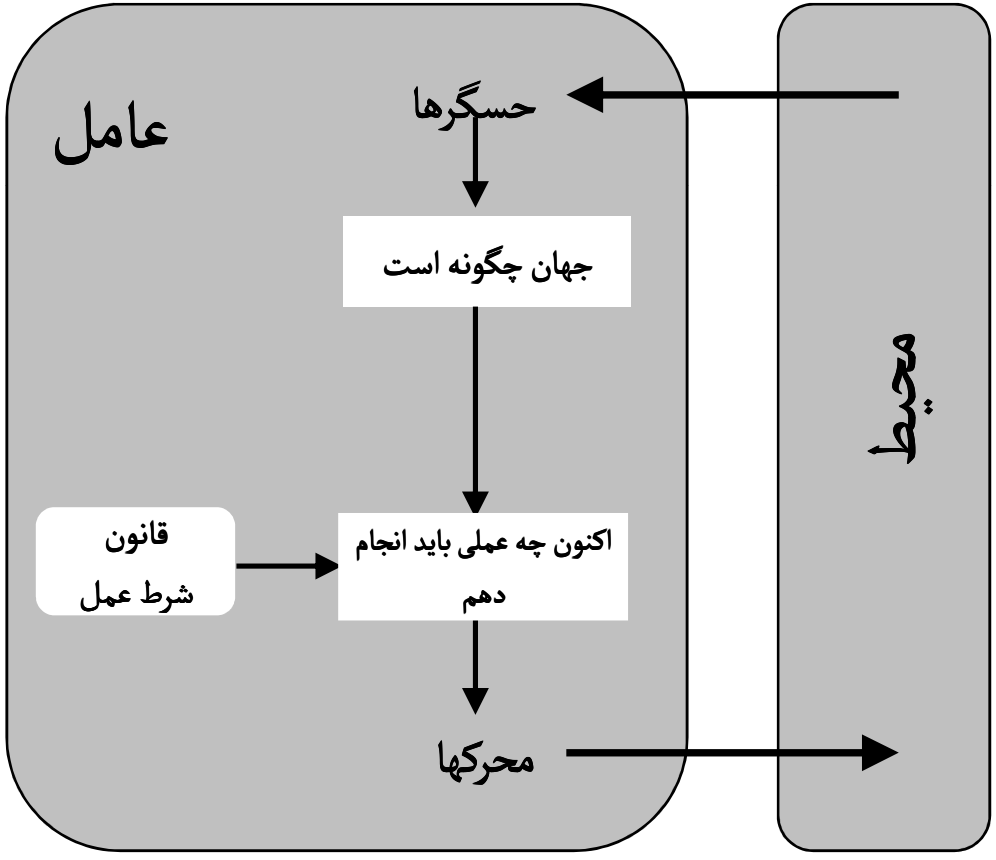
کار هوش مصنوعی طراحی برنامه عامل است که تابع عامل را پیاده سازی میکند

برنامه های عامل

- عاملهای واکنشی ساده
- عاملهای واکنشی مدل گرا
- عاملهای هدف گرا
- عاملهای سودمند

عاملهای هوشمند

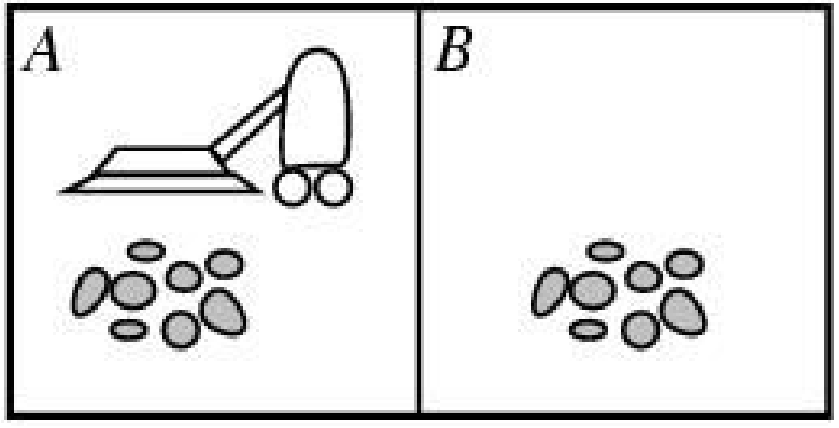
عاملهای واکنشی ساده



- این عاملها فعالیت را بر اساس **درک فعلی** و بدون در نظر گرفتن سابقه ادراک ، انتخاب میکند
- به خاطر **حذف سابقه ادراک** برنامه عامل در مقایسه با جدول آن **بسیار کوچک** است
- انتخاب فعالیت بر اساس یکسری **قوانین موقعیت شرطي** انجام میشود

عاملهای هوشمند

مثالي از عامل واكنشی ساده در دنیای جاروبرقي



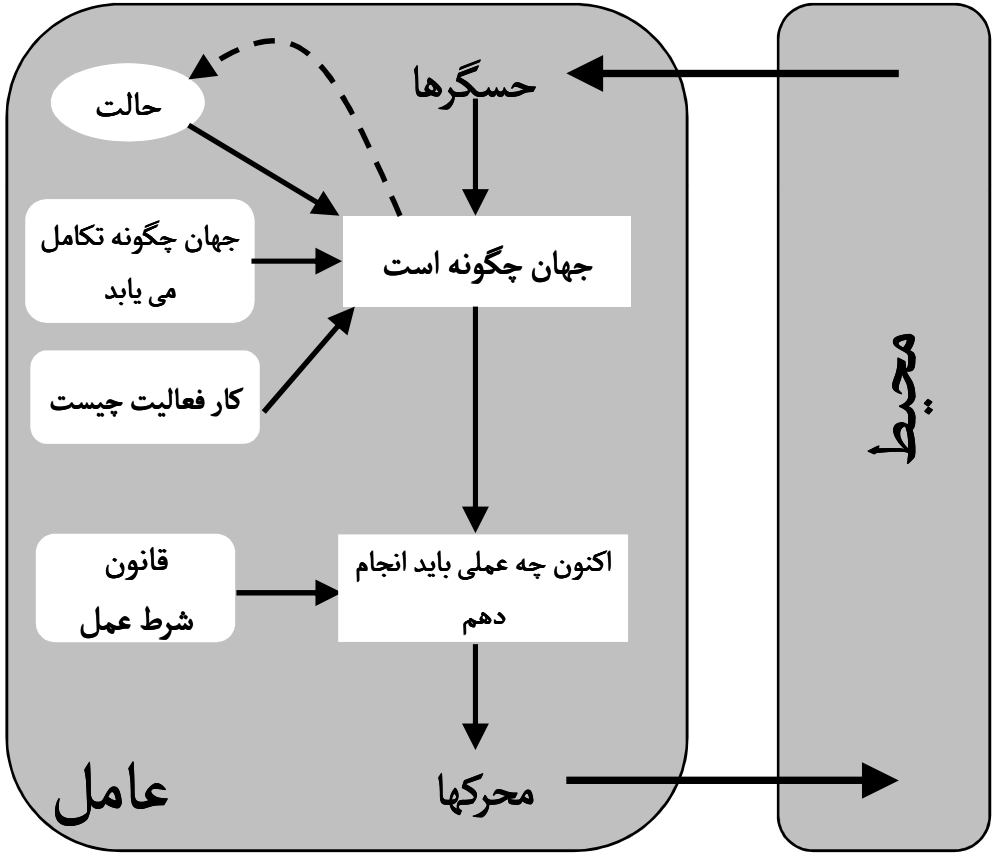
- تصمیم گیری آن بر اساس **مکان فعلی و کثیف بودن** آن مکان صورت میگیرد
- در برنامه عامل در مقایسه با جدول آن ، تعداد حالتهای ممکن از 4 به 4 **کاهش** مي يابد
- انتخاب فعالیت بر اساس **موقعیت** شرطي:

If **dirty** then **suck**

```
function REFLEX-VACUUM-AGENT ([location, status]) return an action
        if status == Dirty then return Suck
        else if location == A then return Right
        else if location == B then return Left
```

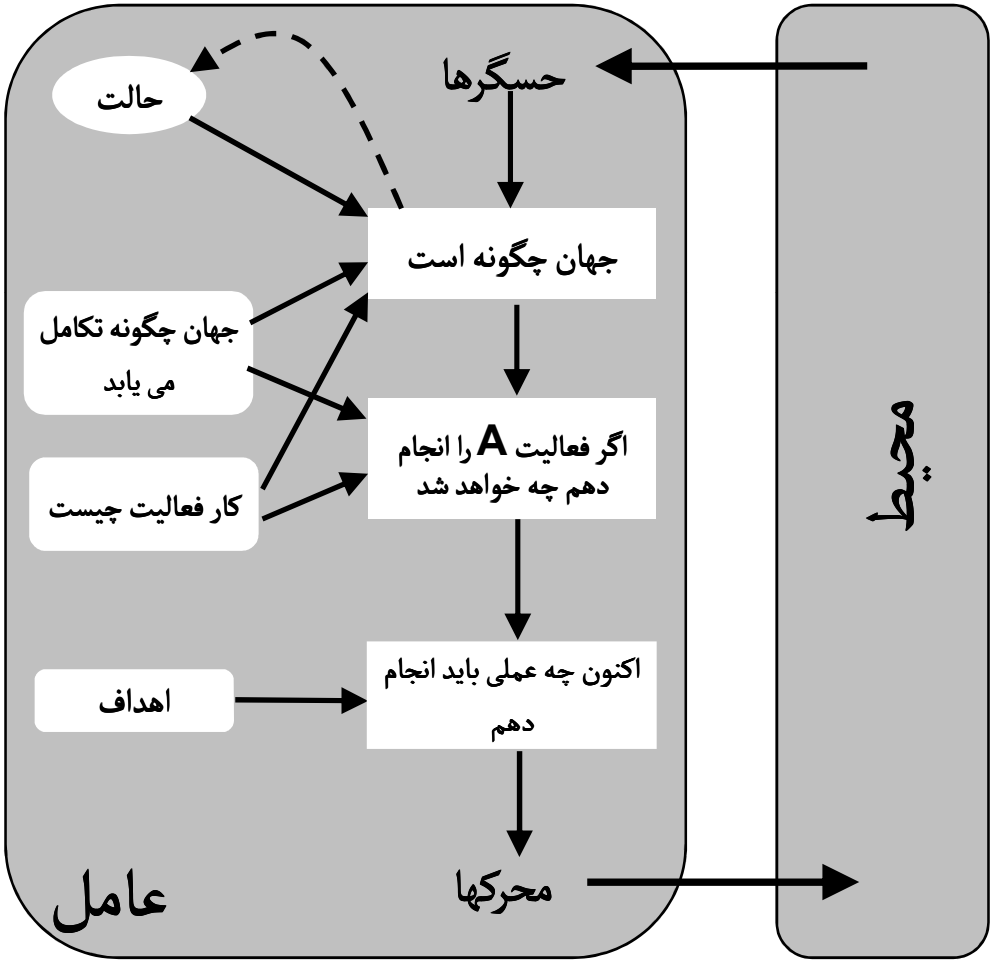
عاملهای هوشمند

عاملهای واکنشي مدل گرا



- استفاده از دانش "**چگونگی عملکرد جهان**" که **مدل** نام دارد
- عامل **بخشي از دنیایي** را که فعلا میبیند **ردیابی** میکند
- عامل باید **حالت داخلي** را ذخیره کند که به **سابقه ادراک** بستگي دارد
- در هر وضعیت, عامل میتواند **توصیف جدیدی** از جهان را کسب کند

عاملهای هوشمند

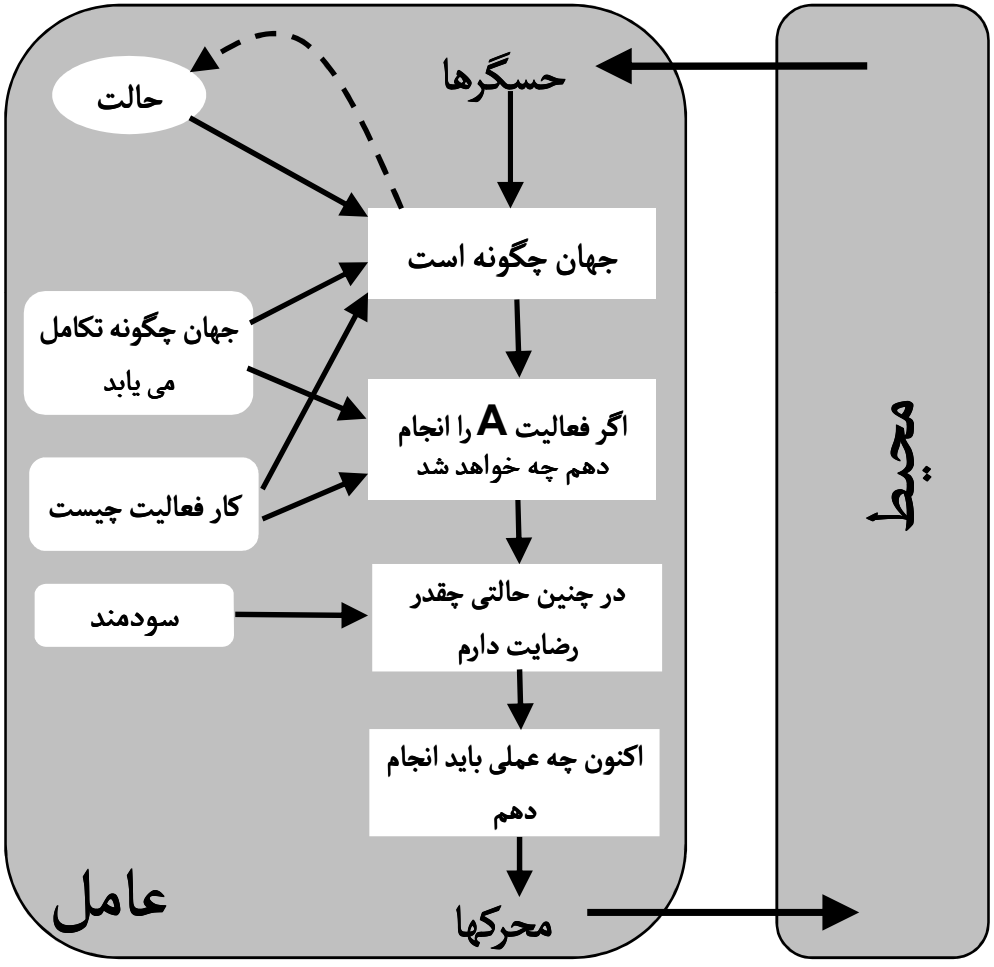


عاملهای هدف گرا

- این عامل علاوه بر توصیف حالت فعلی ، برای انتخاب **موقعیت مطلوب** نیازمند اطلاعات **هدف** نیز میباشد
- **جست و جو و برنامه ریزی** ، دنباله ای از فعالیتها را برای رسیدن عامل به هدف ، پیدا میکند
- این نوع تصمیم گیری همواره **آینده** را در نظر دارد و با قوانین شرط عمل تفاوت دارد
- این نوع عامل کارایی چندانی ندارد ، اما **قابلیت انعطاف** بیشتری دارد

عاملهای هوشمند

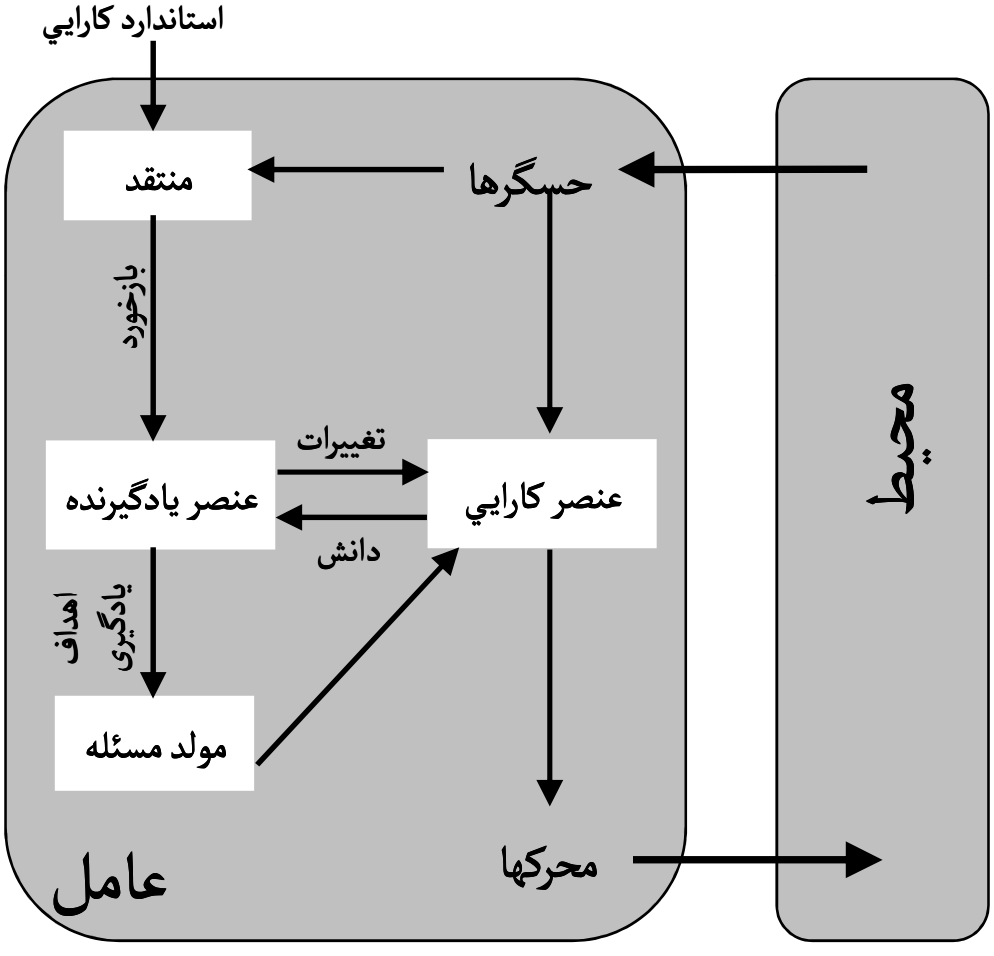
عاملهای سودمند



- این عامل برای **اهداف مشخص** ، **راه های مختلفی** دارد ، که راه حل بهتر برای عامل **سودمندتر** است.
- **تابع سودمندی** ، حالت یا دنباله ای از حالتها را به یک عدد **حقیقی** نگاشت میکند که **درجه رضایت** را توصیف مِیکند.
- وقتی **اهداف متضاد** باشند ، **بعضی** از آنها برآورده میشوند
- اگر **هیچیک از اهداف** به طور قطعی قابل حصول نباشند ، احتمال موفقیت با **اهمیت هدف** مقایسه میشود

عاملهای هوشمند

عاملهای یادگیرنده



- عنصرِ **یادگیرنده** مسئول ایجاد بهبودها
- عنصر **کارایي** مسئول انتخاب فعالیتهای خارجی
- **منتقد** مشخص میکند که یادگیرنده با توجه به استانداردهای کارایي چگونه عمل میکند
- **مولد مسئله** مسئول پیشنهاد فعالیتهایي است که منجر به تجربیات آموزنده جدیدی میشود

هوش مصنوعي

فصل سوم

حل مسئله با جستجو

هوش مصنوعي Artificial Intelligence

فهرست

- عاملهای حل مسئله
- مسئله
- اندازه گیری کارايي حل مسئله
- جستجوی ناآگاهانه
- اجتناب از حالتهای تکراری
- جستجو با اطلاعات ناقص

حل مسئله با جستجو

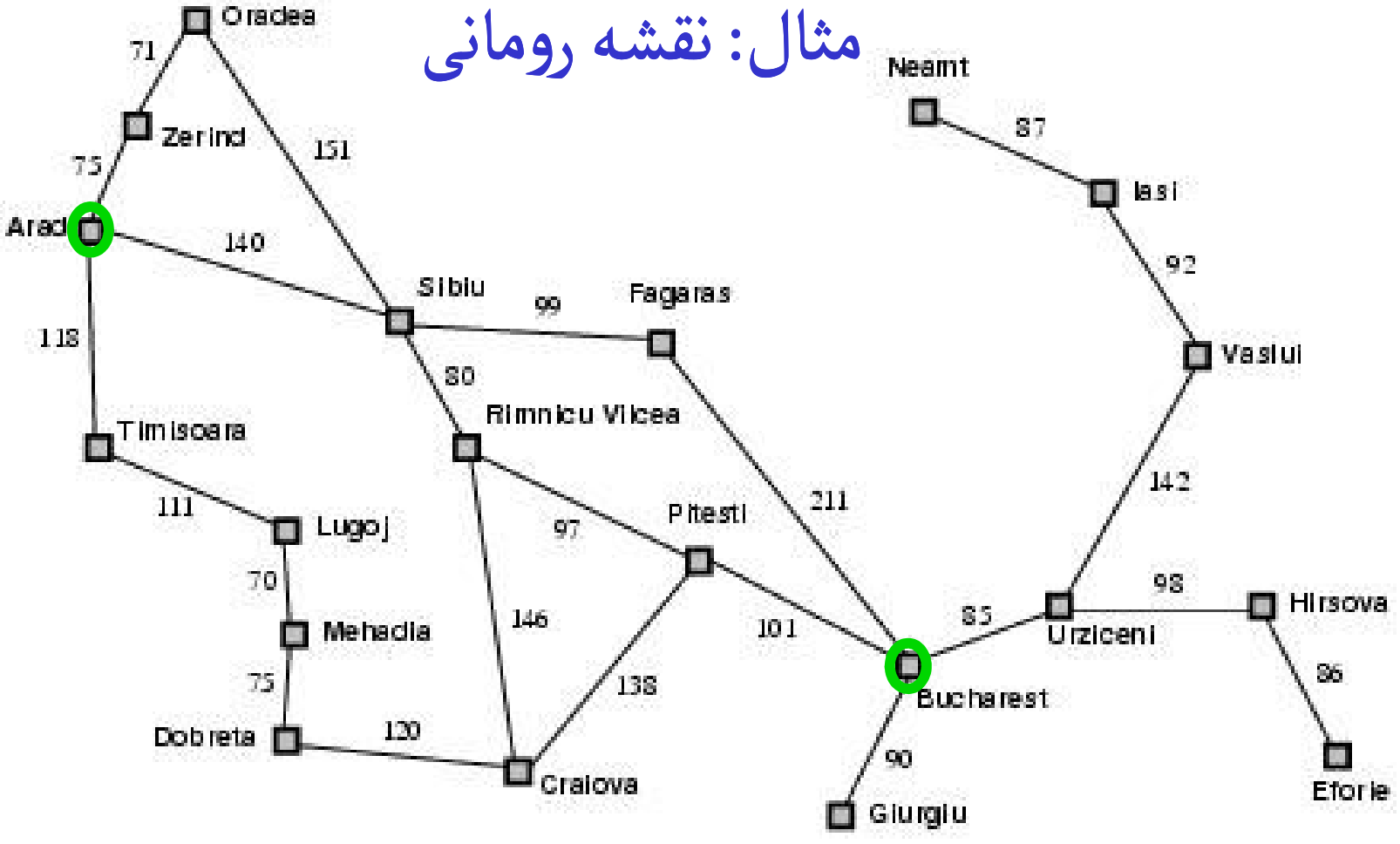
عاملهای حل مسئله

چهار گام اساسي برای حل مسائل

- فرموله کردن هدف: وضعیتهای مطلوب نهايي کدامند؟
- فرموله کردن مسئله: چه فعاليتها و وضعيتهايي برای رسيدن به هدف موجود است؟
- جستجو: انتخاب بهترين دنباله از فعاليتهايي که منجر به حالاتی با مقدار شناخته شده ميشود.
- اجرا: وقتی دنباله فعاليت مطلوب پيدا شد ، فعاليتهای پيشنهادی آن ميتواند اجرا شود.

حل مسئله با جستجو

مثال: نقشه رومانی



حل مسئله با جستجو

مثال: نقشه رومانی

- صورت مسأله: رفتن از آراد به بخارست
- فرموله کردن هدف: رسیدن به بخارست
- فرموله کردن مسئله:
  - وضعیتها: شهرهای مختلف
  - فعالیتها: حرکت بین شهرها
- جستجو: دنباله ای از شهرها مثل:آراد ، سیبیو ، فاگارس ، بخارست
  - این جستجو با توجه به کم هزینه ترین مسیر انتخاب میشود

حل مسئله با جستجو

مسئله

- حالت اولیه: حالتی که عامل از آن شروع میکند.
  - در مثال رومانی: شهر آراد n(Arad)
- تابع جانشین: توصیفی از فعالیتهای ممکن که برای عامل مهیا است.
  - در مثال رومانی: S(Arad)={Zerind,Sibui,Timisoara}
- فضای حالت: مجموعه ای از حالتها که از حالت اولیه میتوان به آنها رسید.
  - در مثال رومانی: کلیه شهرها که با شروع از آراد میتوان به آنها رسید

تابع جانشین + حالت اولیه = فضای حالت

حل مسئله با جستجو

- آزمون هدف: تعیین میکند که آیا حالت خاصی ، حالت هدف است یا خیر
  - هدف صریح: در مثال رومانی ، رسیدن به بخارست
  - هدف انتزاعی: در مثال شطرنج ، رسیدن به حالت کیش و مات

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

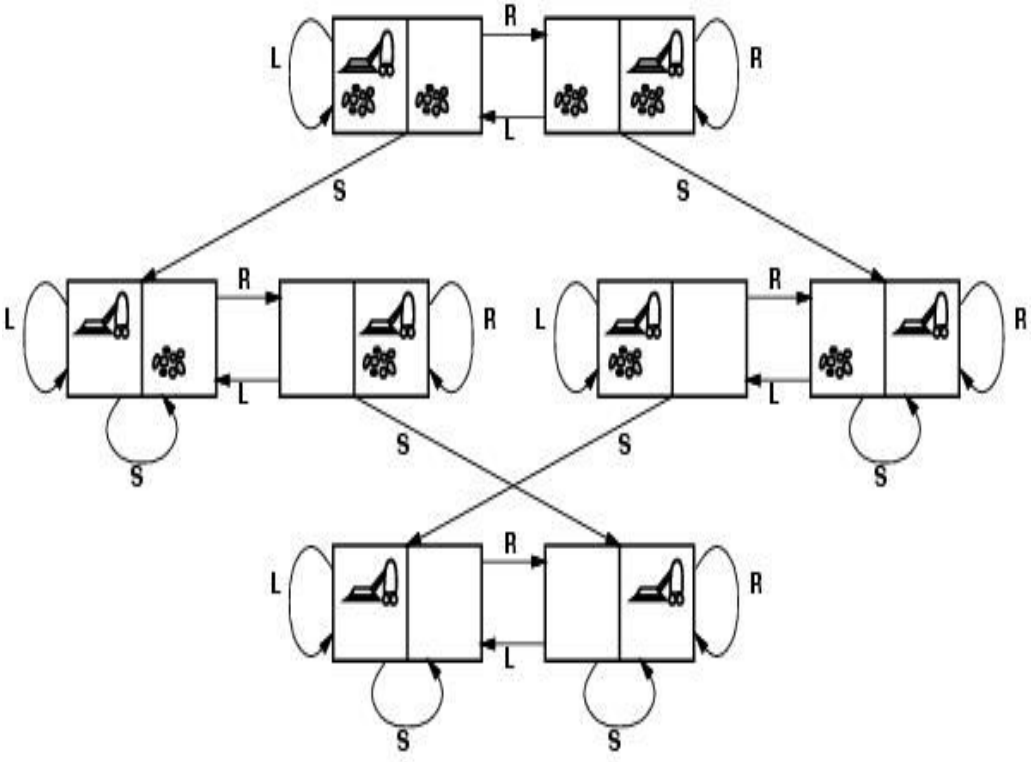
- هزینه مسیر: برای هر مسیر یک هزینه عددی در نظر میگیرد.
  - در مثال رومانی: طول مسیر بین شهرها بر حسب کیلومتر

راه حل مسئله مسیری از حالت اولیه به حالت هدف است

راه حل بهینه کمترین هزینه مسیر را دارد

حل مسئله با جستجو

مثال: دنیای جارو برقي



حالتها:

حالت اولیه:

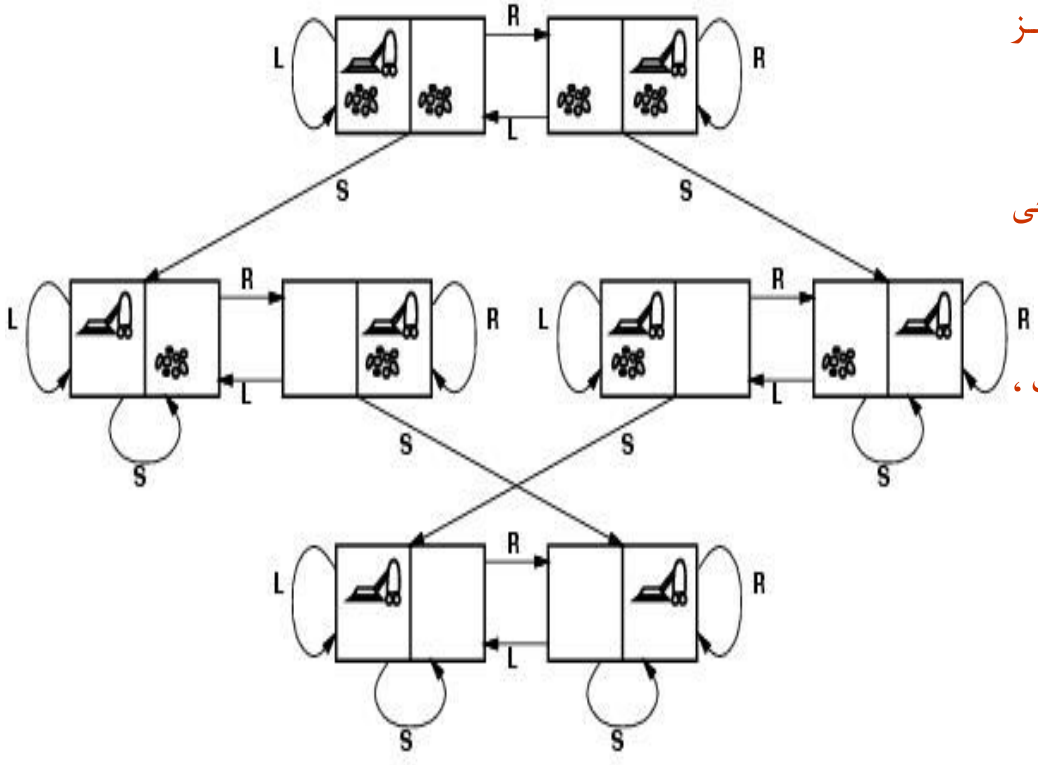
تابع جانشین:

آزمون هدف:

هزینه مسیر:

حل مسئله با جستجو

مثال: دنیای جارو برقي



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

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

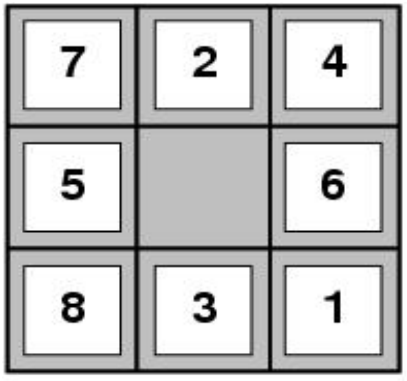
تابع جانشین: حالتهای معتبر از سه عملیات: راست ، چپ ، مکش

آزمون هدف: تمیزی تمام مربعها

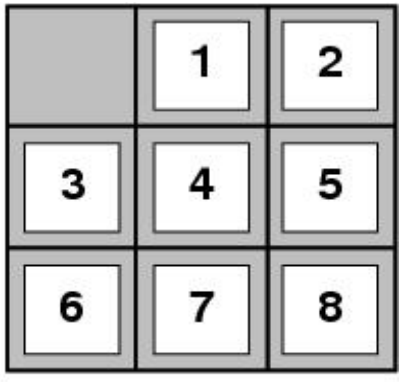
هزینه مسیر: تعداد مراحل در مسیر

حل مسئله با جستجو

مثال: معمای8



Start State



Goal State

حالتها:

حالت اولیه:

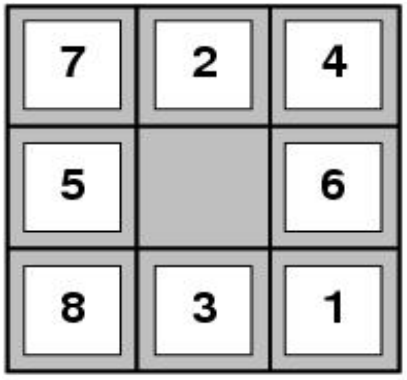
تابع جانشین:

آزمون هدف:

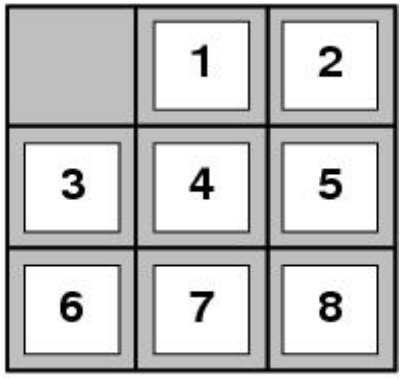
هزینه مسیر:

حل مسئله با جستجو

مثال: معمای8



Start State



Goal State

حالتها: مکان هر هشت خانه شماره دار و خانه خالی در یکي از 9 خانه

حالت اولیه: هر حالتي را میتوان به عنوان حالت اولیه در نظر گرفت

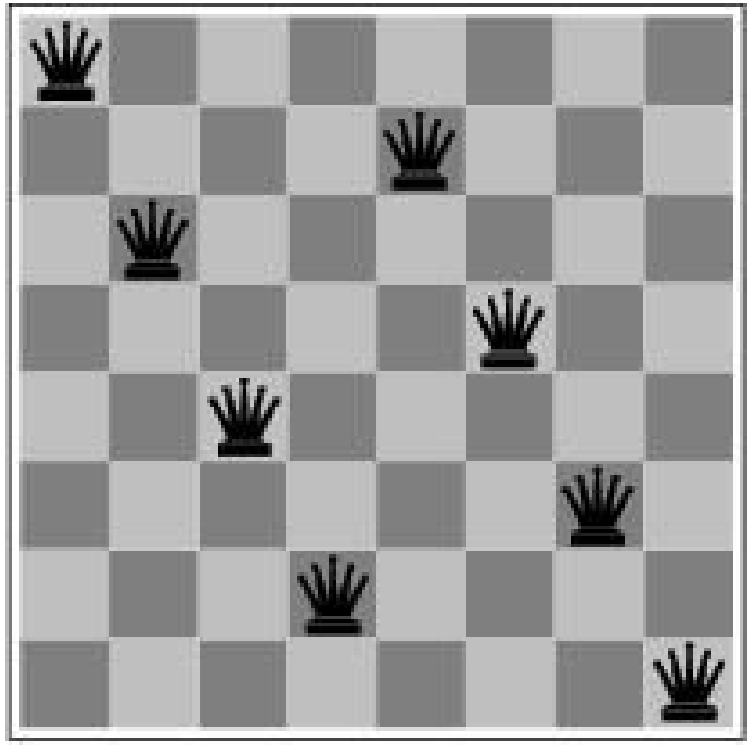
تابع جانشین: حالتهای معتبر از چهار عمل ، انتقال خانه خالی به چپ ، راست ، بالا یا پایین

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

هزینه مسیر: برابر با تعداد مراحل در مسیر

حل مسئله با جستجو

مثال: مسئله 8 وزیر



فرمول بندی افزایشي

حالتها:

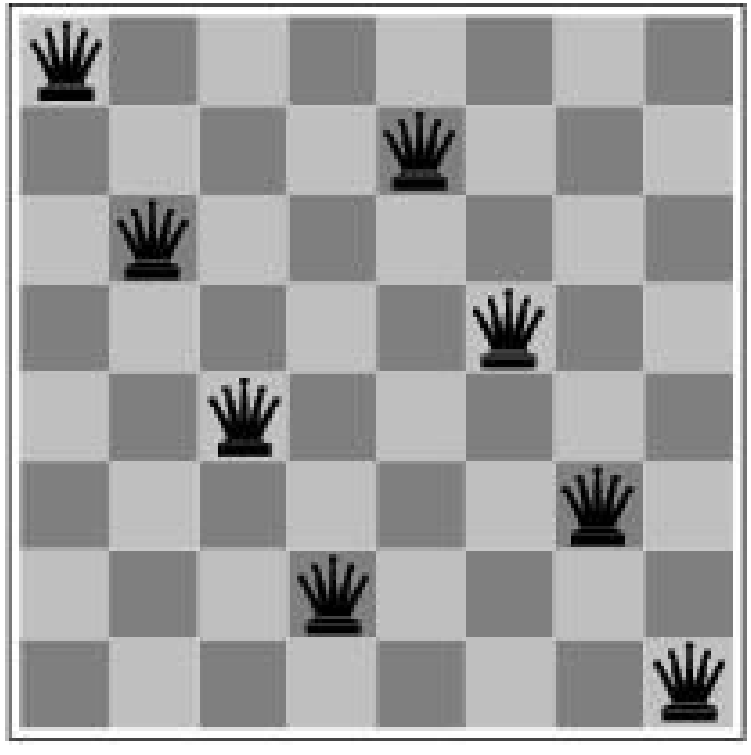
حالت اولیه:

تابع جانشین:

آزمون هدف:

حل مسئله با جستجو

مثال: مسئله 8 وزیر



فرمول بندی افزایشي

حالتها: هر ترتيبي از 0 تا 8 وزير در صفحه ، يک حالت است

حالت اولیه: هیچ وزیری در صفحه نیست

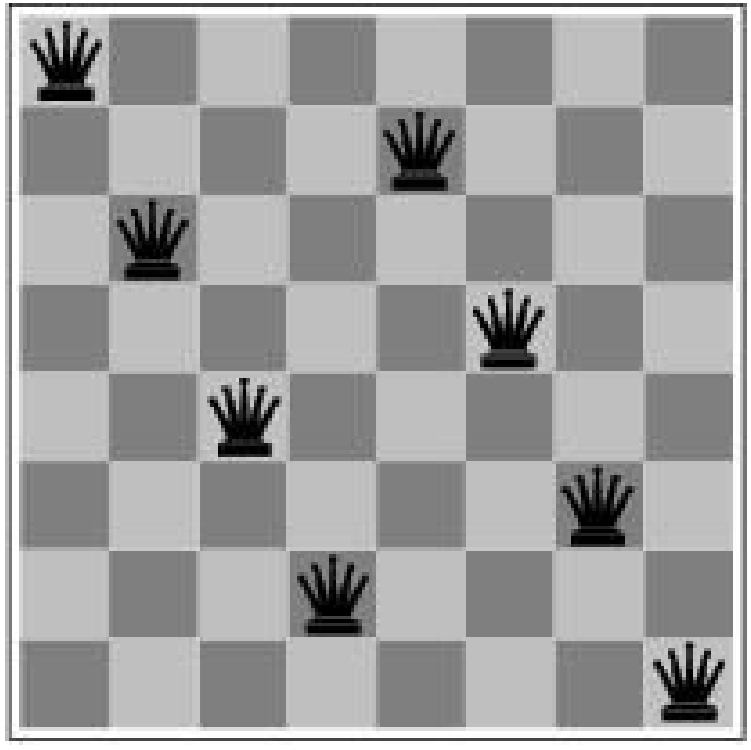
تابع جانشین: وزیری را به خانه خالی اضافه میکند

آزمون هدف: 8وزیر در صفحه وجود دارند و هیچ کدام به یکدیگر گارد نمیگیرند

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

حل مسئله با جستجو

مثال: مسئله 8 وزیر



فرمول بندی حالت کامل

حالتها:

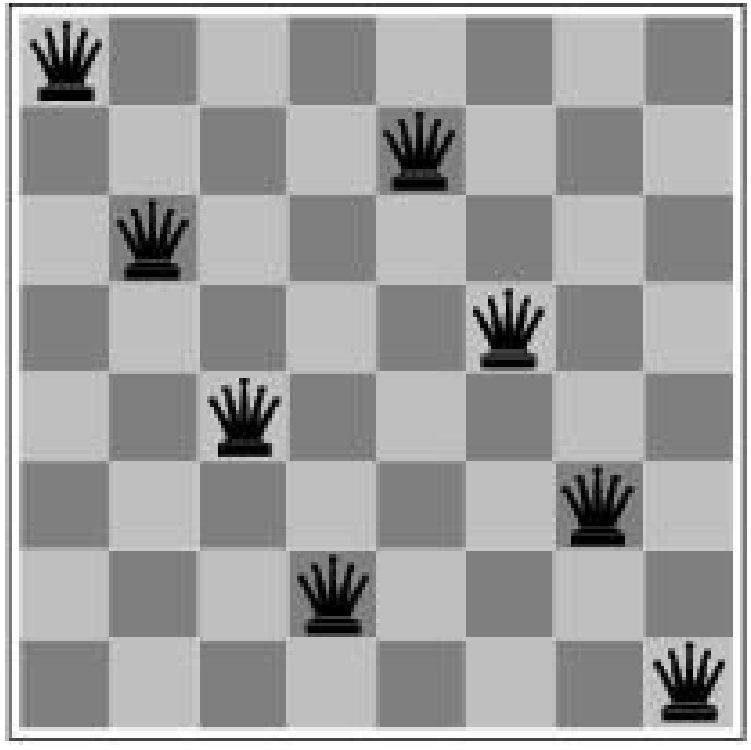
حالت اولیه:

تابع جانشین:

آزمون هدف:

حل مسئله با جستجو

مثال: مسئله 8 وزیر



فرمول بندی حالت کامل

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 بهم گارد نگیرند

حالت اولیه: با 8 وزیر در صفحه شروع میشود

تابع جانشین: وزیری را در سمت چپ ترین ستون خالي قرار میدهد ، بطوری که هیچ وزیری آن را گارد ندهد

آزمون هدف: 8وزیر در صفحه وجود دارند و هیچ کدام به یکدیگر گارد نمیگیرند

این فرمول بندی فضای حالت را از 3*10^14 به 2057 کاهش میدهد

حل مسئله با جستجو

اندازه گیری کارايي حل مسئله

- کامل بودن:
- بهينگي:
- پيچيدگي زمانی:
- پيچيدگی فضا:

حل مسئله با جستجو

اندازه گیری کارایي حل مسئله

- **کامل بودن:** آیا الگوریتم تضمین میکند که در صورت وجود راه حل ، آن را بیابد؟
- **بهینگي:** آیا این راهبرد ، راه حل بهینه ای را ارائه میکند.
- **پیچیدگي زمانی:** چقدر طول میکشد تا راه حل را پیدا کند؟
  - تعداد گره های تولید شده در اثنای جستجو
- **پیچیدگی فضا:** برای جستجو چقدر حافظه نیاز دارد؟
  - حداکثر تعداد گره های ذخیره شده در حافظه

حل مسئله با جستجو

جستجوی ناآگاهانه

- ناآگاهی این است که الگوریتم هیچ اطلاعاتی غیر از تعریف مسئله در اختیار ندارد
- این الگوریتمها فقط میتواند جانشینهایی را تولید و هدف را از غیر هدف تشخیص دهند
- راهبردهایی که تشخیص میدهد یک حالت غیر هدف نسبت به گره غیر هدف دیگر ، امید بخش تر است ، جست و جوی آگاهانه یا جست و جوی اکتشافی نامیده میشود.

راهبردها

- جست و جوی عرضی
- جست و جوی هزینه یکنواخت
- جست و جوی عمقی
- جست و جوی عمقی محدود
- جست و جوی عمیق کننده تکراری
- جست و جوی دو طرفه

حل مسئله با جستجو

جستجوی عرضی

حل مسئله با جستجو

جستجوی عرضی

کامل بودن: بله

بهینگی: بله (مشروط)

در صورتی بهینه است که هزینه مسیر ، تابعی غیر نزولی از عمق گره باشد.(مثل وقتي كه فعالیتها هزینه یکسانی دارند)

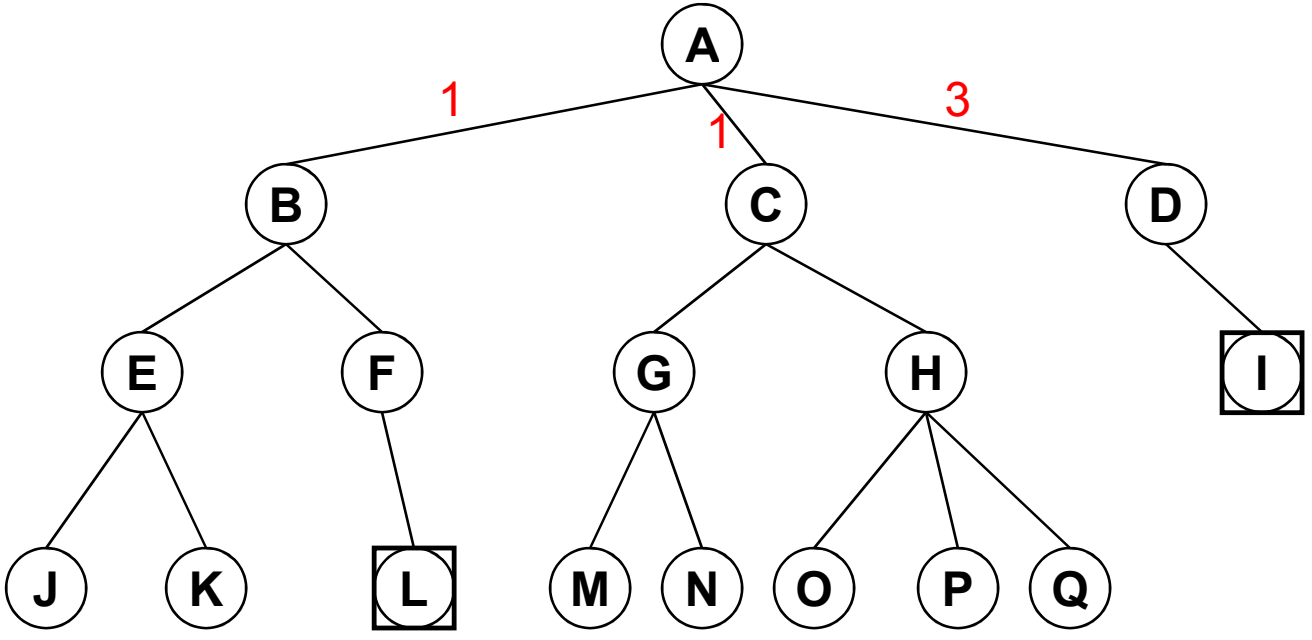
پیچیدگي زماني: $O(b^{d+1})$

پیچیدگی فضا: $O(b^{d+1})$

حل مسئله با جستجو

جستجوی هزینه یکنواخت

این جستجو گره n را با کمترین هزینه مسیر بسط میدهد



حل مسئله با جستجو

جستجوی هزینه یکنواخت

کامل بودن: بله

هزینه هر مرحله بزرگتر یا مساوی یک مقدار ثابت و مثبت ε باشد.(هزینه مسیر با حرکت در مسیر افزایش مي یابد)

بهینگی: بله

هزینه هر مرحله بزرگتر یا مساوی **ε** باشد

پیچیدگي زماني: $O(b^{[C^*/\varepsilon]})$

پیچیدگی فضا: $O(b^{[C^*/\varepsilon]})$

حل مسئله با جستجو

جستجوی عمقی

حل مسئله با جستجو

جستجوی عمقی

کامل بودن:

اگر زیر درخت چپ عمق نامحدود داشت و فاقد هر گونه راه حل باشد ، جستجو هرگز خاتمه نمي يابد.

بهینگی:

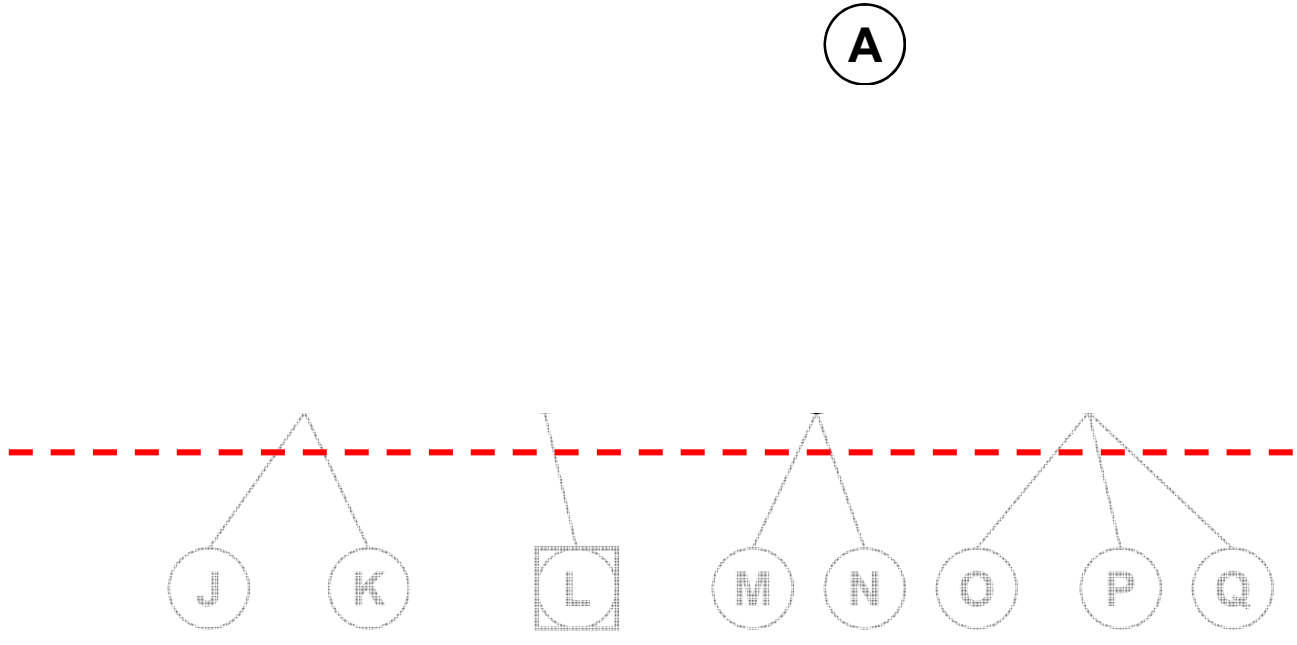
پیچیدگي زماني: $O(b^m)$

پیچیدگی فضا: $O(bm)$

حل مسئله با جستجو

جستجوی عمقی محدود

مسئله درختهای نامحدود میتواند به وسیله جست و جوی عمقي با عمق محدود L بهبود یابد



حل مسئله با جستجو

جستجوی عمقی محدود

کامل بودن:
اگر L>d و سطحی ترین هدف در خارج از عمق محدود قرار داشته باشد ، این راهبرد کامل نخواهد بود.

