

هوش مصنوعی



نام مرجع :

Artificial Intelligence A Modern Approach

نویسنده :

استوارت راسل ، پیتر نورویگ

تهیه کننده :

مجتبی پورمحقق



هوش مصنوعي

فصل اول

مقدمه



فهرست

هوش مصنوعي چیست؟

مباني هوش مصنوعي

تاريخچه هوش مصنوعي



هوش مصنوعي چیست؟

مانند انسان فکر کردن

عقلانه فکر کردن

مانند انسان عمل کردن

عقلانه عمل کردن

Acting humanly

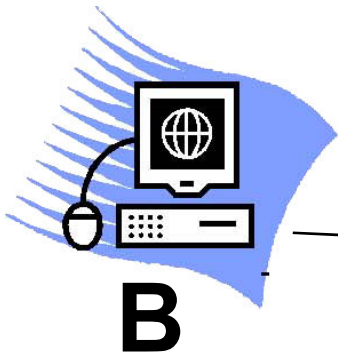
مانند انسان عمل کردن

✓ هنر ساخت ماشینهایی که کارهایی را انجام میدهند که آن کارها توسط انسان با فکر کردن انجام میشوند.

✓ مطالعه برای ساخت کامپیوترها برای انجام کارهایی که فعلاً انسان آنها را بهتر انجام میدهد.



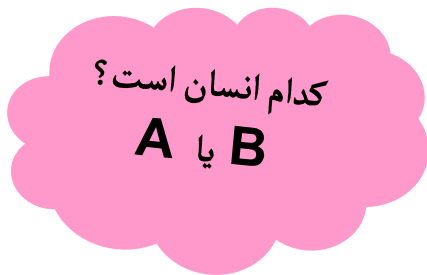
تست تورینگ



B



A



کدام انسان است؟
A یا **B**



Thinking humanly

مانند انسان فکر کردن

✓ تلاش جدید و هیجان انگیز برای ساخت ماشین هایی متفکر و با
حس کامل

✓ خودکارسازی فعالیت های مرتبط با تفکر انسان ، فعالیت هایی مثل
تصمیم گیری ، حل مسئله ، یادگیری



Think rationally

عقلانه فکر کردن

✓ مطالعه توانایی های ذهنی از طریق مدل های محاسباتی (منطق
گرایي)

✓ مطالعه محاسباتی که منجر به درک و استدلال می شود.

Act rationally

عقلانه عمل کردن

طوري عمل کند که بهترین نتیجه را ارائه دهد

هوش محاسباتي ، مطالعه طراحي عامل هاي هوشمند است ✓



مباني هوش مصنوعي

روان شناسي: تطبيق ، اثر طبيعي

ادراك و تاثير آن بر محيط

فلسفه: منطق ، استدلال ، ناشي

شدن تفکر از مغز فيزيکي ، مباني

يادگيري ، زبان و عقلانيت

رياضيات: نمايش رسمي الگوريتمها ،

محاسبات ، تصميم پذيري و تصميم

ناپذيري ، احتمال

زبان شناسي: علم

ارائه ، گرامر



مباني هوش مصنوعي

علوم عصبي: نحوه پردازش
اطلاعات توسط مغز

نظريه کنترل و سيبرنتيك: تحت کنترل
در آوردن محصولات مصنوعي، ثبات و پايداري، طراحي
عامل بهينه

مهندسي کامپيوتر: ساخت
کامپيوترهاي سريع

اقتصاد: نظريه تصميمهاي عقلايي،
نظريه بازي



تاریخچه هوش مصنوعی

1943 ، مک کولوج و والتر پیتز: ارایه مدل نرون مصنوعی بیتی (دو حالتی) قابل یادگیری به منظور محاسبه هر تابع قابل محاسبه.

1950 ، آلن تورینگ اولین بار دید کاملی از هوش مصنوعی را تحت عنوان ”محاسبات ماشینی و هوشمند“ ارایه نمود.

1951 ، هینسکی و ادموندز اولین کامپیوتر شبکه عصبی را طراحی کردند.

1952 ، آرتور سامویل: برنامه ای ساخت که یاد میگرفت بهتر از نویسندگانش بازی کند ؛ در نتیجه این تصور را که ”کامپیوتر فقط کاری را انجام میدهد که به آن گفته شود“ نقض کرد.

مقدمه (تاریخچه هوش مصنوعی)



- 1956، نشست کارگروهی دورتموند: انتخاب نام هوش مصنوعی
- 1959، هربرت جلونتتر: برنامه (GTP) را ساخت که قضایا را با اصل موضوعات مشخص ثابت می کرد.
- 1958، جان مک کارتی: تعریف زبان لیسپ که بهترین زبان هوش مصنوعی شد.
- 1958-1973، جیمز اسلاگل: برنامه حل مسایل انتگرالگیری فرم بسته
 - تام ایوانز: برنامه حل مشابهت های هندسی
 - دانیل بابروز: برنامه حل مسایل جبری
 - دیوید هافمن: پروژه محدوده بینایی روبات در جهان بلوکها
 - دیوید والتز: سیستم بینایی و انتشار محدود
 - پاتریک ونیستون: نظریه یادگیری



(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



(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 شمای نمایش دانش

▪ استفاده از منطق در پرولوگ ، استفاده از ایده مینسکی یعنی قابها و ...



1980 تا کنون: تبدیل هوش مصنوعی به یک صنعت

1986 تا کنون: برگشت به شبکه های عصبی

1987 تا کنون: هوش مصنوعی به علم تبدیل میشود

1995 تا کنون: ظهور عاملهای هوشمند



هوش مصنوعي

فصل دوم

عواملهاي هوشمند



فهرست

↪ عامل

↪ خواص محیط‌های وظیفه

↪ برنامه‌های 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.

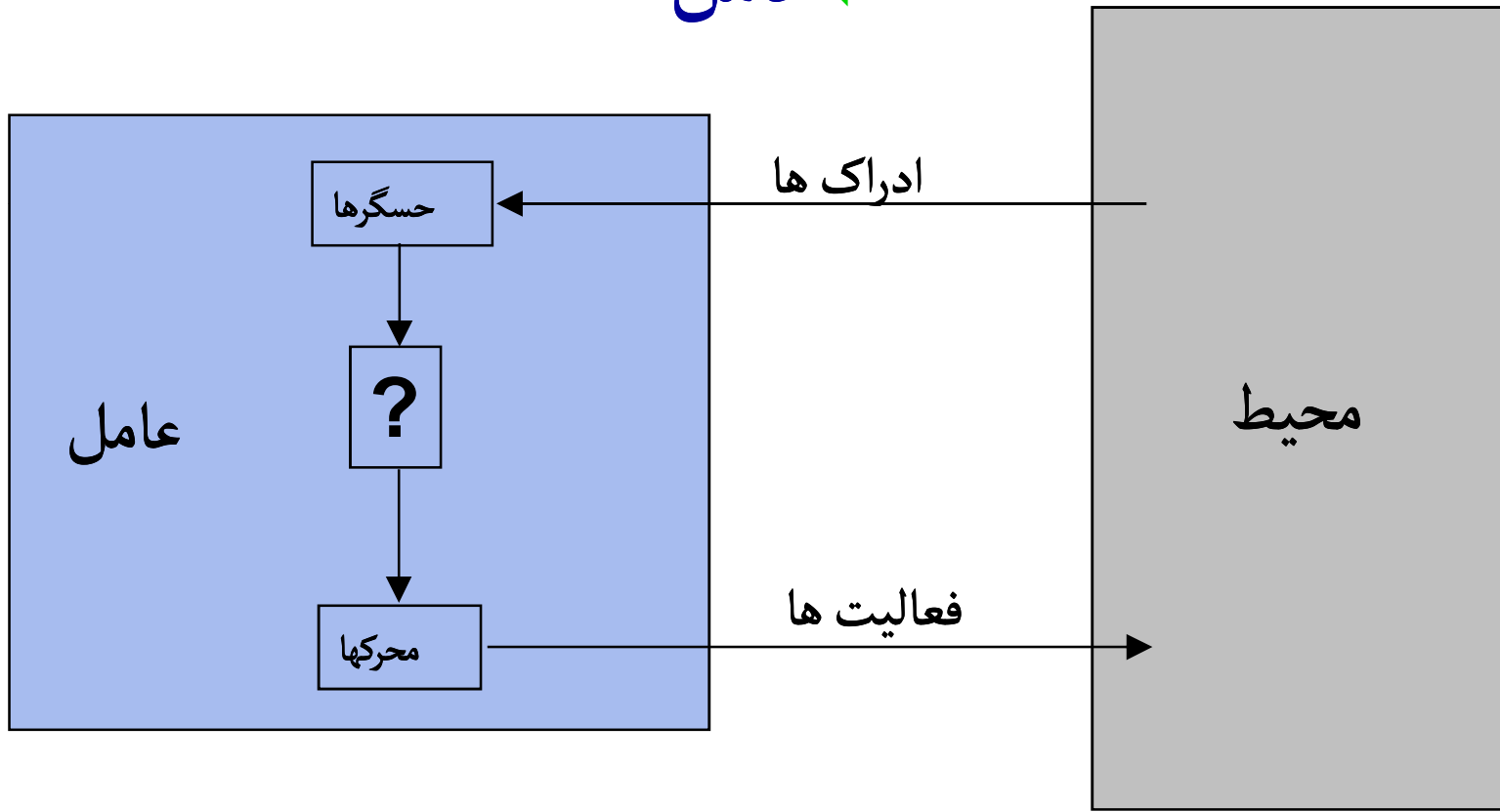
$$f : P^* \rightarrow A$$

دنباله ادراک : تابع عامل $\xrightarrow{\text{فعالیت}}$

عوامل‌های هوشمند



↔ عامل



عوامل‌های هوشمند



معیارهای کارایی

معیار کارایی، معیاری برای موفقیت رفتار عامل است.

- بر اساس خواسته‌های فرد در محیط انتخاب میشود

رفتار عقلایی

معیار کارایی که ملاک‌های موفقیت را تعریف میکند

دانش قبلی عامل نسبت به محیط

فعالیت‌هایی که عامل میتواند انجام دهد

دنباله ادراک عامل در این زمان

عاملهای هوشمند



↪ عامل عالم (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 کارایی اش را حداکثر میکند

• جمع آوری اطلاعات ، اکتشاف ، یادگیری

↪ عامل خود مختار

نقص دانش قبلی خود را میتواند جبران کند



↪ کاملاً قابل مشاهده درمقابل قابلیت مشاهده جزئی

↪ قطعی درمقابل غیر قطعی

↪ راهبردی

↪ رویدادی درمقابل ترتیبی

↪ ایستا درمقابل پویا

↪ گسسته درمقابل پیوسته

↪ تک عاملی درمقابل چند عاملی

↪ چند عاملی رقابتی درمقابل چندعاملی همیاری

خواص

محیط‌های 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 مدل گرا

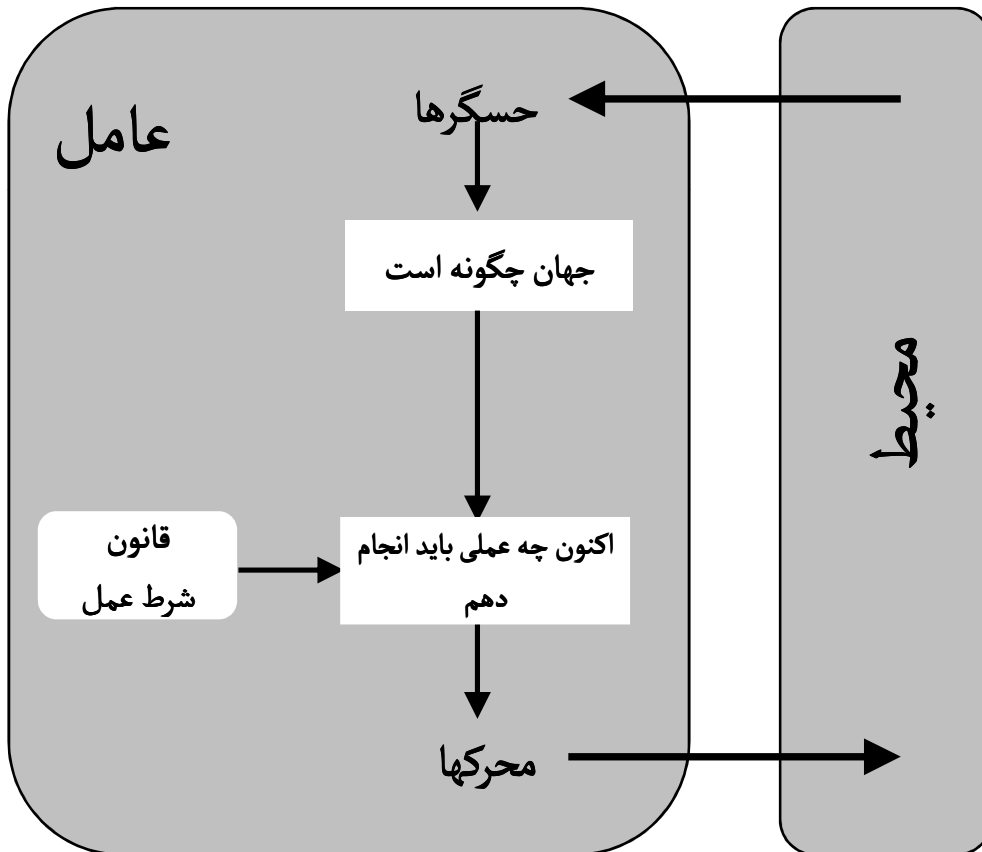
عاملهای واکنشی ساده

عاملهای سودمند

عاملهای هدف گرا

عاملهای هوشمند

عاملهای واکنشی ساده



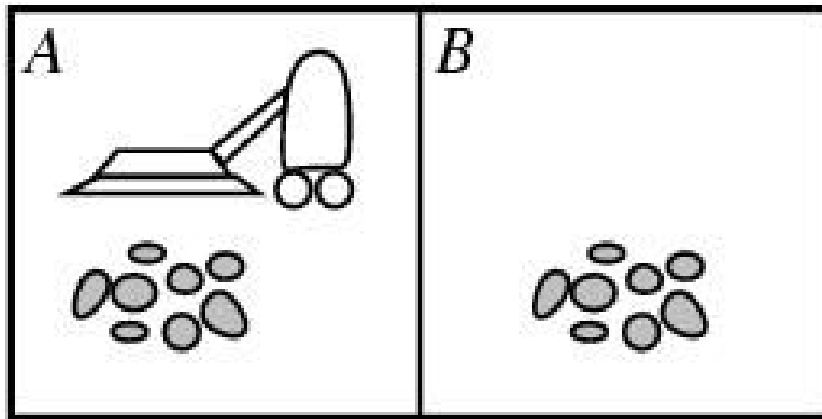
این عاملها فعالیت را بر اساس درک فعلی و بدون در نظر گرفتن سابقه ادراک، انتخاب میکند

به خاطر حذف سابقه ادراک برنامه عامل در مقایسه با جدول آن بسیار کوچک است

انتخاب فعالیت بر اساس یکسری قوانین موقعیت شرطی انجام میشود

عاملهای هوشمند

مثالی از عامل واکنشی ساده در دنیای جاروبرقی



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 می یابد

انتخاب فعالیت بر اساس موقعیت

function REFLEX-VACUUM-AGENT (*location, status*)
return an action

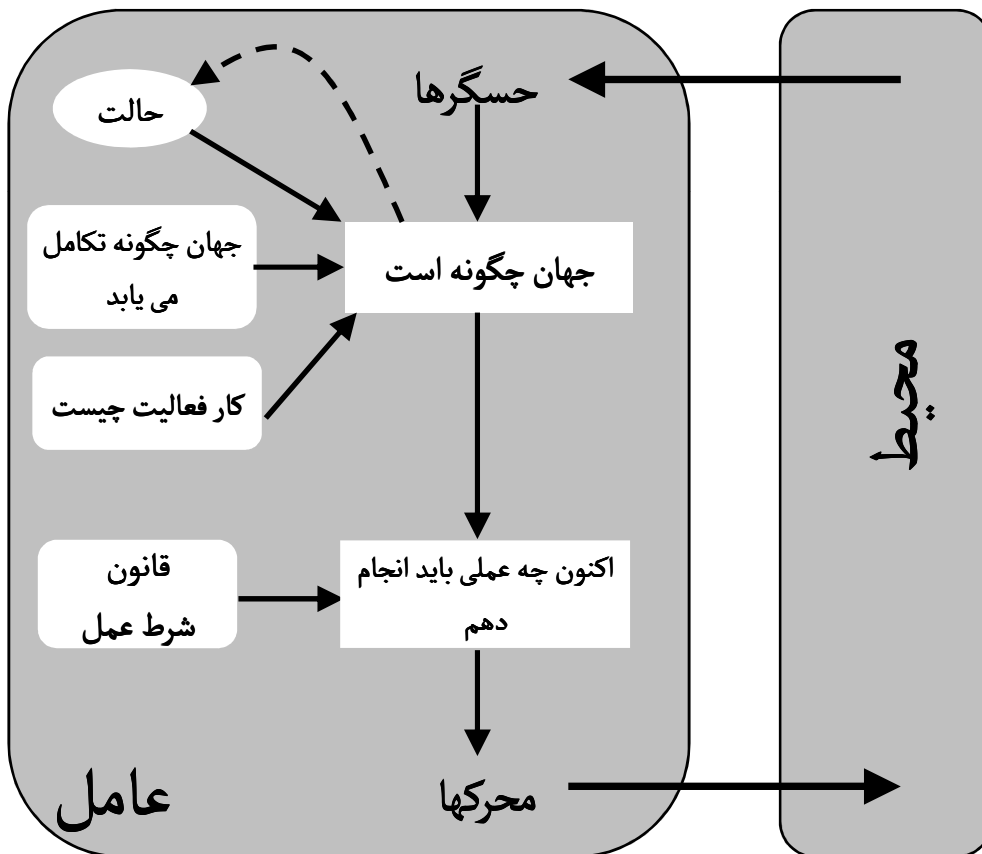
if *status* == *Dirty* then return *Suck*
else if *location* == *A* then return *Right*
else if *location* == *B* then return *Left*

شرطی:

If dirty then suck

عوامل‌های هوشمند

عوامل‌های واکنشی مدل گرا



استفاده از دانش "چگونگی عملکرد جهان" که مدل نام دارد

عامل بخشی از دنیایی را که فعلا میبیند ردیابی میکند

عامل باید حالت داخلی را ذخیره کند که به سابقه ادراک بستگی دارد

در هر وضعیت، عامل میتواند توصیف جدیدی از جهان را کسب کند



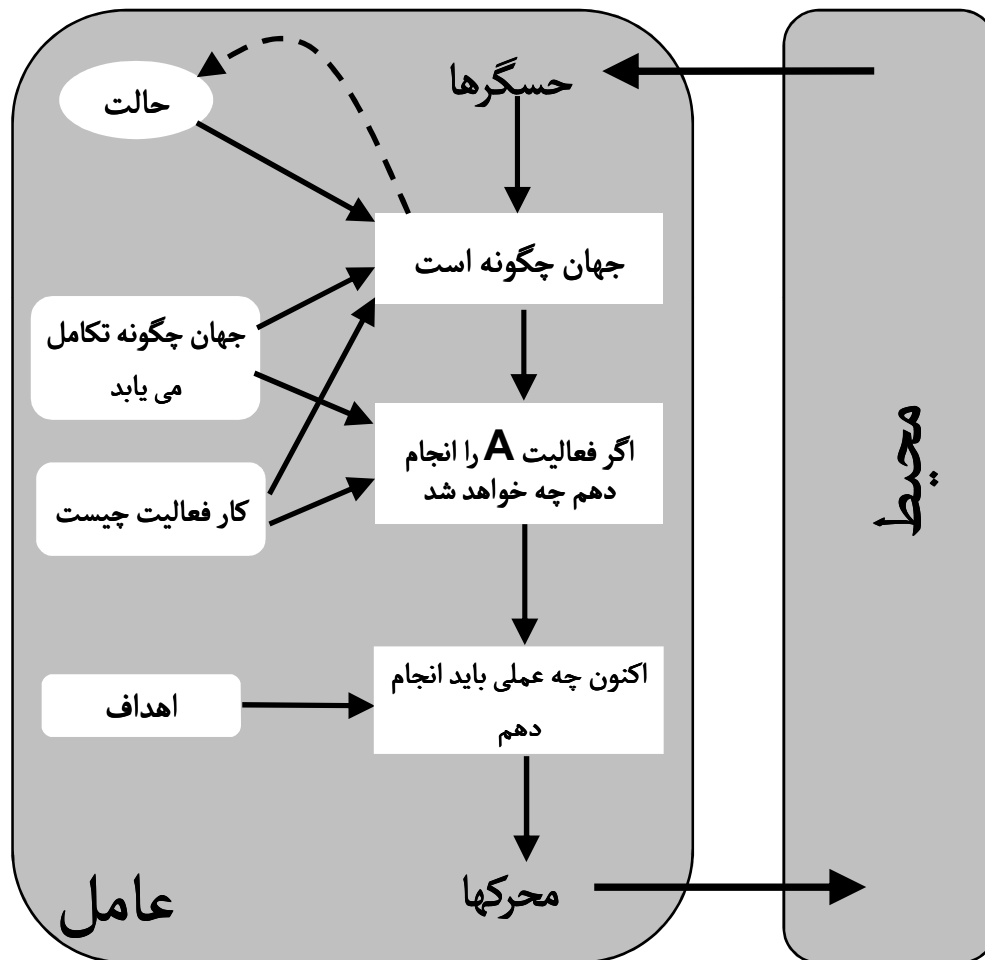
عوامل‌های هدف‌گرا

این عامل علاوه بر توصیف حالت فعلی، برای انتخاب موقعیت مطلوب نیازمند اطلاعات هدف نیز میباشد

جست و جو و برنامه‌ریزی، دنباله‌ای از فعالیتها را برای رسیدن عامل به هدف، پیدا میکند

این نوع تصمیم‌گیری همواره آینده را در نظر دارد و با قوانین شرط عمل تفاوت دارد

این نوع عامل کارایی چندانی ندارد، اما قابلیت انعطاف بیشتری دارد



عوامل‌های هوشمند



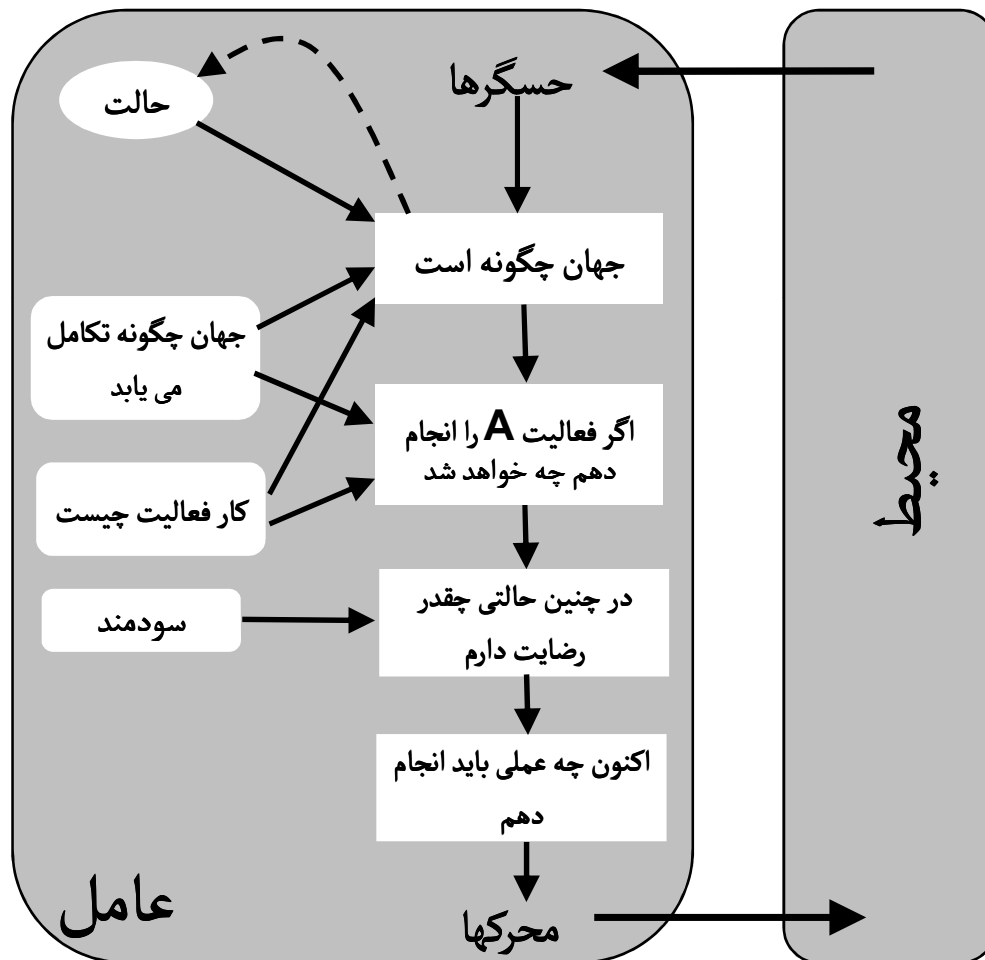
عوامل‌های سودمند

این عامل برای اهداف مشخص، راه‌های مختلفی دارد، که راه حل بهتر برای عامل سودمندتر است.

تابع سودمندی، حالت یا دنباله‌ای از حالتها را به یک عدد حقیقی نگاشت میکند که درجه رضایت را توصیف میکند.

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





عوامل‌های یادگیرنده

عنصر یادگیرنده مسئول ایجاد بهبودها

عنصر کارایی مسئول انتخاب فعالیت‌های

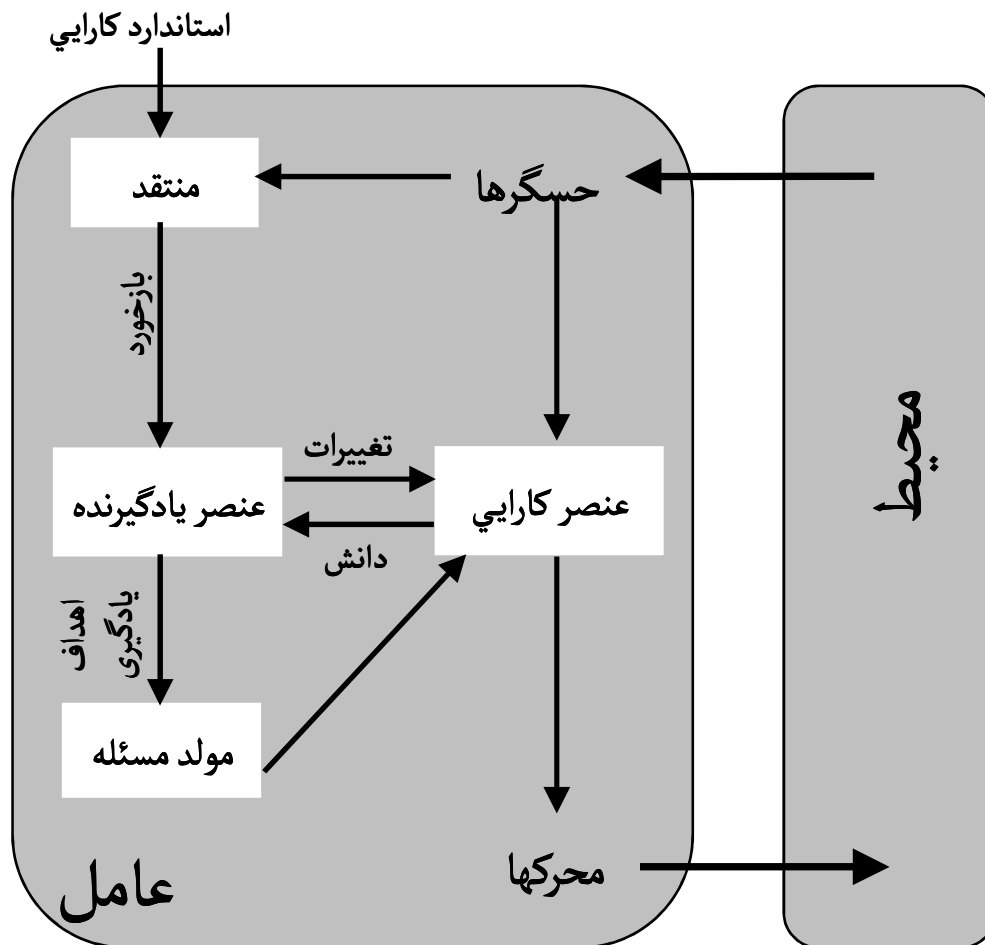
خارجی

منتقد مشخص میکند که یادگیرنده با توجه به

استانداردهای کارایی چگونه عمل میکند

مولد مسئله مسئول پیشنهاد فعالیت‌هایی است

که منجر به تجربیات آموزنده جدیدی میشود





هوش مصنوعي

فصل سوم

حل مسئله با جستجو



فهرست

↩ عواملی حل مسئله

↩ مسئله

↩ اندازه گیری کارایی حل مسئله

↩ جستجوی ناآگاهانه

↩ اجتناب از حالت‌های تکراری

↩ جستجو با اطلاعات ناقص

حل مسئله با جستجو



عوامل حل مسئله

چهار گام اساسی برای حل مسائل

فرموله کردن هدف: وضعیتهای مطلوب نهایی کدامند؟

فرموله کردن مسئله: چه فعالیتها و وضعیتهایی برای رسیدن به هدف موجود است؟

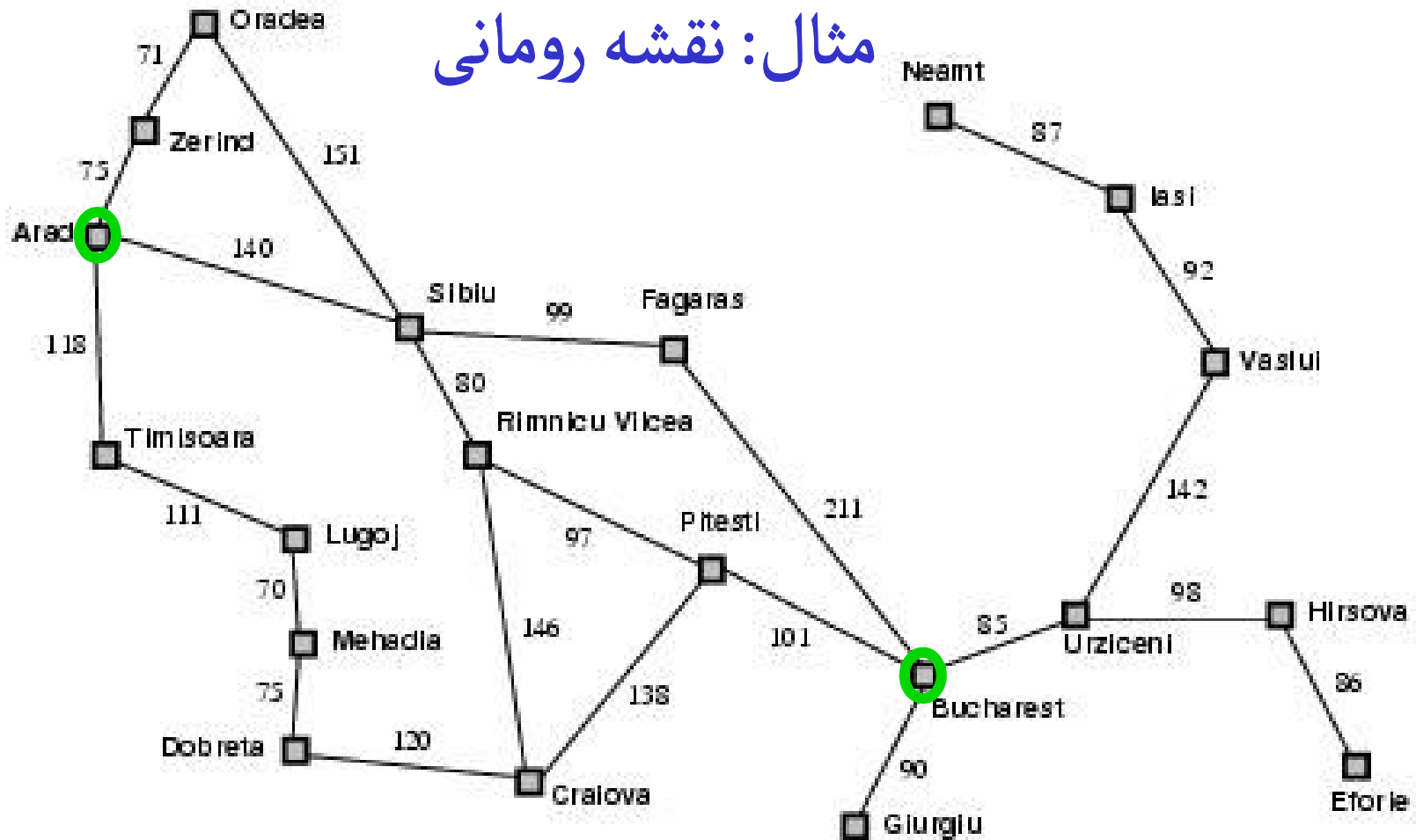
جستجو: انتخاب بهترین دنباله از فعالیتهایی که منجر به حالاتی با مقدار شناخته شده میشود.

اجرا: وقتی دنباله فعالیت مطلوب پیدا شد، فعالیتهای پیشنهادی آن میتواند 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 ، سیبویو ، 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 $n(\text{Arad})$

تابع جانشین: توصیفی از فعالیتهای ممکن که برای عامل مهیا است.

در مثال رومانی: $S(\text{Arad}) = \{ \text{Zerind}, \text{Sibui}, \text{Timisoara} \}$

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 فعالیتهای آنها 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.

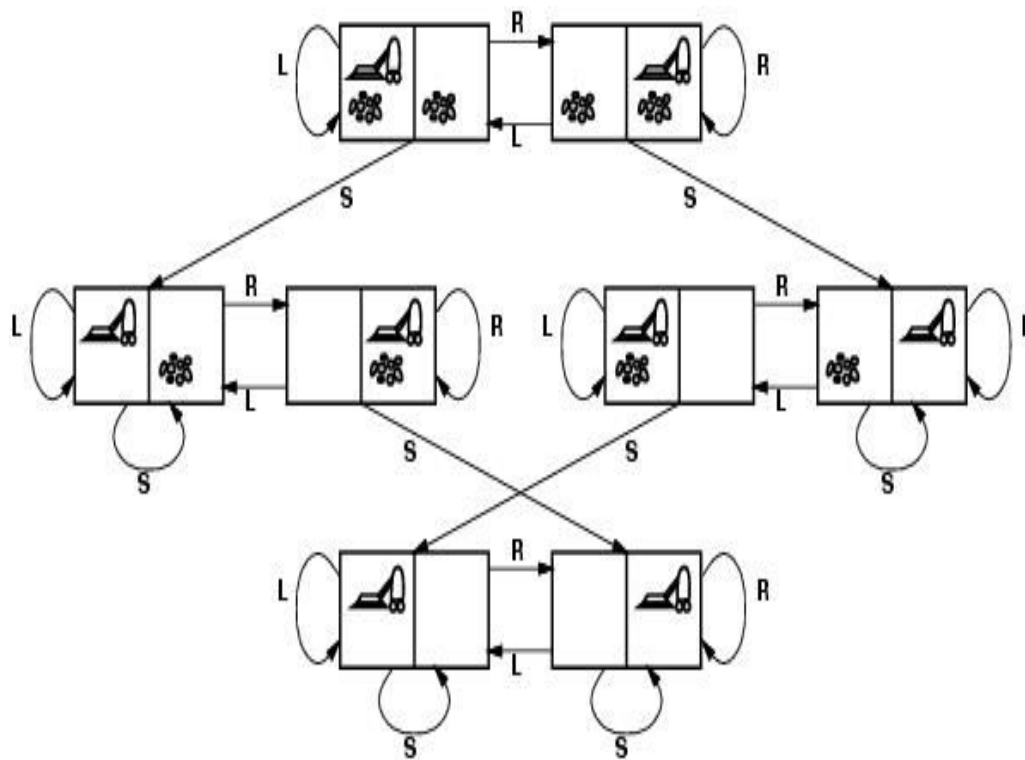
در مثال رومانی: طول مسیر بین شهرها بر حسب کیلومتر

راه حل مسئله مسیری از حالت اولیه به حالت هدف است

راه حل بهینه کمترین هزینه مسیر را دارد



مثال: دنیای جارو برقی



حالتها:

حالت اولیه:

تابع جانشین:

آزمون هدف:

هزینه مسیر:

حل مسئله با جستجو

مثال: دنیای جارو برقی

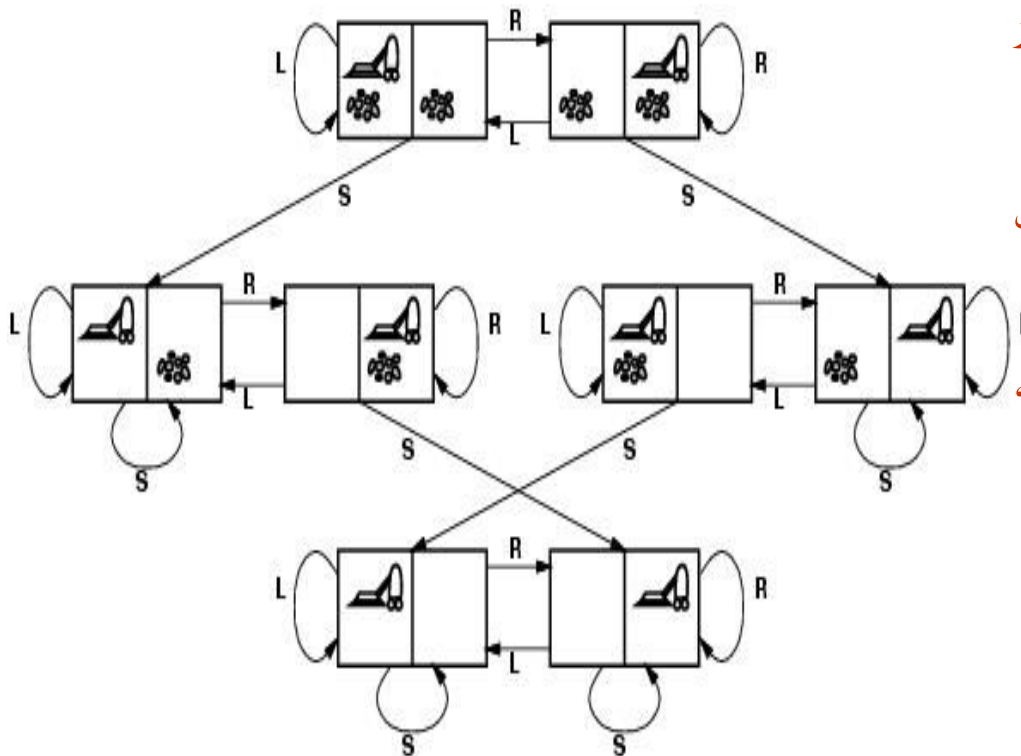
حالتها: دو مکان که هر یک ممکن است کثیف یا تمیز باشند. لذا $2 \times 2 = 8$ حالت در این جهان وجود دارد

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

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

آزمون هدف: تمیزی تمام مربعها

هزینه مسیر: تعداد مراحل در مسیر



حل مسئله با جستجو



مثال: معمای 8

7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

Goal State

حالتها:

حالت اولیه:

تابع جانشین:

آزمون هدف:

هزینه مسیر:

حل مسئله با جستجو



مثال: معمای 8

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

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

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

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

هزینه مسیر: برابر با تعداد مراحل در مسیر

7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

Goal State

حل مسئله با جستجو



مثال: مسئله 8 وزیر

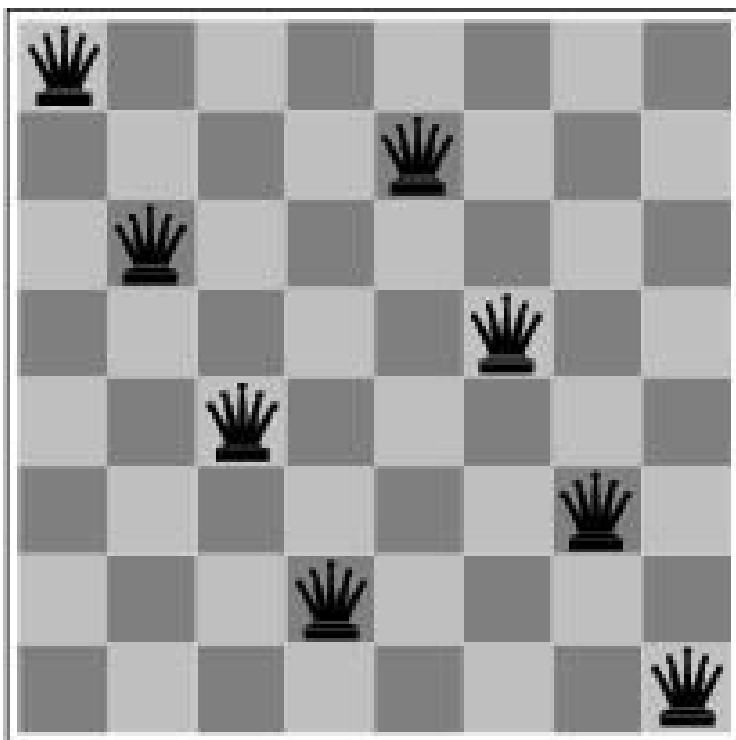
فرمول بندی افزایشی

حالتها:

حالت اولیه:

تابع جانشین:

آزمون هدف:



حل مسئله با جستجو

مثال: مسئله 8 وزیر

فرمول بندی افزایشی

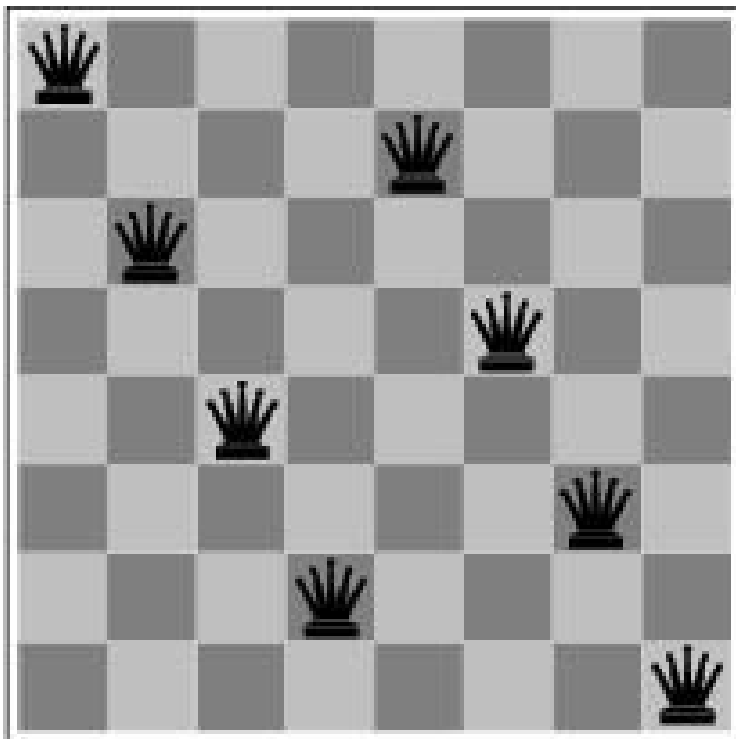
حالتها: هر ترتیبی از 0 تا 8 وزیر در صفحه، یک حالت است

حالت اولیه: هیچ وزیری در صفحه نیست

تابع جانشین: وزیری را به خانه خالی اضافه میکند

آزمون هدف: 8 وزیر 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

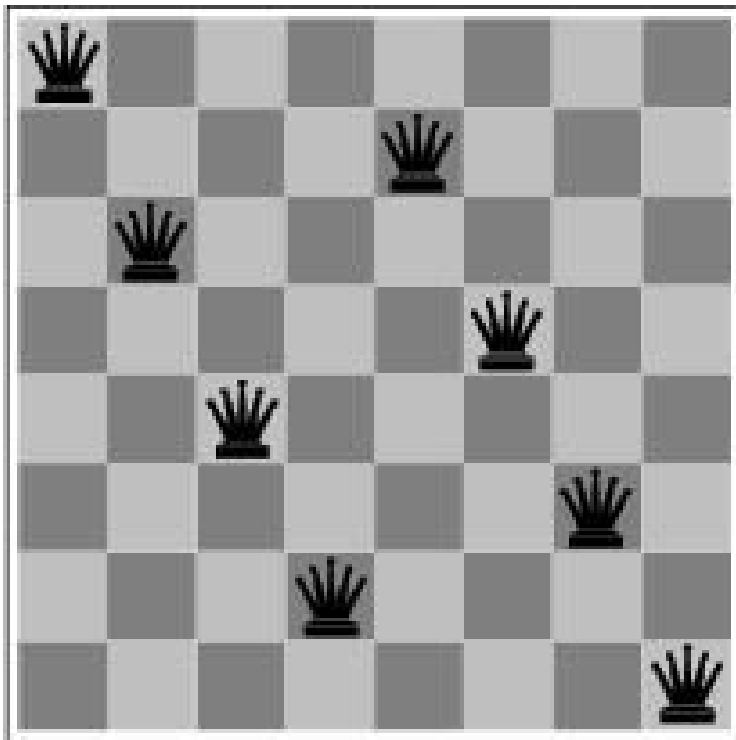
فرمول بندی حالت کامل

حالتها:

حالت اولیه:

تابع جانشین:

آزمون هدف:



حل مسئله با جستجو



مثال: مسئله 8 وزیر

فرمول بندی حالت کامل

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

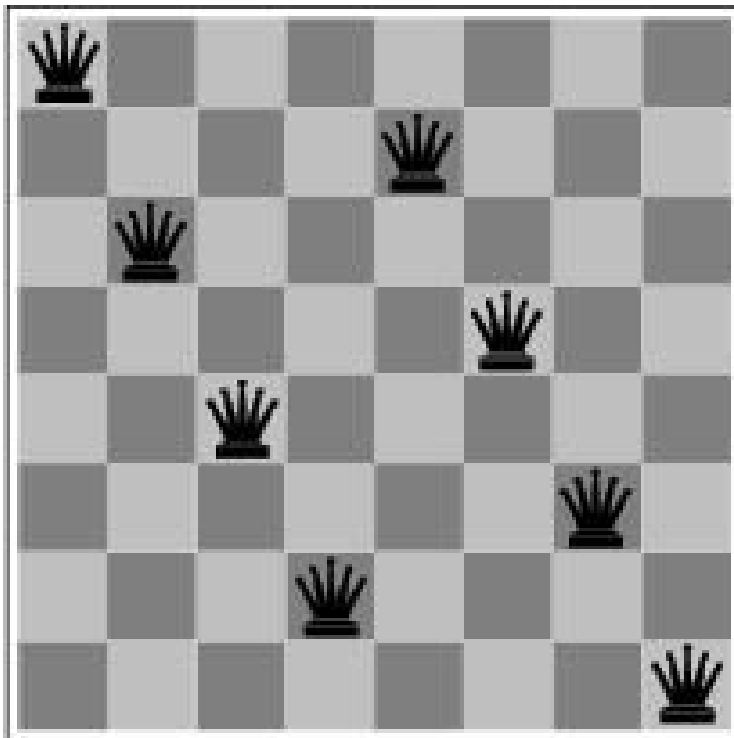
حالت اولیه: با 8 وزیر در صفحه شروع میشود

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

آزمون هدف: 8 وزیر 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 مسئلہ با جستجو



اندازه گیری کارایی حل مسئله

↩ کامل بودن:

↩ بهینگی:

↩ پیچیدگی زمانی:

↩ پیچیدگی فضا:



اندازه گیری کارایی حل مسئله

↩️ **کامل بودن:** آیا الگوریتم تضمین میکند که در صورت وجود راه حل ، آن را بیابد؟

↩️ **بهینگی:** آیا این راهبرد ، راه حل بهینه ای را ارائه میکند.

