دانشگاه آزاد
دانشکده کامپوتر

هوش مصنوعی

Artificial Intelligence

گردآوری : مهندس بهروز نیرومندفام

Email: ComputerCollege_Fam@Yahoo.Com

- رفع اشکال در ساعات کلاس.
- ارتباط دائمی از طریق Computercollege_fam@Yahoo.com

حضور در کلاس

حضور در کلاس الزامی است و براساس قوانین آموزشی، غیبت بیش از ۱۶/۳ منجر به اعلام غیبت برای دانشجو خواهد شد.

Email: ComputerCollege_Fam@Yahoo.Com

- امتحان نهائی.....۱۴نمره
- میان ترم.....۴نمره
- تمرین، تحقیق، فعالیت کلاسی و حضور در کلاس ..... ۳ نمره
- جمع:۲۱ نمره

سرفصل مطالب

- هوش مصنوعی چیست؟ مبانی و تاریخچه هوش مصنوعی
- عامل های هوشمند
- حل مساله
- روش های جست و جو
- مساله های ارضای محدودیت
- تئوری بازی (جست و جوی خصمانه)
- عامل های مبتنی بر دانش، نمایش منطق، استدلال
- منطق رتبه اول، استنتاج، نمایش دانش
- معرفی و آموزش زبان Prolog
- معرفی برخی کاربردها در سیستم های خبره، پردازش زبان طبیعی، بینایی ماشین و رباتیک

کتاب هوش مصنوعی رهیافتی نوین
اثر: استوارت راسل و پیتر نورویگ
مترجم: عین الله جعفر نژاد قمی

هوش مصنوعی و برنامه نویسی پرولوگ
اثر: ایوان برانکو
مترجم: مهندس رامین مولاناپور

فصل اول

هوش مصنوعی

Email: ComputerCollege_Fam@Yahoo.Com

فهرست

- هوش مصنوعی چیست؟
- مبانی هوش مصنوعی
- تاریخچه هوش مصنوعی

هوش چیست؟

- توانایی پاسخ به محرک های تجربه نشده
- قدرت استدلال
- درک ارتباط حقایق
- درک و کشف معنی
- تشخیص حقیقت
- یادگیری

هوش مصنوعی، شاخه‌ایست از علم کامپیوتر که ملزومات محاسباتی اعمالی همچون ادراک ،(Perception) استدلال (reasoning)و یادگیری (learning)را بررسی کرده و سیستمی جهت انجام چنین اعمالی ارائه می‌دهد.

معمایی برای سنجش هوش

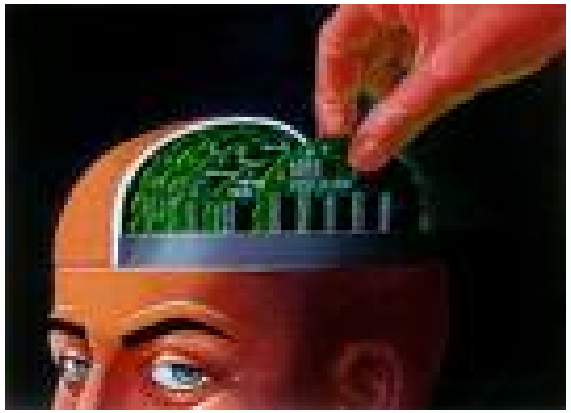
- **سکه های تقلبی**

ده تا کیسه داریم که داخل هر کدام ۲۵ عدد سکه است. ۹ تا از این کیسه ها حاوی سکه های واقعی و یک کیسه حاوی سکه های تقلبی است.

حال با یکبار (فقط یکبار) عمل توزین کیسه حاوی سکه های تقلبی را پیدا کنید با این فرض که سکه های حقیقی هر کدام ۱۰ گرم و سکه های تقلبی هر کدام ۹ گرم وزن دارند.

توضیح و راهنمایی اینکه درب کیسه ها باز است و شما هر طور که راحتید عمل کنید منتهی فقط یکبار باید عمل توزین انجام شود

هوش مصنوعی چیست؟



- قابلیت یک سیستم غیر جاندار دارای هوش از دو جهت

۱- بازسازی رفتار انسان (نگرش مهندسی) با تاکید بر خروجی یک سیستم هوش مصنوعی

۲- مدل سازی مکانیزم های هوشمند (نگرش نظریه پردازان) با تاکید بر هسته پردازش های هوشمند

هوش مصنوعی چیست؟

چهار دسته از تعاریف هوش مصنوعی

فرایندهای فکر و استدلال :

عاقلانه فکر کردن	مانند انسان فکر کردن

منجر به بروز رفتار:

عاقلانه عمل کردن	مانند انسان عمل کردن

سیستم های که مانند انسان عمل می کنند

Acting humanly

- ✓ هنر ساخت ماشین هایی که کارهایی را انجام می دهند که آن کارها توسط انسان با فکر کردن انجام می شوند.
- ✓ مطالعه برای ساخت کامپیوترها برای انجام کارهایی که فعلاً انسان آنها را بهتر انجام میدهد.

آزمایش تورینگ

- در سال ۱۹۵۰ آلن تورینگ (Alain Turing)، ریاضی دان انگلیسی، معیار سنجش رفتار یک ماشین هوشمند را چنین بیان داشت:

- سزاوارترین معیار برای هوشمند شمردن یک ماشین، اینست که آن ماشین بتواند انسانی را توسط یک پایانه (تله تایپ) به گونه ای بفریبد که آن فرد متقاعد گردد با یک انسان روبروست.

- در این آزمایش شخصی از طریق ۲ عدد پایانه (کامپیوتر یا تله تایپ) که امکان برقراری ارتباط (Chat) را برای وی فراهم می کنند با یک انسان و یک ماشین هوشمند، بطور همزمان به پرسش و پاسخ می پردازد. در صورتی که وی نتواند ماشین را از انسان تشخیص دهد، آن ماشین، هوشمند است.

تست تورینگ

B

A

کدام انسان است؟
A یا B

Email: ComputerCollege_Fam@Yahoo.Com

عاقلانه فکر کردن

Think rationally

- ✓ مطالعه توانایی های ذهنی از طریق مدل های محاسباتی (منطق گرایی)
- ✓ مطالعه محاسباتی که منجر به درک و استدلال می شود.

عاقلانه عمل کردن Act rationally

طوری عمل کند که بهترین نتیجه را ارائه دهد

✓هوش محاسباتی، مطالعه طراحی عامل های هوشمند است

مبانی هوش مصنوعی

فلسفه: منطق، استدلال، ناشی شدن تفکر از مغز

روان شناسی: تطبیق، اثر طبیعی ادراک و تاثیر آن بر محیط

ریاضیات: نمایش رسمی الگوریتمها، محاسبات، تصمیم پذیری و تصمیم ناپذیری، احتمال

زبان شناسی: علم ارائه، گرامر

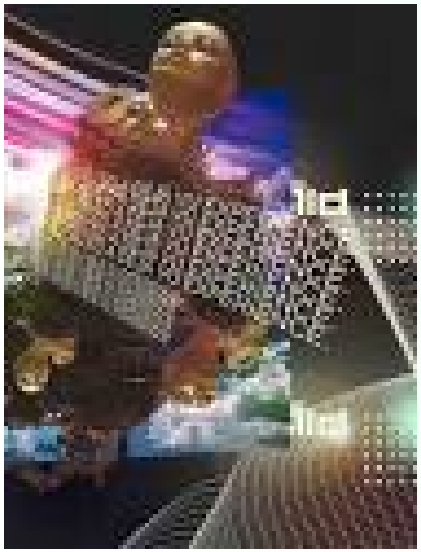
فیزیکی، مبانی یادگیری، زبان و عقلانیت

Email: ComputerCollege_Fam@Yahoo.Com

فلسفه (۴۲۸ قبل از میلاد مسیح تا کنون)

- آیا قوانین رسمی می توانند نتایج معتبری را ارائه کنند؟
- تفکر ذهنی چگونه از مغز فیزیکی ناشی می شود؟
- دانش از کجا می آید؟
- دانش چگونه منجر به فعالیتی می شود؟

ریا ضیات (۸۰۰ میلادی تا کنون)



- قوانین رسمی برای اخذ نتایج معتبر کدامند؟
- چه چیزی می تواند محاسبه شود؟
- چگونه می توان با اطلاعات نا مطمئن استدلال کرد؟

اقتصاد (۱۷۷۶ تا کنون)

- چگونه تصمیم هایی اتخاذ می کنیم تا سود را به حداکثر برسانیم؟
- اگر افراد دیگر در این مسیر گام بر نمی دارند، چگونه باید این کار را انجام داد؟
- اگر سود دهی در دراز مدت رخ دهد، چه باید کرد؟

علوم عصبی (۱۸۶۱ تا کنون)

- مغز چگونه اطلاعات را پردازش می کند؟

روان شناسی (۱۸۷۹ تا کنون)

- انسان ها و حیوانات چگونه فکر می کنند؟

مهندسی کامپیوتر (۱۹۴۰ تا کنون)

- چگونه می توان کامپیوتر کارامدی ساخت؟

نظریه کنترل و سیبرنتیک(۱۹۴۸ تا کنون)

- محصولات مصنوعی چگونه می توانند تحت کنترل خود ساخته شوند؟

زبانی (۱۹۵۷ تا کنون)

- زبان چگونه با تفکر ارتباط دارد؟

Email: ComputerCollege_Fam@Yahoo.Com

مباحث هوش مصنوعی پیش از بوجود آمدن علوم الکترونیک، توسط فلاسفه و ریاضی دانانی نظیر بول (Boole) که اقدام به ارائه قوانین و نظریه‌هایی در باب منطق نمودند، مطرح شده بود.

در سال ۱۹۴۳، با اختراع رایانه‌های الکترونیکی، هوش مصنوعی، دانشمندان را به چالشی بزرگ فراخواند. بنظر می‌رسید، فناوری در نهایت قادر به شبیه‌سازی رفتارهای هوشمندانه خواهد بود.

با وجود مخالفت گروهی از متفکرین با هوش مصنوعی که با دیده تردید به کارآمدی آن می‌نگریستند تنها پس از چهار دهه، شاهد تولد ماشین های شطرنج باز و دیگر سامانه‌های هوشمند در صنایع گوناگون هستیم.

تاریخچه هوش مصنوعی

نام هوش مصنوعی در سال ۱۹۶۵ میلادی به عنوان یک دانش جدید ابداع گردید. البته فعالیت در زمینه این علم از سال ۱۹۶۰ میلادی شروع شده بود. بیشتر کارهای پژوهشی اولیه در هوش مصنوعی بر روی انجام ماشینی بازی‌ها و نیز اثبات قضیه‌های ریاضی با کمک رایانه‌ها بود. در آغاز چنین به نظر می‌آمد که رایانه‌ها قادر خواهند بود چنین اموری را تنها با بهره گرفتن از تعداد بسیار زیادی کشف و جستجو برای مسیرهای حل مسئله و سپس انتخاب بهترین آن‌ها به انجام رسانند.

این اصطلاح(هوش مصنوعی) برای اولین بار توسط جان مکارتی (John Mccorthy) که از آن به‌عنوان پدر «علم و دانش تولید ماشین های هوشمند» یاد می‌شود استفاده شد. با این عنوان می‌توان به هویت هوشمند یک ابزار مصنوعی اشاره کرد. (ساخته دست بشر، غیر طبیعی، مصنوعی). نام هوش مصنوعی در سال ۱۹۶۵ میلادی به عنوان یک دانش جدید ابداع گردید.

حال آنکه AI به عنوان یک اصطلاح عمومی پذیرفته شده که شامل محاسبات هوشمندانه و ترکیبی (مرکب از مواد مصنوعی) می‌باشد.

تاریخچه هوش مصنوعی

۱۹۴۳: اولین کار مرتبط با هوش مصنوعی توسط وارن مک کولوچ و والتر پیتز در زمینه پیشنهاد مدلی از نرون مصنوعی

۱۹۴۹: دونالد هِبا قاعده یادگیری را برای اصلاح تقویت اتصالات بین نرون ها تعریف نمود.

۱۹۵۰: مقاله تورینگ تحت عنوان «محاسبات ماشینی و هوشمند»

۱۹۵۱: سخت اولین کامپیوتر شبکه عصبی توسط هینکسی و ادموندز

۱۹۵۶: تشکیل کارگروه پژوهشگران علاقه مند به نظریه ماشین های خودکار، شبکه‌های عصبی و مطالعه هوش مصنوعی توسط جان مک کارتی در کالج دورتموند

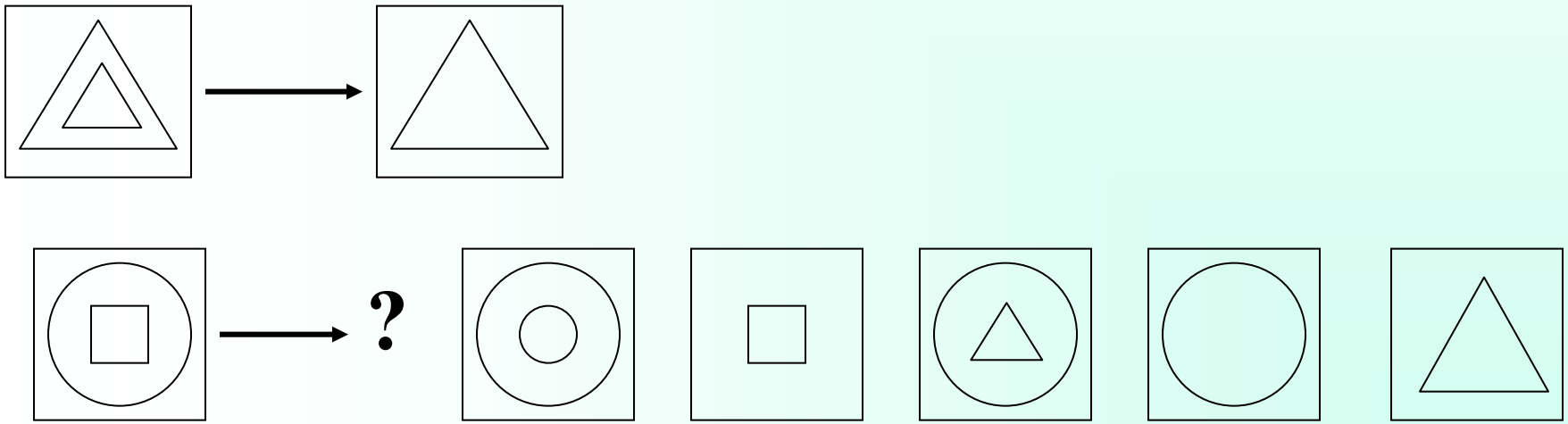
تاریخچه هوش مصنوعی

۱۹۵۸: تعریف زبان LISP توسط جان مک کارتی

۱۹۵۹: ساخت برنامه GTP توسط هربرت جلونتر قضایا را با اصل موضوعات مشخص ثابت کرد.

۱۹۶۳: برنامه SAWT مربوط به جیمز اسلاگل که قادر به حل مسائل انتگرال گیری فرم بسته در سطح ریاضیات سال اول کالج بود.

۱۹۶۸: برنامه NALOGY مربوط به تام ایوانز مسئله های مشابهت هندسی را حل کرد



AI به صنعت تبدیل می شود

۱۹۸۱: طرح پروژه نسل پنجم برای ساخت رایانه های هوشمند توسط ژاپن

۱۹۸۲: اولین سیستم خبره تجاری در انتخاب قطعات برای سیستم های رایانه ای

۱۹۸۶: برگشت به شبکه های عصبی

AI به علم تبدیل می شود

۱۹۸۷ تا کنون: تحولات عمده در

- ربات ها
- بینایی رایانه ای
- یادگیری ماشینی
- نمایش دانش

ظهور عامل های هوشمند

۱۹۹۵ تا کنون:

- اینترنت یکی از بهترین محیط ها برای عامل های هوشمند است.

با وجودی که برآورده سازی نیازهای صنایع نظامی، مهم‌ترین عامل توسعه و رشد هوش مصنوعی بوده‌است، هم اکنون از فراورده‌های این شاخه از علوم در

- صنایع پزشکی
- رباتیک
- پیش بینی وضع هوا
- نقشه‌برداری و شناسایی عوارض
- تشخیص صدا
- تشخیص گفتار
- دست خط
- بازی‌ها
- نرم افزارهای رایانه‌ای استفاده می‌شود.