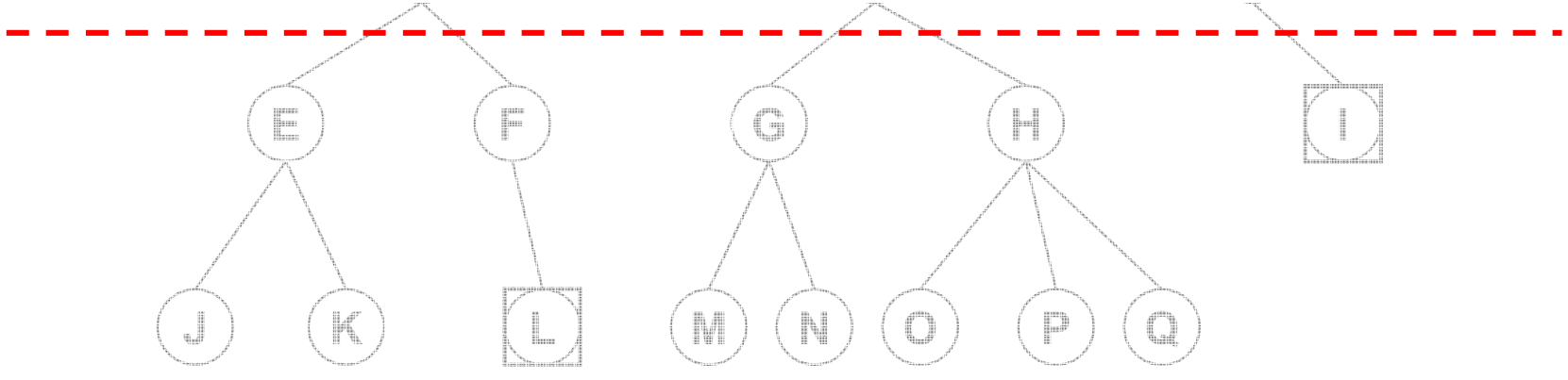
بهینگی:
اگر L>d انتخاب شود ، این راهبرد بهینه نخواهد بود.

پیچیدگی زمانی: $O(b^L)$

پیچیدگی فضا: $O(bL)$

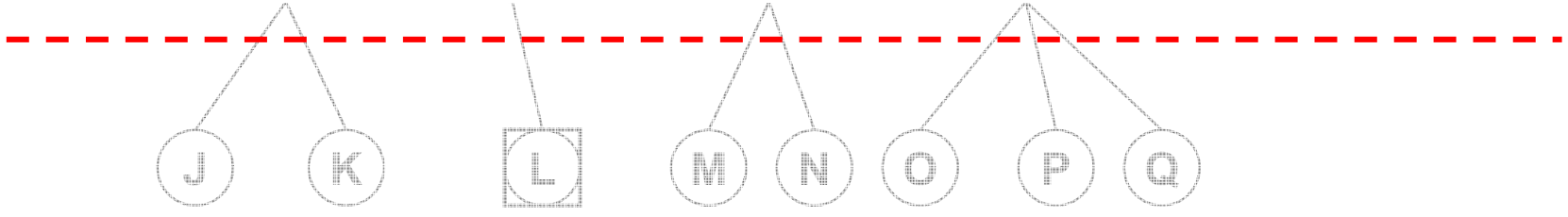
حل مسئله با جستجو

جستجوی عمیق کننده تکراري



حل مسئله با جستجو

جستجوی عمیق کننده تکراري



حل مسئله با جستجو

جستجوی عمیق کننده تکراري

حل مسئله با جستجو

جستجوی عمیق کننده تکراري

کامل بودن:

در صورتی که فاکتور انشعاب محدود باشد

بهینگی:

وقتی که هزینه مسیر ، تابعی غیر نزولی از عمق گره باشد

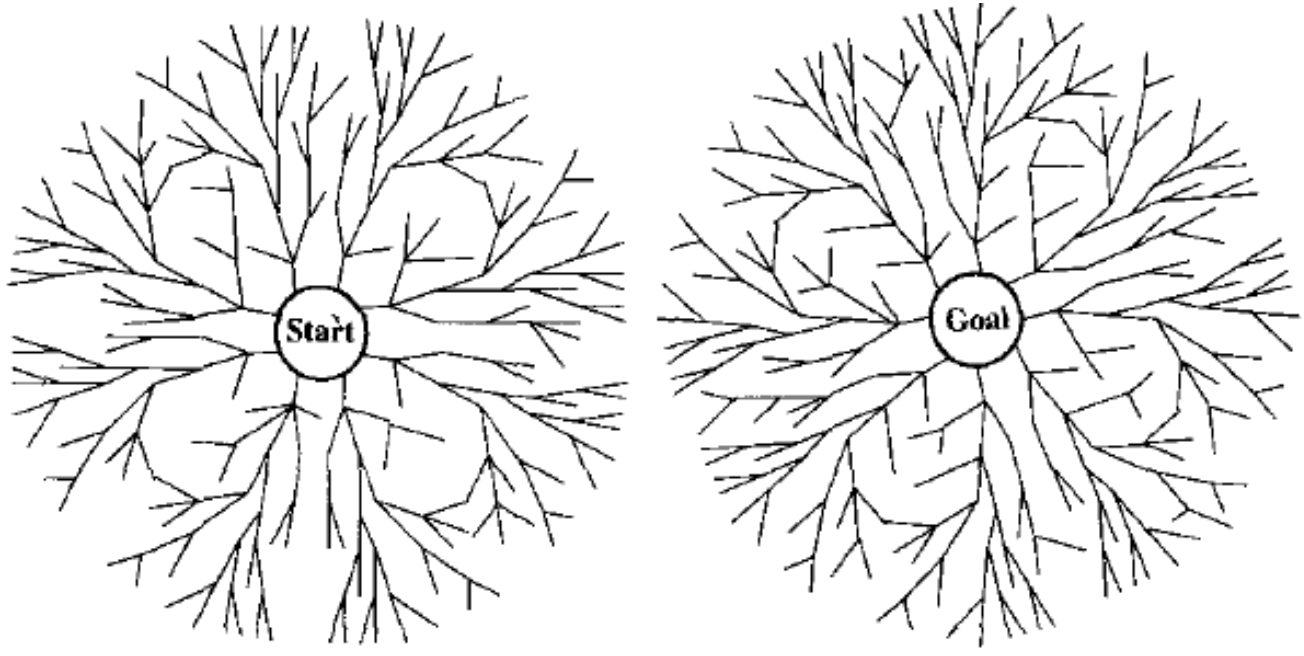
پیچیدگي زماني: $O(b^d)$

پیچیدگی فضا: $O(bd)$

حل مسئله با جستجو

جستجوی دو طرفه

انجام دو جست و جوی همزمان ، یکي از حالت اولیه به هدف و دیگری از هدف به حالت اولیه تا زمانی که دو جست و جو به هم برسند



حل مسئله با جستجو

جستجوی دو طرفه

کامل بودن:

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

بهینگی:

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

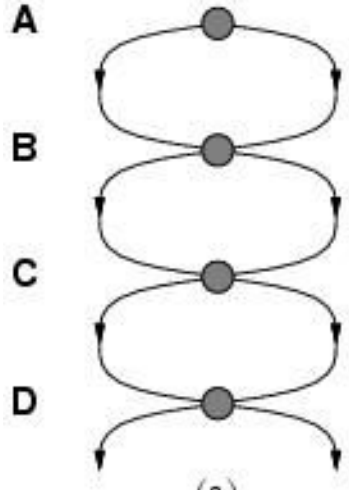
پیچیدگی زمانی: $O(b^{d/2})$

پیچیدگی فضا: $O(b^{d/2})$

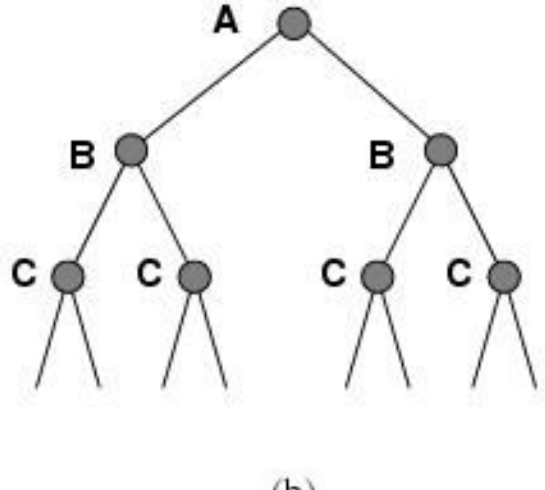
حل مسئله با جستجو

اجتناب از حالتهای تکراری

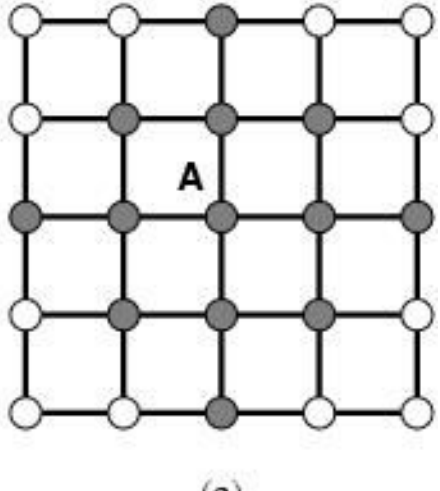
وجود حالتهای تکراری در یک مسئله قابل حل ، میتواند آن را به مسئله غیر قابل حل تبدیل کند



(a)



(b)



(c)

حل مسئله با جستجو

جستجو با اطلاعات ناقص

مسئله های فاقد حسگر: اگر عامل فاقد حسگر باشد ، میتواند در يكي از چند حالت اوليه باشد و هر فعاليت ميتواند آن را به يكي از چند حالت جانشين ببرد

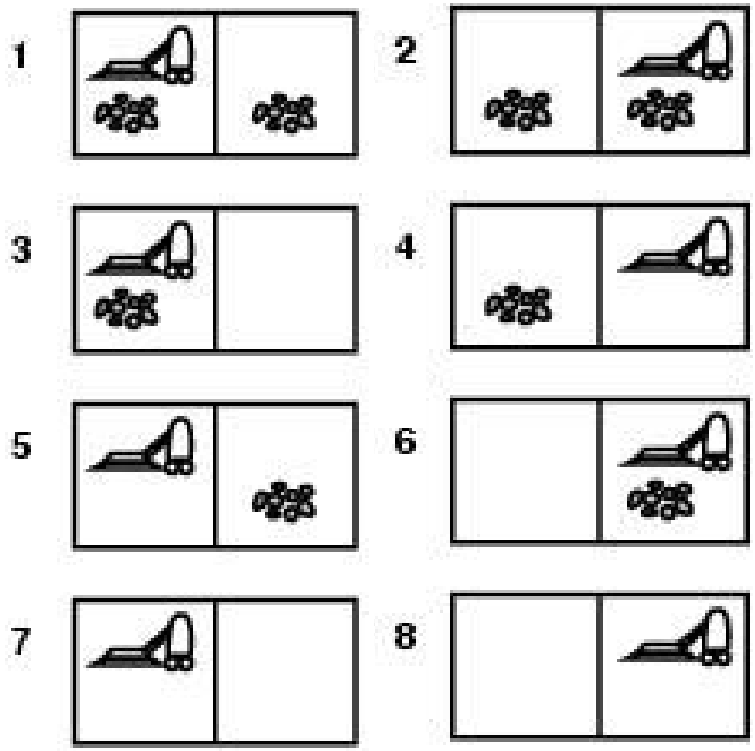
مسئله های اقتضايي: اگر محيط به طور جزئی قابل مشاهده باشد يا اگر فعاليتها قطعي نباشد ، ادراکات عامل ، پس از هر عمل ، اطلاعات جديدي را تهيه ميكنند. هر ادراک ممکن ، اقتضايی را تعريف ميكند كه بايد برای آن برنامه ريزی شود

- **مسائل خصمانه:** اگرعدم قطعيت در اثر فعاليتهای عامل ديگری بوجود آيد ، مسئله را خصمانه گويند

مسئله های اکتشافی: وقتی حالتها و فعاليتهای محيط ناشناخته باشند ، عامل بايد سعي كند آنها را كشف كند. مسئله های اكتشافی را ميتوان شكل نهايی مسئله های اقتضايي دانست

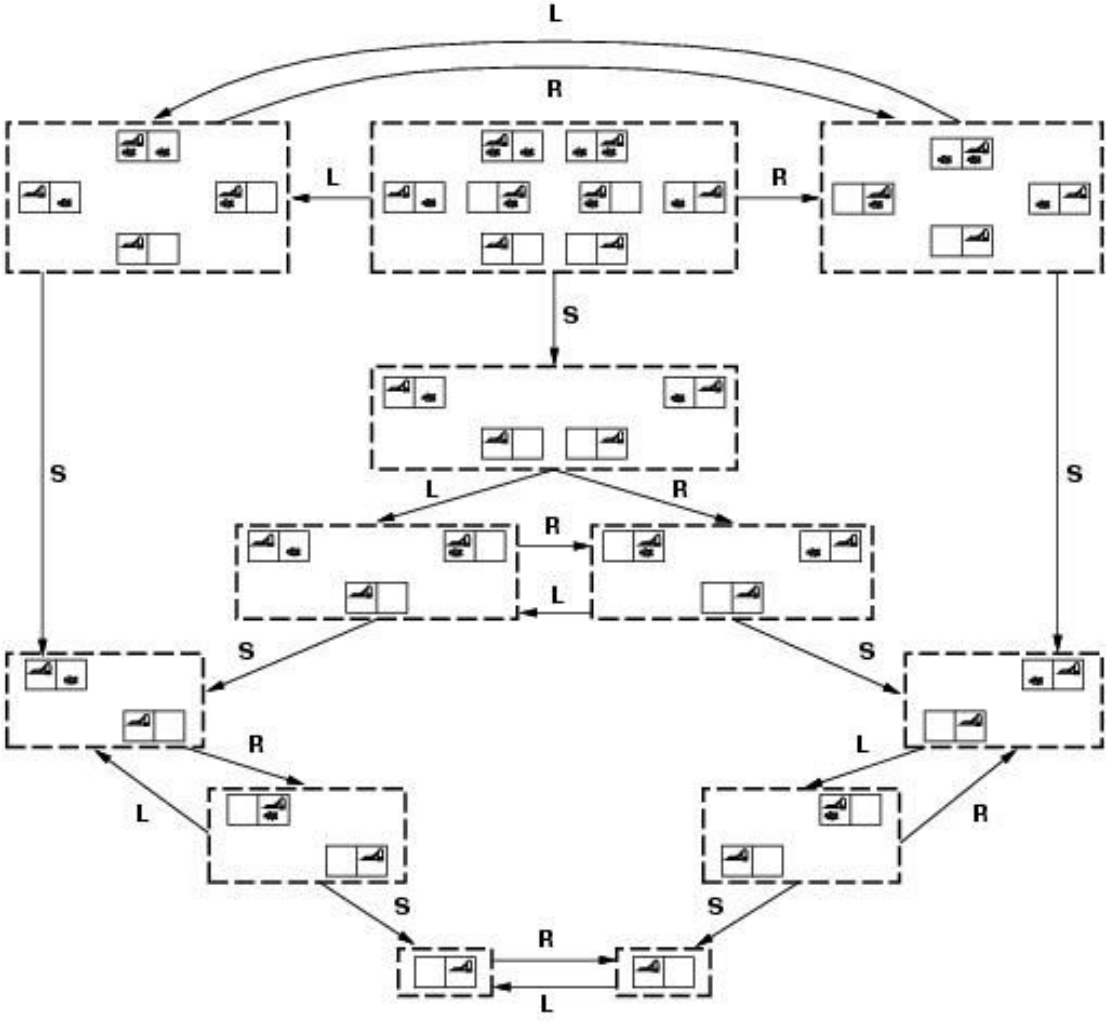
حل مسئله با جستجو

مثال: دنیای 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 7 برسد

حل مسئله با جستجو



دنیای جاروبرقی فاقد حسگر

- عامل باید راجع به مجموعه هاي حالتی که میتواند به آنها برسد استدلال کند. این مجموعه از حالتها را حالت باور گوییم.
- اگر فضای حالت فیزیکي دارای S حالت باشد فضای حالت باور 2^S حالت باور خواهد داشت.

هوش مصنوعي

فصل چهارم

جست و جوی آگاهانه و اکتشاف

هوش مصنوعي Artificial Intelligence

فهرست

- متدهای جست و جوی آگاهانه
- یادگیری برای جست و جوی بهتر
- جست و جوی محلی و بهینه سازی
- جست و جوی محلی در فضاهای پیوسته
- عاملهای جست و جوی Online

جست و جوی آگاهانه و اکتشاف

متدهای جستجوی آگاهانه

- بهترین جستجو
  - حریصانه
  - A*
  - IDA*
  - RBFS
  - MA* و SMA*
- جستجوی محلی و بهینه سازی
  - تپه نوردی
  - شبیه سازی حرارت
  - پرتو محلی
  - الگوریتمهای ژنتیک

جست و جوی آگاهانه و اکتشاف

تعاریف

- تابع هزینه مسیر ، g(n) : هزینه مسیر از گره اولیه تا گره n
- تابع اکتشافی ، h(n) : هزینه تخمینی ارزان ترین مسیر از گره n به گره هدف
- تابع بهترین مسیر ، h*(n) : ارزان ترین مسیر از گره n تا گره هدف
- تابع ارزیابی ، f(n) : هزینه تخمینی ارزان ترین مسیر از طریق n

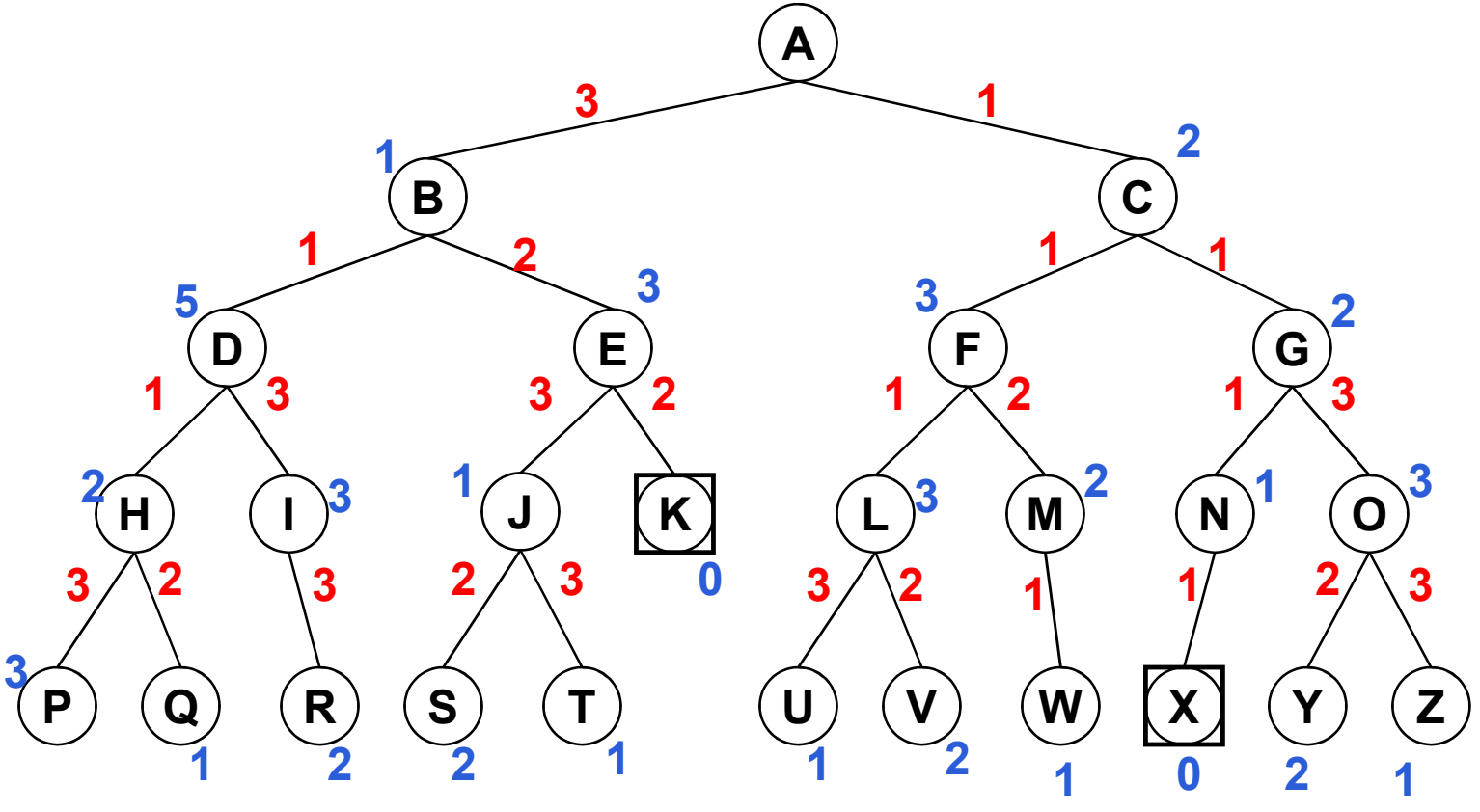
f(n): g(n) + h(n)

- f*(n) : هزینه ارزان ترین مسیر از طریق n

f*(n): g(n) + h*(n)

جست و جوی آگاهانه و اکتشاف

جستجوی حریصانه

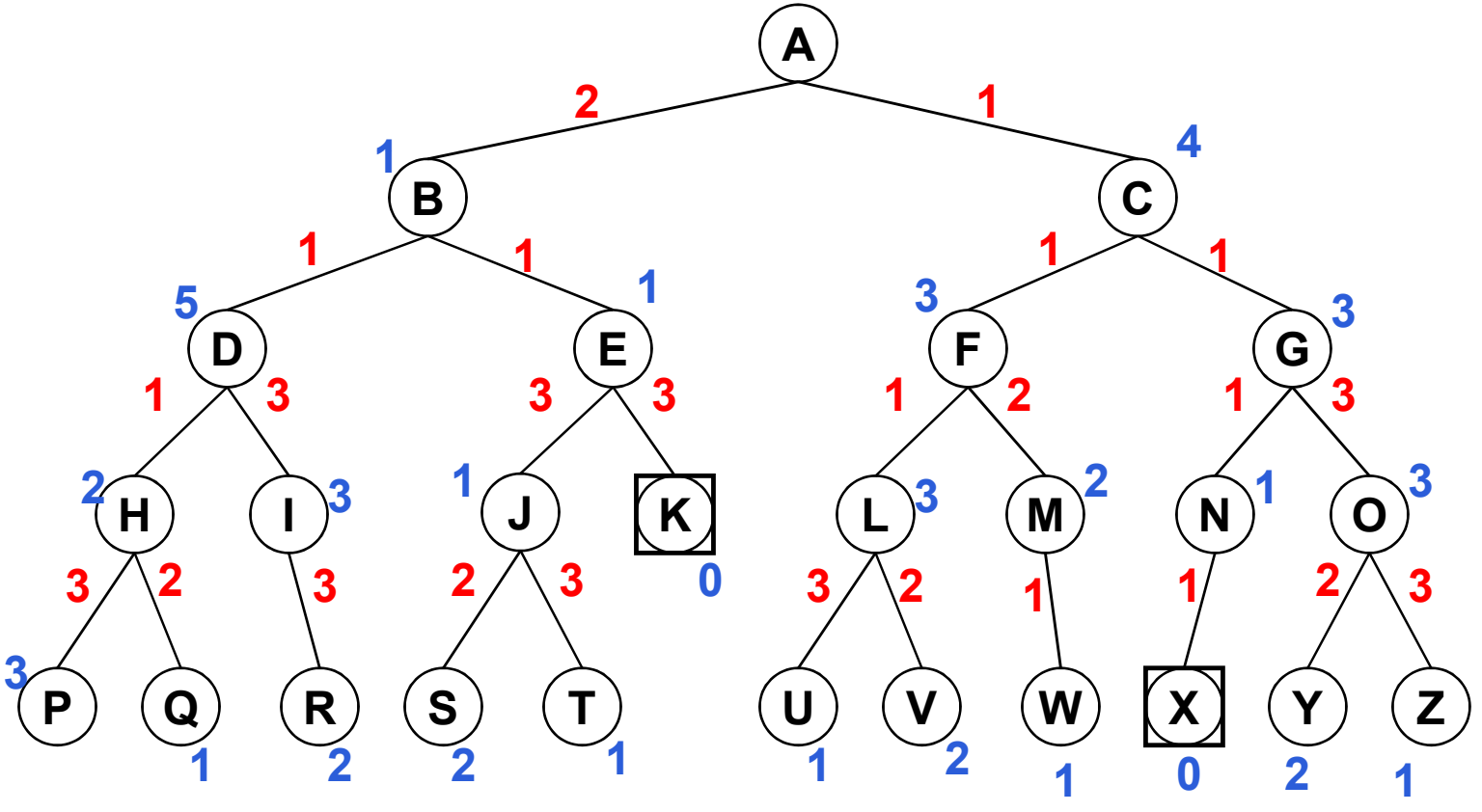


جست و جوی آگاهانه و اکتشاف

جستجوی حریصانه

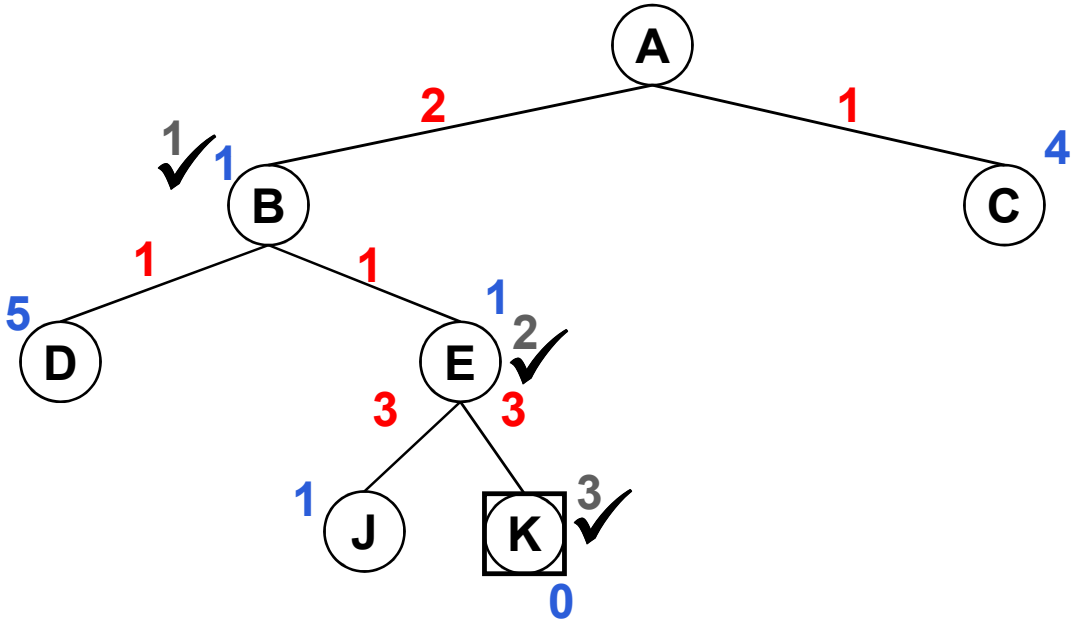
جست و جوی آگاهانه و اکتشاف

جستجوی حریصانه



جست و جوی آگاهانه و اکتشاف

جستجوی حریصانه



جست و جوی آگاهانه و اکتشاف

جستجوی حریصانه

کامل بودن: خیر

- اما اگر $h = h^*$ آنگاه جستجو کامل میشود

بهینگی: خیر

- اما اگر $h = h^*$ آنگاه جستجو کامل میشود

پیچیدگي زماني: $O(b^m)$

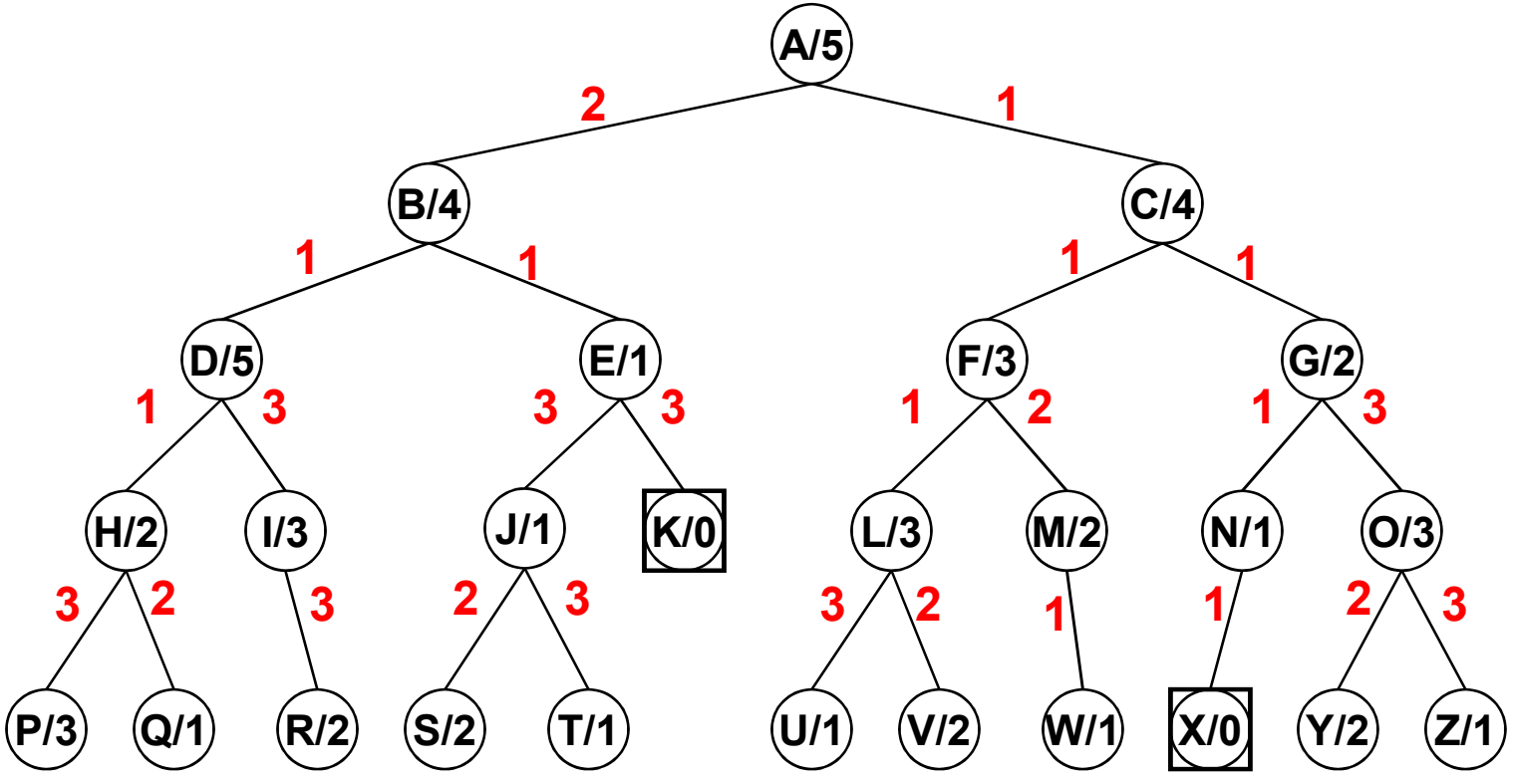
- اما اگر $h = h^*$ آنگاه $O(bd)$

پیچیدگی فضا: $O(b^m)$

- اما اگر $h = h^*$ آنگاه $O(bd)$

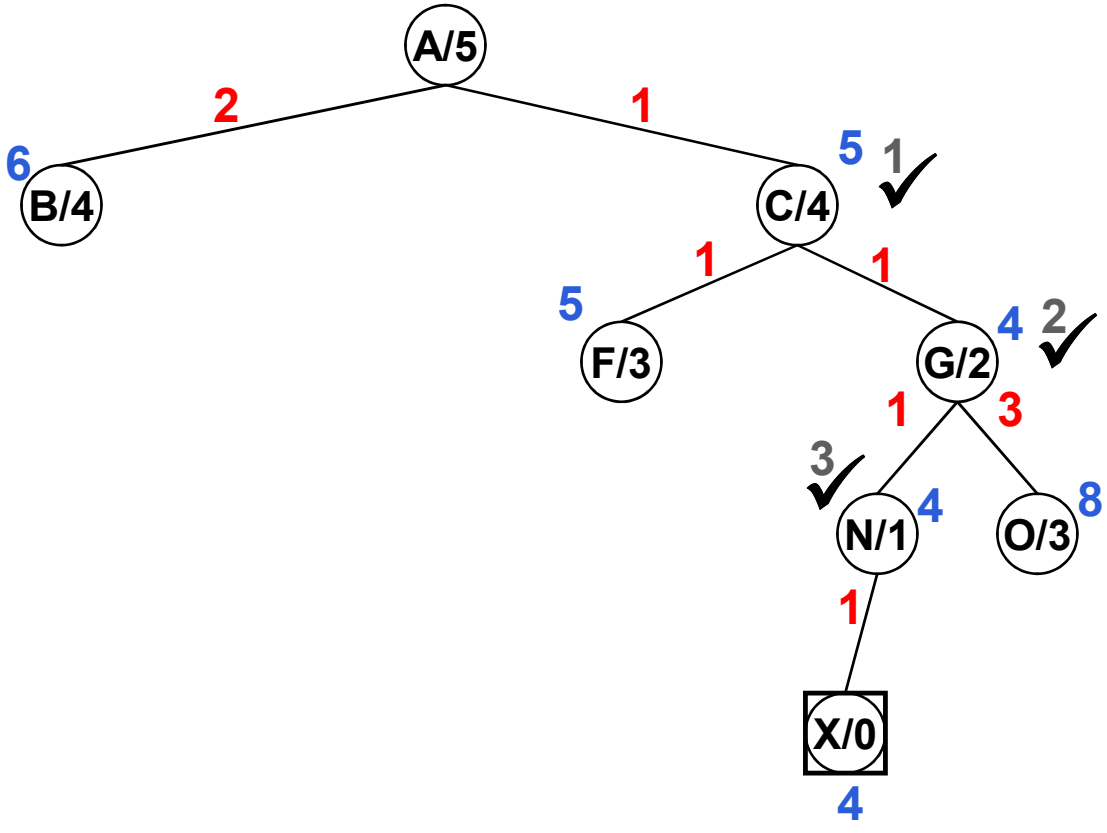
جست و جوی آگاهانه و اکتشاف

جستجوی A*



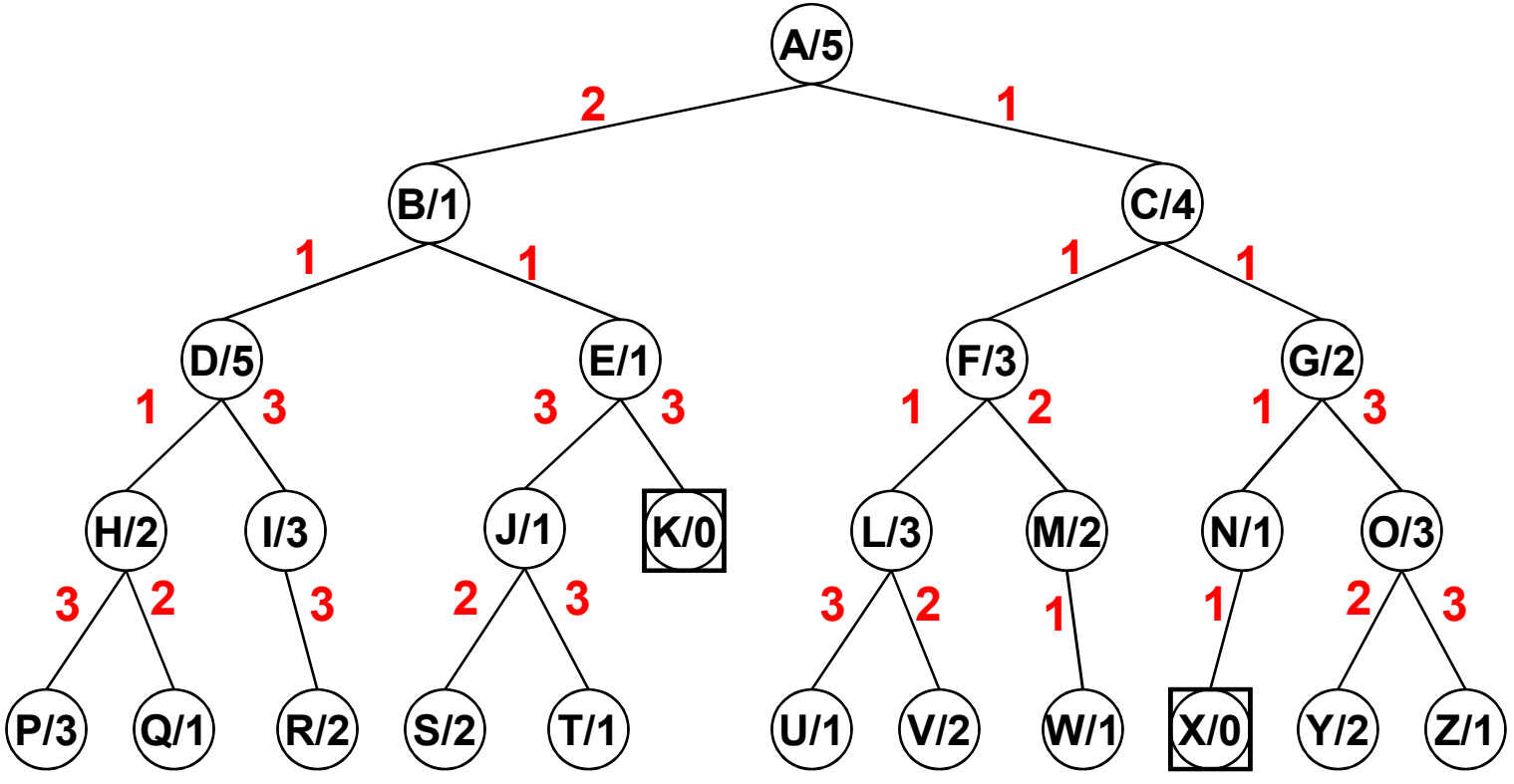
جست و جوی آگاهانه و اکتشاف

جستجوی A*



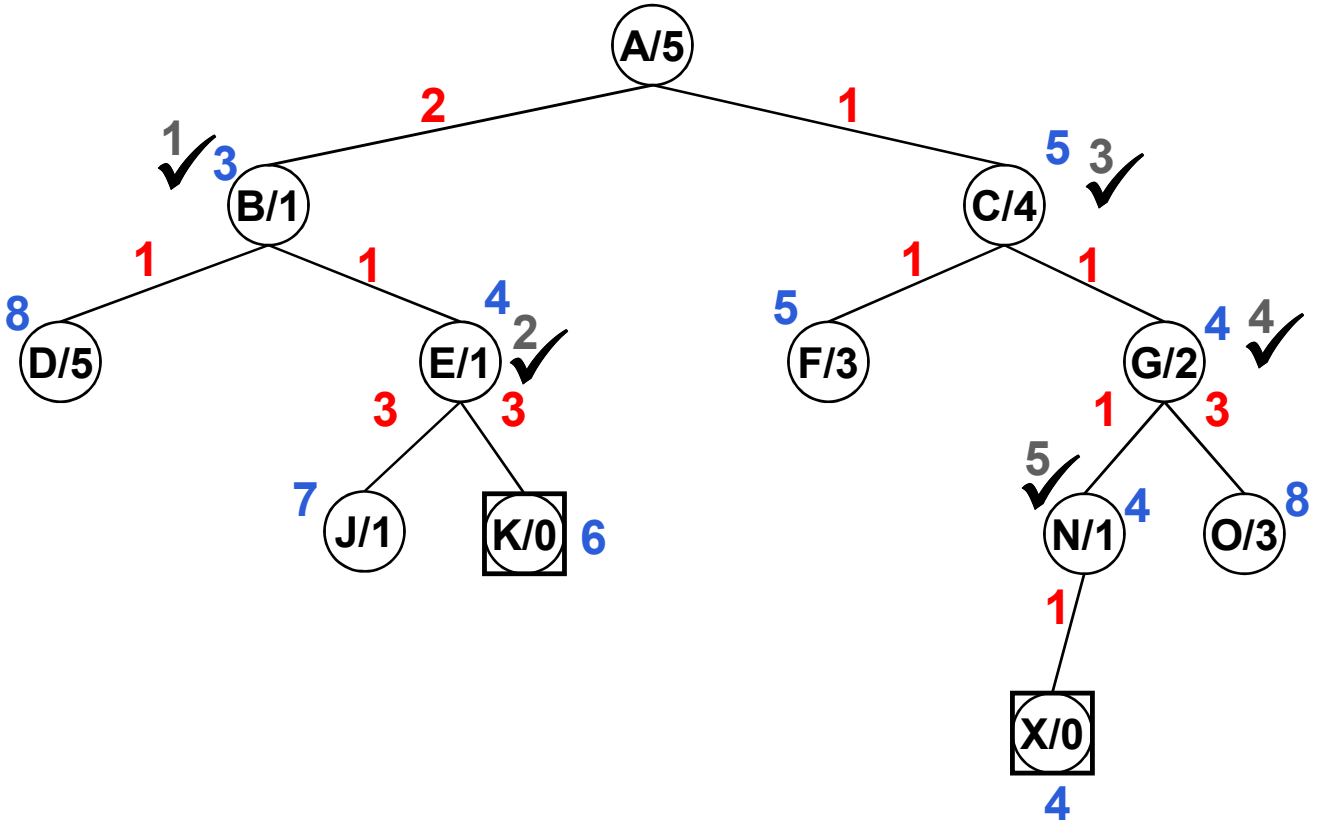
جست و جوی آگاهانه و اکتشاف

جستجوی A*



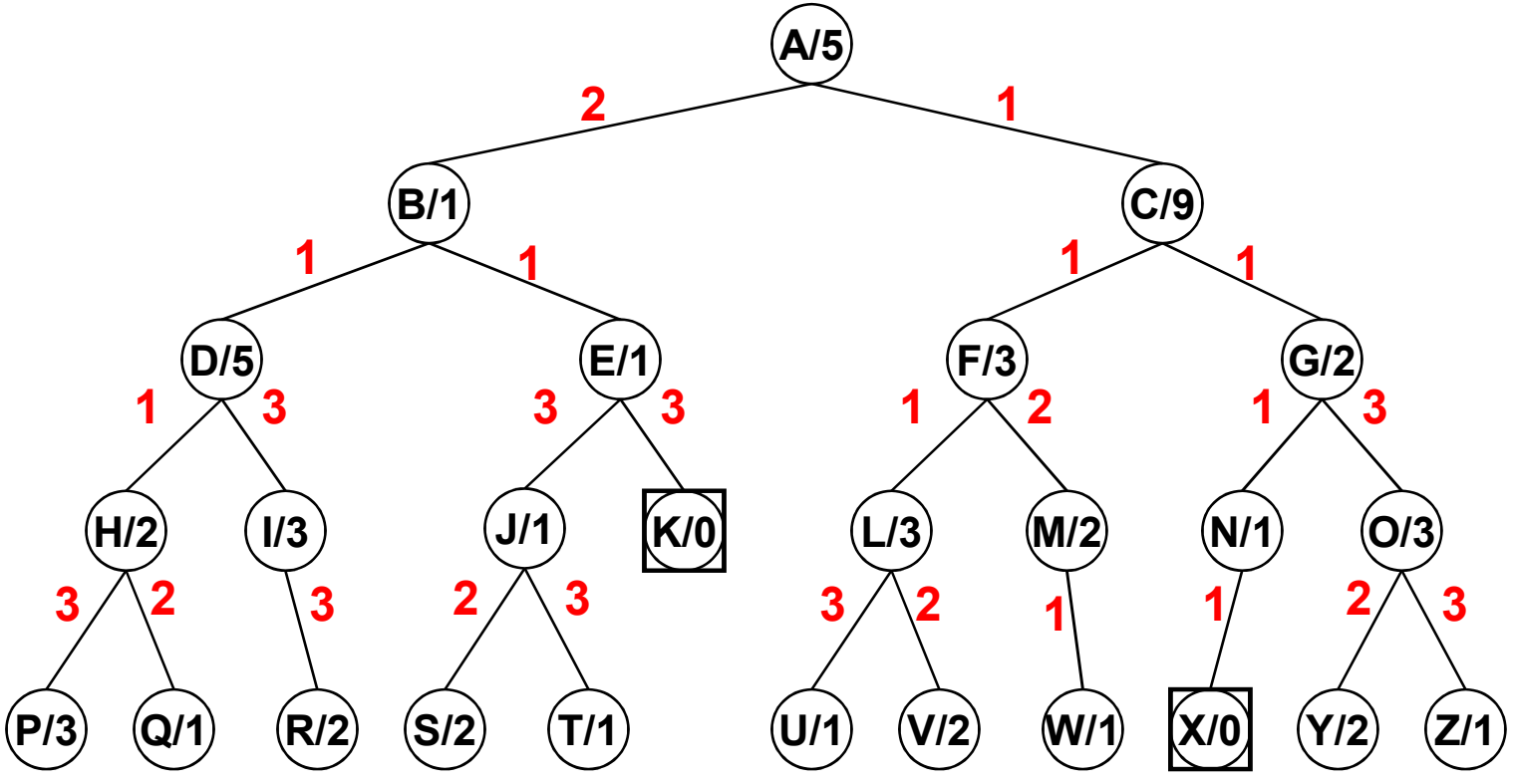
جست و جوی آگاهانه و اکتشاف

جستجوی A*



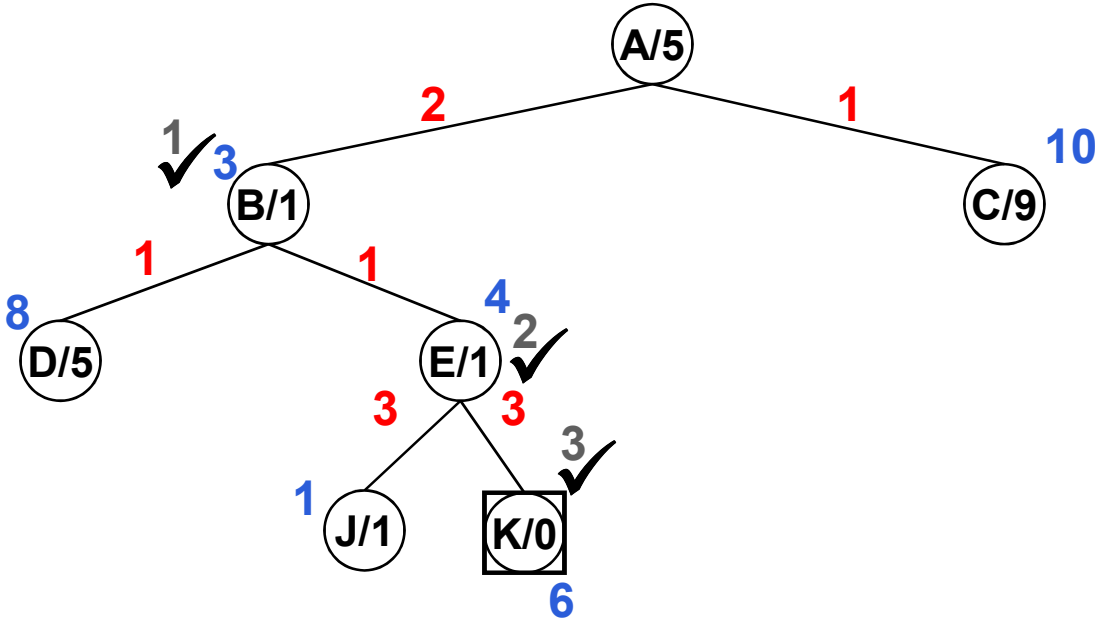
جست و جوی آگاهانه و اکتشاف

جستجوی A*



جست و جوی آگاهانه و اکتشاف

جستجوی A*



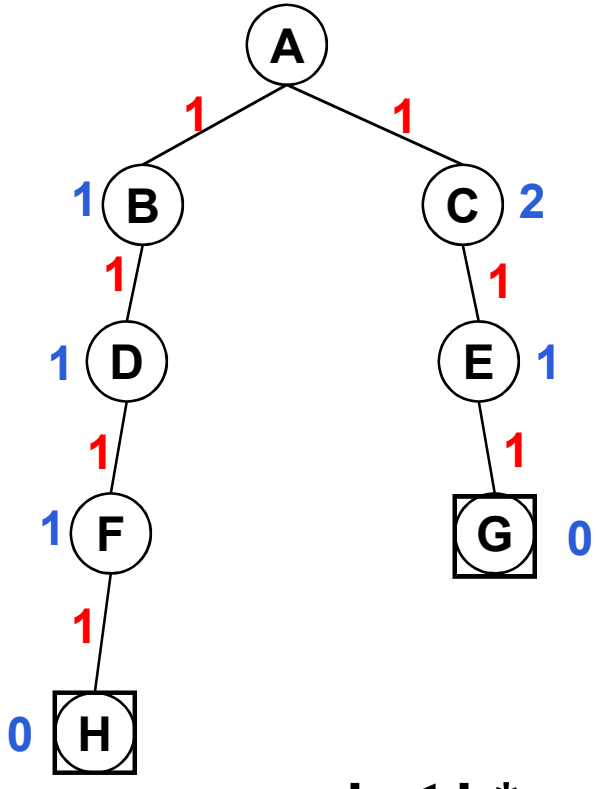
جست و جوی آگاهانه و اکتشاف

جستجوی A*

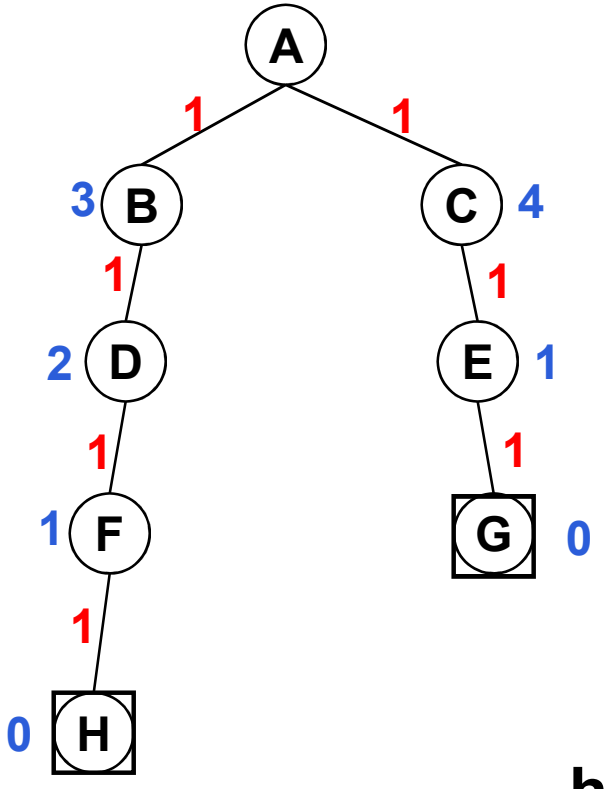
- کامل بودن: بله
- بهینگی: بله
- پیچیدگي زماني: $O(b^m)$
  - اما اگر h = h* آنگاه $O(bd)$
- پیچیدگی فضا: $O(b^m)$
  - اما اگر h = h* آنگاه $O(bd)$