↩️ **پیچیدگی 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 عمقی
- جست و جوی عمیق کننده تکراری
- جست و جوی هزینه یکنواخت
- جست و جوی عمقی محدود
- جست و جوی دو طرفه

حل مسئله با جستجو



جستجوی عرضی

A

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 وقتی که فعالیتها هزینه یکسانی دارند)

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

پیچیدگی زمانی:

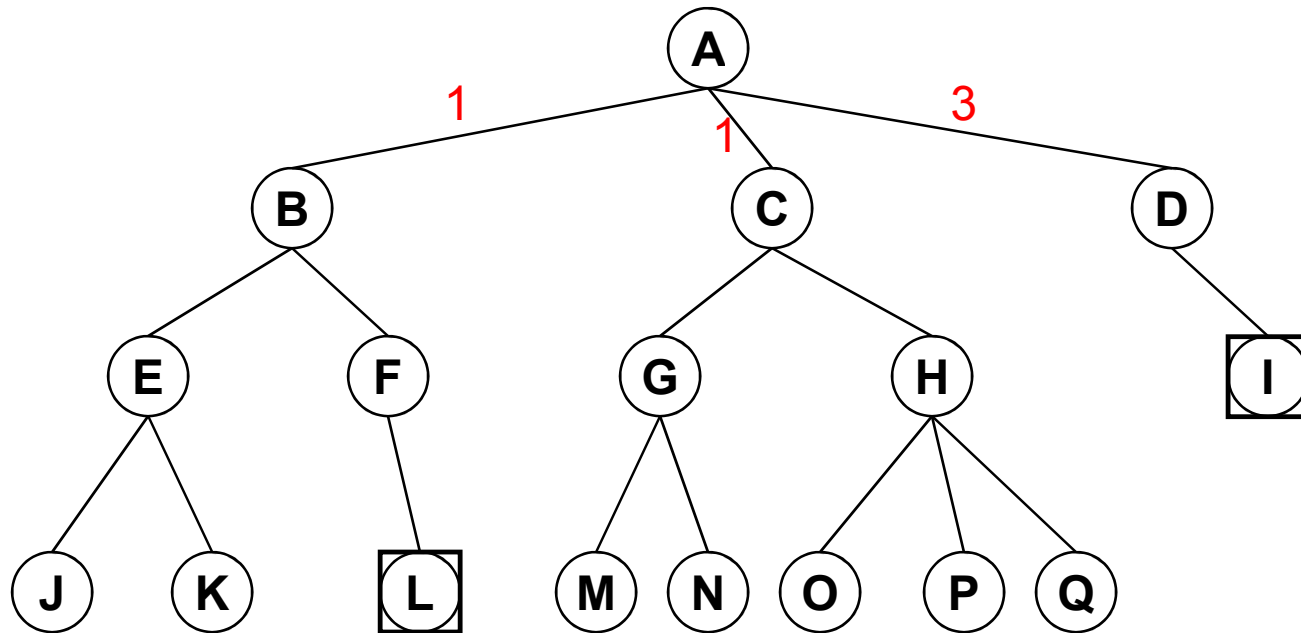
$$O(b^{d+1})$$

پیچیدگی فضا:

حل مسئله با جستجو

جستجوی هزینه یکنواخت

این جستجو n گره 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 می یابد)

پهینگی: بله

هزینه هر مرحله بزرگتر یا مساوی ϵ باشد

$$O(b^{[C^*/\epsilon]})$$

پیچیدگی زمانی:

$$O(b^{[C^*/\epsilon]})$$

پیچیدگی فضا:

حل مسئله با جستجو



جستجوی عمقی

A

حل مسئله با جستجو



جستجوی عمقی

کامل کلودن بوخین:

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

نمی یابد.

بهینگی نگیز

$$O(b^m)$$

پیچیدگی زمانی:

$$O(bm)$$

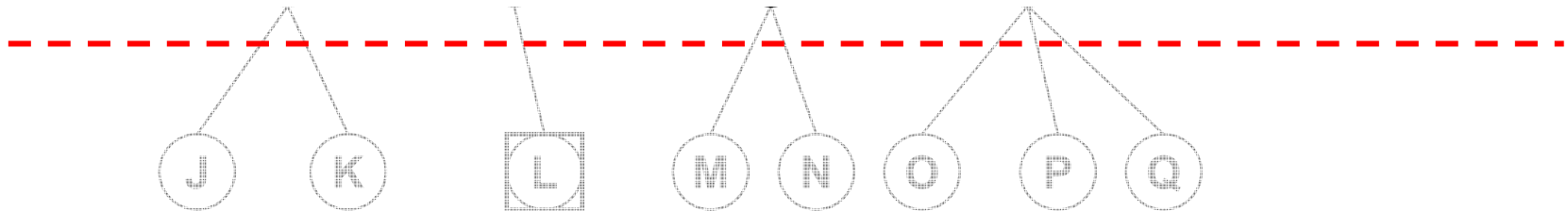
پیچیدگی فضا:

حل مسئله با جستجو

جستجوی عمقی محدود

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

(A)



حل مسئله با جستجو



جستجوی عمقی محدود

کامل کلودن بوخین:

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

بهینه‌نگین‌گیز

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

$$O(b^L)$$

پیچیدگی زمانی:

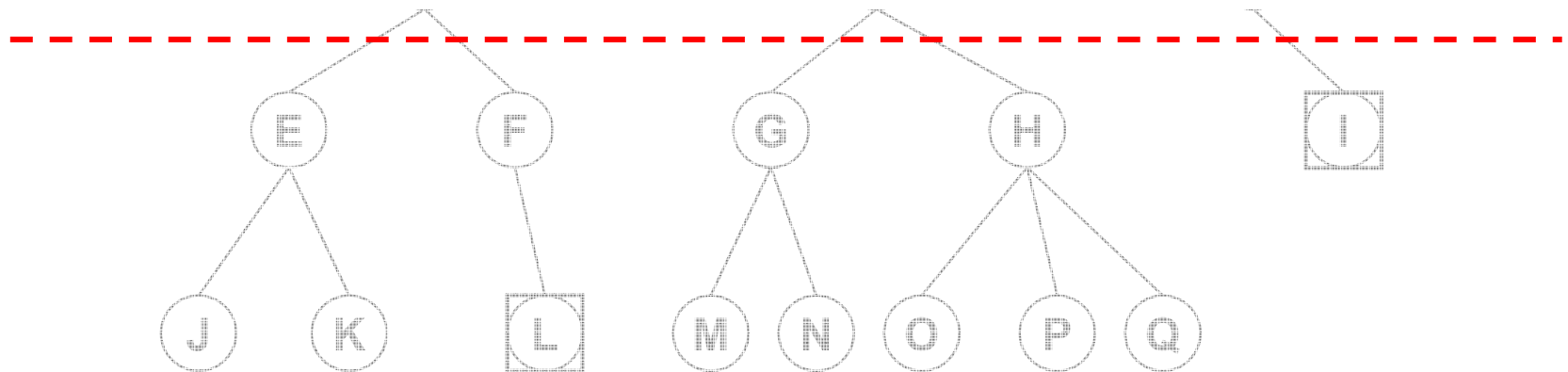
$$O(bL)$$

پیچیدگی فضا:



جستجوی عمیق کننده تکراری

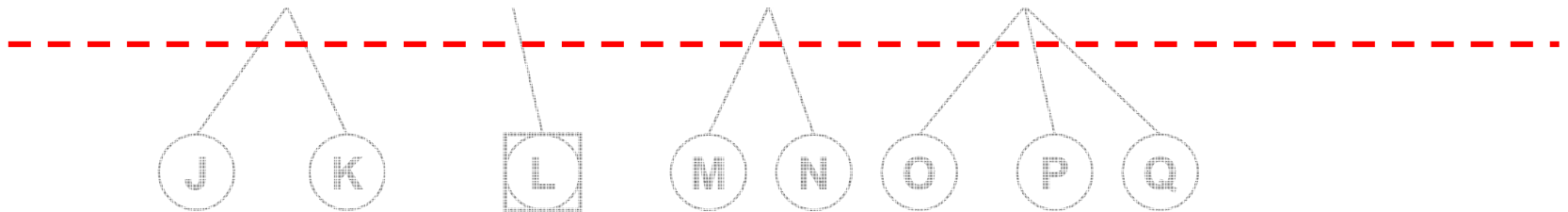
(A)





جستجوی عمیق کننده تکراری

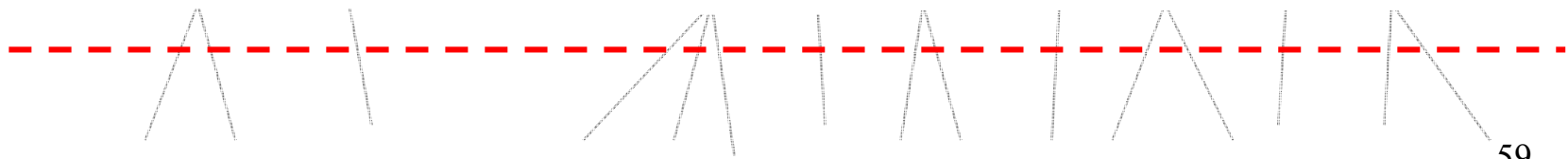
A





جستجوی عمیق کننده تکراری

A





جستجوی عمیق کننده تکراری

کامل کلودن بودن:

در صورتی که فاکتور انشعاب محدود باشد

بهینه‌نگی:

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

$$O(b^d)$$

پیچیدگی زمانی:

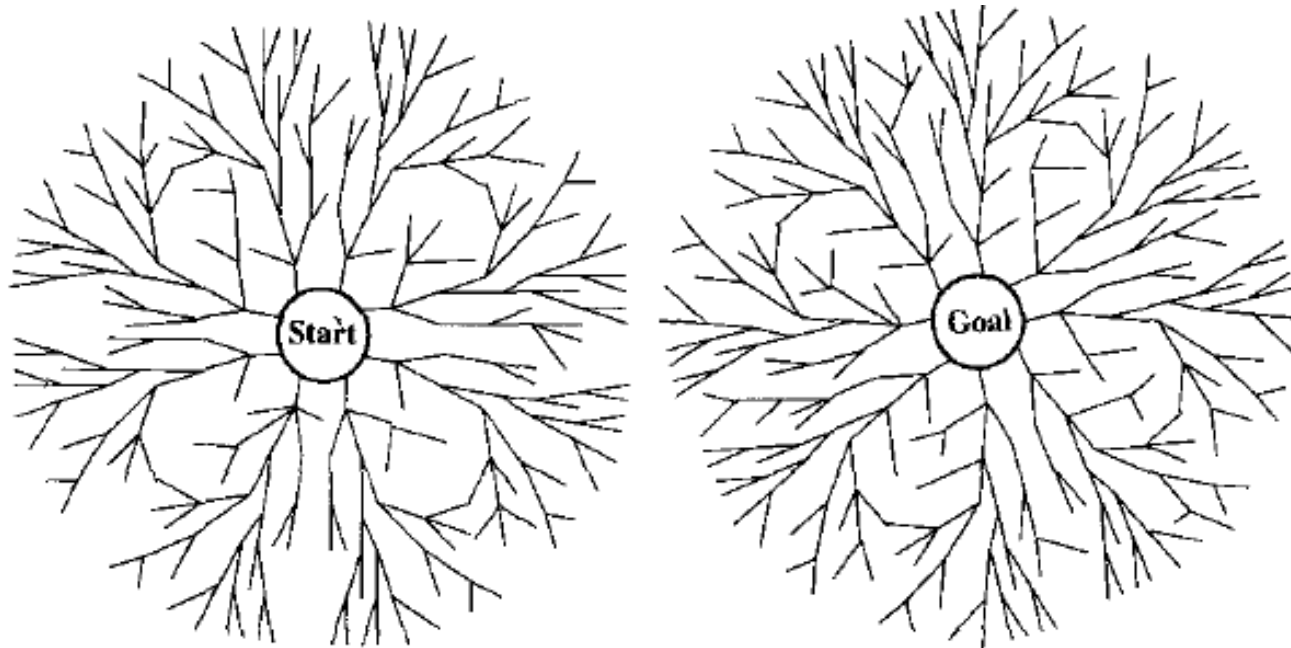
$$O(bd)$$

پیچیدگی فضا:



جستجوی دو طرفه

انجام دو جست و جوی همزمان ، یکی 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 کردن بودن:

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

بهینه‌ترین گره:

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

$$O(b^{d/2})$$

پیچیدگی زمانی:

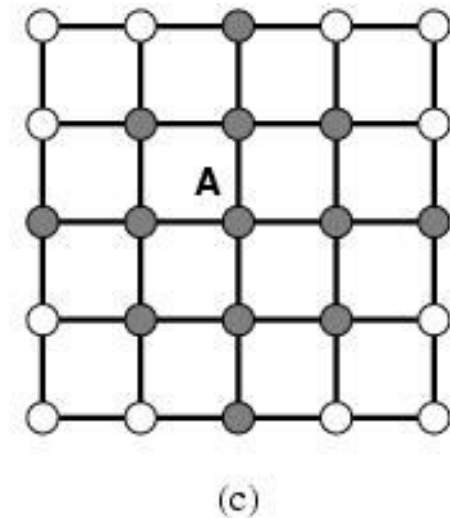
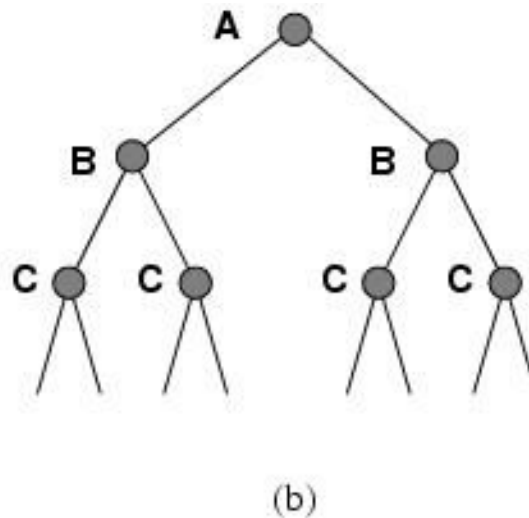
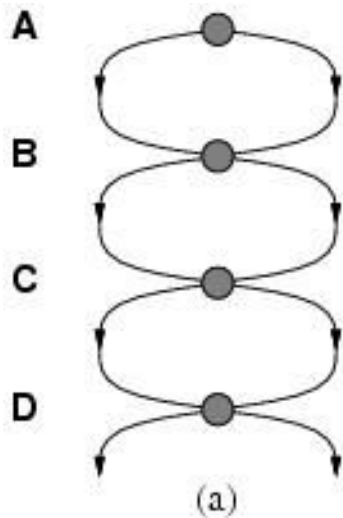
$$O(b^{d/2})$$

پیچیدگی فضا:



اجتناب از حالت‌های تکراری

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



حل مسئله با جستجو

جستجو با اطلاعات ناقص

🔍 **مسئله های فاقد حسگر:** اگر عامل فاقد حسگر باشد ، میتواند در یکی از چند حالت اولیه باشد و هر فعالیت میتواند آن را به یکی از چند حالت جانشین ببرد

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

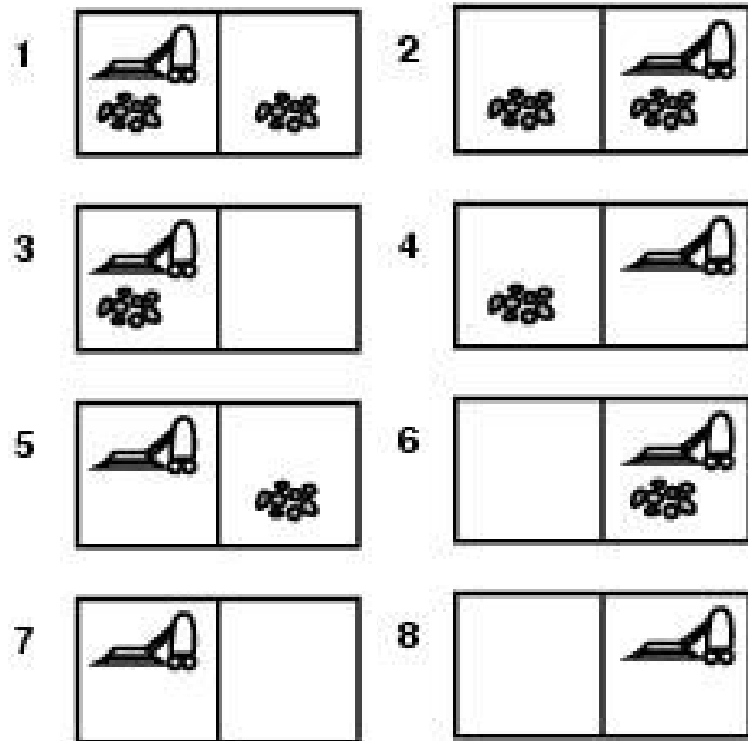
🔍 **مسائل خصمانه:** اگر عدم قطعیت در اثر فعالیتهای عامل دیگری بوجود آید ، مسئله را خصمانه گویند

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

حل مسئله با جستجو



مثال: دنیای جاروبرقی فاقد حسگر



↪ عامل جارو تمام اثرات فعالیتهايش را ميداند اما فاقد حسگر است.

↪ حالت اوليه آن يکي از اعضاي مجموعه $\{1, 2, 3, 4, 5, 6, 7, 8\}$ مي باشد

↪ فعالیت (Right) $\leftarrow \{2, 4, 6, 8\}$

↪ فعالیت (Right,Suck) $\leftarrow \{4, 8\}$

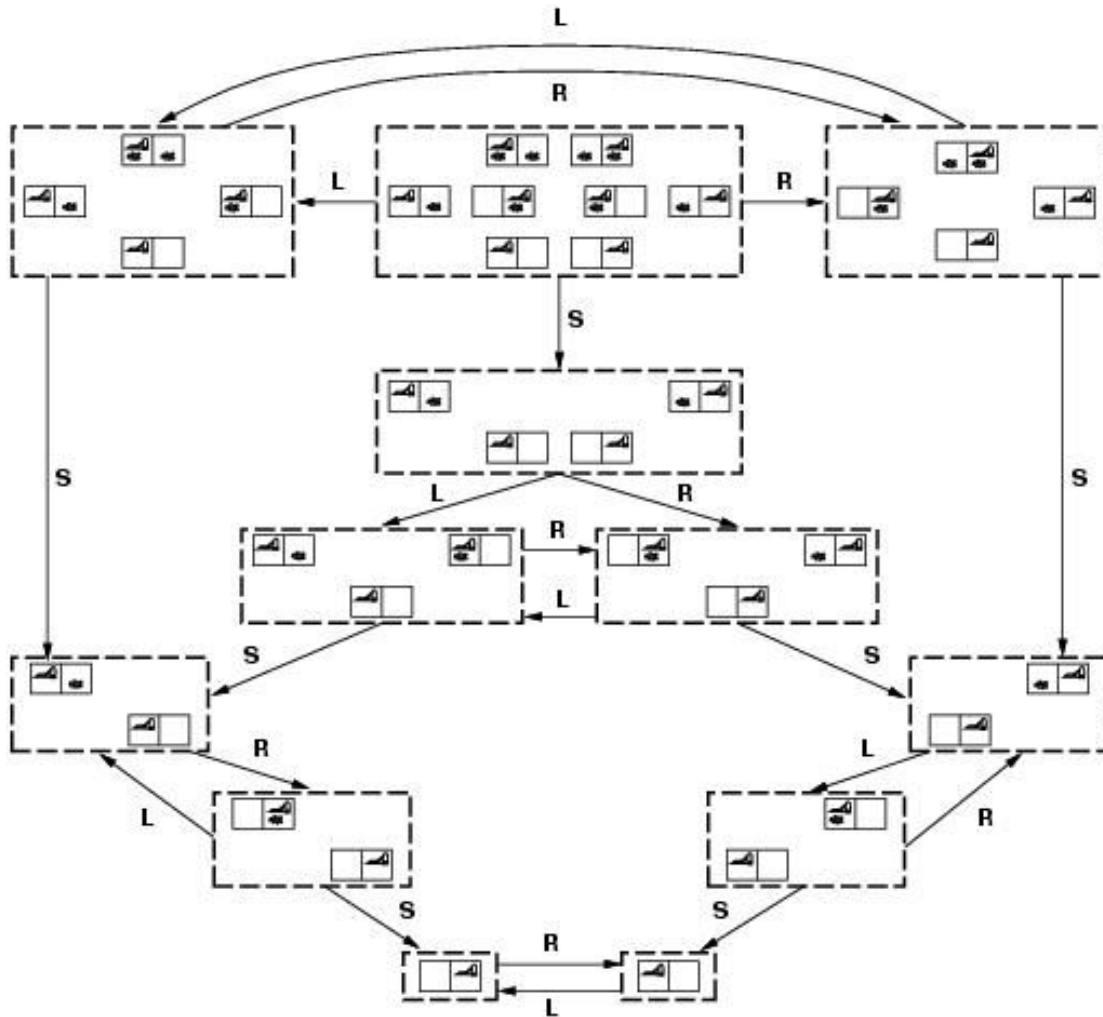
↪ فعالیت (Right,Suck,Left,Suck) تضمين ميکند

که صرف نظر از حالت اوليه ، به حالت هدف ، يعني 7 برسد

حل مسئله با جستجو



دنیای جاروبرقی فاقد حسگر



↪ عامل باید راجع به مجموعه های حالتی که میتواند به آنها برسد استدلال کند. این مجموعه از حالتها را حالت باور گوئیم.

↪ اگر فضای حالت فیزیکی دارای S حالت باشد فضای حالت باور 2^S حالت باور خواهد داشت.



هوش مصنوعي

فصل چهارم

جست و جوی آگاهانه و اکتشاف



فهرست

متدهای جست و جوی آگاهانه

یادگیری برای جست و جوی بهتر

جست و جوی محلی و بهینه سازی

جست و جوی محلی در فضاها پیوسته

عاملهای جست و جوی **Online**



متمدهای جستجوی آگاهانه

و بهینه سازی

جستجوی محلی

تپه نوردی

شبیه سازی حرارت

پرتو محلی

الگوریتمهای ژنتیک

بهترین جستجو

حریصانه

A*

IDA*

RBFS

SMA* و MA*



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

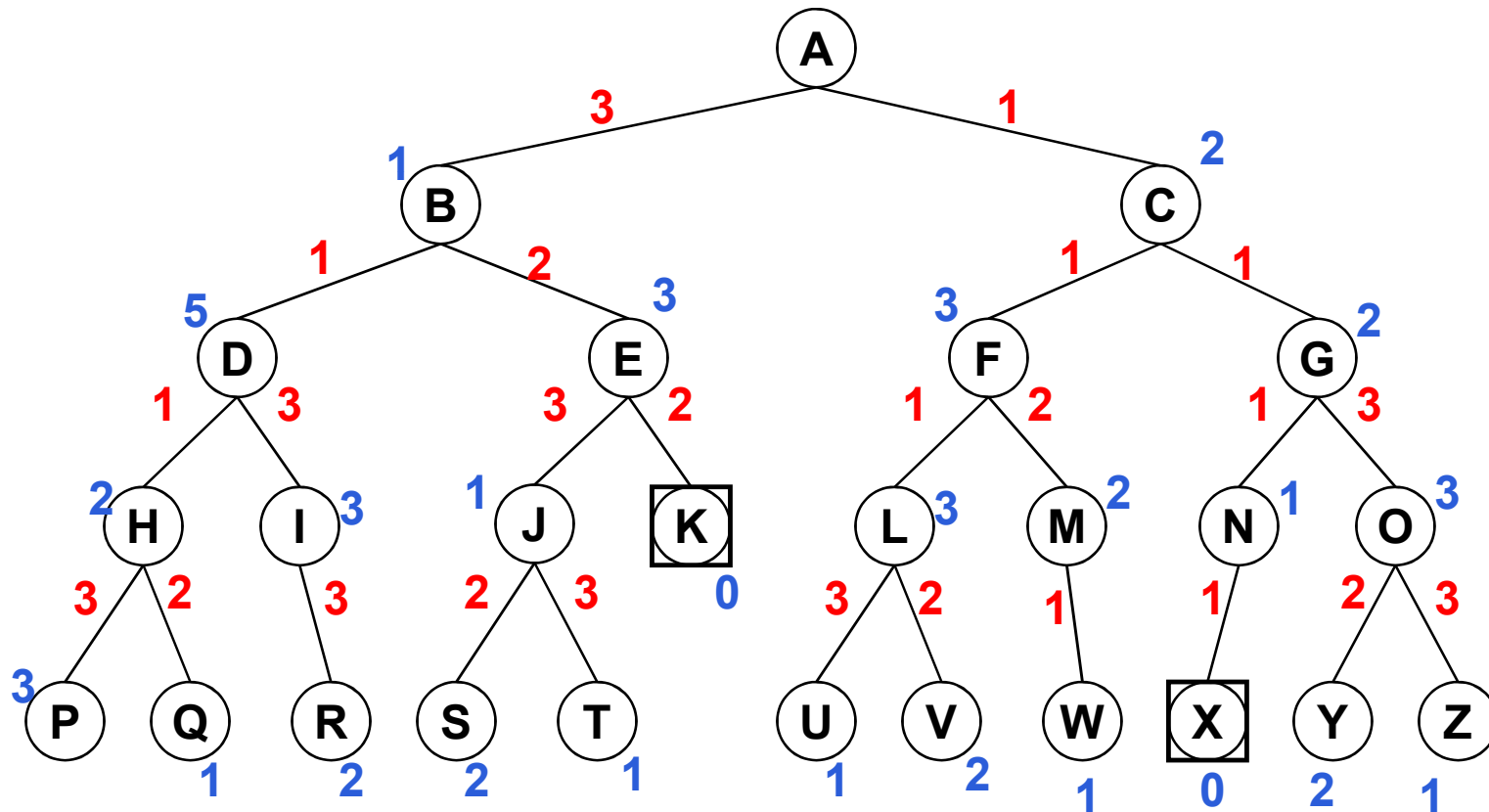
$$f(n): g(n) + h(n)$$

$$f^*(n): g(n) + h^*(n) \quad \text{هزینه ارزان ترین مسیر از طریق } n$$

جست و جوی آگاهانه و اکتشاف



جستجوی حریصانه



جست و جوی آگاهانه و اکتشاف



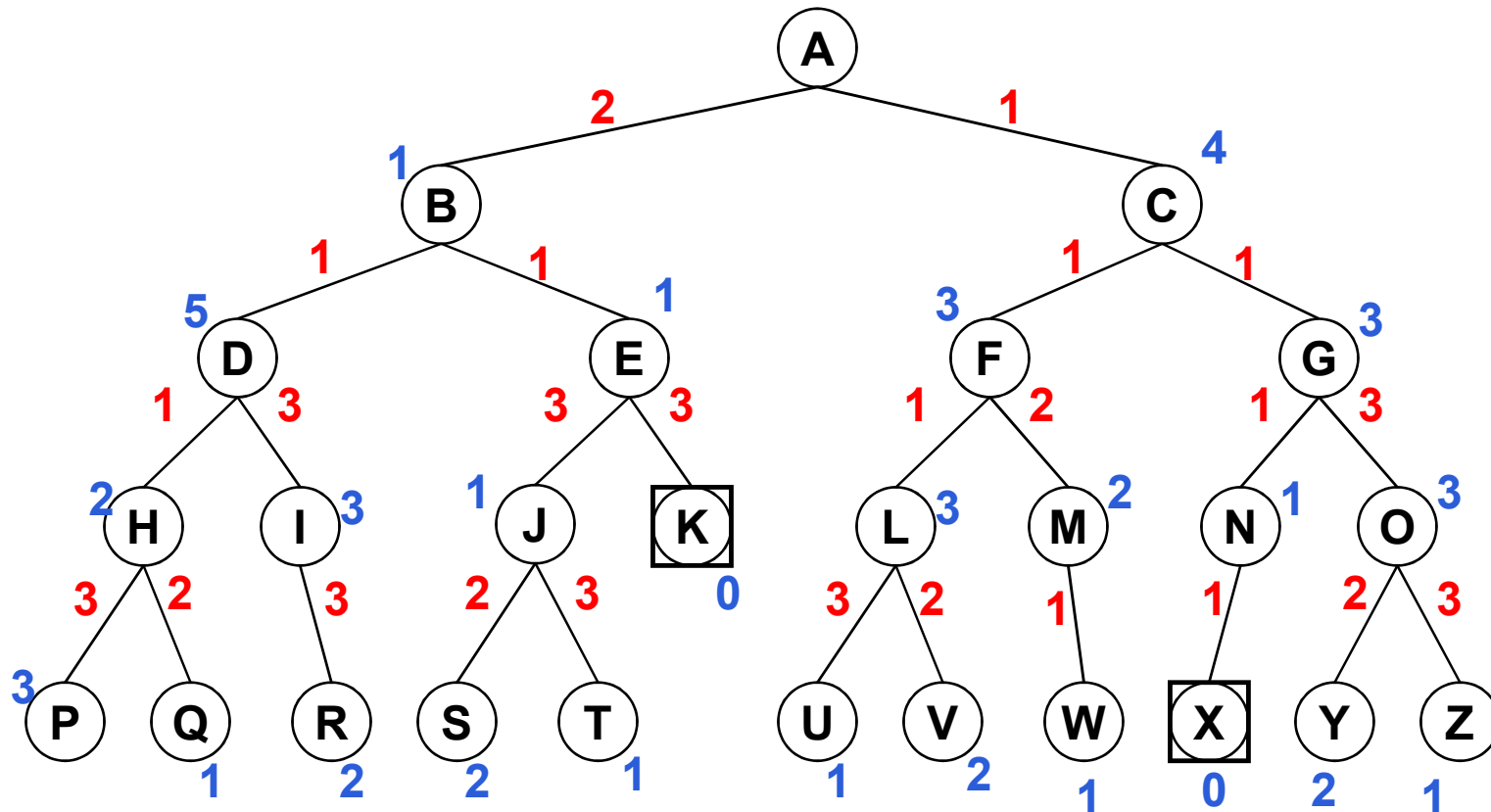
جستجوی حریصانه

(A)

جست و جوی آگاهانه و اکتشاف



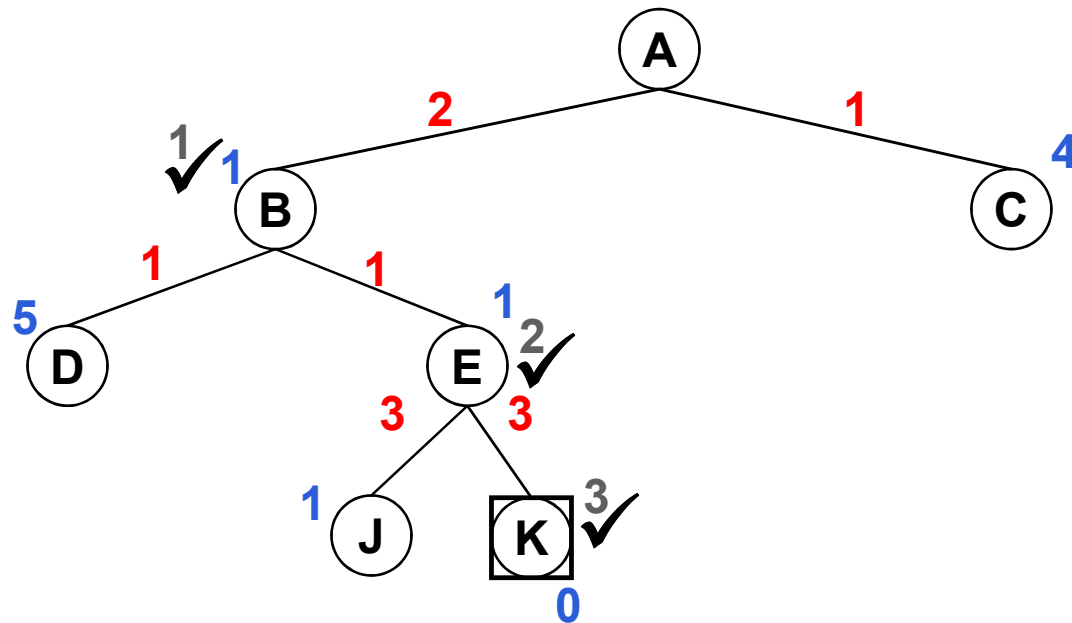
جستجوی حریصانه



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

← پهنگی: خیر

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

← پیچیدگی زمانی: $O(b^m)$

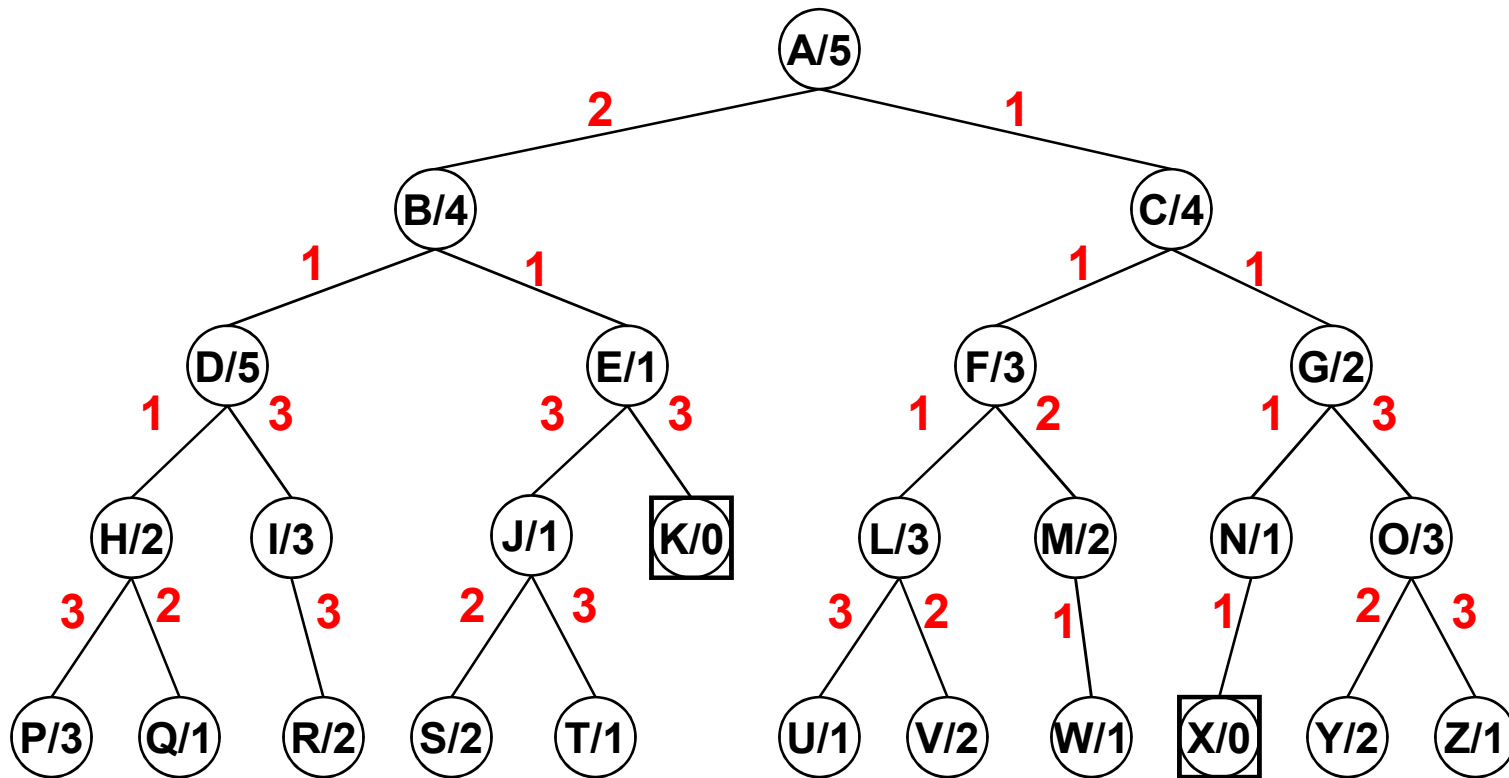
← اما اگر $h = h^*$ آنگاه $O(bd)$

← پیچیدگی فضا: $O(b^m)$

← اما اگر $h = h^*$ آنگاه $O(bd)$

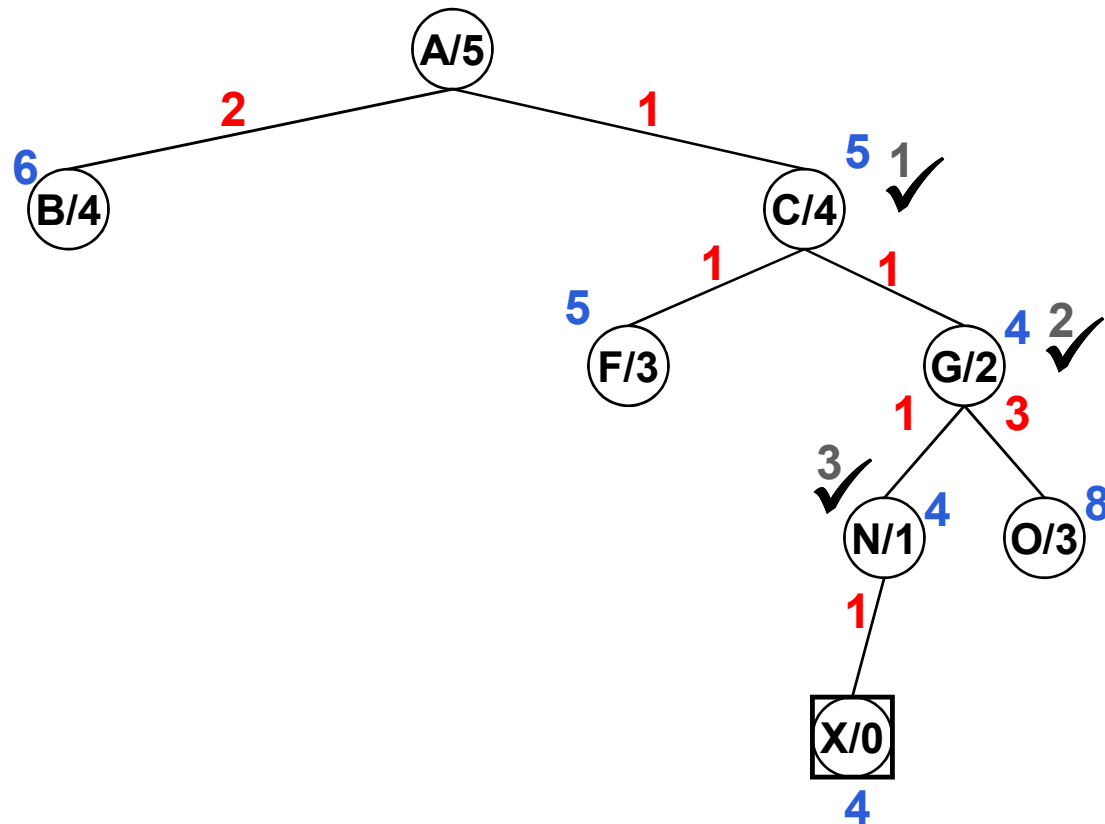
جست و جوی آگاهانه و اکتشاف

جستجوی A^*



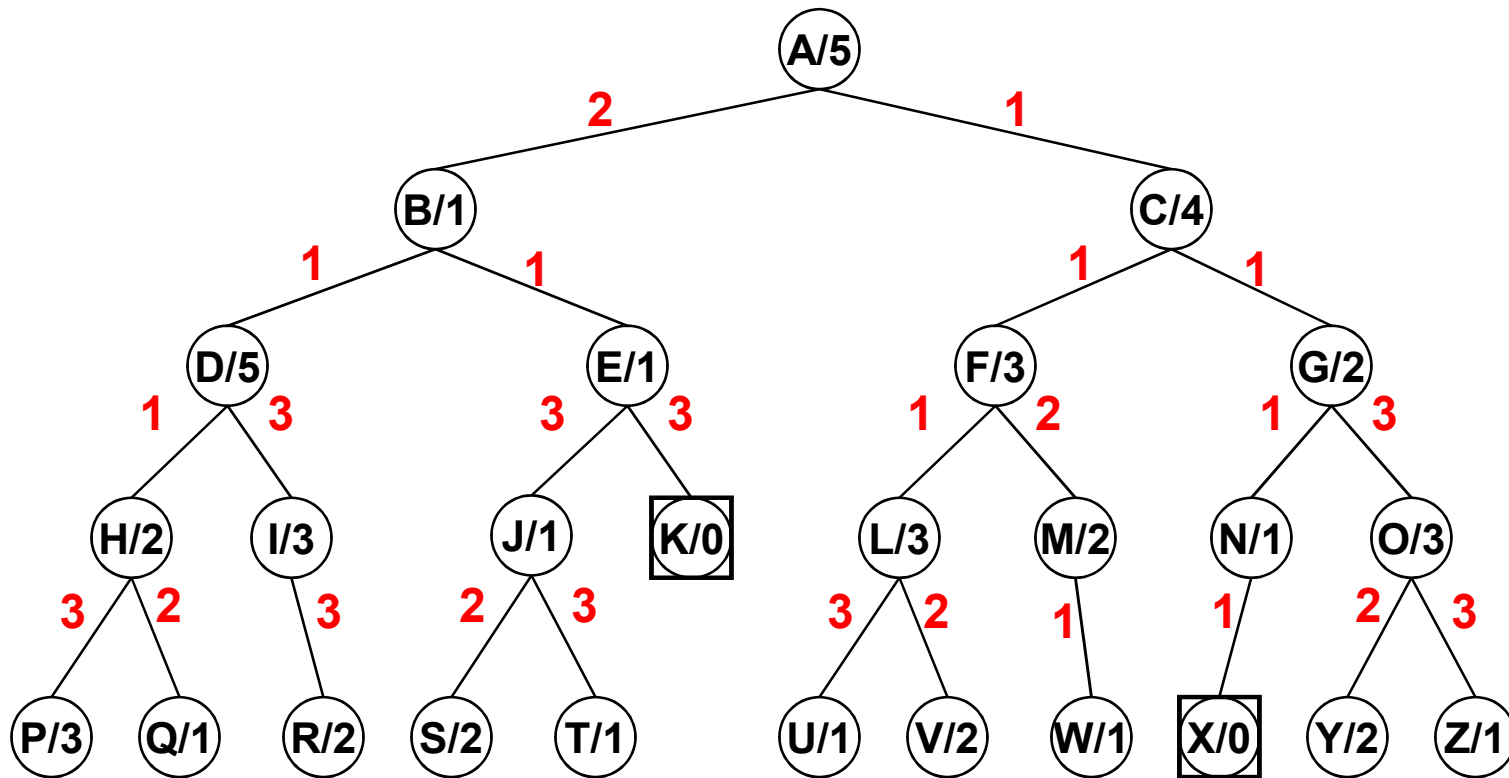
جست و جوی آگاهانه و اکتشاف

جستجوی A^*



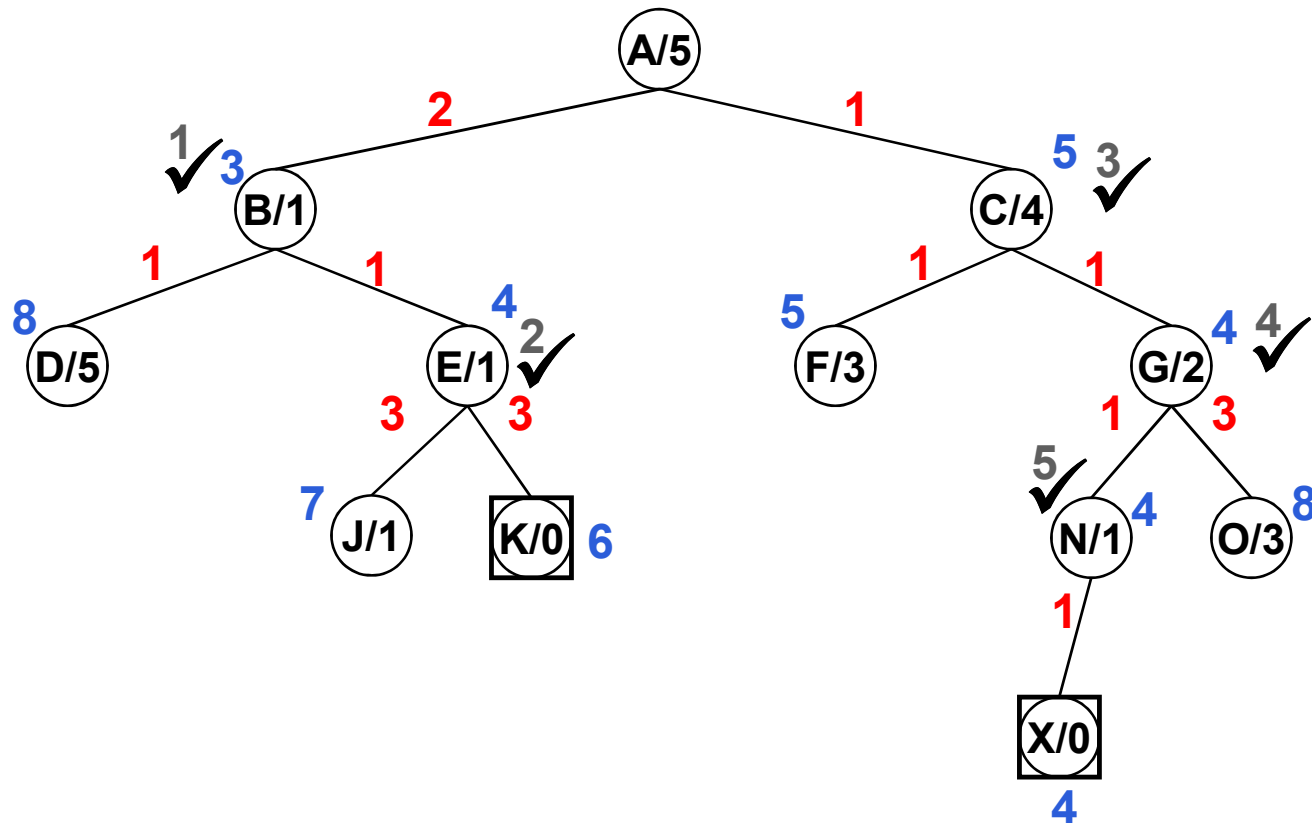
جست و جوی آگاهانه و اکتشاف

جستجوی A^*



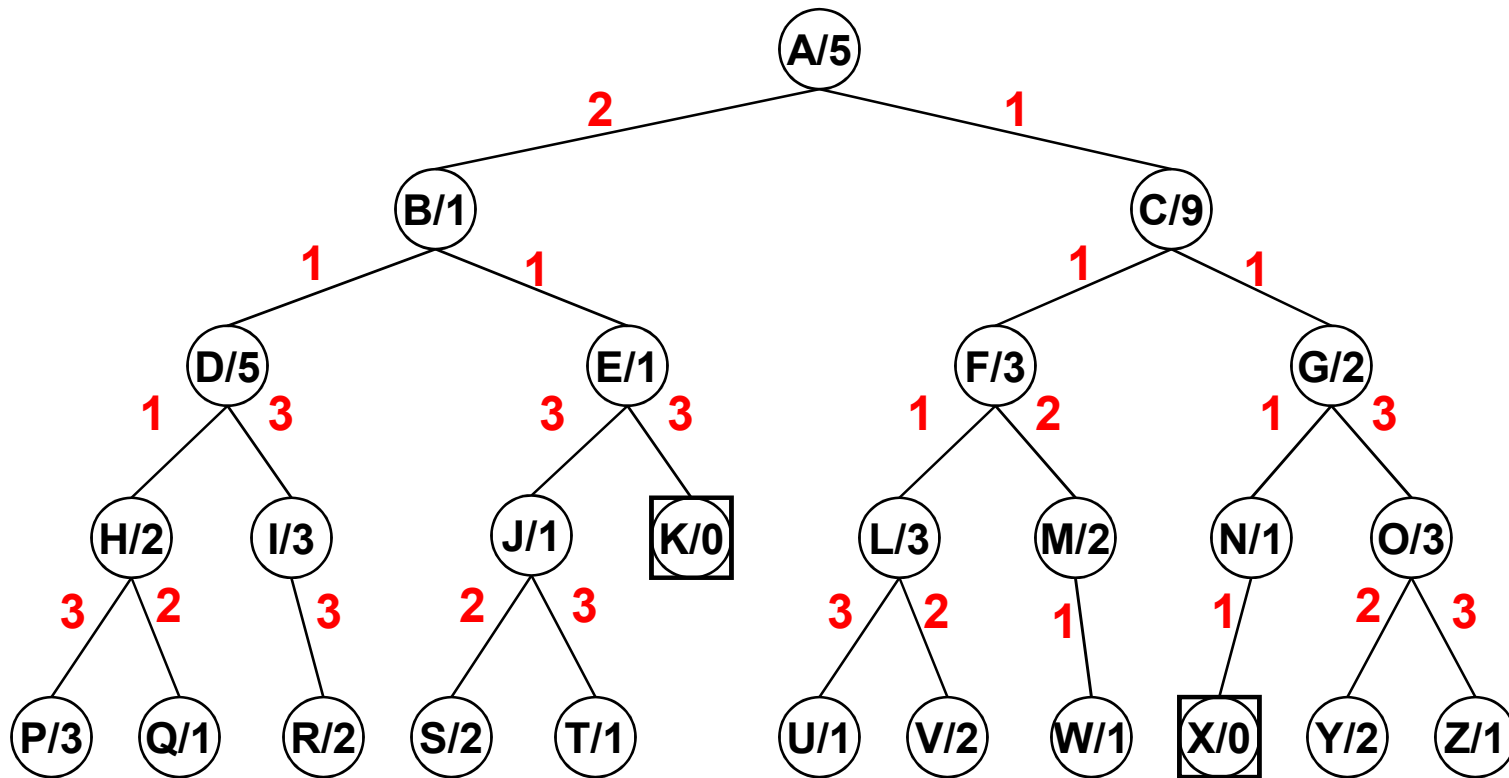
جست و جوی آگاهانه و اکتشاف

جستجوی A*



جست و جوی آگاهانه و اکتشاف

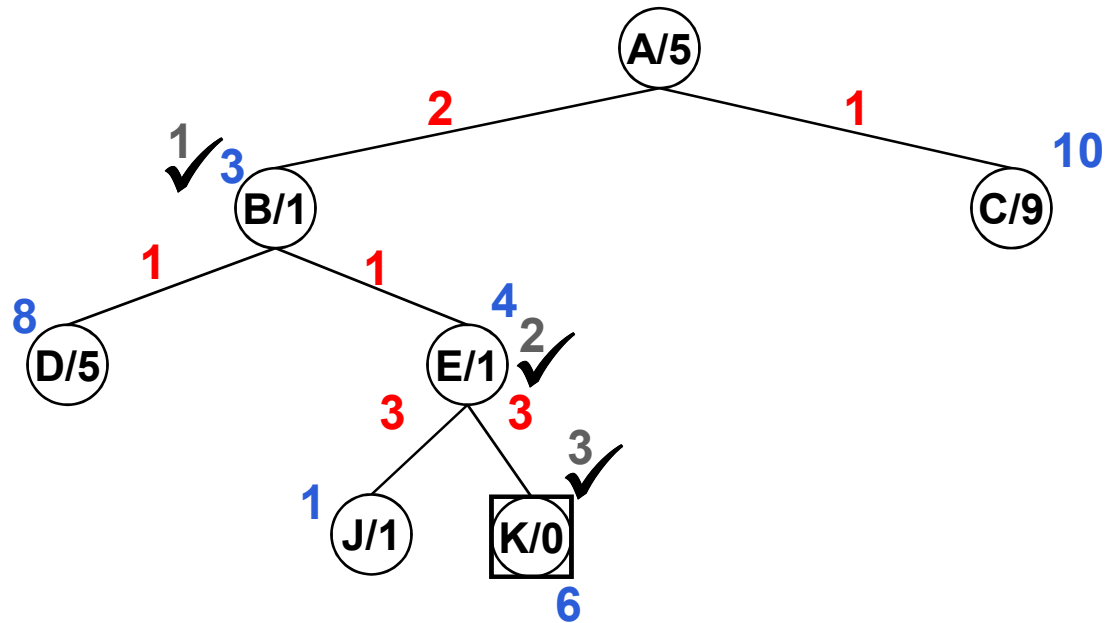
جستجوی A^*



جست و جوی آگاهانه و اکتشاف



جستجوی A*



جست و جوی آگاهانه و اکتشاف



جستجوی A^*

← کامل بودن: بله

← بهینگی: بله

← پیچیدگی زمانی:

$$O(b^m)$$

← اما اگر $h = h^*$ آنگاه $O(bd)$

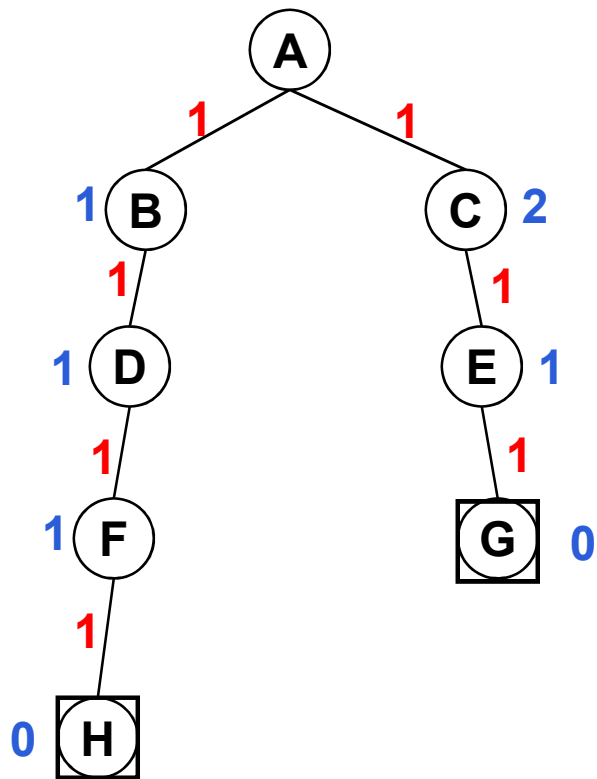
← پیچیدگی فضا:

$$O(b^m)$$

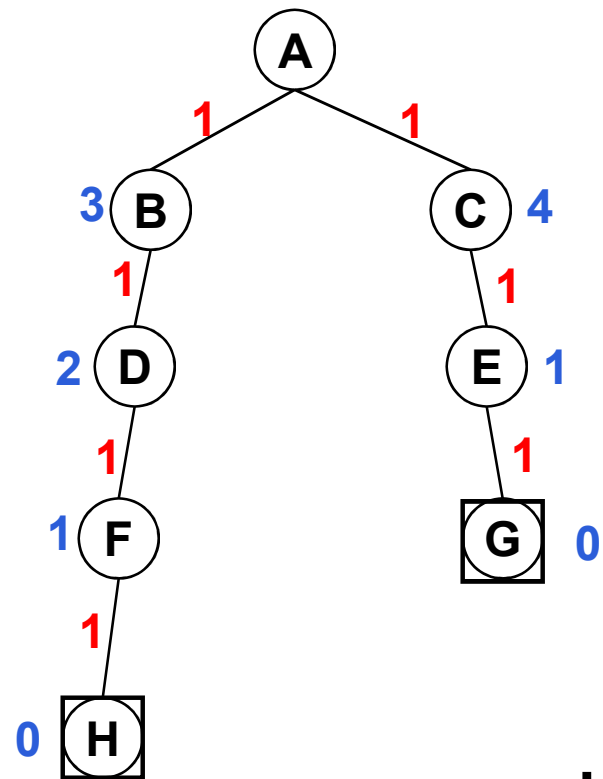
← اما اگر $h = h^*$ آنگاه $O(bd)$

جست و جوی آگاهانه و اکتشاف

جستجوی A^*



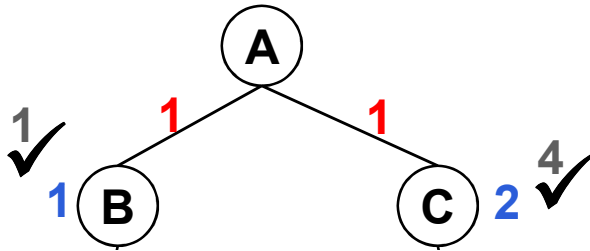
$$h \leq h^*$$



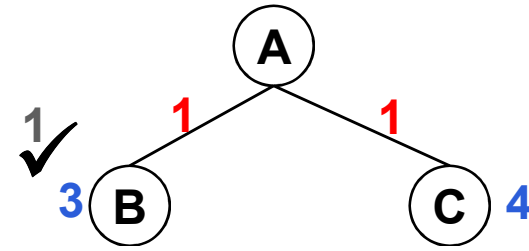
$$h \not\leq h^*$$

جست و جوی آگاهانه و اکتشاف

جستجوی A^*



$$h \leq h^*$$

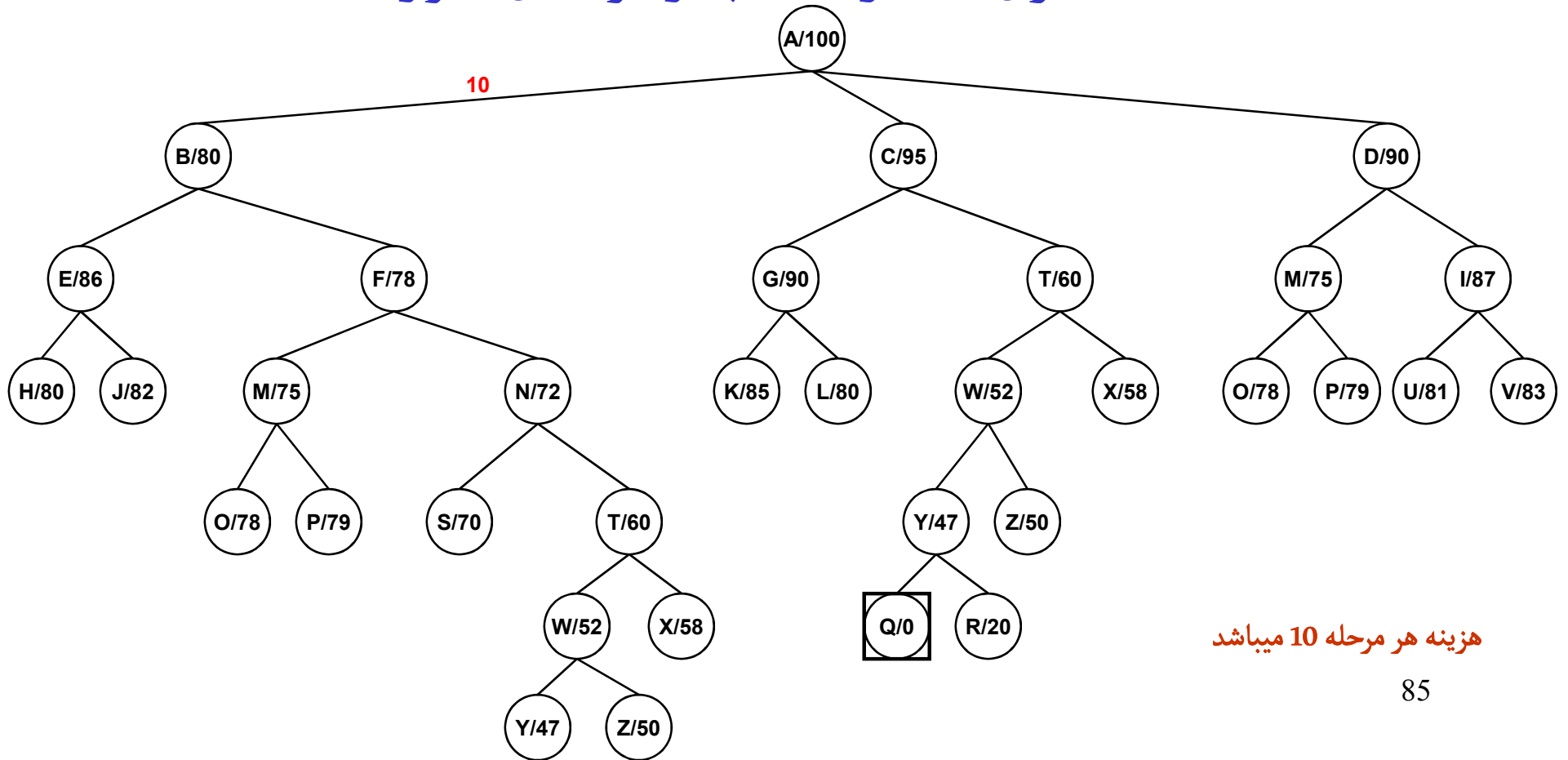


$$h \not\leq h^*$$

جست و جوی آگاهانه و اکتشاف



جستجوی A^* و اجتناب از گره های تکراری

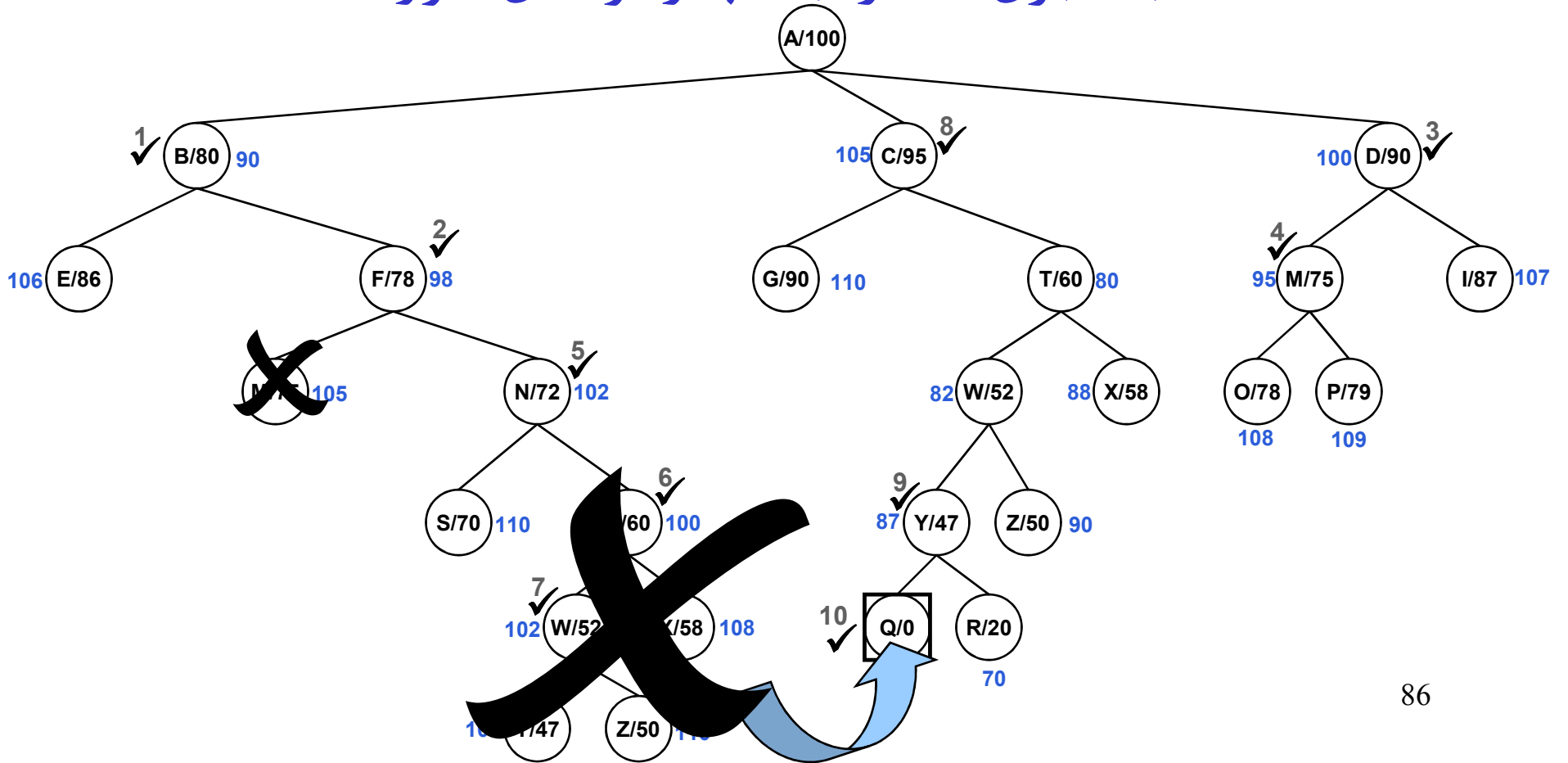


هزینه هر مرحله 10 میباشد

جست و جوی آگاهانه و اکتشاف



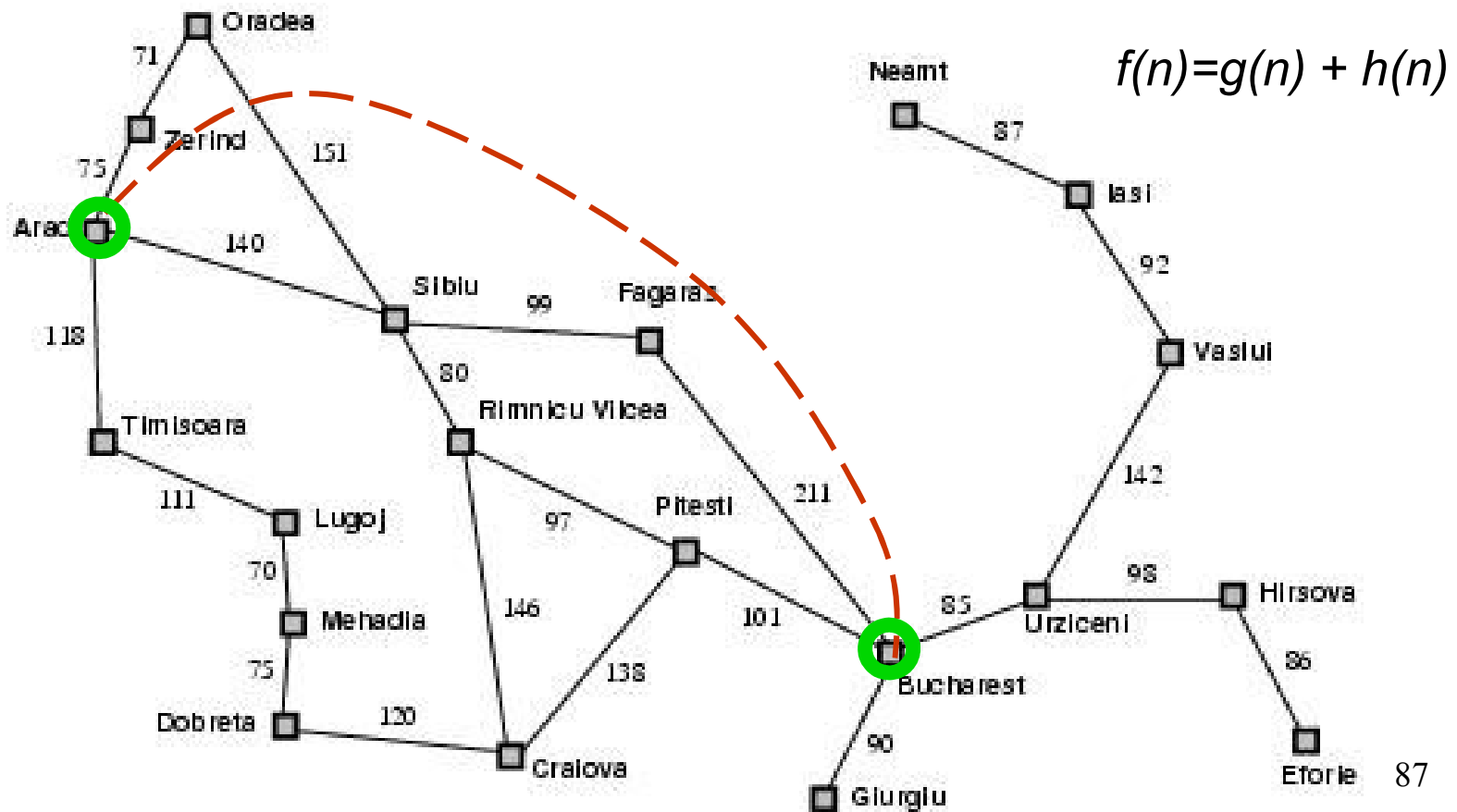
جستجوی A^* و اجتناب از گره های تکراری



جست و جوی آگاهانه و اکتشاف



مثال دیگر از جستجوی A^*



جست و جوی آگاهانه و اکتشاف

جستجوی A^* در نقشه رومانی

(a) The initial state



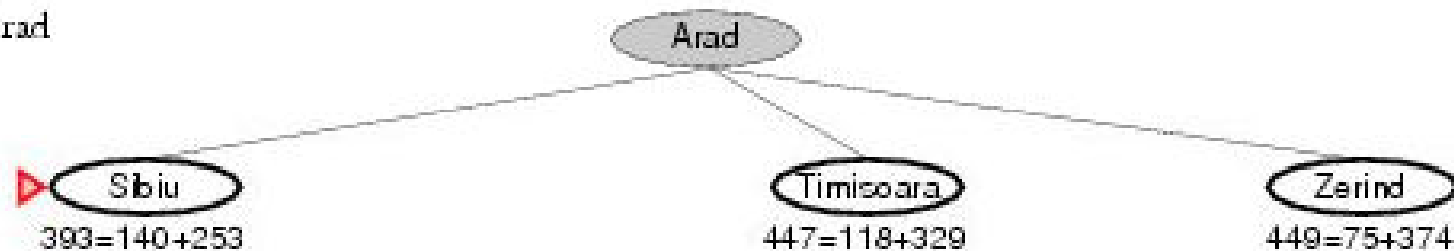
جستجوی Bucharest با شروع از Arad

$$f(\text{Arad}) = g(\text{Arad}) + h(\text{Arad}) = 0 + 366 = 366$$

جست و جوی آگاهانه و اکتشاف

جستجوی A^* در نقشه رومانی

After expanding Arad



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

$$f(\text{Sibiu})=c(\text{Arad},\text{Sibiu})+h(\text{Sibiu})=140+253=393$$

$$f(\text{Timisoara})=c(\text{Arad},\text{Timisoara})+h(\text{Timisoara})=118+329=447$$

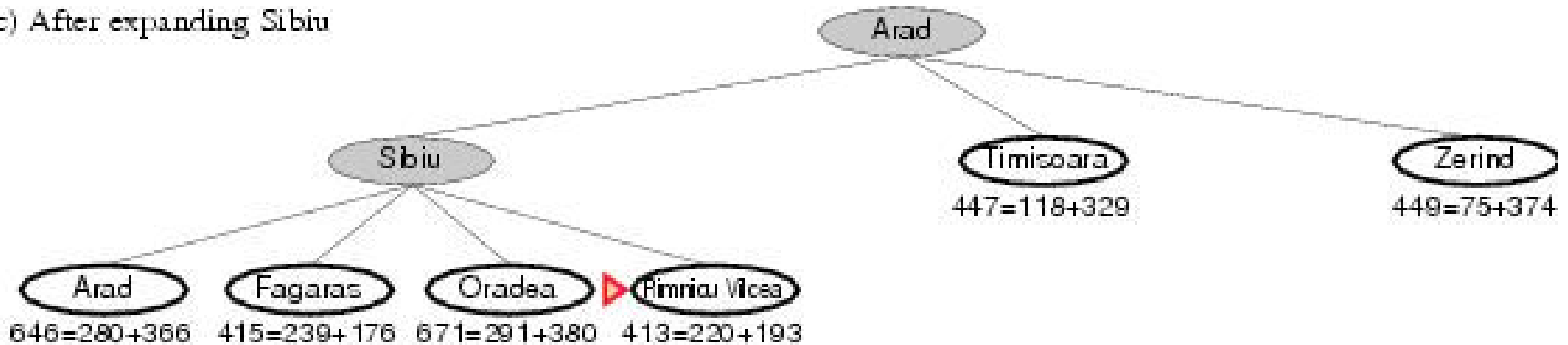
$$f(\text{Zerind})=c(\text{Arad},\text{Zerind})+h(\text{Zerind})=75+374=449$$

بهترین انتخاب شهر Sibiu است

جست و جوی آگاهانه و اکتشاف

جستجوی A^* در نقشه رومانی

(c) After expanding Sibiu



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

$$f(\text{Arad}) = c(\text{Sibiu}, \text{Arad}) + h(\text{Arad}) = 280 + 366 = 646$$

$$f(\text{Fagaras}) = c(\text{Sibiu}, \text{Fagaras}) + h(\text{Fagaras}) = 239 + 179 = 415$$

$$f(\text{Oradea}) = c(\text{Sibiu}, \text{Oradea}) + h(\text{Oradea}) = 291 + 380 = 671$$

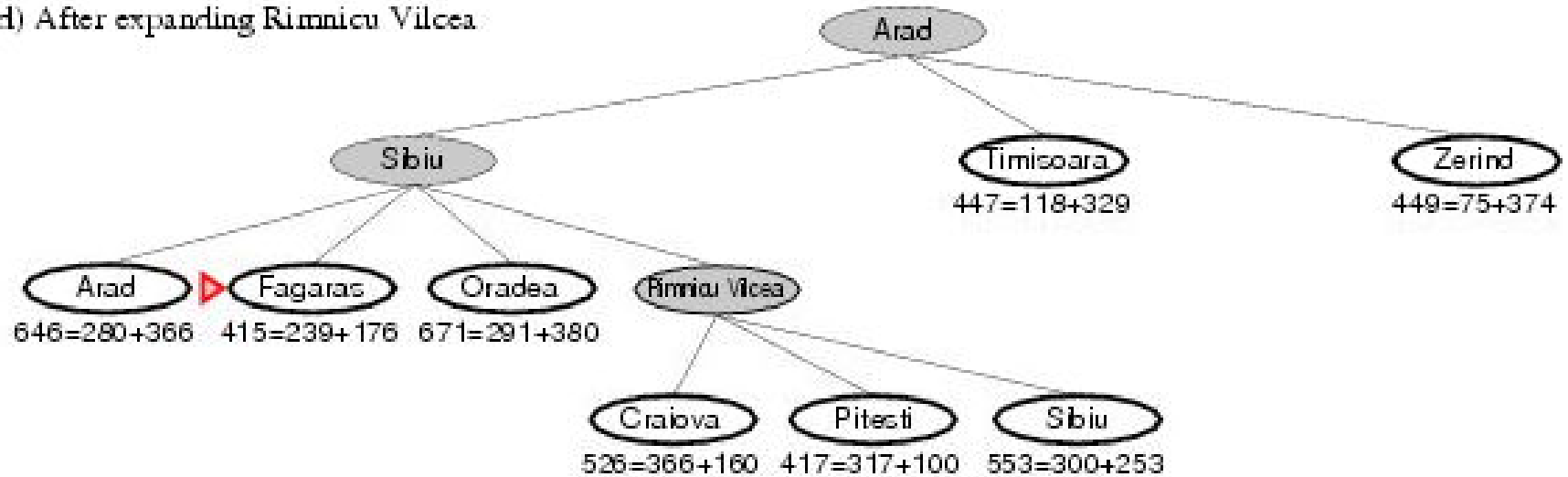
$$f(\text{Rimnicu Vilcea}) = c(\text{Sibiu}, \text{Rimnicu Vilcea}) + h(\text{Rimnicu Vilcea}) = 220 + 192 = 413$$

بهترین انتخاب شهر Rimnicu Vilcea است

جست و جوی آگاهانه و اکتشاف

جستجوی A^* در نقشه رومانی

(d) After expanding Rimnicu Vilcea



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

$$f(Craiova) = c(Rimnicu\ Vilcea, Craiova) + h(Craiova) = 360 + 160 = 526$$

$$f(Pitesti) = c(Rimnicu\ Vilcea, Pitesti) + h(Pitesti) = 317 + 100 = 417$$

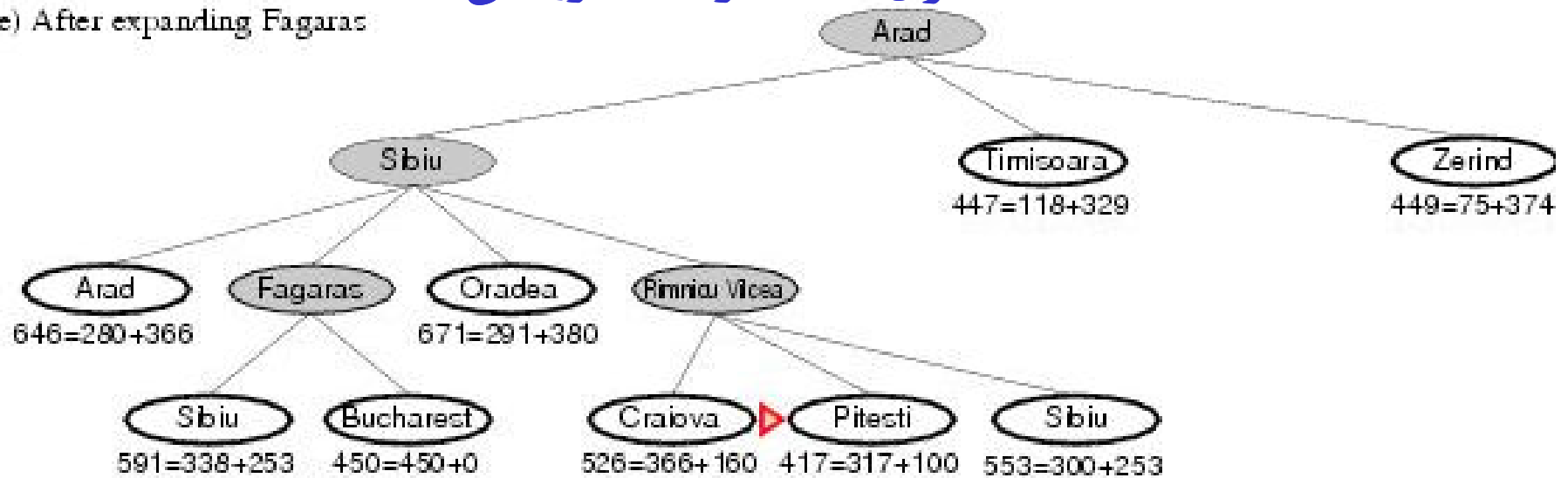
$$f(Sibiu) = c(Rimnicu\ Vilcea, Sibiu) + h(Sibiu) = 300 + 253 = 553$$

بهترین انتخاب شهر Fagaras است

جست و جوی آگاهانه و اکتشاف

جستجوی A^* در نقشه رومانی

(e) After expanding Fagaras



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

$$f(\text{Sibiu}) = c(\text{Fagaras}, \text{Sibiu}) + h(\text{Sibiu}) = 338 + 253 = 591$$

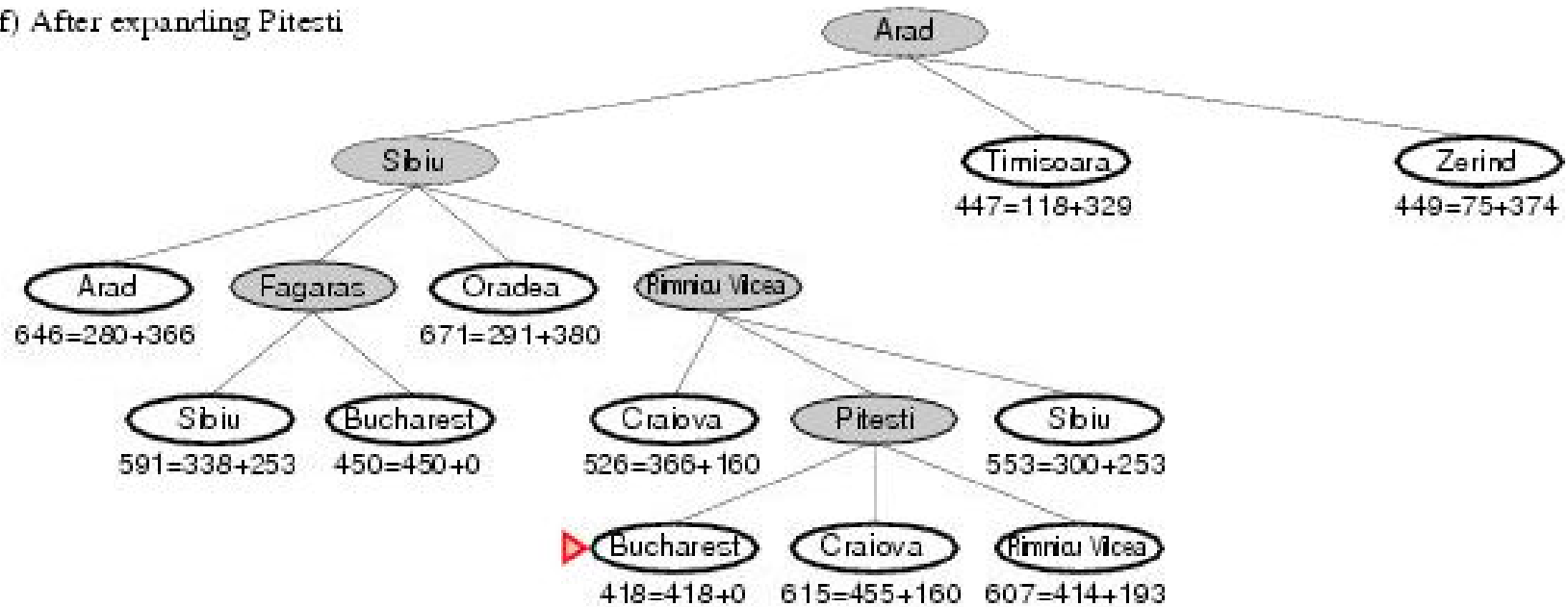
$$f(\text{Bucharest}) = c(\text{Fagaras}, \text{Bucharest}) + h(\text{Bucharest}) = 450 + 0 = 450$$

بهترین انتخاب شهر **Pitesti !!!** است

جست و جوی آگاهانه و اکتشاف

جستجوی A^* در نقشه رومانی

(f) After expanding Pitesti



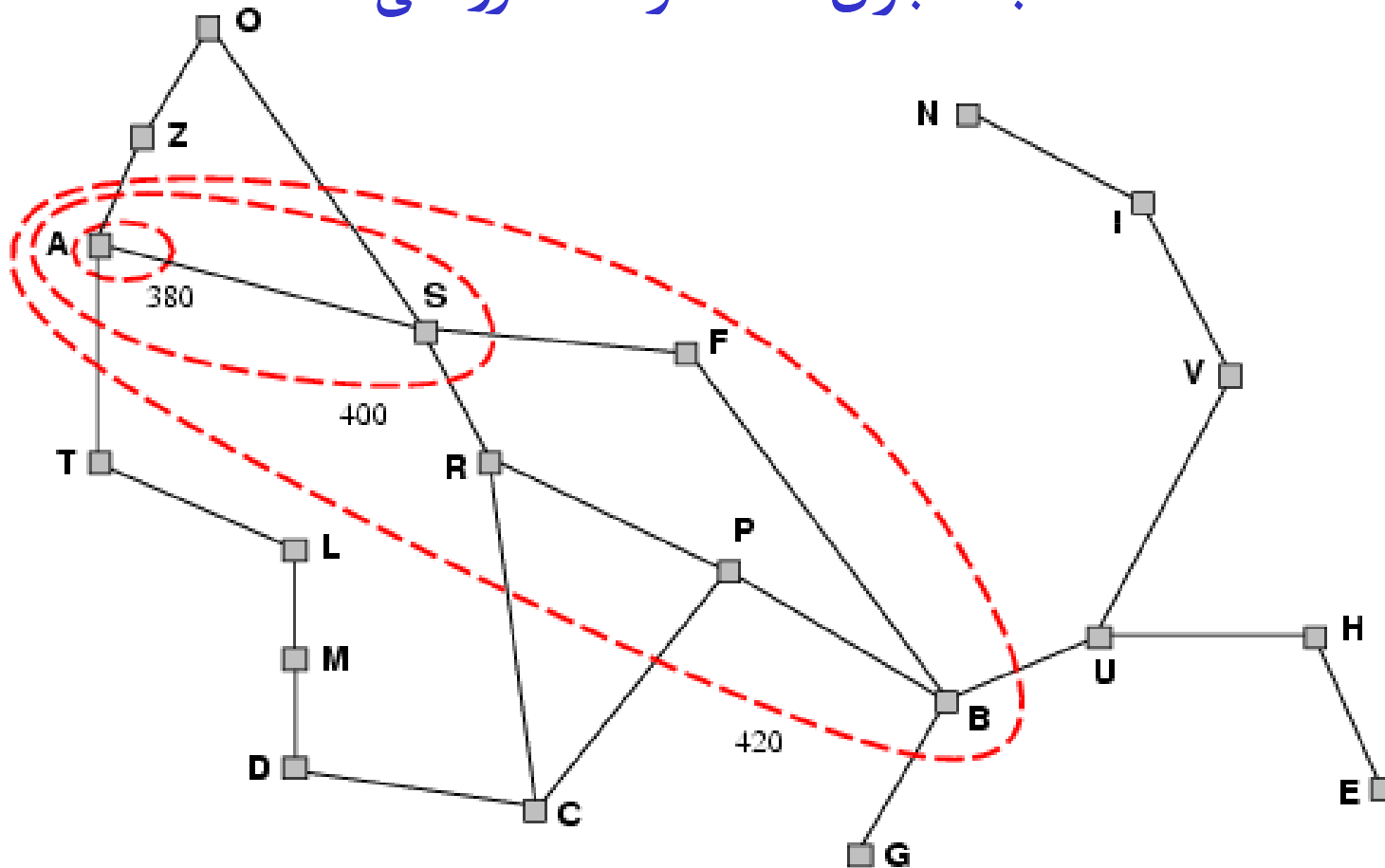
۶۶ Pitesti را باز کرده و $f(n)$ را برای هر یک از زیربرگها محاسبه میکنیم:

$$f(\text{Bucharest}) = c(\text{Pitesti}, \text{Bucharest}) + h(\text{Bucharest}) = 418 + 0 = 418$$

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 اکتشافی است.

↩ الگوریتم عمیق کننده تکرار A^* ← IDA^*

↩ در جستجوی IDA^* مقدار برش مورد استفاده ، عمق نیست بلکه هزینه $f(g+h)$ است.

↩ برای اغلب مسئله های با هزینه های مرحله ای ، مناسب است و از سر بار ناشی از نگهداری صف مرتبی از گره ها اجتناب میکند

جست و جوی آگاهانه و اکتشاف



بهترین جستجوی بازگشتی RBFS

↩ ساختار آن شبیه جست و جوی عمقی بازگشتی است ، اما به جای اینکه دائماً 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 دیگری را انتخاب کند.

↩ این جستجو اگر تابع اکتشافی قابل قبولی داشته باشد ، بهینه است.

↩ پیچیدگی فضایی 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



بهترین جستجوی بازگشتی RBFS

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

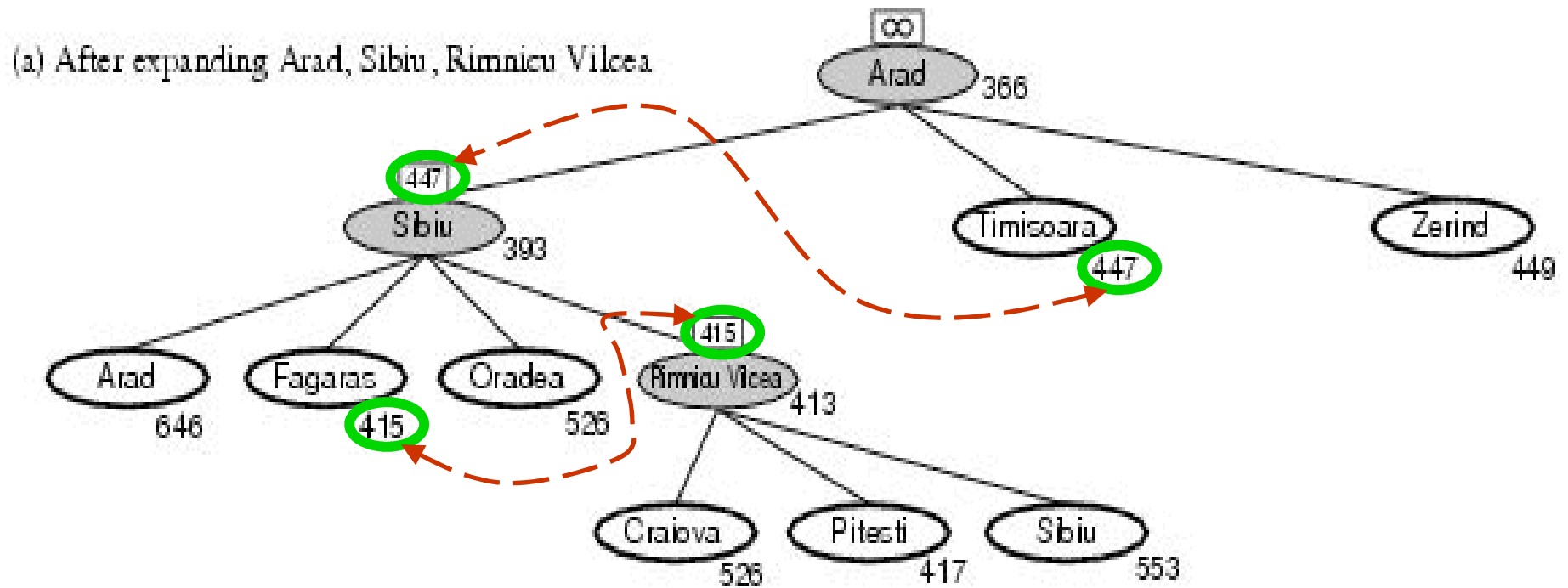
IDA* و RBFS در معرض افزایش توانی پیچیدگی قرار دارند که در جست و جوی گرافها مرسوم است ، زیرا نمیتوانند حالت های تکراری را در غیر از مسیر فعلی بررسی کنند. لذا ، ممکن است یک حالت را چندین بار بررسی کنند.