فصل دوم:

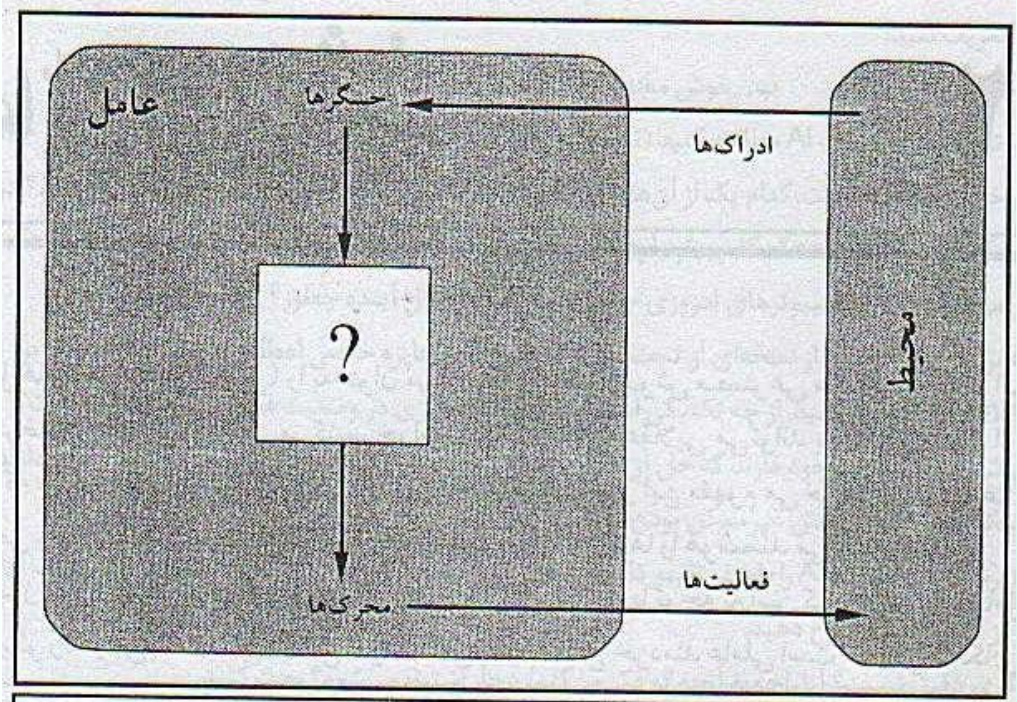
عامل های هوشمند

Intelligent Agent

Email: ComputerCollege_Fam@Yahoo.Com

- عامل

هر چیزی است که قادر محیط خود را از طریق حسگرها (sensors) درک کند و از طریق محرک ها (actuators) عمل نماید.



شکل ۱-۲ عامل‌ها از طریق حسگرها و محرک‌ها با محیط‌ها تعامل دارند.

عامل ها و محیط ها

- ادراک(دریافت ها): اطلاعات فرستاده شده به حسگرهای یک عامل می باشد. مانند: نور؛ صدا

- حسگرها: روش های یک عامل برای جمع آوری اطلاعات در مورد محیطش می باشد. مانند: چشم ها، گوش ها

- محرک ها: عامل به وسیله محرک ها بر روی محیط اثر می گذارد. مانند: تایر، بازوها

- عملکرد: کار یا عملی است که بر روی محیط انجام می شود

- دنباله ادراک عامل

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

انتخاب فعالیتی توسط عامل در هر لحظه، می تواند به کل دنباله ادراک تا آن زمان بستگی داشته باشد.

از نظر ریاضی، رفتار عامل توسط تابع عامل توصیف می شود که هر دنباله ادراک را به یک فعالیت نقش می کند.

- می توان رفتار عامل را با یک تابع F توصیف نمود

عملکرد=F(تاریخچه ادراک،ادراک فعلی)

بکارگیری این تابع، کار یک برنامه ی عامل می باشد

Email: ComputerCollege_Fam@Yahoo.Com

تابع عامل

رفتار عامل توسط تابع عامل توصیف می شود که هر دنباله ادراک را به یک فعالیت نقش می کند.

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

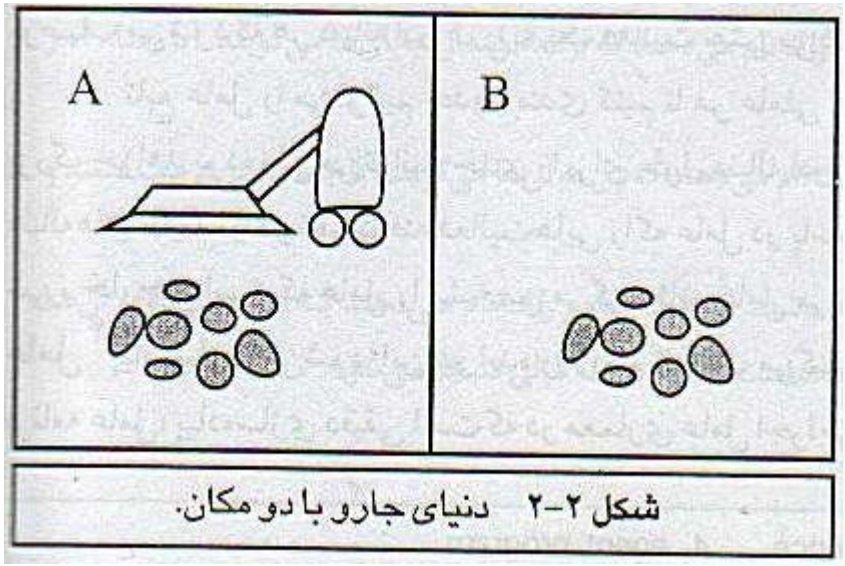
فعالیت ⟵ دنباله ادراک : تابع عامل

Email: ComputerCollege_Fam@Yahoo.Com

- دنیای جارو

این دنیای ویژه فقط دو مکان **A** و **B** دارد. عامل جارو درک می کند که در کدام مربع قرار دارد و آیا گرد و خاک در آن مربع وجود دارد یا خیر.

می تواند به چپ و راست برود. گرد و خاک را مکش کند یا هیچ کاری انجام ندهد.



شکل ۲-۲ دنیای جارو با دو مکان.

تابع عامل:

اگر مربع فعلی کثیف است، آن را تمیز کن و گرنه به مربع دیگر برو

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

شکل ۳-۲ بخشی از جدول‌بندی تابع عامل برای دنیای جارو شکل ۲-۲.

Email: ComputerCollege_Fam@Yahoo.Com

•عامل خردمند

برای هر دنباله ادراک ممکن، عامل خردمند باید فعالیتی را انتخاب کند که انتظار می رود معیار کارایی اش را به حداکثر برساند. این کار با توجه به شواهدی که از طریق دنباله ادراک به دست می آید و دانش درونی عامل ، صورت می گیرد.

معیار کارایی: معیاری برای موفقیت رفتار عامل است.

- ماهیت محیط ها

محیط های وظیفه (task environment): مسائلی هستند که عامل های خردمند باید آنها را حل کنند.

تعیین محیط وظیفه

محیط وظیفه شامل معیار کارایی، محیط، حسگرها و محرکها می باشد.

Performance, Environment, Actuators, Sensors (PEAS)

نوع عامل	معیار کارایی	محیط	محرک‌ها	حسگرها
راننده تاکسی				

شکل ۴-۲ توصیف PEAS مربوط به محیط وظیفه تاکسی خودکار.

نوع عامل	معیار کارایی	محیط	محرک‌ها	حسگرها
سیستم تشخیص پزشکی	سلامتی بیمار، کمترین هزینه، دادخواهی	بیمار، بیمارستان، کارکنان	نمایش پرسش‌ها، تست‌ها، تشخیص‌ها، مداوا، مراجعه	صفحه کلید و ورود علائم بیماری، جست‌وجو، پاسخ بیمار
سیستم تحلیل تصویر ماهواره‌ای	دسته‌بندی صحیح تصاویر	پیوند روبه‌پایین از ماهواره مداری	نمایش صحنه طبقه‌بندی شده	آرایه‌های پیکسل رنگی
روبات جابه‌جا کننده قطعات	درصد قطعات در جعبه‌های مناسب	تسمه نقاله با قطعات، جعبه‌ها	بازو و دست متصل	حسگرهای زاویه‌ای متصل، دوربین
کنترلگر پالایشگاه	افزایش خلوص، محصول، ایمنی	پالایش، اپراتورها	مقادیر، پمپ‌ها، اجاق‌ها، نمایشگرها	حسگرهای شیمیایی، دما، فشار
آموزش انگلیسی محاوره‌ای	افزایش امتیاز دانش‌آموزان در آزمون	مجموعه‌ای از دانش‌آموزان، تست عاملیت	نمایش تمرین‌ها، پیشنهادات، تصحیح آزمون‌ها	ورودی صفحه کلید

شکل ۵-۲ نمونه‌هایی از انواع عامل و توصیف PEAS آن‌ها.

Email: ComputerCollege_Fam@Yahoo.Com

خواص محیط های وظیفه

- کاملا قابل مشاهده، در مقابل قابلیت مشاهده جزیی
  - اگر حسگرهای عامل، در هر زمان امکان دستیابی کامل به محیط را فراهم کنند، آن محیط کاملا قابل مشاهده است
- قطعی در مقابل غیر قطعی
  - اگر حالت بعدی محیط کاملا توسط حالت فعلی و عملی که عامل در حال انجام آن است، تعیین شود، آن محیط قطعی است.

دوره ای در مقابل ترتیبی

- در محیط های دوره ای، انتخاب فعالیت در هر رویداد، به خود رویداد بستگی دارد. مثل قطعات معیوب خط مونتاژ
- در محیط های ترتیبی، تصمیم فعلی می تواند در تمام تصمیمات بعدی موثر باشد. مثل شطرنج

ایستا در مقابل پویا

- اگر محیط در در زمان تصمیم گیری عامل تغییر کند، می گوییم آن محیط برای آن عامل پویا و گرنه ایستاست.
- اگر محیط با گذر زمان تغییر نکند، اما معیار کارایی تغییر کند، محیط نیمه پویا است.

•گسسته در مقابل پیوسته

- تمایز بین گسسته و پیوسته می تواند به حالت محیط، اداره کردن زمان، و به ادراکات و فعالیت های عامل، اِعمال شود.

•تک عاملی در مقابل چند عاملی

۱. چند عاملی رقابتی: شطرنج

۲. چند عاملی همیاری: رانندگی تاکسی

Email: ComputerCollege_Fam@Yahoo.Com

محیط وظیفه	قابل‌مشاهده	قطعی	رویدادی	ایستا	گسسته	عامل
جدول کلمات متقاطع	کامل	قطعی	ترتیبی	ایستا	گسسته	تک
شطرنج با ساعت	کامل	راهبردی	ترتیبی	ایستا	گسسته	چندتایی
پوکر	جزئی	راهبردی	ترتیبی	ایستا	گسسته	چندتایی
تخته نرد	کامل	غیرقطعی	ترتیبی	ایستا	گسسته	چندتایی
رانندگی تاکسی	جزئی	غیرقطعی	ترتیبی	پویا	پیوسته	تک
تشخیص پزشکی	جزئی	غیرقطعی	ترتیبی	پویا	پیوسته	چندتایی
تحلیل تصویر	کامل	قطعی	رویدادی	نیمه‌پویا	پیوسته	تک
روبات انتقال قطعات	جزئی	غیرقطعی	رویدادی	پویا	پیوسته	تک
کنترلگر پالایشگاه	جزئی	غیرقطعی	ترتیبی	پویا	پیوسته	تک
آموزش انگلیسی پویا	جزئی	غیرقطعی	ترتیبی	پویا	پیوسته	چندتایی

شکل ۶-۲ نمونه‌هایی از محیط‌های وظیفه و ویژگی‌های آن‌ها.

Email: ComputerCollege_Fam@Yahoo.Com

ساختار عامل ها

کار هوش مصنوعی، طراحی برنامه عامل است که تابع عامل را پیاده سازی می کند.

تابع عامل، ادراکات را به فعالیت ها نگاشت می کند.

انواع عامل ها

- عامل های واکنشی ساده
- عامل های واکنشی مدل گرا
- عامل های هدف گرا
- عامل های سودمند

Email: ComputerCollege_Fam@Yahoo.Com

ساختار عامل ها

برنامه + معماری = عامل

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

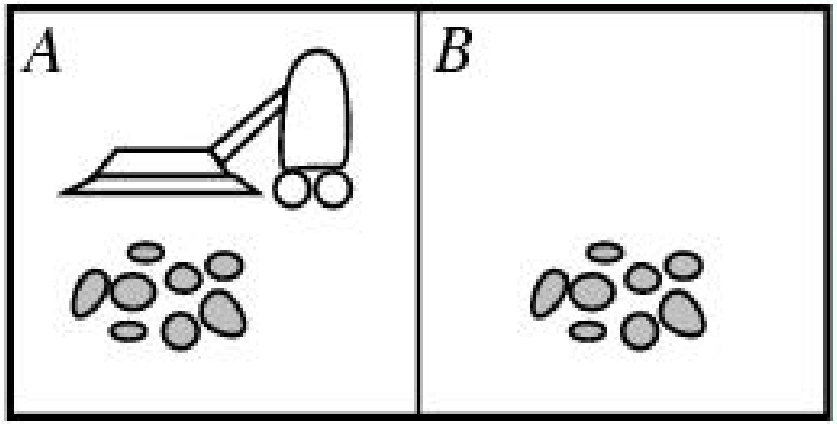
تابع عامل، ادراکات را به فعالیت ها نگاشت می کند.

برنامه های عامل

- عامل های واکنشی ساده
- عامل های واکنشی مدل گرا
- عامل های هدف گرا
- عامل های سودمند

Email: ComputerCollege_Fam@Yahoo.Com

مثالی از عامل واکنشی ساده در دنیای جاروبرقی



- تصمیم گیری آن بر اساس مکان فعلی و کثیف بودن آن مکان صورت میگیرد
- انتخاب فعالیت بر اساس موقعیت شرطی:

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

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

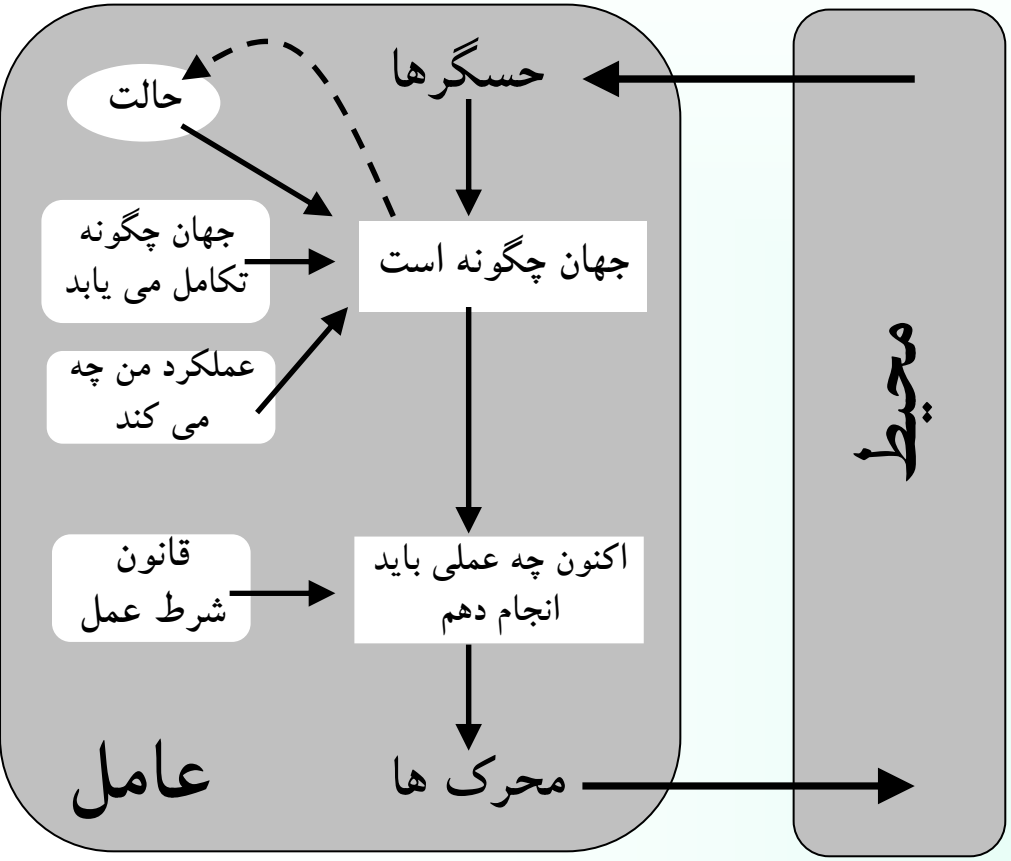
عامل های واکنشی مدل گرا Reflex Agent With State

مؤثرترین راه برای قابلیت مشاهده جزئی این است که عامل، بخشی از دنیایی را که فعلاً می بیند ردیابی کند.