جست و جوی آگاهانه و اکتشاف

جستجوی A*



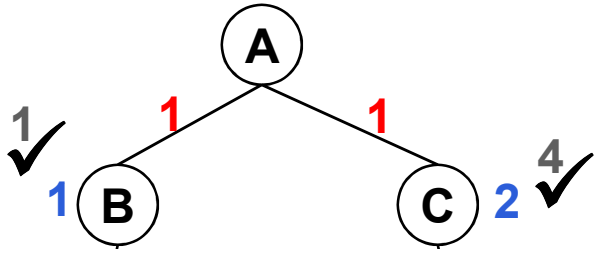
$h \leq h^*$

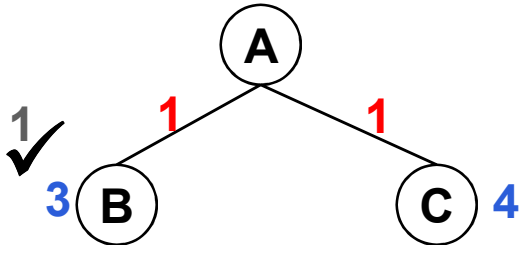


$h \not\leq h^*$

جست و جوی آگاهانه و اکتشاف

جستجوی A*



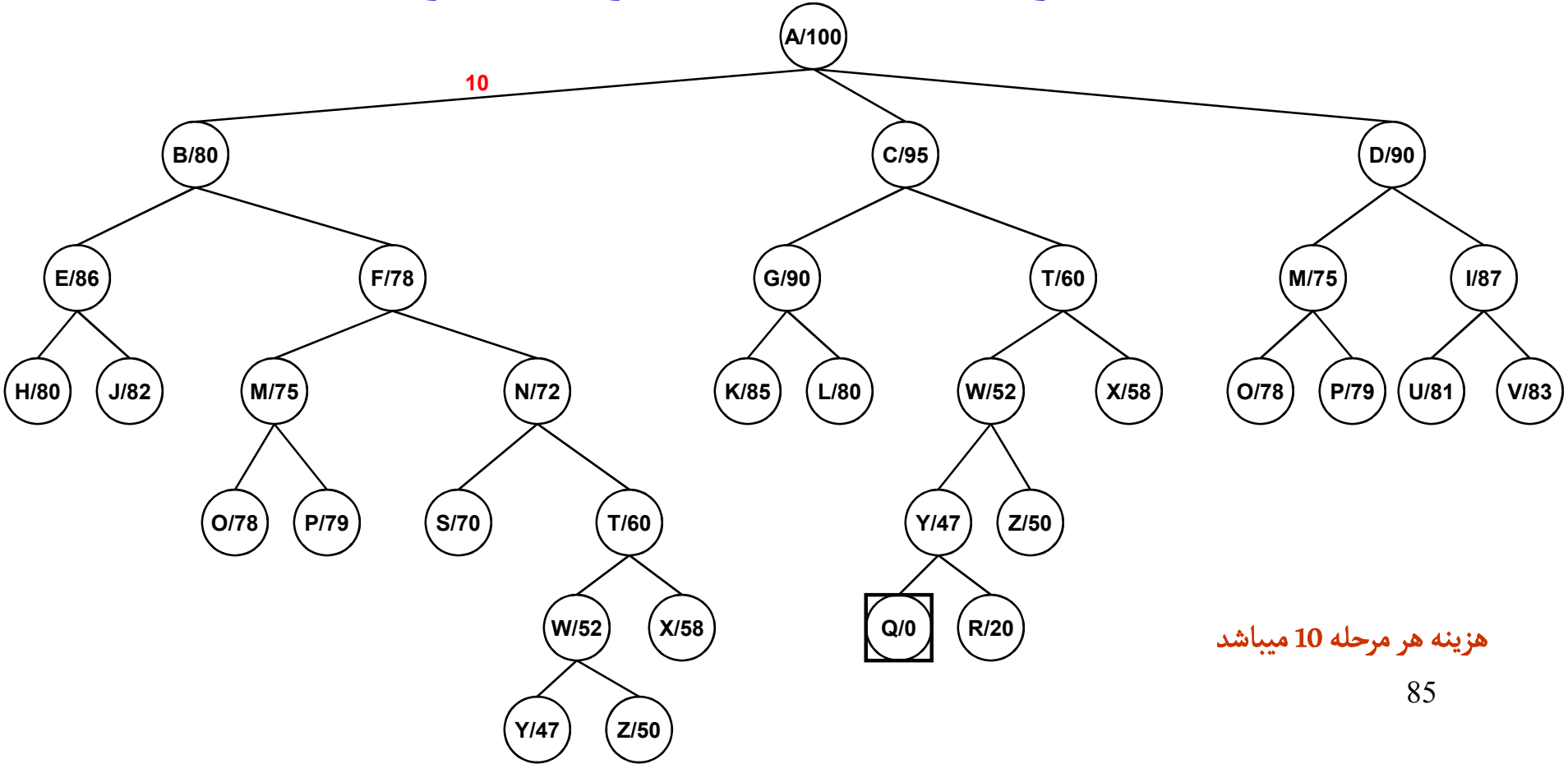


$h \leq h^*$

$h \nleq h^*$

جست و جوی آگاهانه و اکتشاف

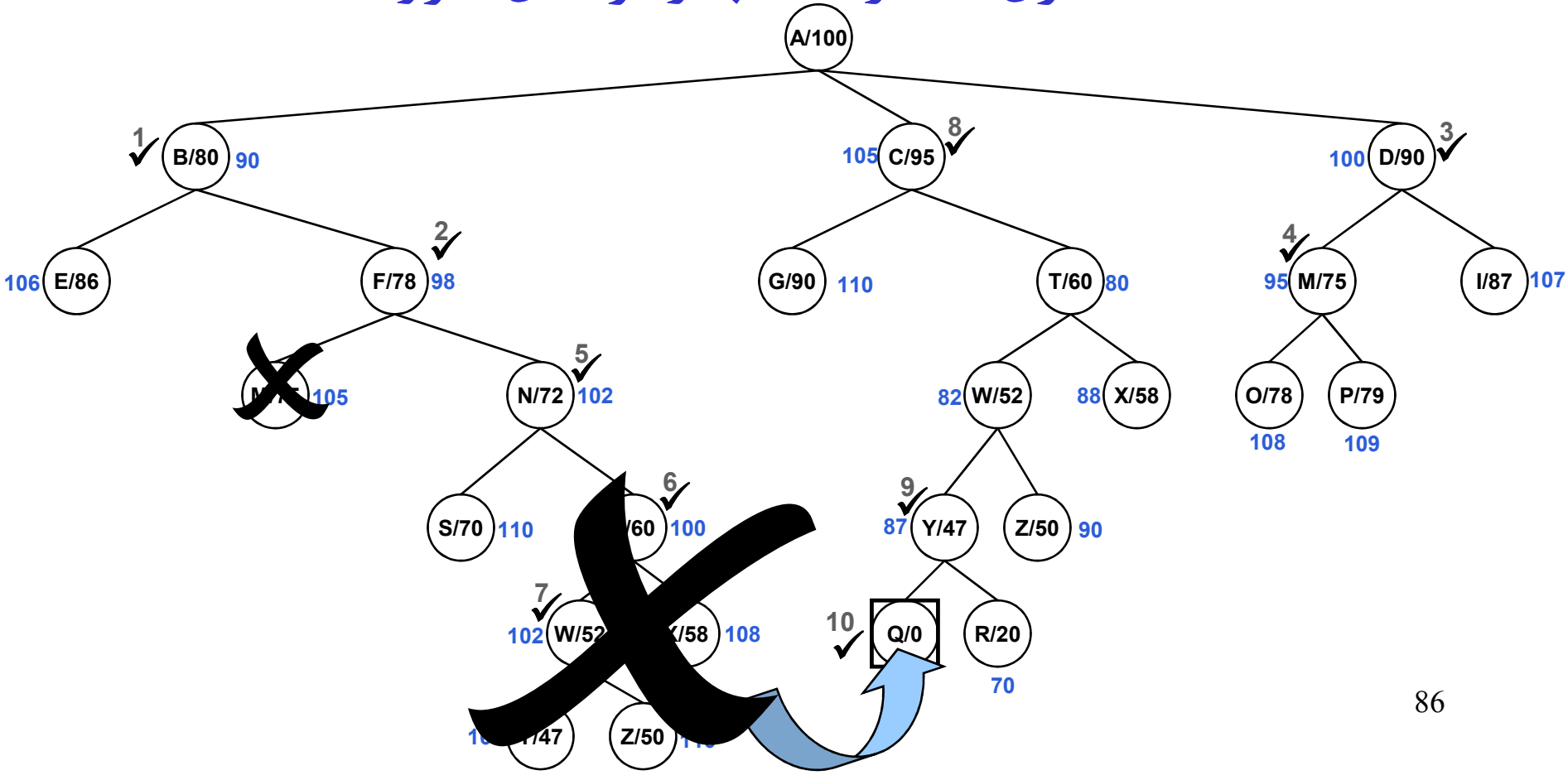
جستجوی A* و اجتناب از گره های تکراری



هزینه هر مرحله 10 میباشد

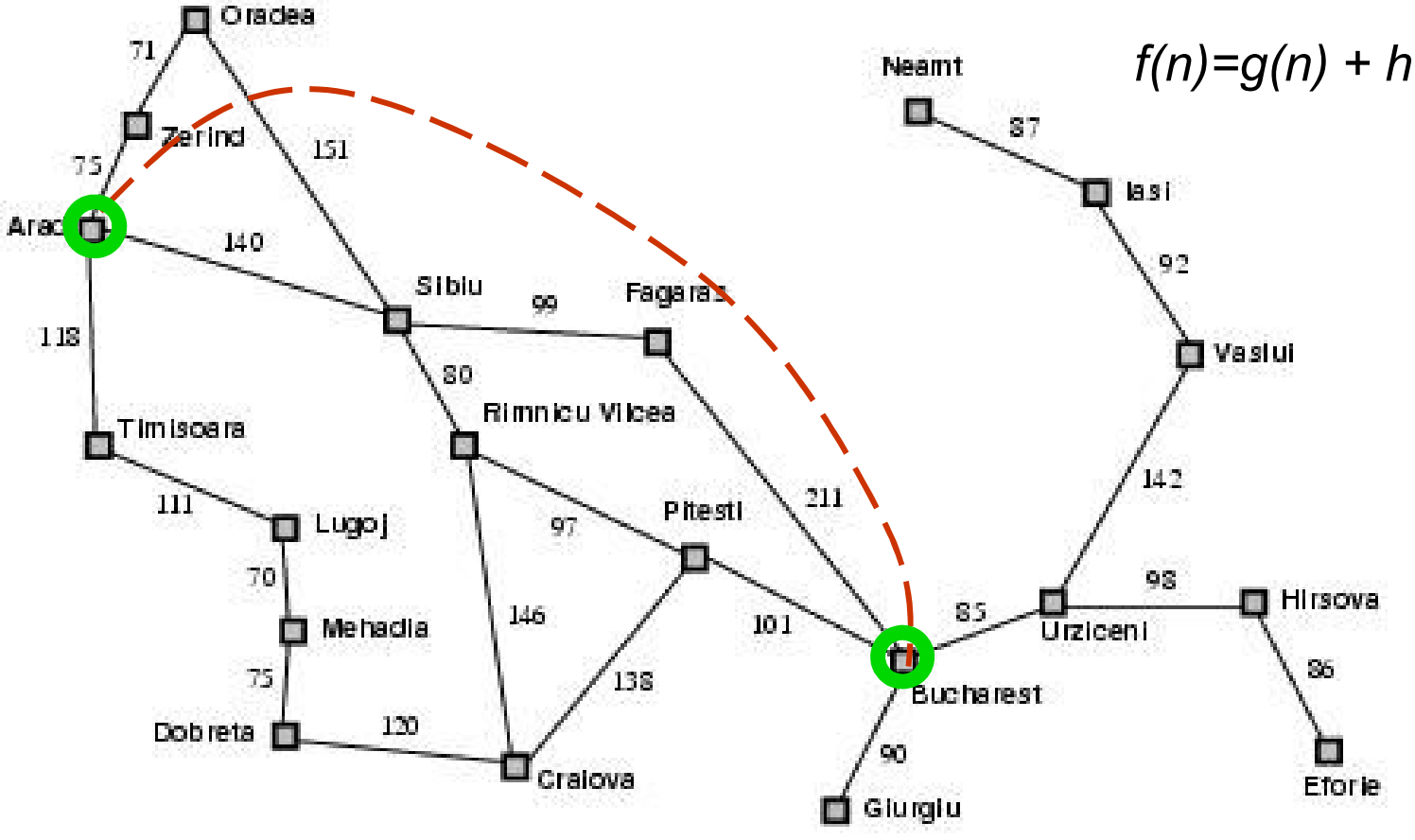
جست و جوی آگاهانه و اکتشاف

جستجوی A* و اجتناب از گره های تکراری



جست و جوی آگاهانه و اکتشاف

مثال دیگر از جستجوی A*



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

جست و جوی آگاهانه و اکتشاف

جستجوی A* در نقشه رومانی

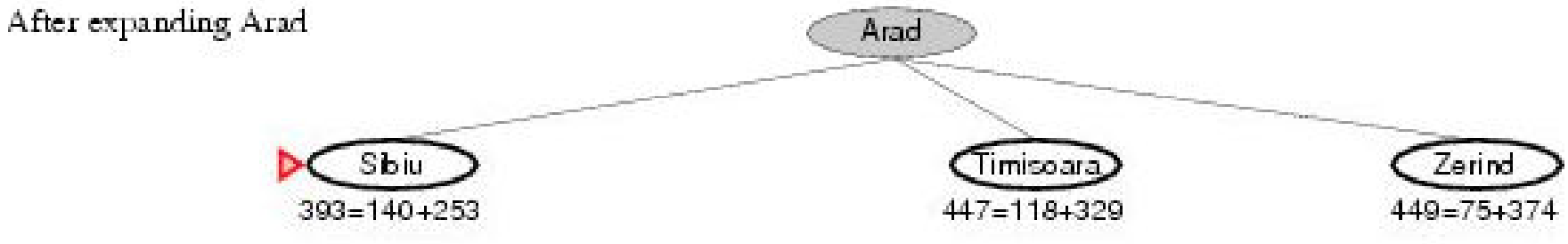
(a) The initial state

جستجوی Bucharest با شروع از Arad

f(Arad) = g(Arad)+h(Arad)=0+366=366

جست و جوی آگاهانه و اکتشاف

جستجوی A* در نقشه رومانی



Arad را باز کرده و f(n) را برای هر یک از زیربرگها محاسبه میکنیم:

f(Sibiu)=c(Arad,Sibiu)+h(Sibiu)=140+253=393

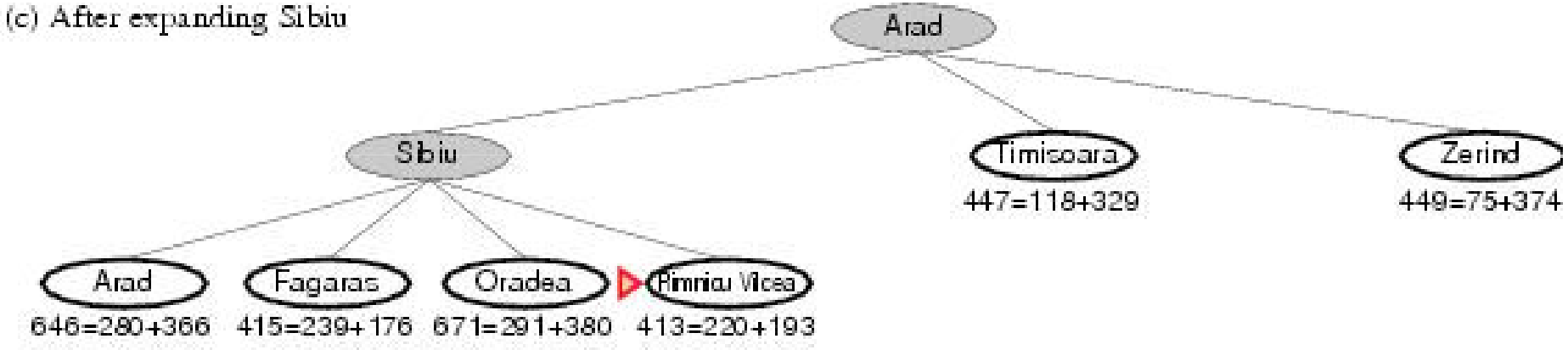
f(Timisoara)=c(Arad,Timisoara)+h(Timisoara)=118+329=447

f(Zerind)=c(Arad,Zerind)+h(Zerind)=75+374=449

بهترین انتخاب شهر Sibiu است

جست و جوی آگاهانه و اکتشاف

جستجوی A* در نقشه رومانی



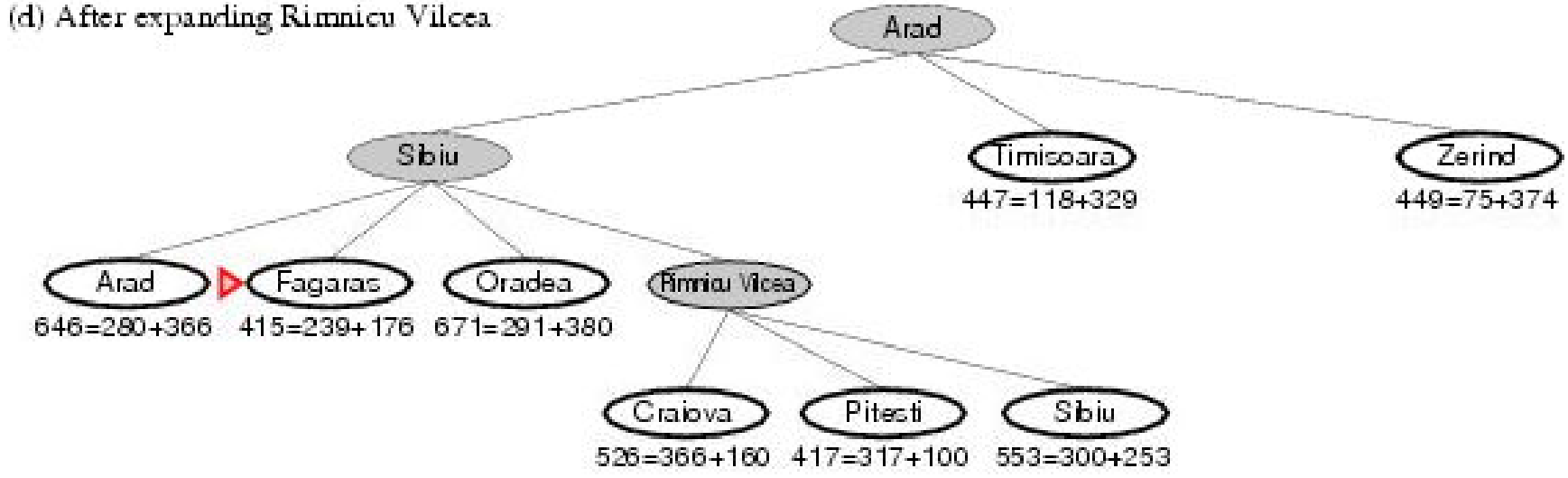
Sibiu را باز کرده و f(n) را برای هر یک از زیربرگها محاسبه میکنیم:

f(Arad)=c(Sibiu,Arad)+h(Arad)=280+366=646
f(Fagaras)=c(Sibiu,Fagaras)+h(Fagaras)=239+179=415
f(Oradea)=c(Sibiu,Oradea)+h(Oradea)=291+380=671
f(Rimnicu Vilcea)=c(Sibiu,Rimnicu Vilcea)+ h(Rimnicu Vilcea)=220+192=413

بهترین انتخاب شهر Rimnicu Vilcea است

جست و جوی آگاهانه و اکتشاف

جستجوی A* در نقشه رومانی



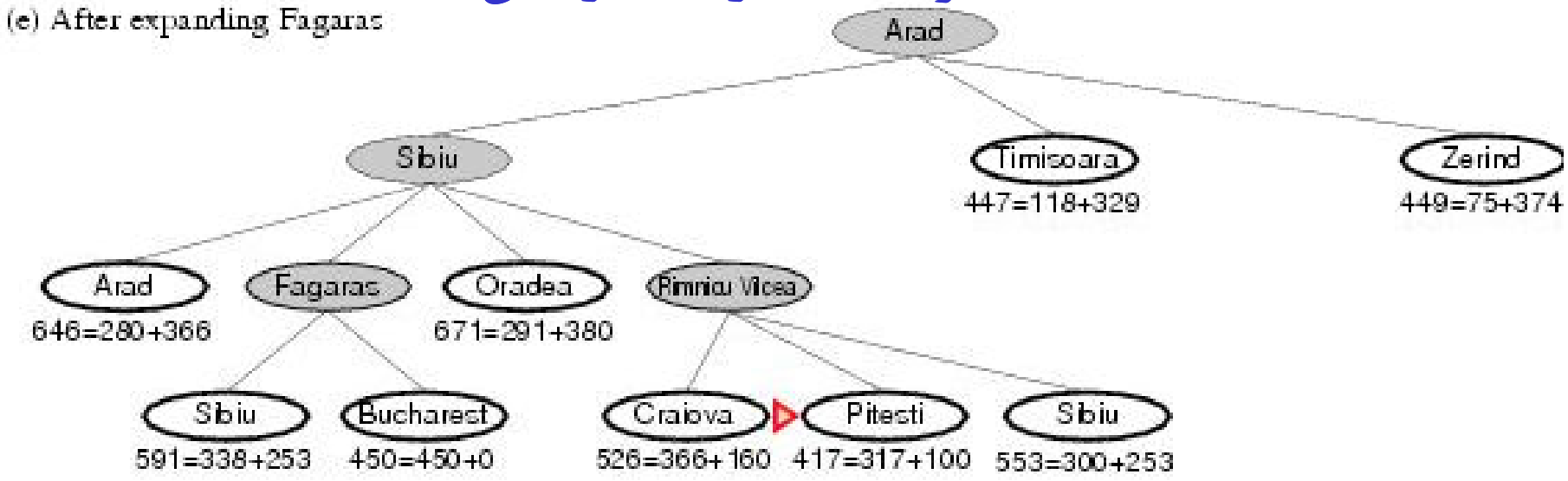
Rimnicu Vilcea را باز کرده و f(n) را برای هر یک از زیربرگها محاسبه میکنیم:

f(Craiova)=c(Rimnicu Vilcea, Craiova)+h(Craiova)=360+160=526
f(Pitesti)=c(Rimnicu Vilcea, Pitesti)+h(Pitesti)=317+100=417
f(Sibiu)=c(Rimnicu Vilcea,Sibiu)+h(Sibiu)=300+253=553

بهترین انتخاب شهر Fagaras است

جست و جوی آگاهانه و اکتشاف

جستجوی A* در نقشه رومانی



Fagaras را باز کرده و f(n) را برای هر یک از زیربرگها محاسبه میکنیم:

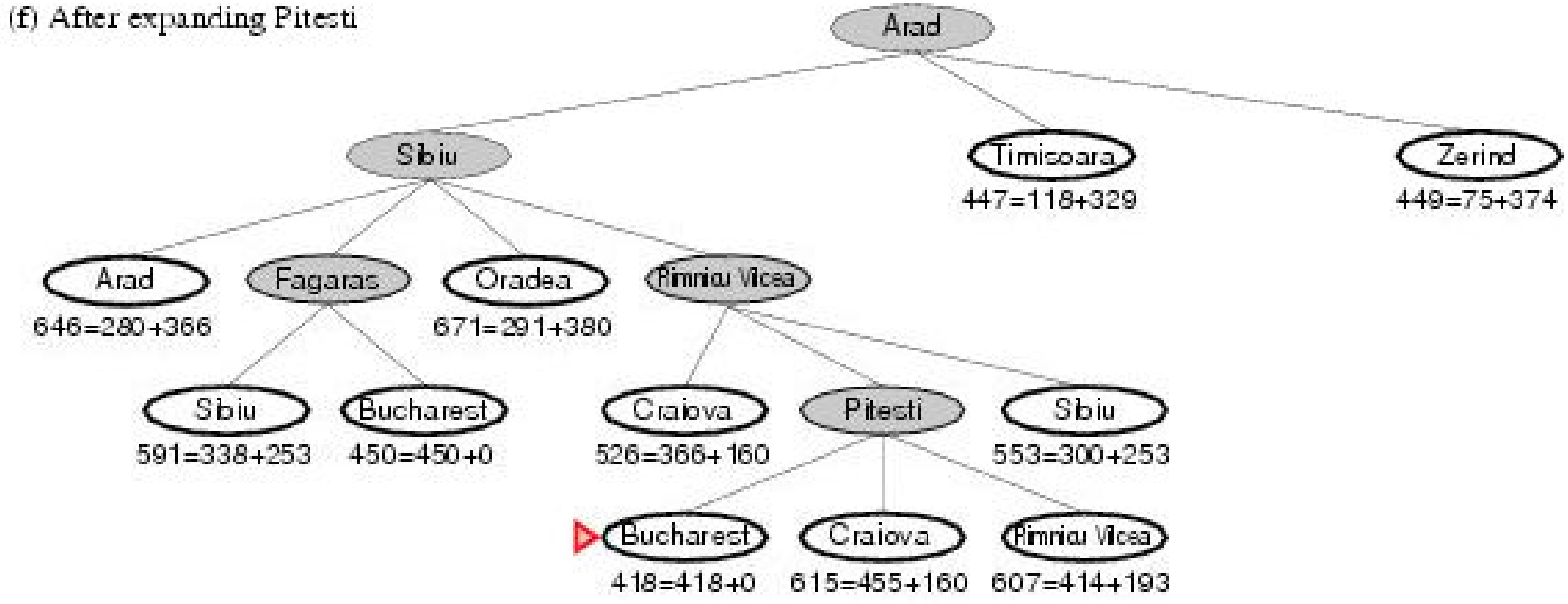
f(Sibiu)=c(Fagaras, Sibiu)+h(Sibiu)=338+253=591

f(Bucharest)=c(Fagaras,Bucharest)+h(Bucharest)=450+0=450

بهترین انتخاب شهر Pitesti !!! است

جست و جوی آگاهانه و اکتشاف

جستجوی A* در نقشه رومانی



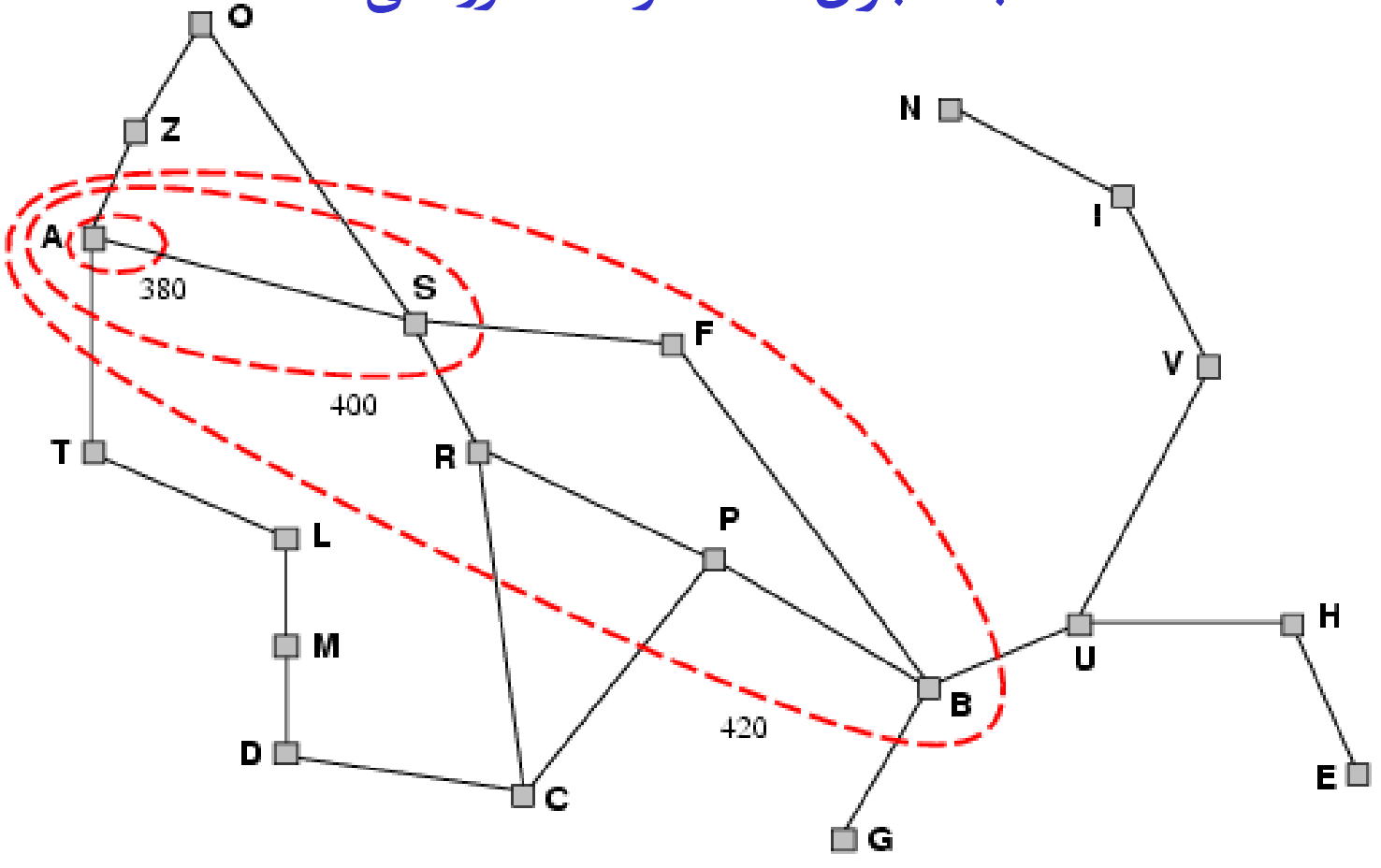
Pitesti را باز کرده و f(n) را برای هر یک از زیربرگها محاسبه میکنیم:

f(Bucharest)=c(Pitesti,Bucharest)+h(Bucharest)=418+0=418

بهترین انتخاب شهر Bucharest !!! است

جست و جوی آگاهانه و اکتشاف

جستجوی A* در نقشه رومانی



جست و جوی آگاهانه و اکتشاف

جستجوی اکتشافی با حافظه محدود IDA*

- ساده ترین راه برای کاهش حافظه مورد نیاز A* استفاده از عمیق کننده تکرار در زمینه جست و جوی اکتشافي است.
- الگوریتم عمیق کننده تکرار A* ~~IDA*~~ ←
- در جستجوی IDA* مقدار برش مورد استفاده ، عمق نیست بلکه هزینه f(g+h) است.
- IDA* برای اغلب مسئله های با هزینه های مرحله ای ، مناسب است و از سربار ناشي از نگهداری صف مرتبي از گره ها اجتناب میکند

جست و جوی آگاهانه و اکتشاف

بهترین جستجوی بازگشتي RBFS

- ساختار آن شبیه جست و جوی عمقي بازگشتي است ، اما به جای اینکه دائما به طرف پایین مسیر حرکت کند ، مقدار f مربوط به بهترین مسیر از هر جد گره فعلی را نگهداری میکند ، اگر گره فعلی از این حد تجاوز کند ، بازگشتی به عقب برمیگردد تا مسیر دیگري را انتخاب کند.
- این جستجو اگر تابع اکتشافی قابل قبولی داشته باشد ، بهینه است.
- پیچیدگي فضایي آن O(bd) است
- تعیین پیچیدگی زمانی آن به دقت تابع اکتشافی و میزان تغییر بهترین مسیر در اثر بسط گره ها بستگی دارد.

جست و جوی آگاهانه و اکتشاف

بهترين جستجوی بازگشتي RBFS

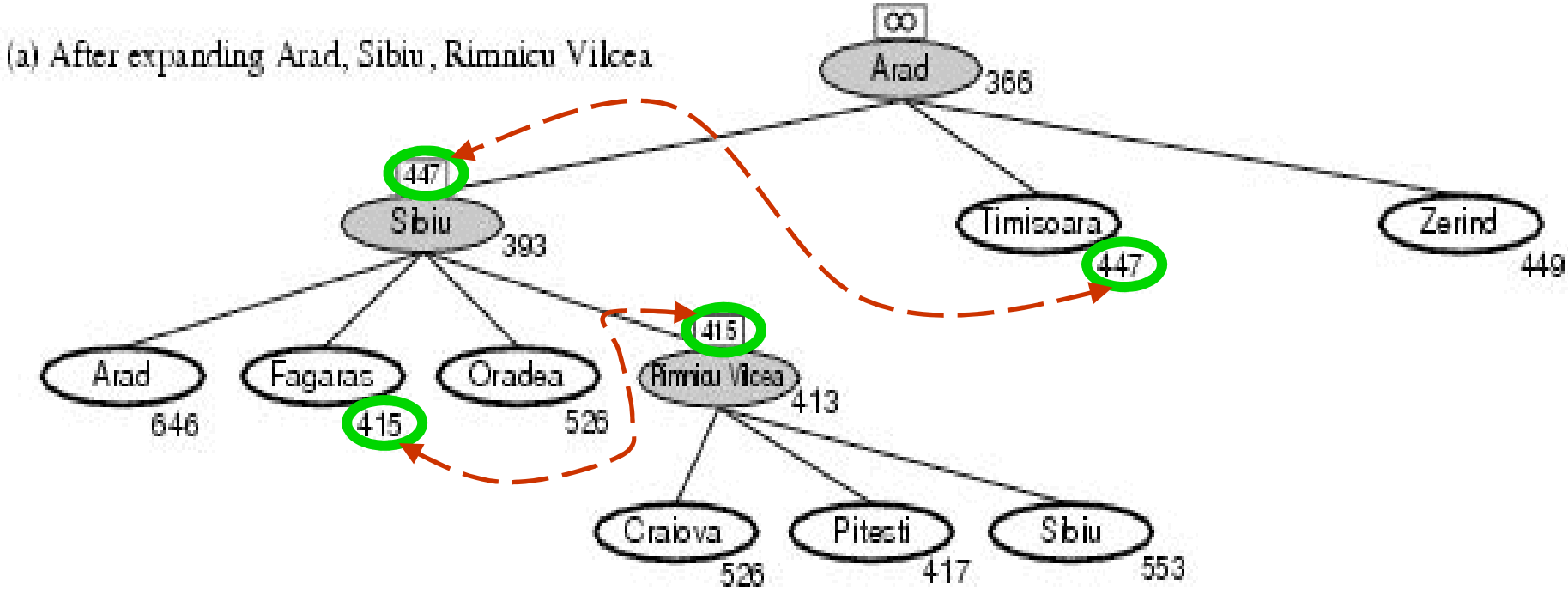
- RBFS تا حدی از IDA* کارآمدتر است ، اما گره های زیادی تولید میکند.

- IDA* و RBFS در معرض افزايش توانی پيچيدگي قرار دارند که در جست و جوی گرافها مرسوم است ، زيرا نميتوانند حالتهای تکراری را در غير از مسير فعلي بررسي کنند. لذا ، ممکن است يک حالت را چندين بار بررسي کنند.

- IDA* و RBFS از فضای اندکي استفاده ميکنند که به آنها آسيب ميرساند. IDA* بين هر تکرار فقط يک عدد را نگهداری ميکند که فعلي هزينه f است. RBFS اطلاعات بيشتری در حافظه نگهداری ميکند

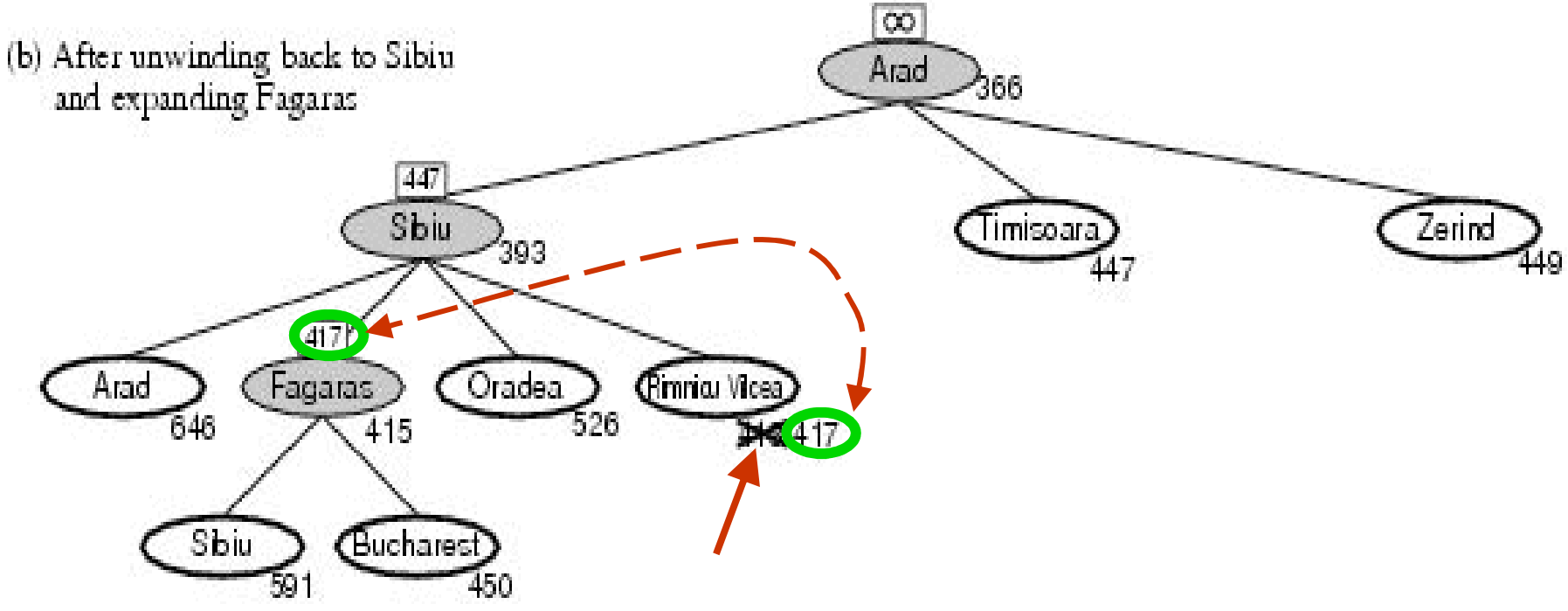
جست و جوی آگاهانه و اکتشاف

بهترین جستجوی بازگشتي در نقشه رومانی



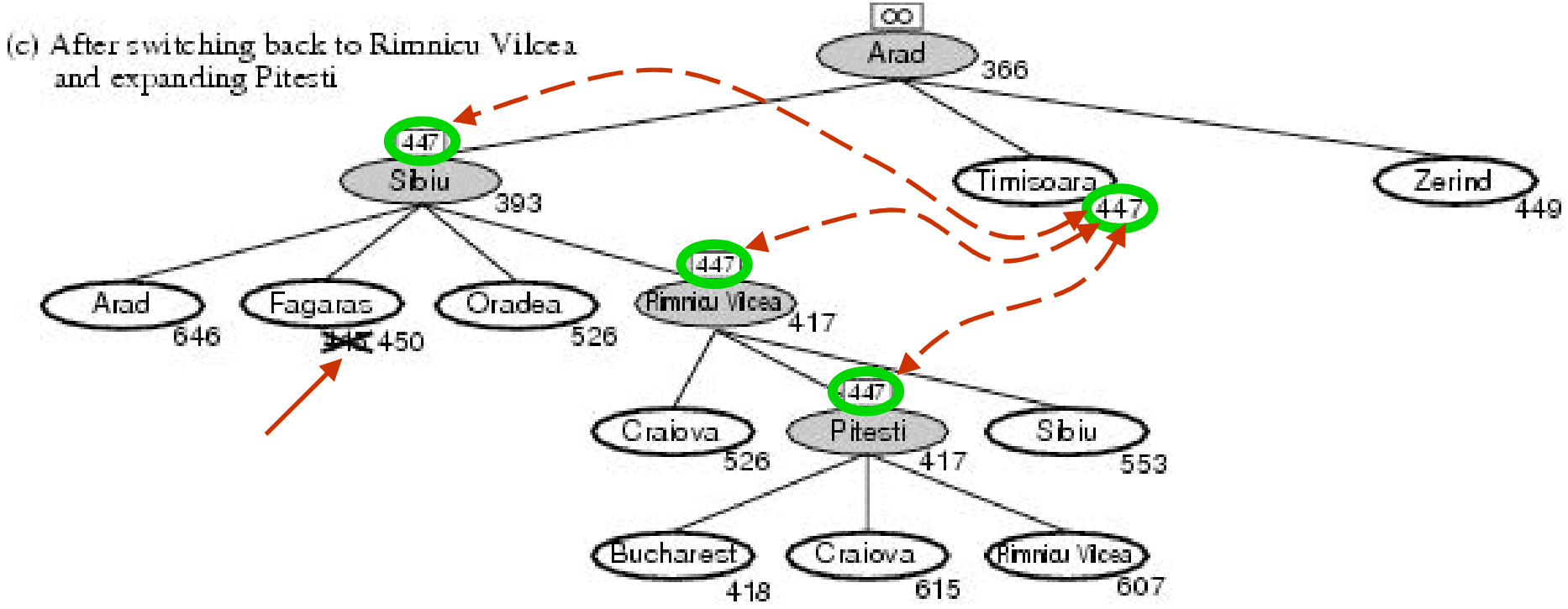
جست و جوی آگاهانه و اکتشاف

بهترین جستجوی بازگشتي در نقشه رومانی



جست و جوی آگاهانه و اکتشاف

بهترین جستجوی بازگشتي در نقشه رومانی



جست و جوی آگاهانه و اکتشاف

جستجوی حافظه محدود ساده SMA*

- SMA* بهترین برگ را بسط میدهد تا حافظه پر شود. در این نقطه بدون از بین بردن گره های قبلي نمیتواند گره جدیدی اضافه کند

- SMA* همیشه بدترین گره برگ را حذف میکند و سپس از طریق گره فراموش شده به والد آن بر میگردد. پس جد زیر درخت فراموش شده ، کیفیت بهترین مسیر را در آن زیر درخت میداند

- اگر عمق سطحی ترین گره هدف کمتر از حافظه باشد, کامل است.

- SMA* بهترین الگوریتم همه منظوره برای یافتن حلهای بهینه میباشد

جست و جوی آگاهانه و اکتشاف

جستجوی حافظه محدود ساده SMA*

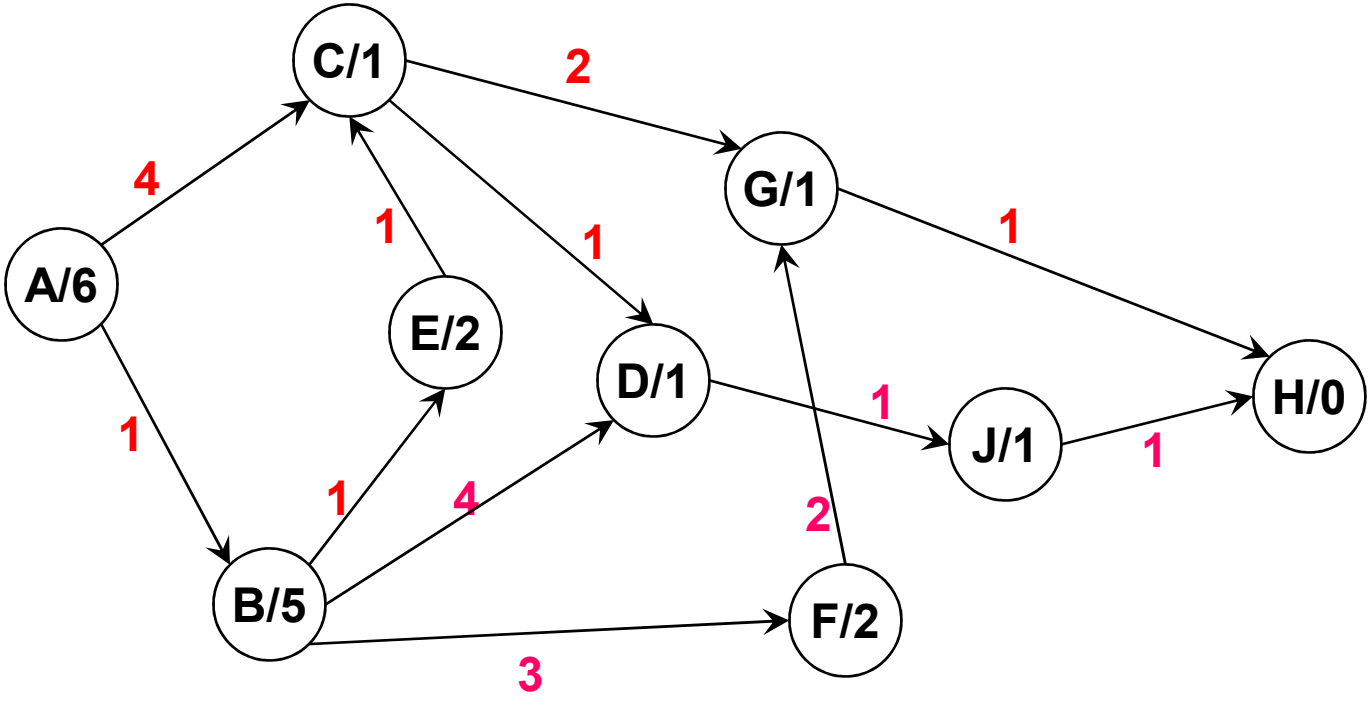
- اگر مقدار f تمام برگها یکسان باشد و الگوریتم یک گره را هم برای بسط و هم برای حذف انتخاب کند ، SMA* این مسئله را با بسط بهترین برگ جدید و حذف بهترین برگ قدیمی حل میکند

- ممکن است SMA* مجبور شود دائما بین مجموعه ای از مسیرهای حل کاندید تغییر موضع دهد ، در حالی که بخش کوچکی از هر کدام در حافظه جا شود

- محدودیتهای حافظه ممکن است مسئله ها را از نظر زمان محاسباتی ، غیر قابل حل کند.