IDA* و RBFS از فضای اندکی استفاده میکنند که به آنها آسیب میرساند. IDA* بین هر تکرار فقط یک عدد را نگهداری میکند که فعلی هزینه f است. RBFS اطلاعات بیشتری در حافظه نگهداری میکند

جست و جوی آگاهانه و اکتشاف



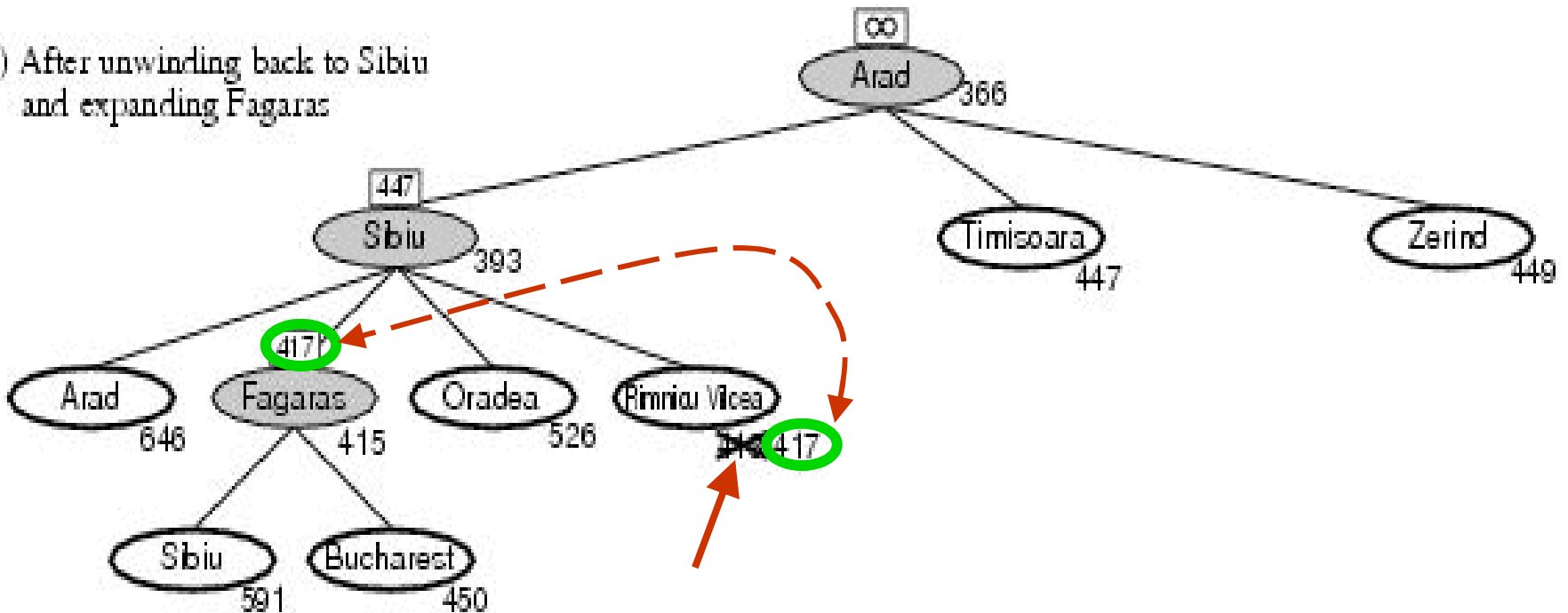
بهترین جستجوی بازگشتی در نقشه رومانی



جست و جوی آگاهانه و اکتشاف

بهترین جستجوی بازگشتی در نقشه رومانی

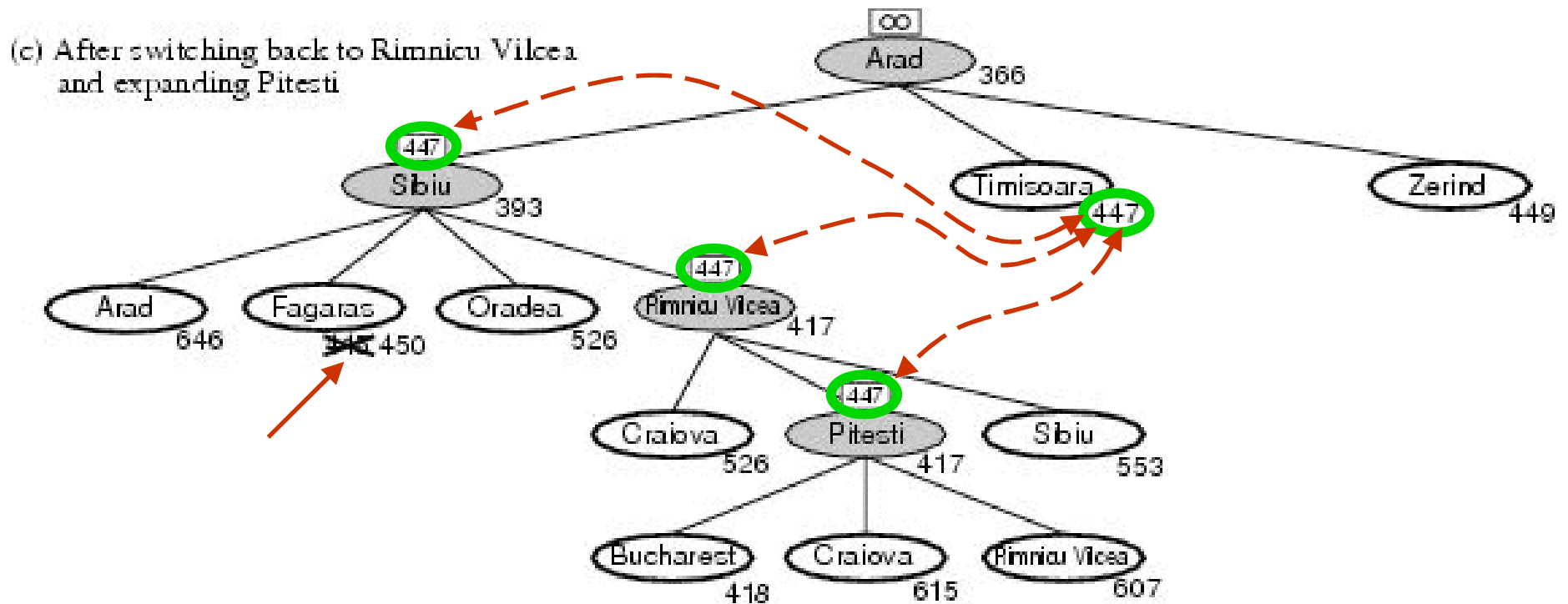
(b) After unwinding back to Sibiu and expanding Fagaras



جست و جوی آگاهانه و اکتشاف



بهترین جستجوی بازگشتی 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 قبلی 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 حل های بهینه میباشد



جستجوی حافظه محدود ساده SMA^*

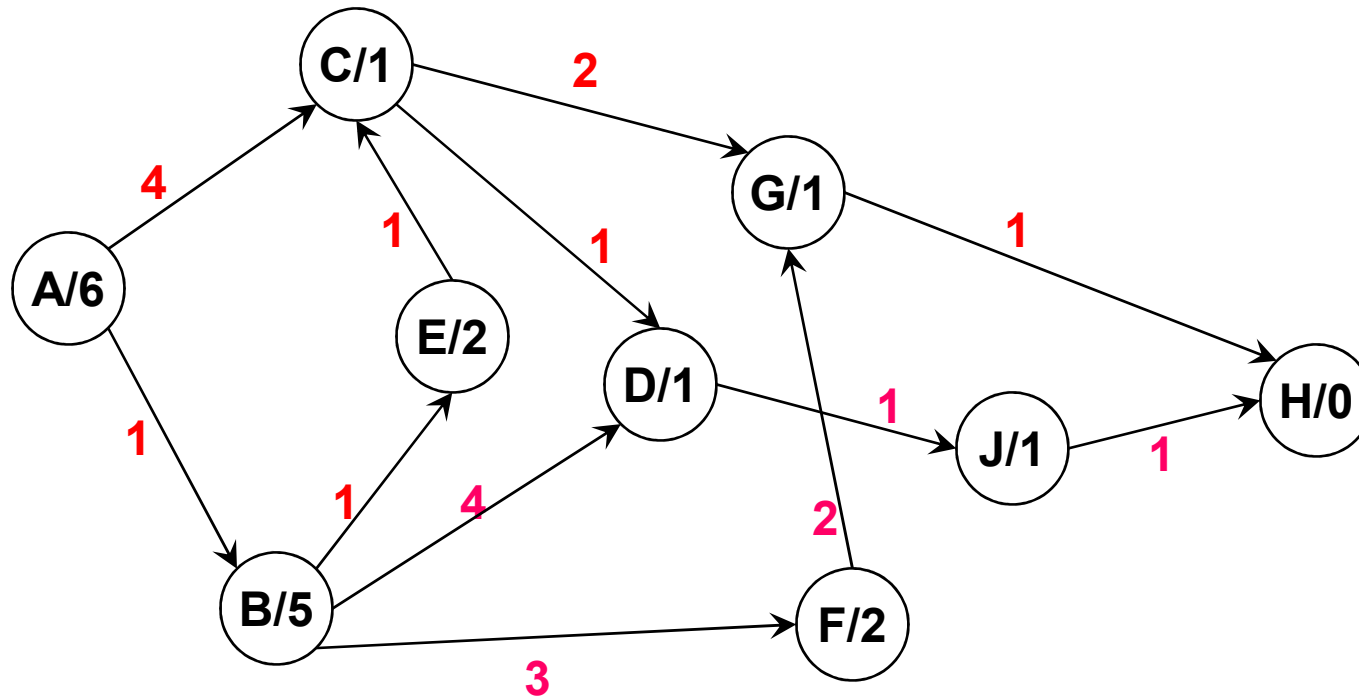
↩ اگر مقدار f تمام برگها یکسان باشد و الگوریتم یک گره را هم برای بسط و هم برای حذف انتخاب کند ، SMA^* این مسئله را با بسط بهترین برگ جدید و حذف بهترین برگ قدیمی حل میکند

↩ ممکن است SMA^* مجبور شود دائماً 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.

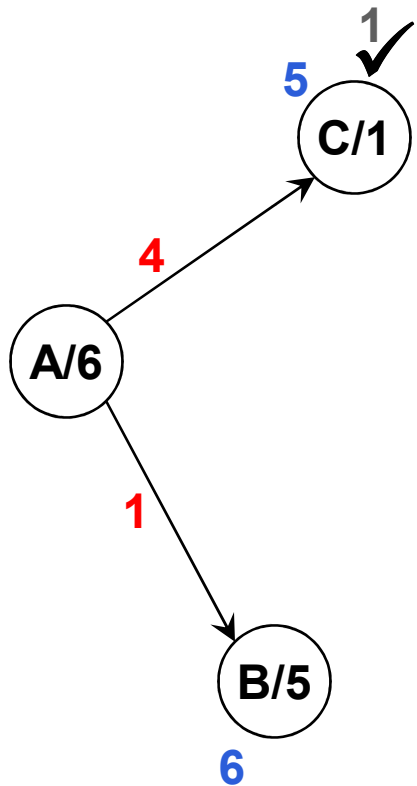


جستجوی گراف با A^*



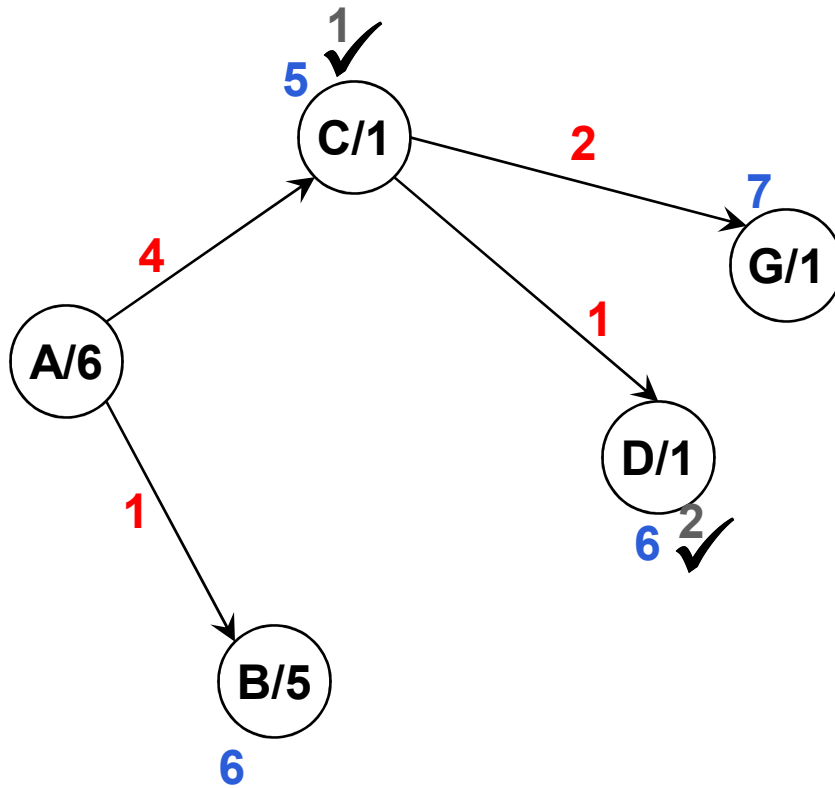


جستجوی گراف با A^*



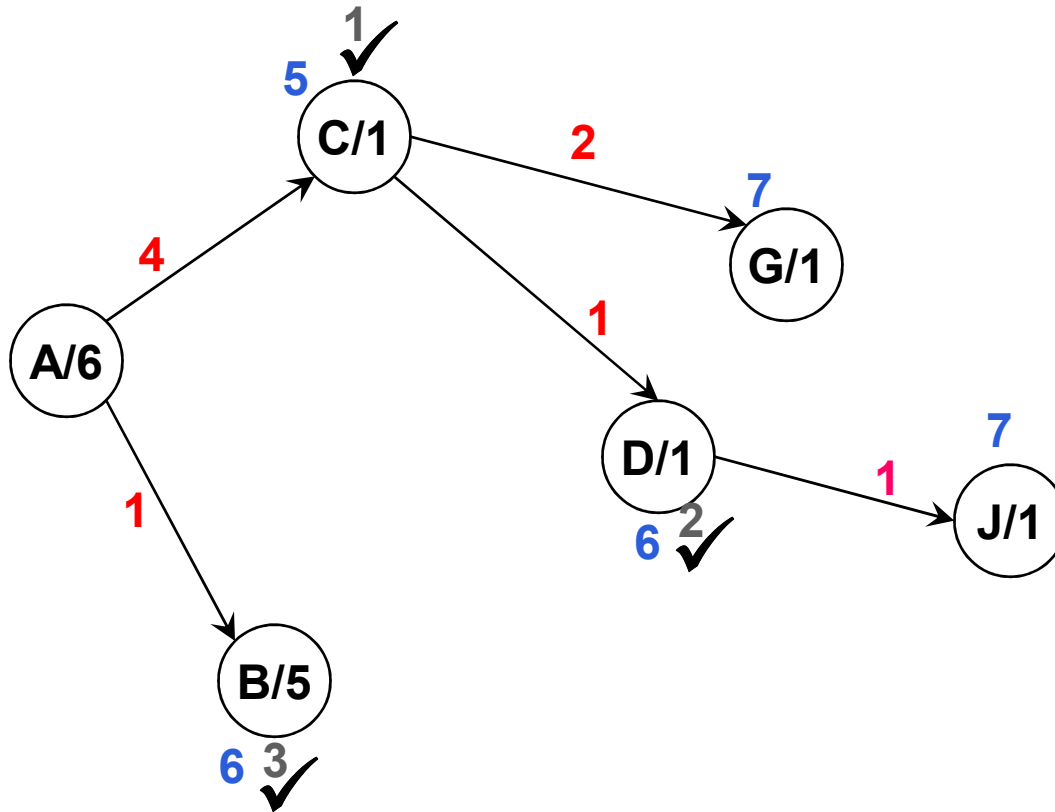


جستجوی گراف با A^*



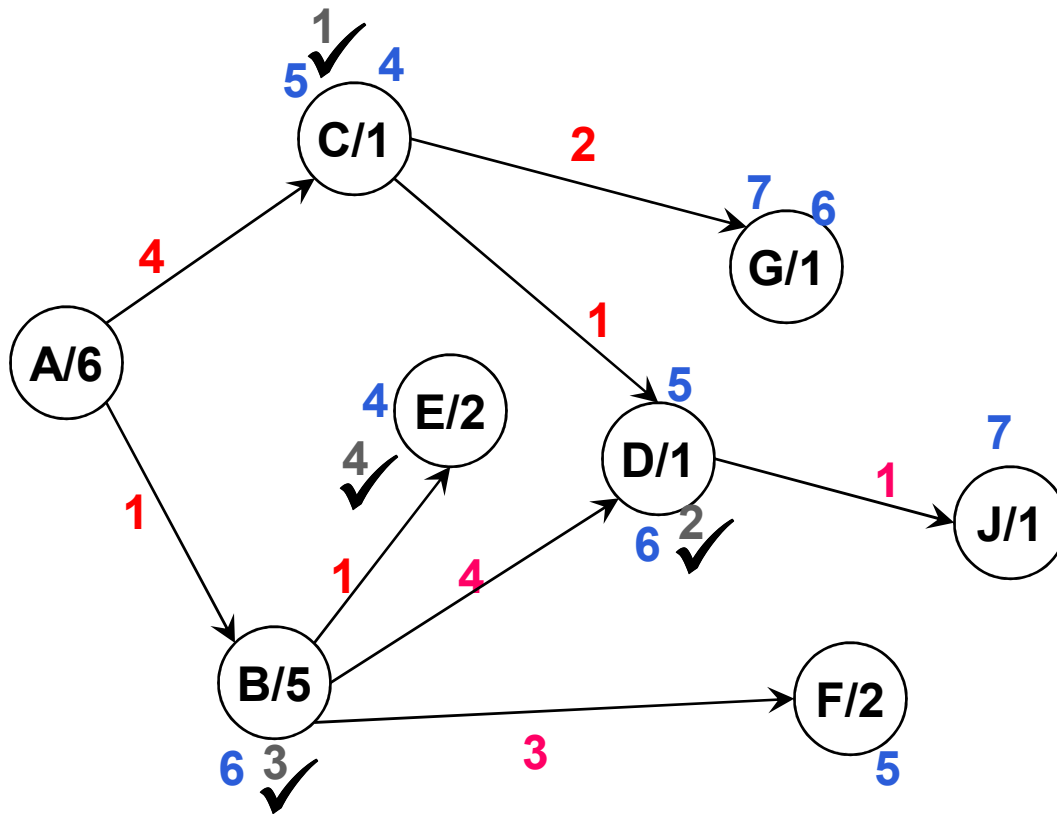


جستجوی گراف با A^*



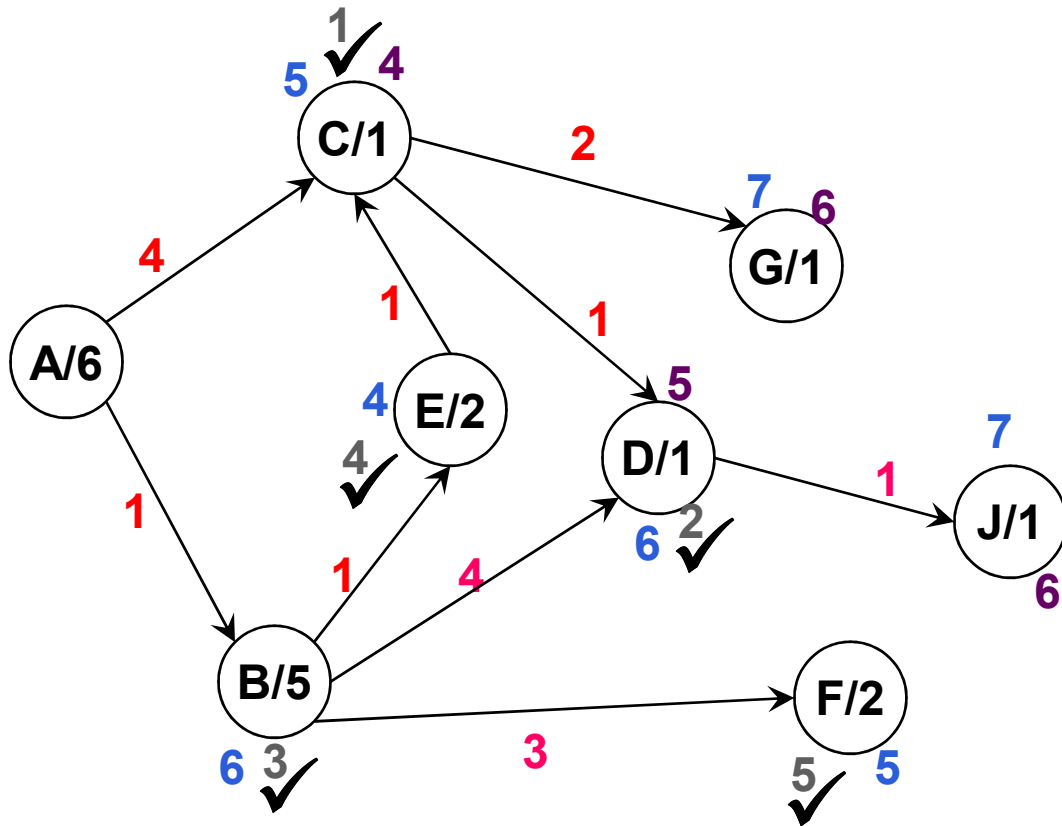


جستجوی گراف با A^*



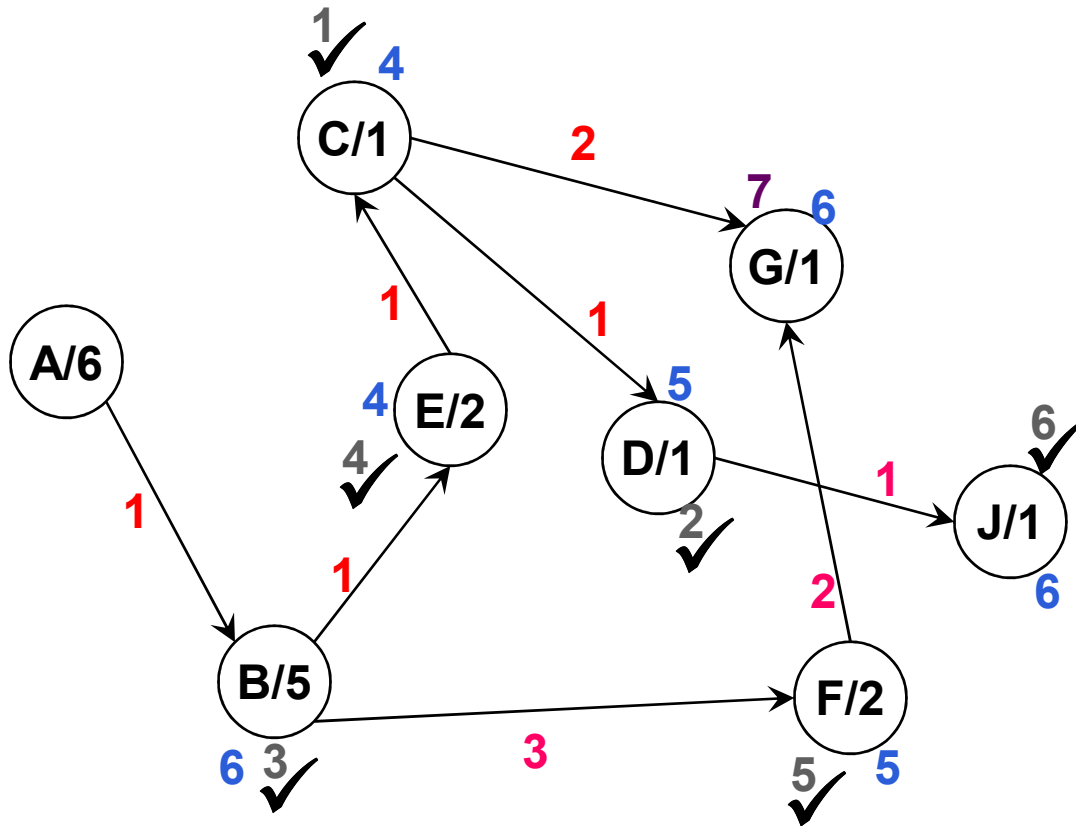


جستجوی گراف با A^*



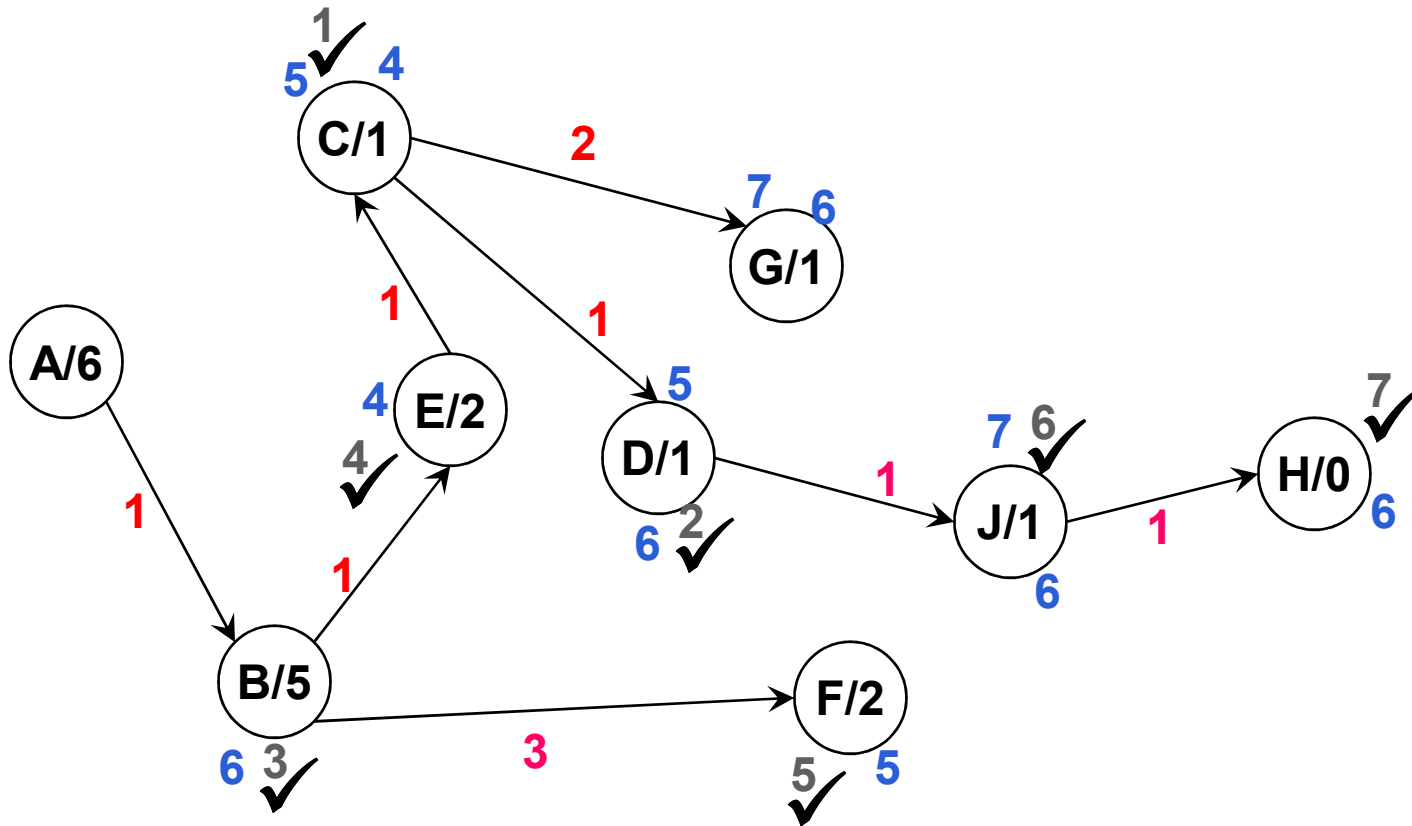


جستجوی گراف با A^*





جستجوی گراف با A^*



جست و جوی آگاهانه و اکتشاف



یادگیری برای جست و جوی بهتر

↪ روشهای جست و جوی قبلی ، 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 فراسطحی ، حالت (محاسباتی) داخلی برنامه ای را تسخیر میکند که فضای حالت سطح شیء ، مثل رومانی را جست و جو میکند

↪ الگوریتم یادگیری فراسطحی میتواند چیزهایی 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

7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

Goal State

مثال برای معمای 8

- میانگین هزینه حل تقریباً 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 اکتشافی متداول برای معمای 8

7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

Goal State

$$h_1 = 8$$

تعداد کاشیها در مکانهای نادرست
در حالت شروع

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

جست و جوی آگاهانه و اکتشاف



دو روش اکتشافی متداول برای معمای 8

7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

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.



دو روش اکتشافی متداول برای معمای 8

7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

Goal State

$h_2 =$ مجموعه فواصل کاشیها از موقعیتهای هدف آنها
 h_2 قابل قبول است ، زیرا هر جابجایی که میتواند انجام گیرد ، به اندازه یک مرحله به هدف نزدیک میشود.

هیچ کدام از این برآوردها ، هزینه واقعی راه حل نیست

هزینه واقعی 36 است

جست و جوی آگاهانه و اکتشاف



اثر دقت اکتشاف بر کارایی

فاکتور انشعاب مؤثر b^*

اگر تعداد گره هایی 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

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

فاکتور انشعاب مؤثر معمولاً برای مسئله های سخت ثابت است

اندازه گیریهای تجربی b^* بر روی مجموعه کوچکی از مسئله ها میتواند راهنمای خوبی برای مفید بودن اکتشاف باشد

مقدار b^* در اکتشافی با طراحی خوب، نزدیک 1 است

جست و جوی آگاهانه و اکتشاف



اثر دقت اکتشاف بر کارایی

d	هزینه جست و جو			فاکتور انشعاب مؤثر		
	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 h_1 و h_2

جست و جوی آگاهانه و اکتشاف



اثر دقت اکتشاف بر کارایی

↪ اگر برای هر گره n داشته باشیم: $h2(n) \geq h1(n)$

↪ $h2$ بر $h1$ غالب است

↪ غالب بودن مستقیماً به کارایی ترجمه میشود

↪ تعداد گره هایی که با بکارگیری $h2$ بسط داده میشود، هرگز بیش از بکارگیری $h1$ نیست

همیشه بهتر است از تابع اکتشافی با **مقادیر بزرگ** استفاده کرد، به

شرطی که **زمان محاسبه اکتشاف**، خیلی بزرگ نباشد



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 محلی

استفاده از حافظه کمکی

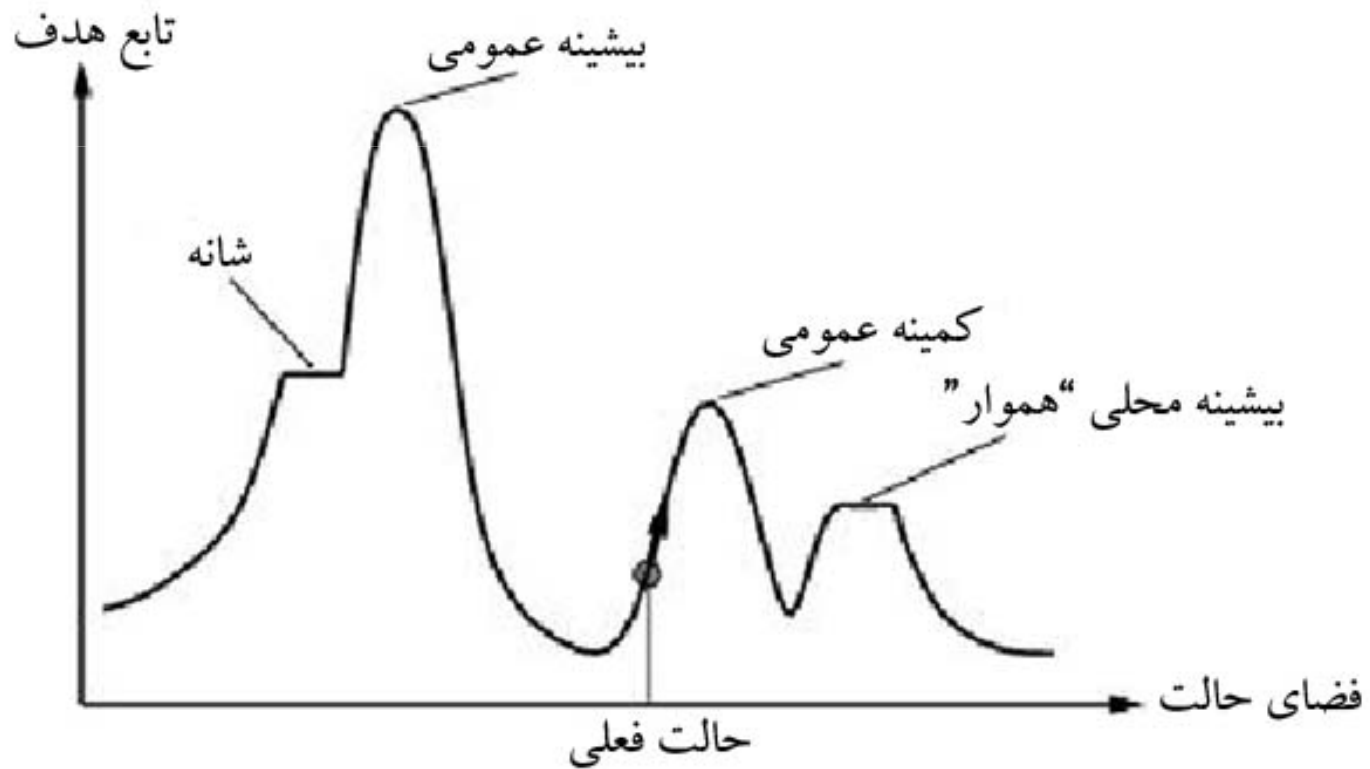
ارائه راه حل های منطقی در فضا های بزرگ و نامتناهی

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 ای که در جهت افزایش مقدار حرکت میکند (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 حریم‌ناهی نیز نام دارد

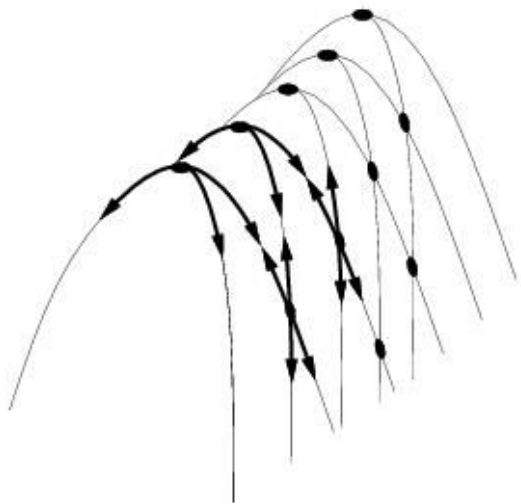
بدون فکر قبلی حالت همسایه خوبی را انتخاب میکند

تپه نوردی به دلایل زیر میتواند متوقف شود:

بیشینه محلی

برآمدگی‌ها

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

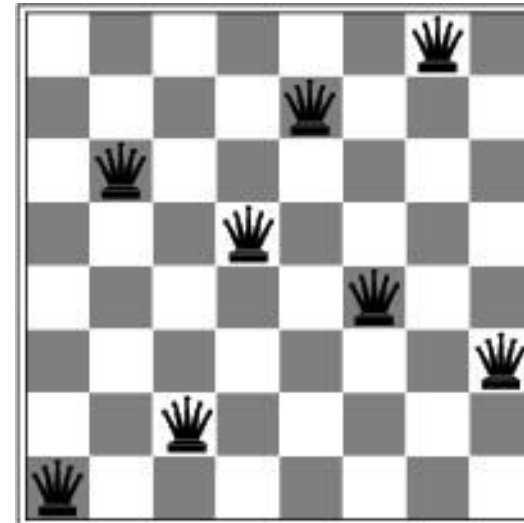
جست و جوی آگاهانه و اکتشاف

مثال جست و جوی تپه نوردی

الف

18	12	14	13	13	12	14	14
14	16	13	15	12	14	12	16
14	12	18	13	15	12	14	14
15	14	14	♙	13	16	13	16
♙	14	17	15	♙	14	16	16
17	♙	16	18	15	♙	15	♙
18	14	♙	15	15	14	♙	16
14	14	13	17	12	14	12	18

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 محلی 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 می‌یابد

جست و جوی آگاهانه و اکتشاف



جست و جوی پرتو محلی

به جای یک حالت k ، حالت را نگهداری میکند

حالت اولیه: k حالت تصادفی

گام بعد: جانشین همه k حالت تولید میشود

اگر یکی 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 فرایندها مبادله میشود

جست و جوی پرتو غیرقطعی

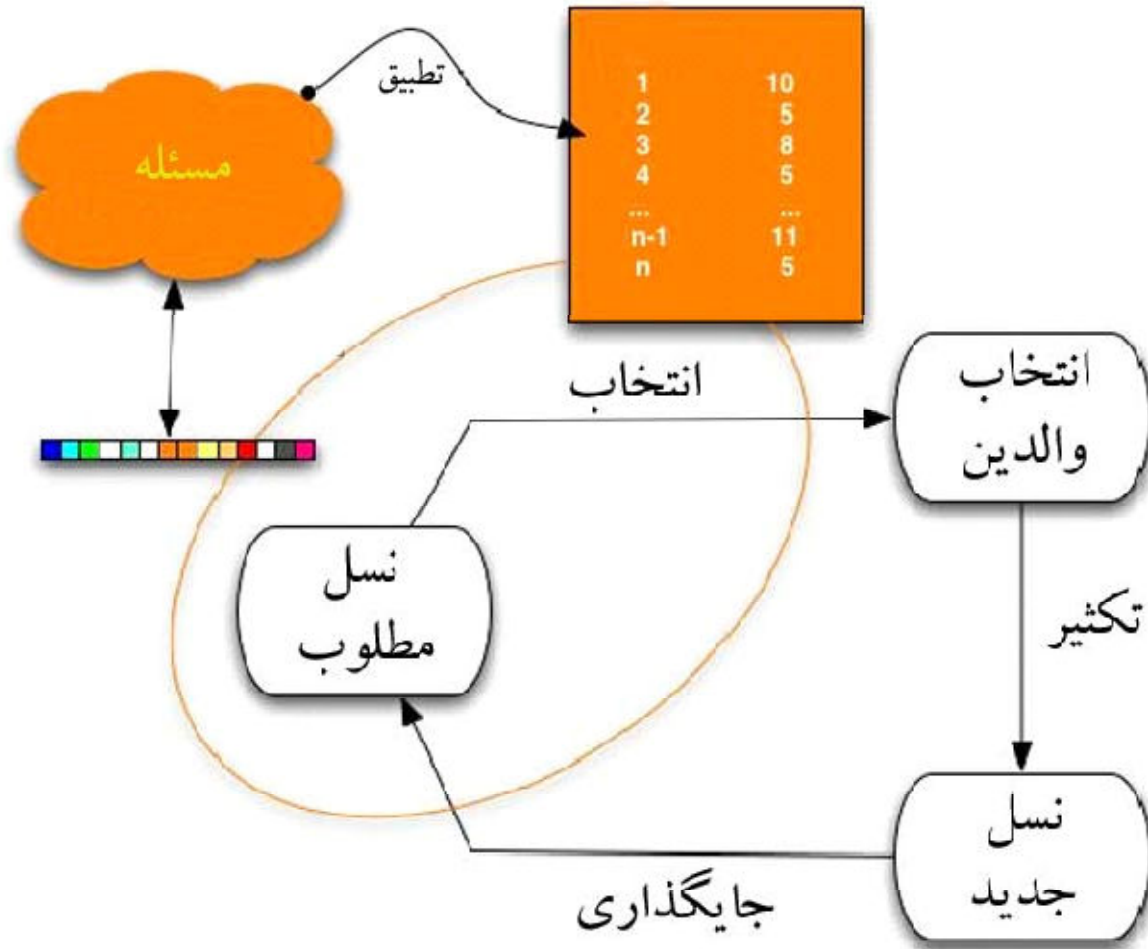
به جای انتخاب بهترین k از جانشینها، k جانشین تصادفی را بطوریکه احتمال انتخاب یکی تابعی صعودی از مقدارش باشد،

انتخاب میکند

جست و جوی آگاهانه و اکتشاف



الگوریتم های ژنتیک

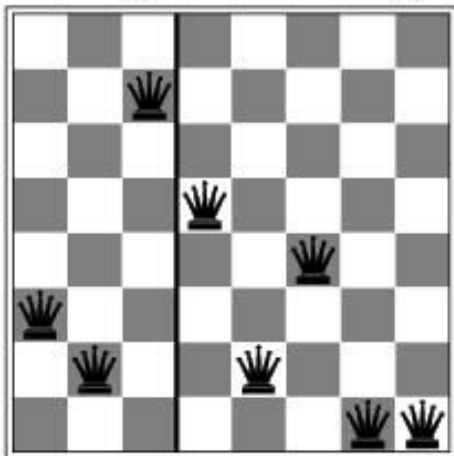
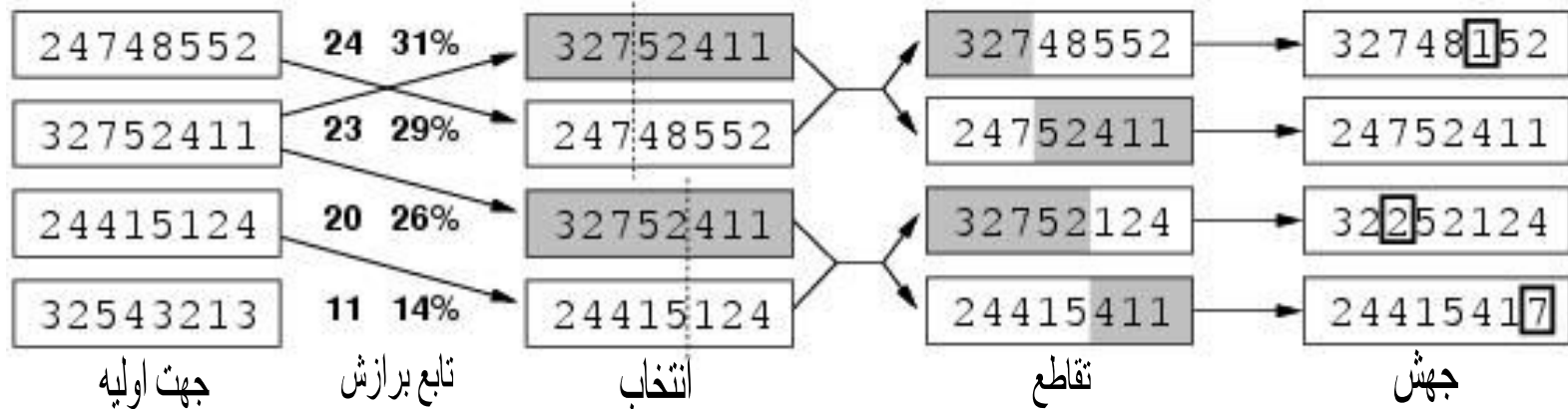


شکلی از جست و جوی پرتو غیر قطعی که حالت‌های جانشین از طریق ترکیب دو حالت والد تولید میشود

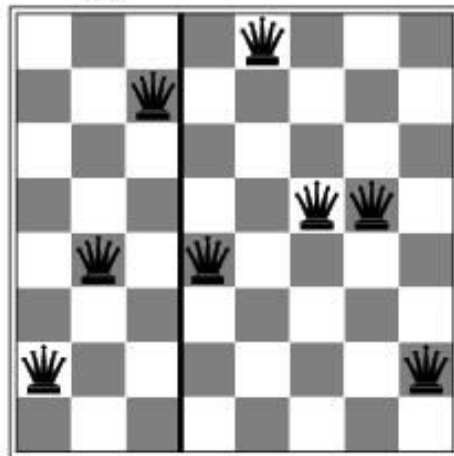
جست و جوی آگاهانه و اکتشاف



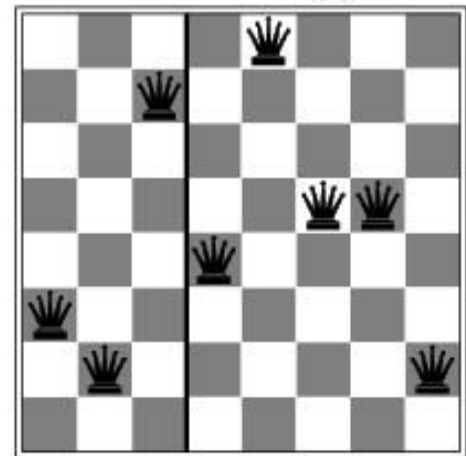
الگوریتم های ژنتیک



+



=



جست و جوی آگاهانه و اکتشاف

جست و جوی محلی در فضاها پیوسته

گسسته در مقابل محیط های پیوسته

در فضاها پیوسته ، تابع جانشین در اغلب موارد ، حالت های نامتناهی را بر میگرداند

حل مسئله:

گسسته کردن همسایه هر حالت

استفاده از شیب منظره

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

روش نیوتن رافسون

جست و جوی آگاهانه و اکتشاف



عواملهای جست و جوی Online و محیطهای ناشناخته

تا به حال همه الگوریتمها برون خطی 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 محیطی 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

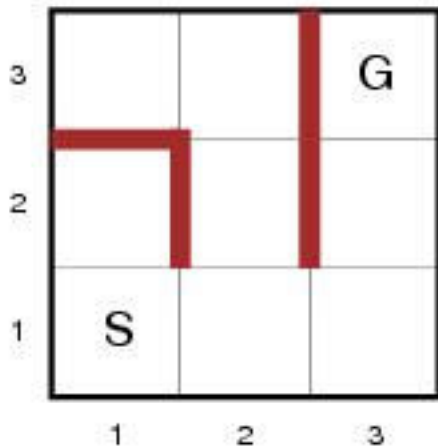
تابع هزینه مرحله ای $C(s, a, s')$: استفاده وقتی که بدانند نتیجه است

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

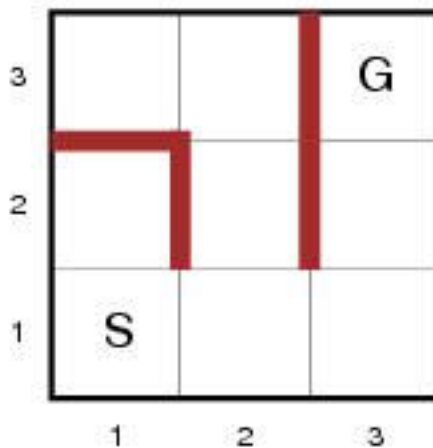
مسئله های جست و جوی Online

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

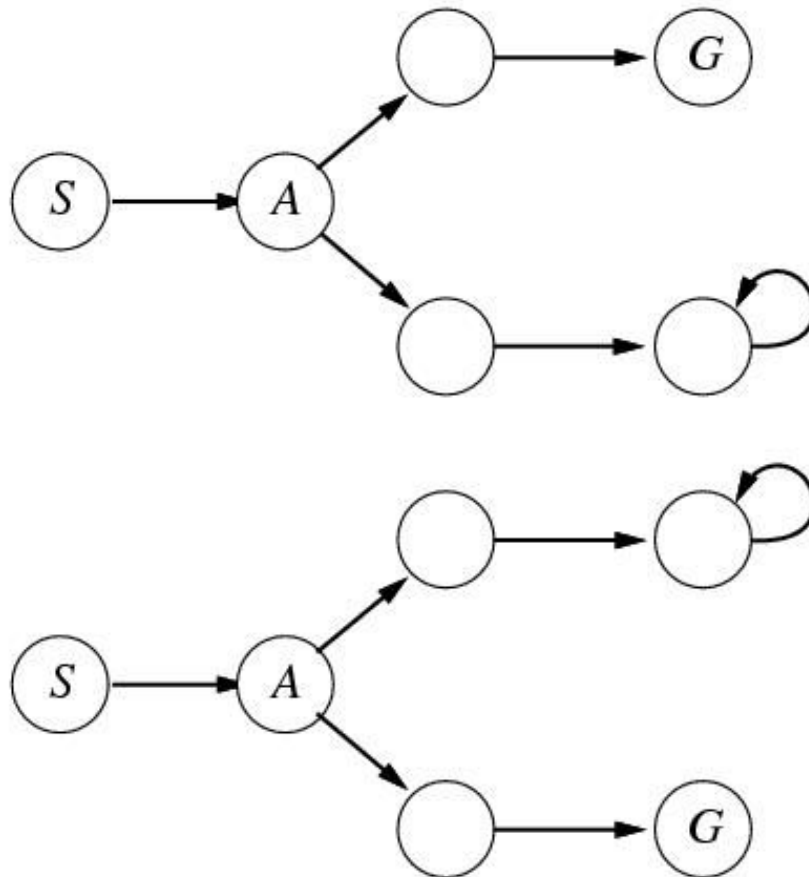


در بعضی موارد ، بهترین نسبت رقابتی نامتناهی است

ممکن است جستجو به یک حالت بن بست برسد که نتوان از طریق آن به هدف رسید



مسئله های جست و جوی Online

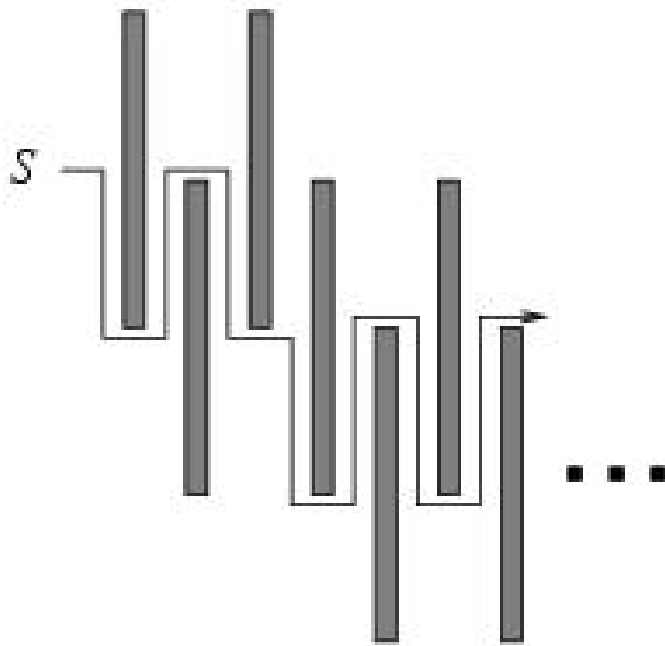


دو فضای حالت که عامل جست و جوی **Online** را به بن بست می‌رسانند. هر عاملی حداقل در یکی از این دو فضا شکست می‌خورد