اطلاعات عامل به تنهائی در مورد مشاهده پذیری جزئی کافی نمی باشد، لازم است که جریان تغییرات جهان را نگهداری نماییم و تکامل ها، به طور مستقل از عامل یا سبب شده توسط عملکرد عامل می باشند

Email: ComputerCollege_Fam@Yahoo.Com

عامل های واکنشی مدل گرا



- استفاده از دانش "چگونگی عملکرد جهان" که مدل نام دارد
- عامل بخشی از دنیایی را که فعلا می بیند ردیابی می کند
- 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

عامل های واکنشی مدل گرا

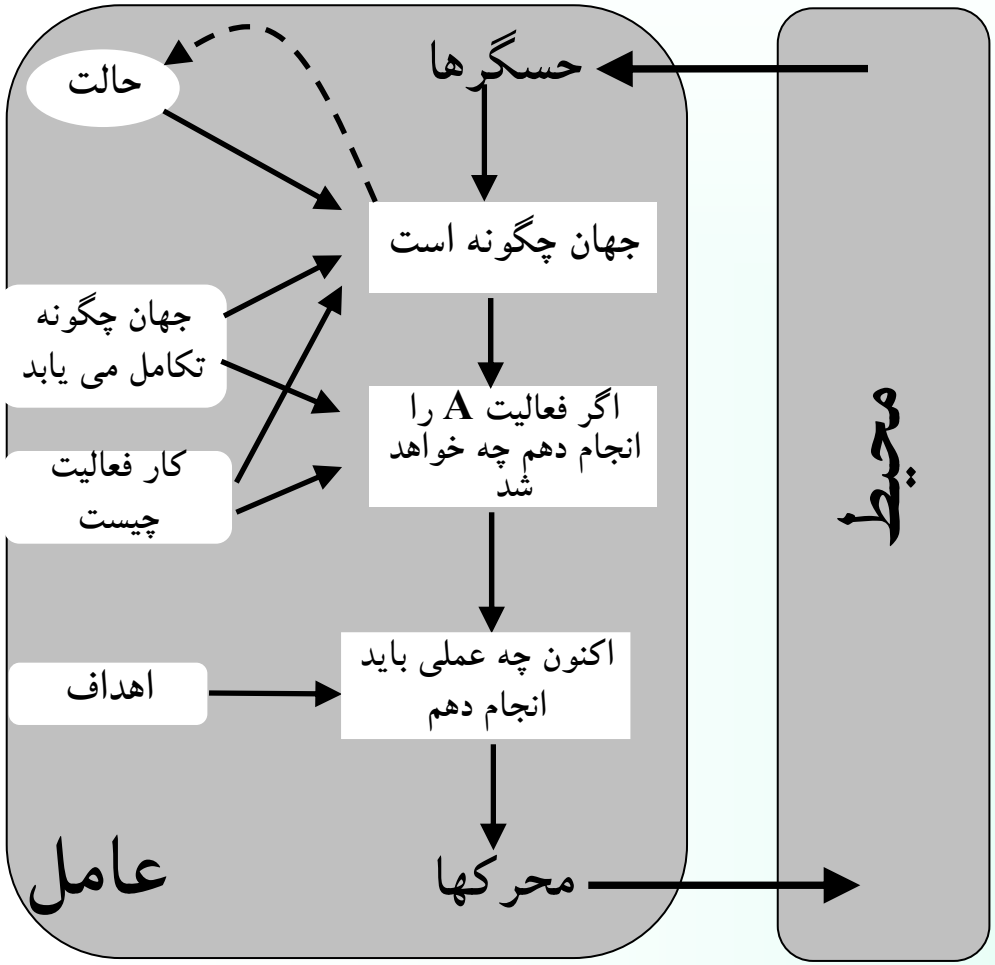
```
function REFLEX-AGENT-WITH-STATE(percept) returns an action
  static: state, a description of the current world state
          rules, a set of condition–action rules
          action, the most recent action, initially none

  state ← UPDATE-STATE(state, action, percept)
  rule ← RULE-MATCH(state, rules)
  action ← RULE-ACTION[rule]
  return action
```

شکل ۲-۱۲ عامل واکنشی مدل‌گرا. حالت فعلی جهان را با یک مدل داخلی ردیابی، و همانند عامل واکنشی ساده، فعالیتی را انتخاب می‌کند.

Email: ComputerCollege_Fam@Yahoo.Com

Goal-Based Agent

عامل های هدف گرا



- این عامل علاوه بر توصیف حالت فعلی، برای انتخاب موقعیت مطلوب نیازمند اطلاعات هدف نیز می باشد
- جست و جو و برنامه ریزی، دنباله ای از فعالیت ها را برای رسیدن عامل به هدف، پیدا می کند
- این نوع تصمیم گیری همواره آینده را در نظر دارد و با قوانین شرط عمل تفاوت دارد
- این نوع عامل کارایی چندانی ندارد، اما قابلیت انعطاف بیشتری دارد
- استدلال هدف گرا برای محیط های ترتیبی بسیار مفید است: شطرنج، تاکسی و ...

Email: ComputerCollege_Fam@Yahoo.Com

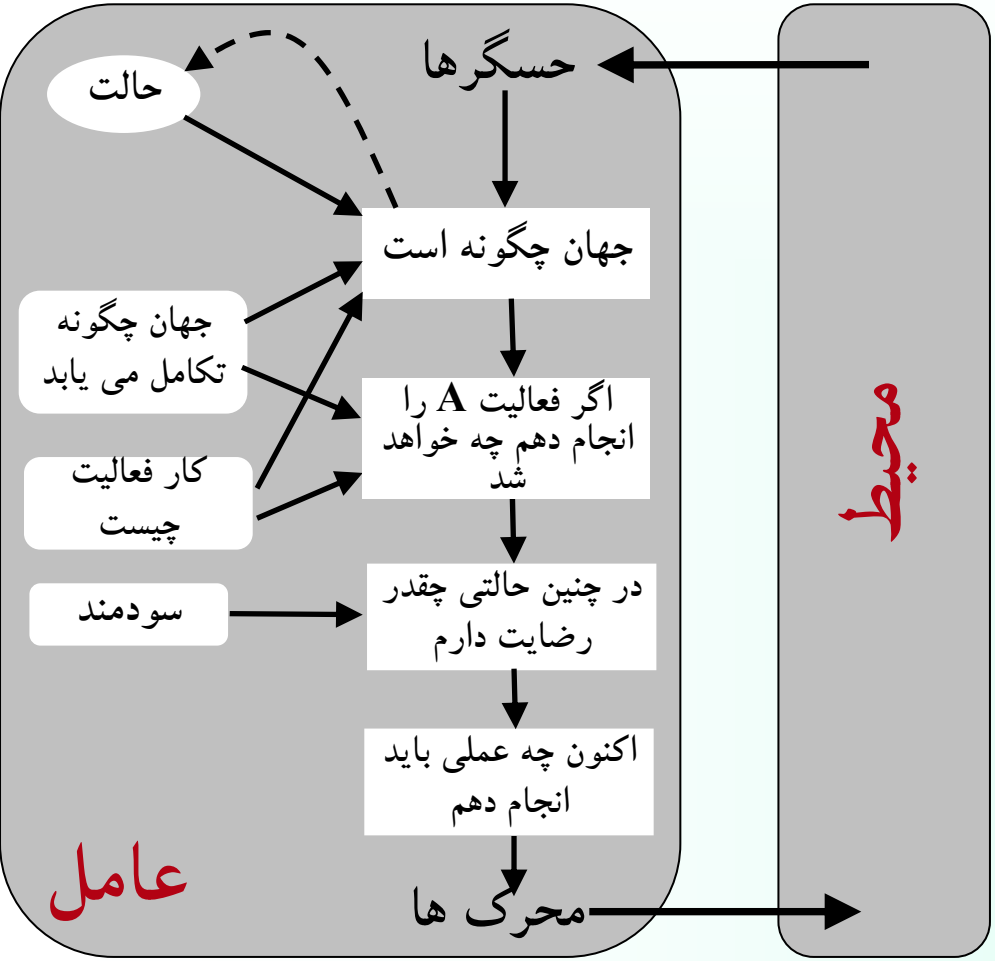
عامل های سودمند Utility-Based Agents

اگر یک حالت دنیا به حالت دیگر ترجیح داده شود، آن حالت برای عامل سودمندتر است.

تابع سودمندی، حالت (یا دنباله ای از حالت ها) را به یک عدد حقیقی نگاشت می کند که درجه رضایت را توصیف می نماید.

Email: ComputerCollege_Fam@Yahoo.Com

عامل های سودمند



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

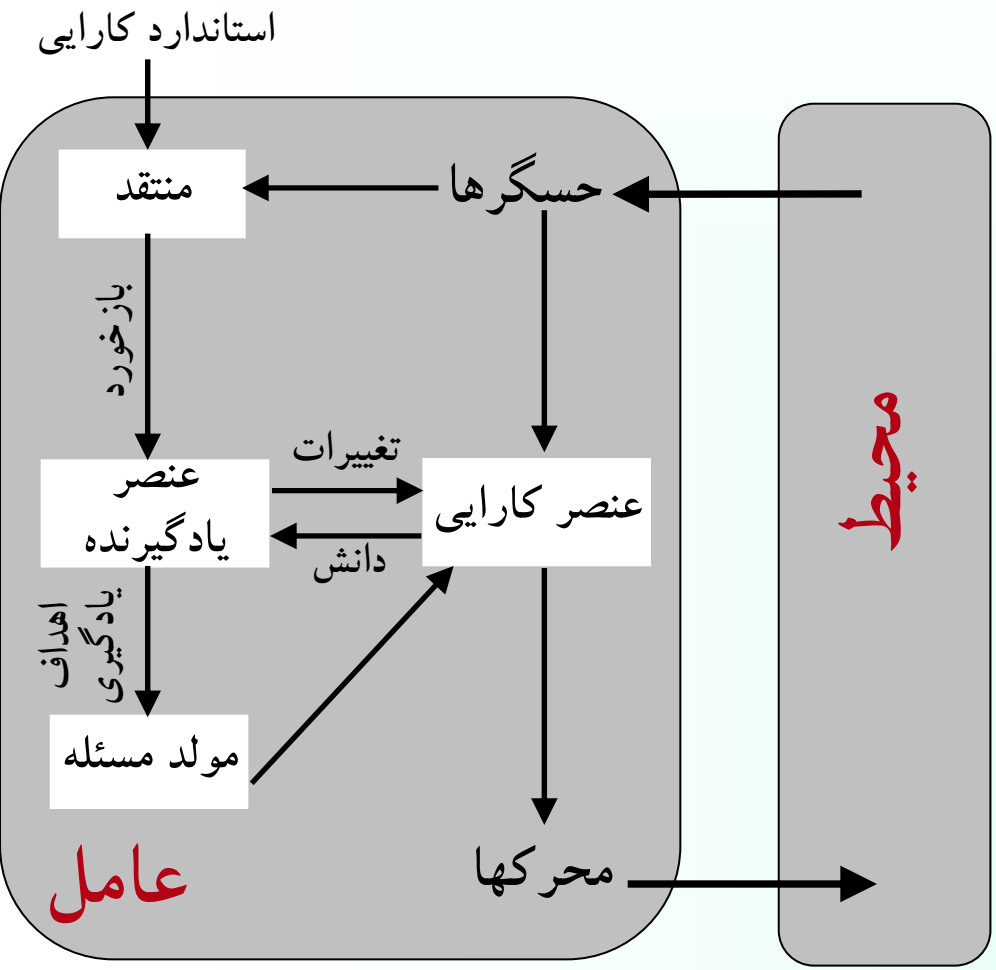
Email: ComputerCollege_Fam@Yahoo.Com

عامل های یادگیرنده

عامل یادگیرنده می تواند به چهار مولفه مفهومی تقسیم شود:

- عنصر یادگیرنده
- عنصر کارایی
- مخالفین
- مولد مساله

عامل های یادگیرنده



- عنصرِیادگیرنده مسئول ایجاد بهبودها
- عنصر کارایی مسئول انتخاب فعالیت های خارجی
- منتقد مشخص می کند که یادگیرنده با توجه به استانداردهای کارایی چگونه عمل می کند
- مولد مسئله مسئول پیشنهاد فعالیت هایی است که منجر به تجربیات آموزنده جدیدی می شود

Email: ComputerCollege_Fam@Yahoo.Com

هوش مصنوعی

فصل سوم

حل مسئله با جستجو

Problem Solving and Search

Email: ComputerCollege_Fam@Yahoo.Com

فهرست این فصل

- عامل های حل مسئله
- مسئله
- اندازه گیری کارایی حل مسئله
- جستجوی ناآگاهانه
- اجتناب از حالت های تکراری
- جستجو با اطلاعات ناقص

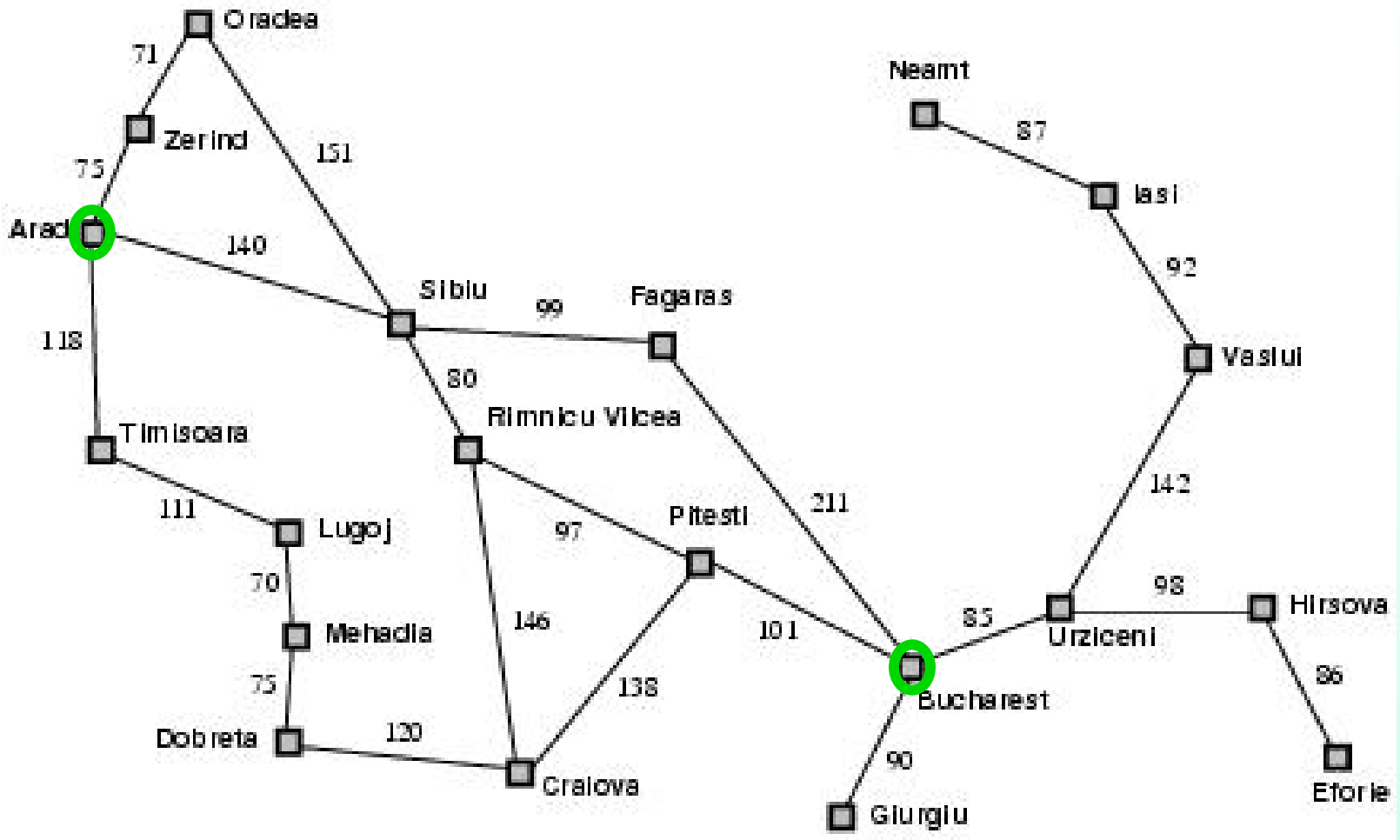
عامل های حل مسئله

چهار گام اساسی برای حل مسائل

- **فرموله کردن هدف:** وضعیت های مطلوب نهایی کدامند؟
- **فرموله کردن مسئله:** چه فعالیت ها و وضعیت هایی برای رسیدن به هدف موجود است؟
- **جستجو:** انتخاب بهترین دنباله از فعالیت هایی که منجر به حالاتی با مقدار شناخته شده می شود.
- **اجرا:** وقتی دنباله فعالیت مطلوب پیدا شد، فعالیت های پیشنهادی آن می تواند اجرا شود.