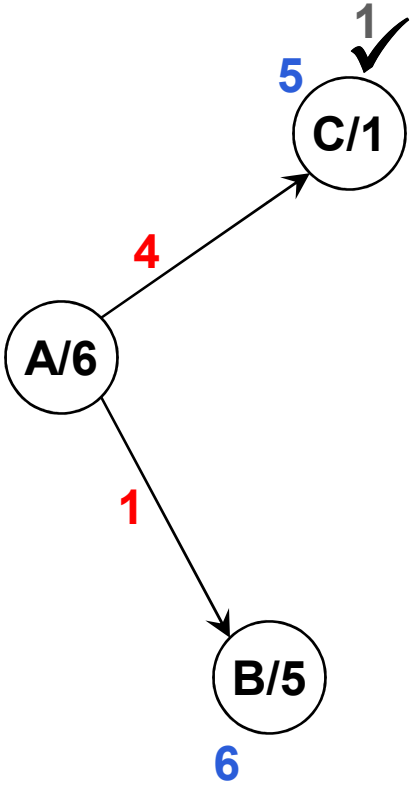
جست و جوی آگاهانه و اکتشاف

جستجوی گراف با A*



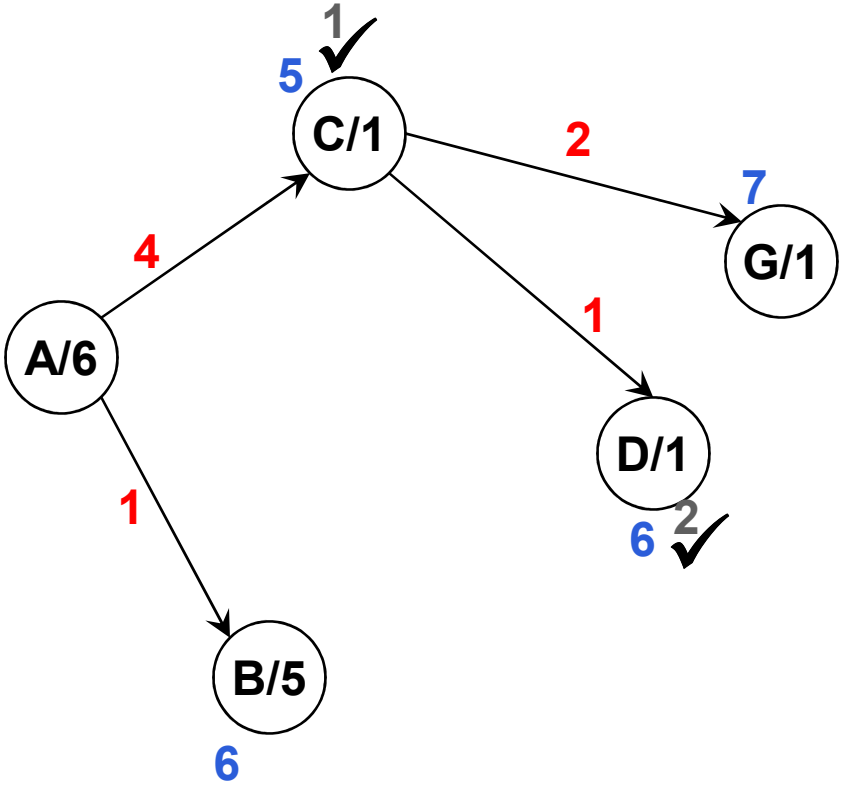
جست و جوی آگاهانه و اکتشاف

جستجوی گراف با A*



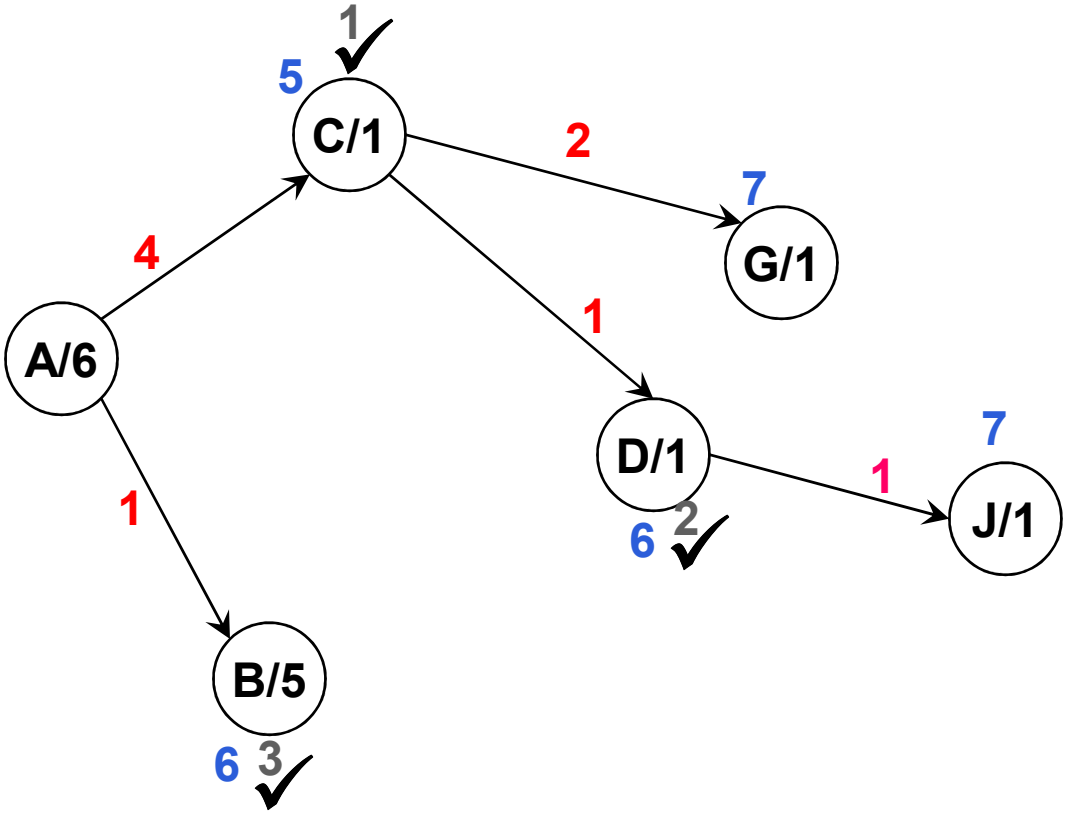
جست و جوی آگاهانه و اکتشاف

جستجوی گراف با A*



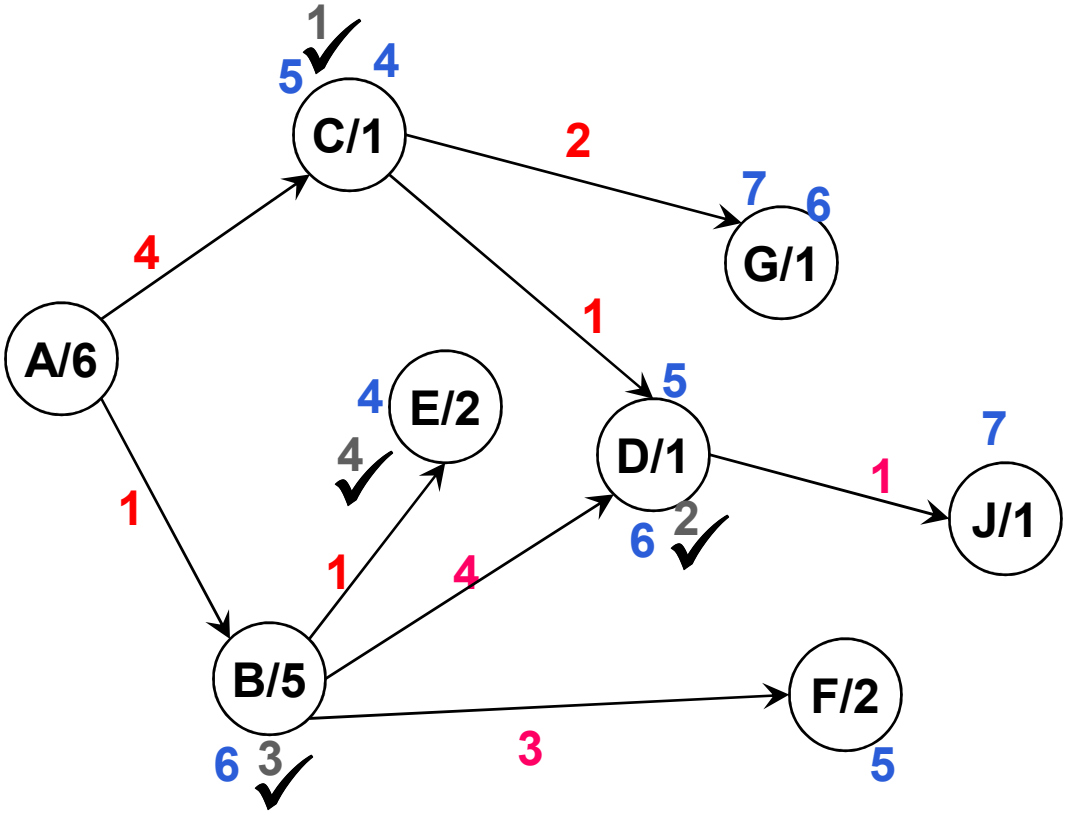
جست و جوی آگاهانه و اکتشاف

جستجوی گراف با A*



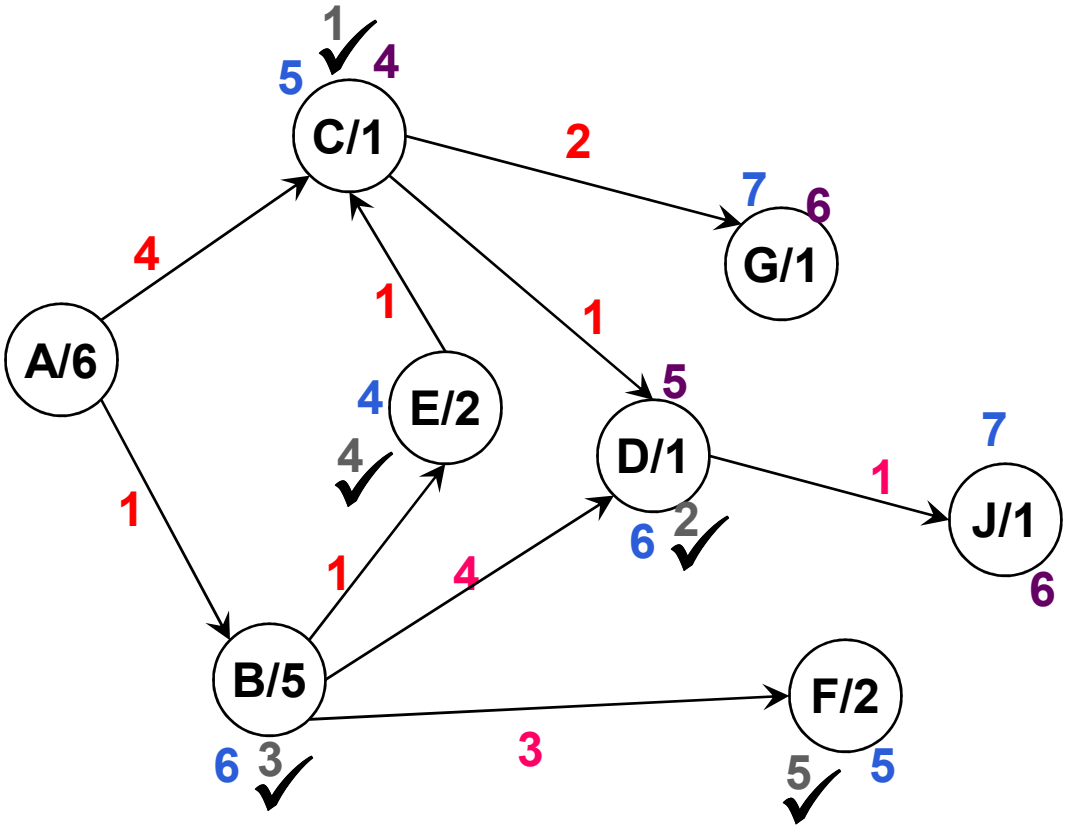
جست و جوی آگاهانه و اکتشاف

جستجوی گراف با A*



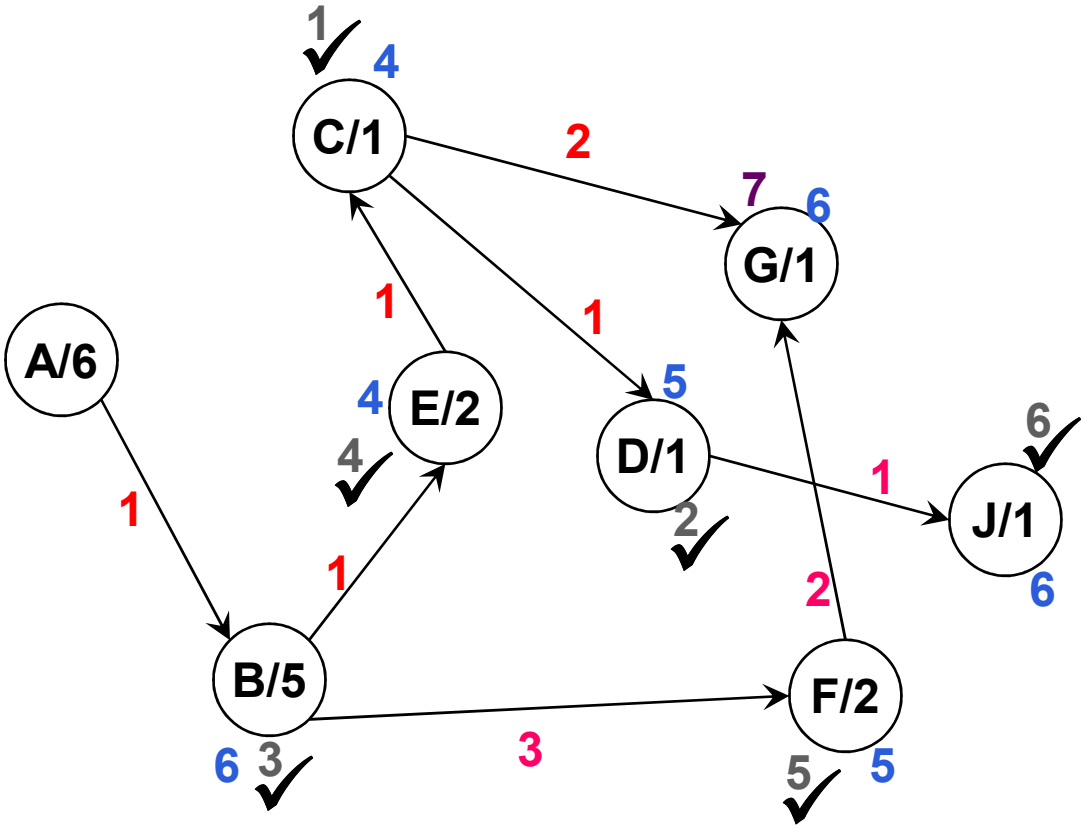
جست و جوی آگاهانه و اکتشاف

جستجوی گراف با A*



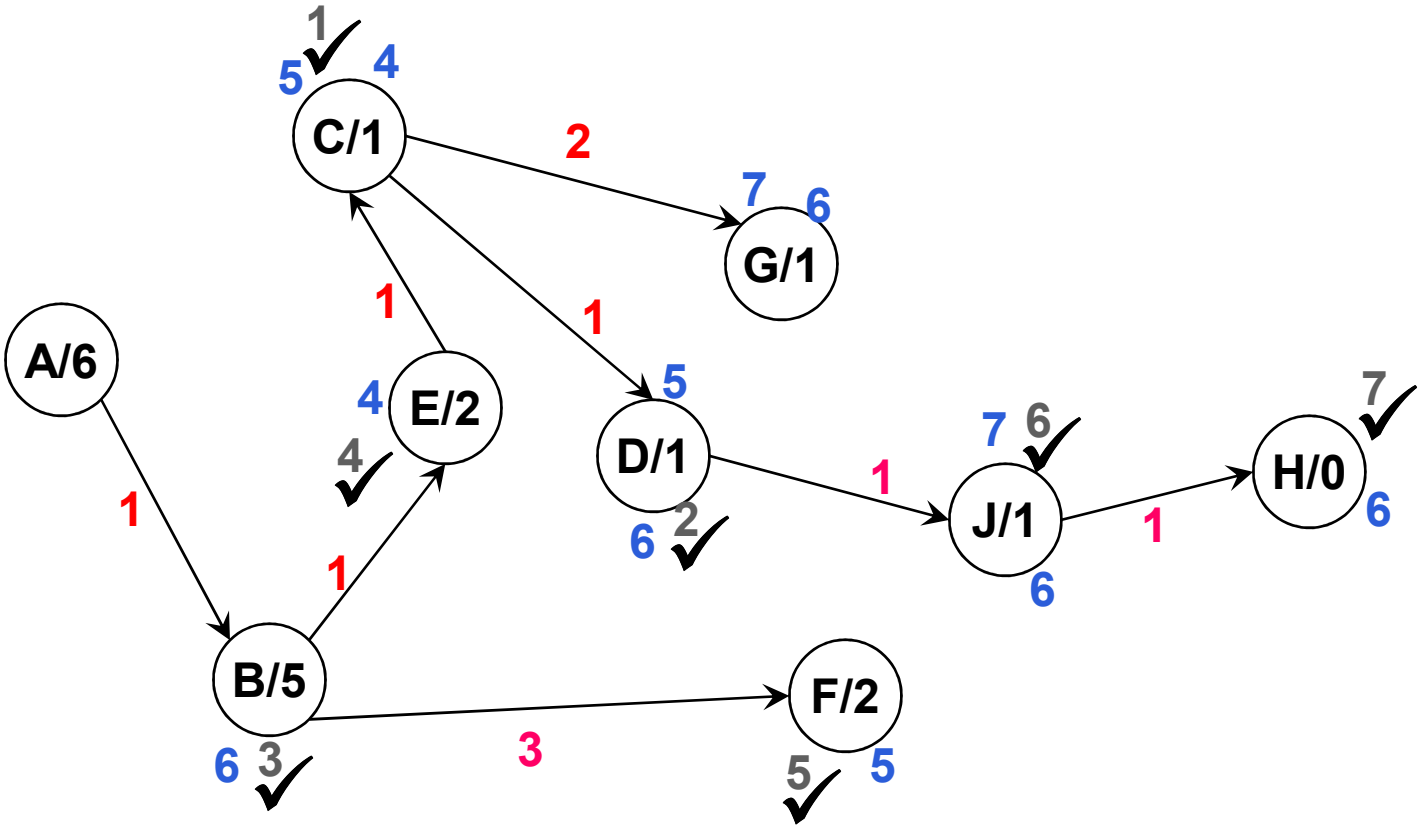
جست و جوی آگاهانه و اکتشاف

جستجوی گراف با A*



جست و جوی آگاهانه و اکتشاف

جستجوی گراف با A*



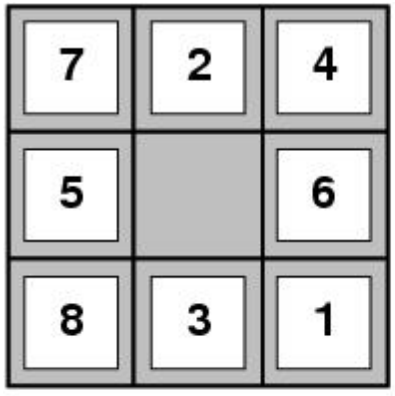
جست و جوی آگاهانه و اکتشاف

یادگیری برای جست و جوی بهتر

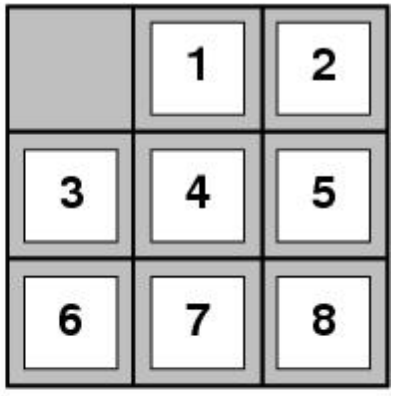
- روشهای جست و جوی قبلي ، از **روشهای ثابت** استفاده میکردند.
- عامل با استفاده از **فضای حالت فراسطحی** میتواند یاد بگیرد که بهتر جست و جو کند
- هر حالت در **فضای حالت فرا سطحی** ، حالت(محاسباتی) داخلیِ برنامه ای را تسخیر میکند که فضای **حالت سطح شیء** ، مثل رومانی را جست و جو میکند
- الگوریتم **یادگیری فراسطحی** میتواند چیزهایي را از تجربیات بیاموزد تا زیردرختهای غیر قابل قبول را کاوش نکند.
- هدف یادگیری ، **کمینه کردن کل هزینه** ، حل مسئله است

جست و جوی آگاهانه و اکتشاف

توابع اکتشافی



Start State



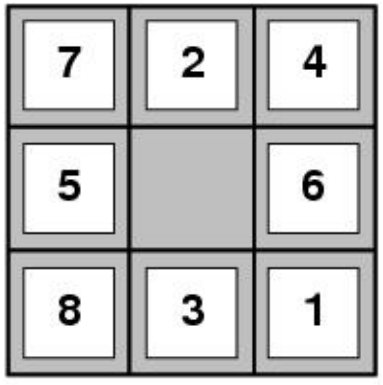
Goal State

مثال برای معمای8

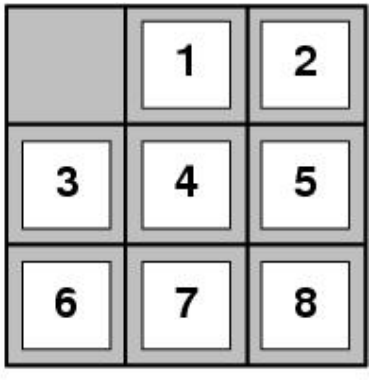
- میانگِین هزینه حل تقریبا 22 مرحله و فاکتور انشعاب در حدود 3 است.
- جست و جوی جامع تا عمق 22 : $3^{22} \approx 3.1 \times 10^{10}$
- با انتخاب یک تابع اکتشافی مناسب میتوان مراحل جستجو را کاهش داد

جست و جوی آگاهانه و اکتشاف

دو روش اکتشافي متداول برای معمای8



Start State



Goal State

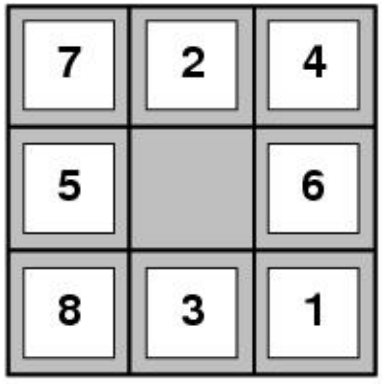
$h_1 =$ تعداد کاشیها در مکانهای نادرست

$h_1 = 8$ در حالت شروع

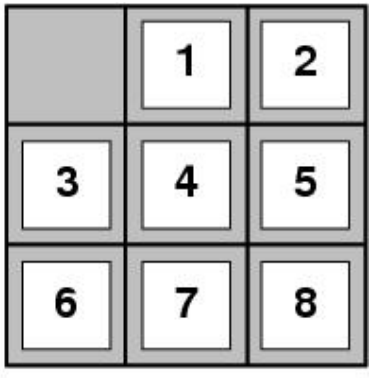
h_1 اکتشاف قابل قبولی است ، زیرا هر کاشي که در جای نامناسبی قرار دارد ، حداقل یکبار باید جابجا شود

جست و جوی آگاهانه و اکتشاف

دو روش اکتشافي متداول برای معمای8



Start State



Goal State

$h_2 =$ مجموعه فواصل کاشیها از موقعیتهای هدف آنها

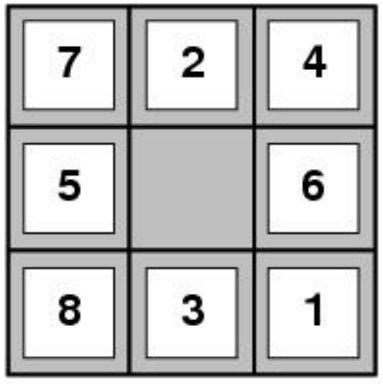
در حالت شروع

$$h_2 = 3+1+2+2+2+3+3+2 = 18$$

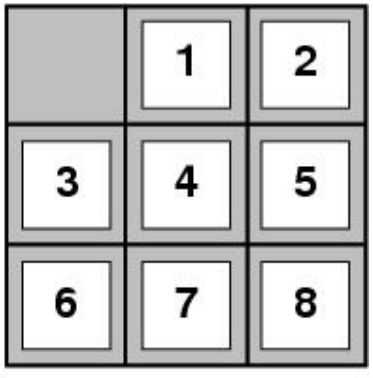
چون کاشیها نمیتوانند در امتداد قطر جا به جا شوند, فاصله ای که محاسبه میکنیم مجموع فواصل افقی و عمودی است. این فاصله را فاصله **بلوک شهر** یا فاصله **مانهاتان** مینامند.

جست و جوی آگاهانه و اکتشاف

دو روش اکتشافي متداول برای معمای8



Start State



Goal State

$h_2 =$ مجموعه فواصل کاشیها از موقعیتهای هدف آنها

h_2 قابل قبول است ، زیرا هر جابجايي که میتواند انجام گیرد ، به اندازه یک مرحله به هدف نزدیک میشود.

هیچ کدام از این برآوردها ، هزینه واقعی راه حل نیست

هزینه واقعي 36 است

جست و جوی آگاهانه و اکتشاف

اثر دقت اکتشاف بر کارايي

فاکتور انشعاب مؤثر b*

- اگر تعداد گره هايي که برای یک مسئله خاص توسط A* تولید میشود برابر با N و عمق راه حل برابر با d باشد ، آن گاه b* **فاکتور انشعابی** است که درخت یکنواختی به عمق d باید داشته باشد تا حاوی N+1 گره باشد

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

- فاکتور انشعاب مؤثر معمولاً برای **مسئله های سخت ثابت** است
- اندازه گیریهای تجربي b* بر روی مجموعه کوچکي از مسئله ها میتواند **راهنمای خوبي برای مفید بودن** اکتشاف باشد
- مقدار b* در اکتشافي با طراحي خوب ، نزدیک 1 است

جست و جوی آگاهانه و اکتشاف

اثر دقت اکتشاف بر کارايي

	هزینه جست و جو			فاکتور انشعاب مؤثر		
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	6384	39	25	2.80	1.33	1.24
10	47127	93	39	2.79	1.38	1.22
12	3644035	227	73	2.78	1.42	1.24
14	–	539	113	–	1.44	1.23
16	–	1301	211	–	1.45	1.25
18	–	3056	363	–	1.46	1.26
20	–	7276	676	–	1.47	1.27
22	–	18094	1219	–	1.48	1.28
24	–	39135	1641	–	1.48	1.26

میانگین گره های بسط یافته در جستجوی IDS و A* و فاکتور انشعاب مؤثر با استفاده از h1 و h2

جست و جوی آگاهانه و اکتشاف

اثر دقت اکتشاف بر کارايي

- اگر برای هر گره n داشته باشیم: h2(n) >= h1(n)
  - h2 بر h1 غالب است
  - غالب بودن مستقيما به کارايي ترجمه ميشود
  - تعداد گره هايي که با بکارگيری h2 بسط داده ميشود ، هرگز بيش از بکارگيری h1 نيست

همیشه بهتر است از تابع اکتشافی با مقادیر بزرگ استفاده کرد ، به شرطی که زمان محاسبه اکتشاف ، خيلي بزرگ نباشد

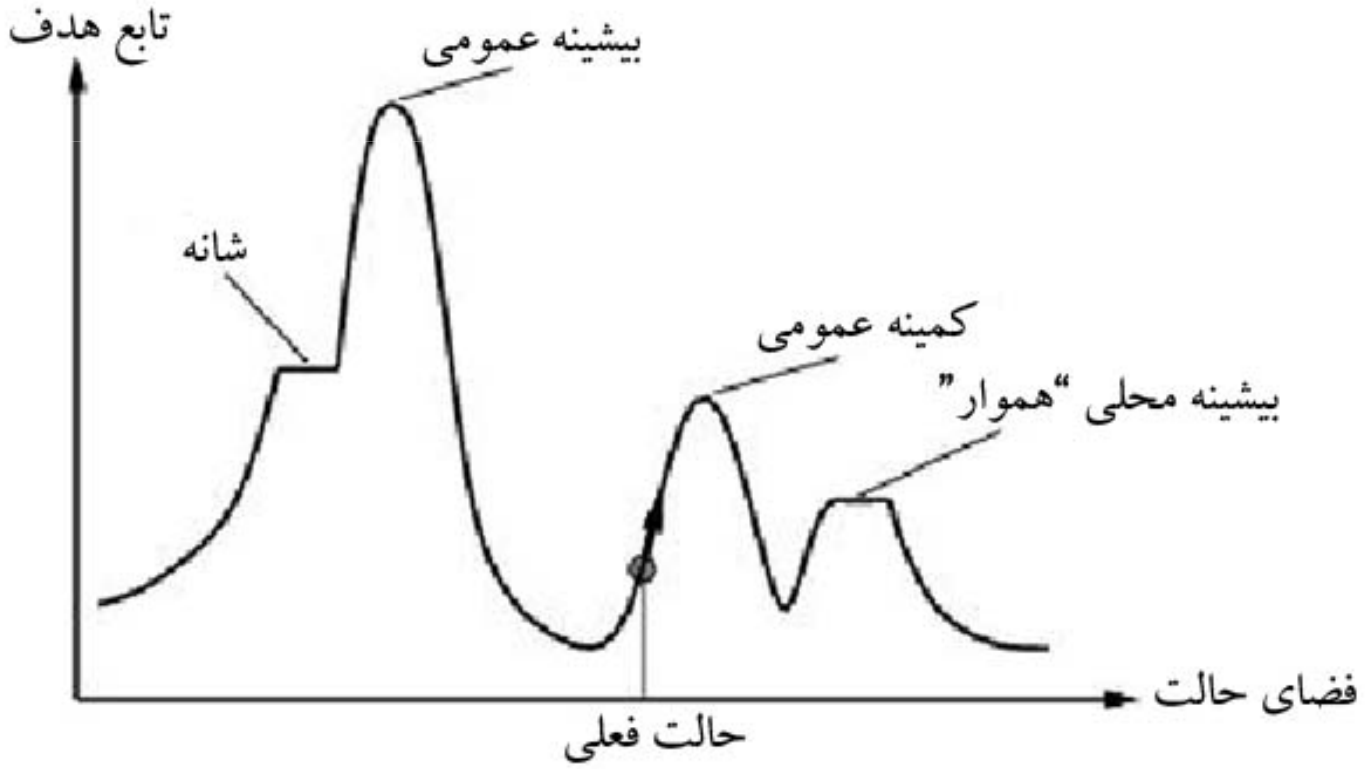
جست و جوی آگاهانه و اکتشاف

الگوریتم های جست و جوی محلی و بهینه سازی

- الگوریتم های قبلی ، فضای جست و جو را به طور سیستماتیک بررسی میکنند
  - تا رسیدن به هدف یک یا چند مسیر نگهداری میشوند
  - مسیر رسیدن به هدف ، راه حل مسئله را تشکیل میدهد
- در الگوریتم های محلی مسیر رسیدن به هدف مهم نیست
  - مثال: مسئله 8 وزیر
- دو امتیاز عمده جست و جوهای محلي
  - استفاده از حافظه کمکی
  - ارائه راه حلهای منطقي در فضاهای بزرگ و نامتناهی
- این الگوریتمها برای حل مسائل بهینه سازی نیز مفیدند
  - یافتن بهترین حالت بر اساس تابع هدف

جست و جوی آگاهانه و اکتشاف

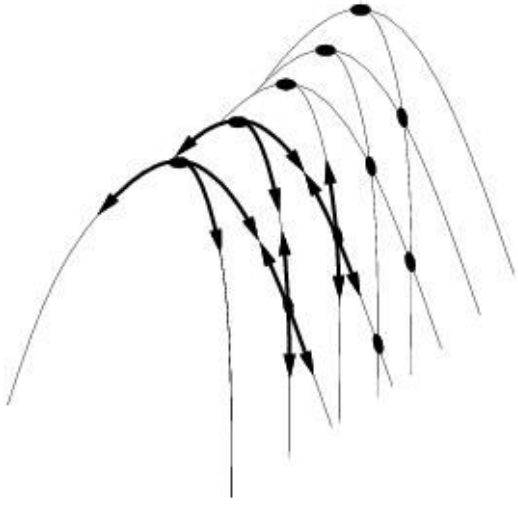
الگوریتم های جست و جوی محلی و بهینه سازی



جست و جوی آگاهانه و اکتشاف

جست و جوی تپه نوردی

- حلقه اي كه در جهت افزايش مقدار حركت ميكند(بطرف بالای تپه)
  - رسیدن به بلندترین قله در همسایگی حالت فعلی ، شرط خاتمه است.
- ساختمان داده گره فعلی ، فقط حالت و مقدار تابع هدف را نگه میدارد
- جست و جوی محلی حریصانه نیز نام دارد
  - بدون فكر قبلي حالت همسايه خوبي را انتخاب ميكند
- تپه نوردی به دلایل زیر میتواند متوقف شود:
  - بيشينه محلي
  - برآمدگي ها
  - فلات



جست و جوی آگاهانه و اکتشاف

جست و جوی تپه نوردی

- انواع تپه نوردی:
  - تپه نوردی غیرقطعی ، تپه نوردی اولین انتخاب ، تپه نوردی شروع مجدد تصادفی

مثال: مسئله 8 وزیر

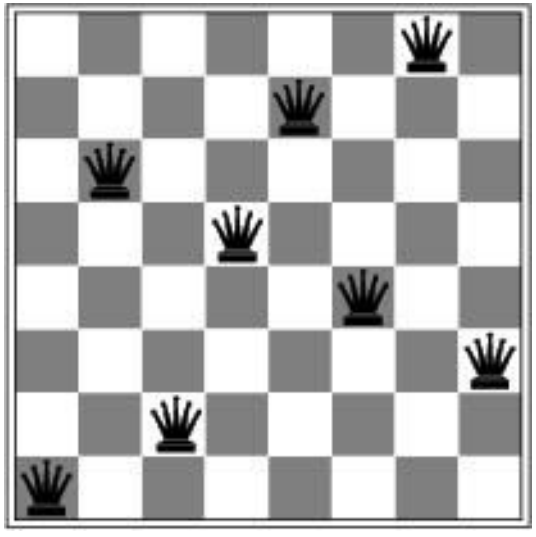
- مسئله 8 وزیر با استفاده از فرمولبندی حالت کامل
  - در هر حالت 8 وزیر در صفحه قرار دارند
- تابع جانشین: انتقال یک وزیر به مربع دیگر در همان ستون
- تابع اکتشاف: جفت وزیرهایی که نسبت به هم گارد دارند
  - مستقیم یا غیر مستقیم

جست و جوی آگاهانه و اکتشاف

مثال جست و جوی تپه نوردی

الف

ب



الف- حالت با هزینه h=17 که مقدار h را برای هر جانشین نشان میدهد

ب- کمینه محلی در فضای حالت 8 وزیر ؛ h=1

جست و جوی آگاهانه و اکتشاف

جست و جوی شبیه سازی حرارت

- تپه نوردی مرکب با حرکت تصادفی
- شبیه سازی حرارت: حرارت با درجه بالا و به تدریج سرد کردن
- مقایسه با حرکت توپ
  - توپ در فرود از تپه به عمیق ترین شکاف میرود
  - با تکان دادن سطح توپ از بیشینه محلي خارج میشود
  - با تکان شدید شروع(دمای زیاد)
  - بتدریج تکان کاهش(به دمای پایین تر)
- با کاهش زمانبندی دما به تدریج ، الگوریتم یک بهینه عمومی را مي یابد

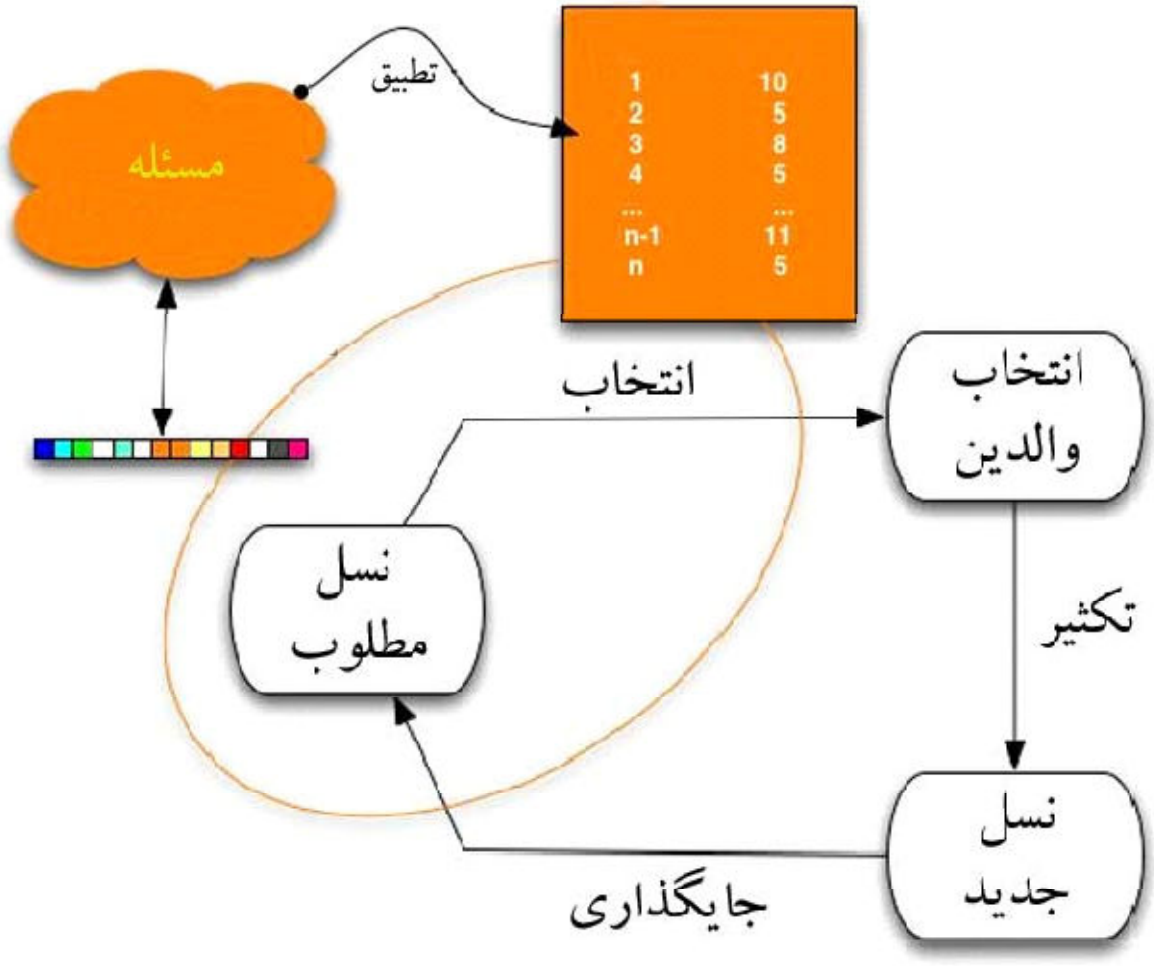
جست و جوی آگاهانه و اکتشاف

جست و جوی پرتو محلي

- به جای یک حالت ، k حالت را نگهداری میکند
  - حالت اولیه: k حالت تصادفی
  - گام بعد: جانشین همه k حالت تولید میشود
  - اگر يكي از جانشین ها هدف بود ، تمام میشود
  - وگر نه بهترین جانشین را انتخاب کرده ، تکرار میکند
- تفاوت عمده با جستجوی شروع مجدد تصادفی
  - در جست و جوی شروع مجدد تصادفی ، هر فرایند مستقل از بقیه اجرا میشود
  - در جست و جوی پرتو محلی ، اطلاعات مفیدی بین k فرایند موازی مبادله میشود
- جست و جوی پرتو غیرقطعی
  - به جای انتخاب بهترین k از جانشینها ، k جانشین تصادفی را بطوریکه احتمال انتخاب يكي تابعی صعودی از مقدارش باشد ، انتخاب میکند

جست و جوی آگاهانه و اکتشاف

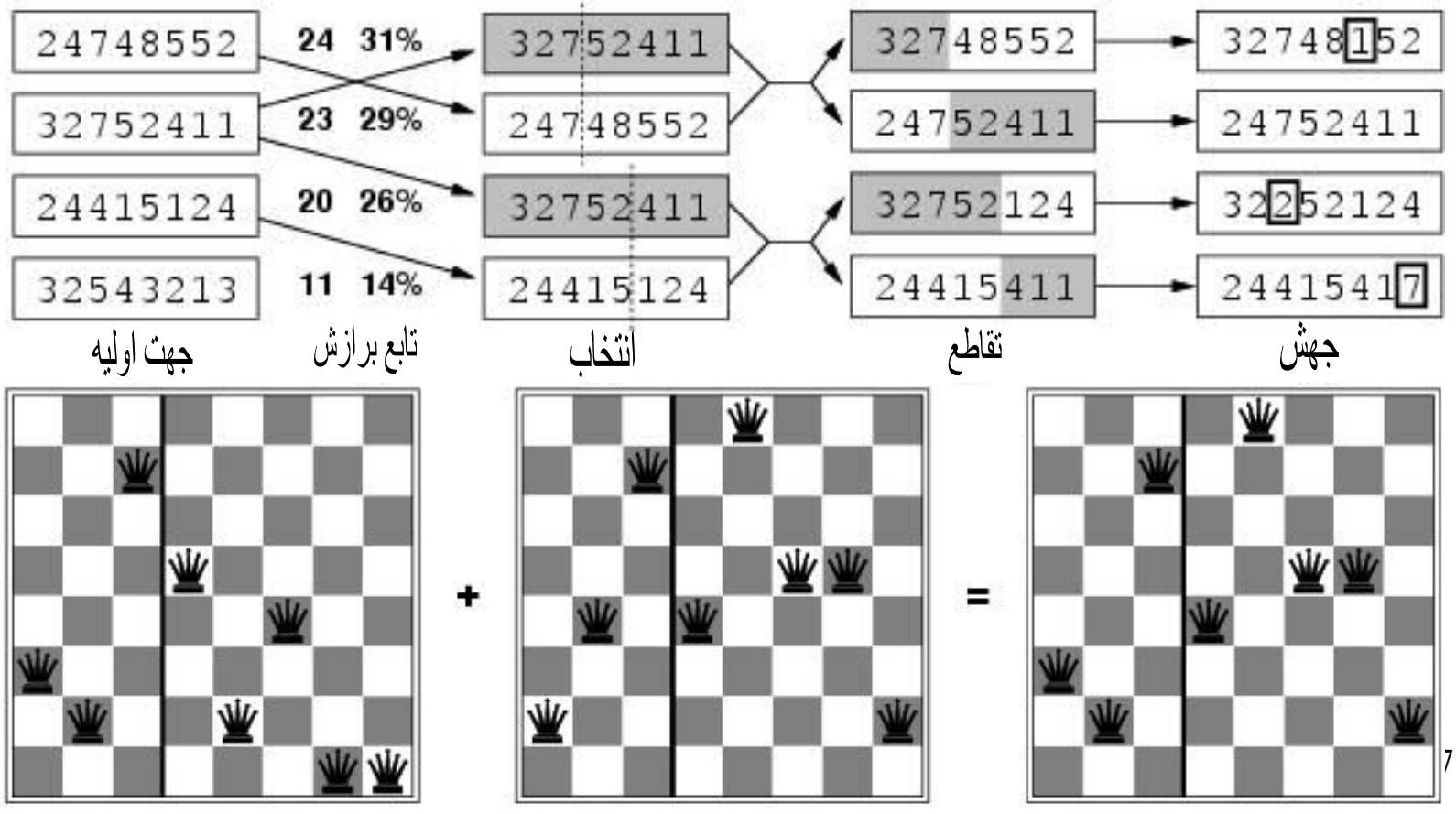
الگوریتم های ژنتیک



شکلی از جست و جوی پرتو غیر قطعی که حالتهای جانشین از طریق ترکیب دو حالت والد تولید میشود

جست و جوی آگاهانه و اکتشاف

الگوریتم های ژنتیک



جست و جوی آگاهانه و اکتشاف

جست و جوی محلی در فضاهای پیوسته

- گسسته در مقابل محیط های پیوسته
  - در فضاهای پیوسته ، تابع جانشین در اغلب موارد ، حالتهای نامتناهی را بر میگرداند
- حل مسئله:
  - گسسته کردن همسایه هر حالت
  - استفاده از شیب منظره

$$\mathbf{x} \leftarrow \mathbf{x} + \alpha \nabla f \quad where\ \nabla f = \left\{ \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots \right\}$$

  - روش نیوتن رافسون

جست و جوی آگاهانه و اکتشاف

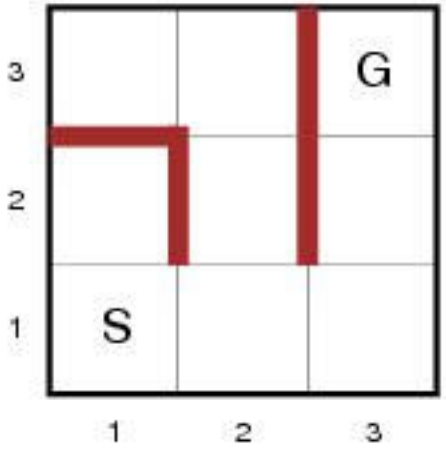
عاملهای جست و جوی Online و محیطهای ناشناخته

- تا به حال همه الگوریتمها برون خطي بودند
  - برون خطی(Offline):راه حل قبل از اجرا مشخص است
  - درون خطی(Online):با یک در میان کردن محاسبات و فعالیت عمل میکند
- جستجوی درون خطی در محیطهای پویا و نیمه پویا مفید است
  - آنچه را که باید واقعا اتفاق بیفتد ، در نظر گرفته نمیشود
- جست و جوی درون خطی ایده ضروری برای **مسئله اکتشاف** است
  - فعالیتها و حالتها برای عامل مشخص نیستند
  - مثال:قرار گرفتن روبات در محیطي جدید, نوزاد تازه بدنیا آمده

جست و جوی آگاهانه و اکتشاف

مسئله های جست و جوی Online

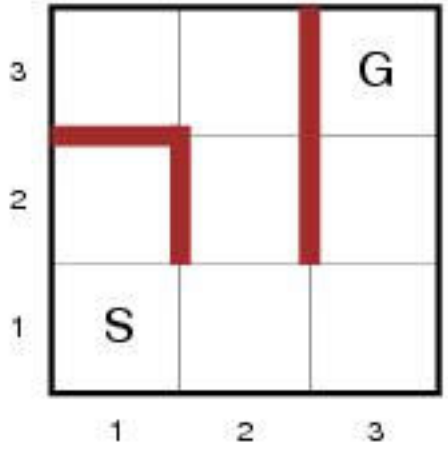
- اطلاعات عامل
  - Actions(s): لیستی از فعالیتهای مجاز در حالت s
  - تابع هزینه مرحله ای c(s,a,s’): استفاده وقتی که بداند s’ نتیجه است
  - Goal-Test(s): آزمون هدف
- عامل حالت ملاقات شده قبلی را تشخیص میدهد
- فعالیتها قطعی اند
- دسترسی به تابع اکتشافی قابل قبول h(s)



جست و جوی آگاهانه و اکتشاف

مسئله های جست و جوی Online

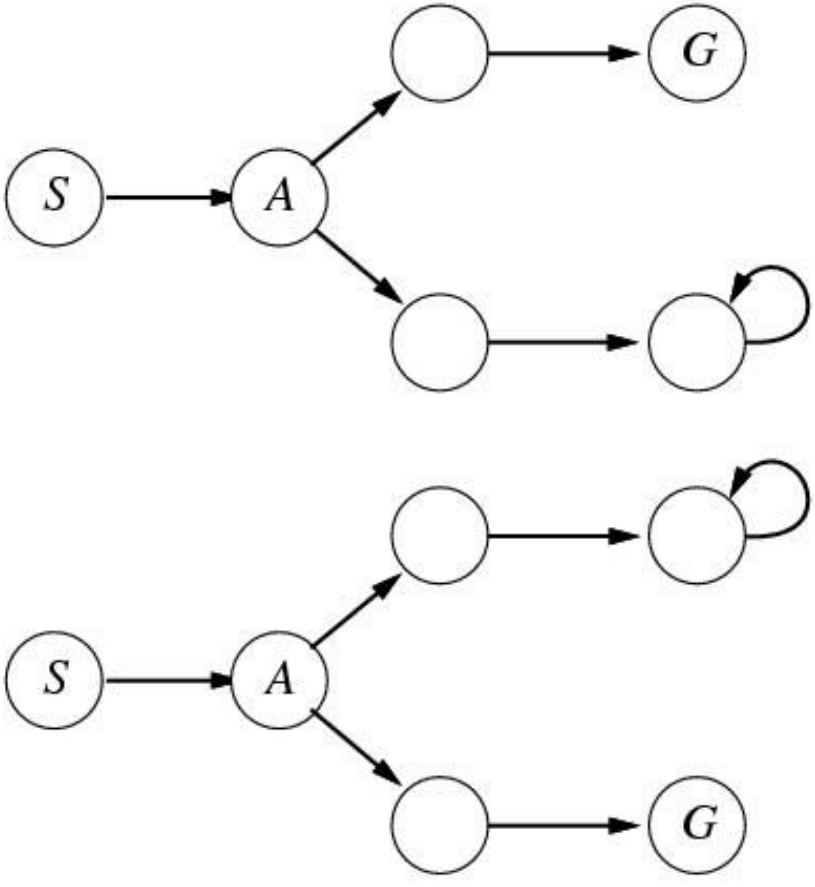
- هدف: رسیدن به G با کمترین هزینه
- هزینه: مجموع هزینه های مراحل مسیری است که عامل طی میکند
- نسبت رقابتی: مقایسه هزینه با هزینه مسیری که اگر عامل فضای حالت را از قبل میشناخت ، طی میکرد



- در بعض موارد ، بهترین نسبت رقابتی نامتناهی است
- ممکن است جستجو به یک حالت بن بست برسد که نتوان از طریق آن به هدف رسید

جست و جوی آگاهانه و اکتشاف

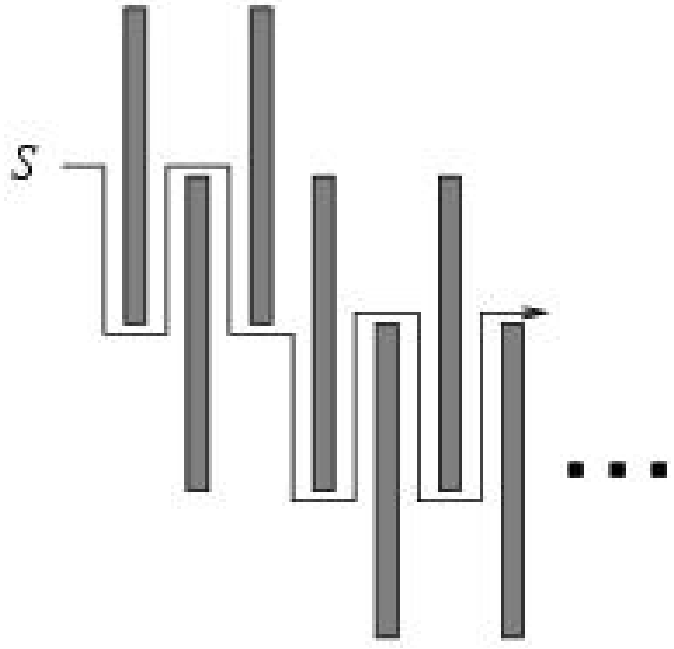
مسئله های جست و جوی Online



دو فضـای حالـت کـه عامـل جسـت و جـوی Online را بـه بـن بسـت میرسانند. هر عاملي حداقل در يكي از اين دو فضا شكست می خورد

جست و جوی آگاهانه و اکتشاف

مسئله های جست و جوی Online



یک محیط دو بعدی که موجب میشود عامل جست و جوی Online مسیر دلخواه ناکارآمدی را برای رسیدن به هدف حل کند

هوش مصنوعي

فصل پنجم

مسائل ارضای محدودیت

هوش مصنوعي Artificial Intelligence

فهرست

- ارضای محدودیت چیست ؟
- جست و جوی عقبگرد برای CSP
- بررسی پیشرو
- پخش محدودیت

مسائل ارضای محدودیت

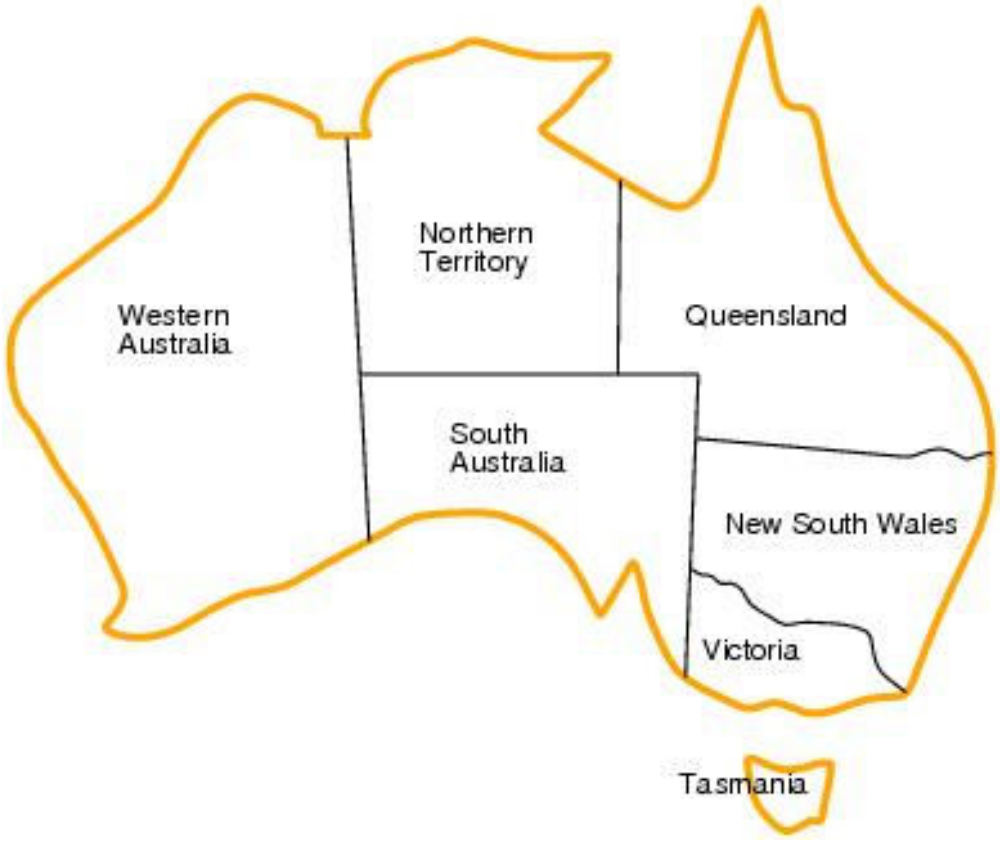
ارضای محدودیت (CSP) چیست ؟

- مجموعه متناهی از متغیرها ؛ $X_1, X_2, \ldots, X_n$
- مجموعه متناهی از محدودیتها ؛ $C_1, C_2, \ldots, C_m$
- دامنه های ناتهی برای هر یک از متغیرها ؛ $D_{X1}, D_{X2}, \ldots, D_{Xn}$
- هر محدودیت C_i زیرمجموعه ای از متغیرها و ترکیبهای ممکنی از مقادیر برای آن زیرمجموعه ها

- هر حالت با **انتساب** مقادیری به چند یا تمام متغیرها تعریف میشود
- انتسابی که هیچ محدودیتی را نقض نکند ، انتساب **سازگار** نام دارد
- انتساب **کامل** آن است که هر متغیری در آن باشد
- راه حل CSP یک انتساب کامل است اگر تمام محدودیتها را برآورده کند
- بعضی از CSPها به راه حلهایی نیاز دارند که **تابع هدف** را بیشینه کنند

مسائل ارضای محدودیت

مثال CSP: رنگ آمیزی نقشه



متغیرها: WA, NT, Q, NSW, V, SA, T

دامنه: D_i = {آبی ، سبز ، قرمز}

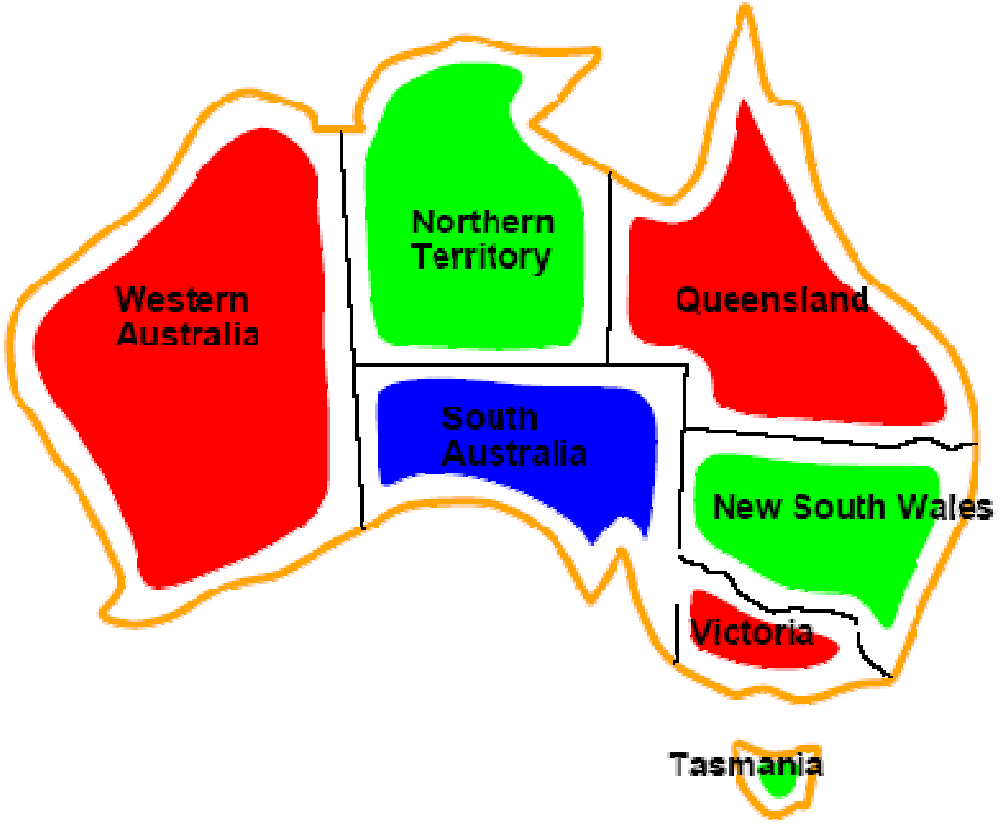
محدودیتها: دو منطقه مجاور ، همرنگ نیستند

مثال: $WA \neq NT$ یعنی (WA,NT) عضو

{(قرمز,سبز),(قرمز,آبی),(سبز,قرمز) ،

(سبز,آبی),(آبی,قرمز),(آبی,سبز)}

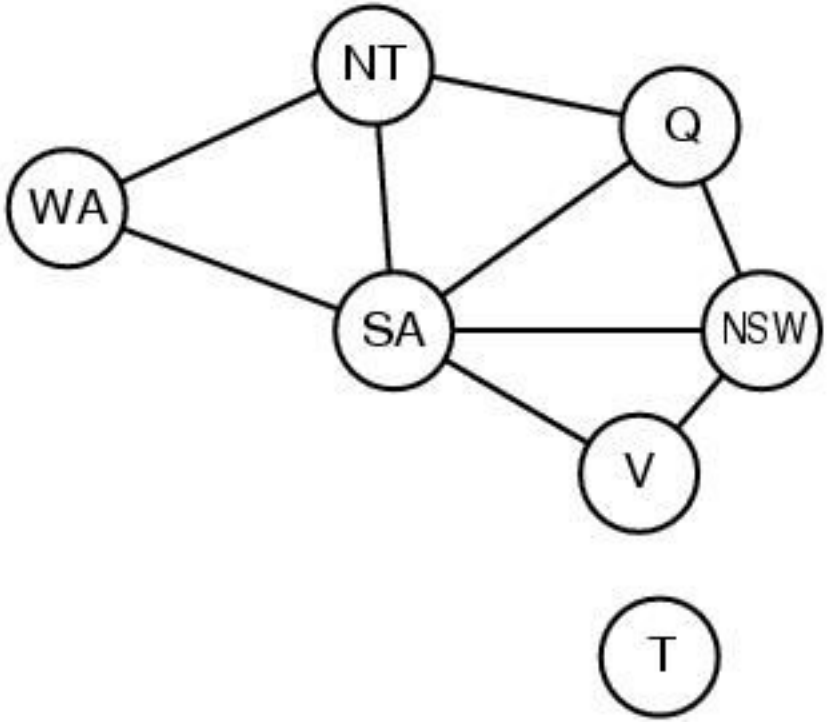
مسائل ارضای محدودیت



راه حل انتساب مقادیری است که محدودیتها را ارضا کند

مسائل ارضای محدودیت

گراف محدودیت

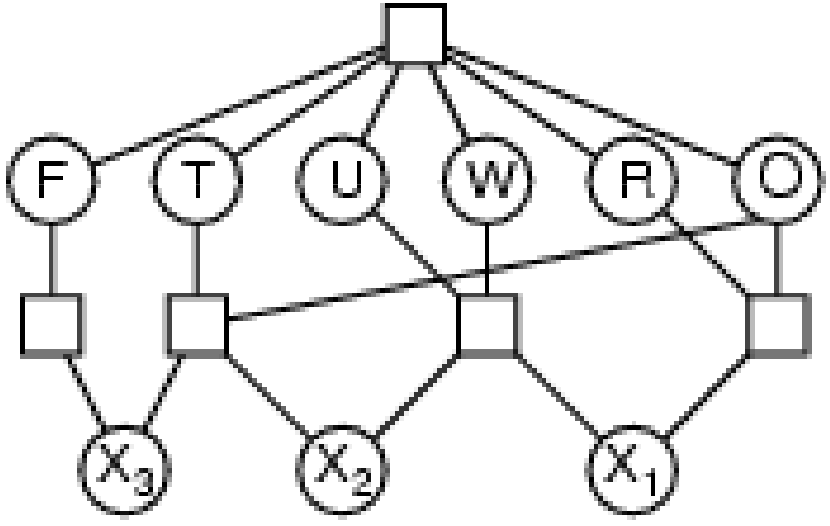


- در گراف محدودیت:
  - گره ها: متغیرها
  - یالها: محدودیتها
- گراف برای ساده تر کردن جست و جو بکار میرود

مسائل ارضای محدودیت

مثال CSP: رمزنگاری

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



متغیرها: F,T,U,W,R,O,X1,X2,X3 دامنه: {0و1و2و3و4و5و6و7و8و9}

محدودیتها: F,T,U,R,O,W مخالفند - O+O=R+10.X1 - ...