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



فهرست

↪ ارضای محدودیت چیست؟

↪ جست و جوی عقبگرد برای **CSP**

↪ بررسی پیشرو

↪ پخش محدودیت

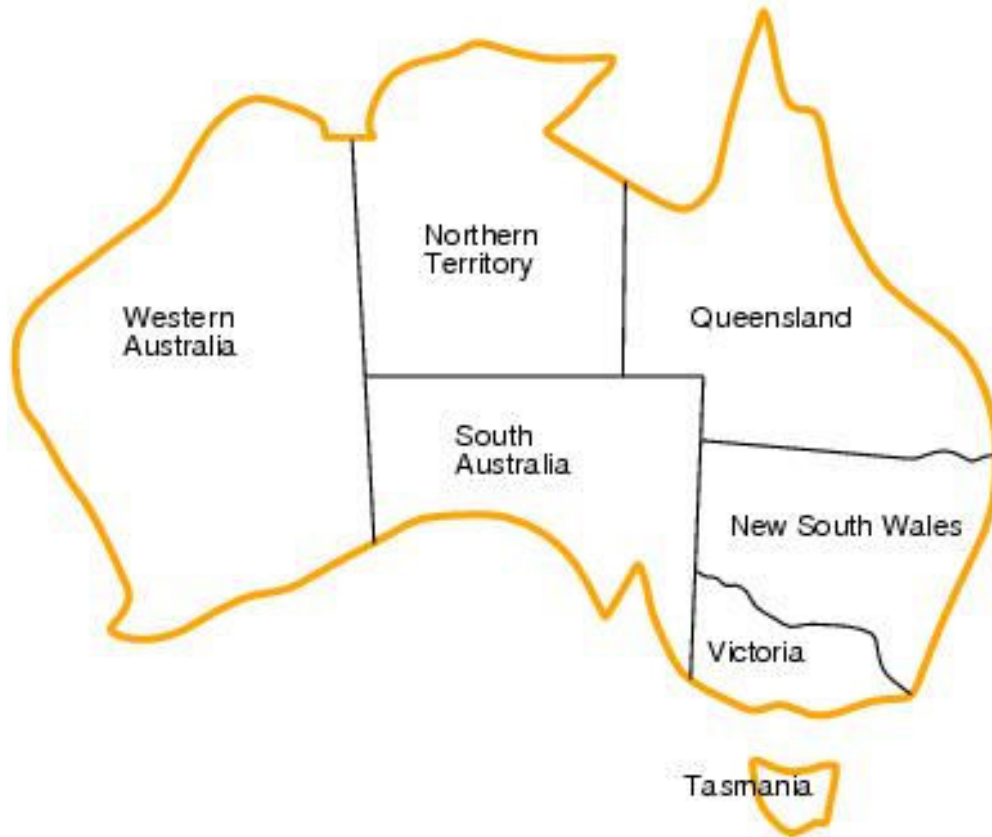
مسائل ارضای محدودیت

ارضای محدودیت (CSP) چیست؟

- مجموعه متناهی از متغیرها: X_1, X_2, \dots, X_n
- مجموعه متناهی از محدودیتها: C_1, C_2, \dots, C_m
- دامنه های ناتهی برای هر یک از متغیرها: $D_{X_1}, D_{X_2}, \dots, D_{X_n}$
- هر محدودیت C_i زیرمجموعه ای از متغیرها و ترکیبهای ممکن 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 CSP ها 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: $D_i = \{\text{قرمز، سبز، آبی}\}$

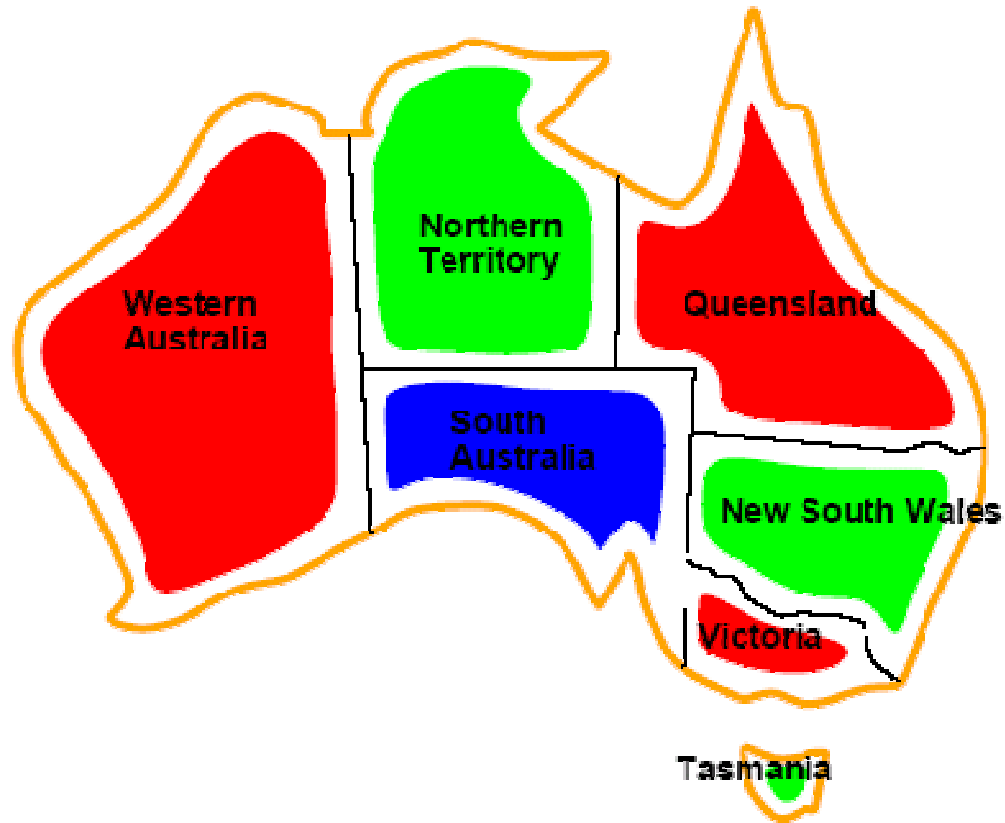
محدودیتها: دو منطقه مجاور، هم‌رنگ نیستند

مثال: $WA \neq NT$ یعنی (WA, NT) عضو

$\{(\text{قرمز، سبز}), (\text{قرمز، آبی}), (\text{سبز، قرمز})\}$

$\{(\text{سبز، آبی}), (\text{آبی، قرمز}), (\text{آبی، سبز})\}$

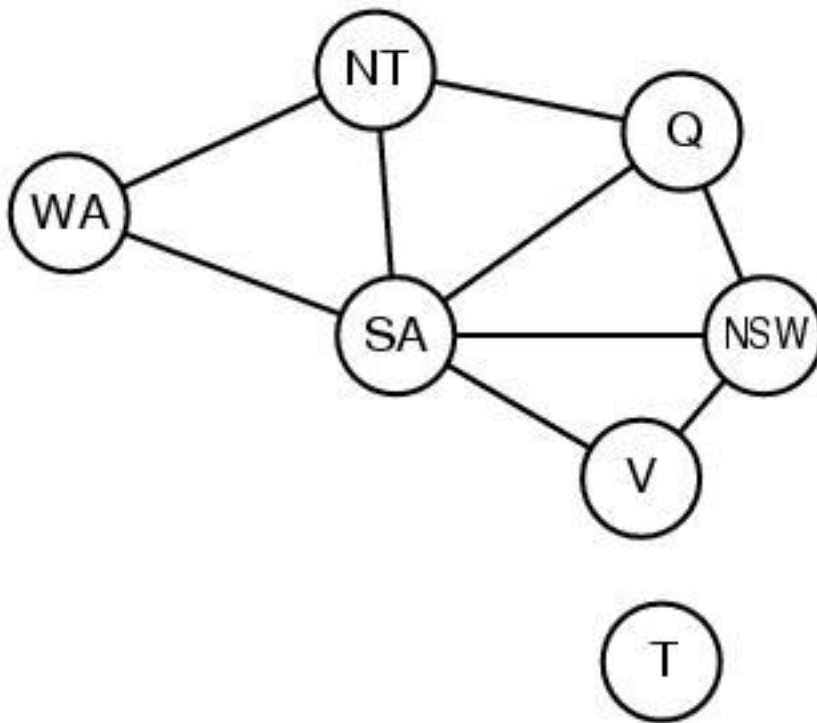
مسائل ارضای محدودیت



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

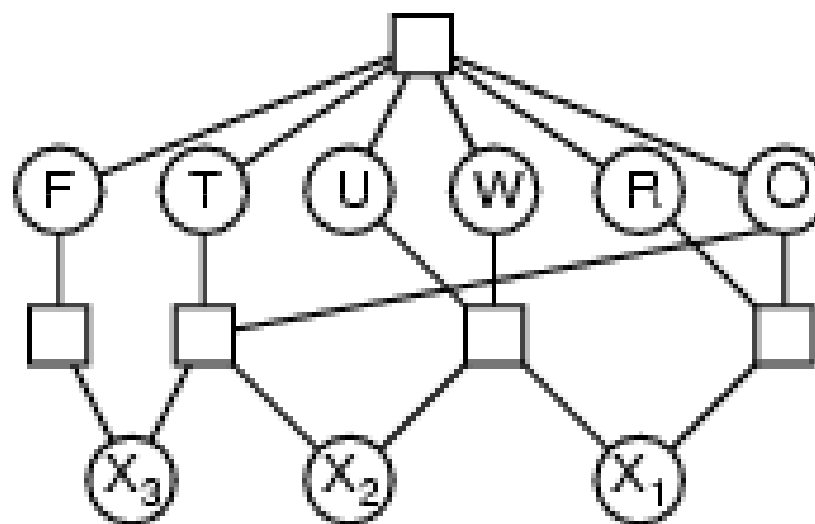
جو بکار میرود

مسائل ارضای محدودیت



مثال CSP: رمزنگاری

$$\begin{array}{r} \text{TWO} \\ + \text{TWO} \\ \hline \text{FOUR} \end{array}$$



متغیرها: $F, T, U, W, R, O, X_1, X_2, X_3$ دامنه: $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

محدودیتها: F, T, U, R, O, W مخالفند - $O + O = R + 10 \cdot X_1 - \dots$

مسائل ارضای محدودیت



↗ نمایش حالتها در **CSP** از الگوی استاندارد پیروی میکند

↗ برای **CSP** میتوان فرمول بندی افزایشی 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 متغیرهایی که قبلا مقدار گرفتند ، متضاد نباشند

↗ آزمون هدف: انتساب فعلی کامل است

↗ هزینه مسیر: هزینه ثابت برای هر مرحله



جست و جوی عقبگرد برای CSP

↩ جست و جوی عمقی

↩ انتخاب مقادیر یک متغیر در هر زمان و عقبگرد در صورت عدم وجود مقداری معتبر
برای انتساب به متغیر

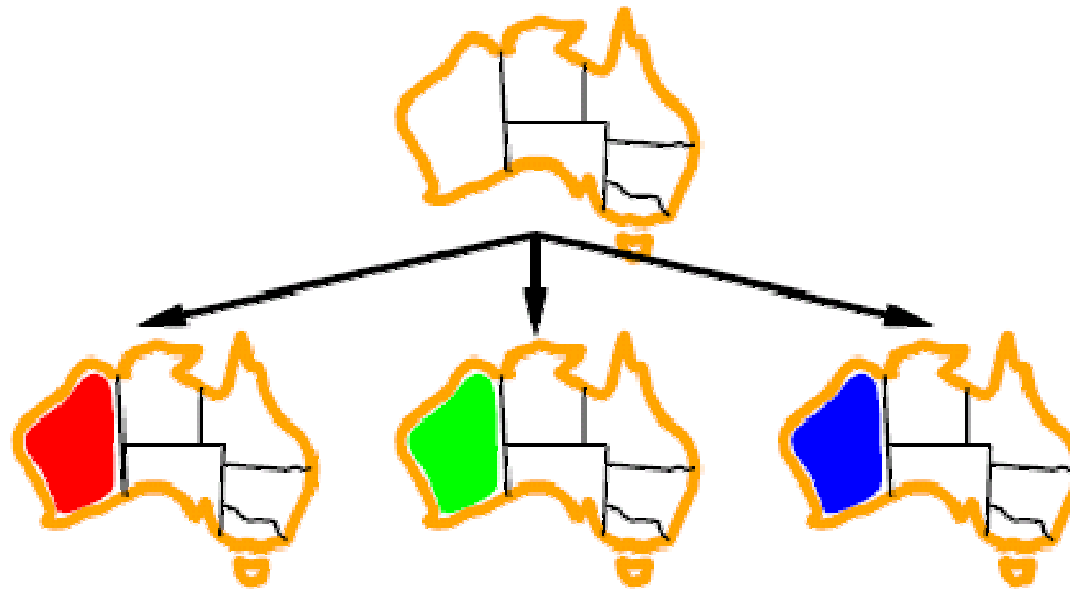
↩ یک الگوریتم ناآگاهانه است
↩ برای مسئله های بزرگ کارآمد نیست

مسائل ارضای محدودیت

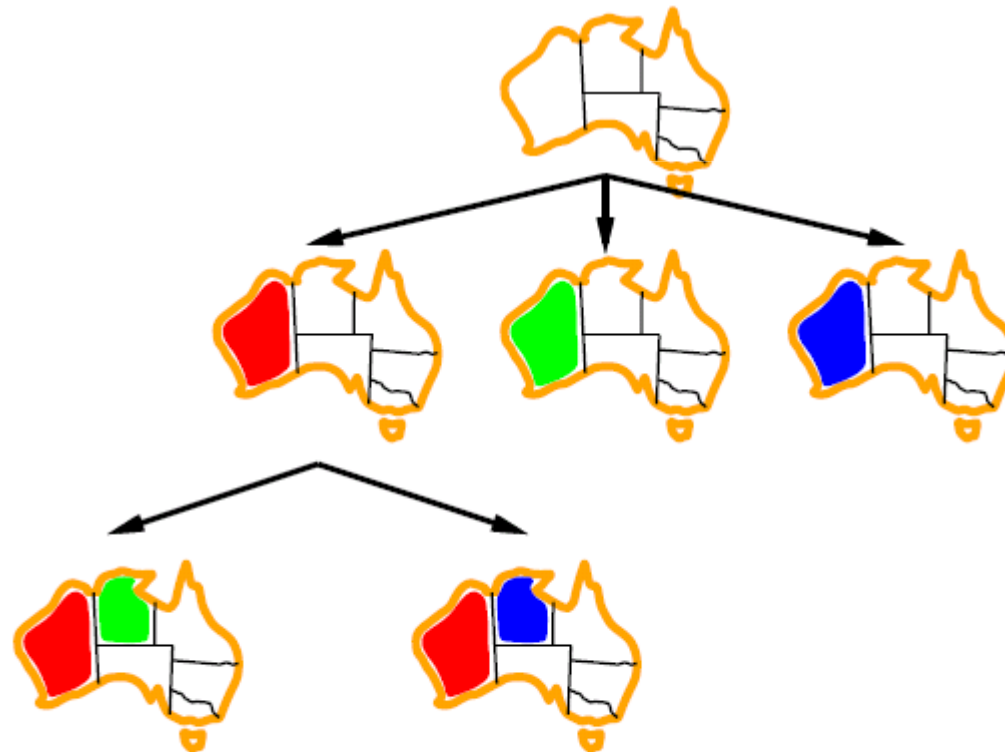
مثال جست و جوی عقبگرد برای CSP



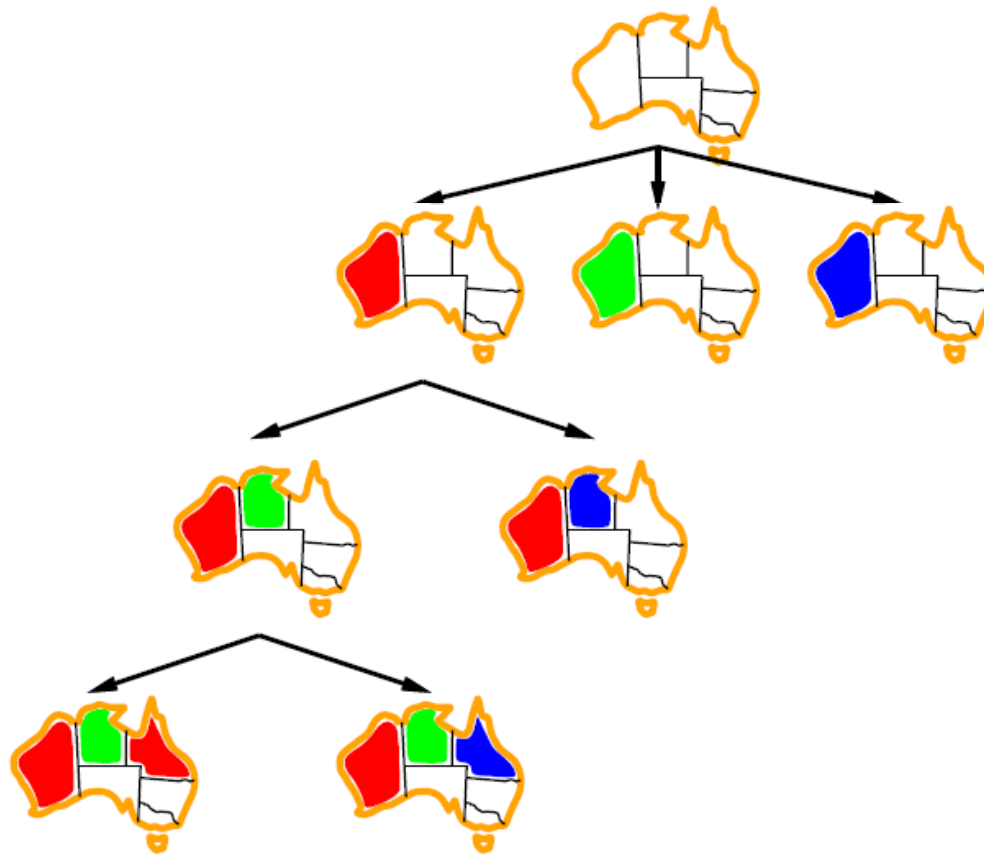
مثال جست و جوی عقبگرد برای CSP



مثال جست و جوی عقبگرد برای CSP

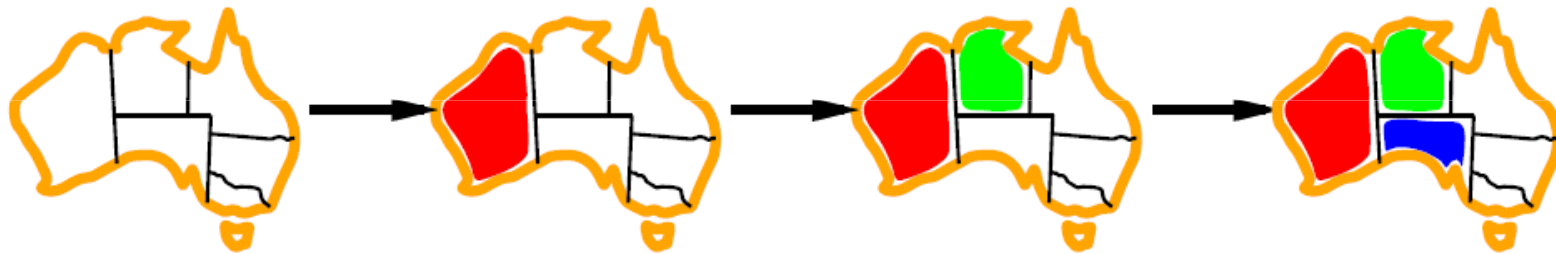


مثال جست و جوی عقبگرد برای CSP



مسائل ارضای محدودیت

مقادیر باقیمانده کمینه (MRV)



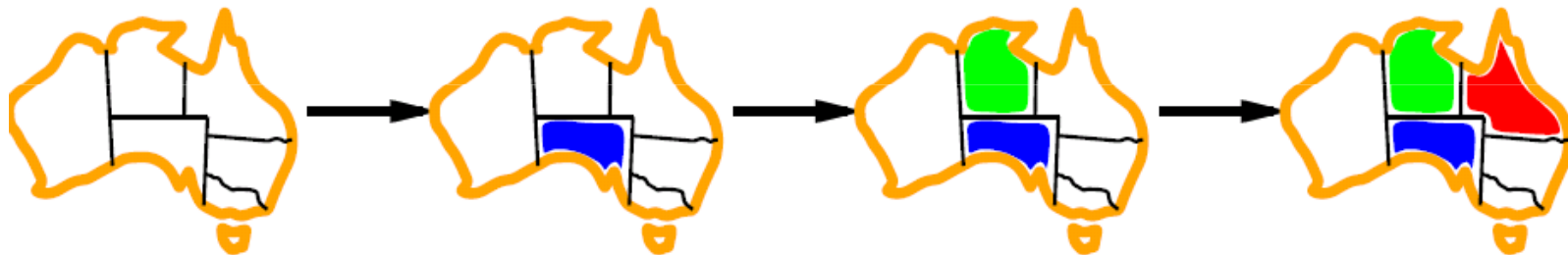
انتخاب متغیری با کمترین مقادیر معتبر

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

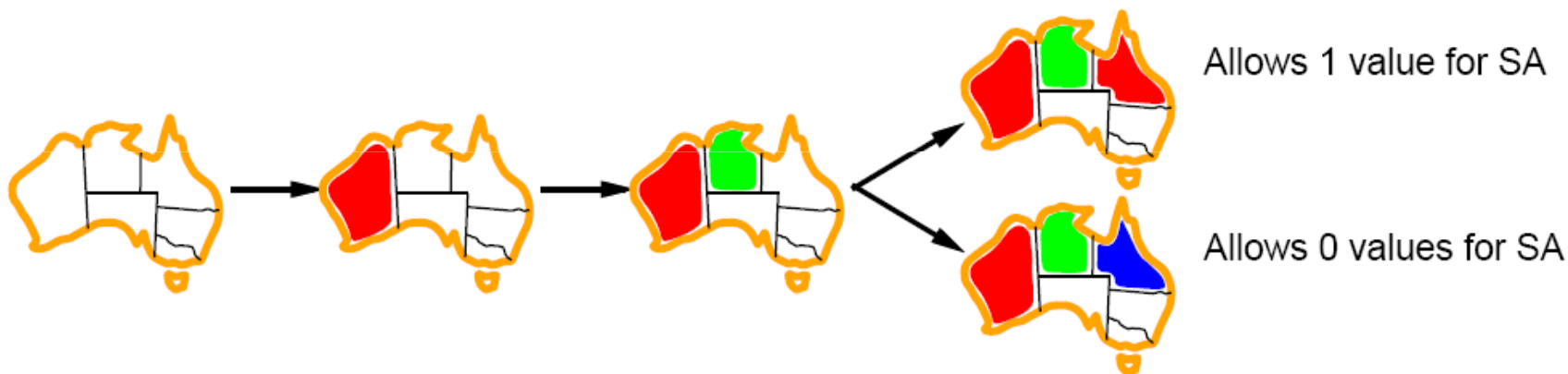


سعی میکند فاکتور انشعاب را در انتخاب آینده کم کند

متغیری انتخاب میکند که در بزرگترین محدودیتهای مربوط به متغیرهای بدون انتساب قرار دارد

مسائل ارضای محدودیت

اکتشاف مقداری باکمترین محدودیت



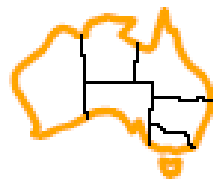
این روش مقداری را ترجیح میدهد که در گراف محدودیت ، متغیرهای همسایه به ندرت آن را انتخاب میکنند

سعی بر ایجاد بیشترین قابلیت انعطاف برای انتساب بعدی متغیرها

مسائل ارضای محدودیت

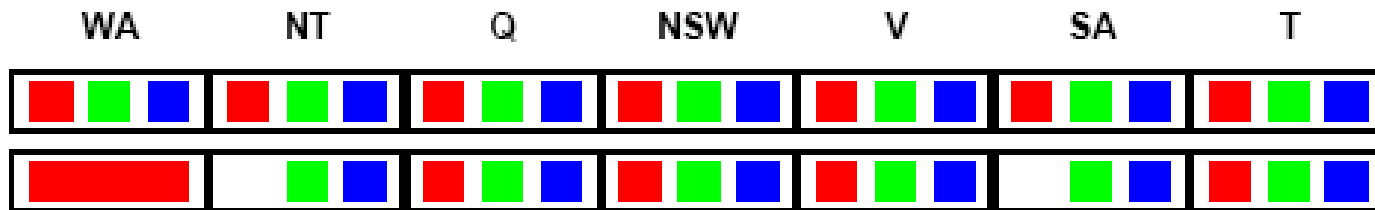
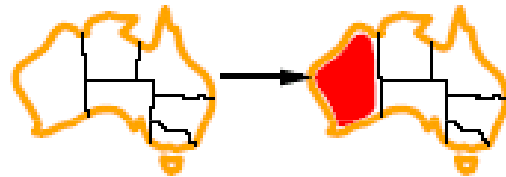
بررسی پیشرو

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



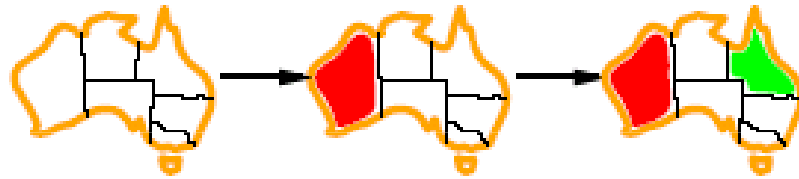
مسائل ارضای محدودیت

بررسی پیشرو



مسائل ارضای محدودیت

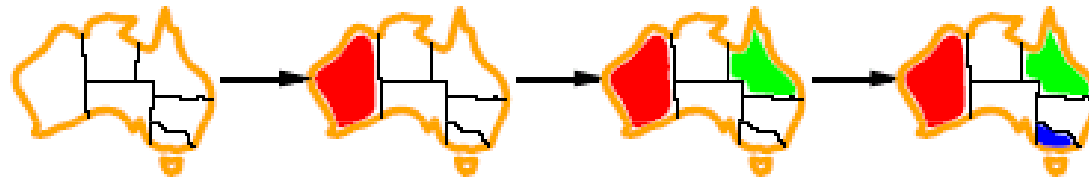
بررسی پیشرو



WA	NT	Q	NSW	V	SA	T		
Red	Green	Blue	Red	Green	Blue	Red	Green	Blue
Red	Green	Blue	Red	Green	Blue	Red	Green	Blue
Red	Green	Blue	Red	Green	Blue	Red	Green	Blue

مسائل ارضای محدودیت

بررسی پیشرو



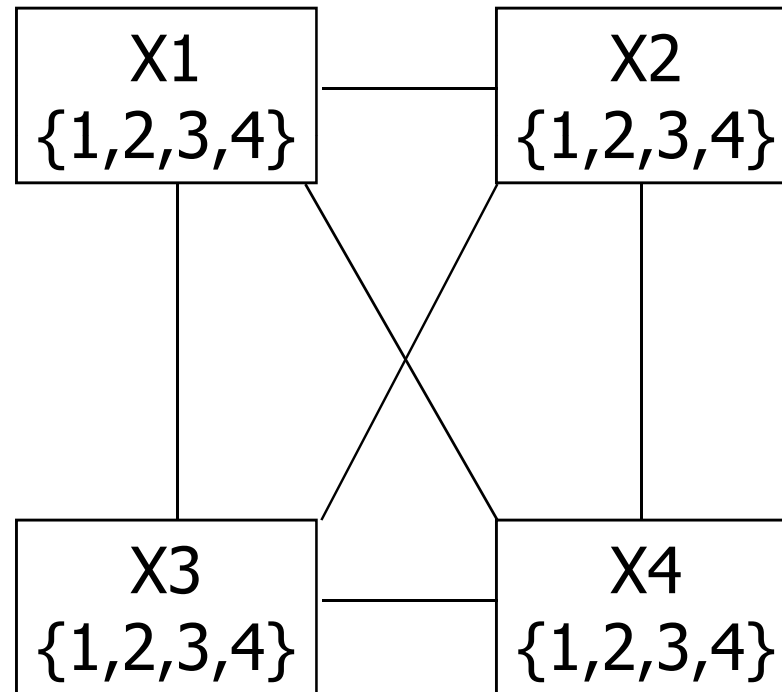
WA	NT	Q	NSW	V	SA	T
Red Green Blue	Red Green Blue	Red Green Blue	Red Green Blue	Red Green Blue	Red Green Blue	Red Green Blue
Red	Green Blue	Red Green Blue	Red Green Blue	Red Green Blue	Green Blue	Red Green Blue
Red	Blue	Green	Red Blue	Red Green Blue	Blue	Red Green Blue
Red	Blue	Green	Red	Blue		Red Green Blue

مسائل ارضای محدودیت



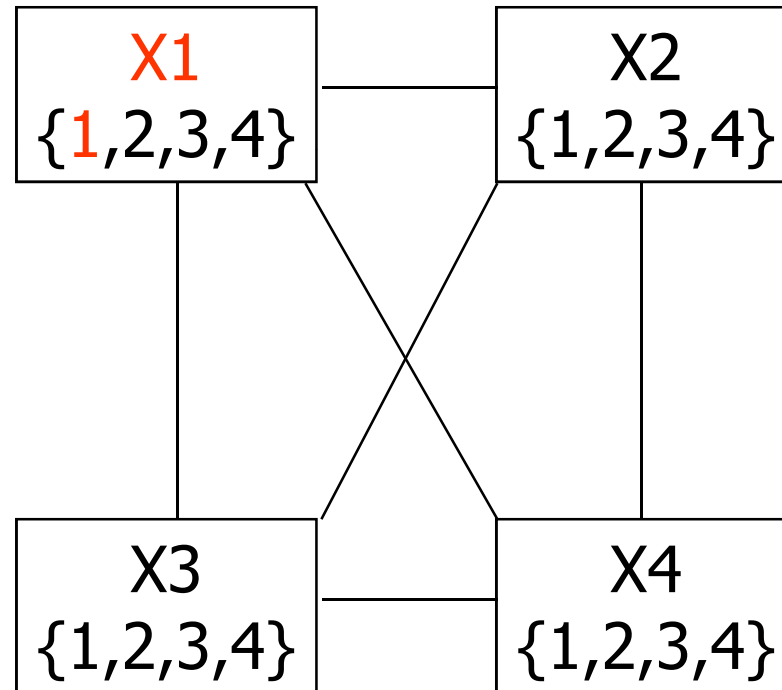
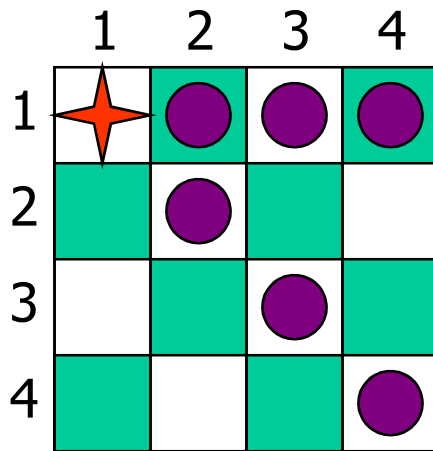
مثال: مسئله 4-وزیر

	1	2	3	4
1				
2				
3				
4				



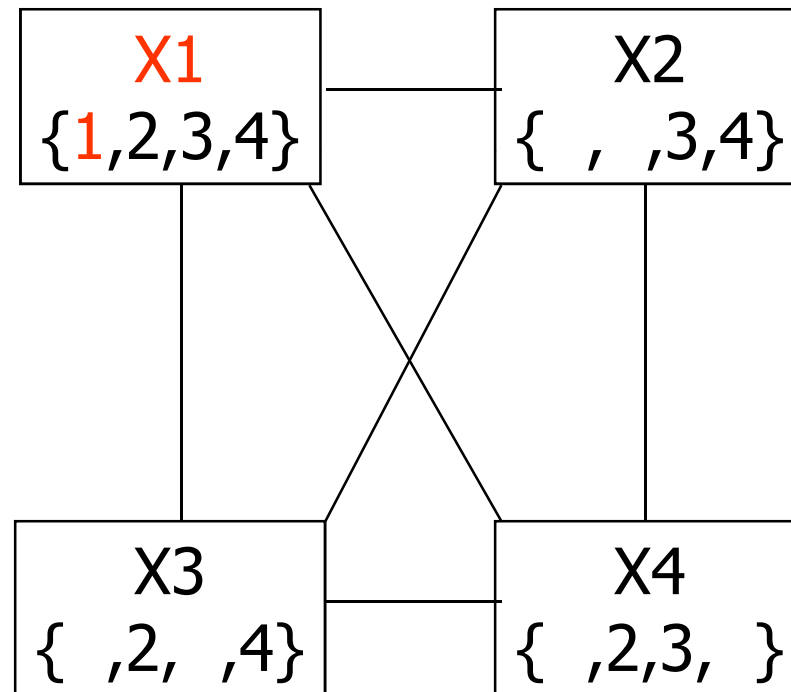
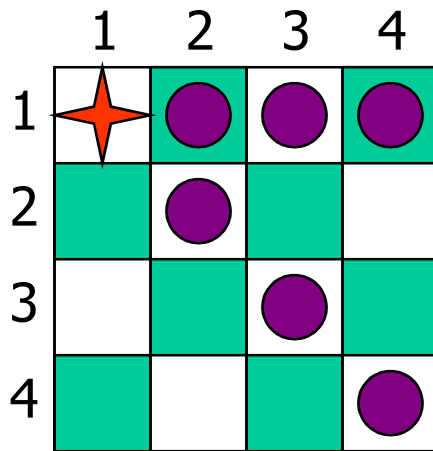
مسائل ارضای محدودیت

مثال: مسئله 4-وزیر



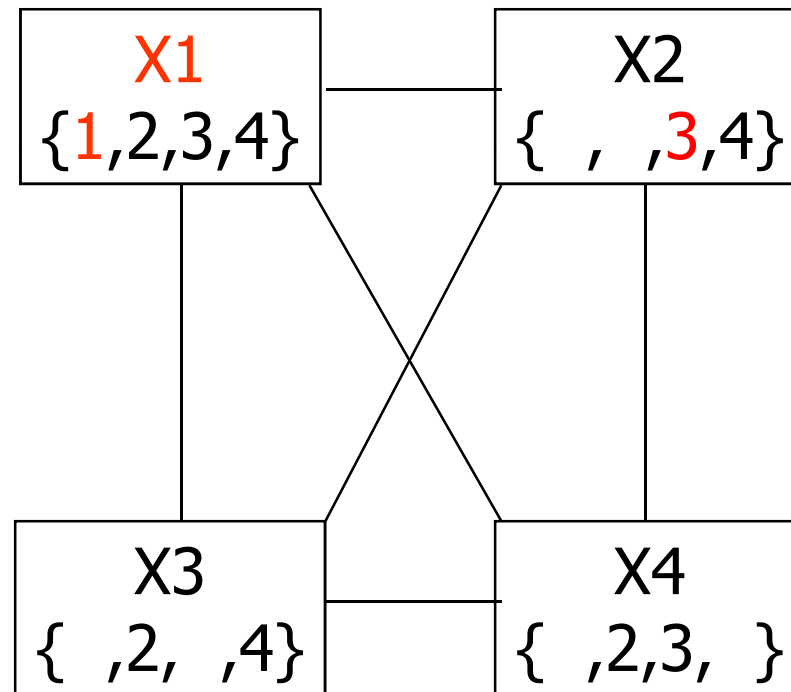
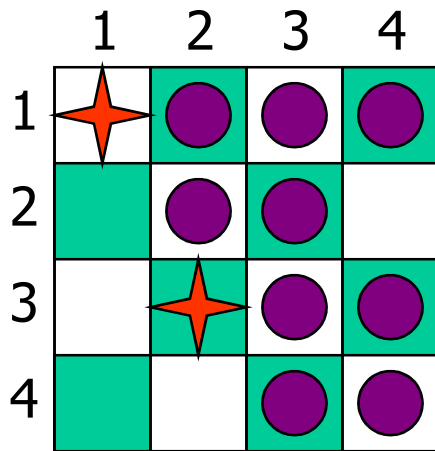
مسائل ارضای محدودیت

مثال: مسئله 4-وزیر



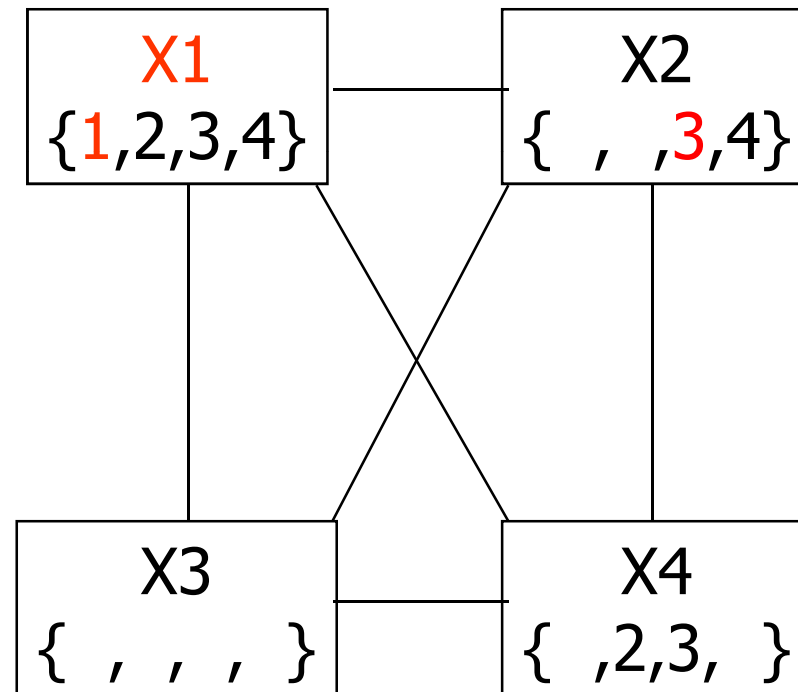
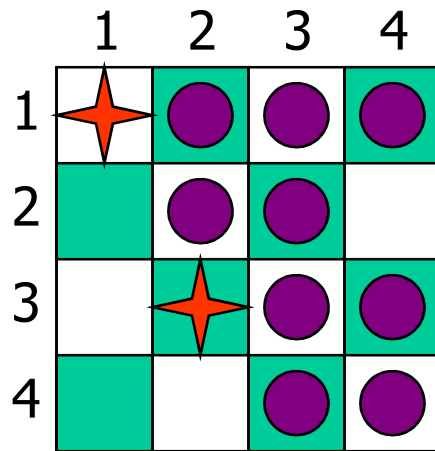
مسائل ارضای محدودیت

مثال: مسئله 4-وزیر



مسائل ارضای محدودیت

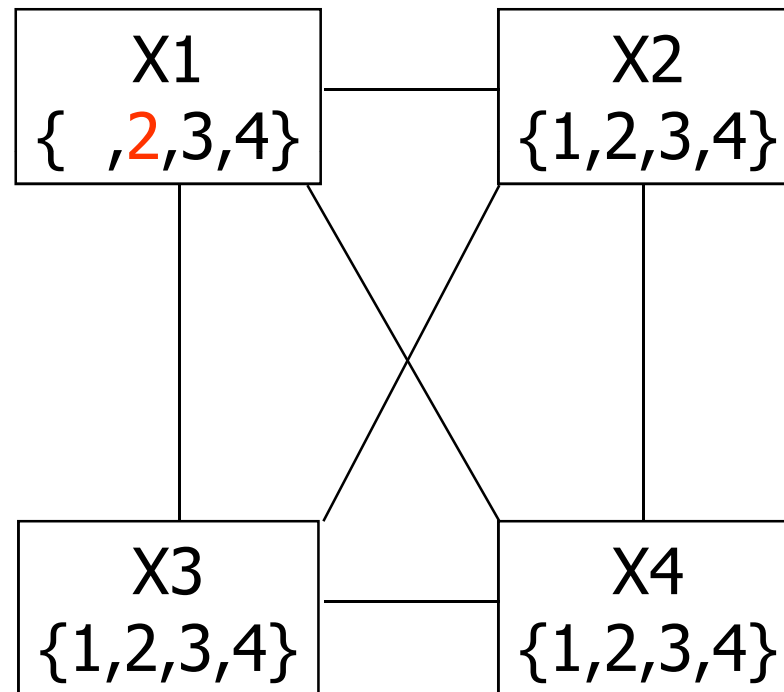
مثال: مسئله 4-وزیر



مسائل ارضای محدودیت

مثال: مسئله 4-وزیر

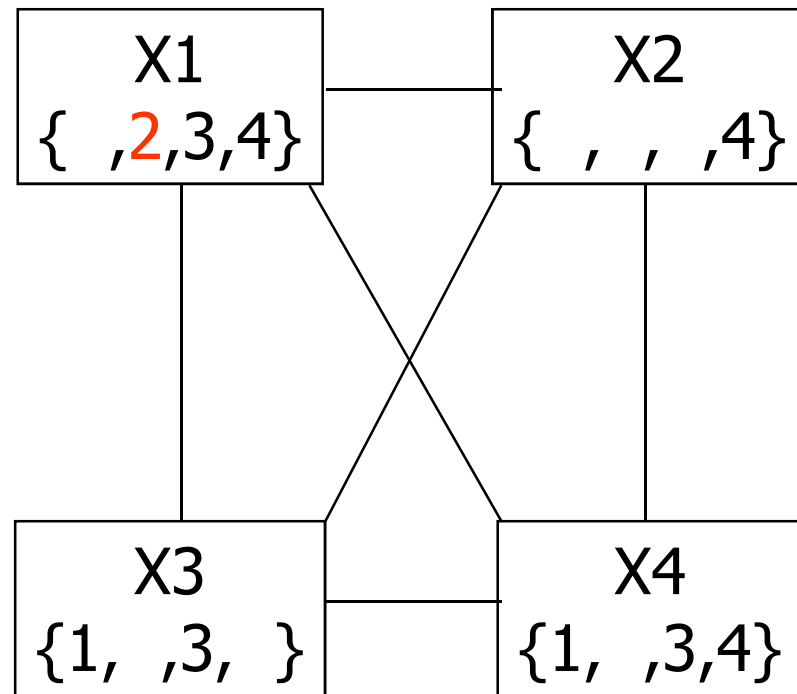
	1	2	3	4
1		●		
2	★	●	●	●
3		●		
4			●	



مسائل ارضای محدودیت

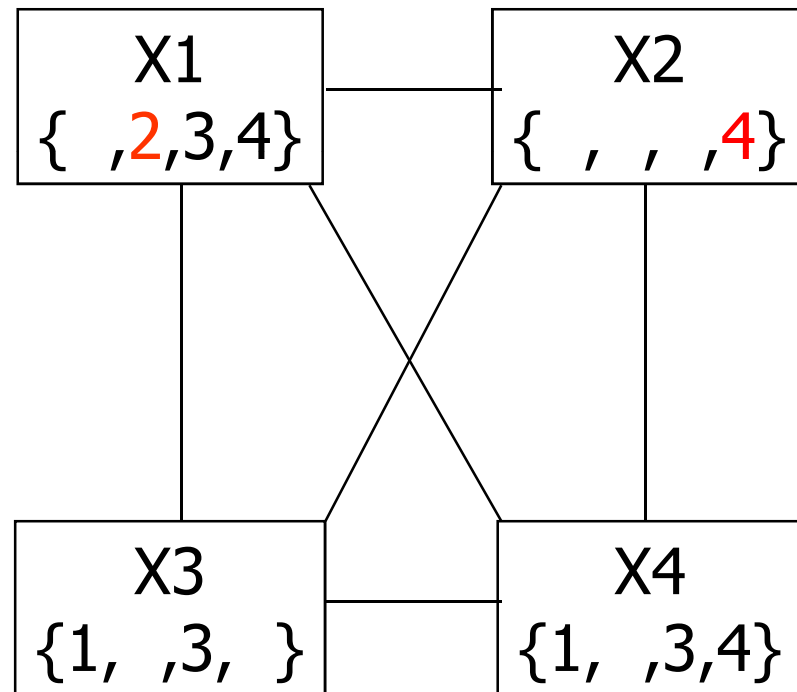
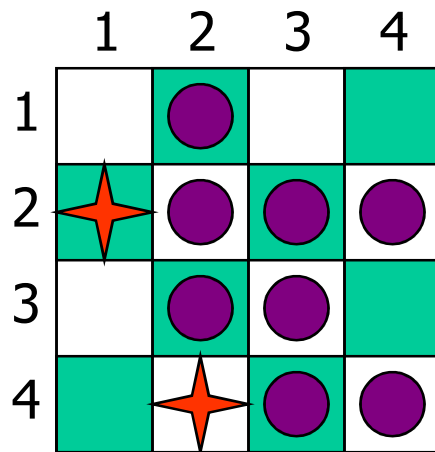
مثال: مسئله 4-وزیر

	1	2	3	4
1		●		
2	★	●	●	●
3		●		
4			●	



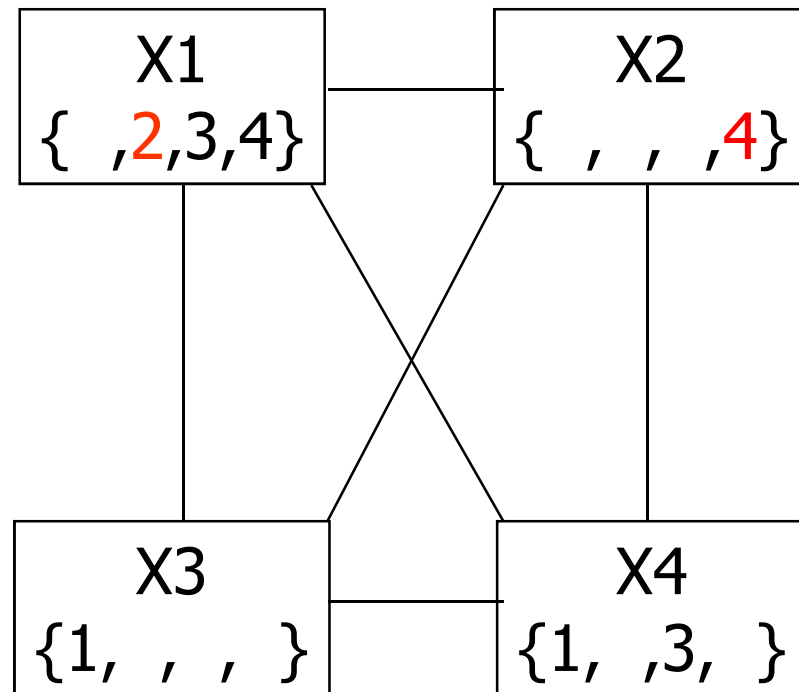
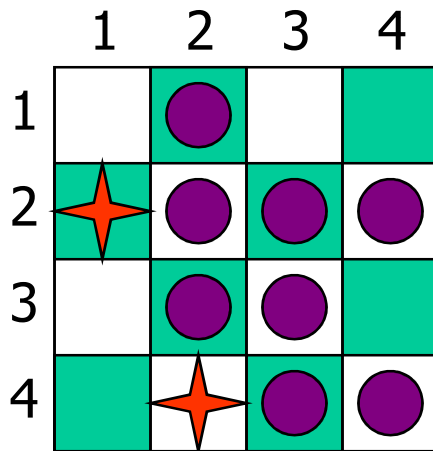
مسائل ارضای محدودیت