Email: ComputerCollege_Fam@Yahoo.Com

مثال: نقشه رومانی



Email: ComputerCollege_Fam@Yahoo.Com

مثال: نقشه رومانی

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

Email: ComputerCollege_Fam@Yahoo.Com

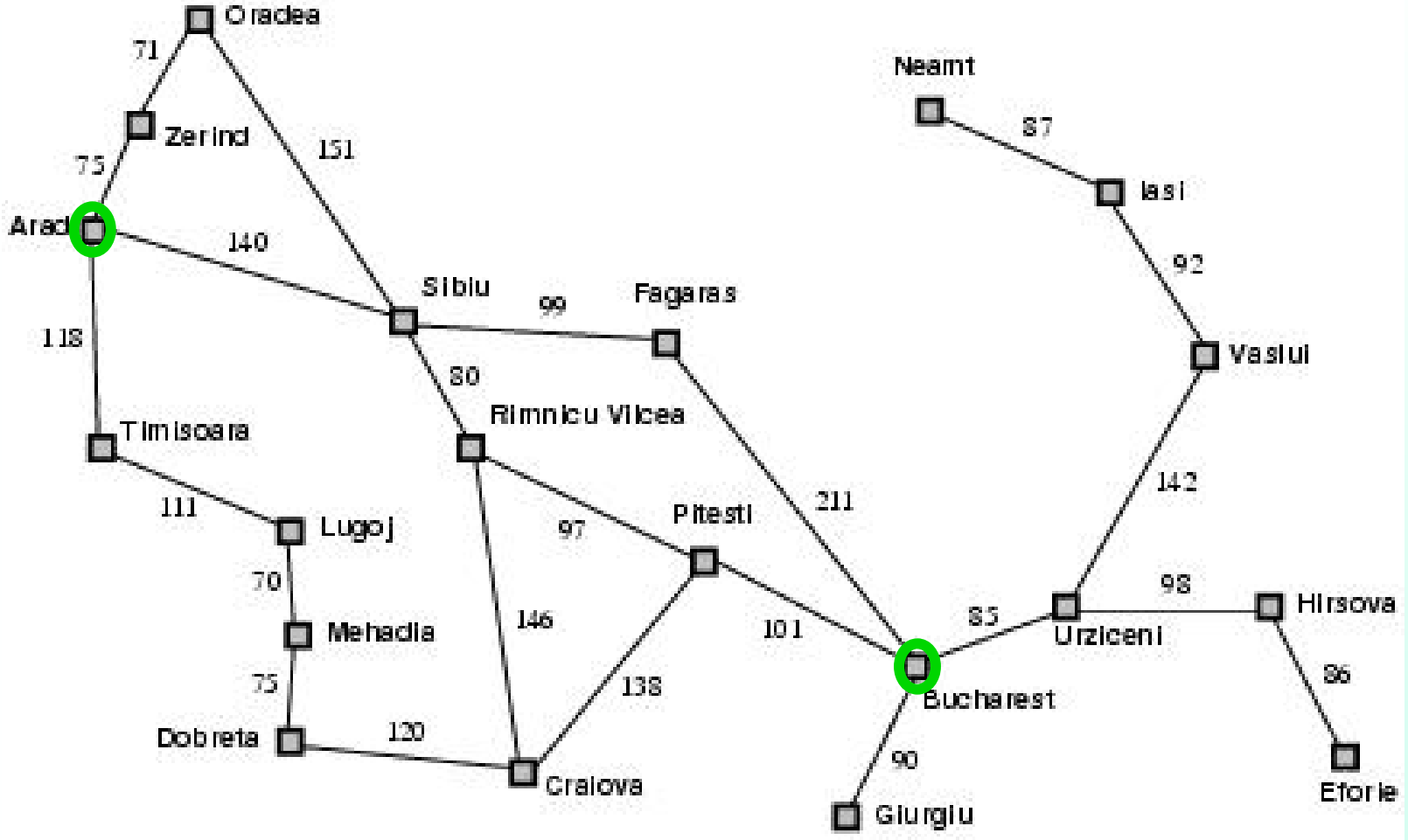
مسئله

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

تابع جانشین + حالت اولیه = فضای حالت

Email: ComputerCollege_Fam@Yahoo.Com

مثال: نقشه رومانی



Email: ComputerCollege_Fam@Yahoo.Com

- آزمون هدف: تعیین می کند 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 حالت ها که دنباله ای از فعالیت ها را به هم متصل می کند.
  - در مثال رومانی: Arad, Sibiu, Fagaras یک مسیر است

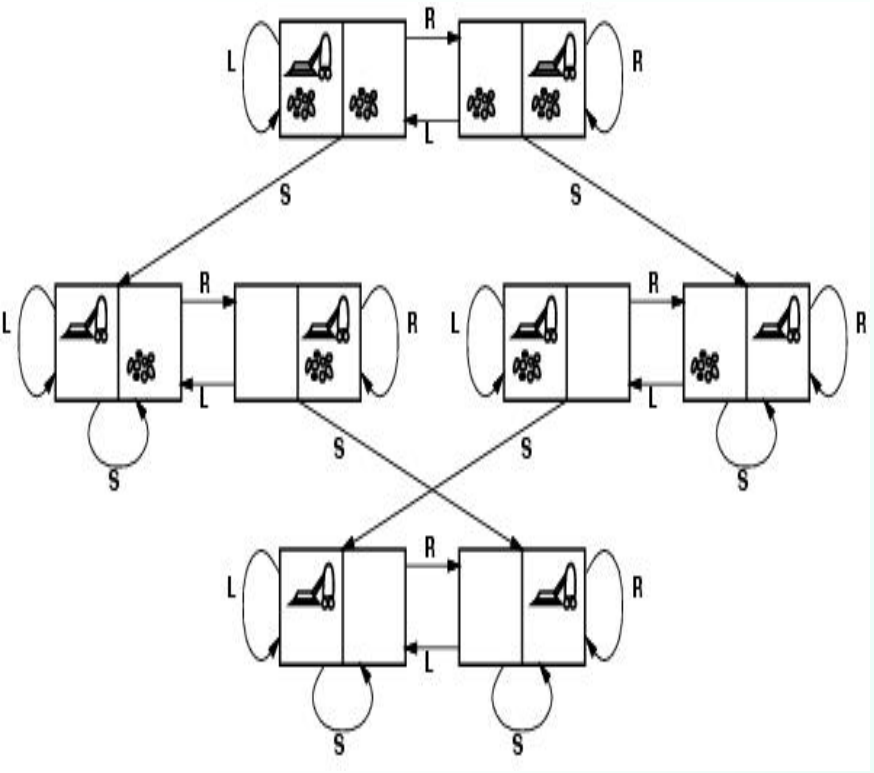
- هزینه مسیر: برای هر مسیر یک هزینه عددی در نظر می گیرد.
  - در مثال رومانی: طول مسیر بین شهرها بر حسب کیلومتر

راه حل مسئله مسیری از حالت اولیه به حالت هدف است

راه حل بهینه کمترین هزینه مسیر را دارد

Email: ComputerCollege_Fam@Yahoo.Com

مثال ۲: دنیای جارو برقی



حالت ها: دو مکان که هر یک ممکن است کثیف یا تمیز باشند. لذا ۸ = ۲^۲*۲حالت در این جهان وجود دارد

حالت اولیه: هر حالتی می تواند به عنوان حالت اولیه طراحی شود

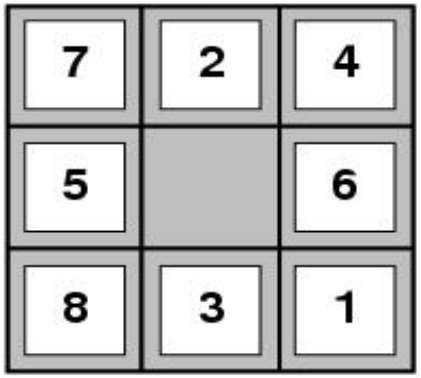
تابع جانشین: حالت های معتبر از سه عملیات: راست، چپ، مکش

آزمون هدف: تمیزی تمام مربع ها

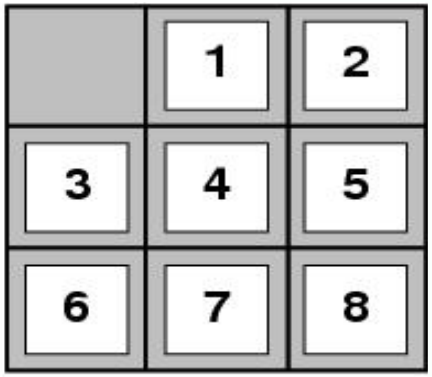
هزینه مسیر: تعداد مراحل در مسیر

Email: ComputerCollege_Fam@Yahoo.Com

مثال: معمای ۸



Start State



Goal State

حالت ها: مکان هر هشت خانه شماره دار و خانه خـالی در یکـی از ۹ خانه

حالت اولیه: هر حالتی را می توان به عنوان حالـت اولیـه در نظـر گرفت

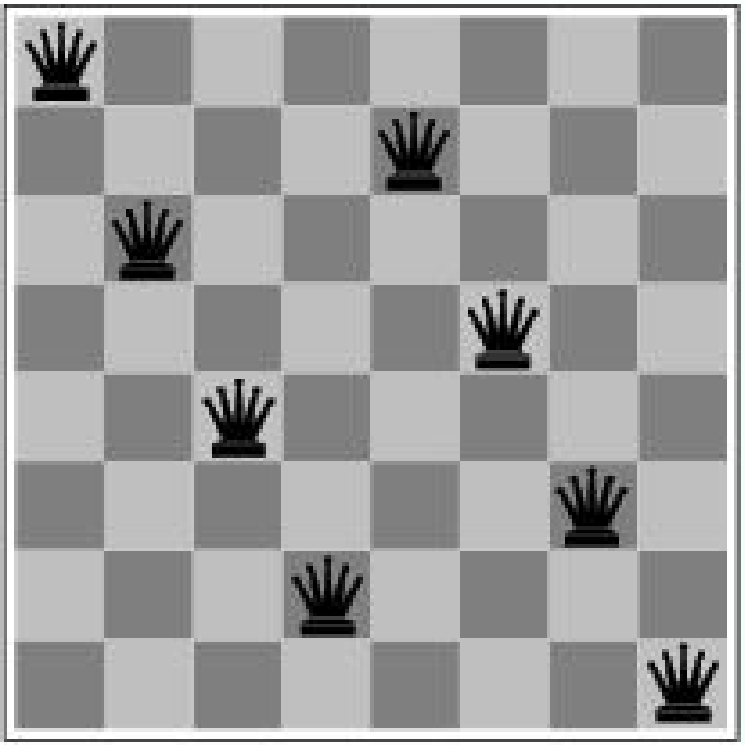
تابع جانشین: حالت های معتبر از چهار عمل، انتقال خانه خالی به چپ، راست، بالا یا پایین

آزمون هدف: بررسی می کند که حالتی که اعداد به ترتیـب چیـده شده اند(طبق شکل روبرو) رخ داده یا نه

هزینه مسیر: برابر با تعداد مراحل در مسیر

Email: ComputerCollege_Fam@Yahoo.Com

مثال: مسئله ۸ وزیر



فرمول بندی افزایشی

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

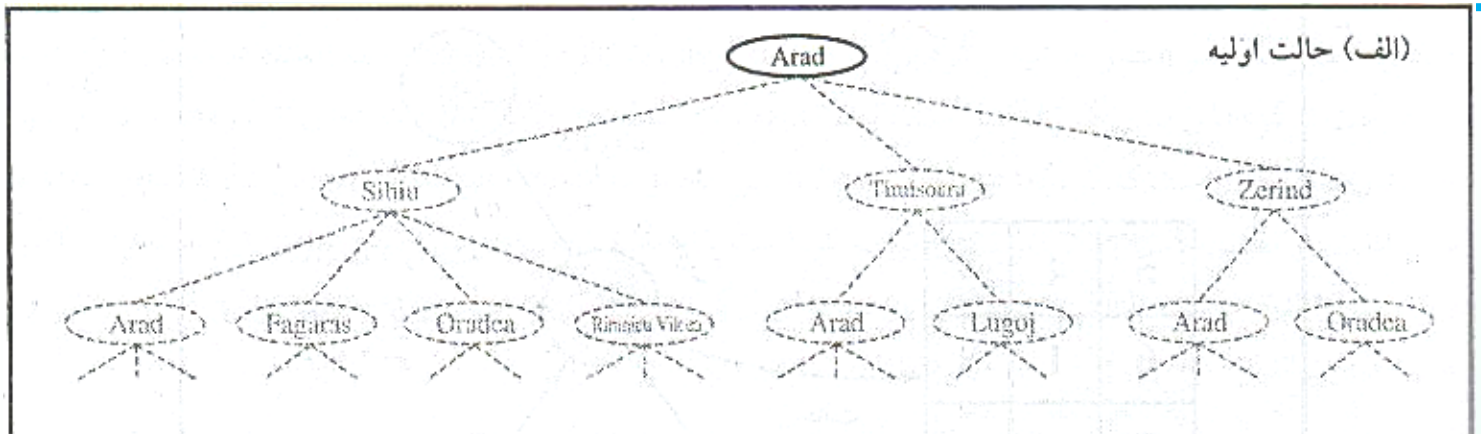
حالت اولیه: هیچ وزیری در صفحه نیست

تابع جانشین: وزیری را به خانه خالی اضافه می کند

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

Email: ComputerCollege_Fam@Yahoo.Com

جستجو برای راه حل ها



STATE: حالتی در فضای حالت که گره متناظر با آن است.

PARENT-NODE: گره ای در درخت جستجو که این گره را تولید کرده است

ACTION: فعالیتی که به والد اعمال شد تا این گره تولید شود

PATH-COST: هزینه مسیری از حالت اولیه به این گره که معمولا با **g(n)** نمایش داده می شود

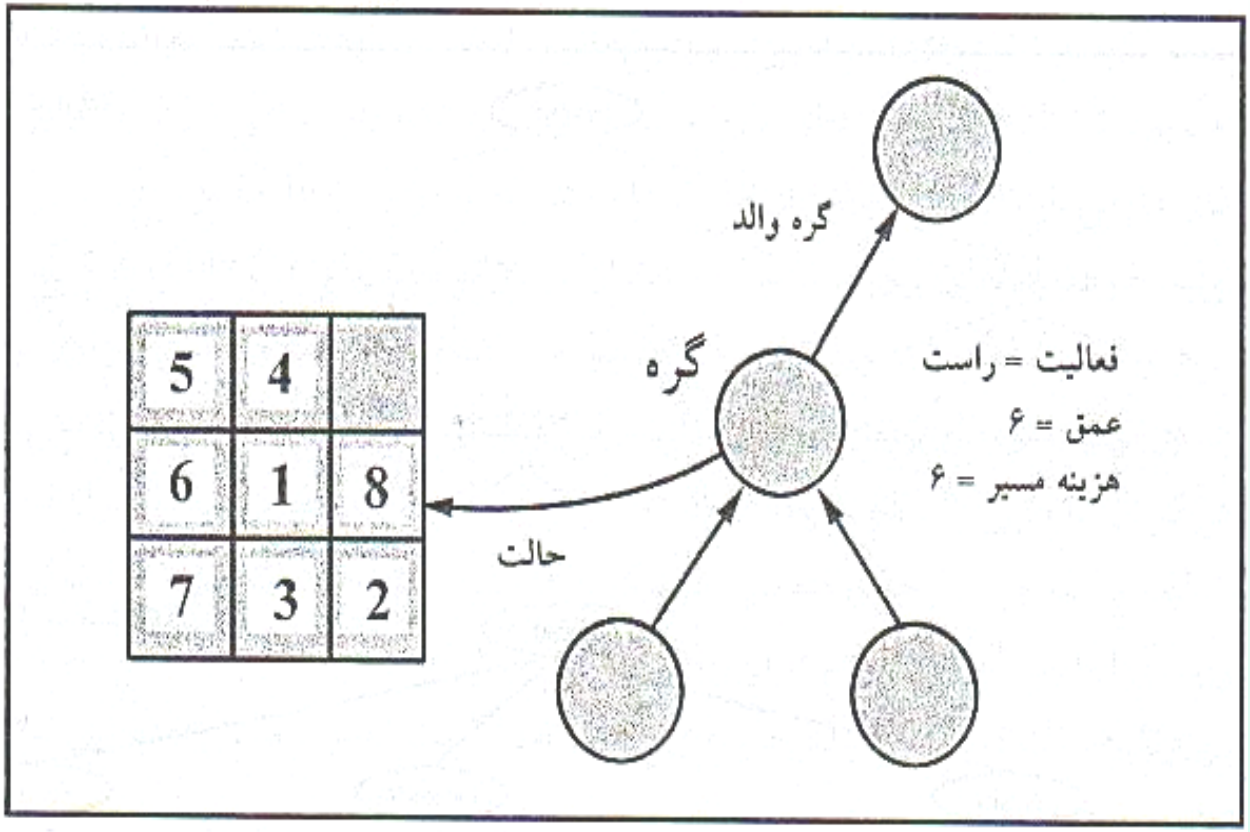
DEPTH: تعداد مراحل موجود در مسیر که از حالت اولیه فاصله دارد

حاشیه: مجموعه ای از گره ها که تولید شده اند اما بسط داده نشده اند

برگ : یک عنصر از حاشیه

ساختمان داده گره ها در درخت

راهبرد جستجو تابعی است که گره بعدی را برای بسط درخت از مجموعه حاشیه انتخاب می کند



شکل ۸-۳ گره‌ها ساختمان داده‌هایی هستند که درخت جست‌وجو از آن‌ها ساخته می‌شود. پیکان‌ها از والد به فرزند رسم شده‌اند.