مسائل ارضای محدودیت

- نمایش حالتها در CSP از الگوی استانداردی پیروی میکند
- برای CSP میتوان فرمول بندی افزایشي ارائه کرد:
  - حالت اولیه: انتساب خالی{} که در آن ، هیچ متغیری مقدار ندارد
  - تابع جانشین: انتساب یک مقدار به هر متغیر فاقد مقدار ، به شرطی که با متغیرهایي که قبلا مقدار گرفتند ، متضاد نباشند
  - آزمون هدف: انتساب فعلی کامل است
  - هزینه مسیر: هزینه ثابت برای هر مرحله

مسائل ارضای محدودیت

جست و جوی عقبگرد برای CSP

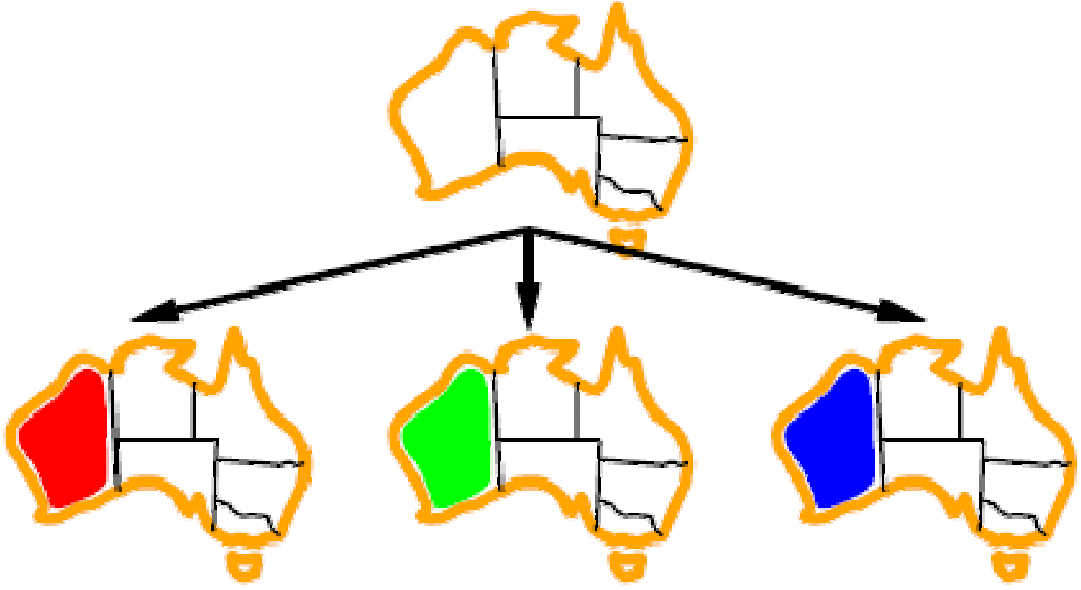
- جست و جوی عمقي
- انتخاب مقادیر یک متغیر در هر زمان و عقبگرد در صورت عدم وجود مقداری معتبر برای انتساب به متغیر
- یک الگوریتم ناآگاهانه است
  - برای مسئله های بزرگ کارآمد نیست

مسائل ارضای محدودیت

مثال جست و جوی عقبگرد برای CSP

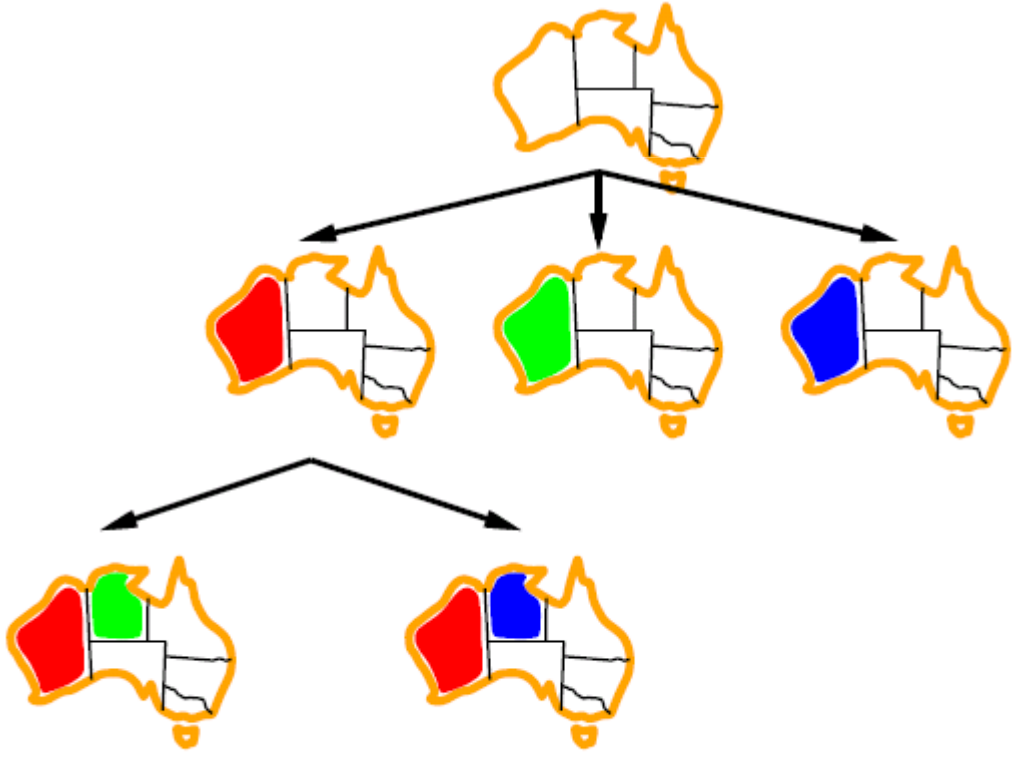
مسائل ارضای محدودیت

مثال جست و جوی عقبگرد برای CSP



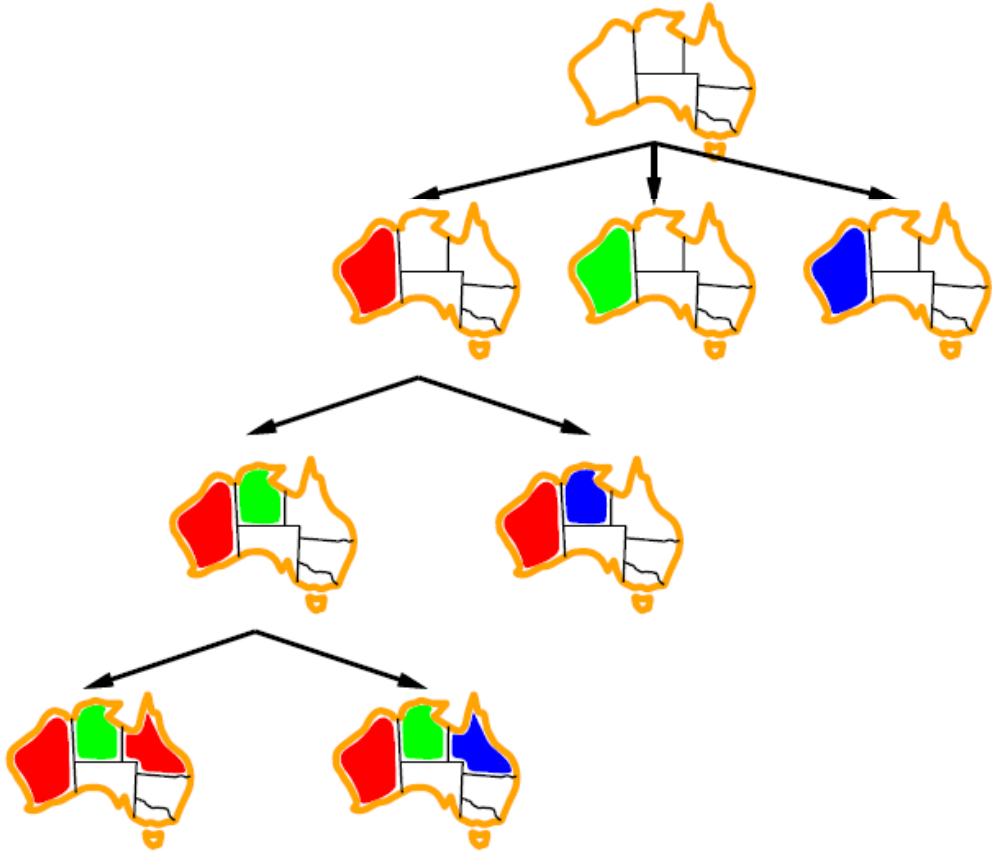
مسائل ارضای محدودیت

مثال جست و جوی عقبگرد برای CSP



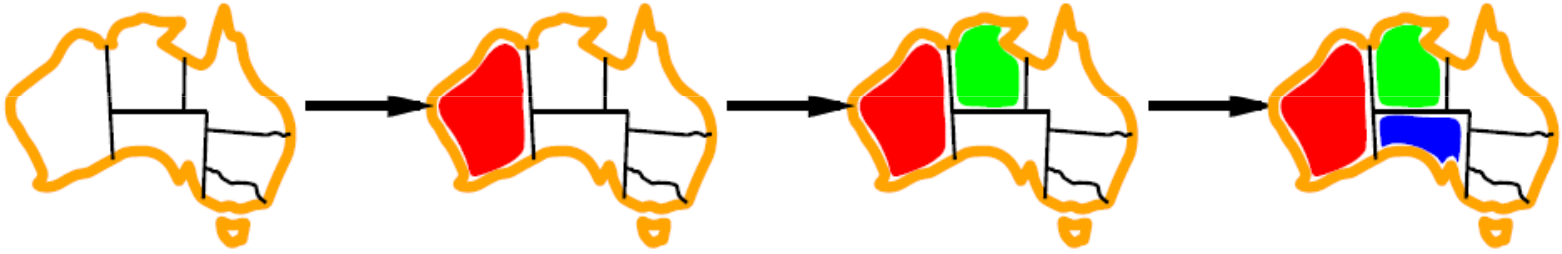
مسائل ارضای محدودیت

مثال جست و جوی عقبگرد برای CSP



مسائل ارضای محدودیت

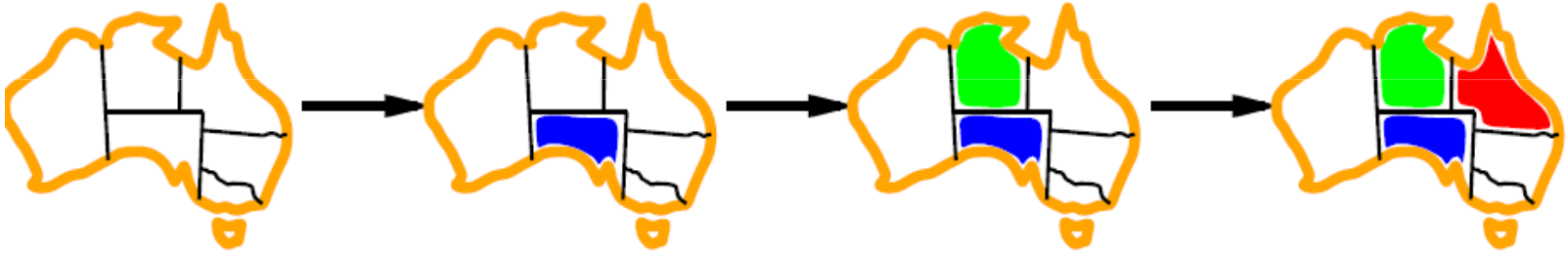
مقادیر باقیمانده کمینه(MRV)



- انتخاب متغیری با کمترین مقادیر معتبر
- متغیری انتخاب میشود که به احتمال زیاد ، بزودی با شکست مواجه شده و درخت جست و جو را هرس میکند

مسائل ارضای محدودیت

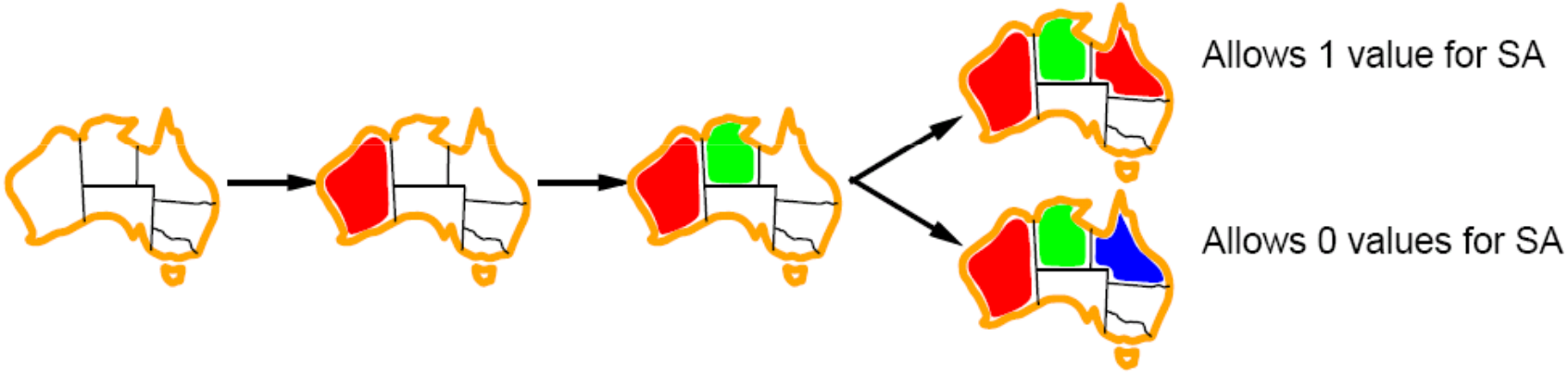
اکتشاف درجه ای



- سعی میکند فاکتور انشعاب را در انتخاب آینده کم کند
- متغیری انتخاب میکند که در بزرگترین محدودیتهای مربوط به متغیرهای بدون انتساب قرار دارد

مسائل ارضای محدودیت

اکتشاف مقداری باکمترین محدودیت

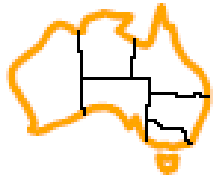


- این روش مقداری را ترجیح میدهد که در گراف محدودیت ، متغیرهای همسایه به ندرت آن را انتخاب میکنند
- سعی بر ایجاد بیشترین قابلیت انعطاف برای انتساب بعدی متغیرها

مسائل ارضای محدودیت

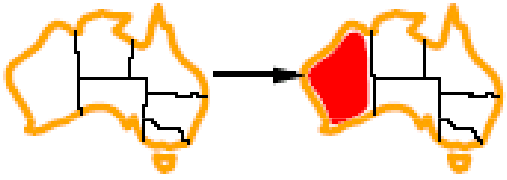
بررسی پیشرو

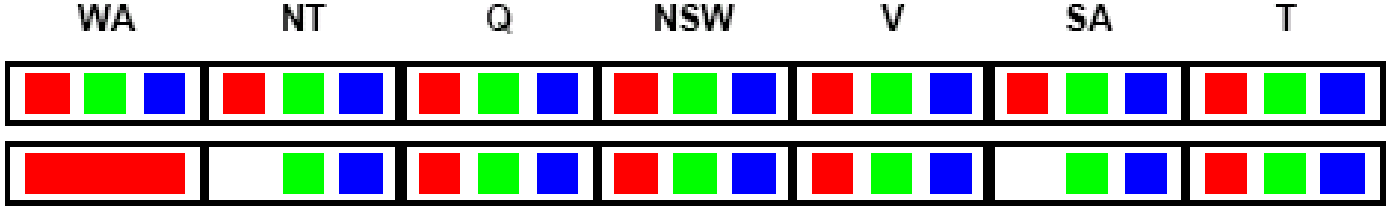
وقتی انتساب به X صورت میگیرد ، فرایند بررسی پیشرو ، متغیرهای بدون انتساب مثل Y را در نظر میگیرد که از طریق یک محدودیت به X متصل است و هر مقداری را که با مقدار انتخاب شده برای X برابر است ، از دامنه Y حذف میکند



مسائل ارضای محدودیت

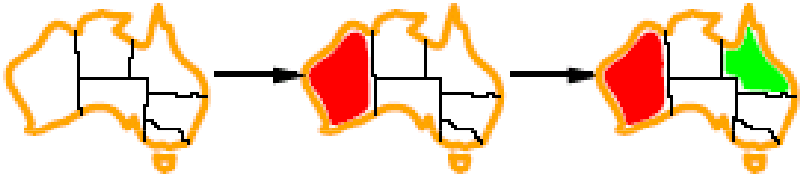
بررسی پیشرو

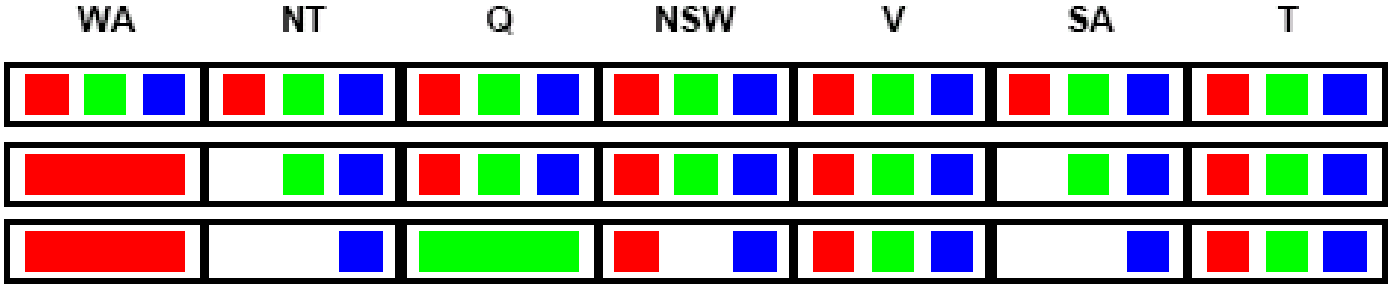




مسائل ارضای محدودیت

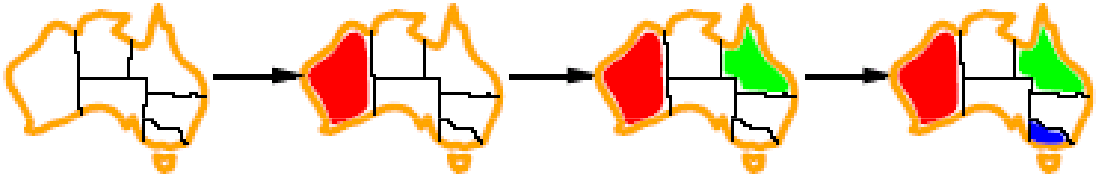
بررسی پیشرو

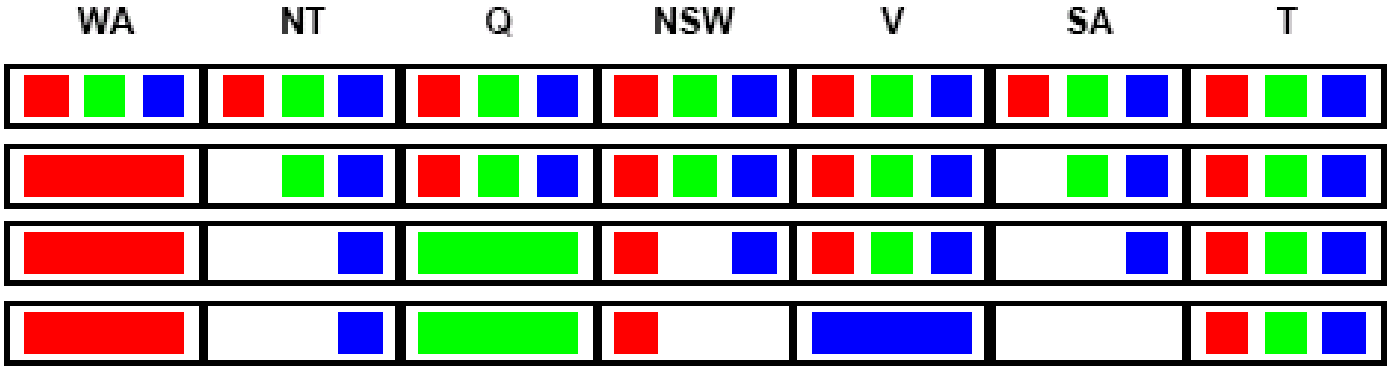




مسائل ارضای محدودیت

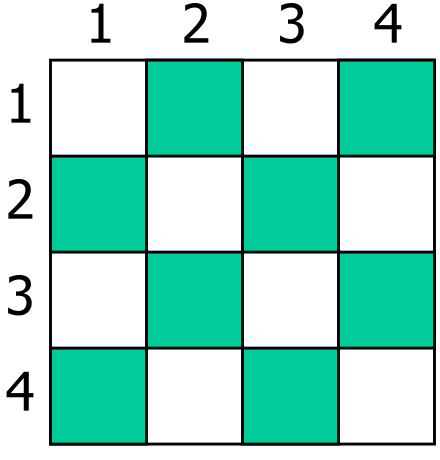
بررسی پیشرو

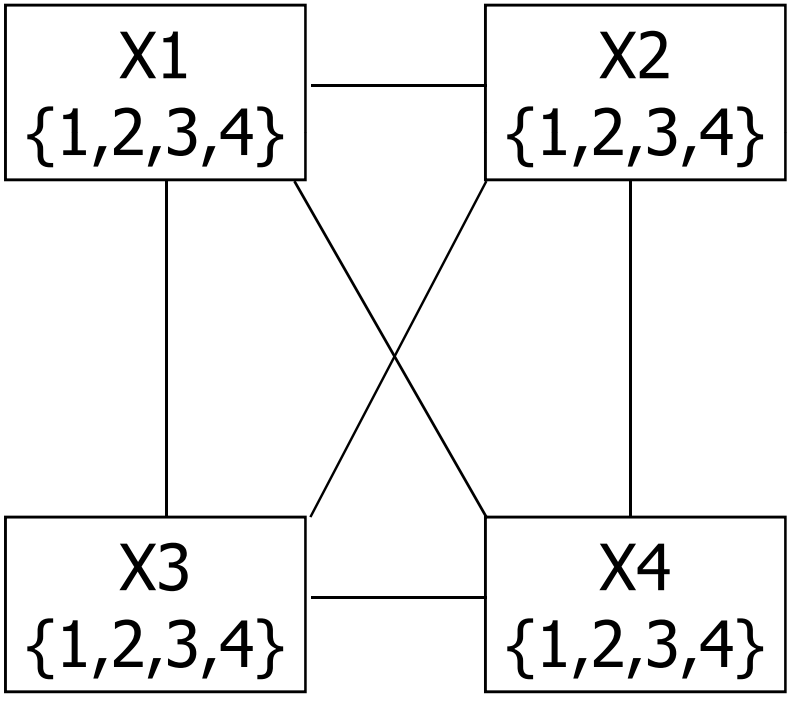




مسائل ارضای محدودیت

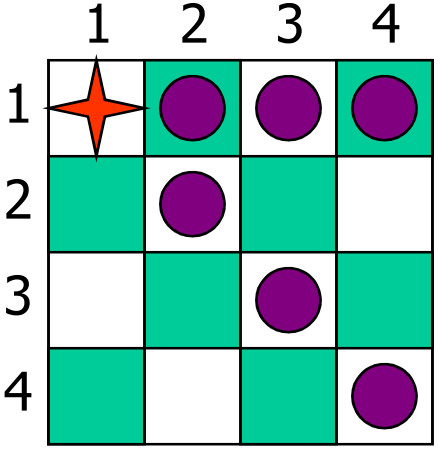
مثال: مسئله 4-وزیر

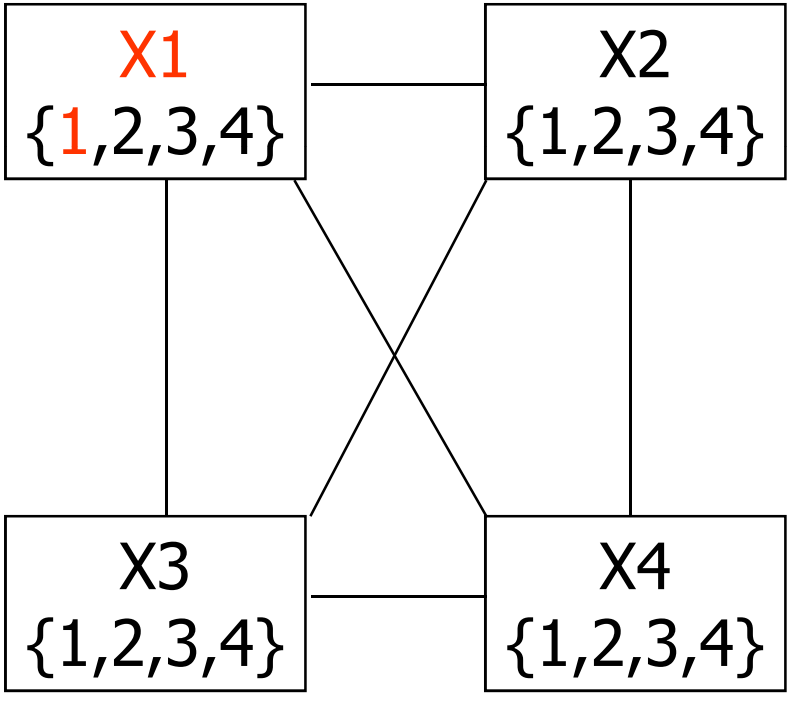




مسائل ارضای محدودیت

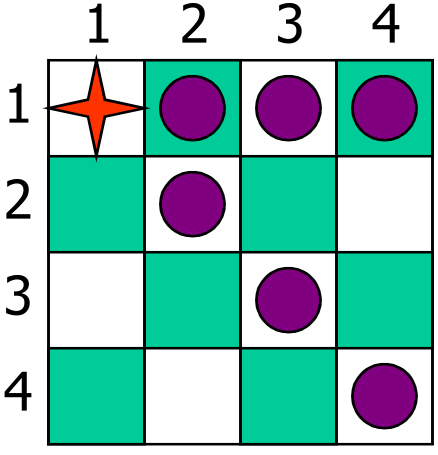
مثال: مسئله 4-وزیر

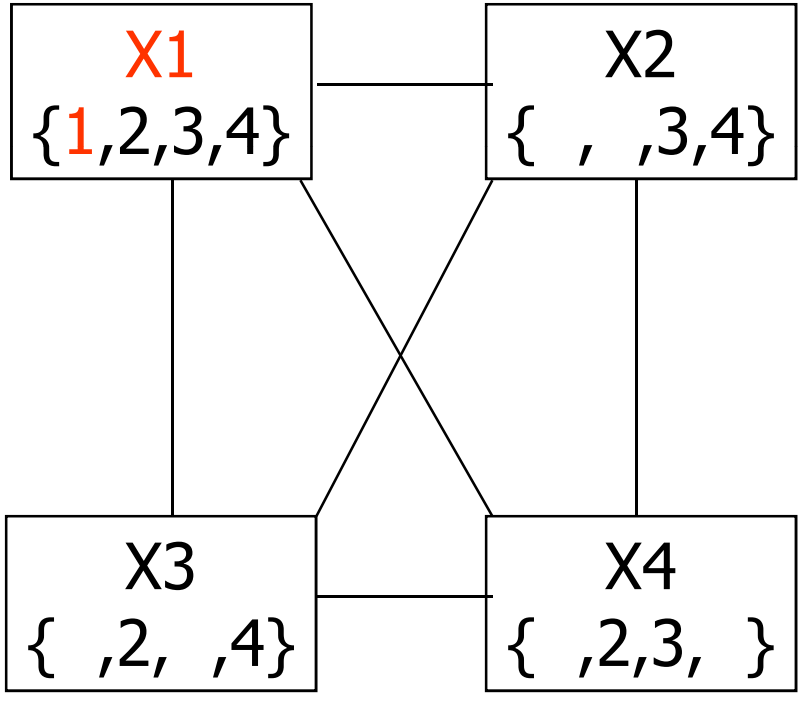




مسائل ارضای محدودیت

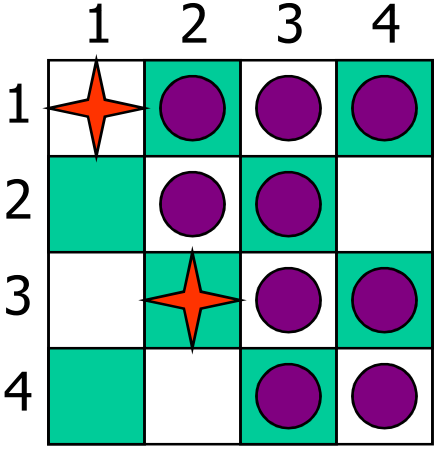
مثال: مسئله 4-وزیر

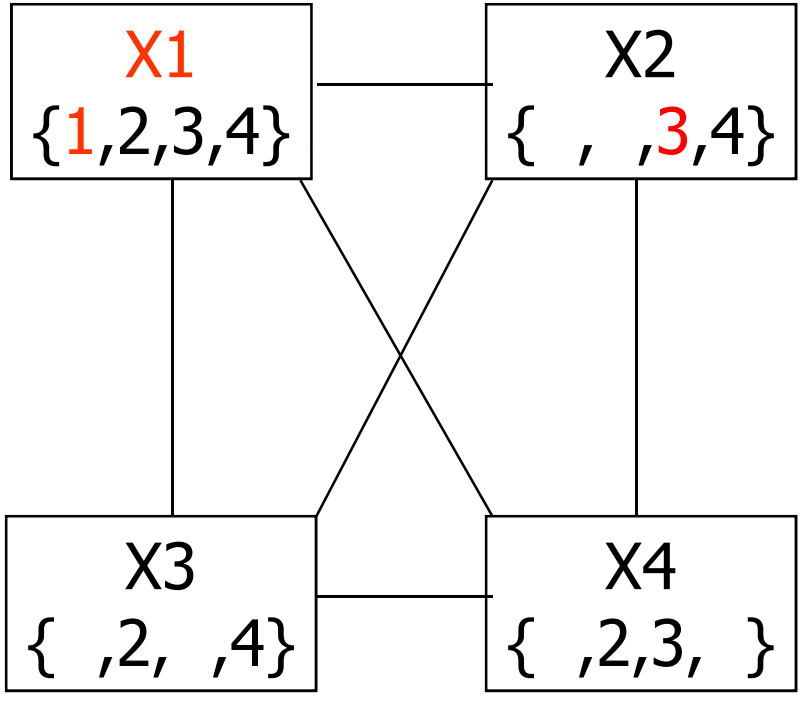




مسائل ارضای محدودیت

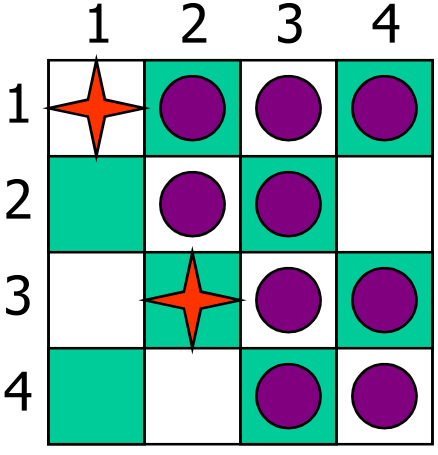
مثال: مسئله 4-وزیر

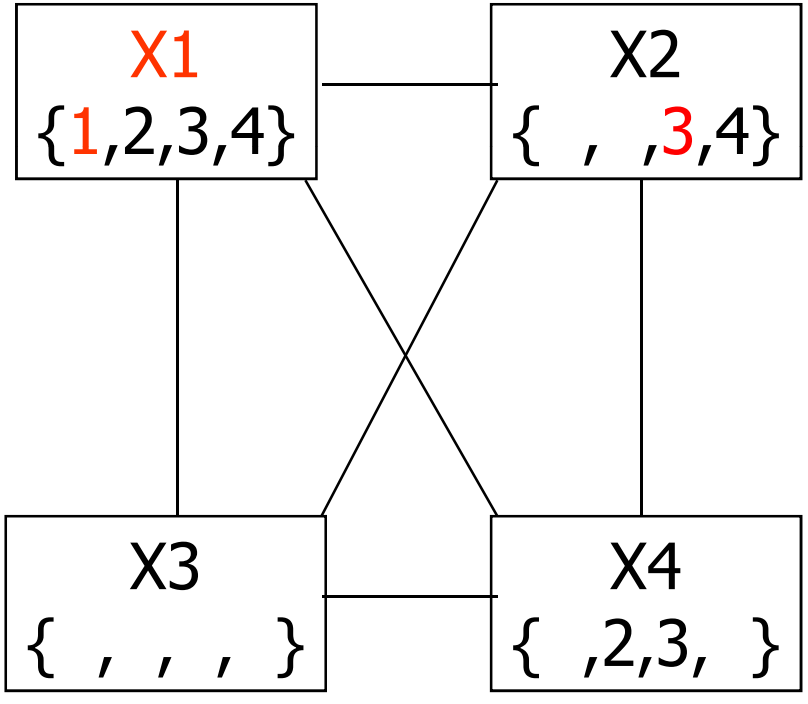




مسائل ارضای محدودیت

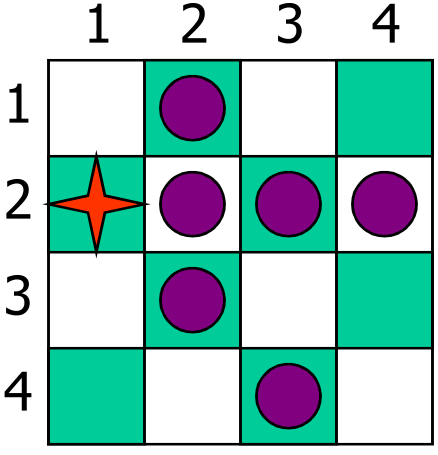
مثال: مسئله 4-وزیر

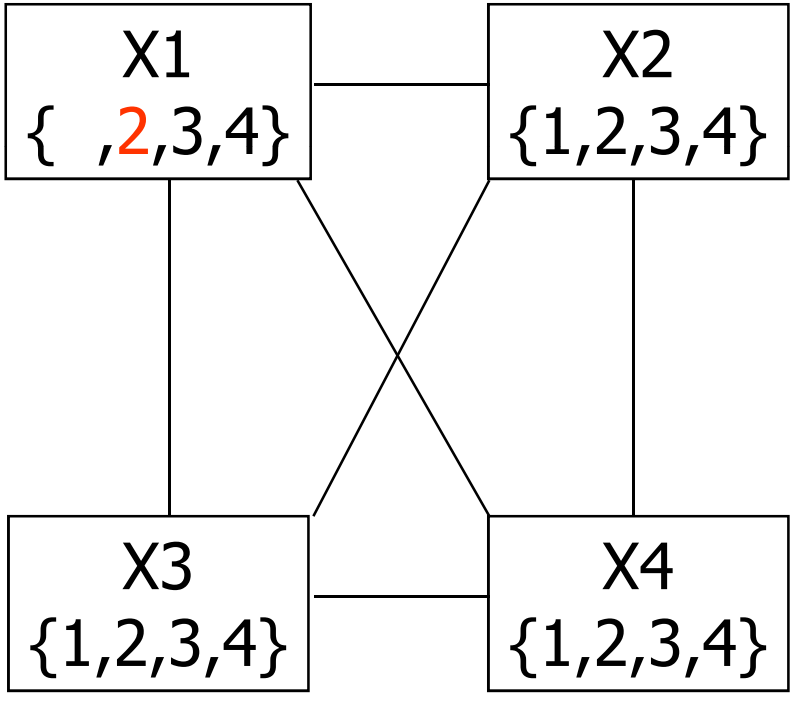




مسائل ارضای محدودیت

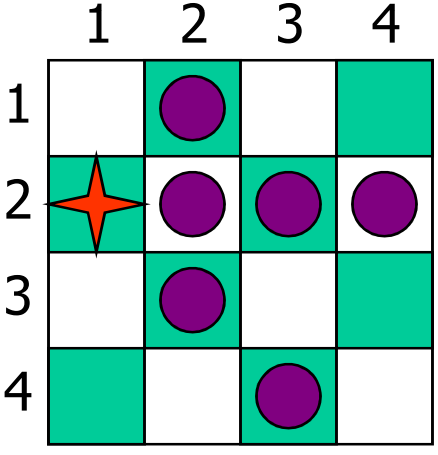
مثال: مسئله 4-وزیر

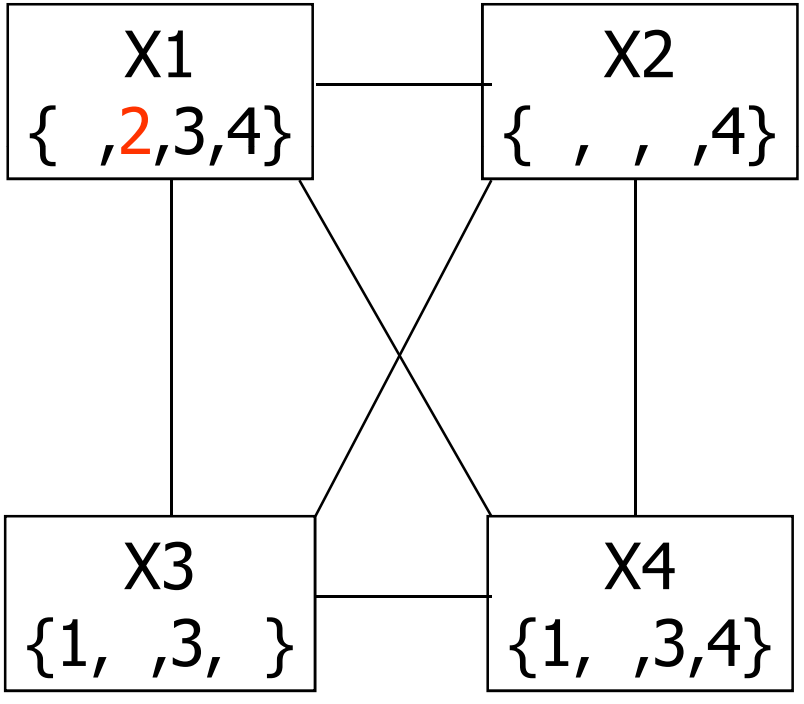




مسائل ارضای محدودیت

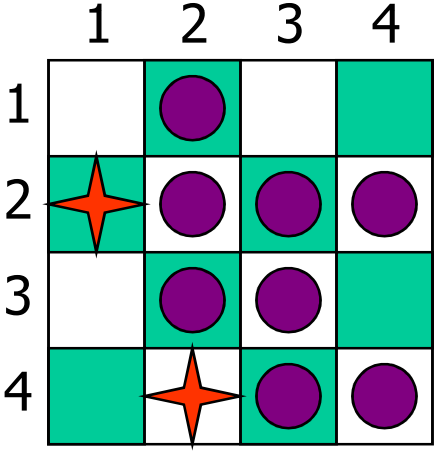
مثال: مسئله 4-وزیر

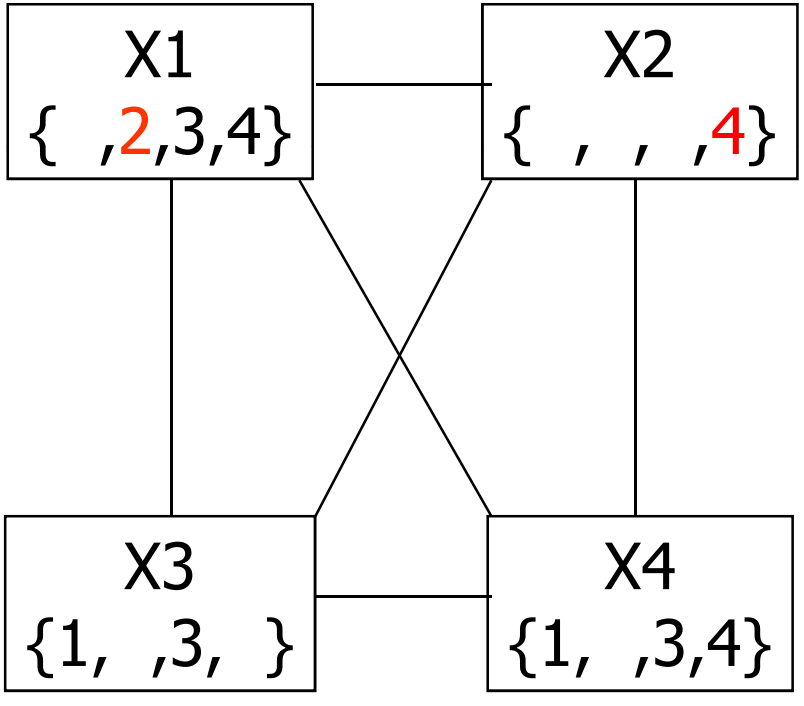




مسائل ارضای محدودیت

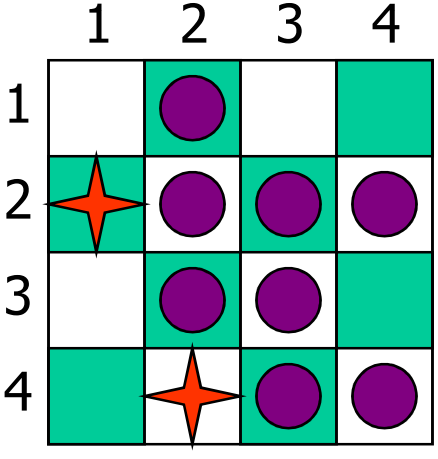
مثال: مسئله 4-وزیر

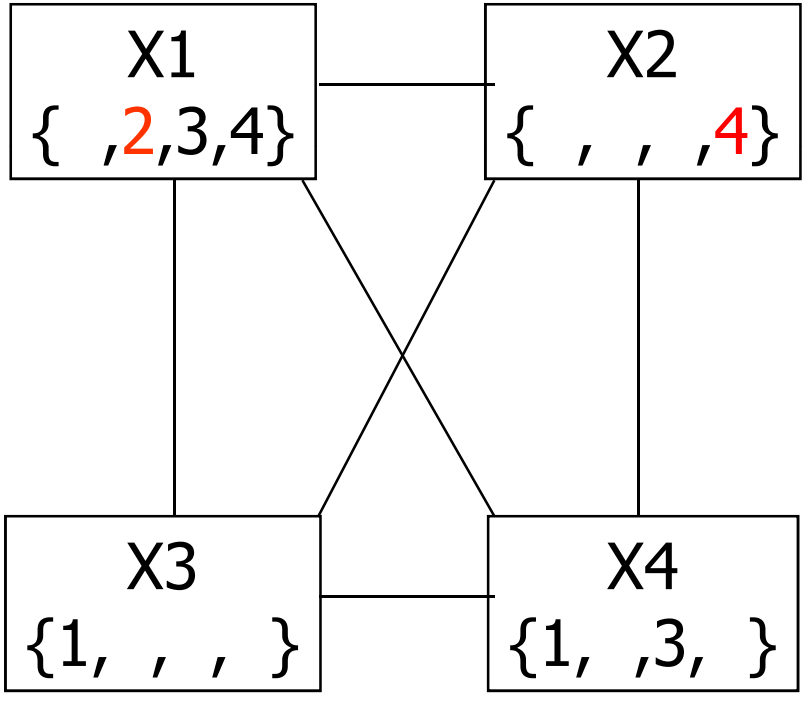




مسائل ارضای محدودیت

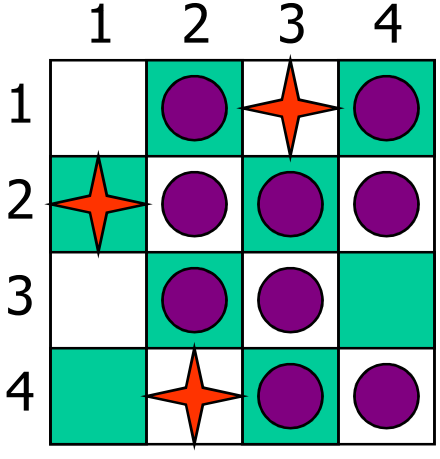
مثال: مسئله 4-وزیر

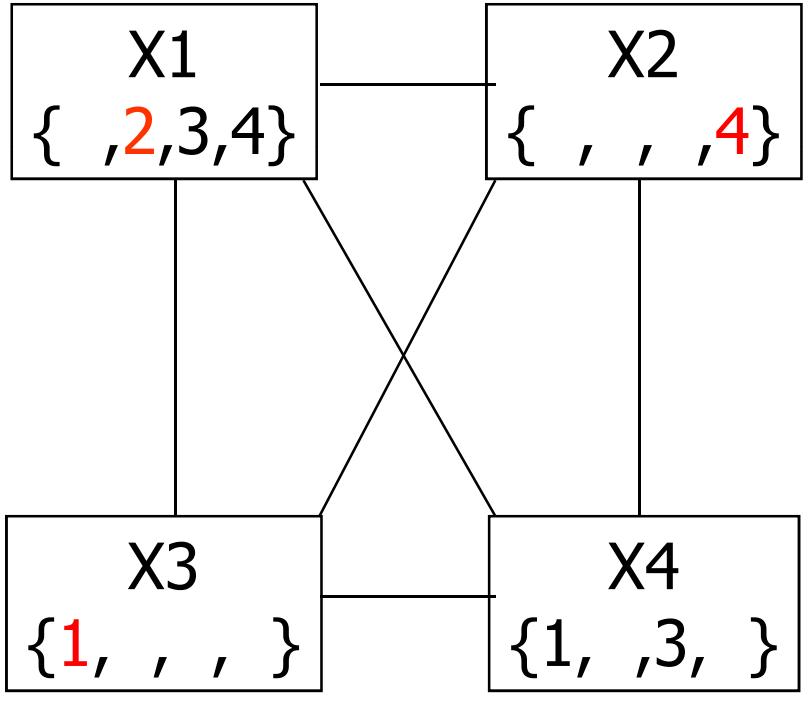




مسائل ارضای محدودیت

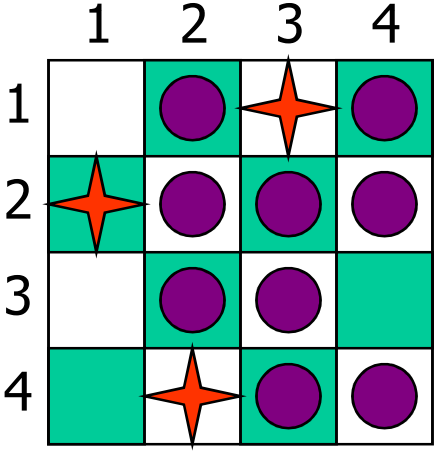
مثال: مسئله 4-وزیر

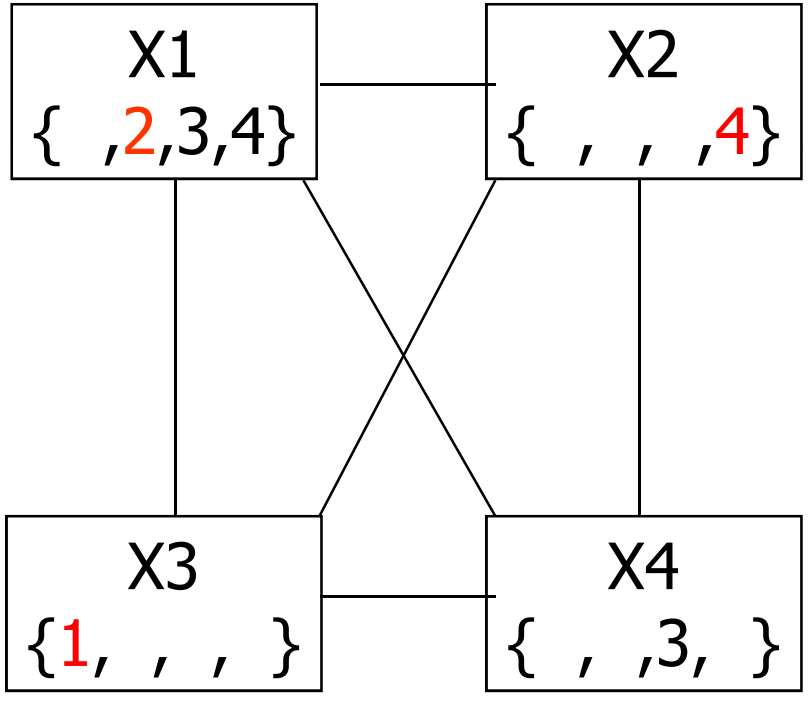




مسائل ارضای محدودیت

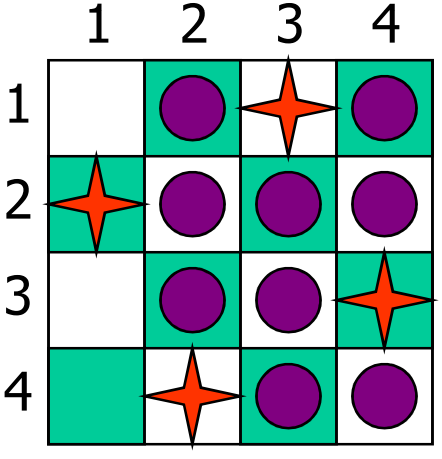
مثال: مسئله 4-وزیر

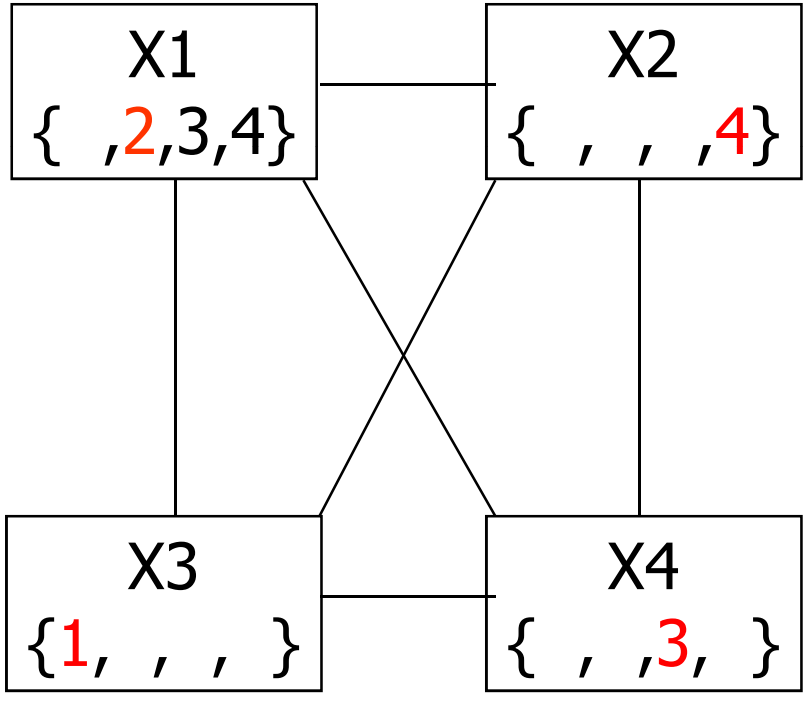




مسائل ارضای محدودیت

مثال: مسئله 4-وزیر



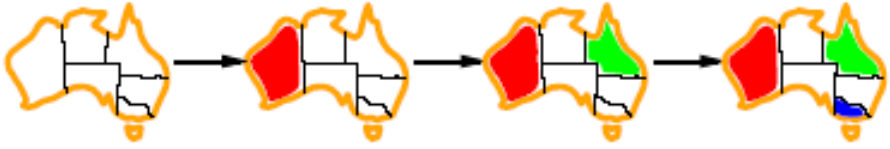


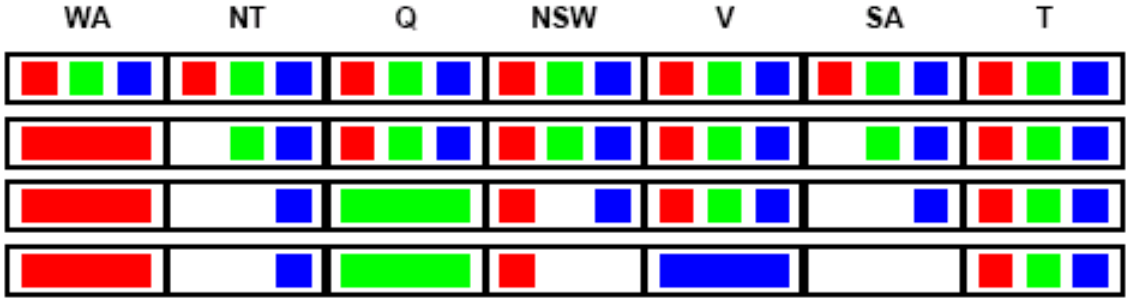
مسائل ارضای محدودیت

پخش محدودیت

پخش الزام محدودیتهای یک متغیر به متغیرهای دیگر

- مثال: پخش محدودیتهای WA و Q به NT و SA





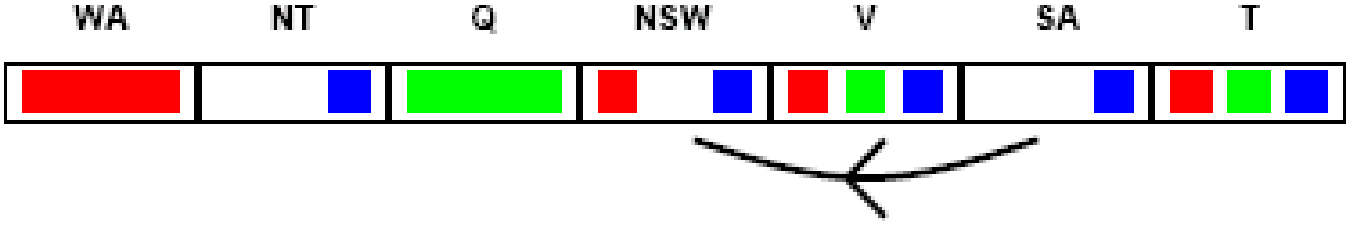
مسائل ارضای محدودیت

سازگاری یال

- روش سریعی برای پخش محدود و قویتر از بررسی پیشرو
- یال ؛ یال جهت دار در گراف محدودیت
- بررسی سازگاری یال
  - یک مرحله پیش پردازش ، قبل از شروع جستجو
  - یک مرحله پخشی پس از هر انتساب در حین جستجو