مثال: مسئله 4-وزیر



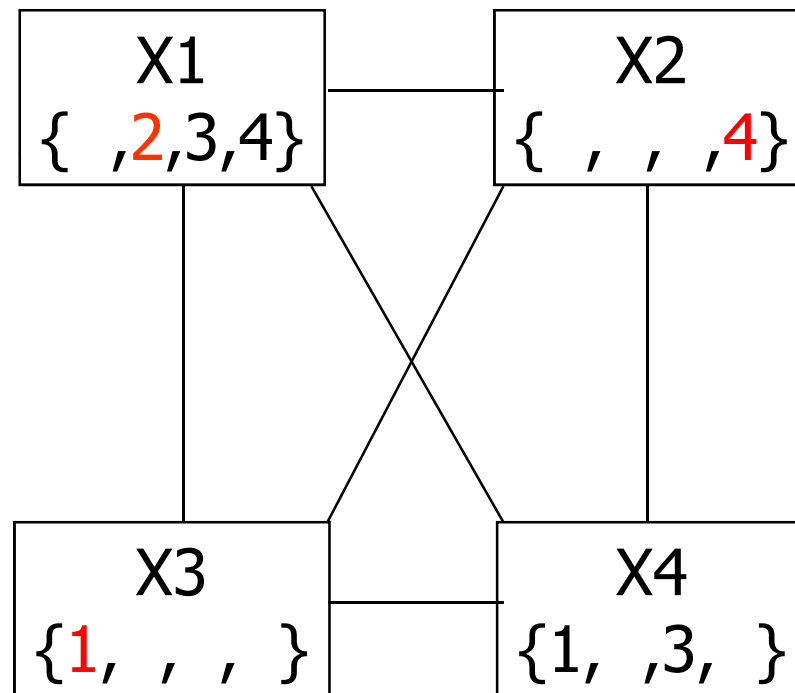
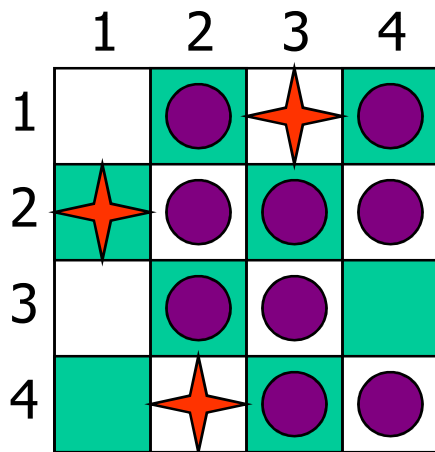
مسائل ارضای محدودیت

مثال: مسئله 4-وزیر



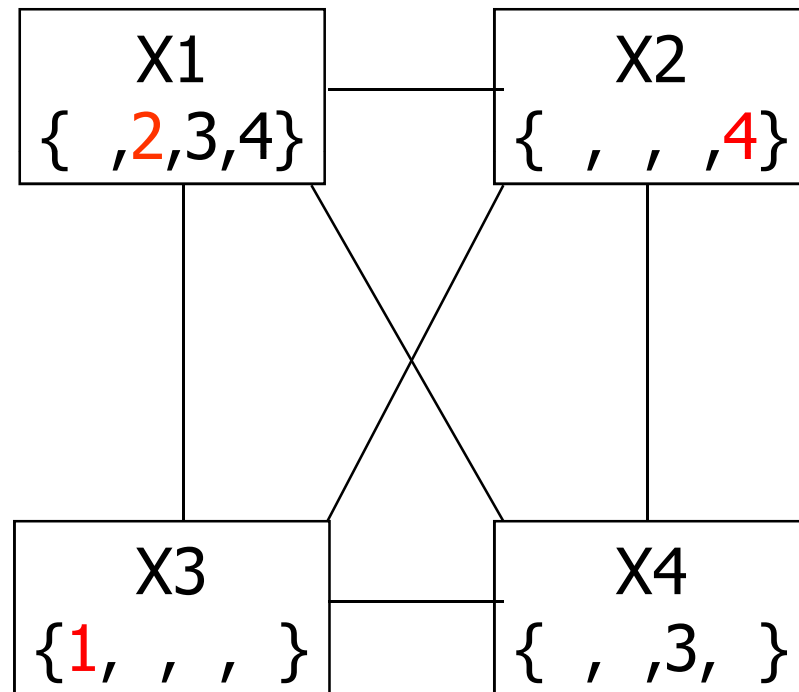
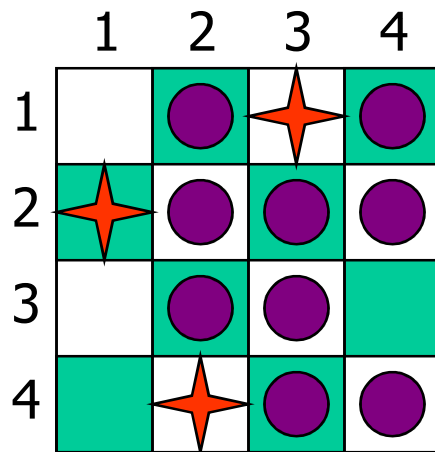
مسائل ارضای محدودیت

مثال: مسئله 4-وزیر



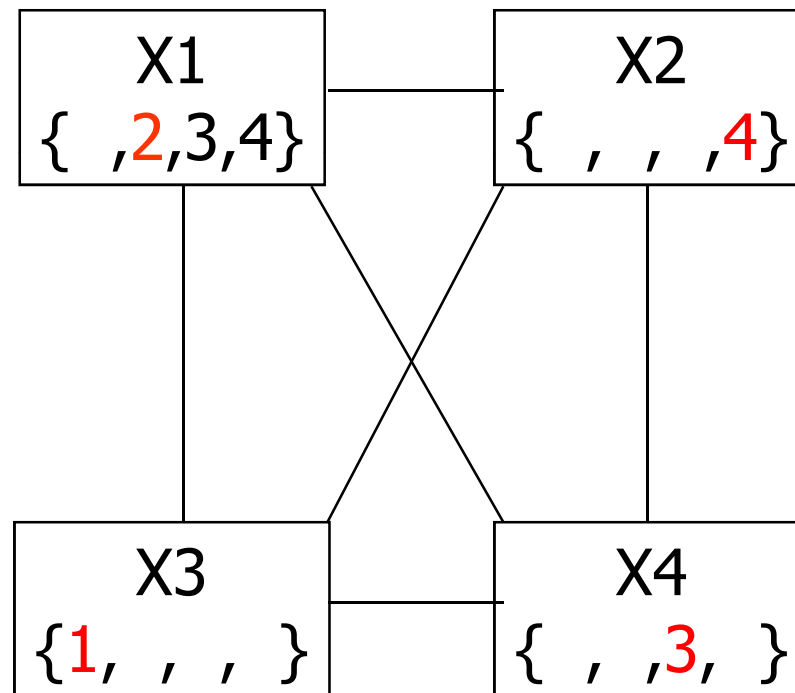
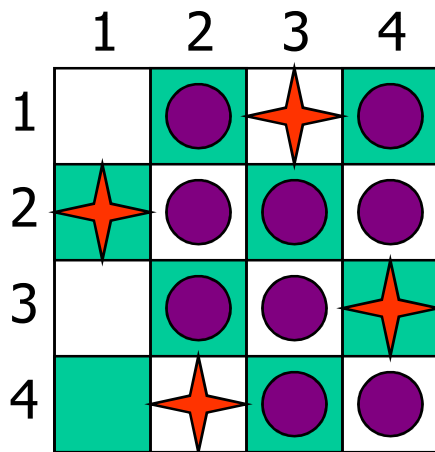
مسائل ارضای محدودیت

مثال: مسئله 4-وزیر



مسائل ارضای محدودیت

مثال: مسئله 4-وزیر

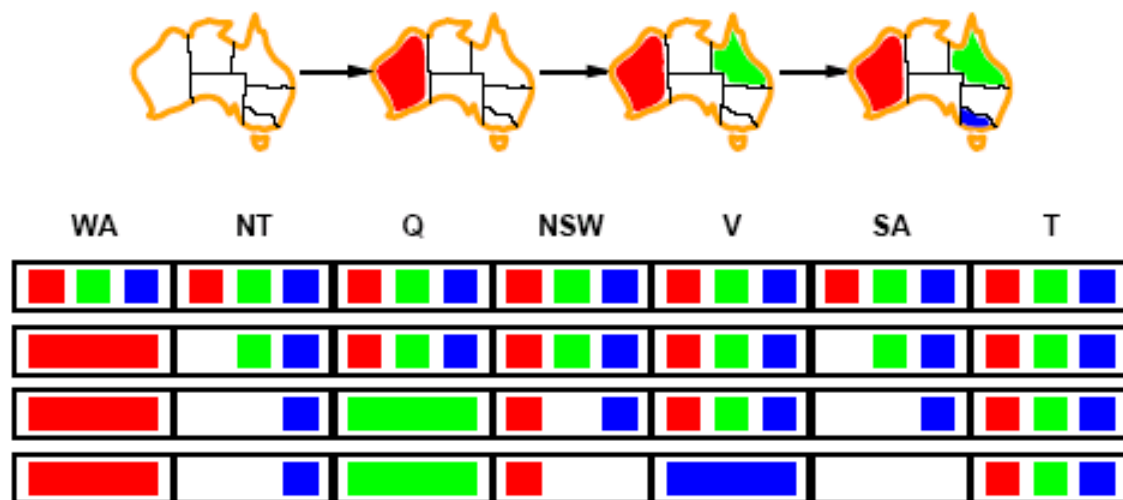


مسائل ارضای محدودیت

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

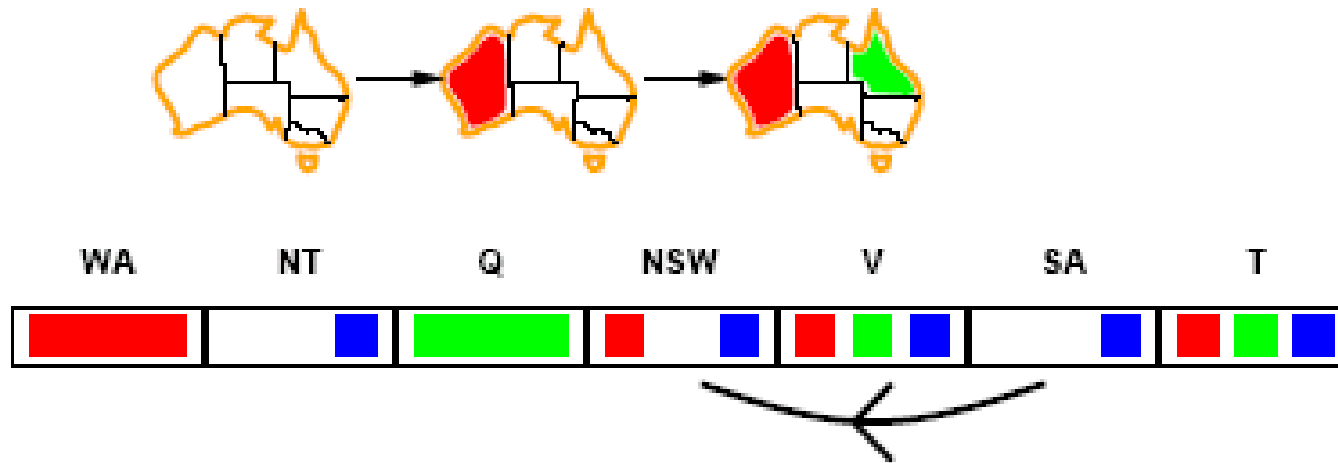
↩️ بررسی سازگاری یال

➤ یک مرحله پیش پردازش ، قبل از شروع جستجو

➤ یک مرحله پخشی پس از هر انتساب در حین جستجو

مسائل ارضای محدودیت

مثال: سازگاری یال

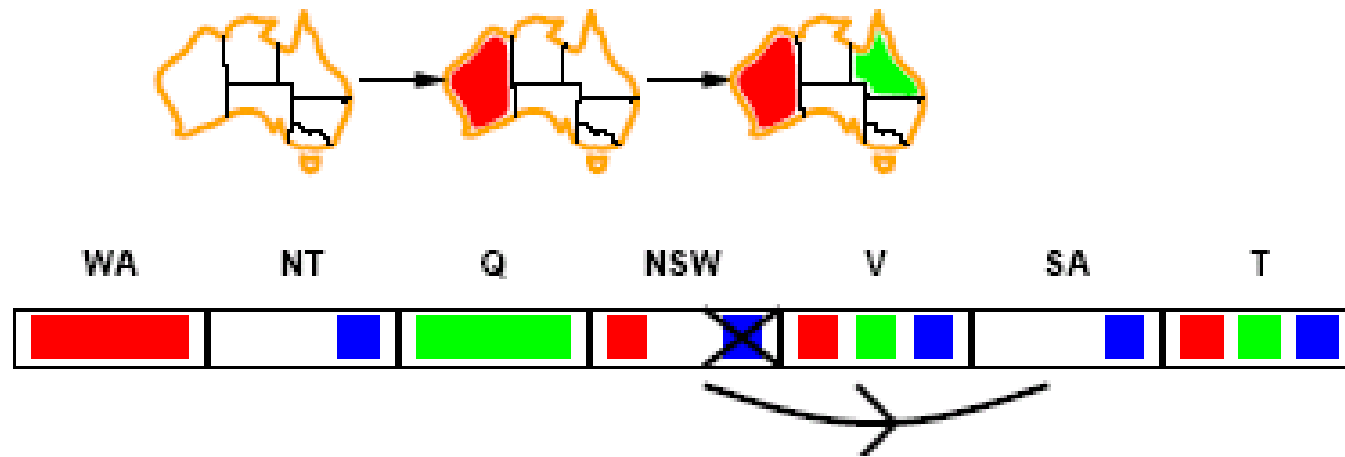


سازگار است اگر $SA \rightarrow NSW$

$SA=blue$ and $NSW=red$

مسائل ارضای محدودیت

مثال: سازگاری یال



■ NSW → SA سازگار است اگر

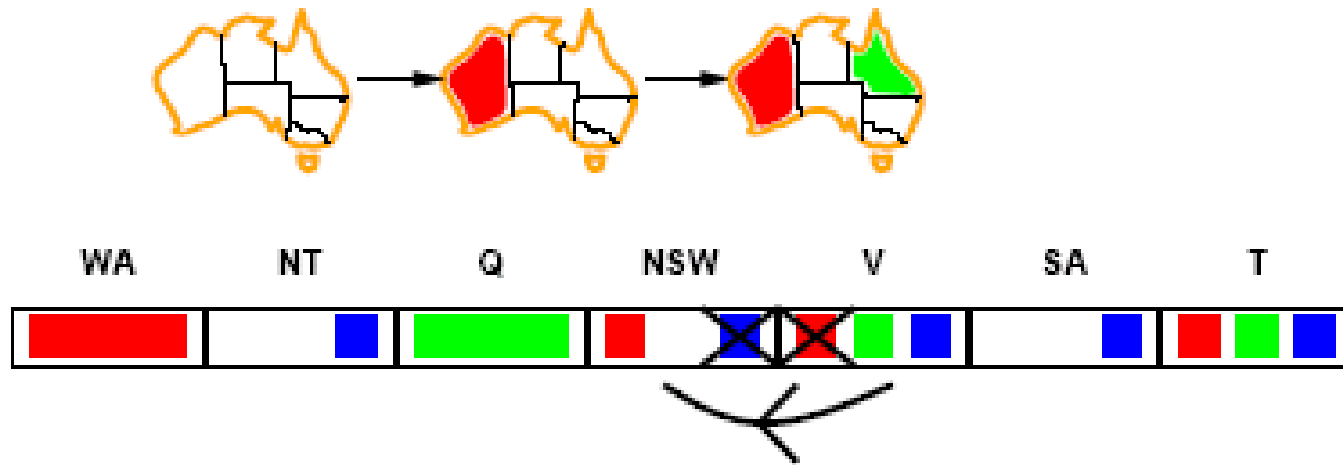
SA=blue and NSW=red

NSW=blue and SA=???

■ 169 یال میتواند سازگار شود با حذف blue از NSW

مسائل ارضای محدودیت

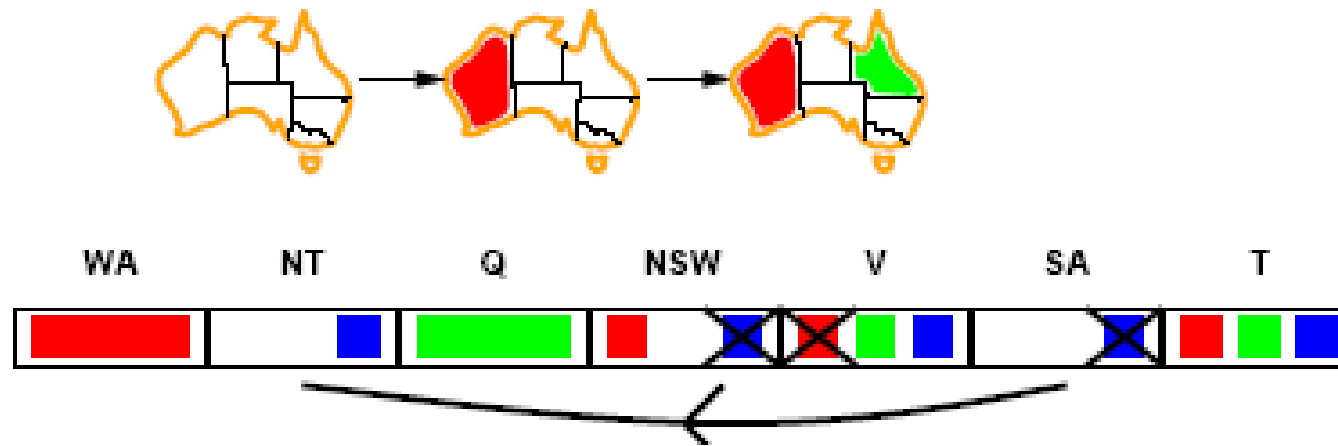
مثال: سازگاری یال



■ 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



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 k ام نسبت داده شود

مسائل ارضای محدودیت



سازگاری K

بطور مثال:

- سازگاری 1: هر متغیر با خودش سازگار است (سازگاری گره)
- سازگاری 2: مشابه سازگاری یال
- سازگاری k: بسط هر جفت از متغیرهای همجوار به سومین متغیر همسایه (سازگاری مسیر)

گراف در صورتی قویا سازگار K است که:

- سازگار k باشد
- همچنین سازگار k-1 و سازگار k-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

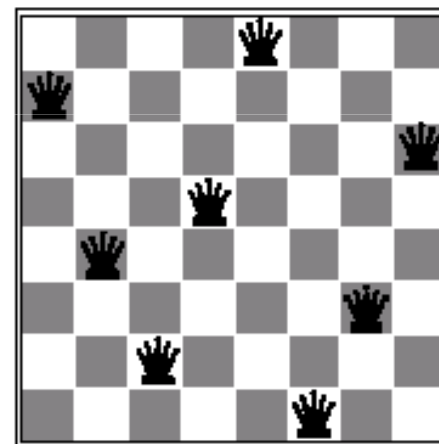
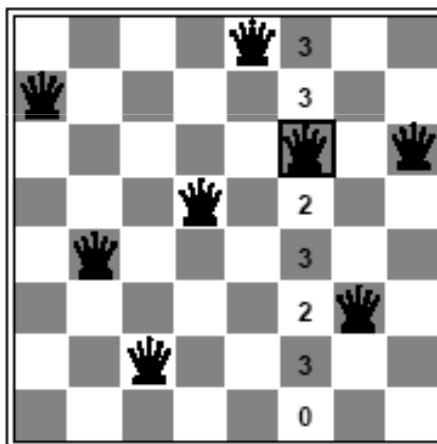
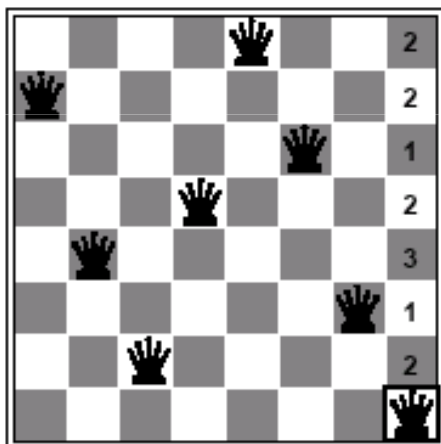
- انتخاب مقداری که کمترین برخورد را با متغیرهای دیگر ایجاد کند (اکتشاف برخورد کم)
- زمان اجرای برخورد کم مستقل از اندازه مسئله است
- برخورد کم ، برای مسئله های سخت نیز کار میکند

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 8 وزیر 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 مربعی با کمترین برخورد انتقال میدهد ، بطوریکه گره ها را بطور تصادفی میشکنند



هوش مصنوعي

فصل ششم

جستجوی خصمانه



فهرست

بازیها چیستند و چرا مطالعه میشوند؟

انواع بازیها

الگوریتم **minimax**

بازیهای چند نفره

هرس آلفا-بتا

بازیهای قطعی با اطلاعات ناقص

بازیهایی که حاوی عنصر شانس هستند

جستجوی خصمانه



بازی‌ها چیستند و چرا مطالعه میشوند؟

بازیها حالتی از محیطهای چند عاملی هستند

- هر عامل نیاز به در نظر گرفتن سایر عاملها و چگونگی تأثیر آنها دارد
- تمایز بین محیطهای چند عامل رقابتی و همکار
- محیطهای رقابتی، که در آنها اهداف عاملها با یکدیگر برخورد دارند، منجر به مسئله‌های خصمانه میشود که به عنوان بازی شناخته میشوند

چرا مطالعه میشوند؟

- قابلیت‌های هوشمندی انسانها را به کار میگیرند
- ماهیت انتزاعی بازی‌ها
- حالت بازی را به راحتی میتوان نمایش داد و عاملها معمولاً به مجموعه کوچکی از فعالیتها محدود هستند که نتایج آنها با قوانین دقیقی تعریف شده اند

جستجوی خصمانه



انواع بازی ها

	قطعی	تصادفی
اطلاعات کامل	شطرنج ریورسی	تخته نرد
اطلاعات ناقص		پوکر



یک نمونه بازی

بازی دو نفره: Max و Min

- 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:

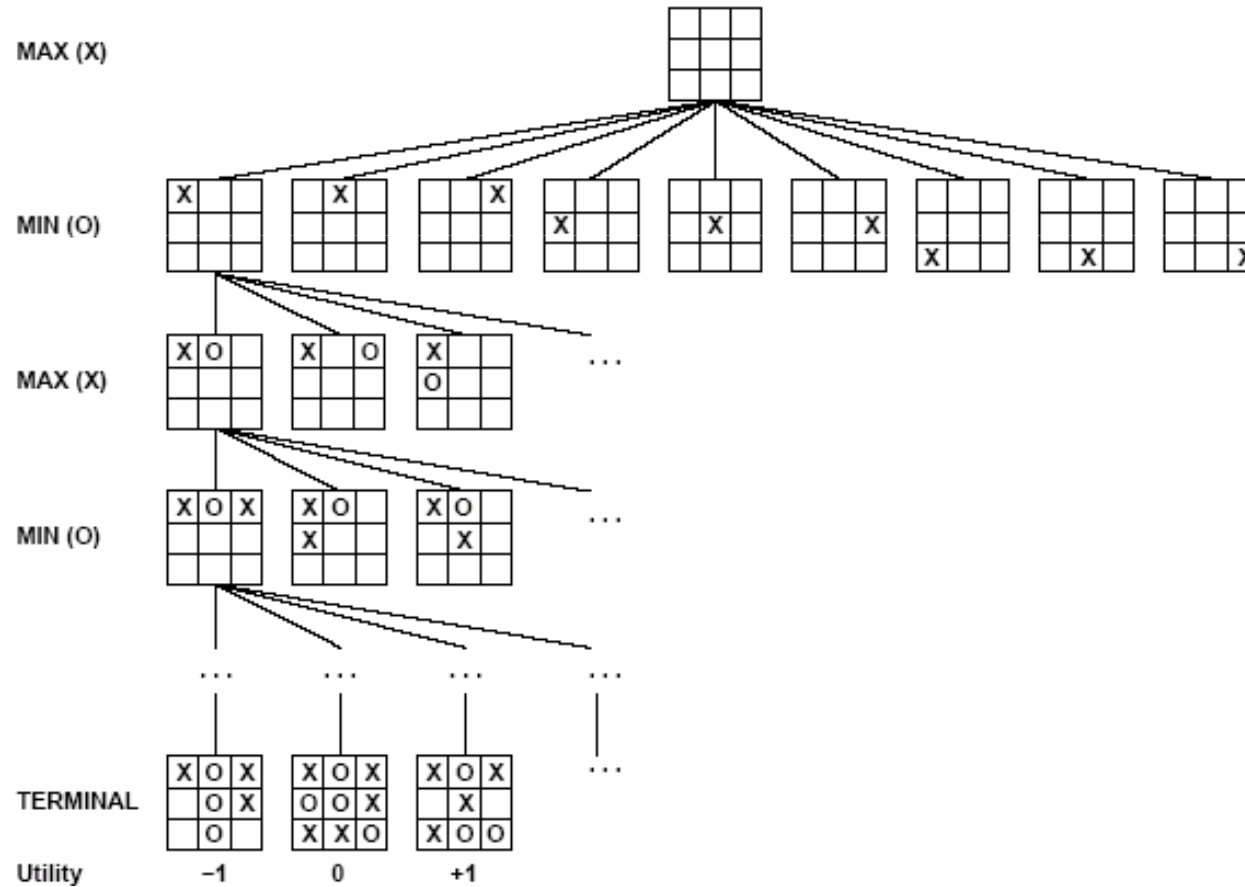
- حالت اولیه: موقعیت صفحه و شناسه های قابل حرکت
- تابع جانشین: لیستی از (حالت, حرکت) که معرف یک حرکت معتبر است
- آزمون هدف: پایان بازی چه موقع است؟ (حالت‌های پایانه)
- تابع سودمندی: برای هر حالت پایانه یک مقدار عددی را ارائه میکند. مثلاً برنده (+1) و بازنده (-1)

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

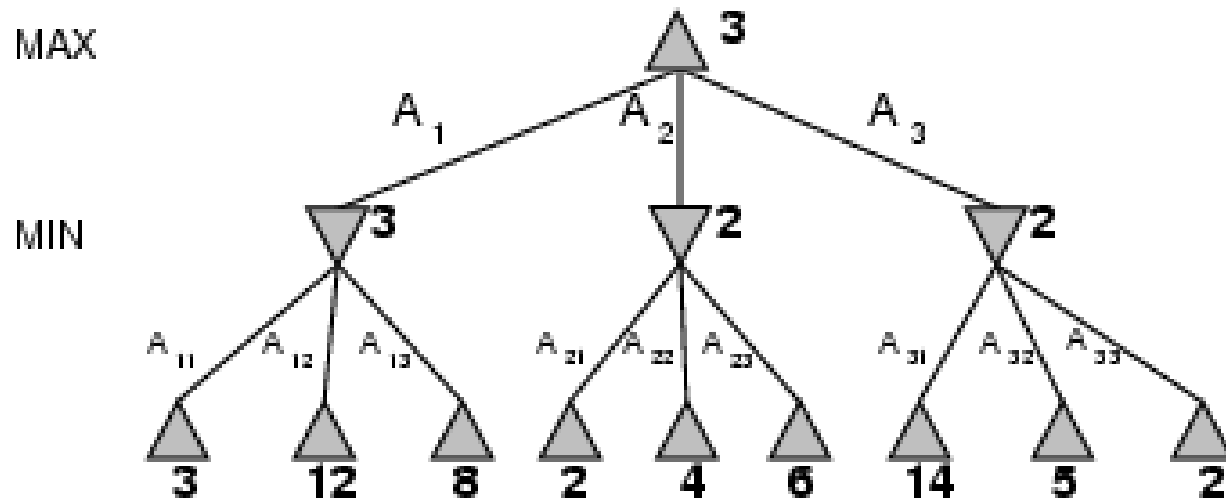
برای خودش یا بدترین وضعیت

برای بازیکن

بازیکن: ماکزیمم حالت

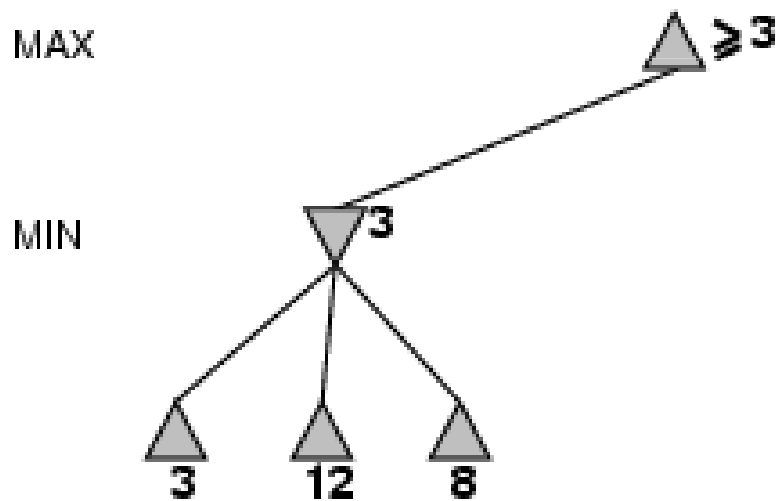
حریف: مینیمم حالت

الگوریتم minimax



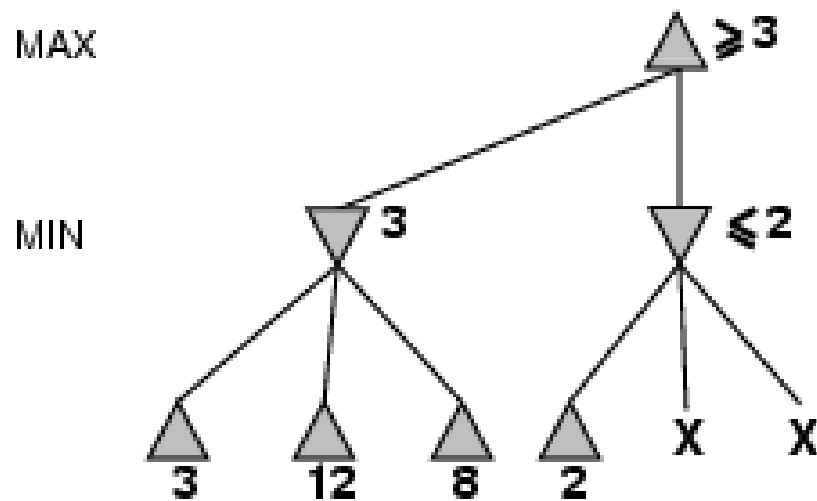


یک نمونه بازی



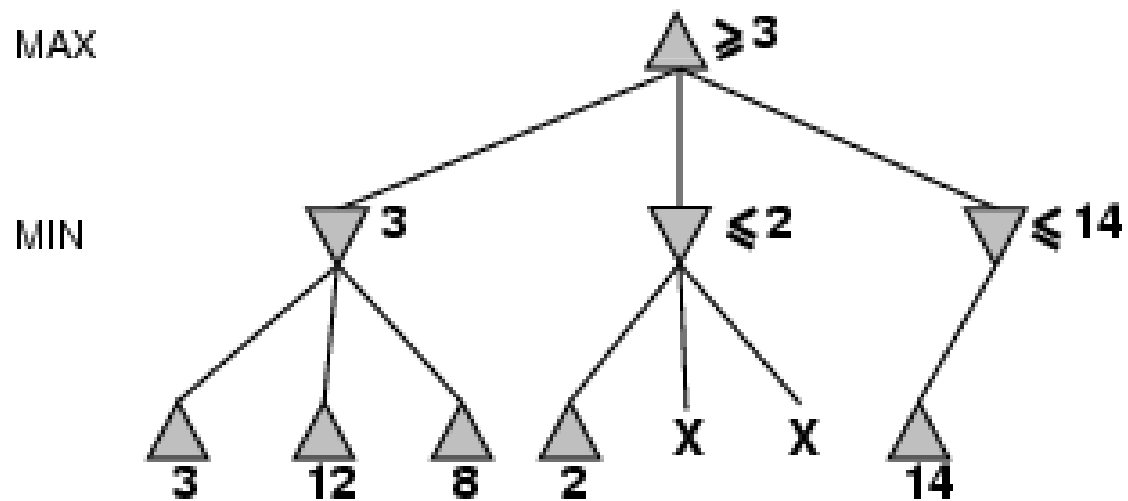


یک نمونه بازی



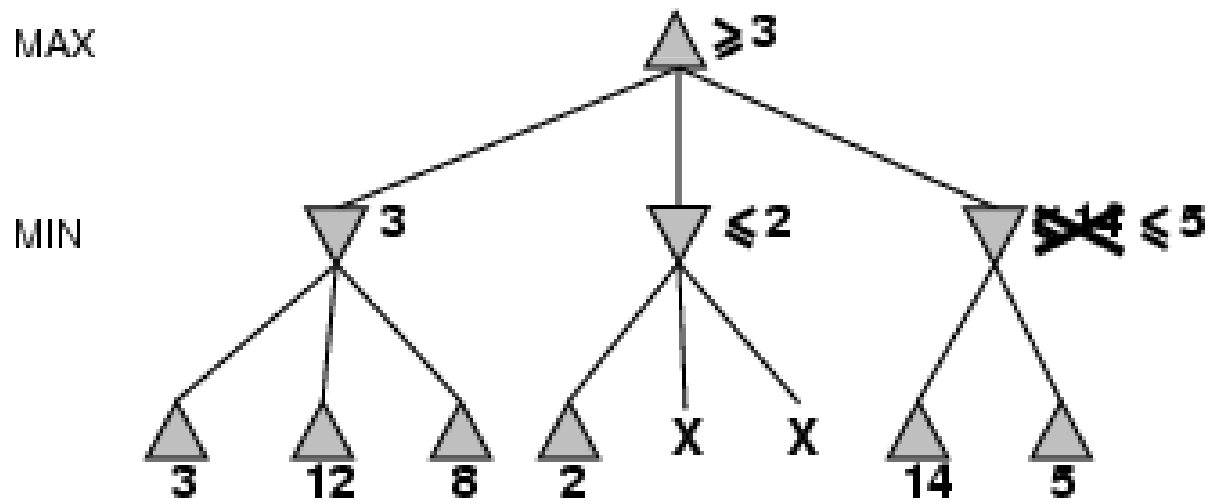


یک نمونه بازی



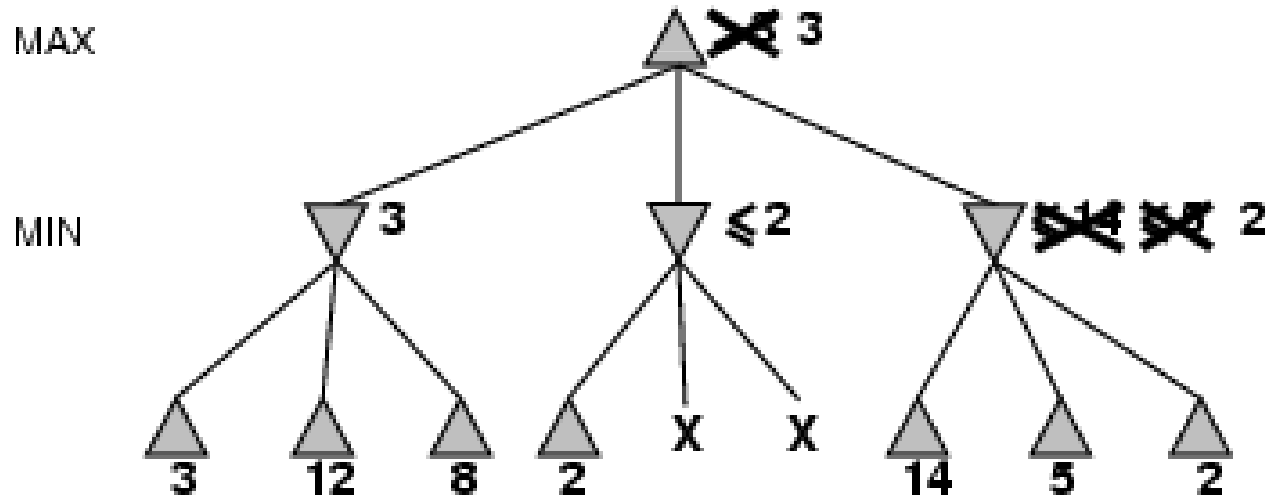


یک نمونه بازی





یک نمونه بازی





الگوریتم minimax

کامل بودن: بله (اگر درخت محدود باشد)

بهینگی: بله

$$O(b^m)$$

پیچیدگی زمانی:

$$O(bm)$$

پیچیدگی فضا:

جستجوی خصمانه

بازیهای چند نفره

تخصیص یک بردار به هر گره، به جای یک مقدار

بازیهای چند نفره معمولاً شامل اتحاد رسمی یا غیر رسمی بین بازیکنان است

اتحاد با پیشروی بازی ایجاد و از بین میرود

بازیکنان بطور خودکار همکاری میکنند، تا به هدف مطلوب انحصاری برسند

to move

A

(1, 2, 6)

B

(1, 2, 6)

(-1, 5, 2)

C

(1, 2, 6) X

(6, 1, 2)

(-1, 5, 2)

(5, 4, 5)

A

(1, 2, 6)

(4, 2, 3)

(6, 1, 2)

(7, 4, -1)

(5, -1, -1)

(-1, 5, 2)

(7, 7, -1)

(5, 4, 5)

;9

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 گره‌ها امکانپذیر است

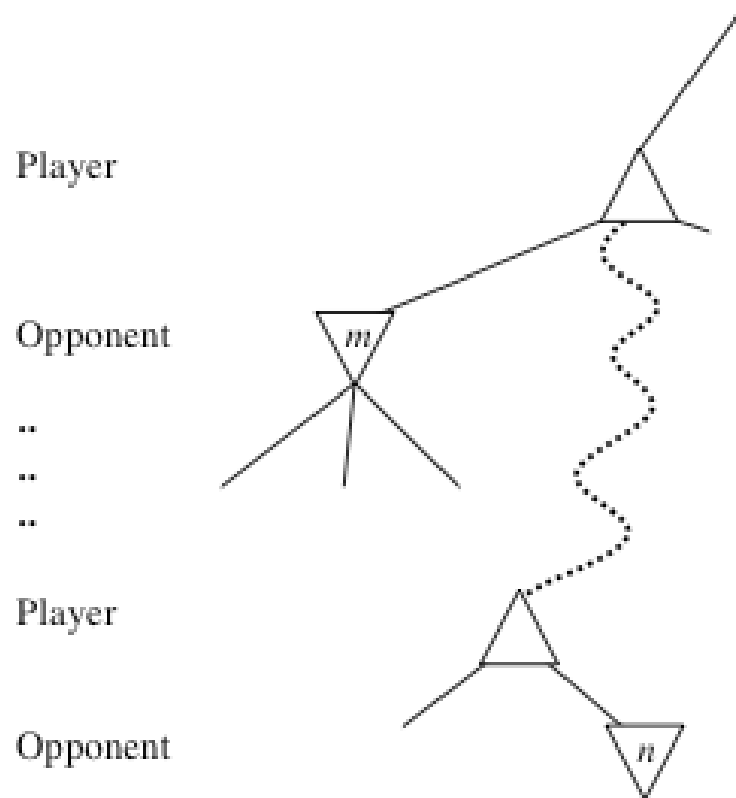
هرس آلفا-بتا:

- انشعابهایی که در تصمیم نهایی تأثیر ندارند را حذف میکند
- 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 گره‌هایی که باید بررسی شوند به $O(b^{d/2})$ تقلیل میابد
- فاکتور انشعاب مؤثر به جای b برابر با جذر b خواهد بود
- پیش بینی آن نسبت به minimax دو برابر است