اندازه گیری کارایی الگوریتم حل مسئله

- **کامل بودن:** آیا الگوریتم تضمین می کند که در صورت وجود راه حل، آن را بیابد؟
- **بهینگی:** آیا این راهبرد، راه حل بهینه ای را ارائه می کند.
- **پیچیدگی زمانی:** چقدر طول می کشد 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

اندازه گیری کارایی حل مساله

پیچیدگی بر حسب سه کمیت بیان می شود

فاکتور انشعاب **b**

عمق نزدیکترین گره هدف **d**

حداکثر طول هر مسیر در فضای حالت **m**

سنجش زمان بر حسب تعدا گره های تولید شده در حین جستجو

سنجش فضا بر حسب حداکثر گره های ذخیره شده در حافظه

Email: ComputerCollege_Fam@Yahoo.Com

جستجوی ناآگاهانه

- ناآگاهی این است که الگوریتم هیچ اطلاعاتی غیر از تعریف مسئله در اختیار ندارد
- این الگوریتم ها فقط می تواند جانشین هایی را تولید و هدف را از غیر هدف تشخیص دهند

جستجوی آگاهانه یا جستجوی اکتشافی

- راهبردهایی که تشخیص می دهد یک حالت غیر هدف نسبت به گره غیر هدف دیگر، امید بخش تر است، جست و جوی آگاهانه یا جست و جوی اکتشافی نامیده می شود.

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

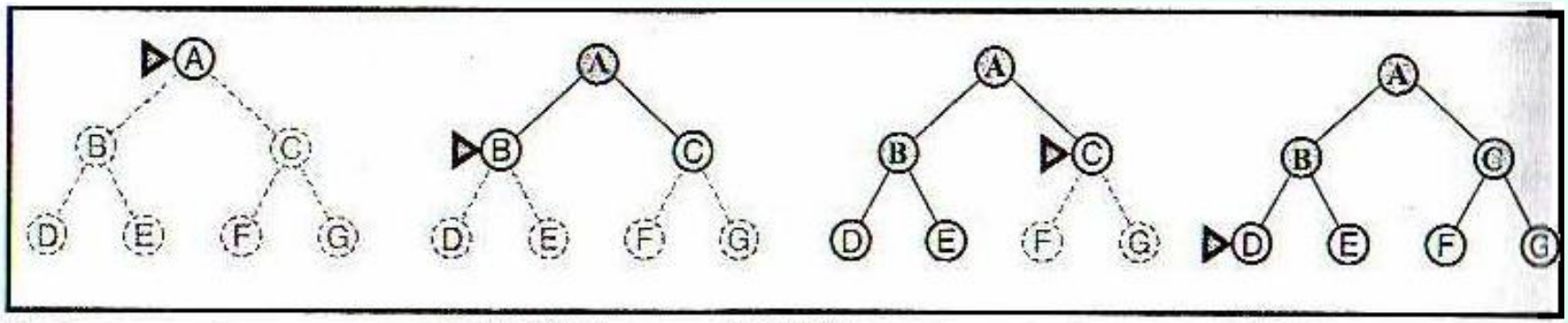
Email: ComputerCollege_Fam@Yahoo.Com

ابتدا ریشه بسط داده می شود

سپس تمام جانشین های ریشه بسط داده می شوند و به همین ترتیب

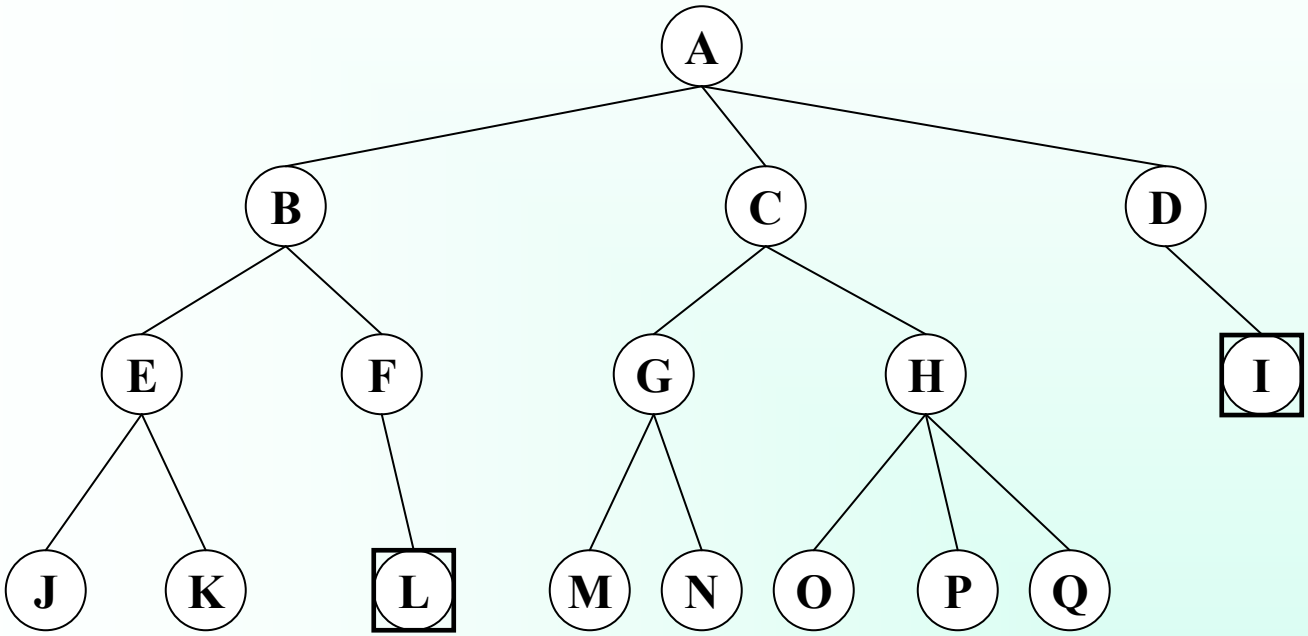
بسط تمام گره های موجود در یک عمق درخت و سپس حرکت در عمق بعدی

پیچیدگی زمانی این الگوریتم $O(b^{d+1})$ می باشد



شکل ۱۰-۳ جست‌وجوی عرضی در یک درخت دودویی. در هر مرحله، گره‌ای که بعداً باید بسط داده شود، علامت‌دار شده است.

جستجوی عرضی



Email: ComputerCollege_Fam@Yahoo.Com

کامل بودن: بله

بهینگی: بله (مشروط)

در صورتی بهینه است که هزینه مسیر، تابعی غیر نزولی از عمق گره باشد. (مثل وقتی که فعالیت ها هزینه یکسانی دارند)

پیچیدگی زمانی: $O(b^{d+1})$

پیچیدگی فضا: $O(b^{d+1})$

Email: ComputerCollege_Fam@Yahoo.Com

همان جستجوی عرضی است که به جای بسط سطحی ترین گره،

گره **n** را با کمترین هزینه مسیر بسط می دهد

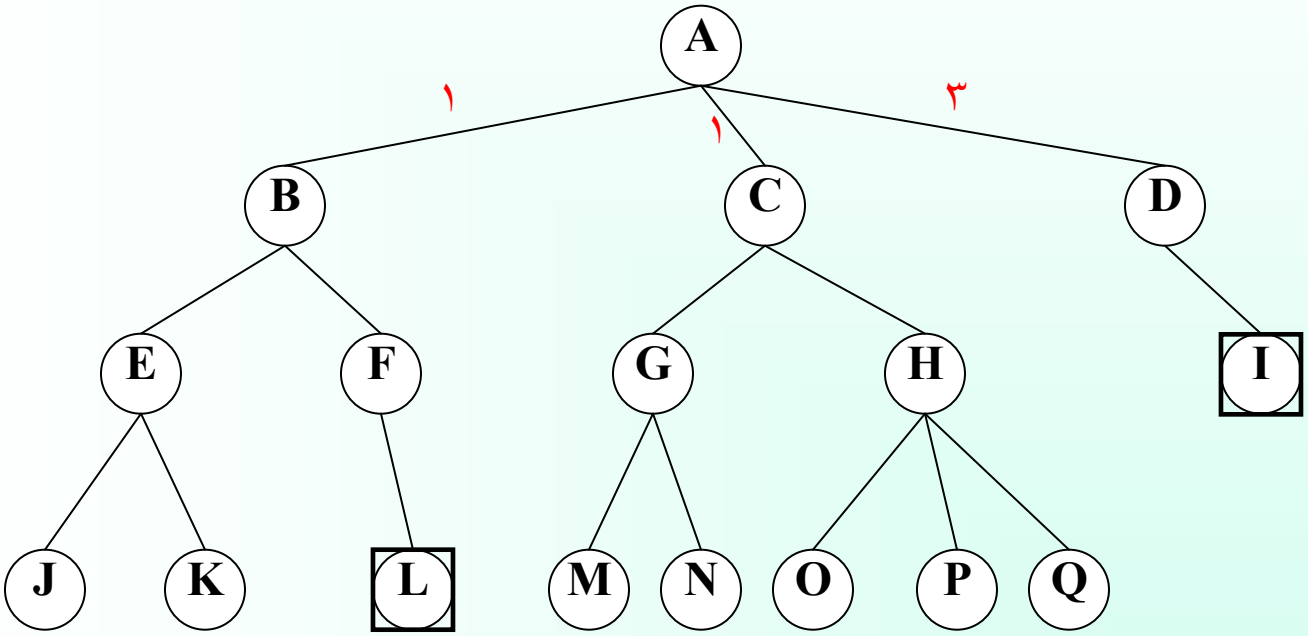
اهمیت در کل هزینه مراحل است نه تعداد مراحل

تضمین کامل بودن: هزینه هر مرحله $\geq \varepsilon$ $\varepsilon > 0$

شرط کافی برای بهینگی نیزمی باشد

پیچیدگی زمان و فضا $O(b^{[C^*/\varepsilon]})$ C^* هزینه راه حل بهینه

این جستجو گره n را با کمترین هزینه مسیر بسط می دهد



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)

بهینگی: بله

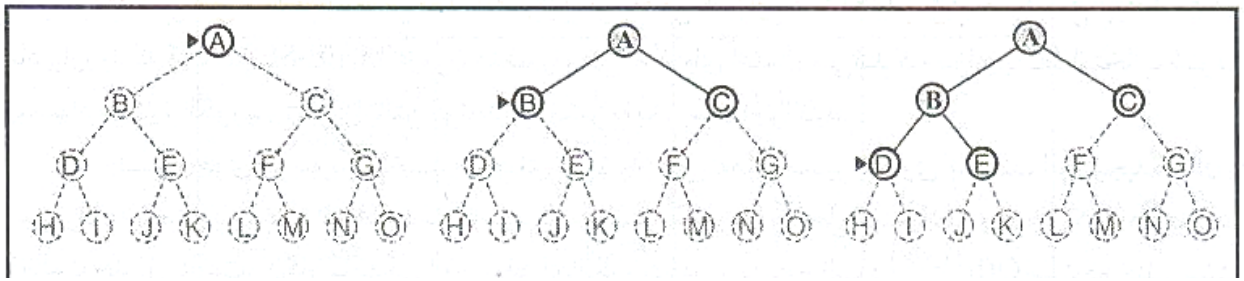
هزینه هر مرحله بزرگتر یا مساوی ε باشد

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

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

Depth First Search

جستجوی عمقی



کامل بودن: خیر

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

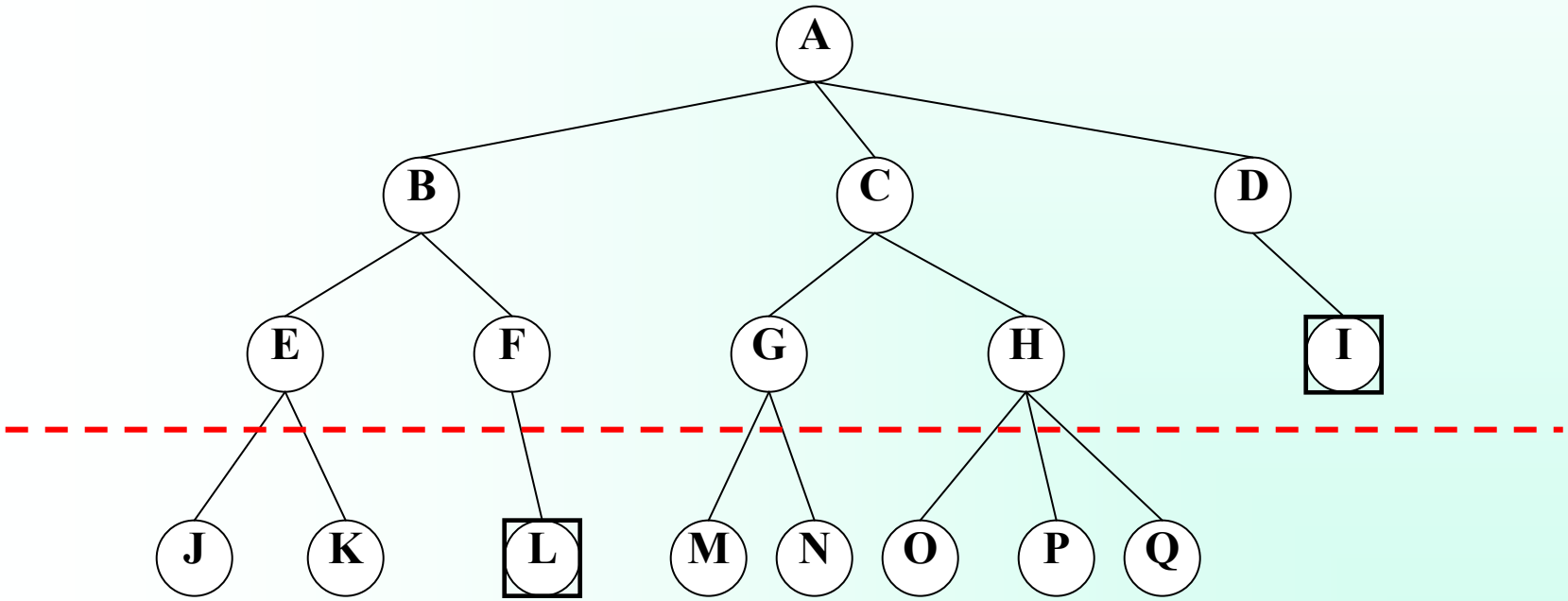
بهینگی: خیر

پیچیدگی زمانی: $O(b^m)$

پیچیدگی فضا: $O(bm)$

جستجوی عمقی محدود

مسئله درخت های نامحدود می تواند به وسیله جست و جوی عمقی با عمق محدود L بهبود یابد



جستجوی عمقی محدود

کامل بودن: خیر

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

بهینگی: خیر

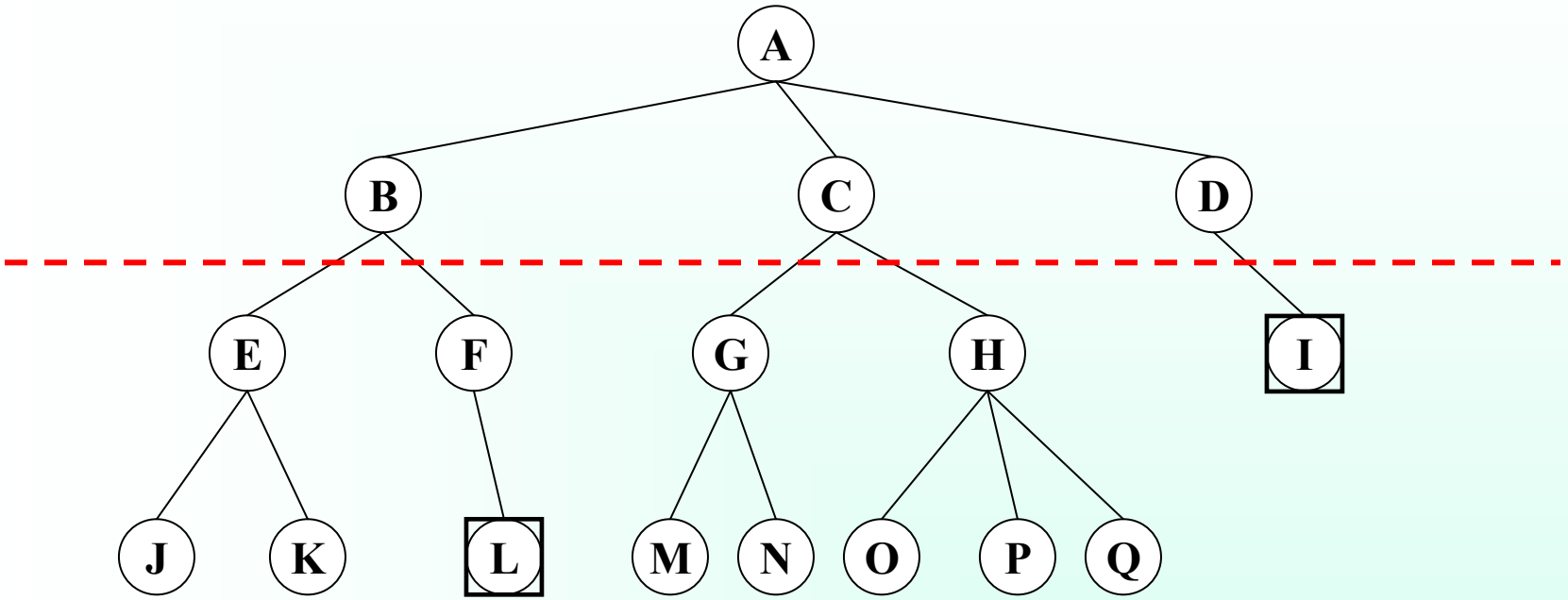
اگر L>d انتخاب شود، این راهبرد بهینه نخواهد بود.