مسائل ارضای محدودیت

مثال: سازگاری یال

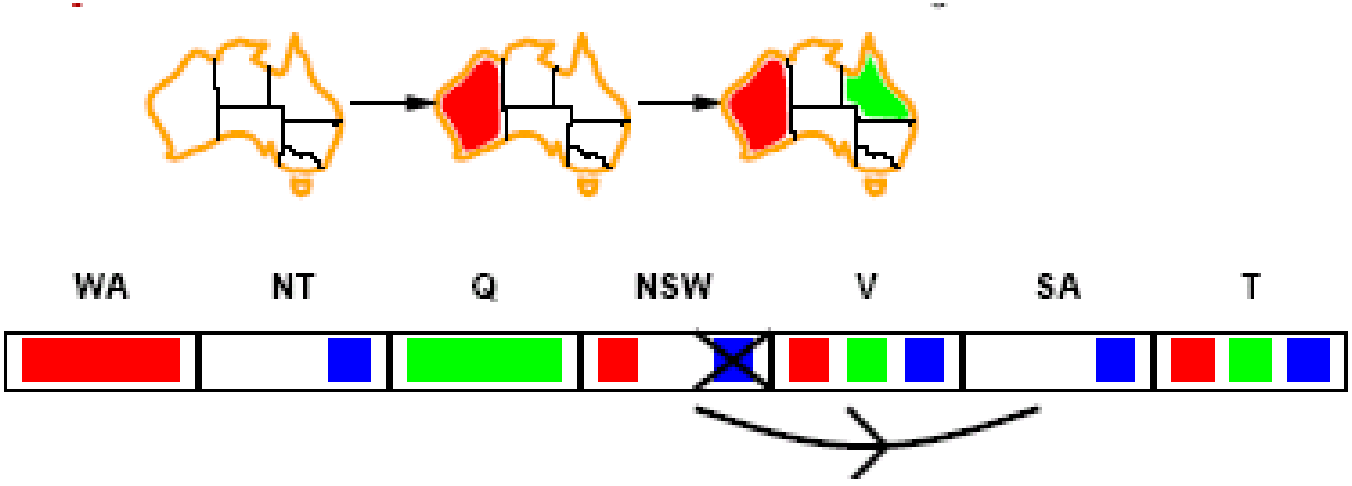


- NSW → SA سازگار است اگر

SA=blue and NSW=red

مسائل ارضای محدودیت

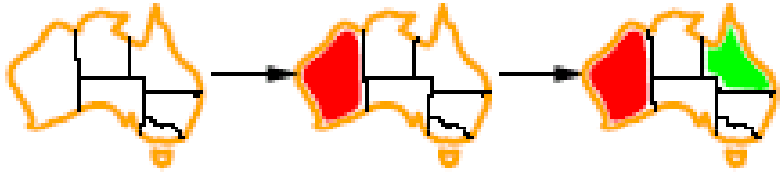
مثال: سازگاری یال

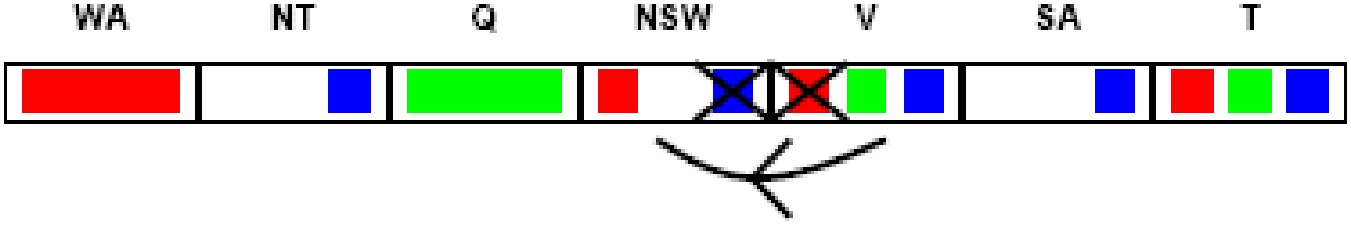


- NSW → SA سازگار است اگر
  SA=blue and NSW=red
  NSW=blue and SA=???
- یال میتواند سازگار شود با حذف blue از NSW

مسائل ارضای محدودیت

مثال: سازگاری یال

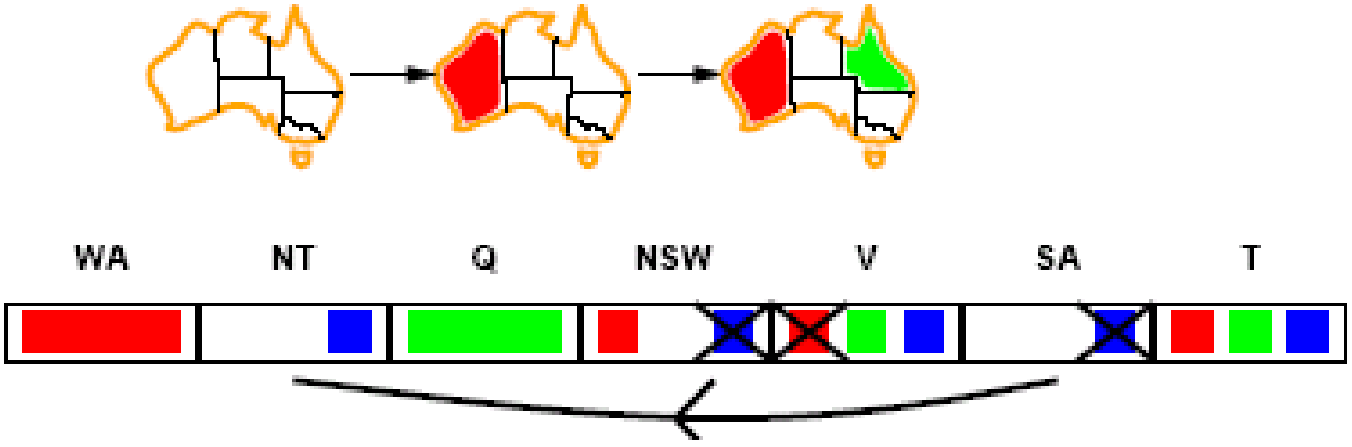




- یال میتواند سازگار شود با حذف blue از NSW
- حذف red از V

مسائل ارضای محدودیت

مثال: سازگاری یال



- یال میتواند سازگار شود با حذف blue از NSW
- حذف red از V
- تکرار تا هیچ ناسازگاری باقی نماند

مسائل ارضای محدودیت

سازگاری K

- سازگاری یال تمام ناسازگاریهای ممکن را مشخص نمیکند
- با روش سازگاری K ، شکلهای قویتری از پخش را میتوان تعریف کرد
- در صورتی CSP سازگاری K است ، که برای هر k-1 متغیر و برای هر انتساب سازگار با آن متغیرها ، یک مقدار سازگار ، همیشه بتواند به متغیر kام نسبت داده شود

مسائل ارضای محدودیت

سازگاری K

بطور مثال:

- سازگاری1: هر متغیر با خودش سازگار است(سازگاری گره)
- سازگاری2: مشابه سازگاری یال
- سازگاریk: بسط هر جفت از متغیرهای همجوار به سومین متغیر همسایه(سازگاری مسیر)

گراف در صورتی قویا سازگارK است که:

- سازگارk باشد
- همچنین سازگار k-1 و سازگارk-2 و... سازگار 1 باشد

در این صورت ، مسئله را بدون عقبگرد میتوان حل کرد

پیچیدگی زمانی آن O(nd) است

مسائل ارضای محدودیت

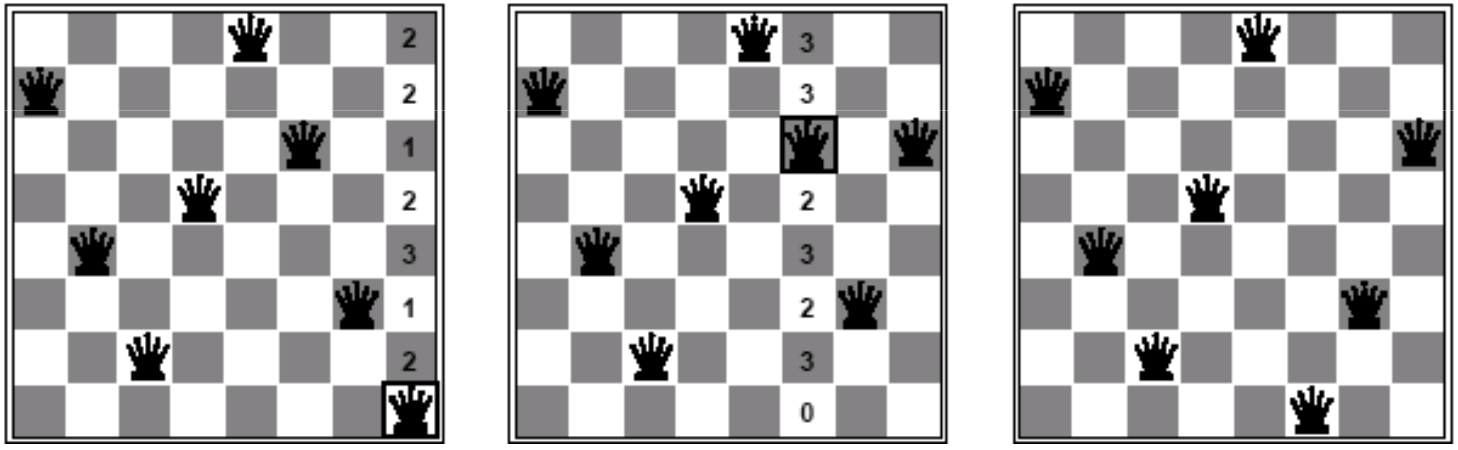
جست و جوی محلی در مسائل ارضای محدودیت

- بسیاری از CSPها را بطور کارآمد حل میکنند
  - حالت اولیه ، مقداری را به هر متغیر نسبت میدهد
  - تابع جانشین ، تغییر مقدار یک متغیر در هر زمان
- انتخاب مقدار جدید برای یک متغیر
  - انتخاب مقداری که کمترین برخورد را با متغیرهای دیگر ایجاد کند(اکتشاف برخورد کم)
  - زمان اجرای برخورد کم مستقل از اندازه مسئله است
  - برخورد کم ، برای مسئله های سخت نیز کار میکند
- جست و جوی محلی میتواند در صورت تغییر مسئله ، تنظیمات Online را انجام دهد

مسائل ارضای محدودیت

مثال

راه حل دو مرحله ای برای مسئله 8وزیر با استفاده از کمترین برخورد



- در هر مرحله ، یک وزیر برای انتساب مجدد در ستون خودش انتخاب میگردد
- تعداد برخوردها در هر مربع نشان داده شده است
- الگوریتم وزیر را به مربعي با کمترین برخورد انتقال میدهد ، بطوریکه گره ها را بطور تصادفی میشکند

هوش مصنوعي

فصل ششم

جستجوی خصمانه

هوش مصنوعي Artificial Intelligence

فهرست

- بازیها چیستند و چرا مطالعه میشوند؟
- انواع بازیها
- الگوریتم **minimax**
- بازیهای چند نفره
- هرس آلفا-بتا
- بازیهای قطعی با اطلاعات ناقص
- بازیهایي که حاوی عنصر شانس هستند

جستجوی خصمانه

بازی ها چیستند و چرا مطالعه میشوند؟

- بازیها حالتی از محیطهای چند عاملي هستند
  - هر عامل نیاز به در نظر گرفتن سایر عاملها و چگونگی تأثیر آنها دارد
  - تمایز بین محیطهای چند عامل رقابتي و همکار
  - محیطهای رقابتی ، که در آنها اهداف عاملها با یکدیگر برخورد دارند ، منجر به مسئله های خصمانه میشود که به عنوان بازی شناخته میشوند
- چرا مطالعه میشوند؟
  - قابلیتهای هوشمندی انسانها را به کار میگیرند
  - ماهیت انتزاعی بازی ها
  - حالت بازی را به راحتی میتوان نمایش داد و عاملها معمولا به مجموعه کوچکی از فعالیتها محدود هستند که نتایج آنها با قوانین دقیقي تعریف شده اند

جستجوی خصمانه

انواع بازی ها

	قطعی	تصادفی
اطلاعات کامل	شطرنج ریورسی	تخته نرد
اطلاعات ناقص		پوکر

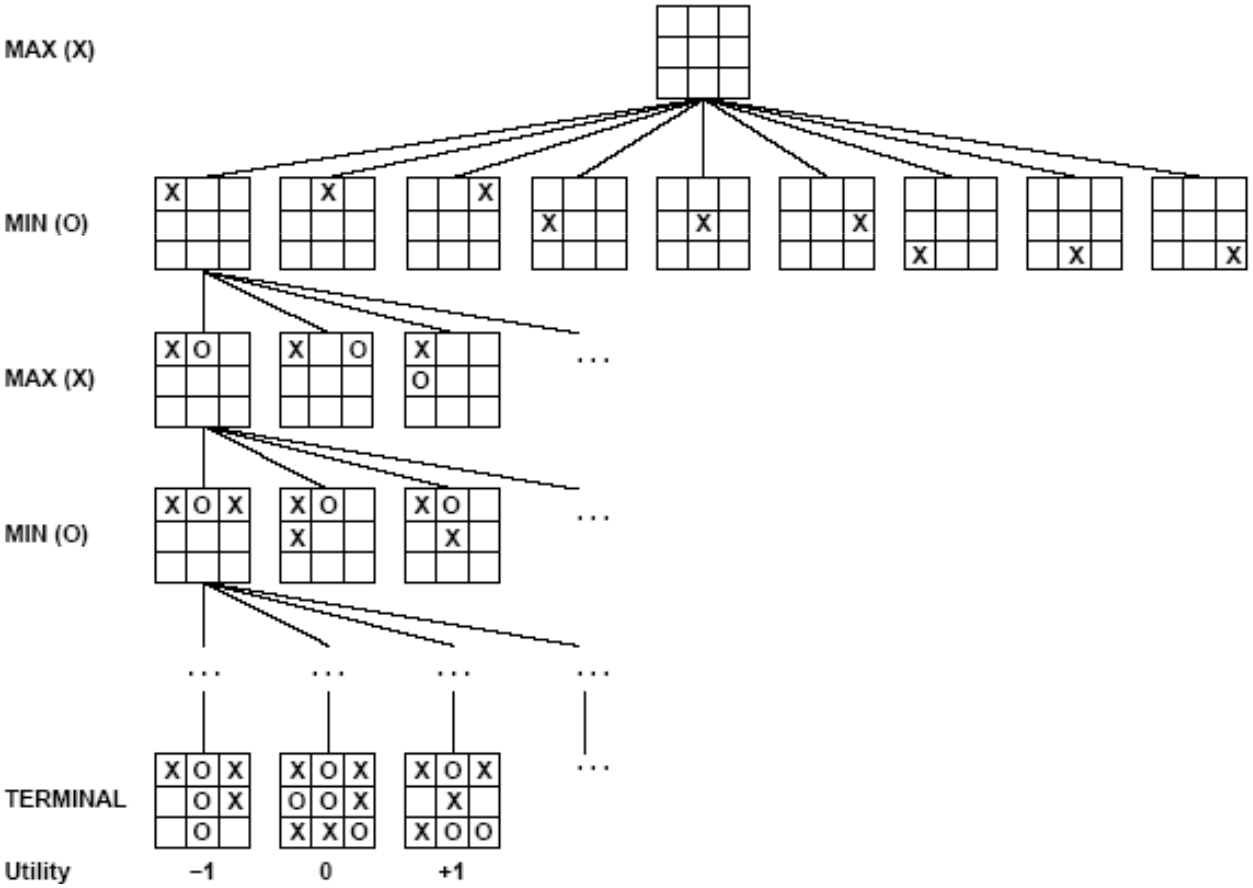
جستجوی خصمانه

یک نمونه بازی

- بازی دو نفره: Min و Max
  - اول Max حرکت میکند و سپس به نوبت بازی میکنند تا بازی تمام شود
  - در پایان بازی ، برنده جایزه و بازنده جریمه میشود
- بازی به عنوان یک جستجو:
  - حالت اولیه: موقعیت صفحه و شناسه های قابل حرکت
  - تابع جانشین:لیستی از (حالت,حرکت) که معرف یک حرکت معتبر است
  - آزمون هدف:پایان بازی چه موقع است ؟(حالتهای پایانه)
  - تابع سودمندی: برای هر حالت پایانه یک مقدار عددی را ارائه میکند. مثلا برنده(+1) و بازنده(-1)
- حالت اولیه و حرکات معتبر برای هر بازیکن ، **درخت بازی** را برای آن بازی ایجاد میکند

جستجوی خصمانه

یک نمونه بازی



الگوریتم ؛

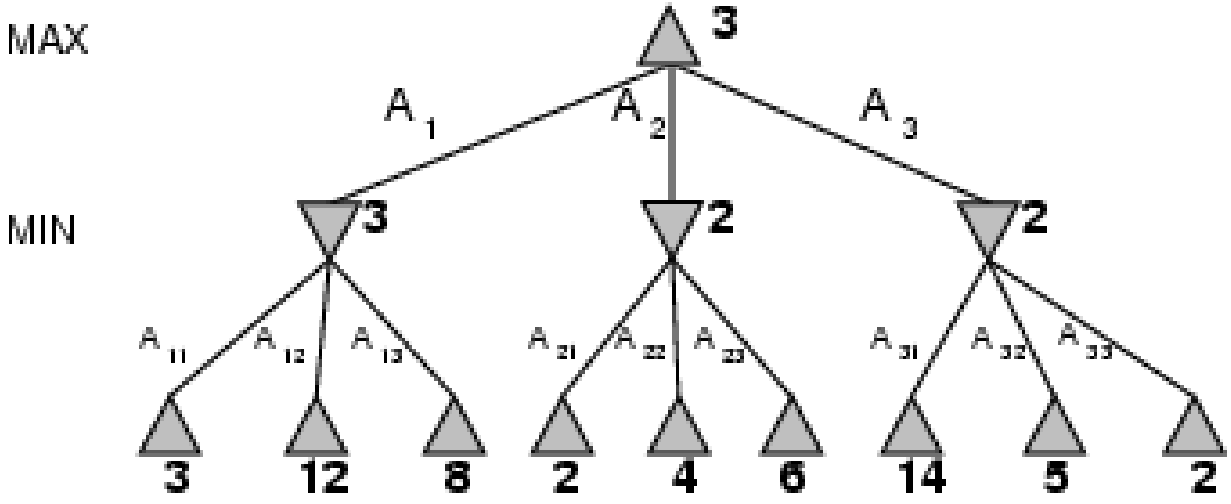
- **بازیکن:** انتخاب **بهترین حالت**
- **حریف:** انتخاب بهترین موقعیت **برای خودش یا بدترین وضعیت برای بازیکن**

بازیکن: ماکزیمم حالت

حریف: مینیمم حالت

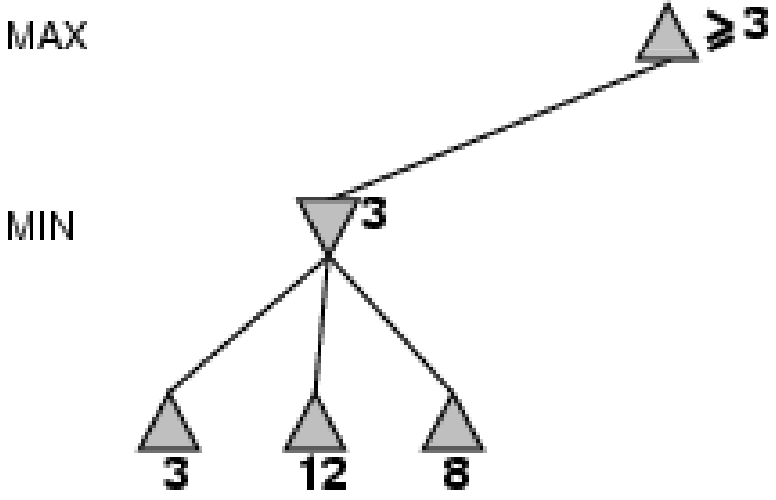
جستجوی خصمانه

الگوریتم minimax



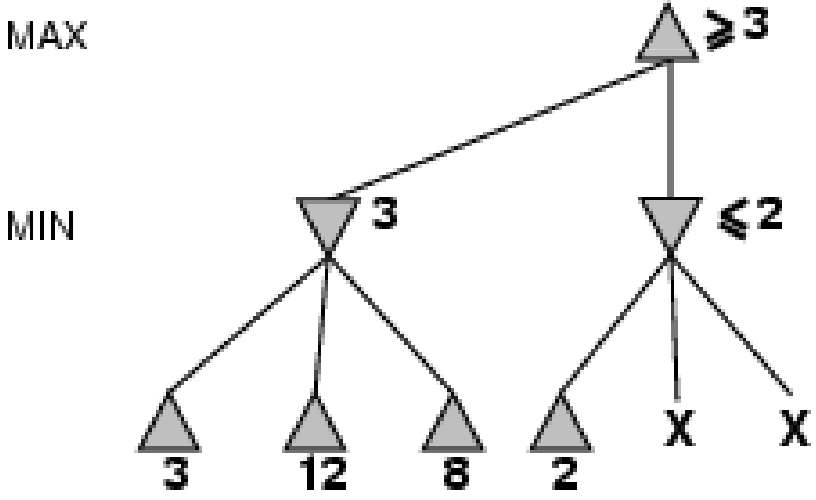
جستجوی خصمانه

یک نمونه بازی



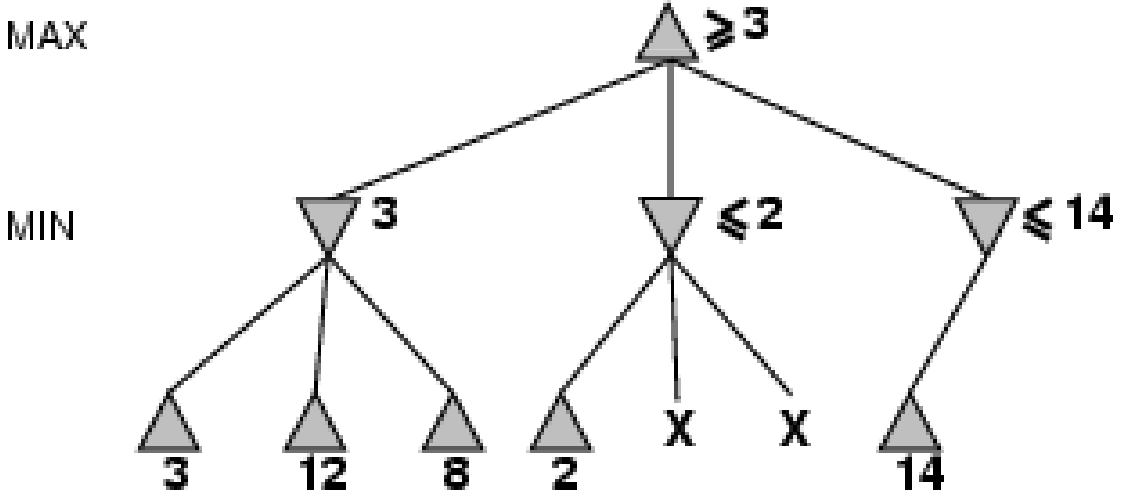
جستجوی خصمانه

یک نمونه بازی



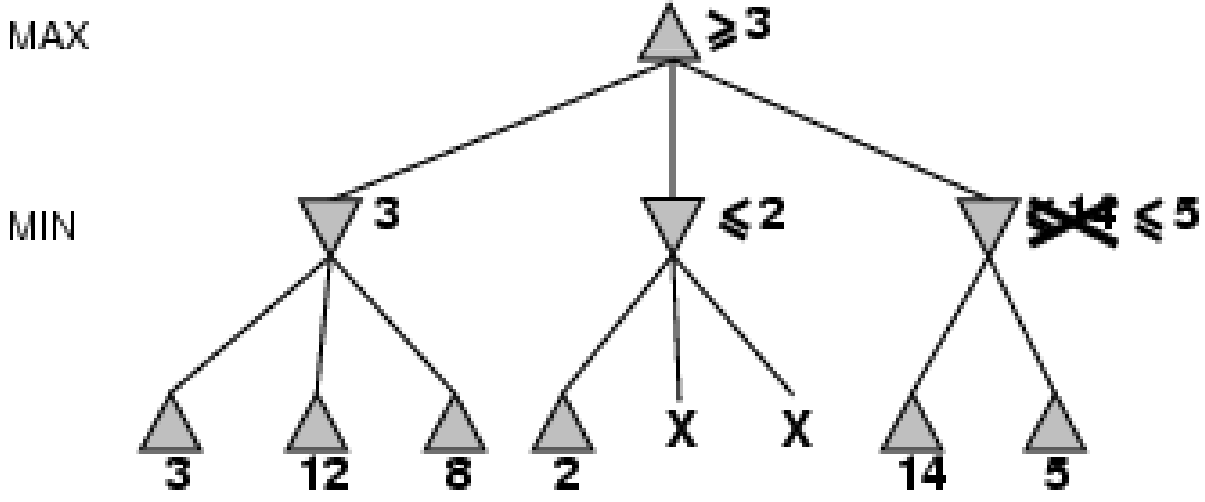
جستجوی خصمانه

یک نمونه بازی



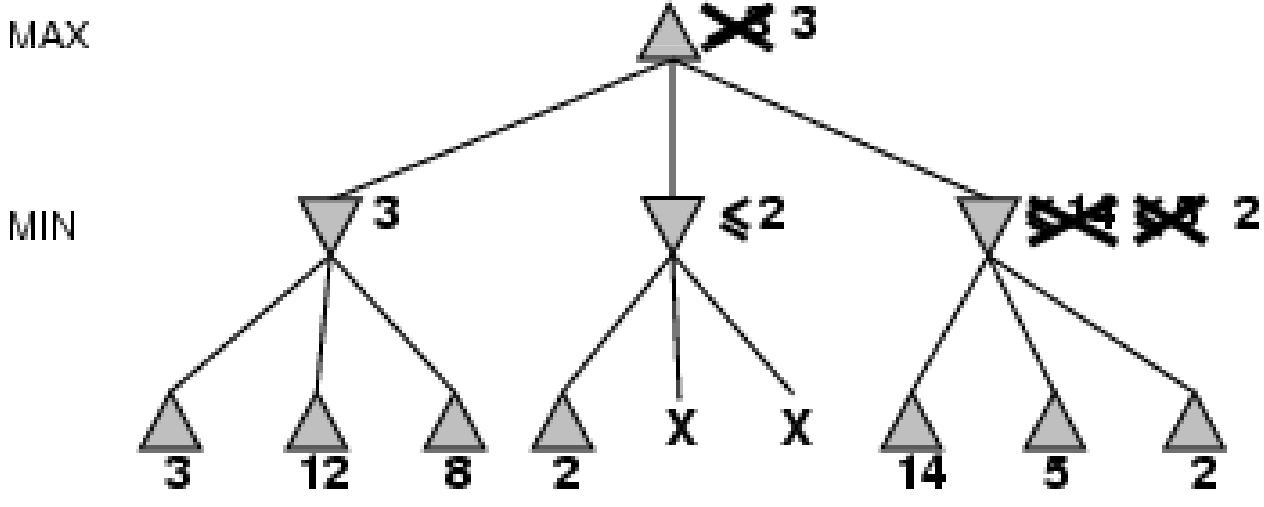
جستجوی خصمانه

یک نمونه بازی



جستجوی خصمانه

یک نمونه بازی



جستجوی خصمانه

الگوریتم minimax

کامل بودن: بله (اگر درخت محدود باشد)

بهینگي: بله

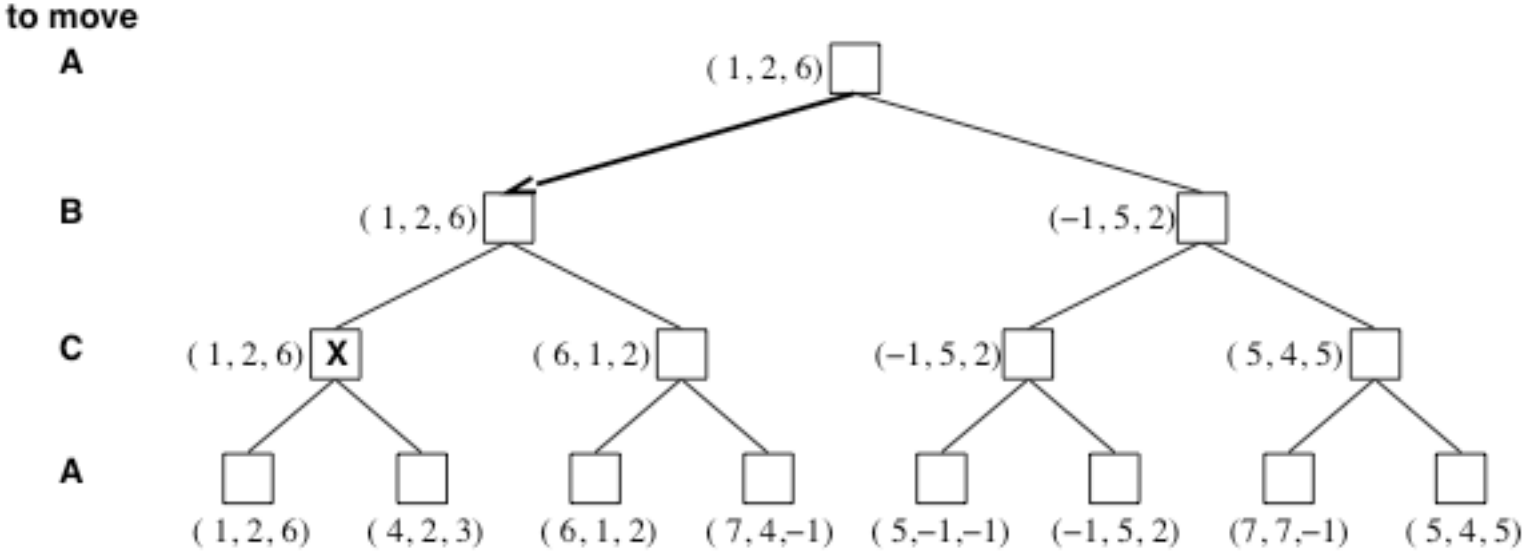
پیچیدگي زمانی: $O(b^m)$

پیچیدگی فضا: $O(bm)$

جستجوی خصمانه

بازیهای چند نفره

- تخصیص یک **بردار** به هر گره ، به جای یک مقدار
- بازیهای چند نفره معولاً شامل **اتحاد** رسمی یا غیر رسمي بين بازیکنان است
  - اتحاد با پیشروی بازی ایجاد و از بین میرود
  - بازیکنان بطور خودکار همکاری میکنند ، تا به هدف مطلوب انحصاری برسند



جستجوی خصمانه

هرس آلفا-بتا

در الگوریتم MaxMin:

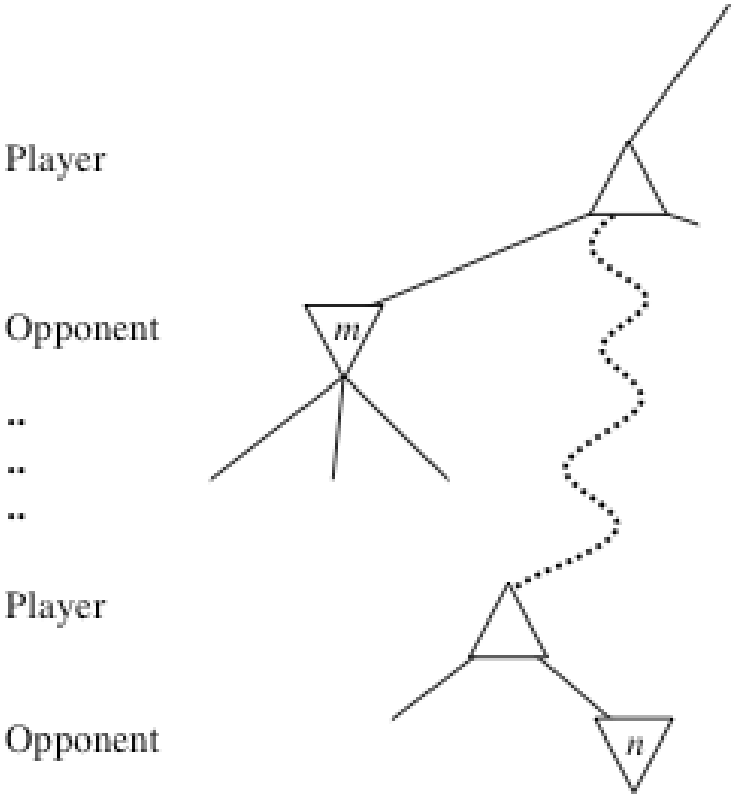
- تعداد حالتهای بازی که باید بررسی شوند ، بر حسب تعداد حرکتها ، توانی است
  - راه حل: محاسبه تصمیم الگوریتم ، بدون دیدن همه گره ها امکانپذیر است

هرس آلفا-بتا:

- انشعابهايي که در تصميم نهايي تأثير ندارند را حذف ميکند
  - آلفا: مقدار بهترین انتخاب در هر نقطه انتخاب در مسیر Max تاکنون
  - بتا: مقدار بهترین انتخاب در هر نقطه انتخاب در مسیر Min تاکنون
  - تعداد گره هايي که بايد بررسي شوند به تقليل ميابد $O(b^{d/2})$
  - فاکتور انشعاب مؤثر به جای b برابر با جذر b خواهد بود
  - پيش بيني آن نسبت به minimax دو برابر است

جستجوی خصمانه

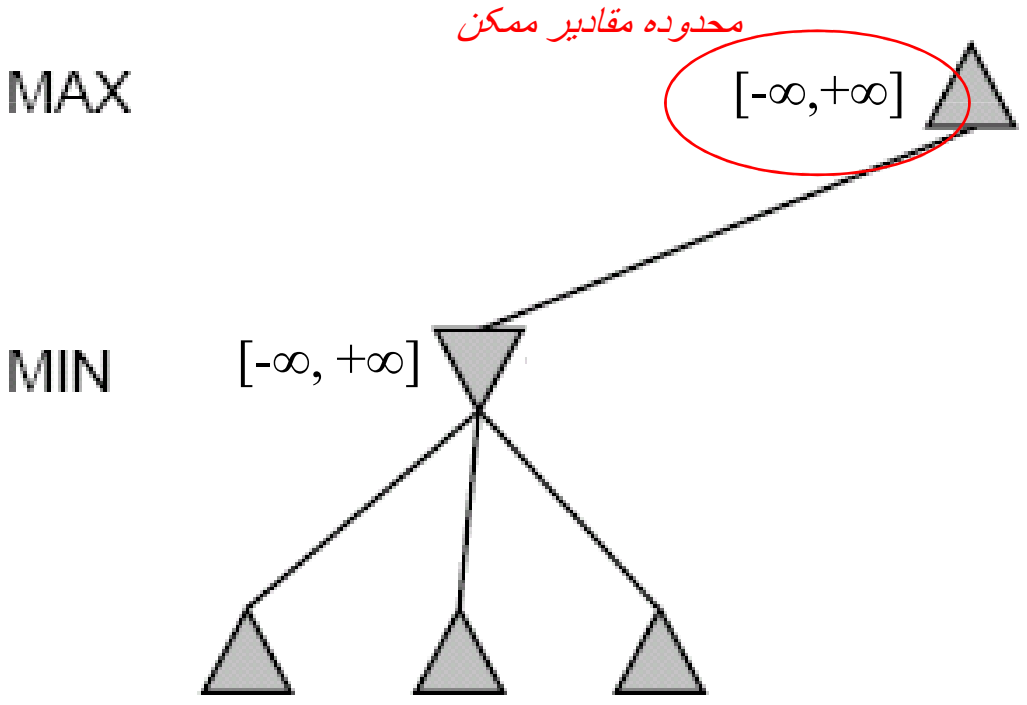
هرس آلفا-بتا



- گره n که هر جای درخت میتواند باشد ، بررسي میشود
- اگر بازیکن انتخاب بهتری داشته باشد
  - در گره والد n
  - یا هر انتخاب بهتری تا کنون
- n هیچوقت در بازی واقعی قابل دسترس نخواهد بود
- در نتیجه n هرس میشود