جستجوی خصمانه



هرس آلفا-بتا



↩ گره n که هر جای درخت میتواند باشد ، بررسی میشود

↩ اگر بازیکن انتخاب بهتری داشته باشد

↩ در گره والد n

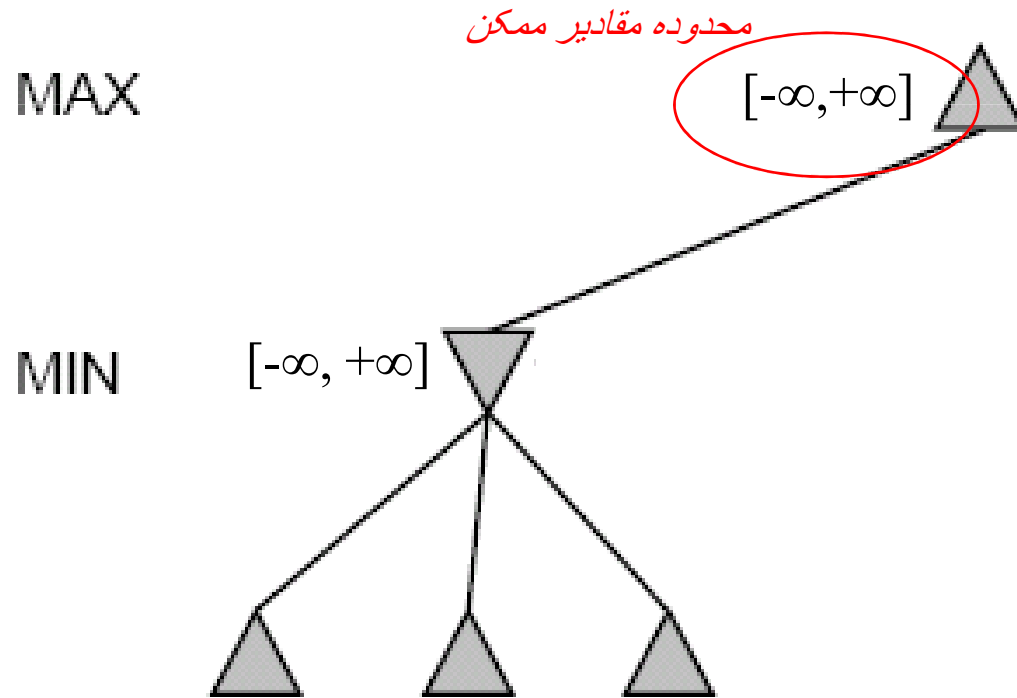
↩ یا هر انتخاب بهتری تا کنون

↩ n هیچوقت در بازی واقعی قابل دسترس نخواهد بود

↩ در نتیجه n هرس میشود

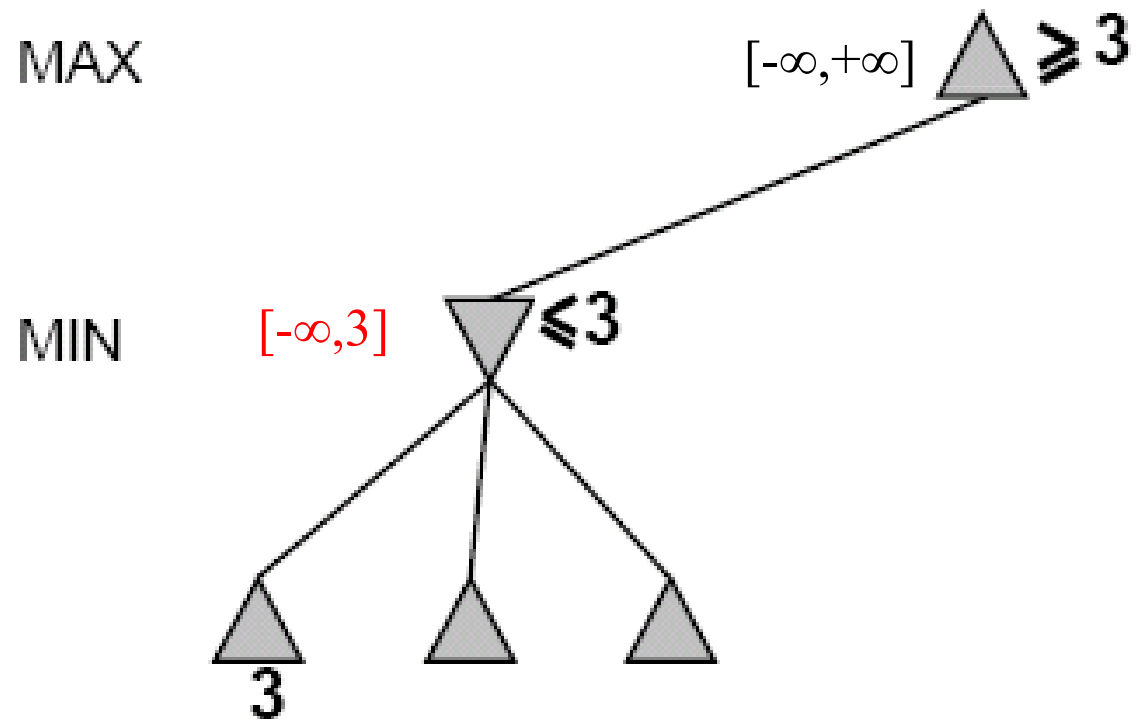
جستجوی خصمانه

مثال: هرس آلفا-بتا



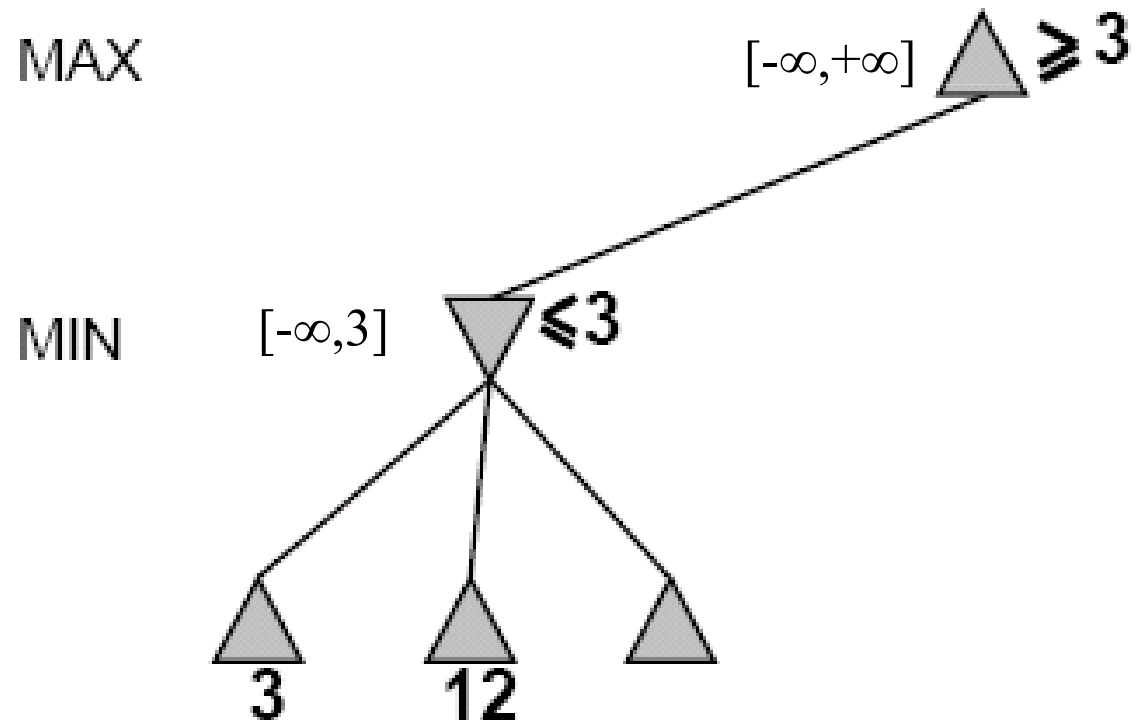
جستجوی خصمانه

مثال: هرس آلفا-بتا

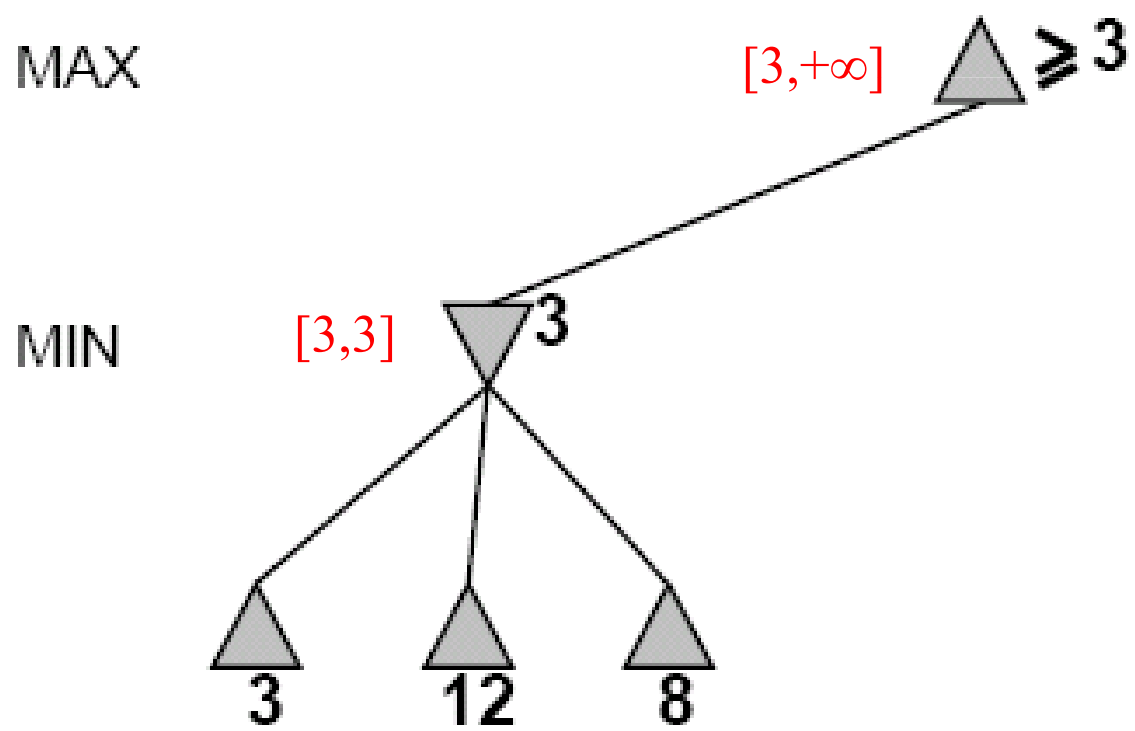


جستجوی خصمانه

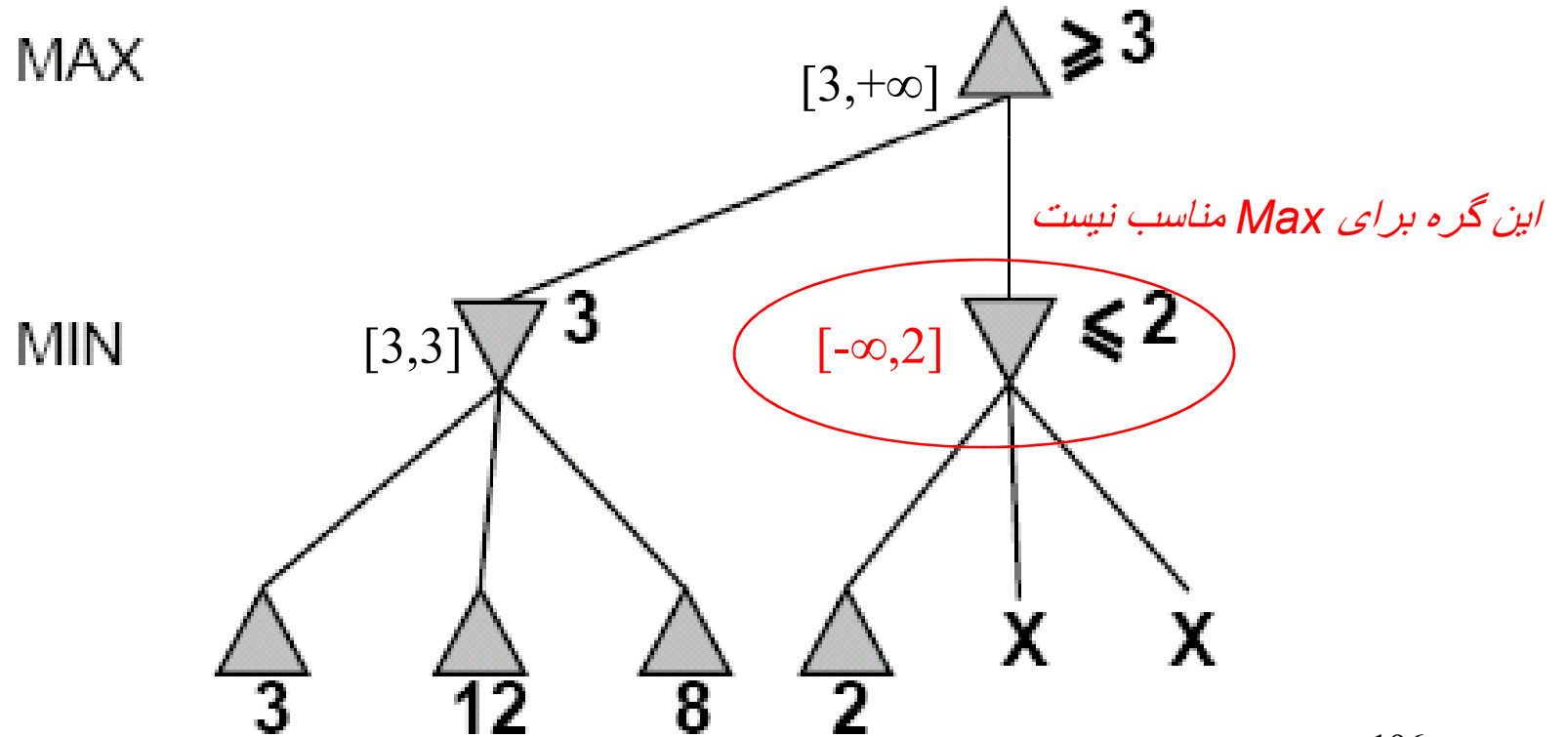
مثال: هرس آلفا-بتا



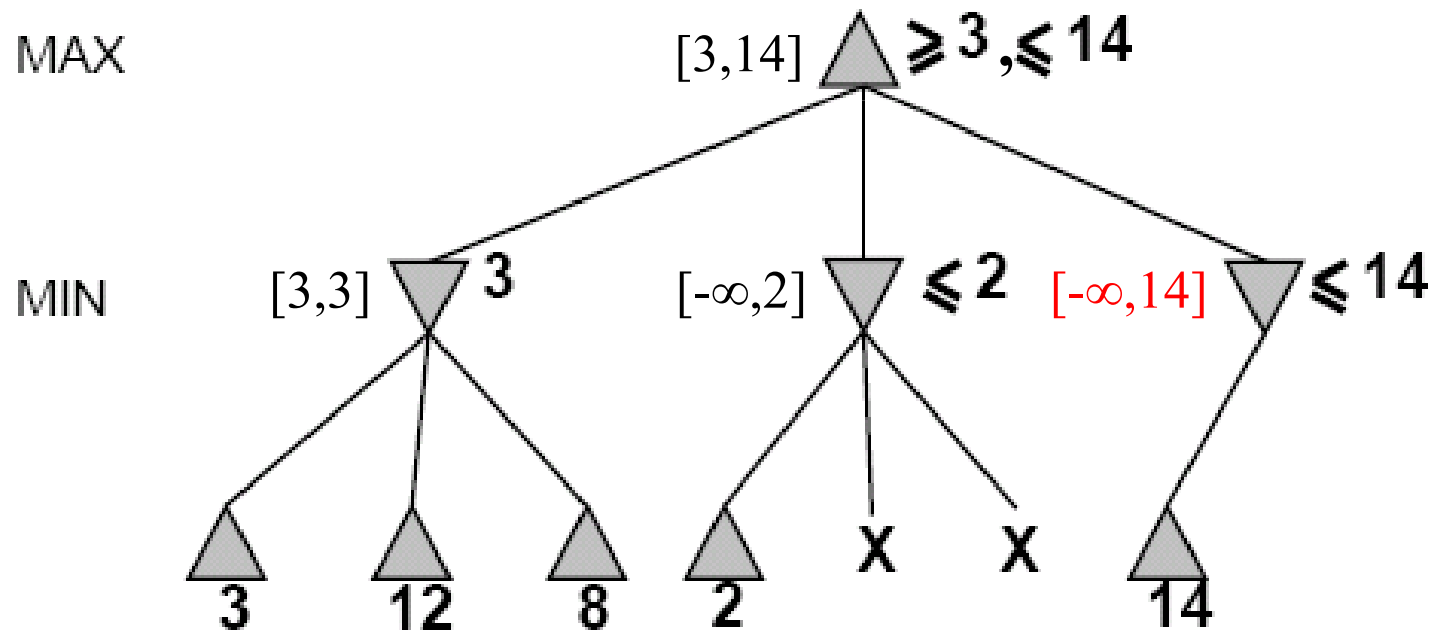
جستجوی خصمانه



مثال: هرس آلفا-بتا

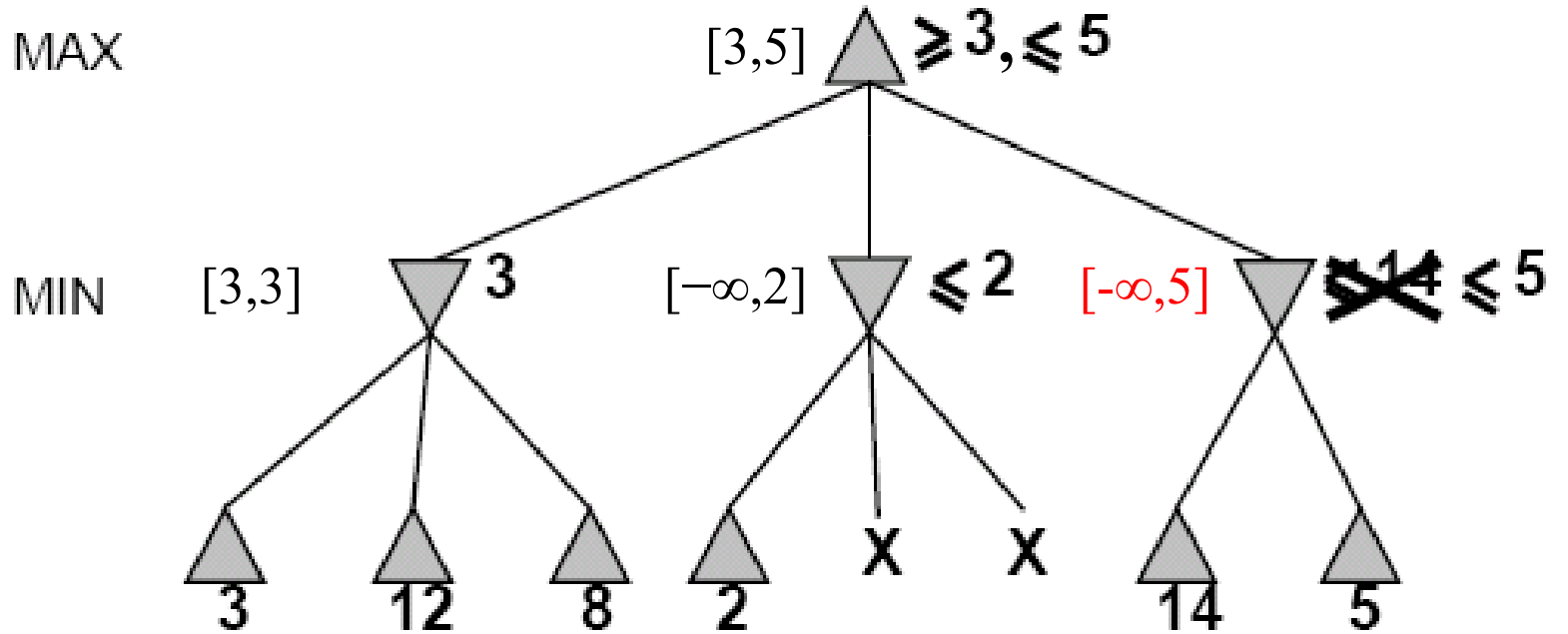


مثال: هرس آلفا-بتا



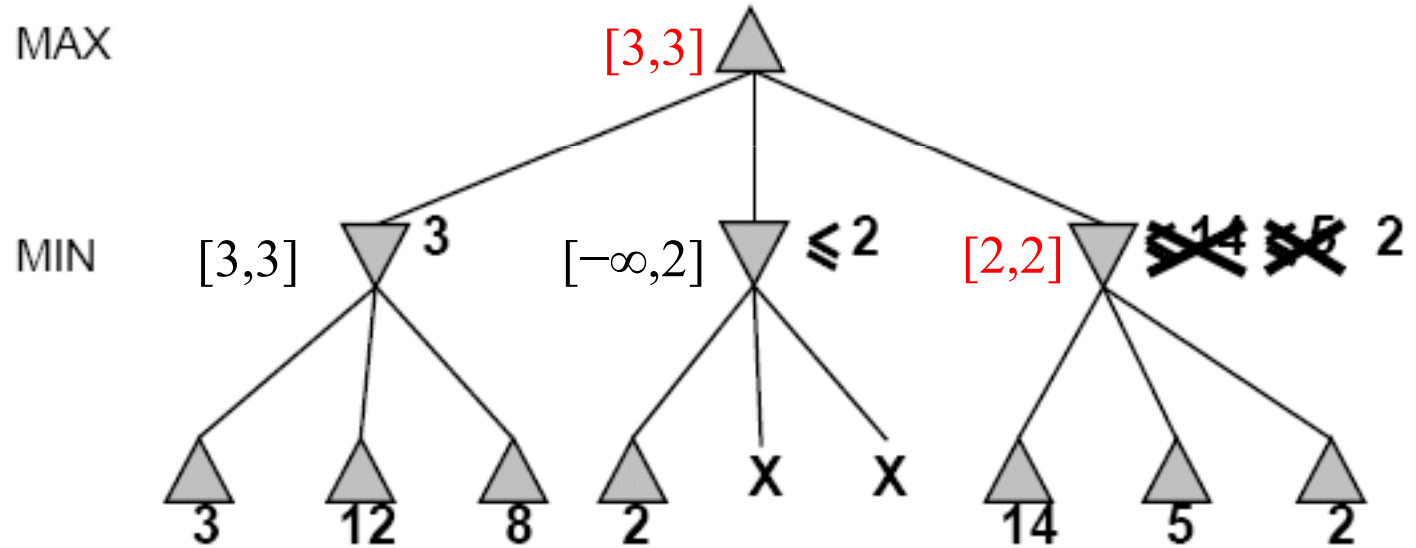


مثال: هرس آلفا-بتا



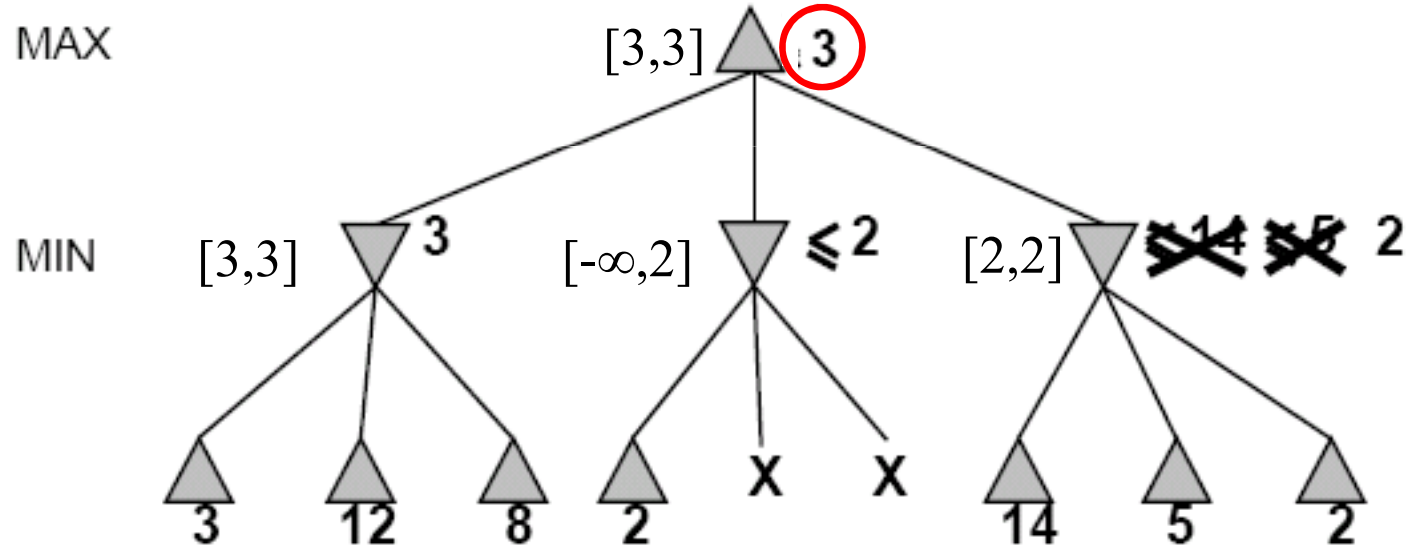
جستجوی خصمانه

مثال: هرس آلفا-بتا



جستجوی خصمانه

مثال: هرس آلفا-بتا



جستجوی خصمانه



بازیهای قطعی 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 بخشی از فضای حالت ، باید جست و جو شود

این عمق عملی نیست ، زیرا حرکات باید در زمانی معقول انجام شود

شانون (1950)

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

جستجوی خصمانه



بازیهای قطعی با اطلاعات ناقص

در شانون، minimax و آلفا-بتا به دو روش بطور متناوب عمل میکنند

جایگزینی 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



تابع ارزیابی اکتشافی EVAL

تابع ارزیابی ، ارائه تخمینی از سودمندی مورد انتظار بازی از یک موقعیت خاص
توابع اکتشافی ، تخمینی از فاصله تا هدف را بر میگردانند

اغلب توابع ارزیابی ، 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

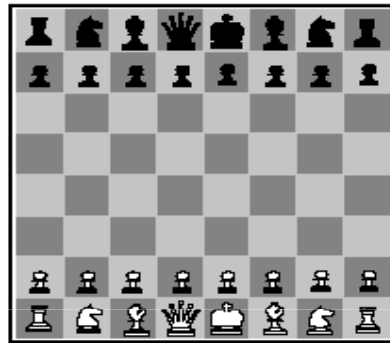
- موجب برنده شدن
- موجب رسم شدن
- منجر به باختن

تابع ارزیابی نمیداند کدام حالت منجر به چه چیزی میشود ، اما میتواند مقداری برگرداند که تناسب حالتها را با هر

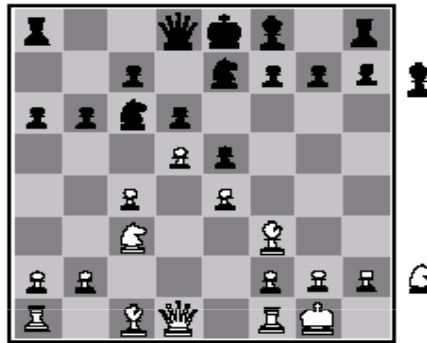
جستجوی خصمانه



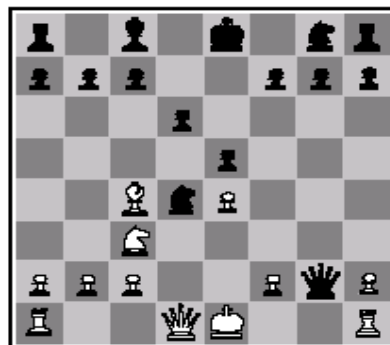
مثال: تابع EVAL



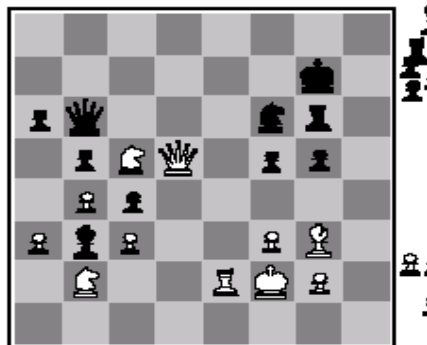
(a) White to move
Fairly even



(b) Black to move
White slightly better



(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:

f_i تعداد هر نوع قطعه در صفحه

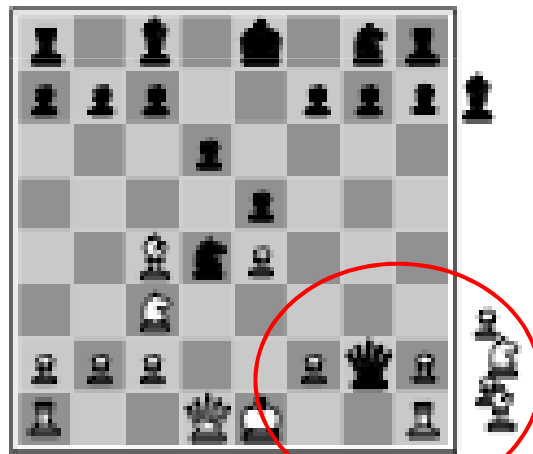
w_i مفادیر آن قطعات (1 برای پیاده، 3 برای اسب یا فیل، 5 برای رخ و ...)

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

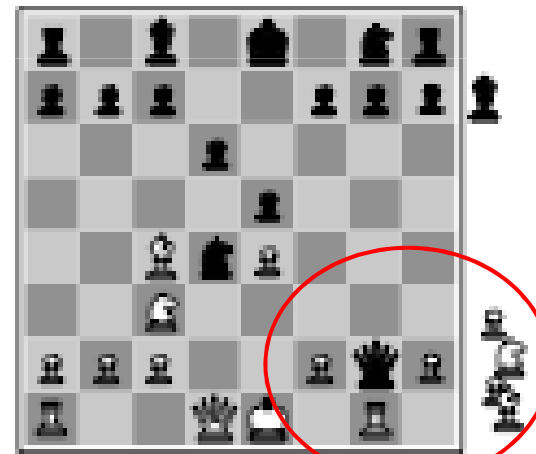


مثال: تابع EVAL

ارزیابی تابع EVAL از مقدار پیروزی در دو موقعیت کاملا متفاوت



الف) سفید حرکت میکند



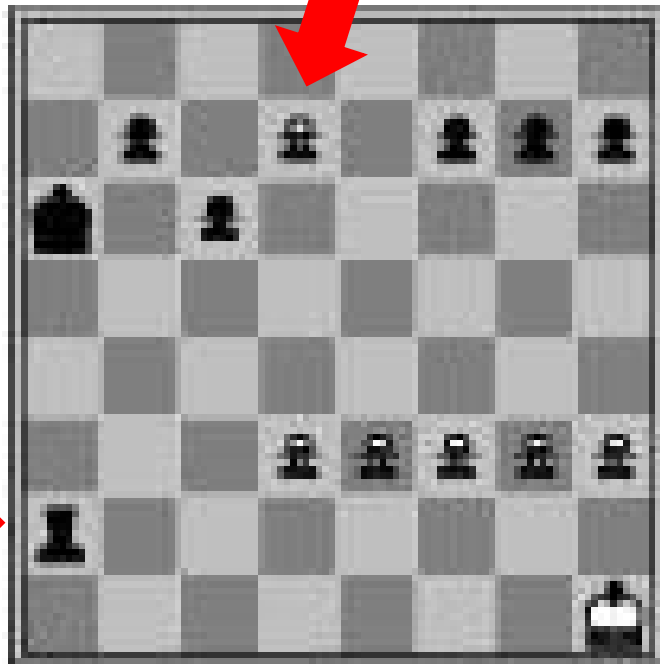
ب) سفید حرکت میکند

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

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

جستجوی خصمانه

اثر افق



Black to move

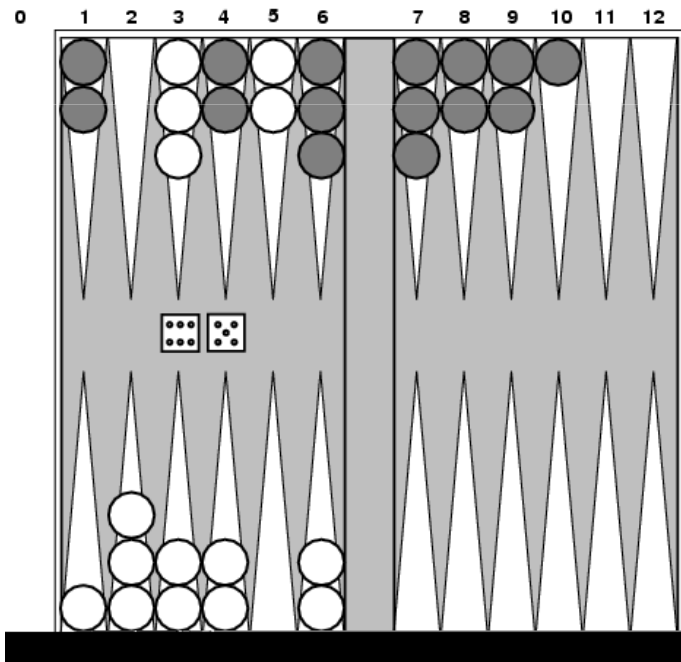
وقتی بوجود می آید که برنامه با اثری از رقیب مواجه شود که منجر به خرابی جدی گشته و اجتناب پذیر است

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 می آید

جستجوی خصمانه

بازیهایی که حاوی عنصر شانس هستند



MAX

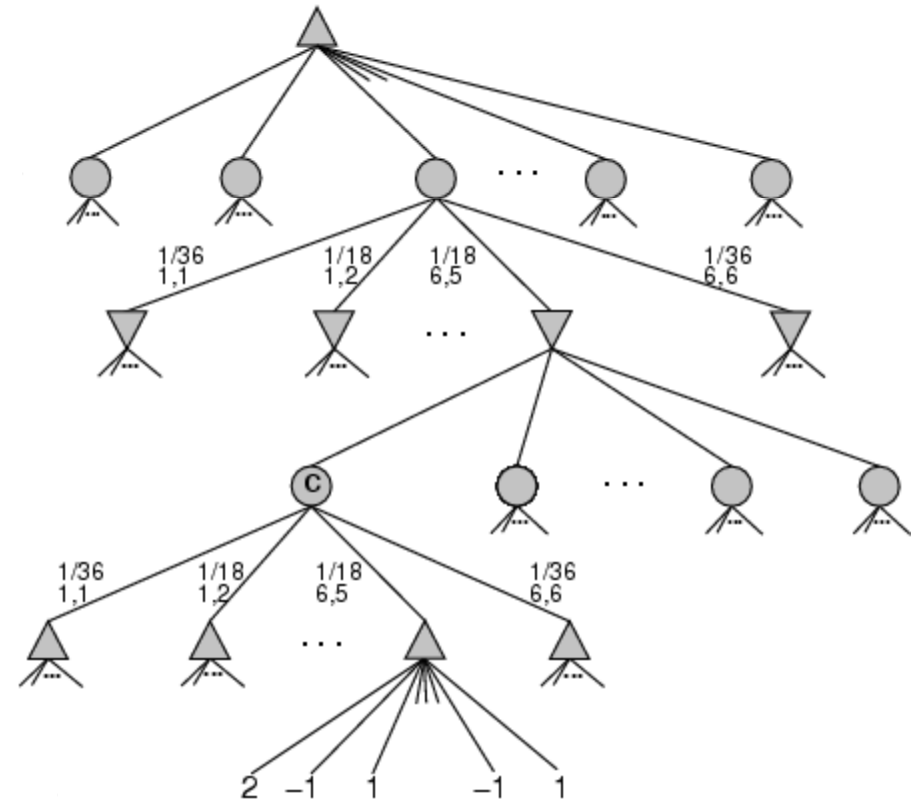
شانس

MIN

شانس

MAX

پایانه





هوش مصنوعي

فصل هفتم

عامل های منطقی



فهرست

↪ عاملهای مبتنی بر دانش

↪ منطق

↪ منطق گزاره ای

↪ الگوهای استدلال در منطق گزاره ای

↪ الگوریتم **resolution**

↪ زنجیر پیشرو و عقبگرد

عاملهای مبتنی بر دانش

مؤلفه اصلی عامل مبتنی بر دانش ، پایگاه دانش آن است

پایگاه دانش: مجموعه ای از جملات

جمله: زبان نمایش دانش و بیان ادعاهایی در مورد جهان



محدوده الگوریتمهای مستقل

محدوده اطلاعات خاص

برای اضافه کردن جملات به پایگاه دانش و درخواست دانسته ها

ASK و TELL

هر دو ممکن است شامل استنتاج باشند

پیروی: انجام فرایند استنتاج تحت مقررات خاص



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

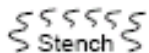
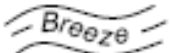


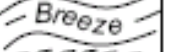
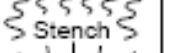
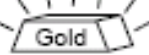

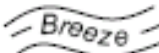
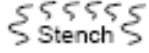
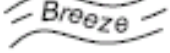

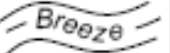

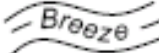
↩↪ عامل پایگاه دانش خیلی شبیه به عاملهایی با حالت درونی است

↩↪ 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

4	 Stench	 Breeze		
3		 Breeze  Stench  Gold	 PIT  Breeze	
2	 Stench	 Breeze		
1	 START	 Breeze	 PIT  Breeze	
	1	2	3	4

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:

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



توصیف جهان WUMPUS

قابل مشاهده کامل: خیر، فقط ادراک محلی

قطعی: بله، نتیجه دقیقاً مشخص است

رویدادی: خیر، ترتیبی از فعالیتهاست

ایستا: بله، WUMPUS و گودالها حرکت ندارند

گسسته: بله

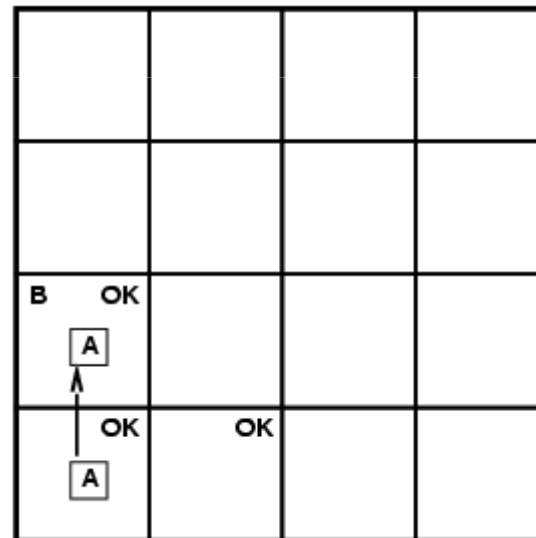
تک عامله: بله، WUMPUS در اصل یک خصوصیت طبیعی است

WUMPUS در جهان

OK			
OK A	OK		

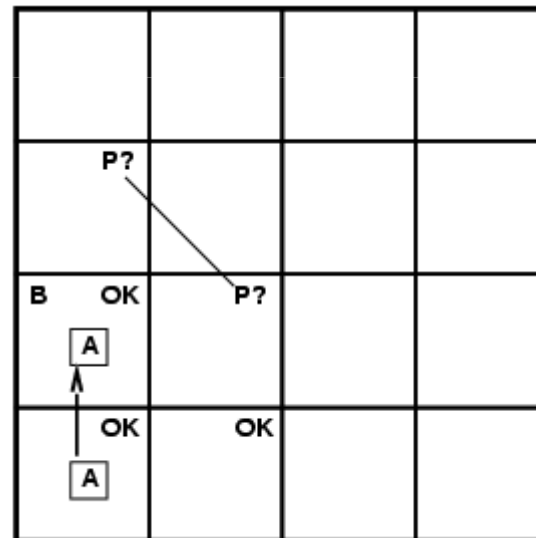
- A = عامل
- B = نسیم
- G = درخشش، طلا
- OK = مربع امن
- P = گودال
- S = تعفن
- V = ملاقات شده
- W = Wumpus

توصیف جهان WUMPUS



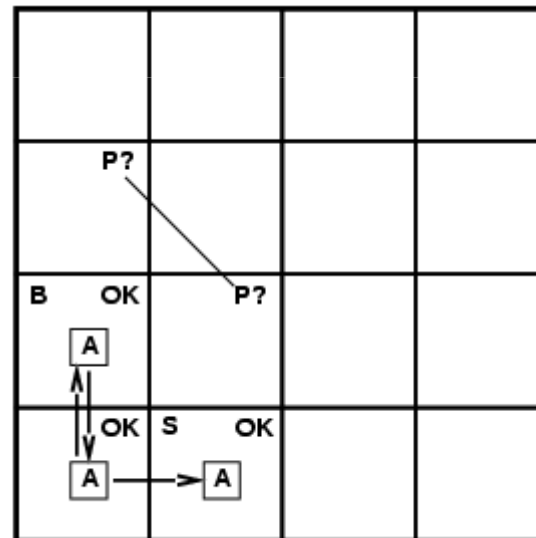
- A = عامل
- B = نسیم
- G = درخشش، طلا
- OK = مربع امن
- P = گودال
- S = تعفن
- V = ملاقات شده
- W = Wumpus

توصیف جهان WUMPUS



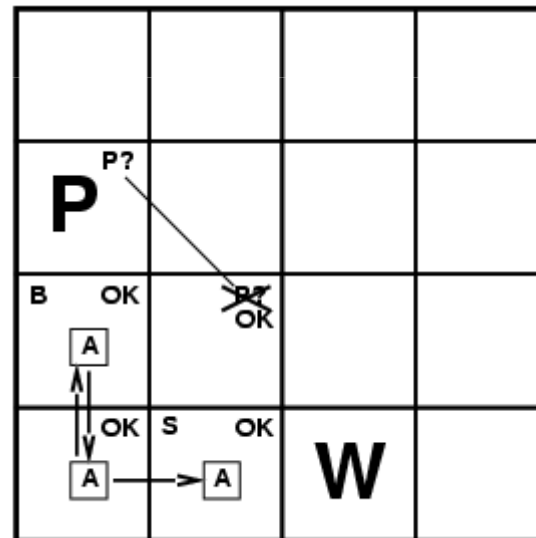
- A = عامل
- B = نسیم
- G = درخشش، طلا
- OK = مربع امن
- P = گودال
- S = تعفن
- V = ملاقات شده
- W = Wumpus

توصیف جهان WUMPUS



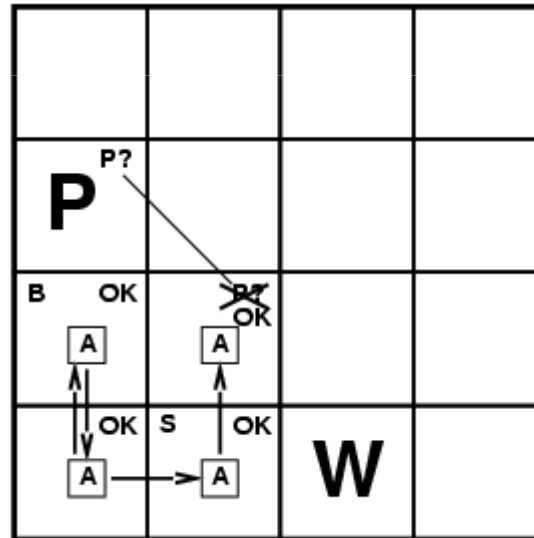
- A = عامل
- B = نسیم
- G = درخشش، طلا
- OK = مربع امن
- P = گودال
- S = تعفن
- V = ملاقات شده
- W = Wumpus

توصیف جهان WUMPUS



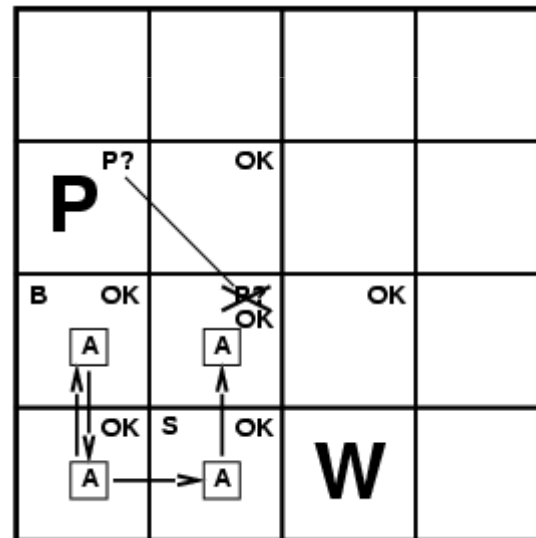
- A = عامل
- B = نسیم
- G = درخشش، طلا
- OK = مربع امن
- P = گودال
- S = تعفن
- V = ملاقات شده
- W = Wumpus

توصیف جهان WUMPUS



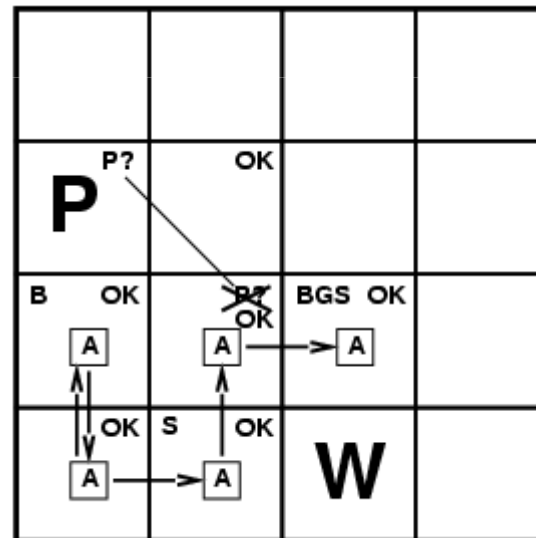
- A** = عامل
- B** = نسیم
- G** = درخشش، طلا
- OK** = مربع امن
- P** = گودال
- S** = تعفن
- V** = ملاقات شده
- W** = Wumpus

توصیف جهان WUMPUS



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

توصیف جهان WUMPUS



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

منطق

یک زبان رسمی:

ترکیب (نحو): چه کلمه بندی صحیح است. (خوش فرم)

معناشناسی: یک کلمه بندی صحیح چه معنایی دارد

در منطق ، معنای زبان ، درستی هر جمله را در برابر هر جهان ممکن تعریف میکند

مثال ، در زبان ریاضیات

$X+2 \geq y$ یک جمله اما x^2+y جمله نیست

$X+2 \geq y$ در جهان درست است اگر $x=7$ و $y=1$

$X+2 \geq y$ در جهان غلط است اگر $x=0$ و $y=6$

استلزام

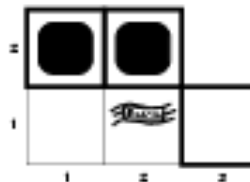
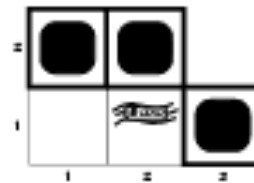
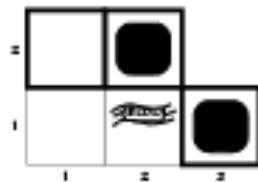
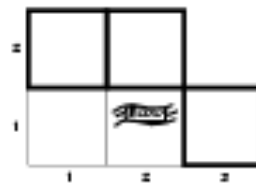
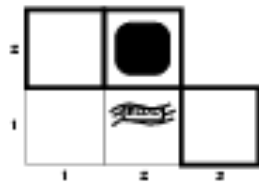
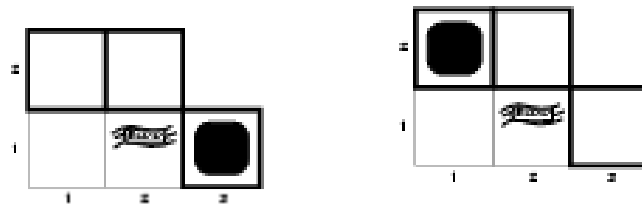
استلزام منطقی بین جملات این است که جمله ای بطور منطقی از جمله دیگر پیروی میکند

$$a \vDash b$$

- جمله **a** استلزام جمله **b** است
- جمله **a** جمله **b** را ایجاد میکند
- اگر و فقط اگر ، در هر مدلی که **a** درست است ، **b** نیز درست است
- اگر **a** درست باشد ، **b** نیز درست است
- درستی **b** در درستی **a** نهفته است

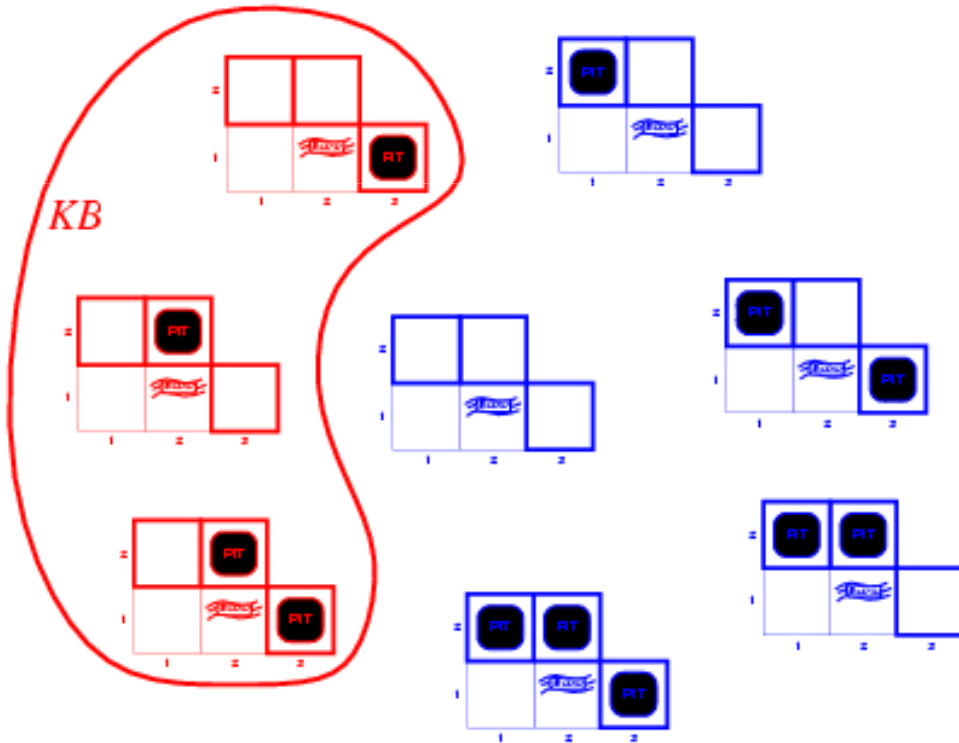
مثال: جمله $x+y=4$ مستلزم جمله $4=x+y$ است

مدلهای Wumpus





مدلهای Wumpus



KB

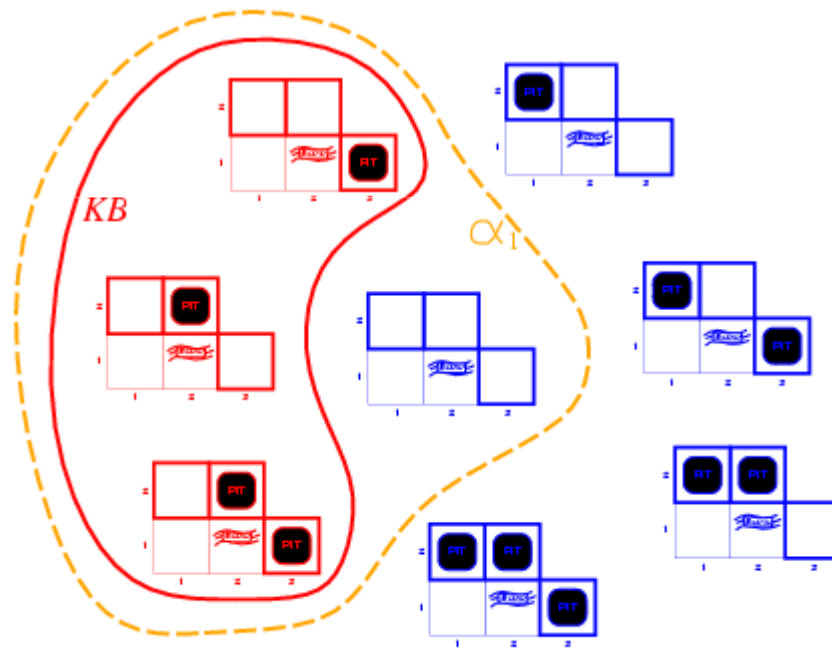
=

قوانین دنیای Wumpus

+

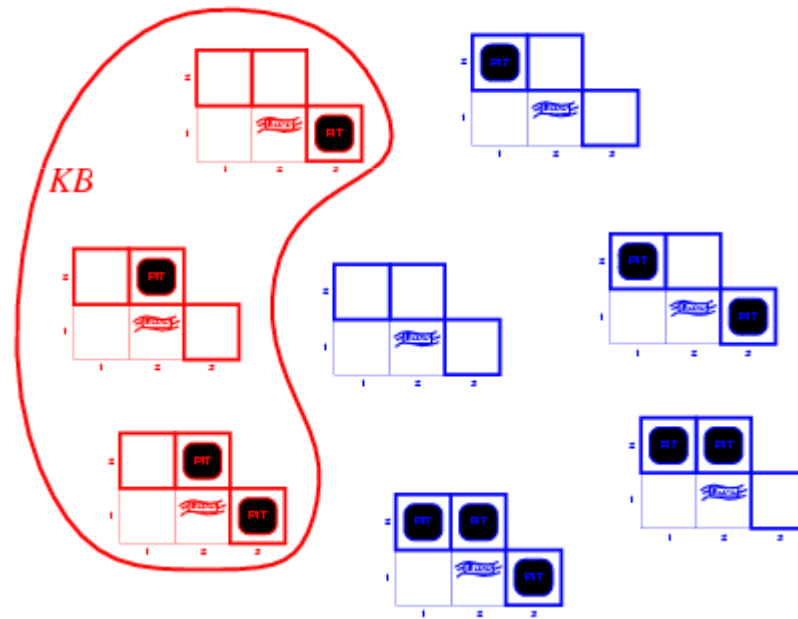
مشاهدات

مدلهای Wumpus



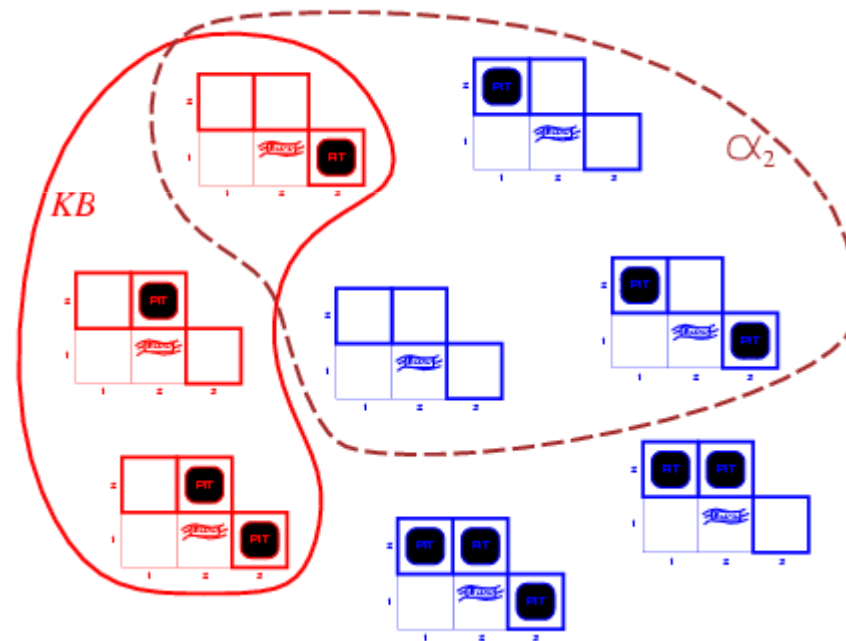
$KB = \text{wumpus}$ + مشاهدات دنیای
 $\alpha_1 = "[1,2]$ امن است", $KB \models \alpha_1$

مدلهای Wumpus



$KB = \text{wumpus}$ دنیای + مشاهدات

مدلهای Wumpus



$KB = \text{wumpus}$ + مشاهدات + دنیای
 $\alpha_2 = "[2,2]$ امن است", $KB \models \alpha_2$



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 منطقی ، از جملات ساده تر ساخته میشوند

↪ \neg (not) جمله ای مثل $W_{1,3} \neg W_{1,3}$ نقیض $W_{1,3}$ است

↪ لیترال یک جمله اتمیک (لیترال مثبت) ، یا یک جمله اتمیک منفی (لیترال منفی) است

↪ \wedge (and) مثل $W_{1,3} \wedge P_{1,3}$ ترکیب عطفی نام دارد. هر بخش آن یک عطف نامیده میشود

↪ \vee (or) مثل $(W_{1,3} \wedge P_{3,1}) \vee W_{2,2}$ ترکیب فصلی مربوط به فصل های $W_{1,3} \wedge P_{3,1}$ و $W_{2,2}$

↪ \Rightarrow (استلزام): $(W_{1,3} \wedge P_{3,1}) \vee \neg W_{2,2}$ استلزام یا شرطی نامیده میشود. مقدمه یا مقدم آن $W_{1,3} \wedge P_{3,1}$ و نتیجه

یا تالی آن $\neg W_{2,2}$ است

↪ جمله $W_{1,3} \Leftrightarrow W_{2,2}$ دو شرطی نام دارد



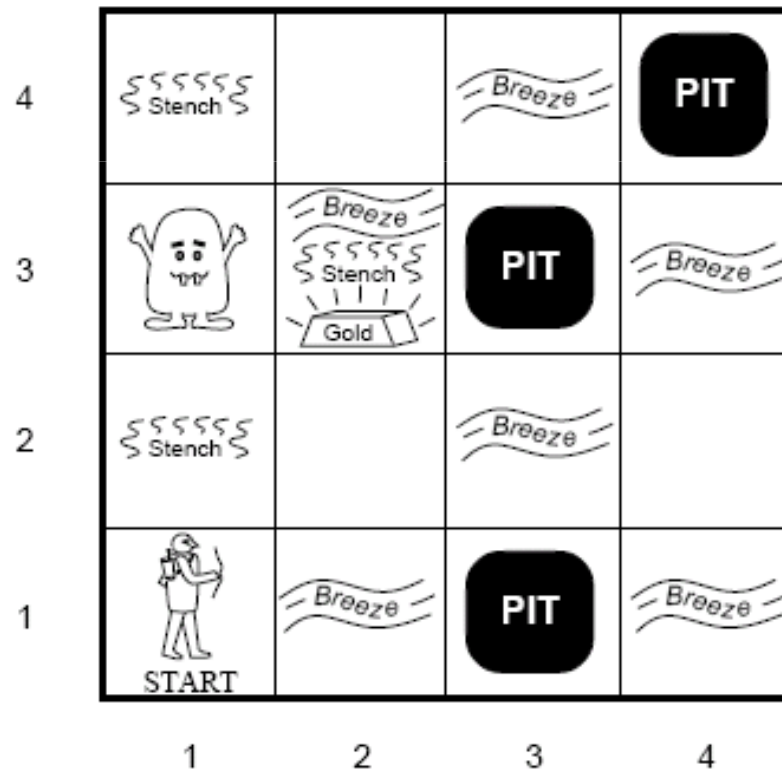
منطق گزاره ای

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

جدول درستی پنج رابطه منطقی

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
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

منطق گزاره ای در دنیای Wumpus



در $B_{1,1}$ نسیمی وجود دارد
 $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

در $[1,1]$ گودالی وجود ندارد
 $R_1: \neg P_{1,1}$



الگوهای استدلال در منطق گزاره ای

قوانین استنتاج: الگوهایی استاندارد که زنجیره ای از نتایج را برای رسیدن به هدف ایجاد میکند

قیاس استثنایی: با استفاده از ترکیب عطفی، میتوان هر عطف را استنتاج کرد (یعنی هر وقت جمله ای به شکل $a \Rightarrow b$ داده شود، جمله b را میتوان استنتاج کرد).