پیچیدگی زمانی: $O(b^L)$

پیچیدگی فضا: $O(bL)$

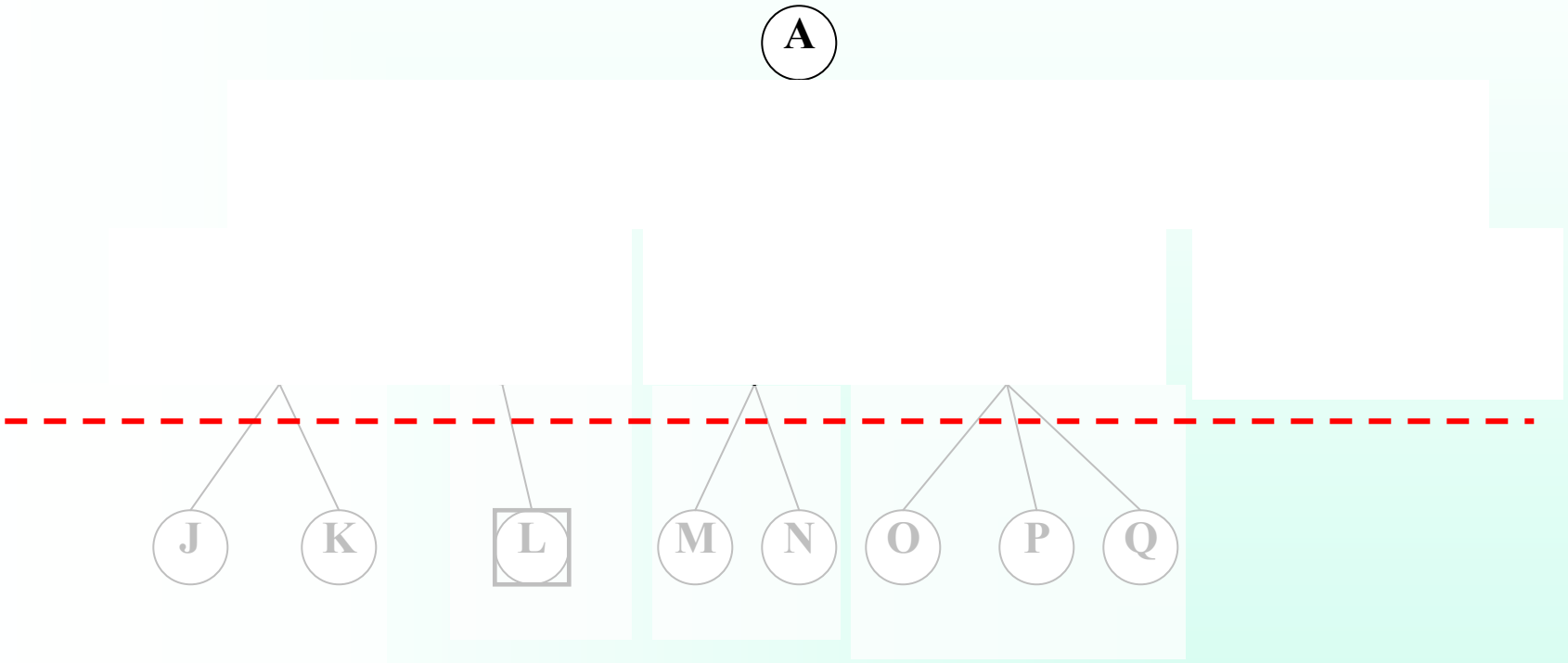
Email: ComputerCollege_Fam@Yahoo.Com

جستجوی عمقی محدود



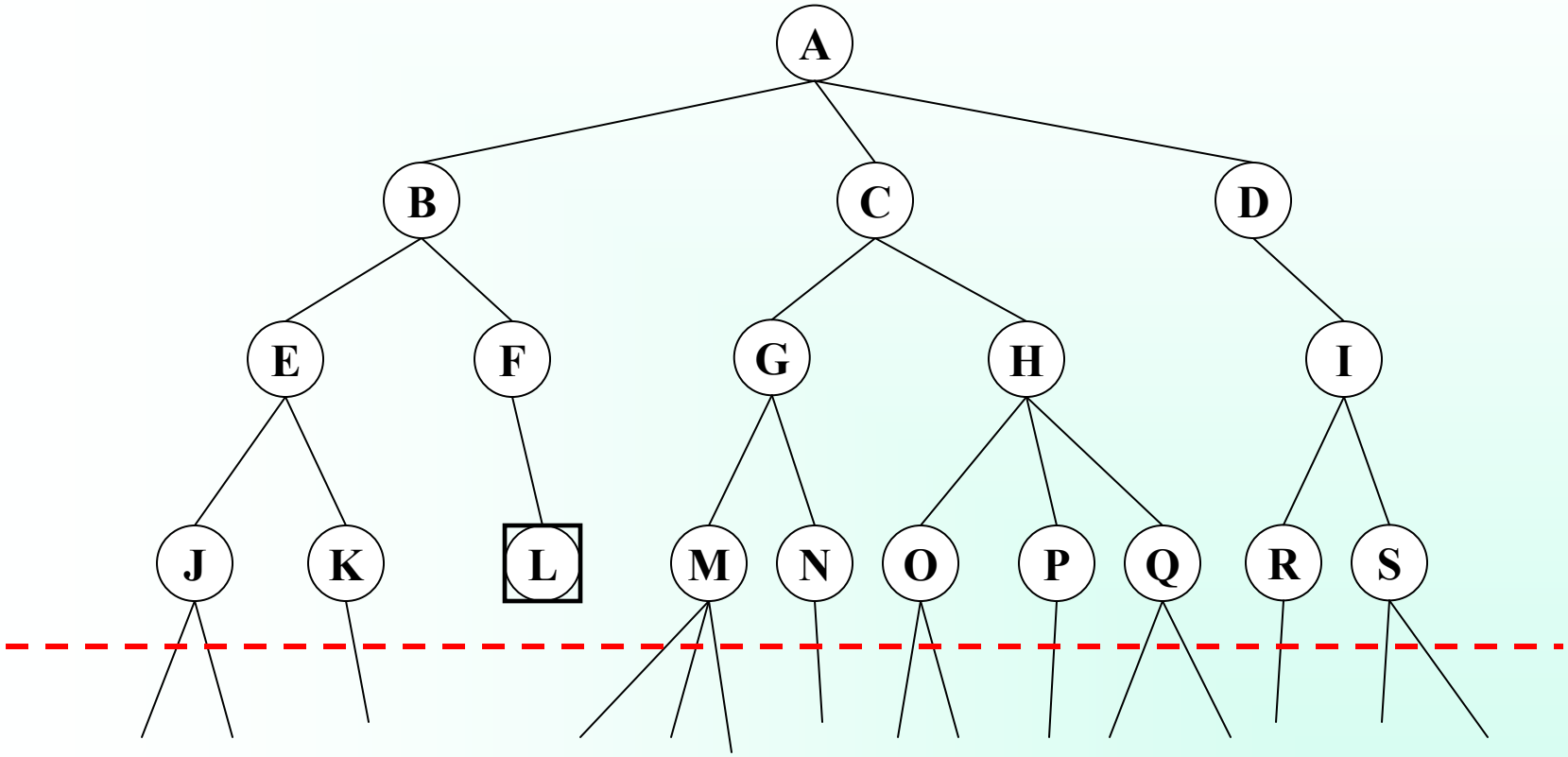
Email: ComputerCollege_Fam@Yahoo.Com

جستجوی عمقی محدود



Email: ComputerCollege_Fam@Yahoo.Com

جستجوی عمیق کننده تکراری



Email: ComputerCollege_Fam@Yahoo.Com

جستجوی عمیق کننده تکراری

کامل بودن: بله

در صورتی که فاکتور انشعاب محدود باشد

بهینگی: بله

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

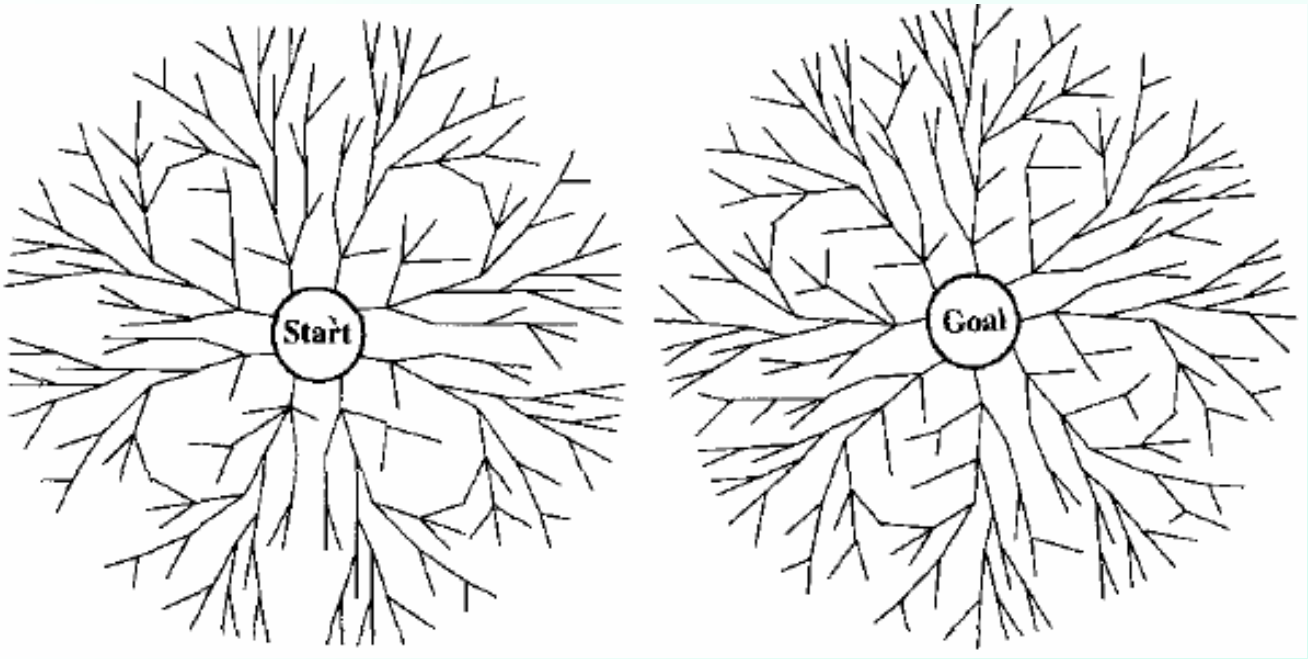
پیچیدگی زمانی: $O(b^d)$

پیچیدگی فضا: $O(bd)$

Email: ComputerCollege_Fam@Yahoo.Com

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



Email: ComputerCollege_Fam@Yahoo.Com

جستجوی دو طرفه

کامل بودن: بله

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

بهینگی: بله

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

پیچیدگی زمانی: $O(b^{d/2})$

پیچیدگی فضا: $O(b^{d/2})$

Email: ComputerCollege_Fam@Yahoo.Com

مقایسه راهبردهای جست و جوی نا آگاهانه

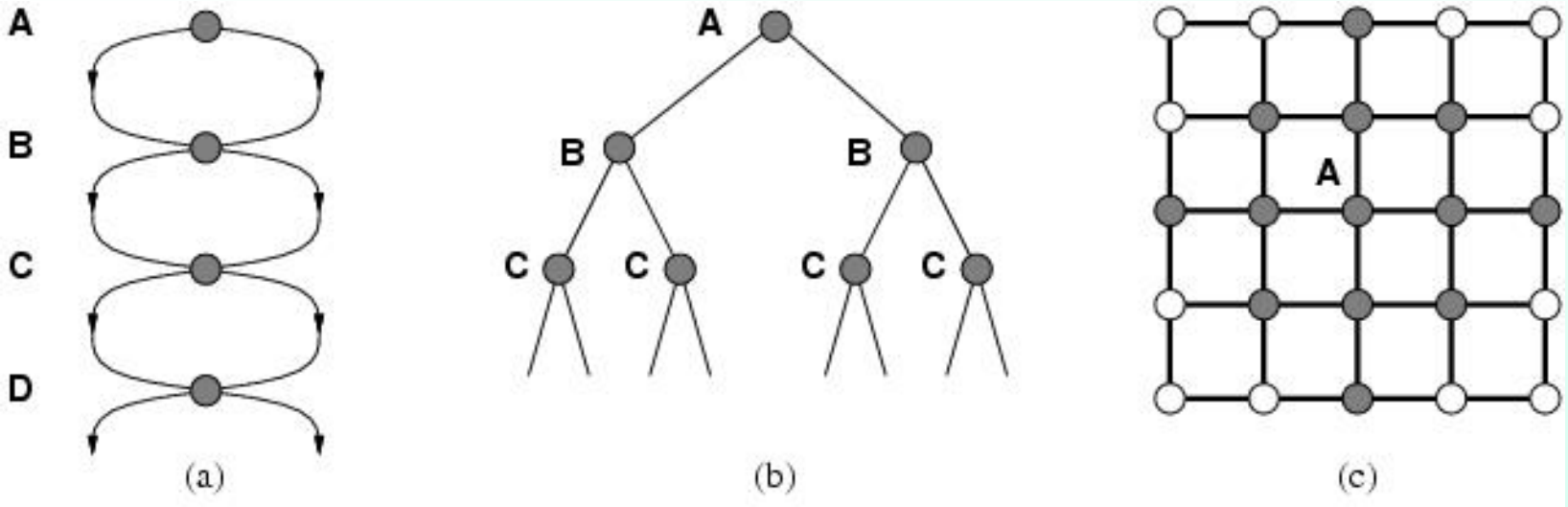
خلاصه ی الگوریتم ها

مقیاس [1]	اوّل سطح	با هزینه ی یکسان	اوّل عمق	عمق محدود شده	عمیق شونده ی تکراری
کامل بودن ؟؟	بله *	بله *	نه	بله ، در صورتی که $l>=d$ باشد	بله
زمان ؟؟	b^d+1	$b^{\lceil C^*/\epsilon \rceil}$	b^m	b^l	b^d
فضا ؟؟	b^d+1	$b^{\lceil C^*/\epsilon \rceil}$	bm	bl	bd
بهینگی ؟؟	بله *	بله	نه	نه	بله *

Email: ComputerCollege_Fam@Yahoo.Com

اجتناب از حالت های تکراری

وجود حالت های تکراری در یک مسئله قابل حل، می تواند آن را به مسئله غیر قابل حل تبدیل کند



Email: ComputerCollege_Fam@Yahoo.Com

جستجو با اطلاعات ناقص

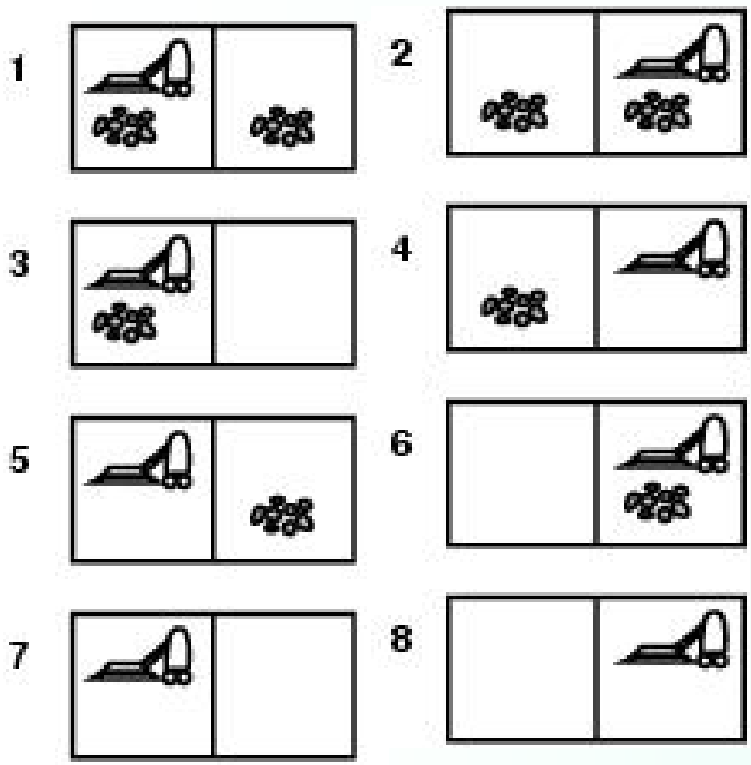
- **مسئله های فاقد حسگر:** اگر عامل فاقد حسگر باشد، می تواند در یکی از چند حالت اولیه باشد و هر فعالیت می تواند آن را به یکی از چند حالت جانشین ببرد