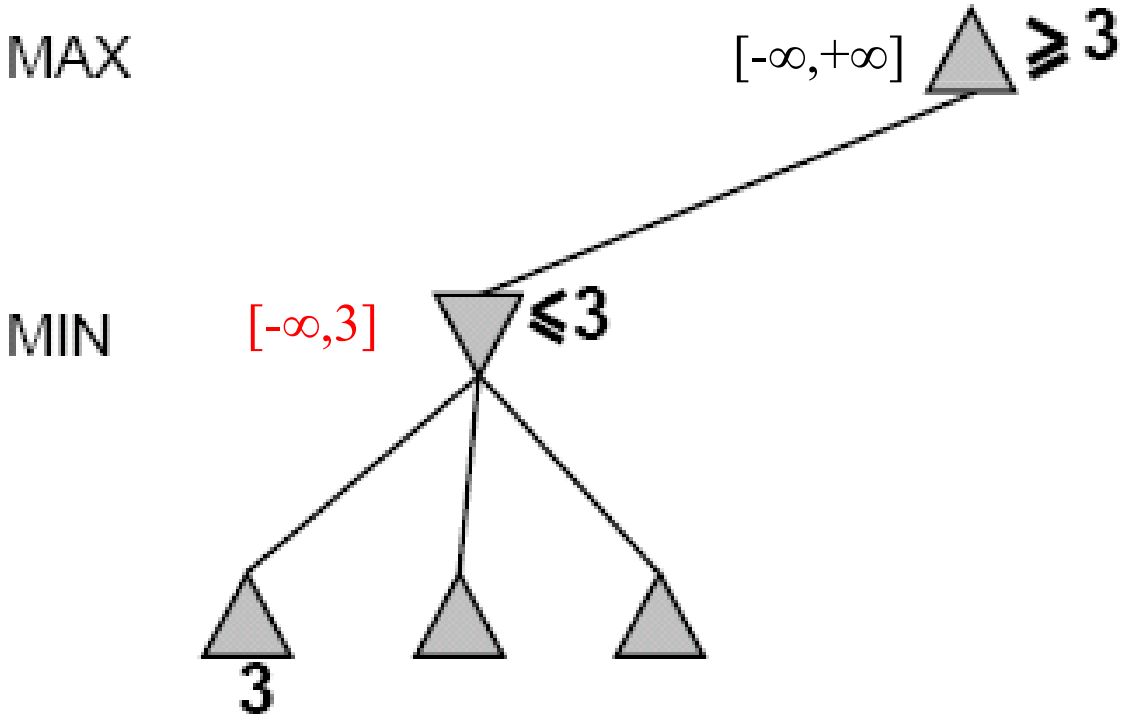
جستجوی خصمانه

مثال: هرس آلفا-بتا



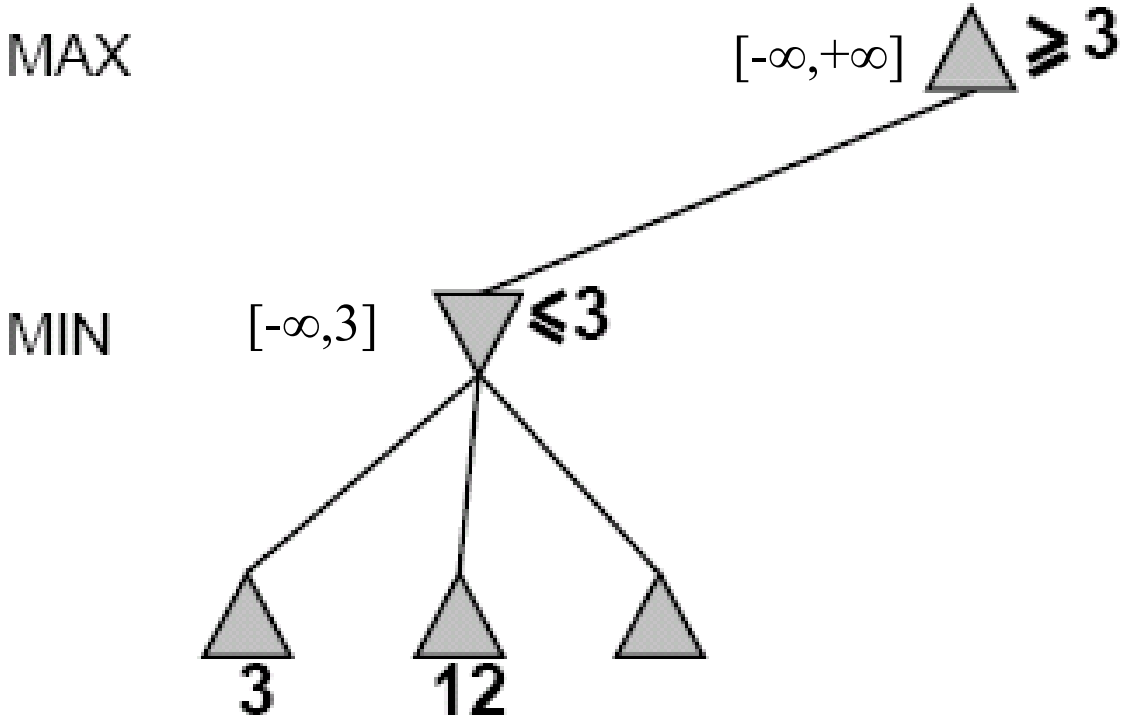
جستجوی خصمانه

مثال: هرس آلفا-بتا

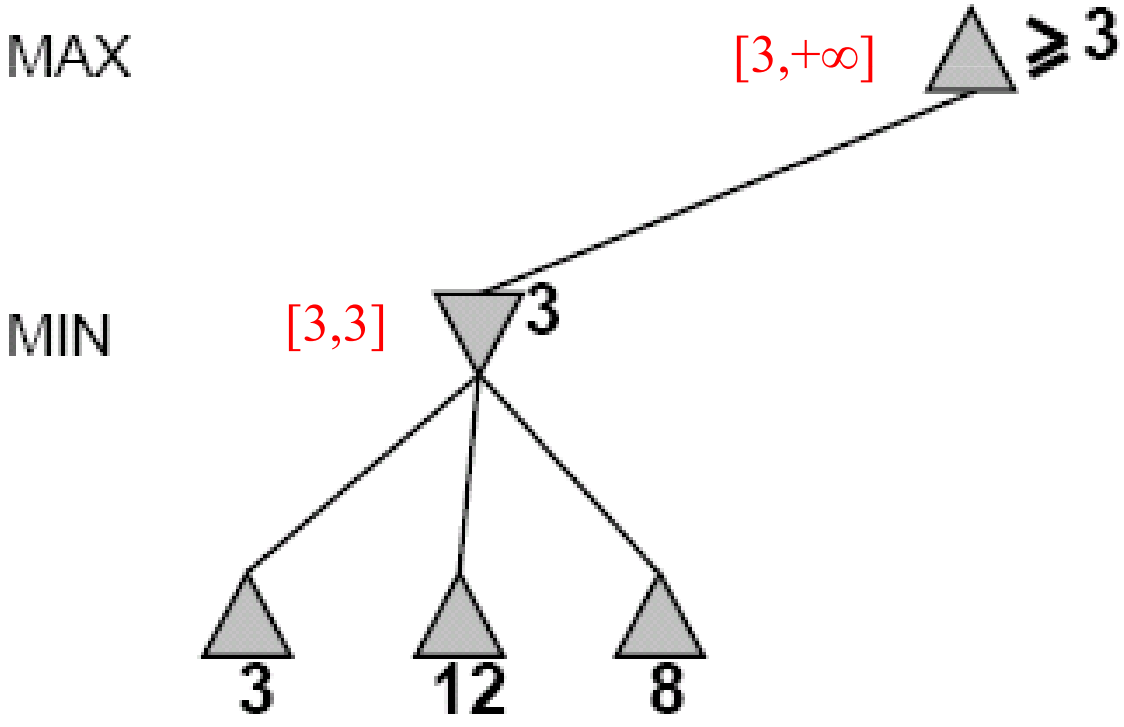


جستجوی خصمانه

مثال: هرس آلفا-بتا

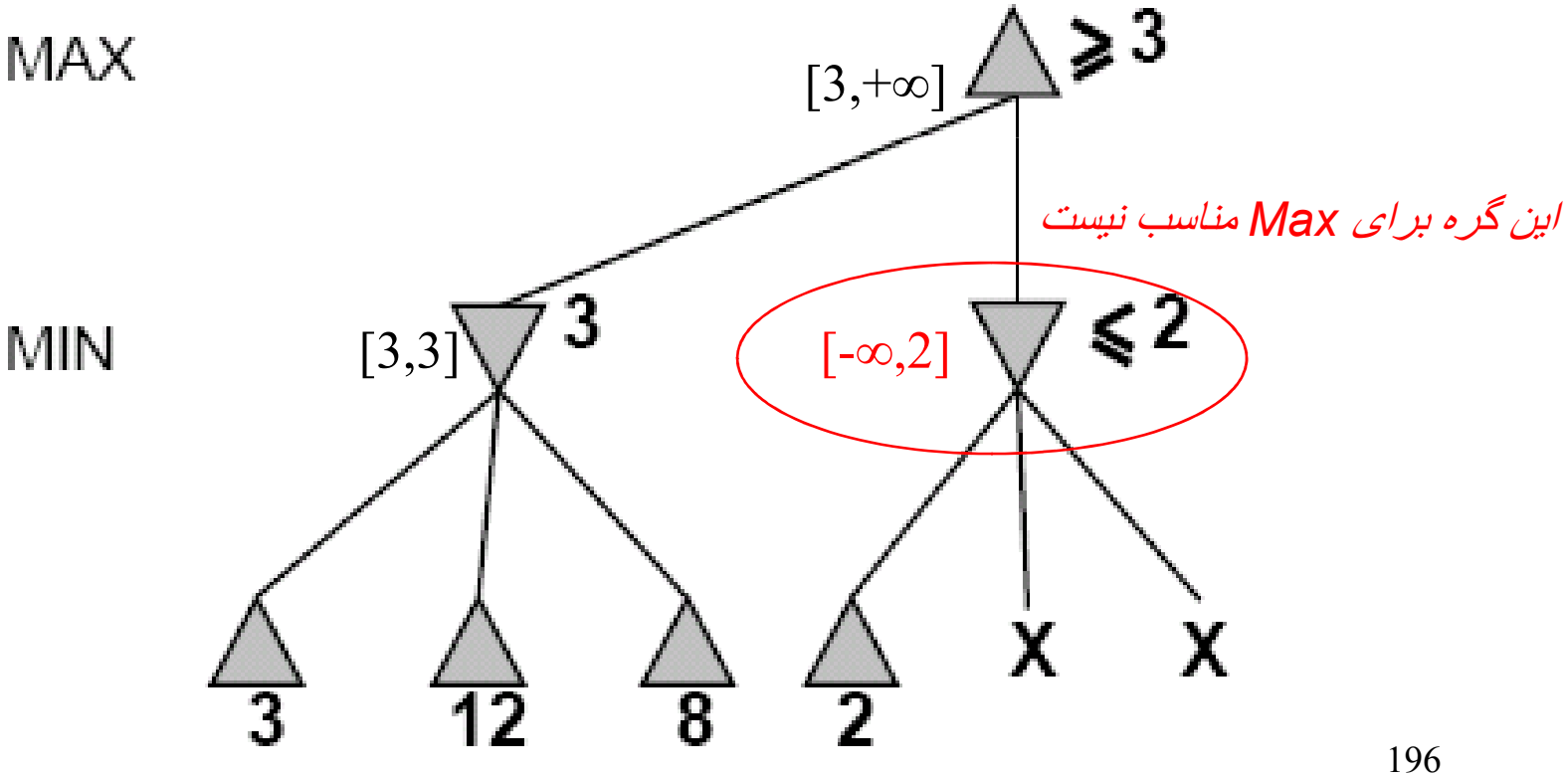


جستجوی خصمانه



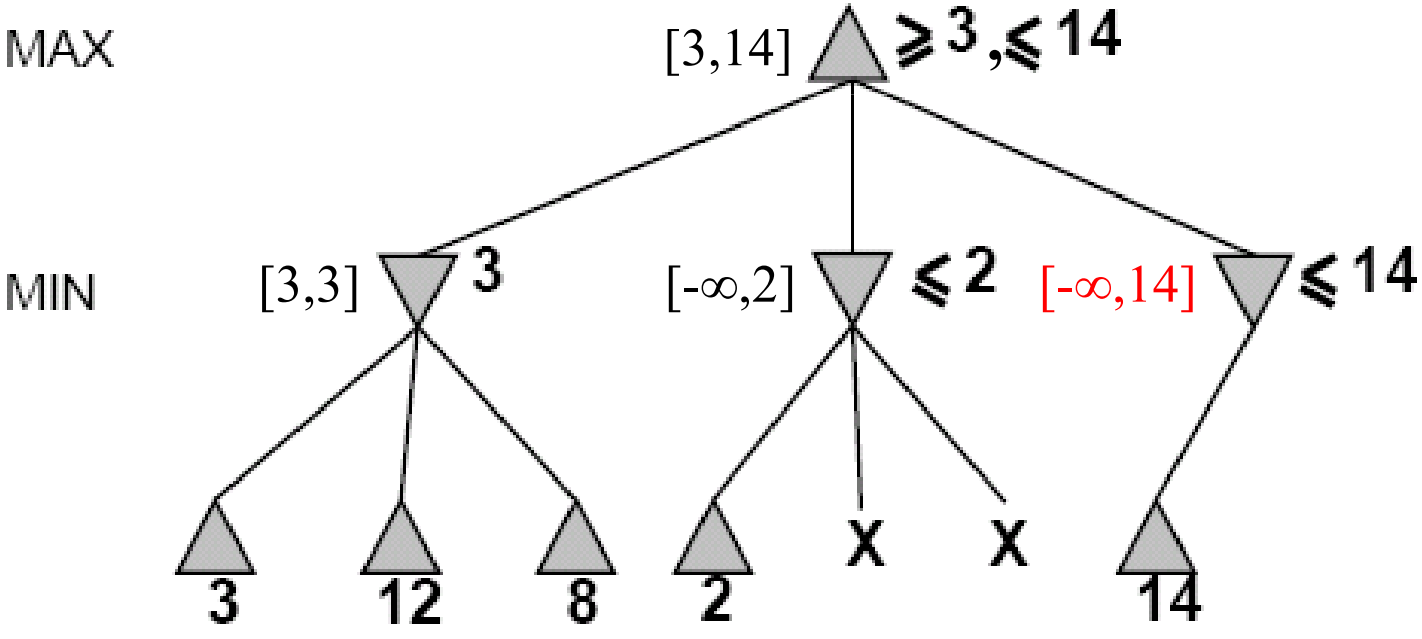
جستجوی خصمانه

مثال: هرس آلفا-بتا



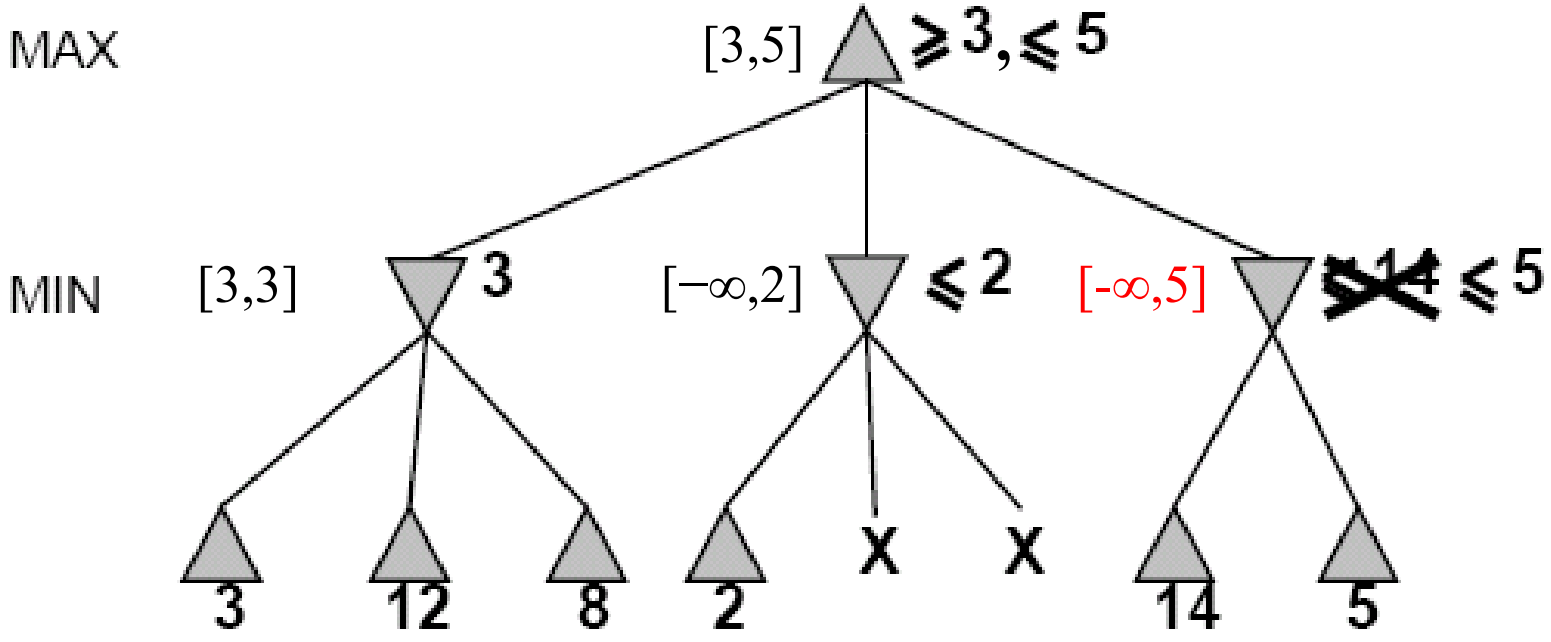
جستجوی خصمانه

مثال: هرس آلفا-بتا



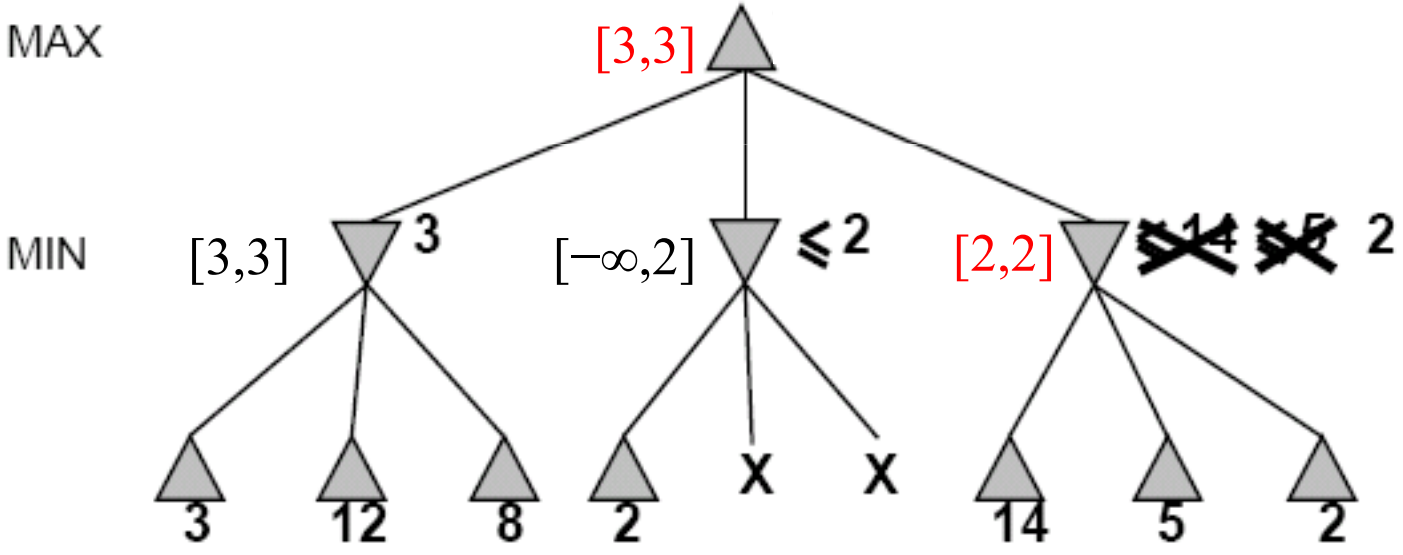
جستجوی خصمانه

مثال: هرس آلفا-بتا



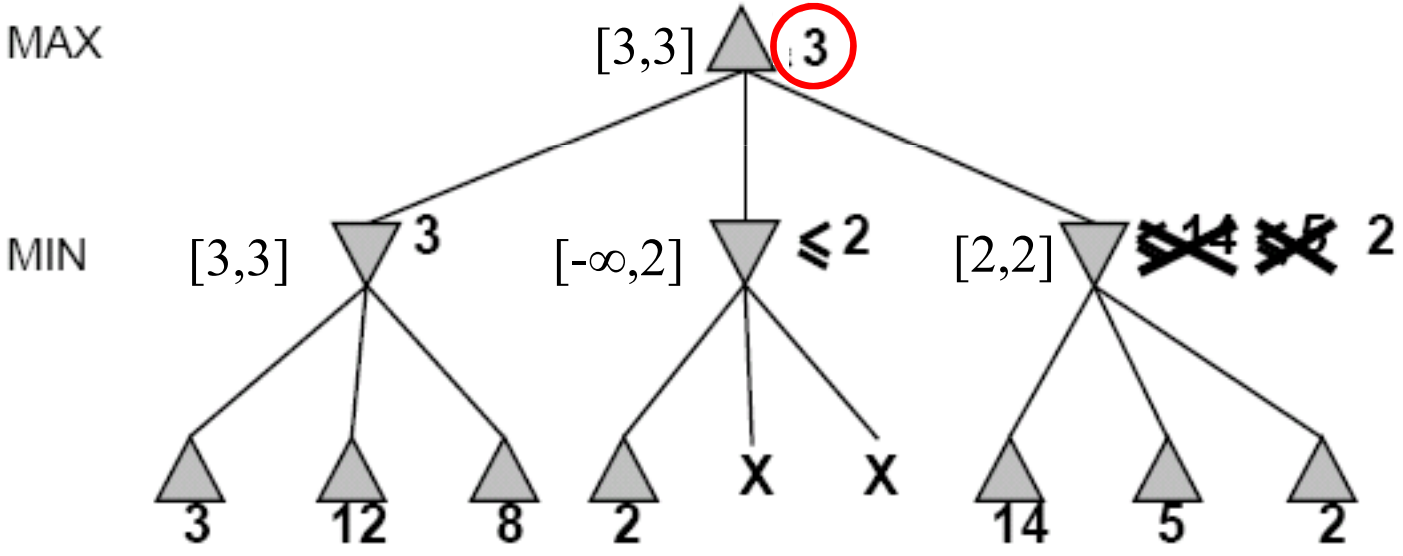
جستجوی خصمانه

مثال: هرس آلفا-بتا



جستجوی خصمانه

مثال: هرس آلفا-بتا



جستجوی خصمانه

بازیهای قطعي با اطلاعات ناقص

معایب الگوریتم های پیشین

- الگوریتم minimax کل فضای جست و جوی بازی را تولید میکند
- الگوریتم آلفا-بتا با وجود هرس درخت ، اما کل مسیر حالتهای پایانه ، حداقل برای بخشي از فضای حالت ، باید جست و جو شود
  - این عمق عملي نیست ، زیرا حرکات باید در زمانی معقول انجام شود

شانون (1950)

برای کمتر شدن زمان جست و جو و اعمال تابع ارزیابي اکتشافی به حالتهای جستجو ، بهتر است از گره های غیر پایانه به گره های پایانه پرداخته شود

جستجوی خصمانه

بازیهای قطعي با اطلاعات ناقص

- در شانون, minimax و آلفا-بتا به دو روش بطور متناوب عمل میکنند
- جایگزيني تابع سودمندی با تابع ارزیابی اکتشافی بنام EVAL
  - تخمینی از سودمندی موقعیت ارائه میکند
- جایگزین تست پایانه با تست توقف
  - تصمیم میگیرد EVAL چه موقع اعمال شود

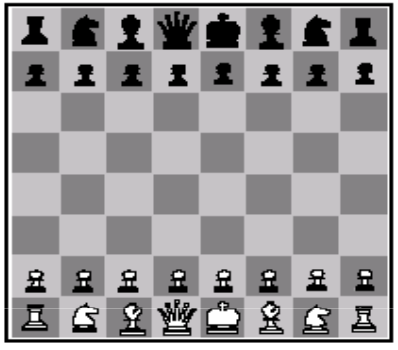
جستجوی خصمانه

تابع ارزیابي اکتشافی EVAL

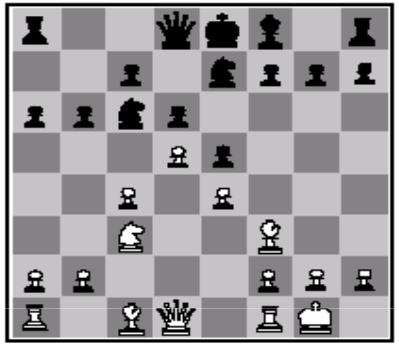
- تابع ارزیابی ، ارائه تخمینی از سودمندی مورد انتظار بازی از یک موقعیت خاص
  - توابع اکتشافی ، تخمینی از فاصله تا هدف را بر میگرداندند
- اغلب توابع ارزیابي ، **خواص** گوناگونی از حالتها را محاسبه میکنند
  - خواص روی هم رفته ، کلاسهای هم ارزی یا دسته های مختلفی از حالتها را تعریف میکنند
  - حالتهای هر دسته ، برای تمام خواص مقدار یکسانی دارند
- هر دسته حاوی چند حالت است که
  - موجب برنده شدن
  - موجب رسم شدن
  - منجر به باختن
- تابع ارزیابی نمیداند کدام حالت منجر به چه چیزی میشود ، اما میتواند مقداری برگرداند که تناسب حالتها را با هر نتیجه نشان دهد

جستجوی خصمانه

مثال: تابع EVAL

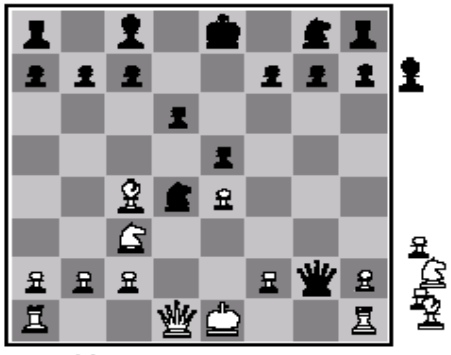


(a) White to move
Fairly even

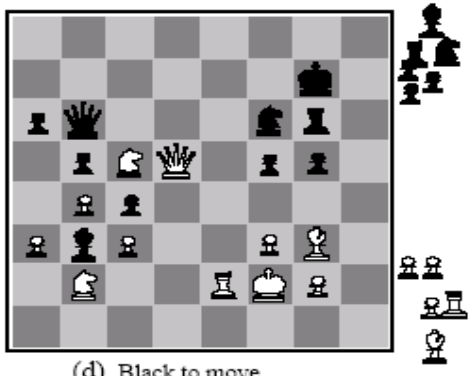


(b) Black to move
White slightly better

f_i



(c) White to move
Black winning



(d) Black to move
White about to lose

- اغلب توابع ارزیابی, مقدار عددی جداگانه ای برای هر خاصیت محاسبه ، سپس آنها را ترکیب میکنند تا مقدار کل بدست آید

مثال در تابع بازی شطرنج:

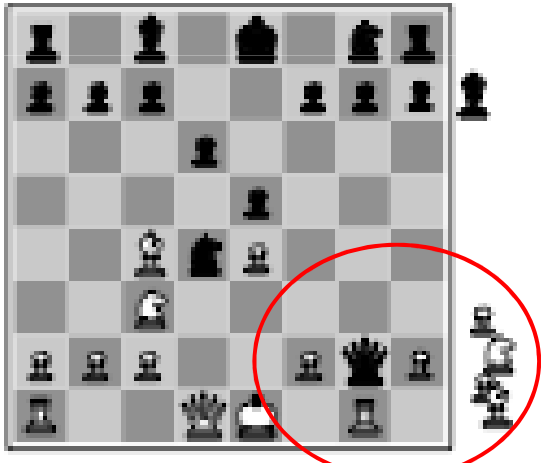
- تعداد هر نوع قطعه در صفحه f_i
- مقادیر آن قطعات(1 برای پیاده ، 3 برای اسب یا فیل ،5 برای رخ و ...) w_i

$$Eval(s) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s)$$

جستجوی خصمانه

مثال: تابع EVAL

ارزیابی تابع EVAL از مقدار پیروزی در دو موقعیت کاملا متفاوت



ب) سفید حرکت میکند

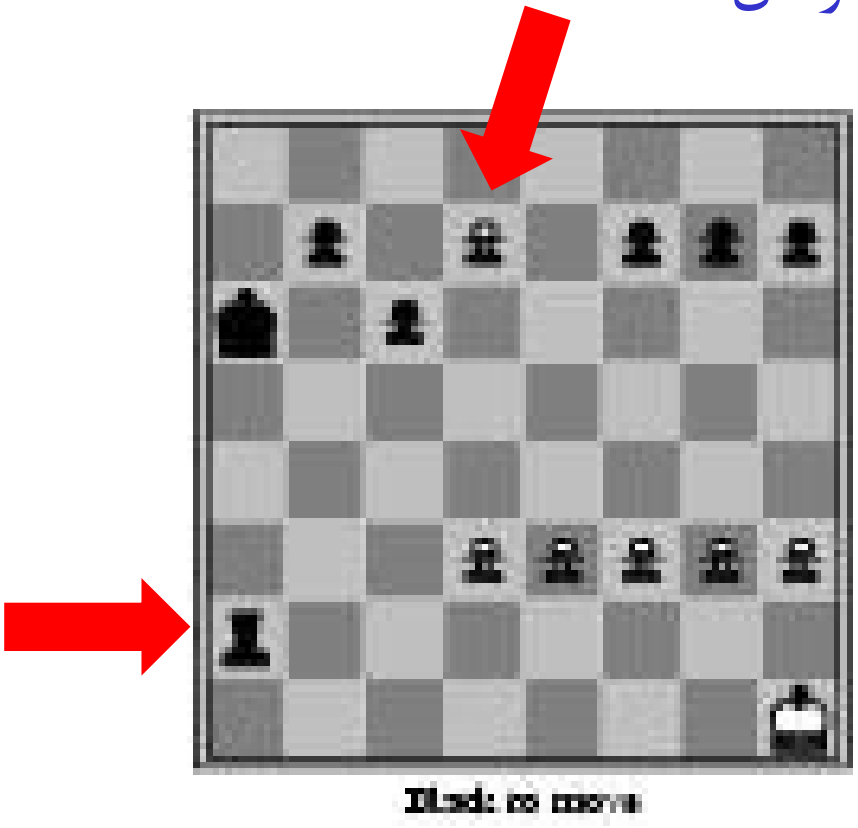
الف) سفید حرکت میکند

الف) سیاه ، مزیت اسب و دو پیاده دارد و بازي را میبرد

ب) پس از اینکه سفید ، وزیر را در اختیار میگیرد ، سیاه میبازد

جستجوی خصمانه

اثر افق

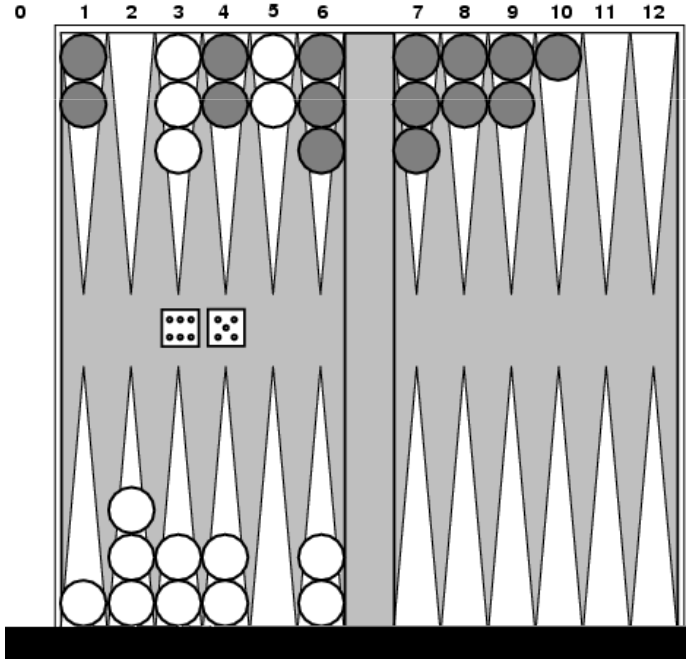


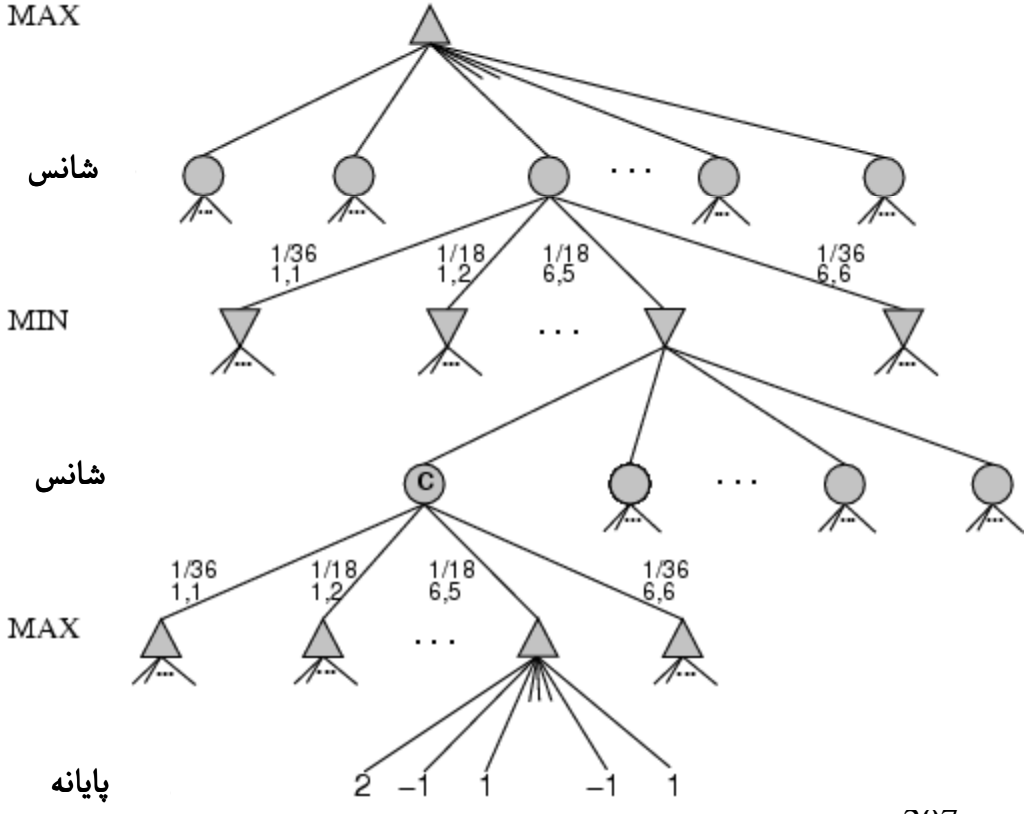
وقتی بوجود مي آید که برنامه با اثری از رقیب مواجه شود که منجر به خرابی جدی گشته و اجتناب پذیر است

مثال: شکل مقابل ؛
سیاه در اصل جلوست ، اما اگر سفید پیاده اش را از سطر هفتم به هشتم ببرد ، پیاده به وزیر تبدیل میشود و موقعیت برد برای سفید بوجود مي آید

جستجوی خصمانه

بازیهایی که حاوی عنصر شانس هستند





هوش مصنوعي

فصل هفتم

عامل های منطقی

هوش مصنوعي Artificial Intelligence

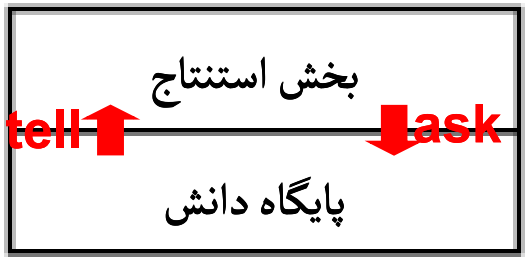
فهرست

- عاملهای مبتنی بر دانش
- منطق
- منطق گزاره ای
- الگوهای استدلال در منطق گزاره ای
- الگوریتم **resolution**
- زنجیر پیشرو و عقبگرد

عاملهای منطقی

عاملهای مبتنی بر دانش

- مؤلفه اصلي عامل مبتنی بر دانش ، **پایگاه دانش** آن است
  - پایگاه دانش: مجموعه ای از جملات
  - جمله: **زبان نمایش دانش** و بيان ادعاهايي در مورد جهان



- برای اضافه کردن جملات به پایگاه دانش و درخواست دانسته ها
  - TELL و ASK
  - هر دو ممکن است شامل استنتاج باشند
- پیروی:انجام فرایند استنتاج تحت مقررات خاص

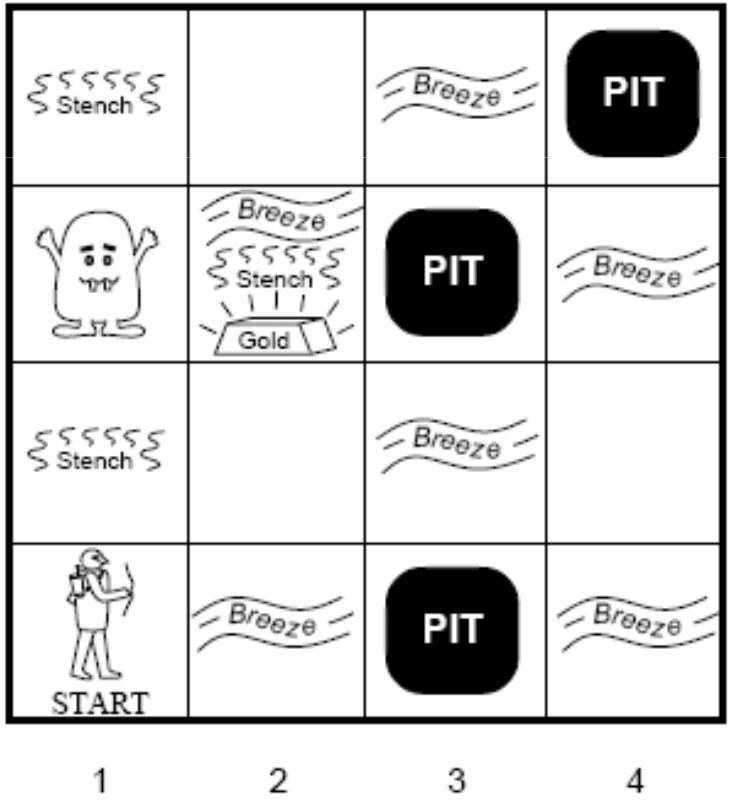
عاملهای منطقی

عاملهای مبتنی بر دانش

- عامل مبتنی بر دانش باید بتواند:
  - نمایش حالات و فعالیتها
  - ترکیب ادراکات جدید
  - بروز کردن تصور داخلی خود از جهان
  - استنباط خصوصیات مخفی جهان
  - استنتاج فعالیتهای مناسب
- عامل پایگاه دانش خیلی شبیه به عاملهایی با حالت درونی است
- عاملها در دو سطح متفاوت تعریف میشوند:
  - سطح دانش: عامل چه چیزی میداند و اهداف آن کدامند؟
  - سطح پیاده سازی: ساختمان داده اطلاعات پایگاه دانش و چگونگی دستکاری آنها

عاملهای منطقی

جهان WUMPUS



معیار کارایی:

- 1000+ انتخاب طلا ، 1000- افتادن در گودال یا خورده شدن ، 1- هر مرحله ، 10- برای 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:

- بو تعفن ، نسیم ، تابش ، ضربه ، جیغ زدن

محرکها:

- گردش به چپ ، گردش به راست ، جلو رفتن ، برداشتن ، انداختن ، شلیک کردن

عاملهای منطقی

توصیف جهان WUMPUS

قابل مشاهده کامل: خیر, فقط ادراک محلي

قطعی: بله ، نتیجه دقیقا مشخص است

رویدادی: خیر ، ترتیبي از فعالیتهاست

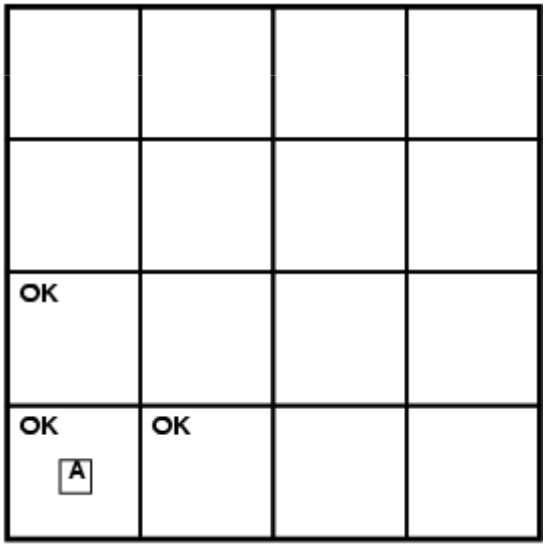
ایستا: بله, WUMPUS و گودالها حرکت ندارند

گسسته: بله

تک عامله: بله ، WUMPUS در اصل یک خصوصیت طبیعي است

عاملهای منطقی

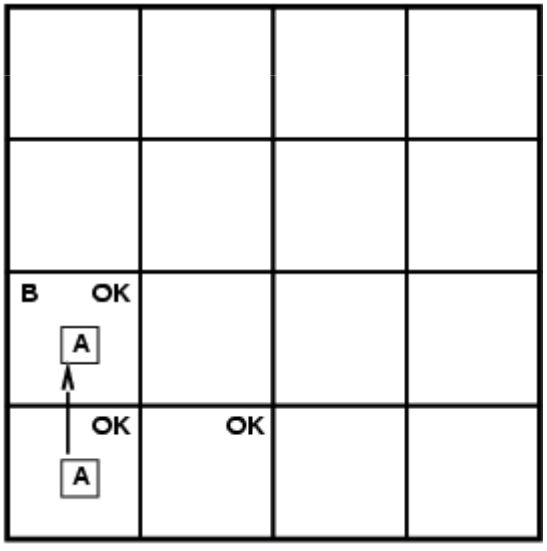
کاوش در جهان WUMPUS



A = عامل
B = نسیم
G = درخشش ،طلا
OK = مربع امن
P = گودال
S = تعفن
V = ملاقات شده
W = Wumpus

عاملهای منطقی

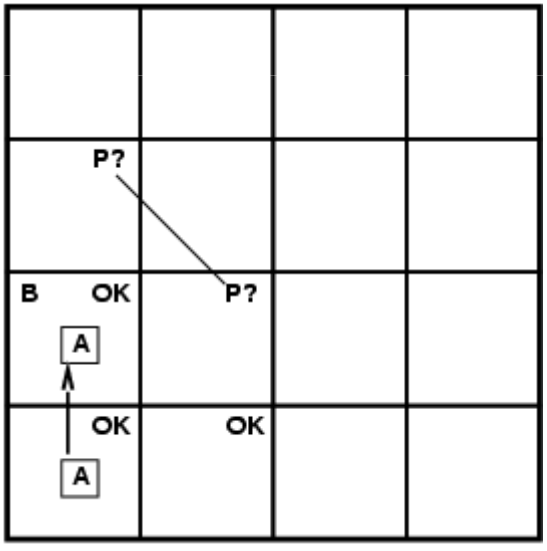
توصیف جهان WUMPUS



A = عامل
B = نسیم
G = درخشش ،طلا
OK = مربع امن
P = گودال
S = تعفن
V = ملاقات شده
W = Wumpus

عاملهای منطقی

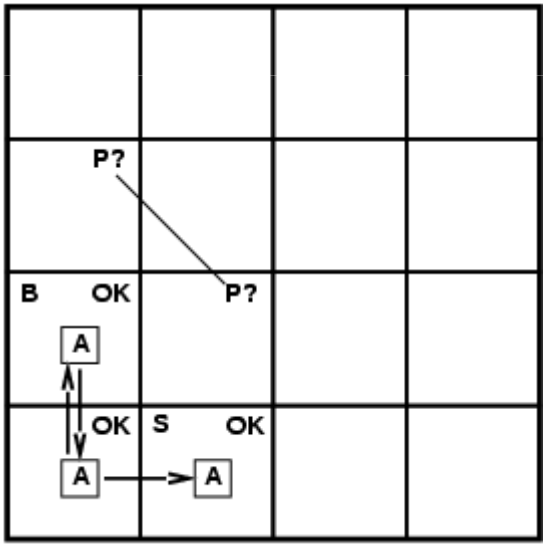
توصیف جهان WUMPUS



A = عامل
B = نسیم
G = درخشش ،طلا
OK = مربع امن
P = گودال
S = تعفن
V = ملاقات شده
W = Wumpus

عاملهای منطقی

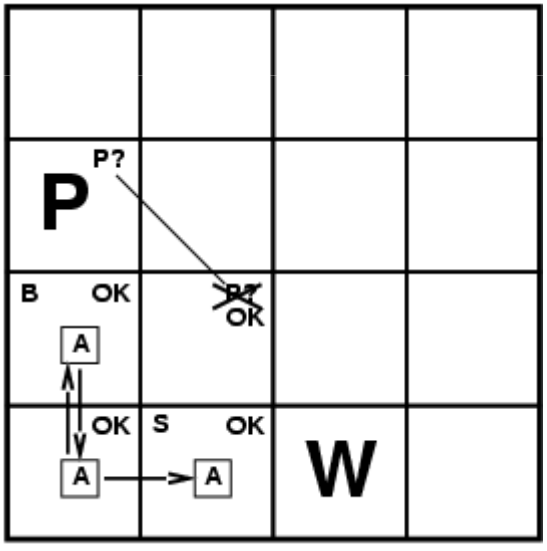
توصیف جهان WUMPUS



A = عامل
B = نسیم
G = درخشش ،طلا
OK = مربع امن
P = گودال
S = تعفن
V = ملاقات شده
W = Wumpus

عاملهای منطقی

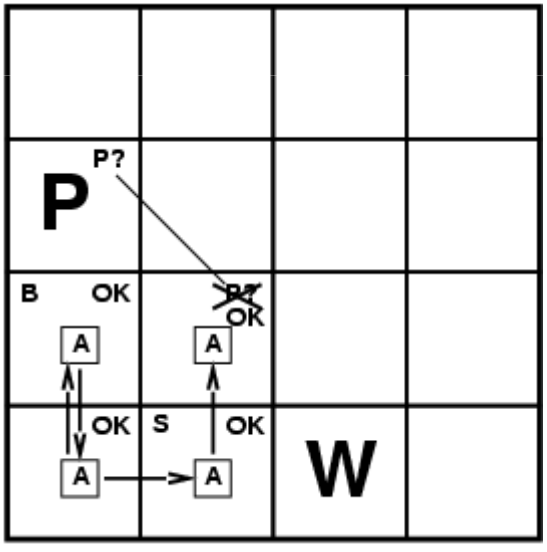
توصیف جهان WUMPUS



A = عامل
B = نسیم
G = درخشش ،طلا
OK = مربع امن
P = گودال
S = تعفن
V = ملاقات شده
W = Wumpus

عاملهای منطقی

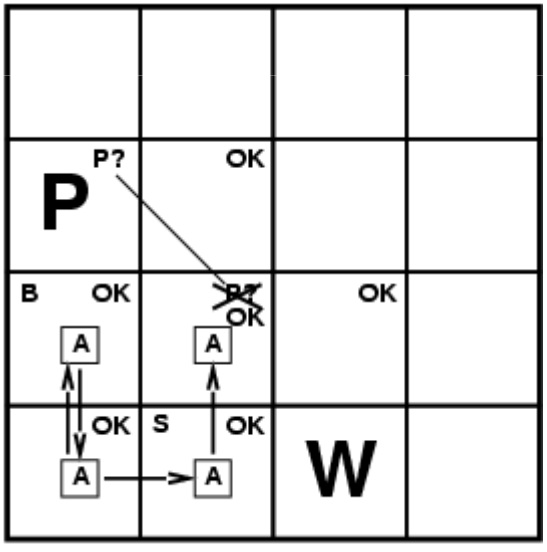
توصیف جهان WUMPUS



A = عامل
B = نسیم
G = درخشش ،طلا
OK = مربع امن
P = گودال
S = تعفن
V = ملاقات شده
W = Wumpus

عاملهای منطقی

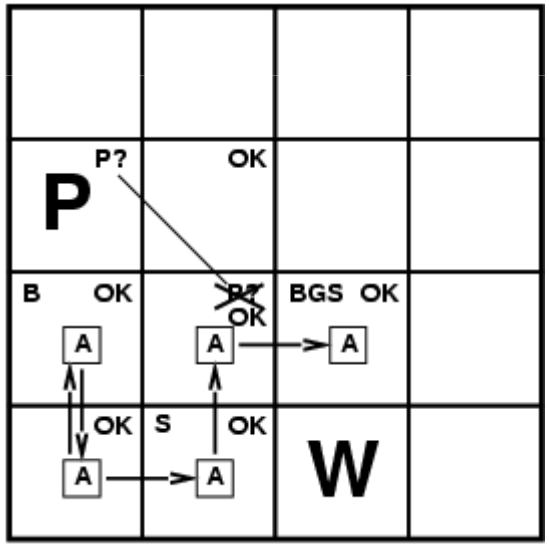
توصیف جهان WUMPUS



A = عامل
B = نسیم
G = درخشش ،طلا
OK = مربع امن
P = گودال
S = تعفن
V = ملاقات شده
W = Wumpus

عاملهای منطقی

توصیف جهان WUMPUS



A = عامل
B = نسیم
G = درخشش ،طلا
OK = مربع امن
P = گودال
S = تعفن
V = ملاقات شده
W = Wumpus

عاملهای منطقی

منطق

- یک زبان رسمي:
  - **ترکیب**(نحو): چه کلمه بندی صحیح است.(خوش فرم)
  - **معناشناسی**: یک کلمه بندی صحیح چه معنايي دارد
  - در منطق ، معنای زبان ، درستی هر جمله را در برابر هر جهان ممکن تعریف میکند

- مثال ، در زبان ریاضیات
  - X+2 >= y یک جمله اما x2+y جمله نیست
  - X+2 >= y در جهان درست است اگر x=7 و y =1
  - X+2 >= y در جهان غلط است اگر x=0 و y =6

عاملهای منطقی

استلزام

استلزام منطقي بين جملات اين است كه جمله ای بطور منطقي از جمله ديگر پيروی ميكند

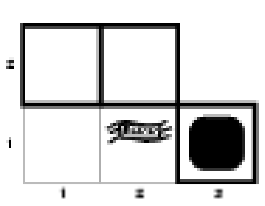
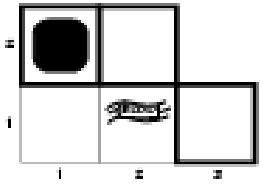
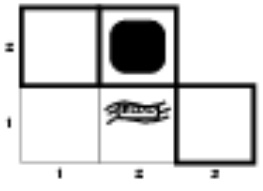
$$a \models b$$

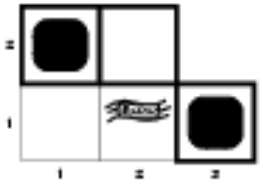
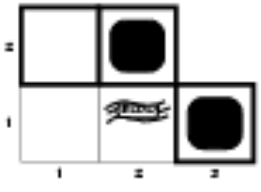
- جمله a استلزام جمله b است
- جمله a جمله b را ايجاد ميكند
- اگر و فقط اگر ، در هر مدلي كه a درست است ، b نيز درست است
- اگر a درست باشد ، b نيز درست است
- درستی b در درستي a نهفته است

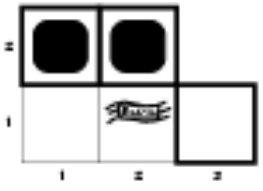
مثال: جمله $x+y=4$ مستلزم جمله $4=x+y$ است

عاملهای منطقی

مدلهای Wumpus

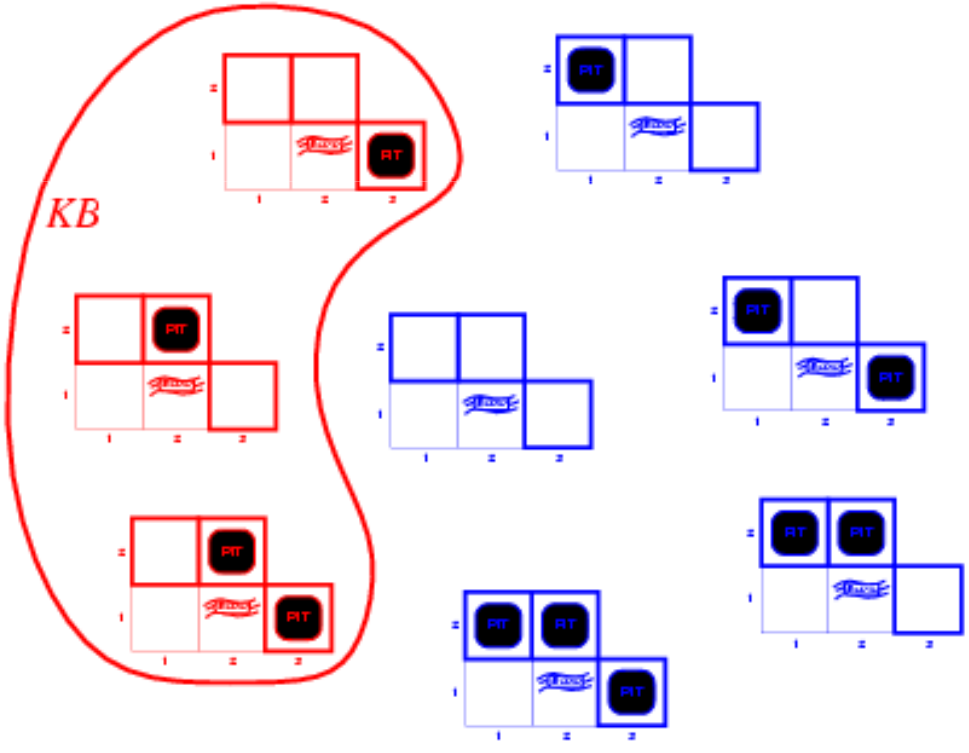






عاملهای منطقی

مدلهای Wumpus



KB

=

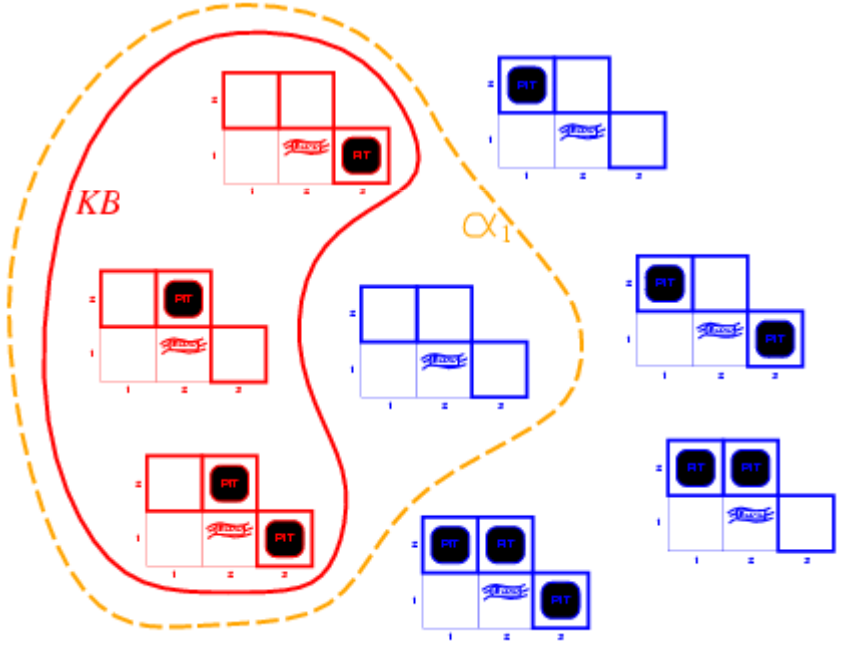
قوانین دنیای Wumpus

+

مشاهدات

عاملهای منطقی

مدلهای Wumpus

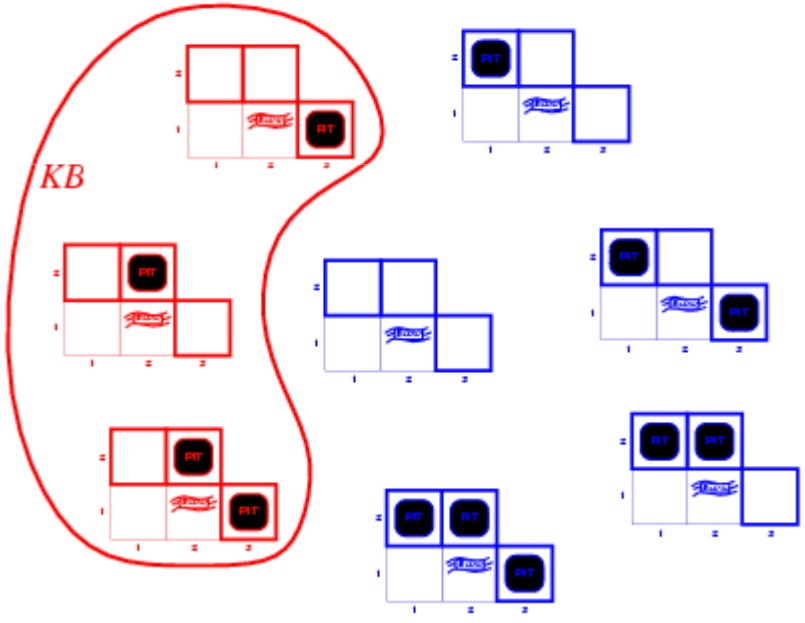


KB = wumpus دنیای + مشاهدات

$\alpha 1$ = "[1,2] امن است", $KB \models \alpha 1$

عاملهای منطقی

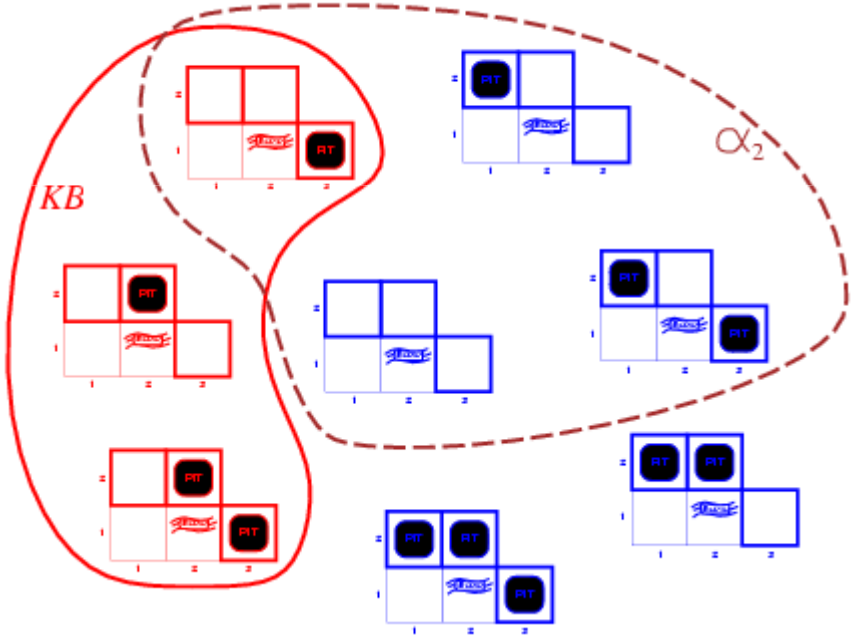
مدلهای Wumpus



KB = wumpus دنياي + مشاهدات

عاملهای منطقی

مدلهای Wumpus



KB = wumpus دنیای + مشاهدات

α_2 = "[2,2] امن است", $KB \not\models \alpha_2$

عاملهای منطقی

منطق گزاره ای

- نحو منطق گزاره ای ، جملات مجاز را تعریف میکند
- جملات اتمیک(عناصر غیر قابل تعمیم) تشکیل شده از یک نماد گزاره
- هر یک از این نمادها به گزاره ای درست یا نادرست اختصاص دارد
  - نمادها از حروف بزرگ مثل R,Q,P استفاده میکنند
- جملات پیچیده با استفاده از رابطهای منطقي ، از جملات ساده تر ساخته میشوند
  - ┐(not) جمله ای مثل $W_{1,3}$ ┐ $W_{1,3}$ نقیض است
  - لیترال یک جمله اتمیک(لیترال مثبت) ، یا یک جمله اتمیک منفي(لیترال منفي) است
  - ^(and) مثل $W_{1,3}$ ^ $P_{1,3}$ ترکیب عطفی نام دارد.هر بخش آن یک عطف نامیده میشود
  - v(or) مثل $W_{2,2}$ v ($W_{1,3}$ ^ $P_{3,1}$) ترکیب فصلي مربوط به فصل های $W_{2,2}$ و $W_{1,3}$ ^ $P_{3,1}$
  - =>(استلزام): W2,2 ┐ v (W1,3 ^ P3,1) استلزام یا شرطی نامیده میشود. مقدمه یا مقدم آن $W_{1,3}$ ^ $P_{3,1}$ و نتیجه یا تالي آن $W_{2,2}$ ┐ است
  - ⇔ جمله $W_{2,2}$ ⇔ $W_{1,3}$ دو شرطی نام دارد

عاملهای منطقی

منطق گزاره ای

$$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha) \quad \text{commutativity of } \wedge$$
$$(\alpha \vee \beta) \equiv (\beta \vee \alpha) \quad \text{commutativity of } \vee$$
$$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma)) \quad \text{associativity of } \wedge$$
$$((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma)) \quad \text{associativity of } \vee$$
$$\neg(\neg\alpha) \equiv \alpha \quad \text{double-negation elimination}$$
$$(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha) \quad \text{contraposition}$$
$$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta) \quad \text{implication elimination}$$
$$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) \quad \text{biconditional elimination}$$
$$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta) \quad \text{de Morgan}$$
$$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta) \quad \text{de Morgan}$$
$$(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) \quad \text{distributivity of } \wedge \text{ over } \vee$$
$$(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) \quad \text{distributivity of } \vee \text{ over } \wedge$$

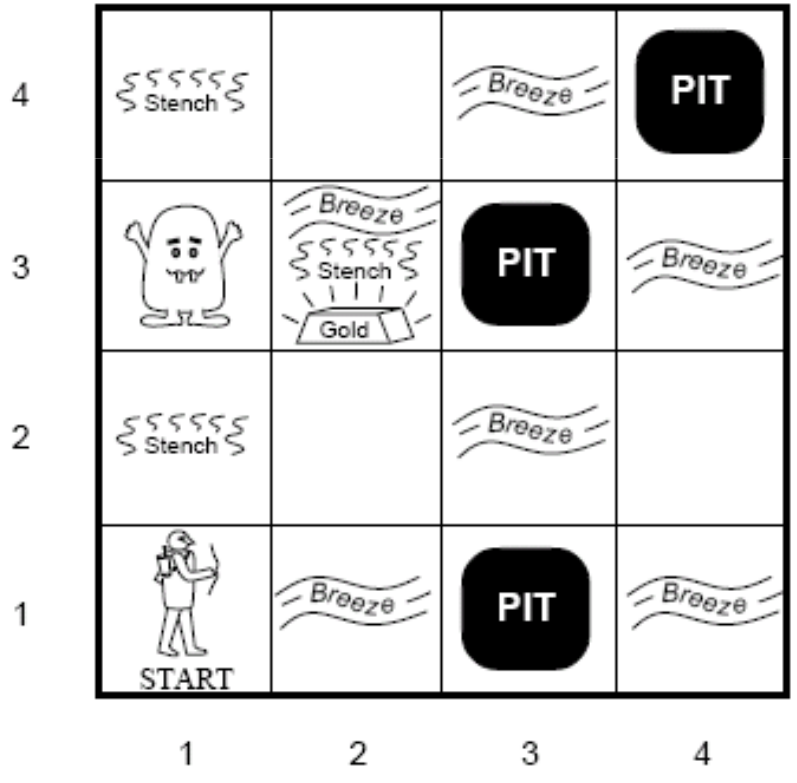
عاملهای منطقی

جدول درستی پنج رابطه منطقی

P	Q	¬ P	P ^ Q	P ν Q	P=>Q	P⇔Q
F	F	T	F	F	T	T
F	T	T	F	T	T	F
T	F	F	F	T	F	F
T	T	F	T	T	T	T

عاملهای منطقی

منطق گزاره ای در دنیای Wumpus



در $B_{1,1}$ نسیمي وجود دارد

$$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$$

در [1,1] گودالی وجود ندارد

$$R_1: \neg P_{1,1}$$

عاملهای منطقی

الگوهای استدلال در منطق گزاره ای

- قوانین استنتاج: الگوهايي استاندارد که زنجیره ای از نتایج را برای رسیدن به هدف ایجاد میکند
- قیاس استثنايي: با استفاده از ترکیب عطفی ، میتوان هر عطف را استنتاج کرد(یعنی هر وقت جمله ای به شکل a=>b داده شود ، جمله b را میتوان استنتاج کرد.)

$$\frac{\alpha \Rightarrow \beta, \alpha}{\beta}$$

- میتوان از

(WumpusAhead ^ WumpusAlive)

و

(WumpusAhead ^ WumpusAlive) => Shoot

Shoot را استنتاج کرد

عاملهای منطقی

حذف and: هر عطف را میتوان از ترکیب عطفی استنتاج کرد

مثال: WumpusAlive را میتوان از جمله زیر استناج کرد
(WumpusAhead ^ WumpusAlive)

$$\frac{\alpha \wedge \beta}{\alpha}$$

خاصیت یکنواختی

مجموعه ای از جملات استلزامی که فقط میتواند در صورت اضافه شدن اطلاعات به پایگاه دانش رشد کند.