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

میتوان از

$(WumpusAhead \wedge WumpusAlive)$

و

$(WumpusAhead \wedge WumpusAlive) \Rightarrow Shoot$

Shoot را استنتاج کرد



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

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

مثال: **WumpusAlive** را میتوان از جمله زیر استنتاج کرد
(**WumpusAhead** \wedge **WumpusAlive**)

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:

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



قانون resolution

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

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

قانون resolution واحد میتواند به قانون resolution کامل تعمیم داد:

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

الگوریتم resolution

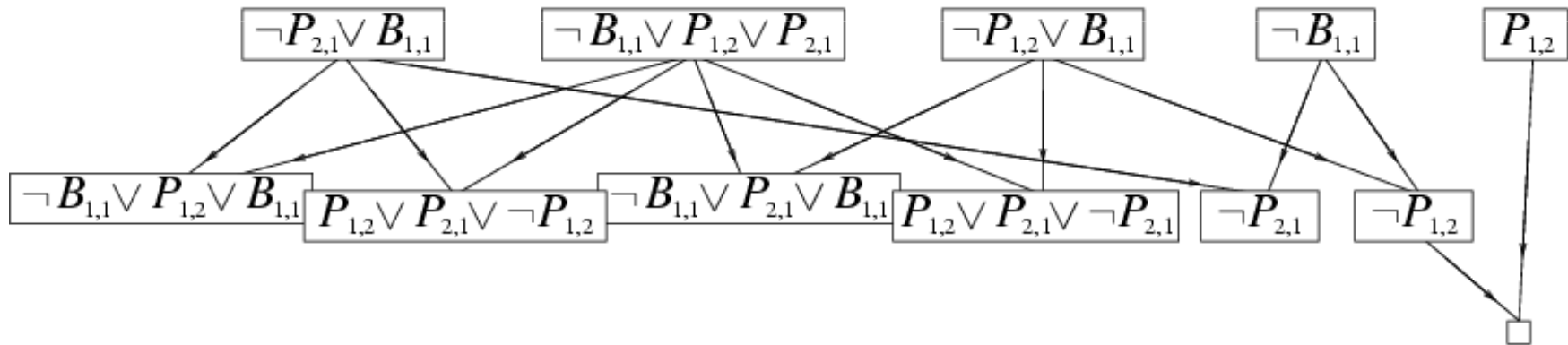
شکل نرمال عطفی (CNF): جمله ای که بصورت ترکیب عطفی از ترکیبات فصلی لیترالها بیان میشود. در هر عبارت موجود در جمله k -CNF دقیقا k لیترال وجود دارد

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

الگوریتم resolution:

- برای اینکه نشان دهیم $KB \models a$, مشخص میکنیم $(KB \wedge \neg a)$ ارضا کننده نیست
- ابتدا $(KB \wedge \neg a)$ را به CNF تبدیل میکنیم
- سپس قانون resolution به عبارات کوچک حاصل اعمال میشود
- هر جفتی که شامل لیترالهای مکمل باشد، resolution میشود تا عبارت جدیدی ایجاد گردد
- اگر این عبارت قبلا در مجموعه نباشد، به آن اضافه میشود
- فرایند تا محقق شدن یکی از شروط زیر ادامه می یابد:
 - 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

مثال: الگوریتم resolution



$$KB = (B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})) \wedge \neg B_{1,1} \quad \alpha = \neg P_{1,2}$$

زنجر پیشرو و عقبگرد

عبارات هورن: ترکیب فصلی لیترالهایی است که فقط یکی 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 معینی که فاقد لیترالهای منفی باشد ، گزاره ای بنام حقیقت نام دارد

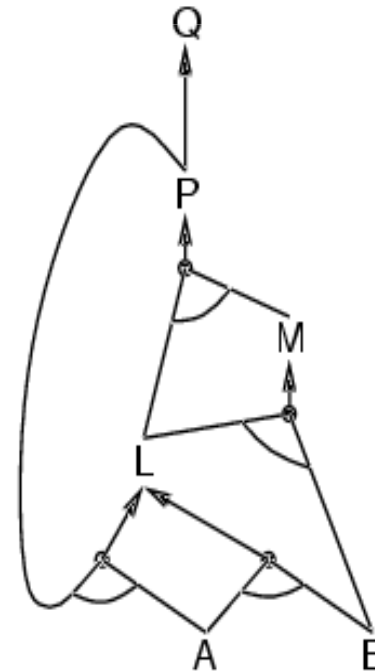
عبارات معین اساس برنامه نویسی منطقی را میسازد

استنتاج با عبارات هورن ، از طریق الگوریتم های زنجر پیشرو و زنجر عقبگرد انجام میگردد

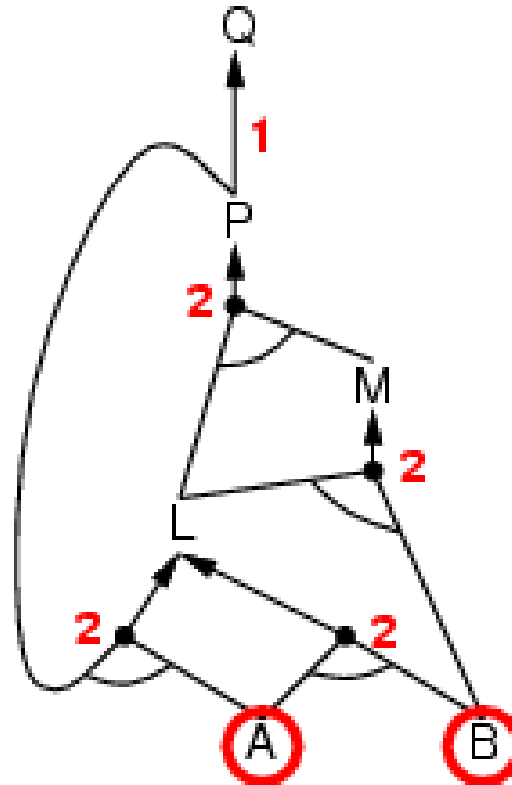
زنجیر پیشرو

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

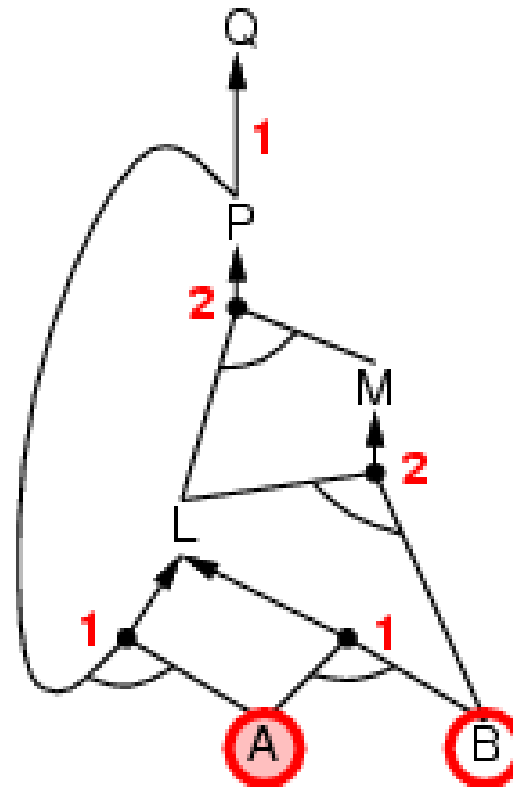
$P \Rightarrow Q$
 $L \wedge M \Rightarrow P$
 $B \wedge L \Rightarrow M$
 $A \wedge P \Rightarrow L$
 $A \wedge B \Rightarrow L$
 A
 B



زنجیر پیشرو

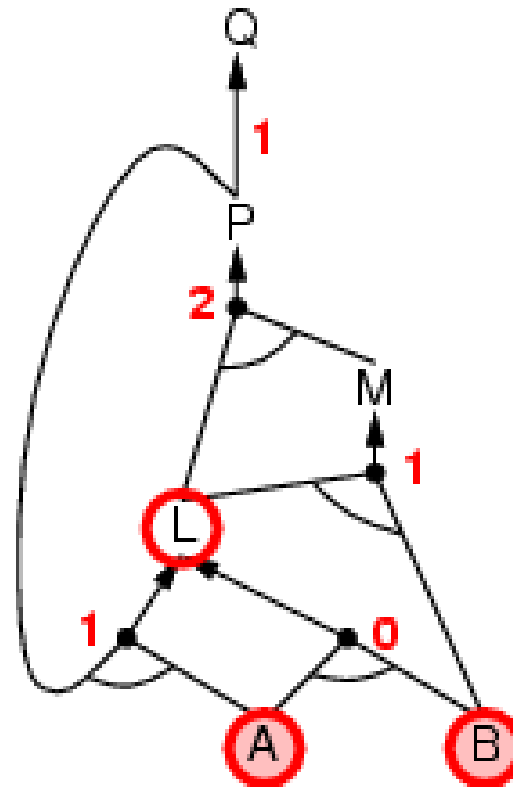


زنجیر پیشرو

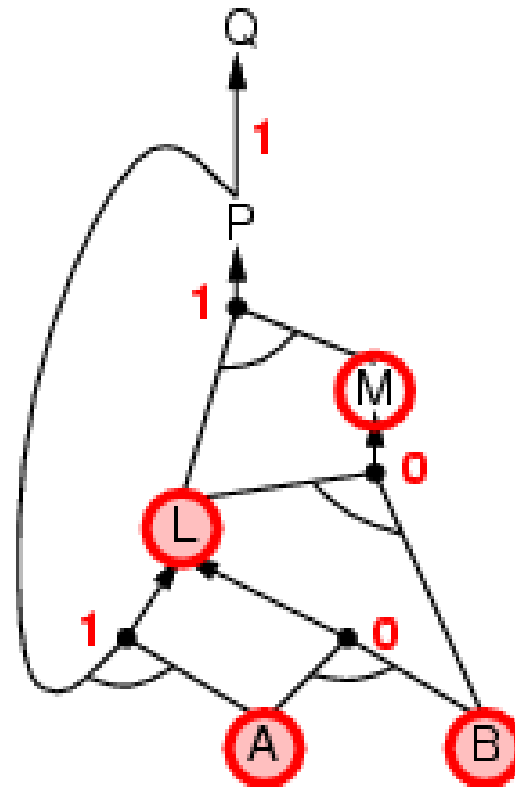




زنجیر پیشرو

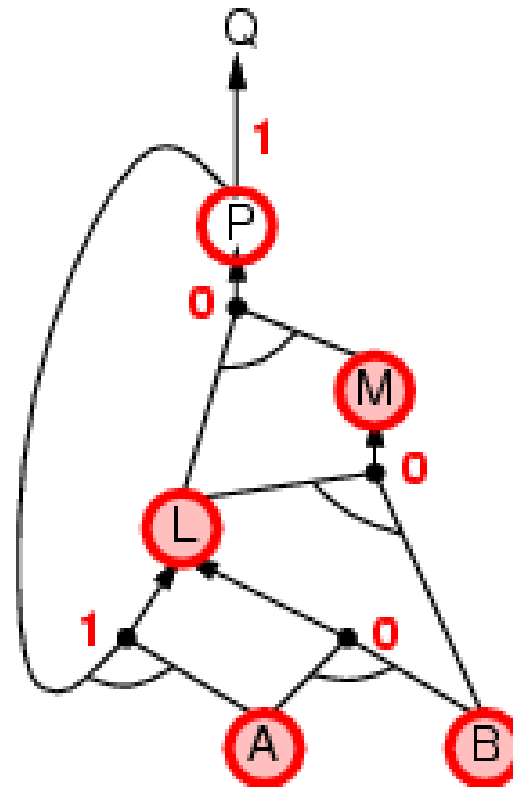


زنجیر پیشرو

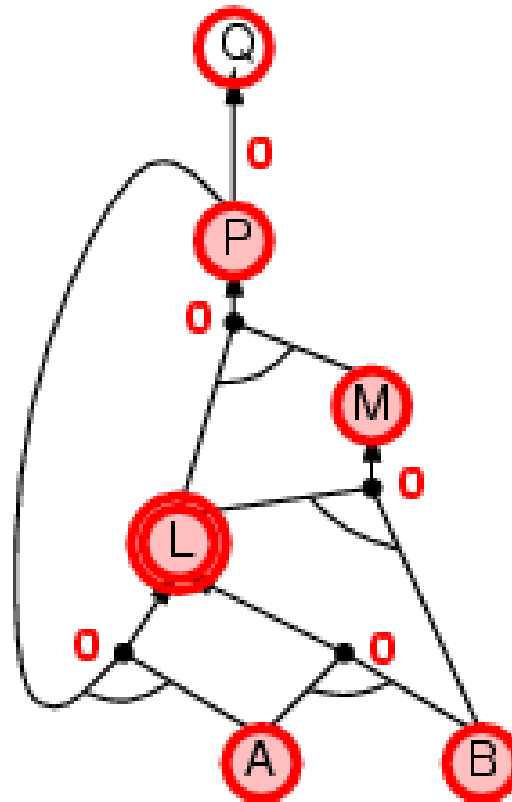




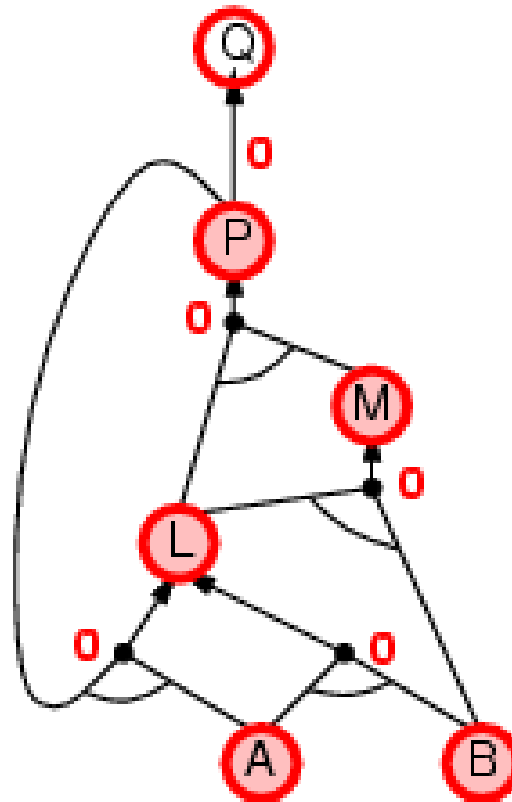
زنجیر پیشرو



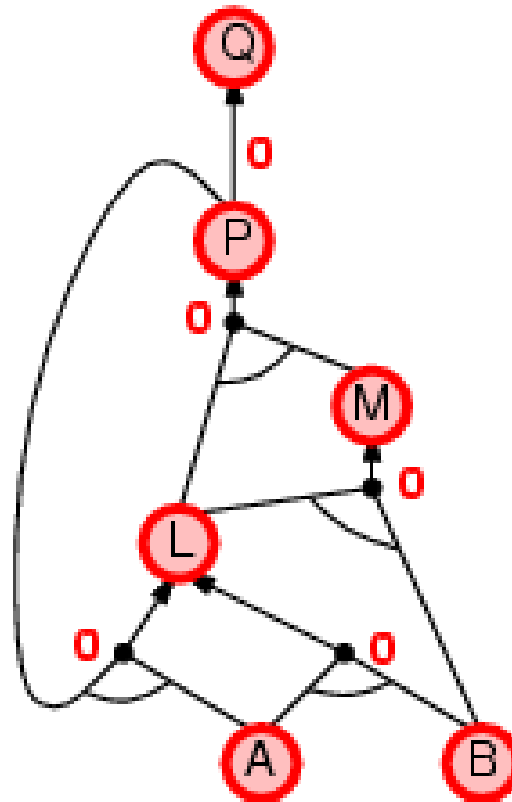
زنجیر پیشرو



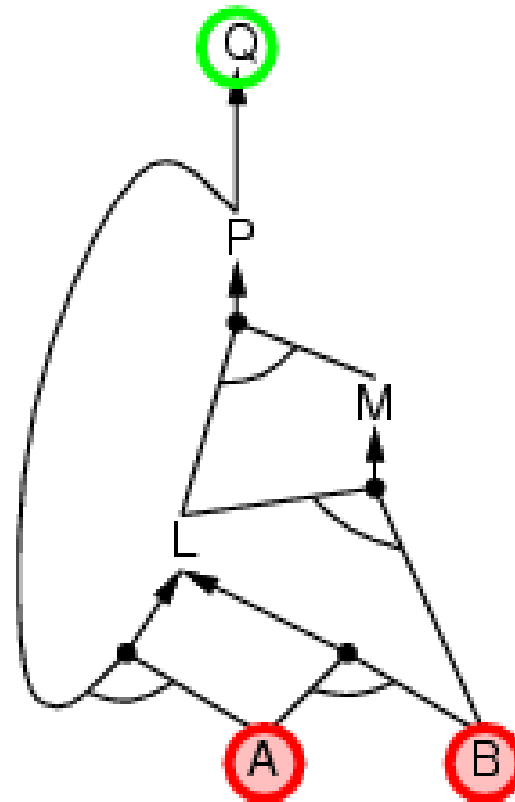
زنجیر پیشرو



زنجیر پیشرو



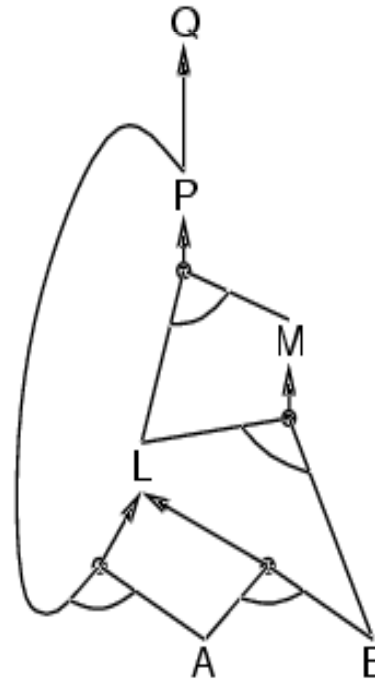
الگوریتم عقبگرد کامل



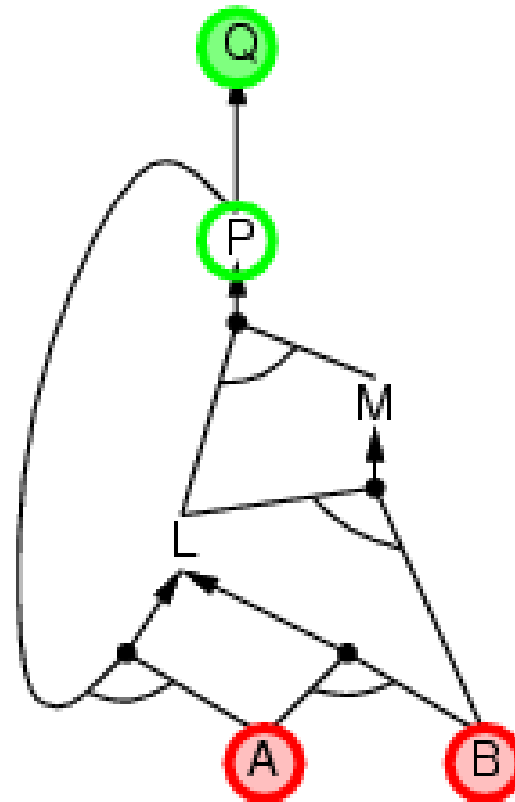
الگوریتم عقبگرد کامل

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

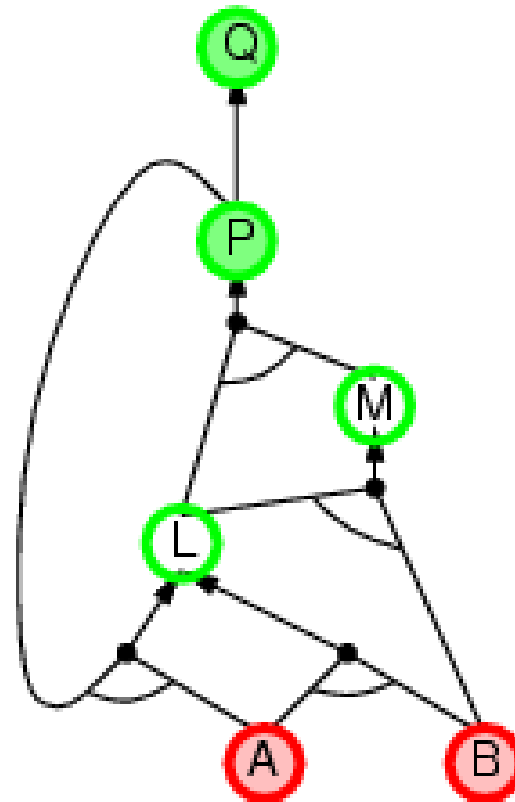
$P \Rightarrow Q$
 $L \wedge M \Rightarrow P$
 $B \wedge L \Rightarrow M$
 $A \wedge P \Rightarrow L$
 $A \wedge B \Rightarrow L$
 A
 B



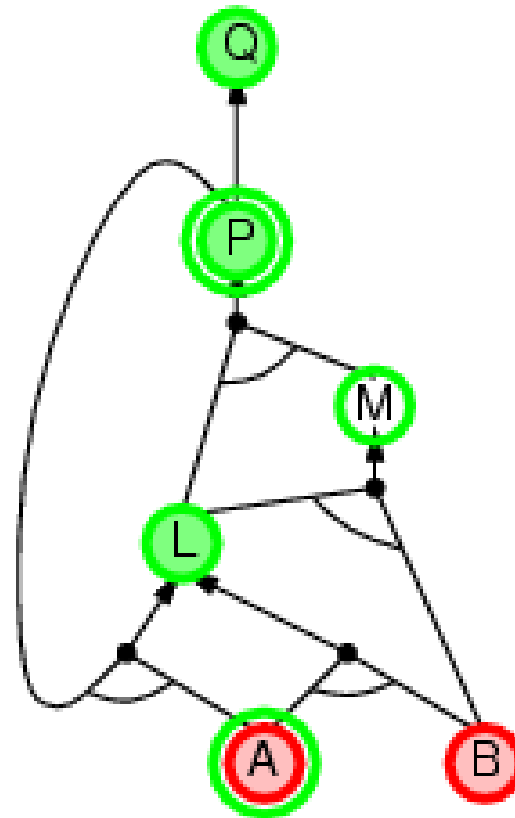
الگوریتم عقبگرد کامل



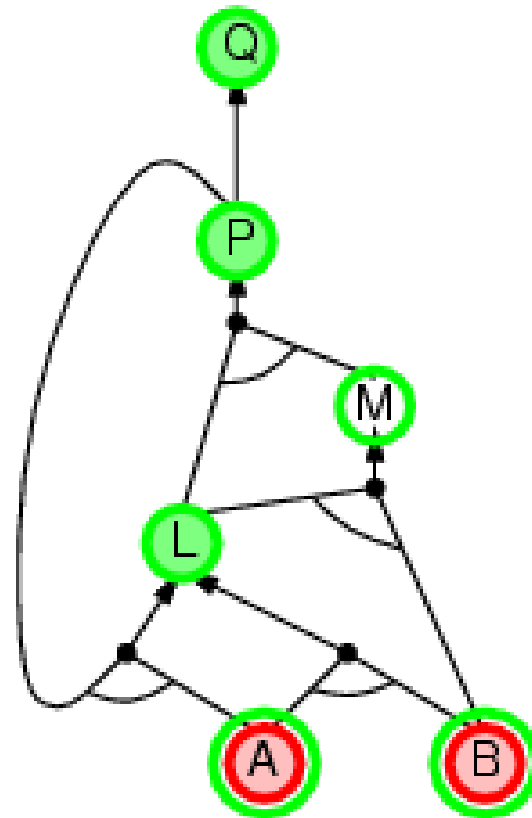
الگوریتم عقبگرد کامل



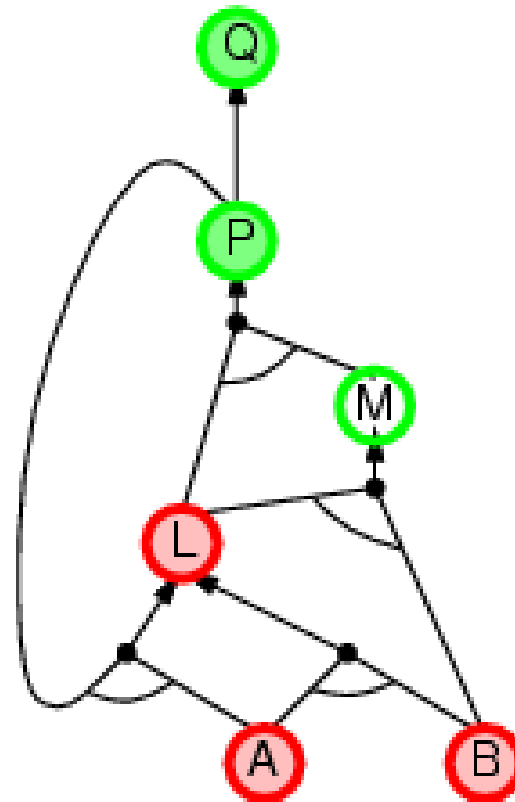
الگوریتم عقبگرد کامل



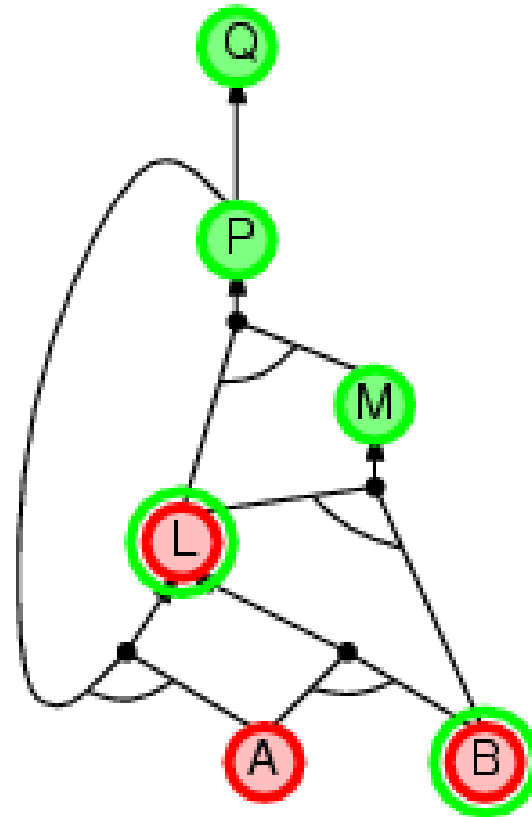
الگوریتم عقبگرد کامل



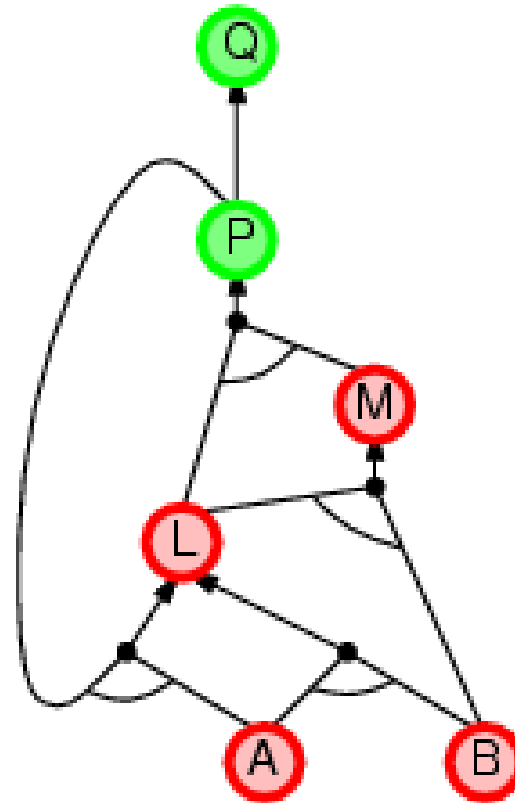
الگوریتم عقبگرد کامل



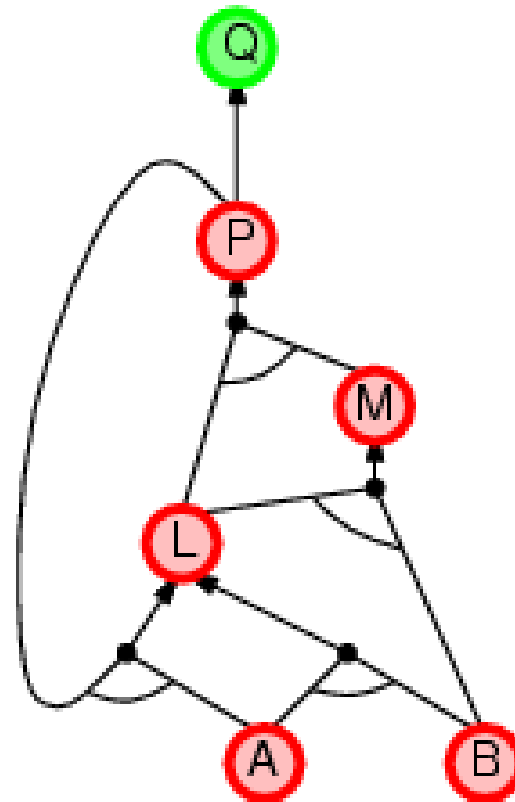
الگوریتم عقبگرد کامل



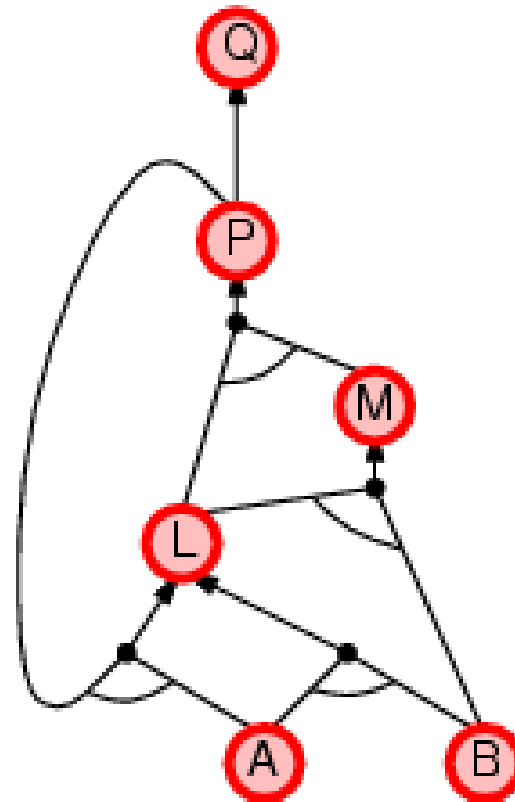
الگوریتم عقبگرد کامل



الگوریتم عقبگرد کامل



الگوریتم عقبگرد کامل





هوش مصنوعي

فصل هشتم

منطق رتبه اول



فهرست

مروری بر منطق گزاره ای

منطق رتبه اول

انواع منطق

نحو و معنای منطق رتبه اول

مهندسی دانش



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 جزئی

■ با استفاده از ترکیب فصلی و نقیض

قابلیت ترکیب

■ معنای جمله ، تابعی از معنای بخشهای آن

معنا ، مستقل از متن است

■ بر خلاف زبانهای طبیعی که ، معنای جملات وابسته به متن است

معایب

فاقد قدرت بیانی برای تشریح دقیق محیطی با اشیای مختلف

■ بر خلاف زبانهای طبیعی

منطق رتبه اول

↪ اساس منطق گزاره ای را پذیرفته و بر اساس آن یک منطق بیانی میسازیم

↪ از ایده های نمایشی زبان طبیعی 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 یکانی یا خواص مثل قرمز ، گرد ، اول و ...

↪ رابطه های چندتایی مثل برادر بودن ، بزرگتر بودن ، بخشی از ، مالکیت و ...

↪ توابع: پدر بودن ، بهترین دوست ، یکی بیشتر از و ...

↪ منطق رتبه اول توسط اشیا و رابطه ها ساخته میشود



انواع منطق

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



نحو و معنای منطق رتبه اول

نمادهای ثابت؛ اشیا را نشان میدهد. مثال: علی، 2، رضا، ...

نمادهای محمول؛ رابطه‌ها 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: \forall, \exists

جملات اتمیک

هر ترم یک عبارت منطقی است که به شیء اشاره میکند

نمادهای ثابت ترم هستند

همیشه استفاده از نماد متمایز برای نامگذاری شیء آسان نیست

پای چپ پای پادشاه John ~~Left leg~~(John)

متغیر یا ثابت یا (ترم 1، ترم 2، ...، ترم n) تابع = ترم

جملات اتمیک: ترکیب ترمهای اشیاء و محولهای روابط

ترم 2 = ترم 1 یا (ترم 1، ترم 2، ...، ترم n) محمول = جملات اتمیک

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

پدر ریچارد با مادر جان ازدواج کرده است



جملات پیچیده

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

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

$\neg \text{Brother}(\text{LeftLeg}(\text{Richard}), \text{John})$

مثال:

$\text{Brother}(\text{Richard}, \text{John}) \wedge \text{Brother}(\text{John}, \text{Richard})$

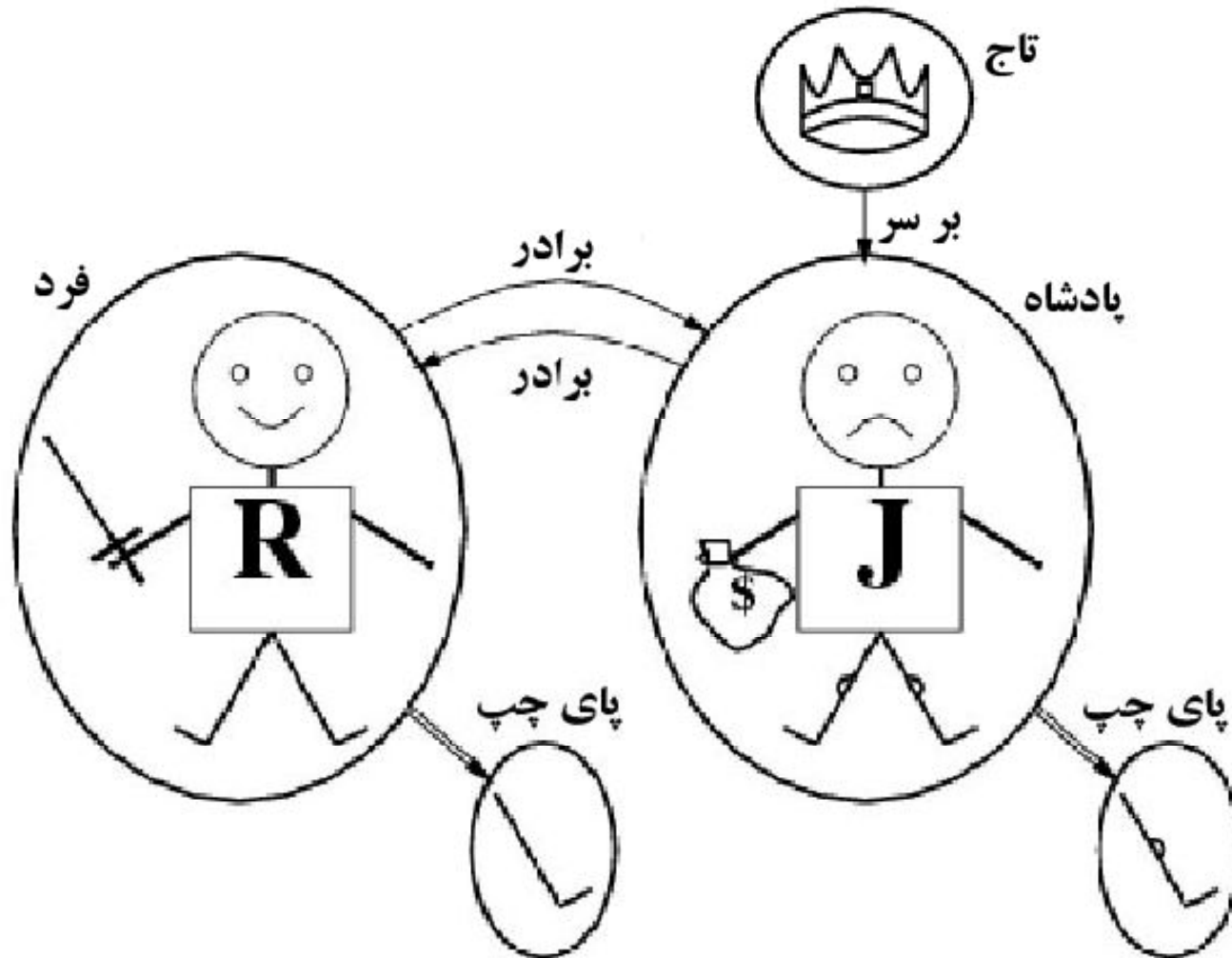
$\text{King}(\text{Richard}) \vee \text{King}(\text{John})$

$\neg \text{King}(\text{Richard}) \Rightarrow \text{King}(\text{John})$

منطق رتبه اول

مثال

مدلی با پنج شیء ،
دو رابطه دودویی ،
سه رابطه یکانی و
یک تایکانی به نام
پای چپ





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...“

سور عمومی

<جمله> <متغیرها> \forall

$\forall x P$ که در آن P یک عبارت منطقی است، بیان میکند که P برای هر شیء x درست است

مثال: $\forall x \text{ King}(x) \Rightarrow \text{Person}(x)$

سور وجودی

<جمله> <متغیرها> \exists

$\exists x P$ که در آن P یک عبارت منطقی است ، بیان میکند که P حداقل برای یک شیء x درست است

مثال: $\exists x \text{Crown}(x) \wedge \text{OnHead}(x, \text{John})$

خصوصیات سورها

↔ ⇒ رابط طبیعی برای کار با \forall و \wedge رابط طبیعی 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 ضعیفی میشود

↔ $\forall x \forall y$ برابر است با $\forall y \forall x$ و $\exists x \exists y$ برابر است با $\exists y \exists x$

↔ $\forall x \forall y$ برابر نیست با $\exists x \exists y$

↔ $\exists x \forall y \text{ Loves}(x,y)$

■ حداقل یک نفر وجود دارد که همه چیز در جهان را دوست دارد

↔ $\forall y \exists x \text{ Loves}(x,y)$

■ همه در دنیا حداقل یک نفر را دوست دارند

خصوصیات سورها

«هر کسی بستنی را دوست دارد» به معنای این است که «هیچ کس وجود ندارد که بستنی را دوست نداشته باشد»

$$\neg \exists x \neg \text{Likes}(x, \text{IceCream}) \text{ هم ارز } \forall x \text{ Likes}(x, \text{IceCream})$$

$$\neg \exists x P \text{ هم ارز } \forall x \neg P$$

$$\exists x \neg P \text{ هم ارز } \neg \forall x P$$

$$\neg \exists x \neg P \text{ هم ارز } \forall x P$$

$$\neg \forall x \neg P \text{ هم ارز } \exists x P$$

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

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



ادعاها و تقاضاها

↩ جملات از طریق **TELL** به پایگاه دانش اضافه میشوند
↩ این جملات را ادعا گویند

- $TELL(KB, King(John))$
- $TELL(KB, \forall x King(x) \Rightarrow Person(x))$

↩ با استفاده از **ASK** تقاضاهایی را از پایگاه دانش انجام میدهیم
↩ این پرسشها، تقاضا یا هدف نام دارد

- $ASK(KB, Person(John))$
- $ASK(KB, \exists x Person(x))$

↩ لیست جانشینی یا انقیاد
↩ لیستی از جانشینها در صورت وجود بیش از یک پاسخ

دامنه خویشاوندی

مادر هر فرد والد مؤنث آن فرد است

$$\forall m,c \text{ Mother}(c) = m \Leftrightarrow \text{Female}(m) \wedge \text{Parent}(m,c)$$

شوهر هر فرد ، همسر مذکر آن فرد است

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

مذکر و مؤنث بودن طبقه های متمایزی هستند

$$\forall x, \text{Male}(x) \Leftrightarrow \neg \text{Female}(x)$$

والد و فرزند ، رابطه های معکوس هستند

$$\forall p,c \text{ Parent}(p,c) \Leftrightarrow \text{Child}(c,p)$$

پدر بزرگ یا مادر بزرگ والدین والدین هر فرد است

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

اعداد و مجموعه ها

$$\Rightarrow \forall s \text{ Set}(s) \Leftrightarrow (s = \{\}) \vee (\exists x, s_2 \text{ Set}(s_2) \wedge s = \{x|s_2\})$$



$$\Rightarrow \neg \exists x, s \{x|s\} = \{\}$$



$$\Rightarrow \forall x, s x \in s \Leftrightarrow s = \{x|s\}$$



$$\Rightarrow \forall x, s x \in s \Leftrightarrow [\exists y, s_2 \{ (s = \{y|s_2\} \wedge (x = y \vee x \in s_2))]$$





مهندسي دانش

فرآیند 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

➤ کدگذاری دانش کلی در مورد دامنه

➤ کد گذاری توصیف نمونه مسئله خاص

➤ اعمال تقاضاها به رویه استنتاج و دریافت پاسخ

➤ اشکال زدایی پایگاه دانش