- **مسئله های اقتضایی:** اگر محیط به طور جزئی قابل مشاهده باشد یا اگر فعالیت ها قطعی نباشد، ادراکات عامل، پس از هر عمل، اطلاعات جدیدی را تهیه می کنند. هر ادراک ممکن، اقتضایی را تعریف می کند که باید برای آن برنامه ریزی شود

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

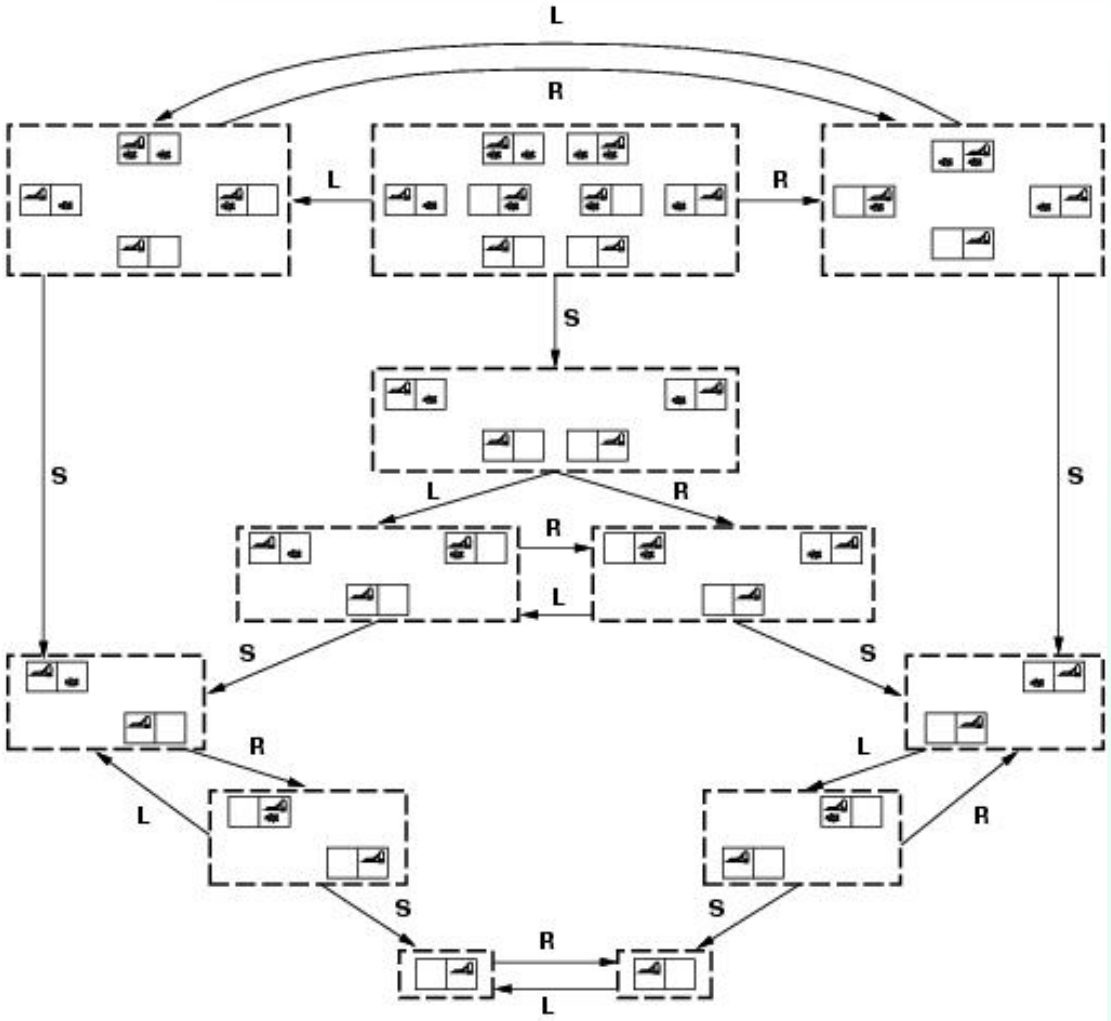
- **مسئله های اکتشافی:** وقتی حالت ها و فعالیت های محیط ناشناخته باشند، عامل باید سعی کند آنها را کشف کند. مسئله های اکتشافی را می توان شکل نهایی مسئله های اقتضایی دانست

مثال: دنیای جاروبرقی فاقد حسگر



- عامل جارو تمام اثرات فعالیت هایش را می داند اما فاقد حسگر است.
- حالت اولیه آن یکی از اعضای مجموعه {۱،۲،۳،۴،۵،۶،۷،۸} می باشد
- فعالیت (Right) {۲،۴،۶،۸}
- فعالیت (Right,Suck) ⟵ {۴،۸}
- فعالیت (Right,Suck,Left,Suck) ⟵ تضمین می کند که صرف نظر از حالت اولیه، به حالت هدف، یعنی ۷ برسد

دنیای جاروبرقی فاقد حسگر

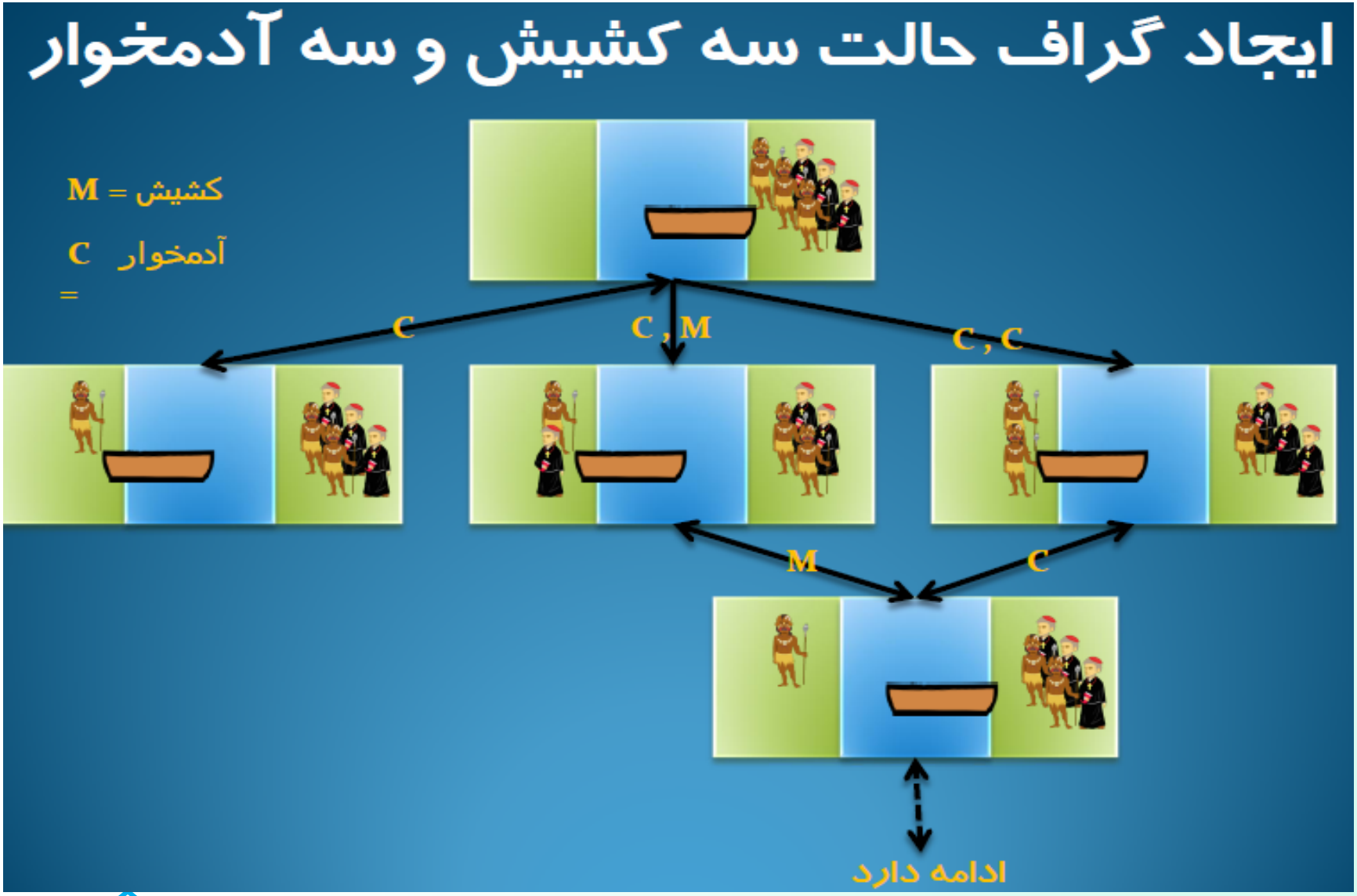


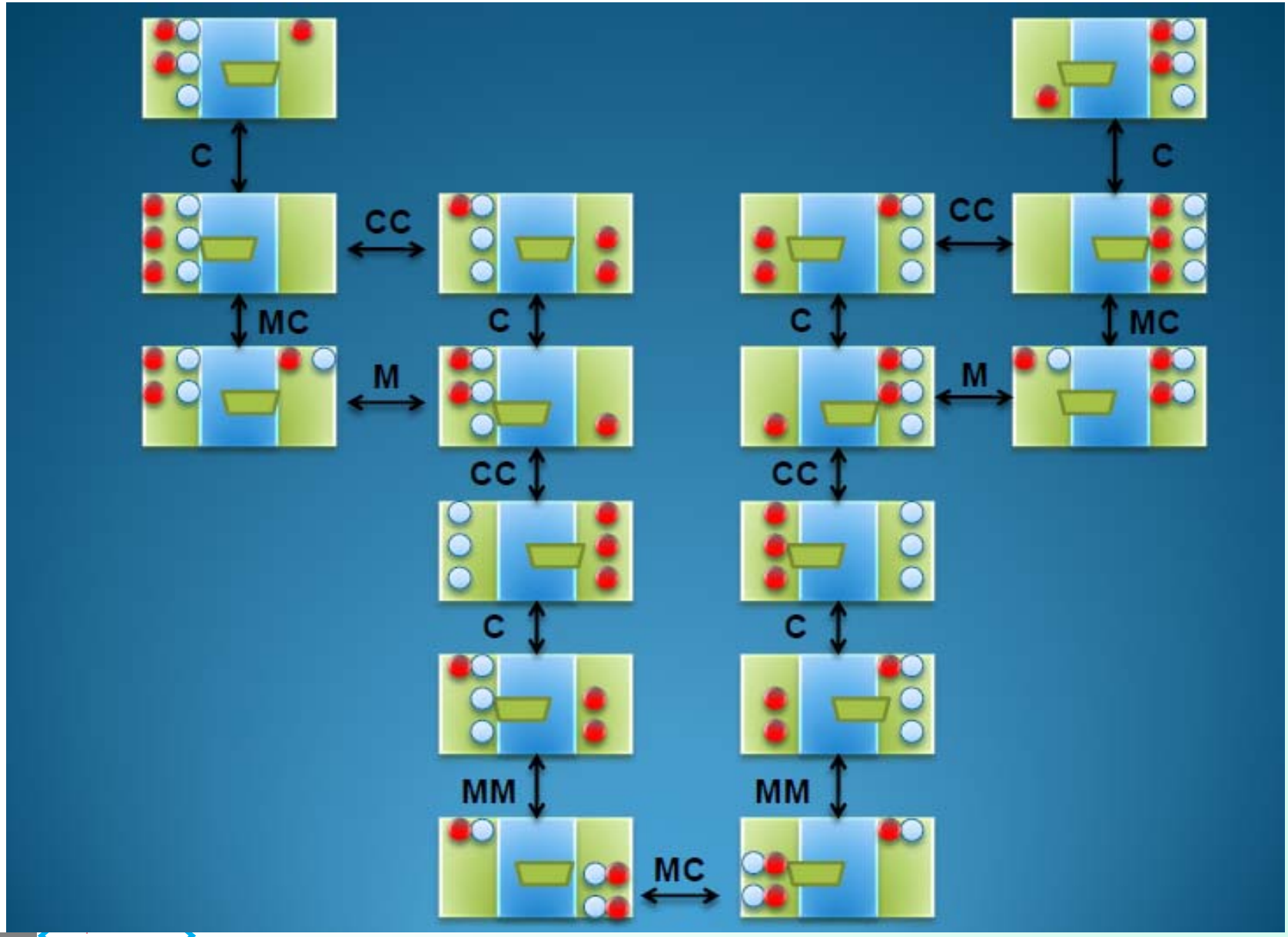
عامل باید راجع به مجموعه های حالتی که می تواند به آنها برسد استدلال کند. این مجموعه از حالت ها را حالت باور گوییم.

اگر فضای حالت فیزیکی دارای s حالت باشد فضای حالت باور 2^{s} حالت باور خواهد داشت.

Email: ComputerCollege_Fam@Yahoo.Com

ایجاد گراف حالت سه کشیش و سه آدمخوار




C
CC
MC
C
M
CC
C
MM
MC
C
CC
MC
C
M
CC
C
MM

گرگ و کلم و گوسفند

کشاورزی پس از خرید یک گرگ, یک کلم و یک گوسفند از بازار بازمیگردد. در راه بازگشت باید از رودخانه ای عبور کند.اما قایق او تنها برای یکی از اینها جا دارد.او نمی تواند گوسفند و کلم را یک جا نگاه دارد زیرا ممکن است گوسفند کلم را ببلعد.از سوی دیگر ماندن گرگ و گوسفند هم در کنار هم موجب از بین رفتن گوسفند می شود.پس چه طور می تواند آنها را بدون اینکه آسیب ببینند جابه جا کند؟

پدر و دو فرزند

پدری با دو فرزند خود به رودخانه ای رسیدند . تنها راه گذشتن از رود خانه این بود که از یک ماهیگیر بخواهند قایق خود را به آنها امانت بدهد.اما قایق تنها می تواند یک آدم بالغ یا دو کودک را حمل کند.پس این خانواده چه طور می توانند به سمت دیگر بروند و در نهایت قایق را به ماهیگیر بازگردانند؟

تمرین :

به دلخواه دو مورد از سه مورد زیر را انجام دهید؟

گراف فضای حالت مسله گرگ و کلم و گوسفند را رسم کنید ؟

گراف فضای حالت مسله پدر و دو فرزند را رسم کنید ؟

گراف فضای حالت مسله میمو ن ها و آدم ها را رسم کنید ؟

هوش مصنوعی

فصل چهارم

جست و جوی آگاهانه و اکتشاف

Email: ComputerCollege_Fam@Yahoo.Com

فهرست

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

انواع جستجو

جستجو

جستجوی نا آگاهانه یا جستجوی کور

۱) هیچ اطلاعات اظافی در مورد حالتها ندارد

۲) تنها می تواند حالات را تولید و تشخیص دهد کدام حالت هدف است یا هدف نیست .

جستجوی آگاهانه یا جستجوی هیوریستیک

مشکلات روش های جستجوی نا آگاهانه

- معیار انتخاب گره بعدی برای گسترش دادن، تنها به شماره سطح گره بستگی دارد
- از ساختار مسئله بهره نمی برند
- در خت جستجو را با یک ساختار از پیش تعریف شده گسترش می دهند.
  - یعنی قابلیت تطبیق پذیری با آنچه که تا کنون در مسیر جستجو دریافته اند و نیز حرکتی که می تواند خوب باشد را ندارند.

Email: ComputerCollege_Fam@Yahoo.Com

جستجوی آگاهانه(Informed) یا اکتشافی(Heuristic)

- در جستجوی اگاهانه اطلاعاتی در رابطه با هزینه رسیدن به هدف، در اختیار عامل قرار داده می شود.

- هیوریستیک ها
  - استراتژیهایی برای جستجو که اغلب اوقات ما را به جواب نزدیکتر می کنند.
  - از ساختار مساله نشات می گیرند و هدفشان راهنمایی و هدایت جستجو می باشد.
  - در این روشها هزینه پرداخت شده تا کنون و هزینه تخمینی رسیدن به هدف در نظر می شود.