برای جملات a و b داریم:

$$KB \models \alpha \Rightarrow KB \wedge \beta \models \alpha$$

عاملهای منطقی

قانون resolution

قانون resolution واحد ، یک عبارت و یک لیترال را گرفته ، عبارت دیگری تولید میکند

$$\frac{l_1 \vee \ldots \vee l_k, m}{l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k}$$

قانون resulotion واحد میتواند به قانون resulotion کامل تعمیم داد:

$$\frac{l_1 \vee \ldots \vee l_k, m_1 \vee \ldots \vee m_n}{l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n}$$

عاملهای منطقی

الگوریتم resolution

شکل نرمال عطفی(CNF): جمله ای که بصورت ترکیب عطفی از ترکیبات فصلي ليترالها بيان ميشود.در هر عبارت موجود در جمله k-CNF دقیقا k لیترال وجود دارد

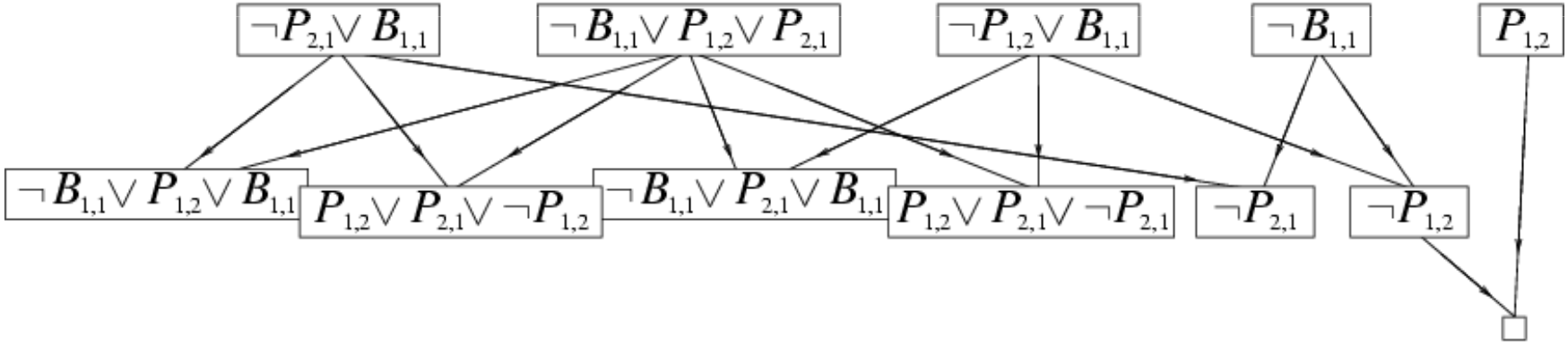
$$(l_{1,1} \vee \ldots \vee l_{1,k}) \wedge \ldots \wedge (l_{n,1} \vee \ldots \vee l_{n,k})$$

الگوریتم resolution:

- برای اینکه نشان دهیم KB|=a, مشخص میکنیم (KB ^ ┐ a) ارضا کننده نیست
- ابتدا (KB ^ ┐ a) را به CNF تبدیل میکنیم
- سپس قانون resulotion به عبارات کوچک حاصل اعمال میشود
- هر جفتی که شامل لیترالهای مکمل باشد ، resulotion میشود تا عبارت جدیدی ایجاد گردد
- اگر این عبارت قبلا در مجموعه نباشد ، به آن اضافه میشود
- فرايند تا محقق شدن يكي از شروط زير ادامه مي يابد:
  - هیچ عبارت دیگری وجود نداشته باشد که بتواند اضافه شود. در این مورد ، b استلزام a نیست
  - کاربرد قانون resolution ، عبارت تهی را بدست میدهد که در این مورد ، b استلزام a است

عاملهای منطقی

مثال:الگوریتم resolution



$KB = (B1,1 \Leftrightarrow (P1,2 \vee P2,1)) \wedge \neg B1,1$ $\alpha = \neg P1,2$

عاملهای منطقی

زنجیر پیشرو و عقبگرد

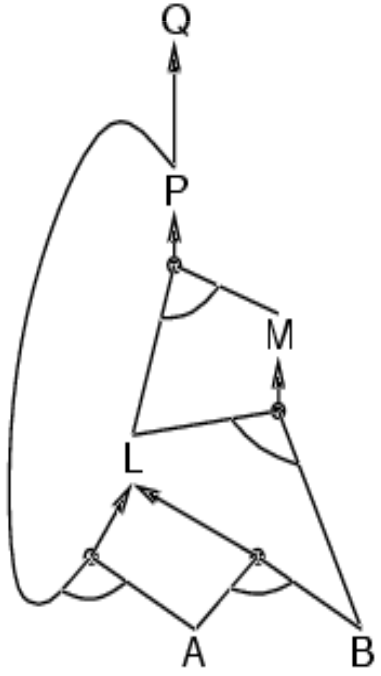
- عبارات هورن: ترکیب فصلی لیترالهایي است که فقط یکي از آنها مثبت است
  - هر عبارت **هورن** را میتوان به صورت یک استلزام نوشت که مقدمه آن ترکیب عطفی لیترالهای مثبت و تالی آن یک لیترال مثبت است
  - این نوع عبارات هورن که فقط یک لیترال مثبت دارند ، **عبارات معین** نامیده میشوند
  - لیترال مثبت را **رأس** و لیترالهای منفی را **بدنه عبارت** گویند
  - عبارت معیني که فاقد لیترالهای منفی باشد ، گزاره ای بنام **حقیقت** نام دارد
  - **عبارات معین** اساس **برنامه نویسي منطقی** را میسازد
- استنتاج با عبارات هورن ، از طریق الگوریتم های **زنجیر پیشرو** و **زنجیر عقبگرد** انجام میگیرد

عاملهای منطقی

زنجیر پیشرو

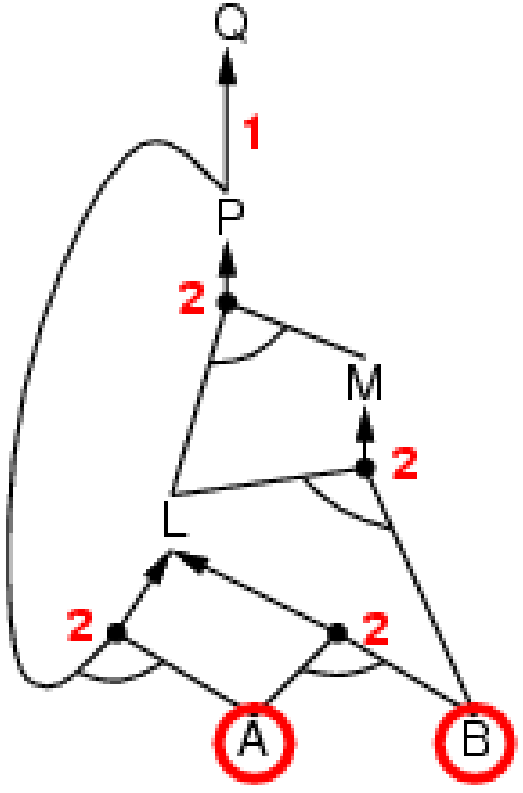
الگوریتم زنجیر پیشرو تعیین میکند آیا نماد گزاره ای q(تقاضا) ، توسط پایگاه دانش عبارات هورن ایجاب میشود یا خیر

$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$



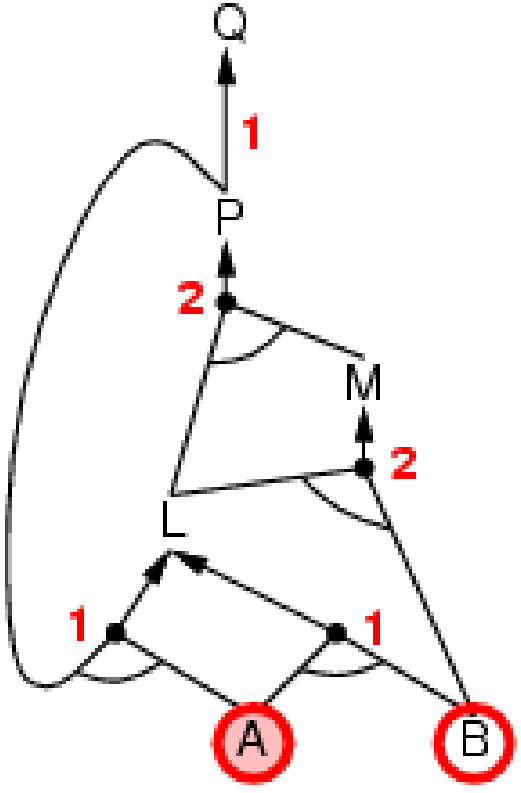
عاملهای منطقی

زنجیر پیشرو



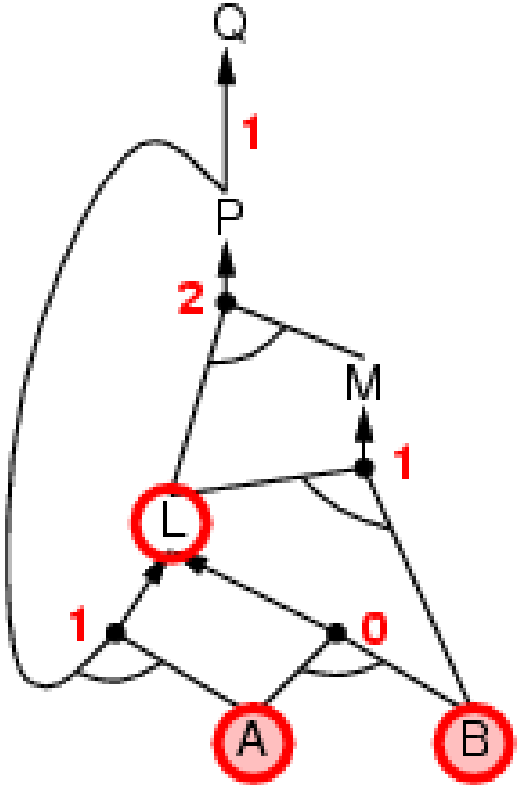
عاملهای منطقی

زنجیر پیشرو



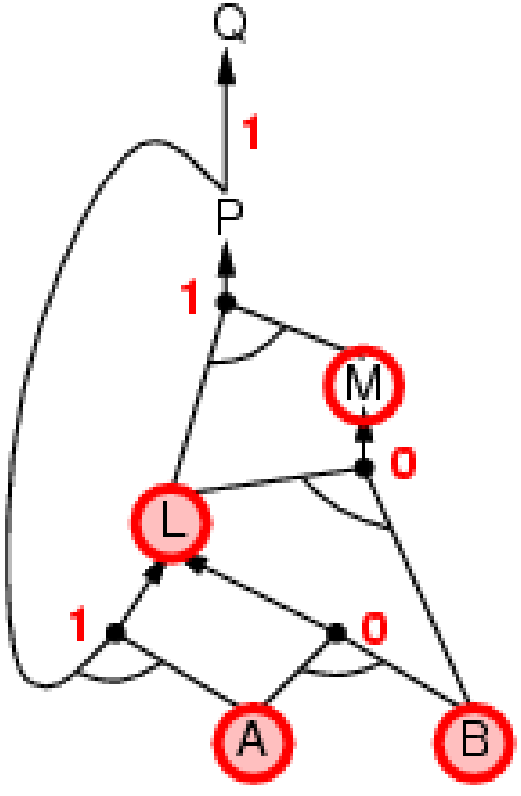
عاملهای منطقی

زنجیر پیشرو



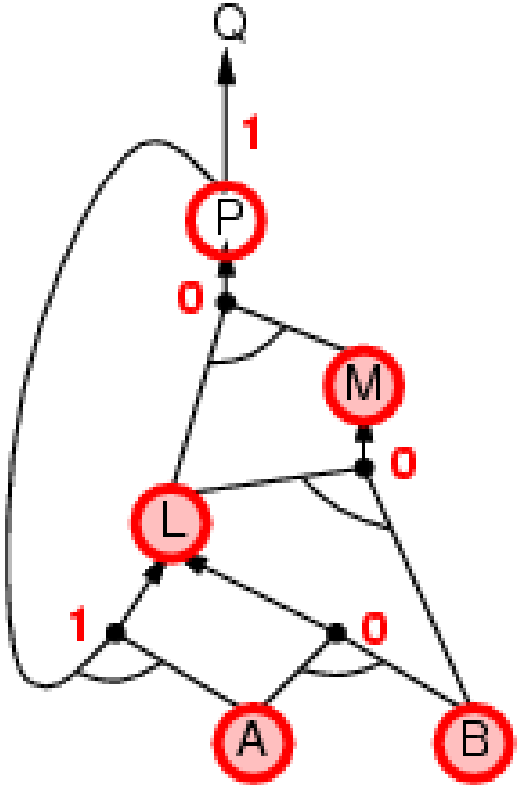
عاملهای منطقی

زنجیر پیشرو



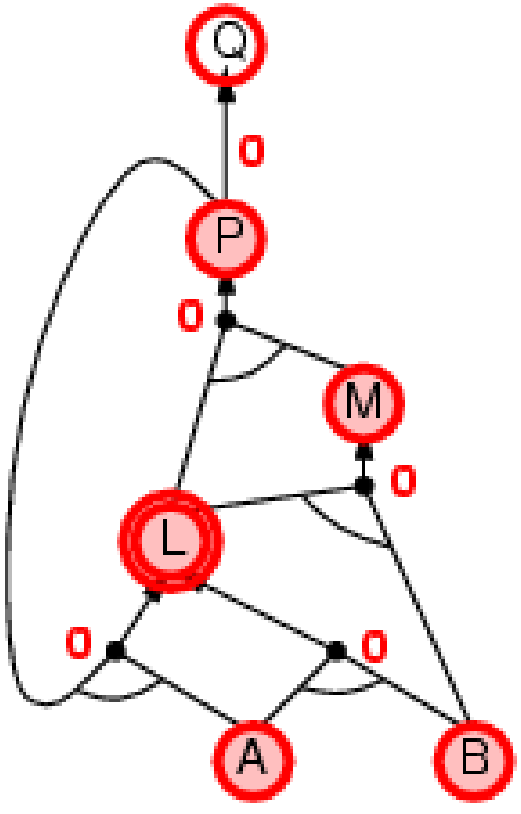
عاملهای منطقی

زنجیر پیشرو



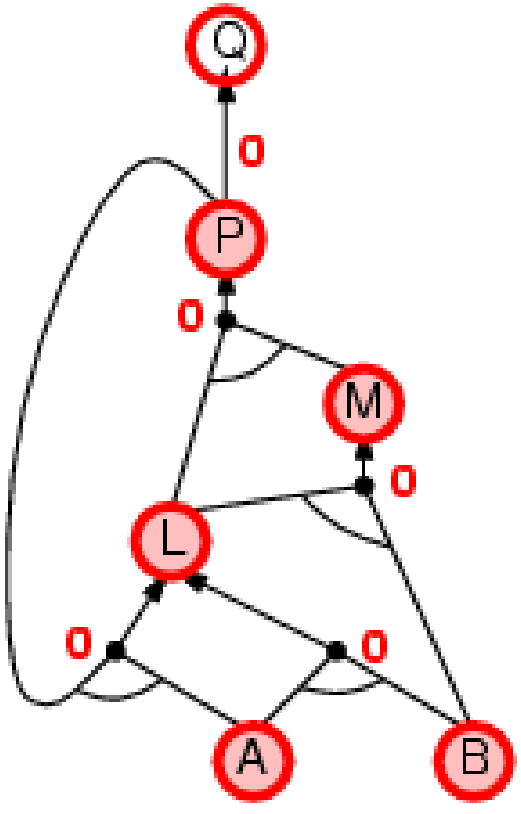
عاملهای منطقی

زنجیر پیشرو



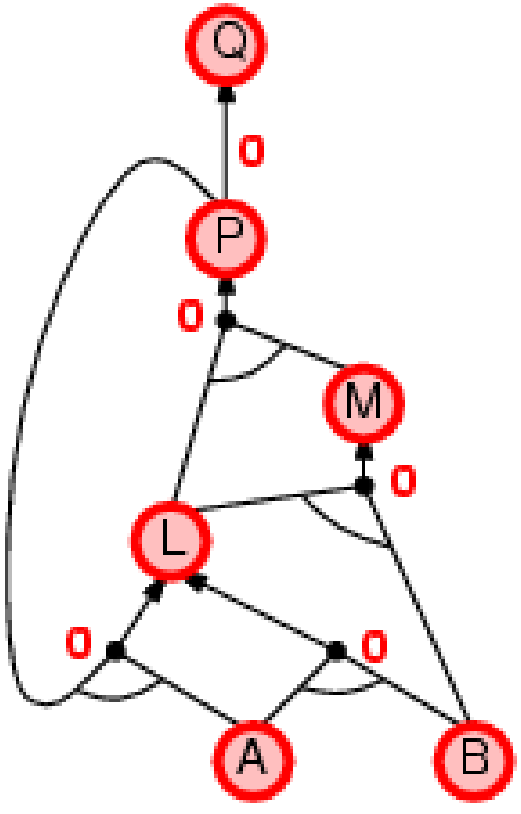
عاملهای منطقی

زنجیر پیشرو



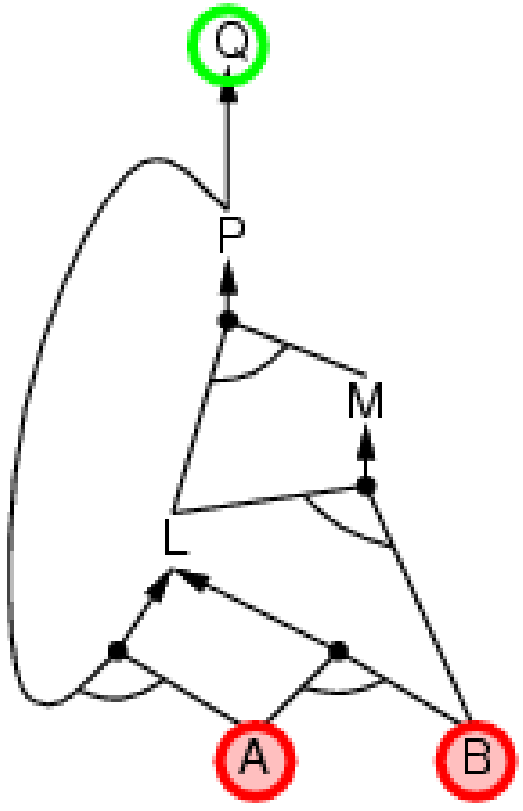
عاملهای منطقی

زنجیر پیشرو



عاملهای منطقی

الگوریتم عقبگرد کامل

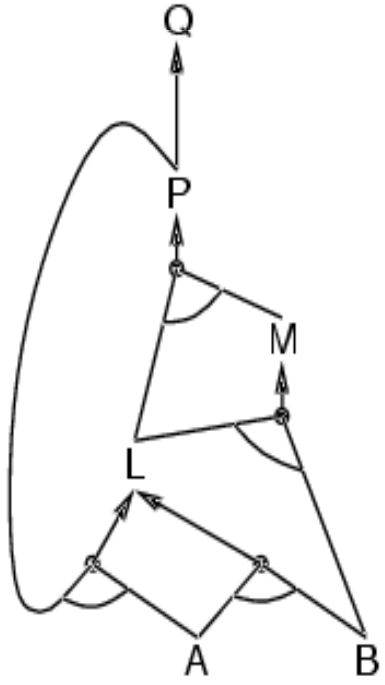


عاملهای منطقی

الگوریتم عقبگرد کامل

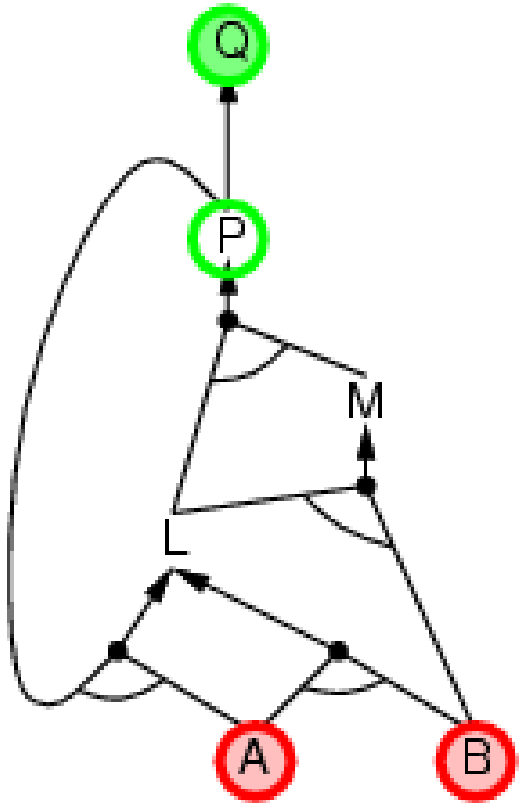
تغییرات عمده: خاتمه زودرس ، اکتشاف نماد محض ، اکتشاف عبارت واحد

$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$



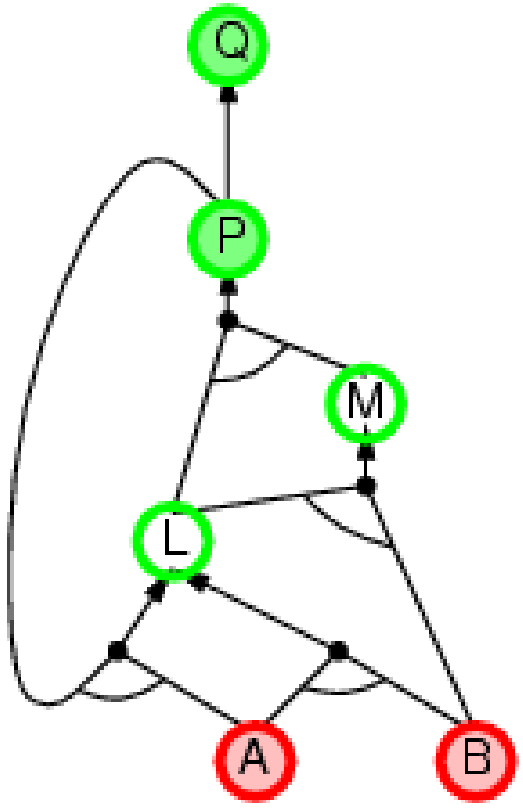
عاملهای منطقی

الگوریتم عقبگرد کامل



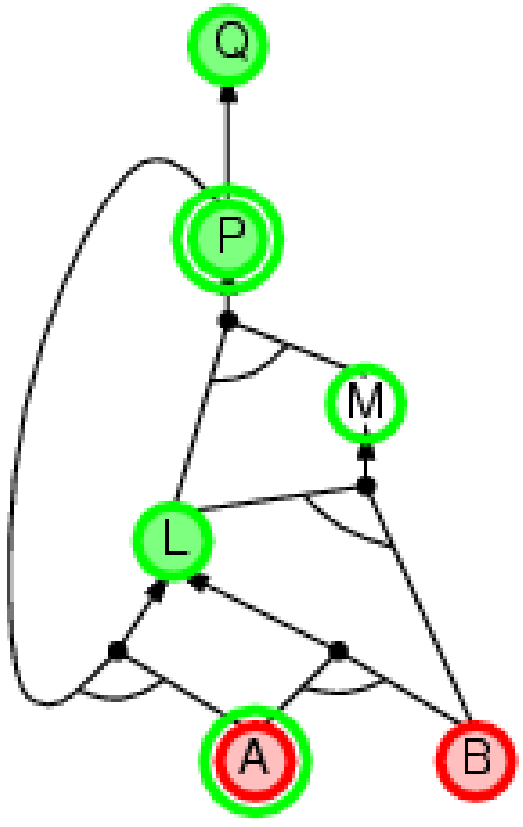
عاملهای منطقی

الگوریتم عقبگرد کامل



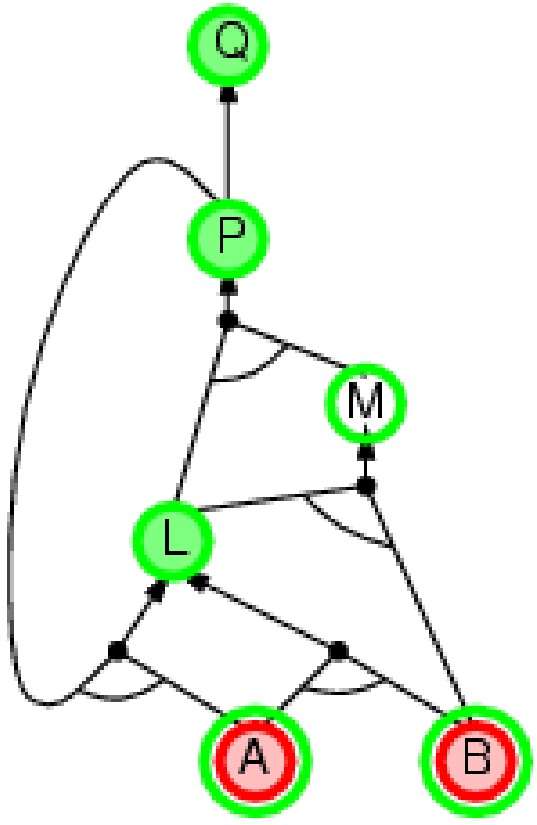
عاملهای منطقی

الگوریتم عقبگرد کامل



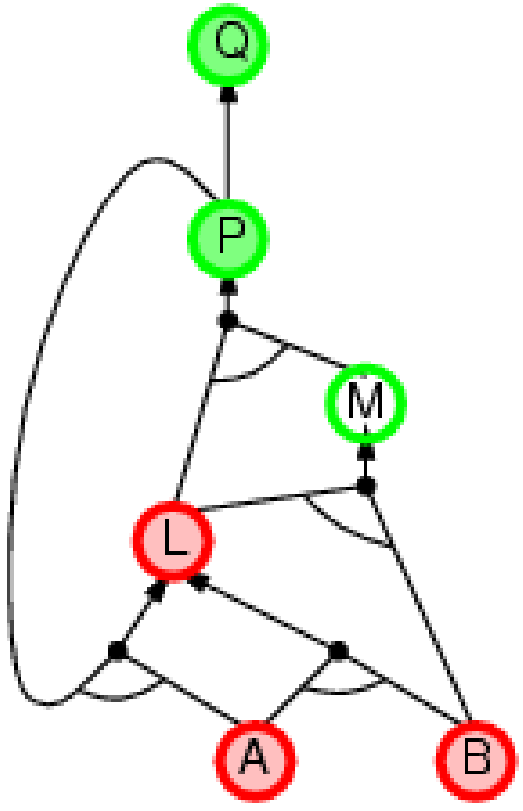
عاملهای منطقی

الگوریتم عقبگرد کامل



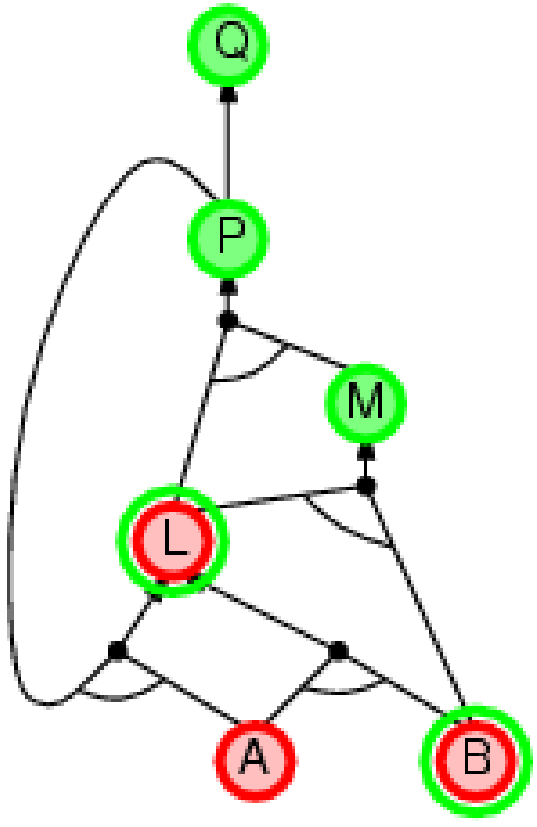
عاملهای منطقی

الگوریتم عقبگرد کامل



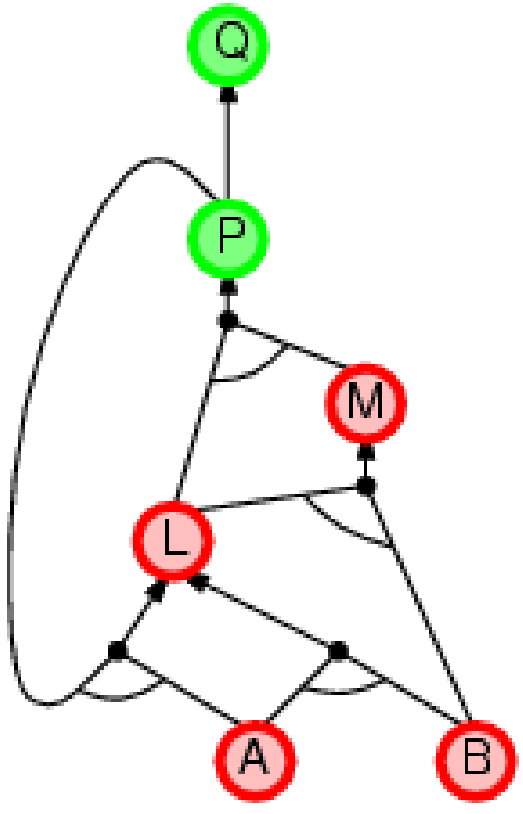
عاملهای منطقی

الگوریتم عقبگرد کامل



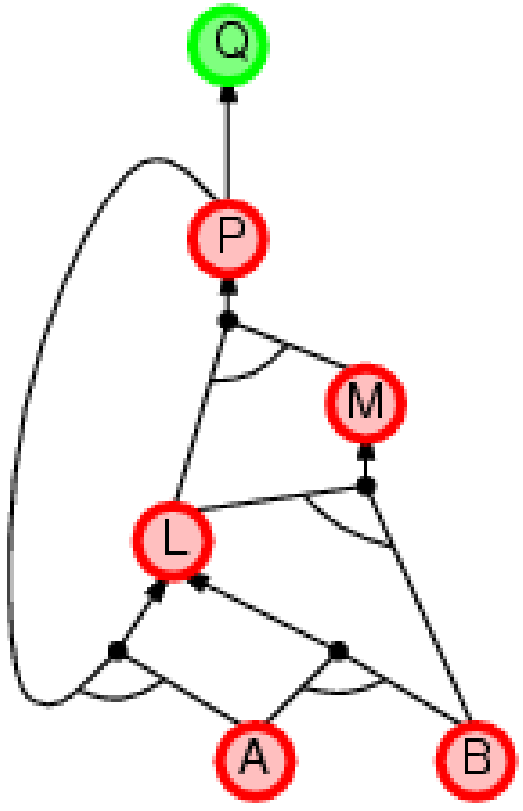
عاملهای منطقی

الگوریتم عقبگرد کامل



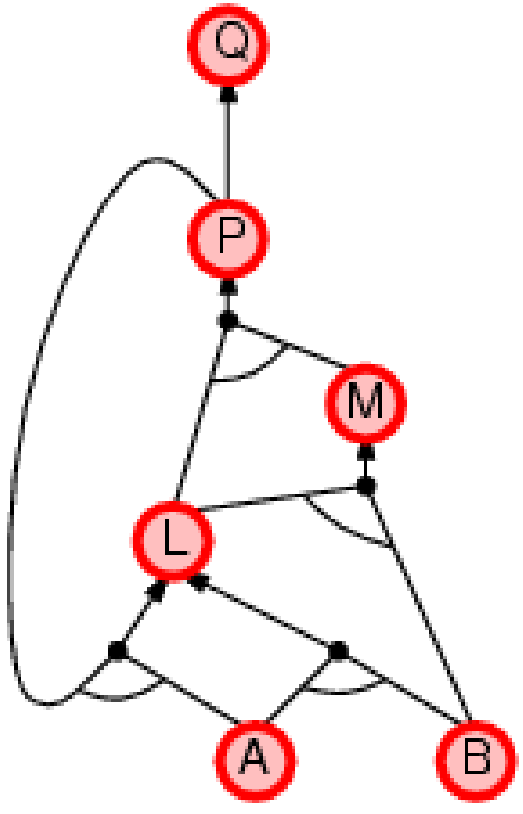
عاملهای منطقی

الگوریتم عقبگرد کامل



عاملهای منطقی

الگوریتم عقبگرد کامل



هوش مصنوعي

فصل هشتم

منطق رتبه اول

هوش مصنوعي Artificial Intelligence

فهرست

- مروری بر منطق گزاره ای
- منطق رتبه اول
- انواع منطق
- نحو و معنای منطق رتبه اول
- مهندسی دانش

منطق رتبه اول

مروری بر منطق گزاره ای

ویژگیها

- ماهیت اعلانی
  - دانش و استنتاج متمایزند و استنتاج کاملاً مستقل از دامنه است
- قدرت بیان کافی برای اداره کردن اطلاعات جزئي
  - با استفاده از ترکیب فصلی و نقیض
- قابلیت ترکیب
  - معنای جمله ، تابعي از معنای بخشهای آن
- معنا ، مستقل از متن است
  - بر خلاف زبانهای طبیعي که ، معنای جملات وابسته به متن است

معایب

- فاقد قدرت بیاني برای تشریح دقیق محیطی با اشیاي مختلف
  - بر خلاف زبانهای طبیعی

منطق رتبه اول

منطق رتبه اول

- اساس منطق گزاره ای را پذیرفته و بر اساس آن یک منطق بیانی میسازیم
- از ایده های نمایشي زبان طبیعي استفاده کرده ، از عیوب آن اجتناب میکنیم
- زبانهای طبیعی از جهان طبقه بندی زیر را دارند
  - **اشیاء:** افراد ، خانه ، اعداد ، رنگها ، بازیهای فوتبال ، آتش و ...
  - **رابطه ها:**
    - رابطه های یکاني یا خواص مثل قرمز ، گرد ، اول و ...
    - رابطه های چندتایي مثل برادر بودن ، بزرگتر بودن ، بخشی از ، مالکیت و ...
  - **توابع:** پدر بودن ، بهترین دوست ، یکي بیشتر از و ...
- منطق رتبه اول توسط اشیا و رابطه ها ساخته میشود

منطق رتبه اول

انواع منطق

زبان	هستی شناسی (آنچه در جهان هست)	حقیقت شناسی (اعتقادات عامل راجع به حقایق)
منطق گزاره ای	حقایق	درست/نادرست/نامشخص
منطق رتبه اول	حقایق ، اشیا ، رابطه ها	درست/نادرست/نامشخص
منطق موقتی	حقایق ، اشیا ، رابطه ها ، زمان	درست/نادرست/نامشخص
نظریه احتمال	حقایق	درجه ای از اعتقاد متعلق به [0,1]
منطق فازی	حقایق با درجه ای از درستی متعلق به [0,1]	در فاصله معین

منطق رتبه اول

نحو و معنای منطق رتبه اول

- نمادهای ثابت ؛ اشیا را نشان میدهد. مثال: علی ، 2 ، رضا ، ...
- نمادهای محمول ؛ رابطه ها را نشان میدهد. مثال:برادر بودن ، بزرگتر بودن از
- نمادهای تابع ؛ توابع را نشان میدهند. مثال: تابع پای چپ(LeftLeg)
- متغیرها: x , y , a ,b
- روابط منطقی: $\neg, \Rightarrow, \wedge, \vee, \Leftrightarrow$
- تساوی: =
- سورها: $\exists, \forall$

منطق رتبه اول

جملات اتمیک

- هر ترم یک عبارت منطقی است که به شیئ اشاره میکند
  - نمادهای ثابت ترم هستند
  - همیشه استفاده از نماد متمایز برای نامگذاری شیء آسان نیست
    - پای چپ پای پادشاه John ~~LeftLeg(John)~~

ترم = تابع(ترم1 ، ترم2 ، ... ، ترمn) یا ثابت یا متغیر

- جملات اتمیک: ترکیب ترمهای اشیاء و محولهای روابط

جملات اتمیک = محمول(ترم1 ، ترم2 ، ... ، ترمn) یا ترم1=ترم2

- مثال: Married(Father(Richard),Mother(John))

پدر ریچارد با مادر جان ازدواج کرده است

منطق رتبه اول

جملات پیچیده

با ترکیب جملات اتمیک و روابط منطقی میتوان جملات پیچیده تری ساخت

$\neg S,\ S1 \wedge S2,\ S1 \vee S2,\ S1 \Rightarrow S2,\ S1 \Leftrightarrow S2$

مثال:

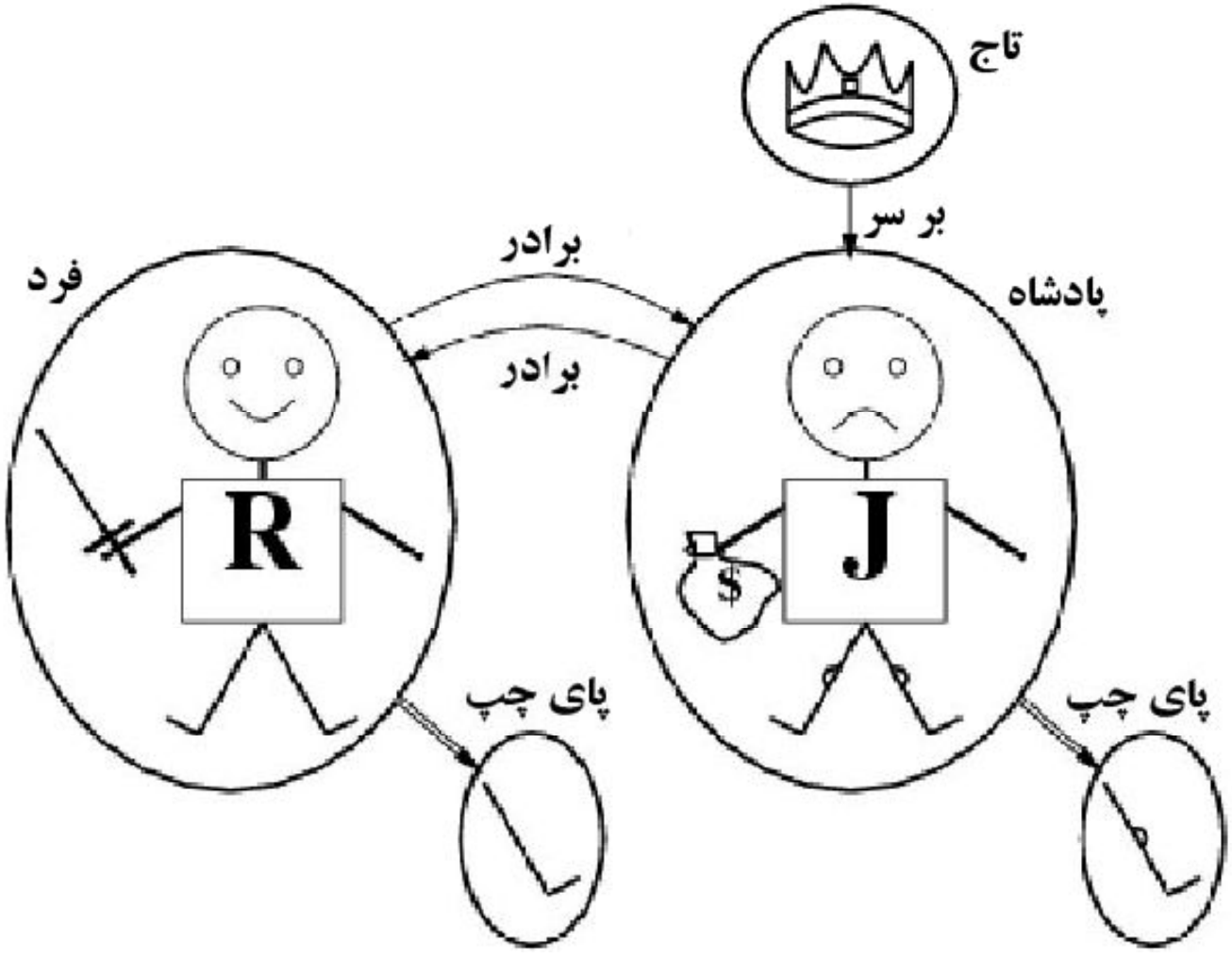
$\neg Brother(LeftLeg(Richard),John)$

$Brother(Richard,John) \wedge Brother(John,Richard)$

$King(Richard) \vee King(John)$

$\neg King(Richard) \Rightarrow King(John)$

منطق رتبه اول



مثال

مدلی با پنج شیء ،
دو رابطه دودویي ،
سه رابطه یکانی و
یک تا یکانی به نام
پای چپ

منطق رتبه اول

سورها

- کمک میکنند تا به جای شمارش اشیا از طریق نام آنها ، خواص کلکسیون اشیا را بیان کرد
- **سور عمومی ؛** $\forall$ "برای همه"
- **سور وجودی ؛** $\exists$ " وجود دارد حداقل..."

منطق رتبه اول

سور عمومی

<جمله> <متغیرها>∀

∀x P که در آن P یک عبارت منطقي است ، بیان میکند که P برای هر شیء x درست است

مثال: ∀x King(x) ⇒ Person(x)

منطق رتبه اول

سور وجودی

∃ <متغیرها> <جمله>

∃ x P که در آن P یک عبارت منطقي است ، بیان میکند که P حداقل برای یک شیء X درست است

مثال: ∃ x Crown(x) ∧ OnHead(x , John)

منطق رتبه اول

خصوصیات سورها

- $\Rightarrow$ رابط طبیعي برای کار با $\forall$ و $\wedge$ رابط طبیعي برای کار با $\exists$ میباشد
- استفاده از $\wedge$ بعنوان رابط اصلی با $\forall$ منجر به حکم قوی میشود
- استفاده از $\Rightarrow$ با $\exists$ منجر به حکم ضعیفي میشود
- $\forall x\ \forall y$ برابر است با $\forall y\ \forall x$ و $\exists x\ \exists y$ برابر است با $\exists y\ \exists x$
- $\exists x\ \forall y$ برابر نیست با $\forall y\ \exists x$
  - $\exists x\ \forall y\ Loves(x,y)$
    - حداقل یک نفر وجود دارد که همه چیز در جهان را دوست دارد
  - $\forall y\ \exists x\ Loves(x,y)$
    - همه در دنیا حداقل یک نفر را دوست دارند

منطق رتبه اول

خصوصیات سورها

- "هر کسی بستنی را دوست دارد" به معنای این است که "هیچ کس وجود ندارد که بستنی را دوست نداشته باشد"
  - $\forall x\ Likes(x , IceCream)$ هم ارز $\neg\exists x\ \neg Likes(x , IceCream)$
- $\forall x\ \neg P$ هم ارز $\neg\exists x\ P$
- $\neg\forall x\ P$ هم ارز $\exists x\ \neg P$
- $\forall x\ P$ هم ارز $\neg\exists x\ \neg P$
- $\exists x\ P$ هم ارز $\neg\forall x\ \neg P$

منطق رتبه اول

تساوی

- با استفاده از = دو ترم به یک شیء اشاره میکنند
- برای تعیین درستی جمله تساوی باید دید که آیا ارجاع ها به دو ترم ، اشیای یکسانی اند یا خیر
- مثال: ریچارد حداقل دو برادر دارد

$$\exists x,y\ Brother(x,Richard) \wedge Brother(y,Richard) \wedge \neg(x=y)$$

منطق رتبه اول

ادعاها و تقاضاها

- جملات از طریق TELL به پایگاه دانش اضافه میشوند
  - این جملات را **ادعا** گویند
    - TELL (*KB* , King(John))
    - TELL (*KB* , ∀x King(x) => Person(x))
- با استفاده از ASK تقاضاهايي را از پليگاه دانش انجام ميدهيم
  - این پرسشها ، **تقاضا** یا **هدف** نام دارد
    - ASK (*KB* , Person(John))
    - ASK(*KB* , ∃x Person(x))
- ليست جانشيني يا انقياد
  - ليستي از جانشينيها در صورت وجود بيش از يک پاسخ

منطق رتبه اول

دامنه خویشاوندی

مادر هر فرد والد مؤنث آن فرد است

$\forall m,c\ Mother(c) = m \Leftrightarrow Femail(m) \wedge Parent(m,c)$

شوهر هر فرد ، همسر مذکر آن فرد است

$\forall w,h\ Husband(h,w) \Leftrightarrow Male(h) \wedge Spouse(h,w)$

مذکر و مؤنث بودن طبقه های متمایزی هستند

$\forall x,\ Male(x) \Leftrightarrow \neg Female(x)$

والد و فرزند ، رابطه های معکوس هستند

$\forall p,c\ Parent(p,c) \Leftrightarrow Child(c,p)$

پدر بزرگ یا مادربزرگ والدینِ والدین هر فرد است

$\forall g,c\ Grandparent(g,c) \Leftrightarrow \exists p\ Parent(g,p) \wedge Parent(p,c)$

منطق رتبه اول

اعداد و مجموعه ها

- $\forall s\ Set(s) \Leftrightarrow (s = \{\}) \vee (\exists x,s2\ Set(s2) \wedge s = \{x|s2\})$
- $\neg\exists x,s\ \{x|s\} = \{\}$
- $\forall x,s\ x \in s \Leftrightarrow s = \{x|s\}$
- $\forall x,s\ x \in s \Leftrightarrow [\exists y,s2\} (s = \{y|s2\} \wedge (x = y \vee x \in s2))]$

منطق رتبه اول

مهندسي دانش

- فرایند کلی ساخت پایگاه دانش که شامل مراحل ذیل میباشد:
  - مشخص کردن کار
  - مونتاژ دانش مربوطه
  - تصمیم گیری در مورد واژه نامه محمولها ، توابع و وراثت
  - کدگزاری دانش کلی در مورد دامنه
  - کد گزاری توصیف نمونه مسئله خاص
  - اِعمال تقاضاها به رویه استنتاج و دریافت پاسخ
  - اشکال زدایي پایگاه دانش