Email: ComputerCollege_Fam@Yahoo.Com

متدهای جستجوی آگاهانه

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

Email: ComputerCollege_Fam@Yahoo.Com

تعاریف

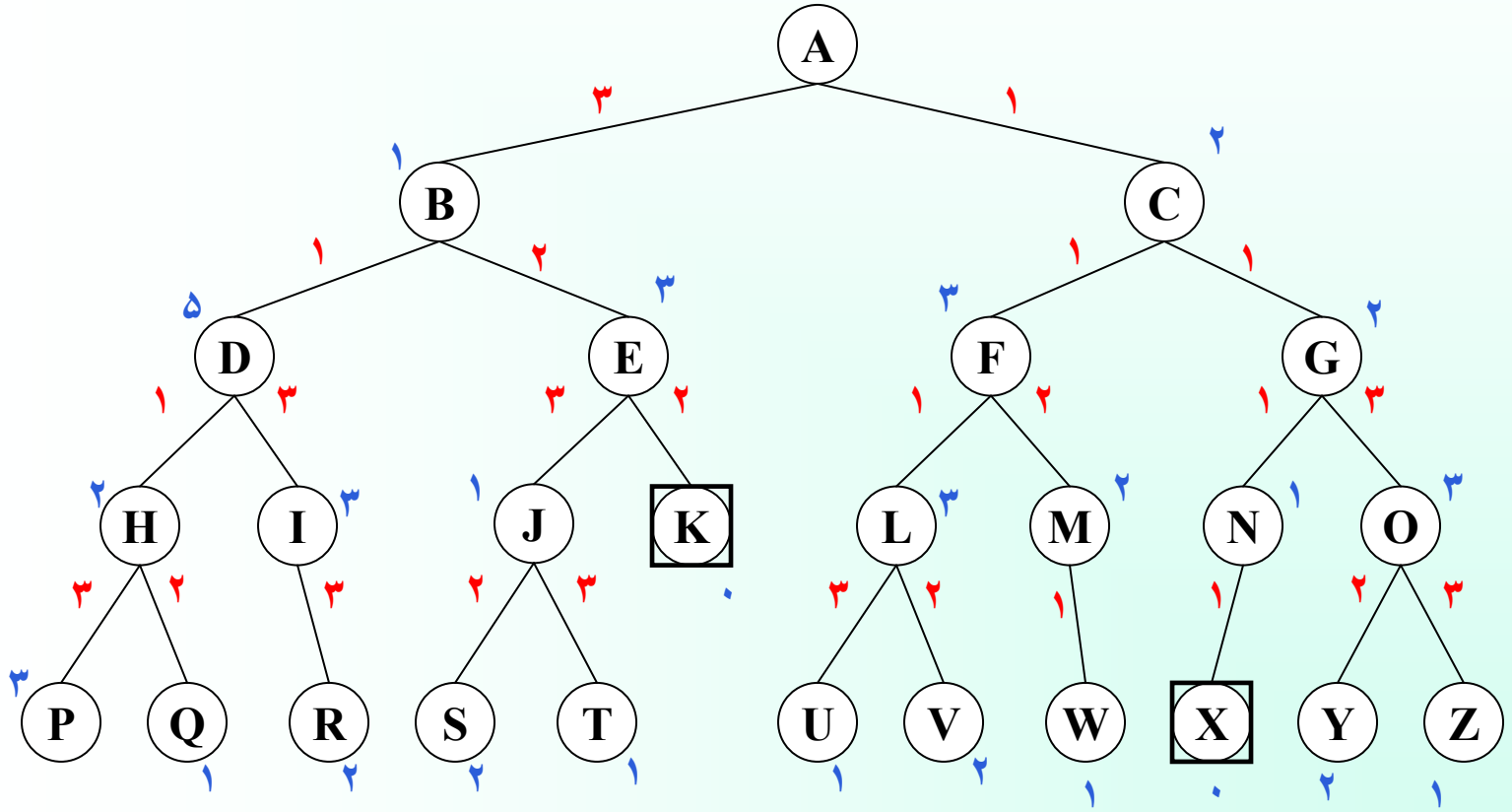
- تابع هزینه مسیر، $g(n)$: هزینه مسیر از گره اولیه تا گره n
- تابع اکتشافی، $h(n)$: هزینه تخمینی ارزان ترین مسیر از گره n به گره هدف
- تابع بهترین مسیر، $h^*(n)$: ارزان ترین مسیر از گره n تا گره هدف
- تابع ارزیابی، $f(n)$: هزینه تخمینی ارزان ترین مسیر از طریق n

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

- $f^*(n)$: هزینه ارزان ترین مسیر از طریقn $f^*(n): g(n) + h^*(n)$

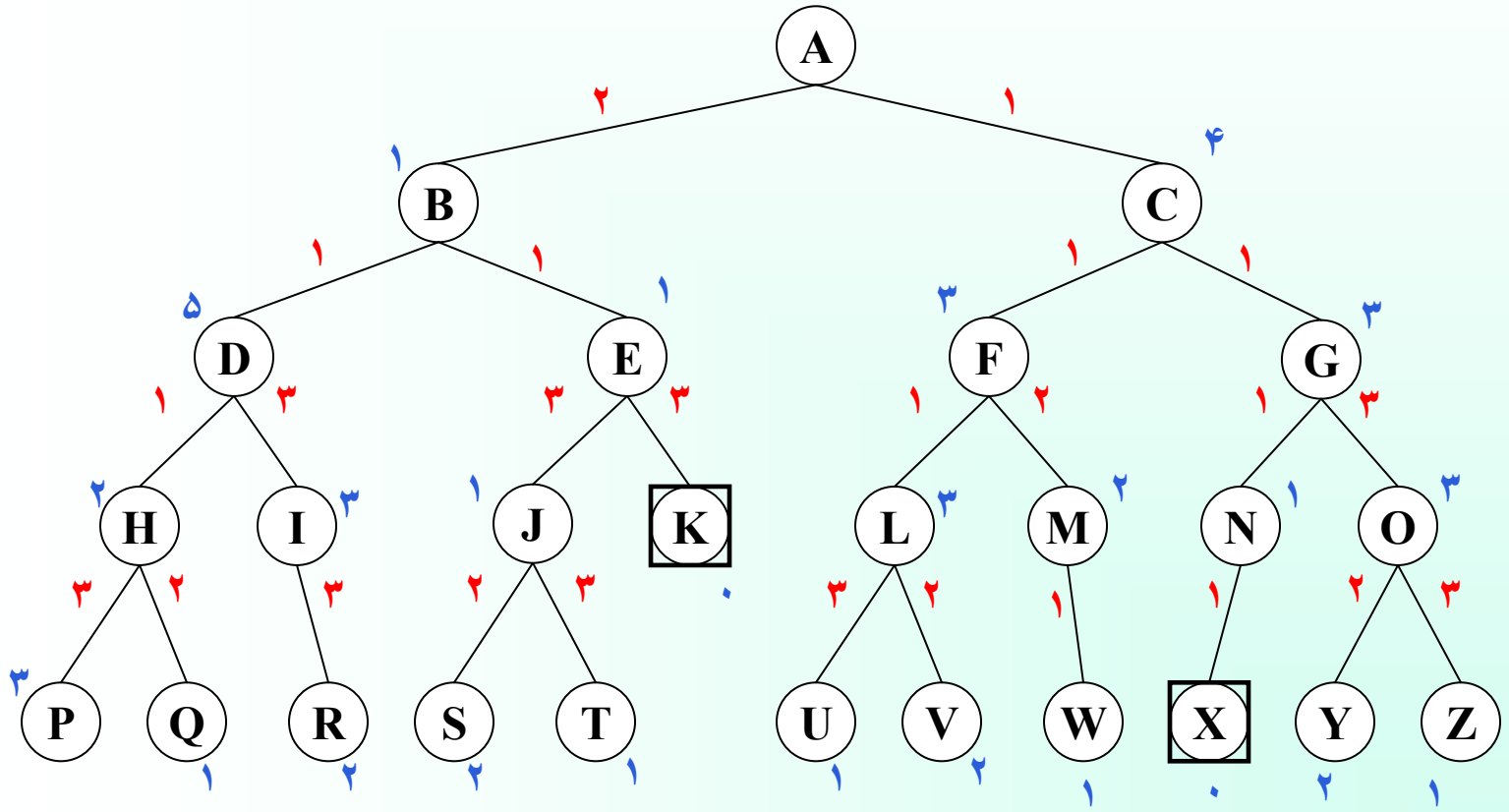
Email: ComputerCollege_Fam@Yahoo.Com

جستجوی حریصانه

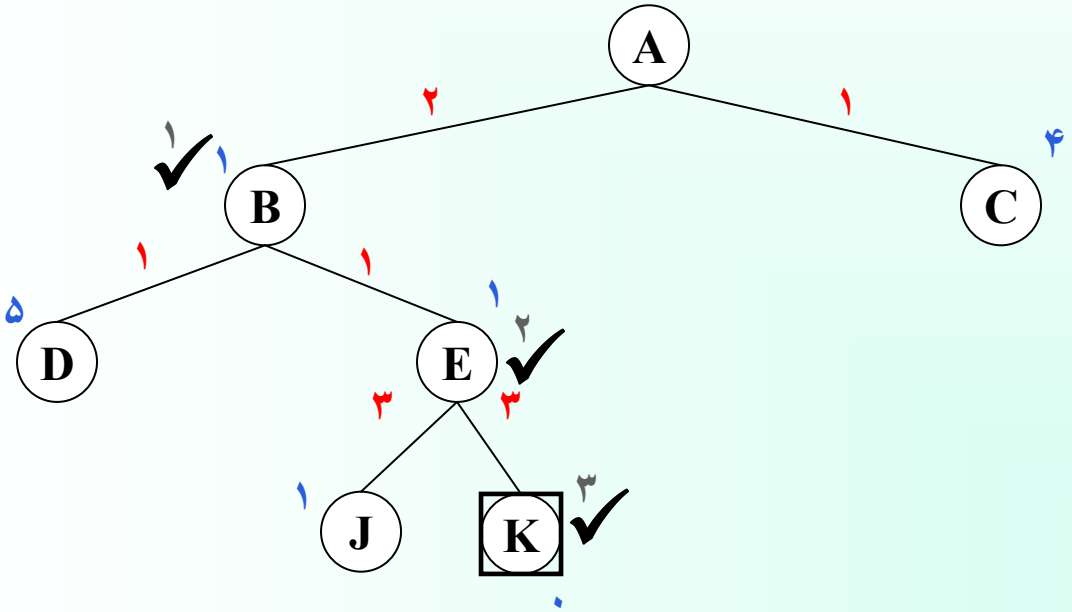


جستجوی حریصانه

جستجوی حریصانه



جستجوی حریصانه

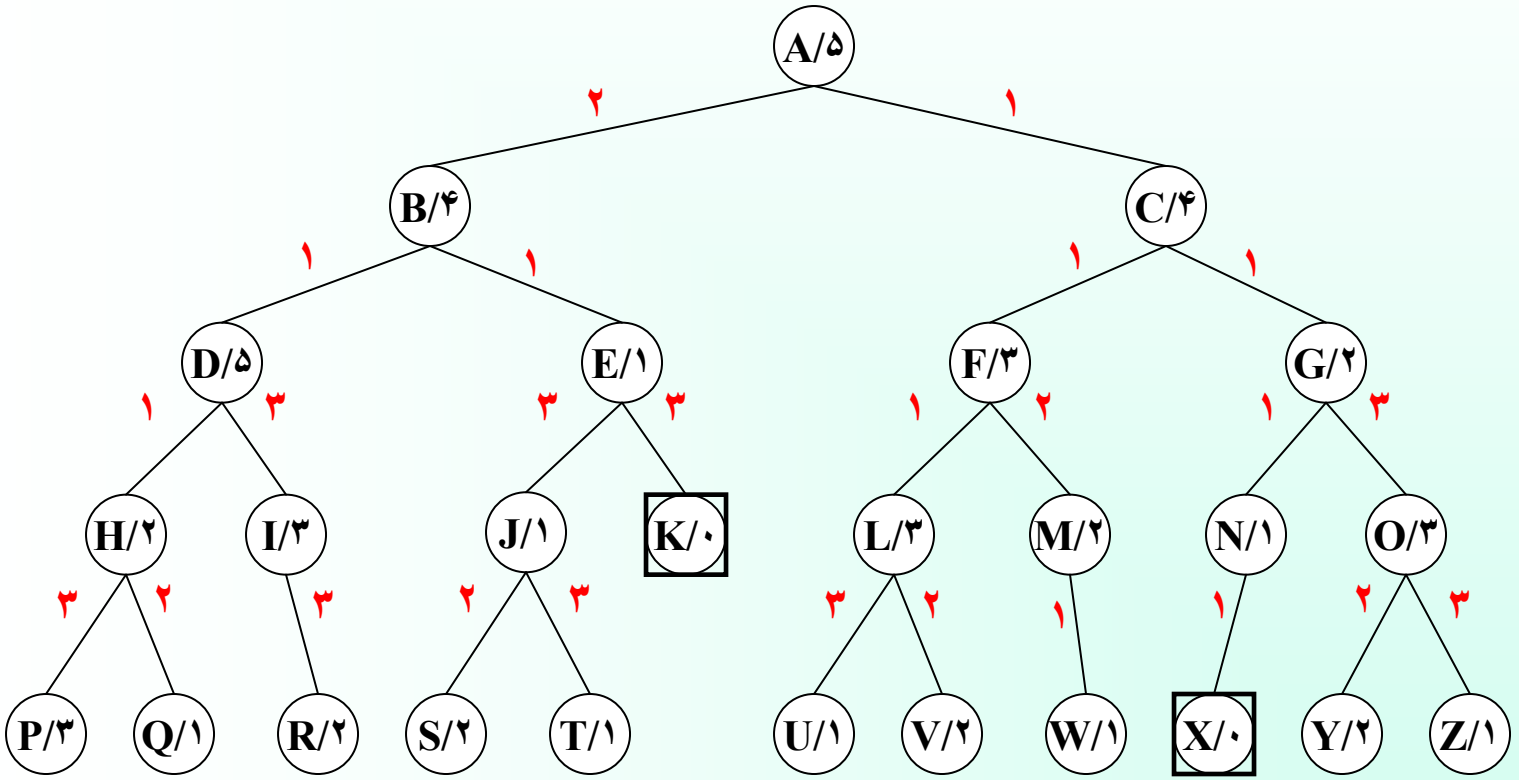


جستجوی حریصانه

- کامل بودن:
  - خیر اما اگر $h = h^*$ آنگاه جستجو کامل می شود
- بهینگی:
  - خیر اما اگر $h = h^*$ آنگاه جستجو کامل می شود
- پیچیدگی زمانی: $O(b^m)$
  - اما اگر $h = h^*$ آنگاه $O(bd)$
- پیچیدگی فضا: $O(b^m)$
  - اما اگر $h = h^*$ آنگاه $O(bd)$

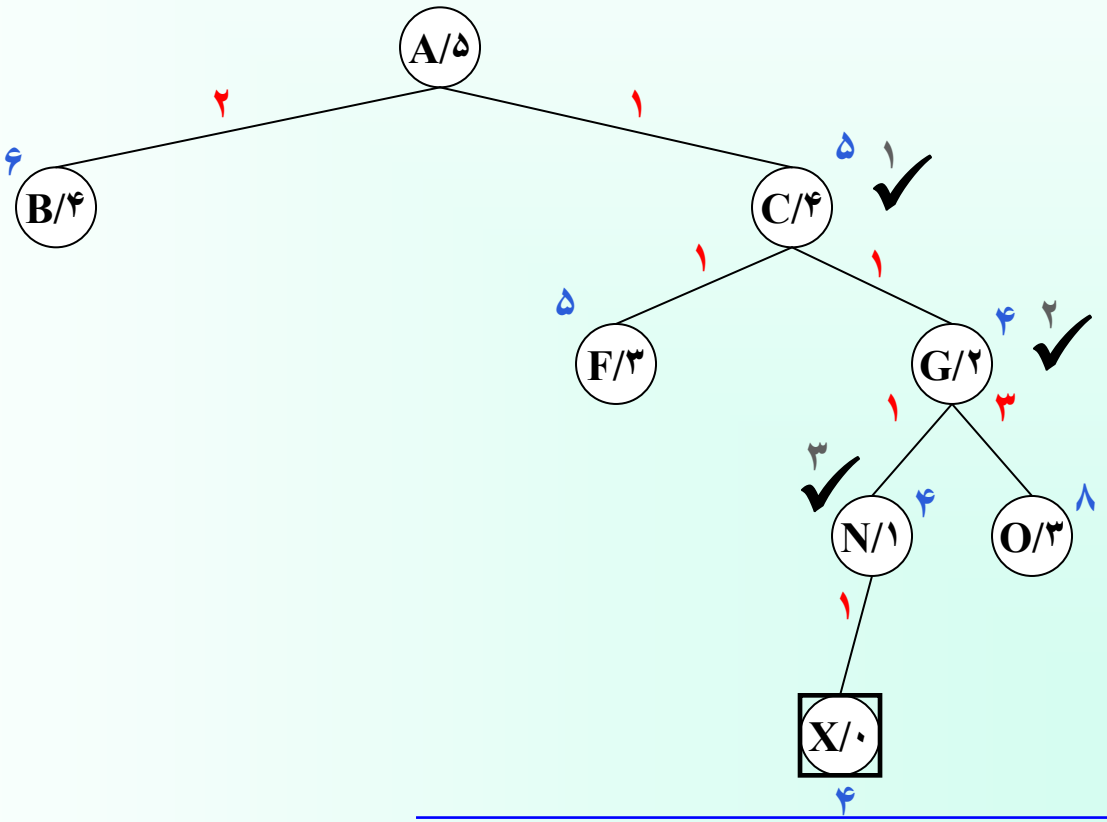
Email: ComputerCollege_Fam@Yahoo.Com

جستجوی A*



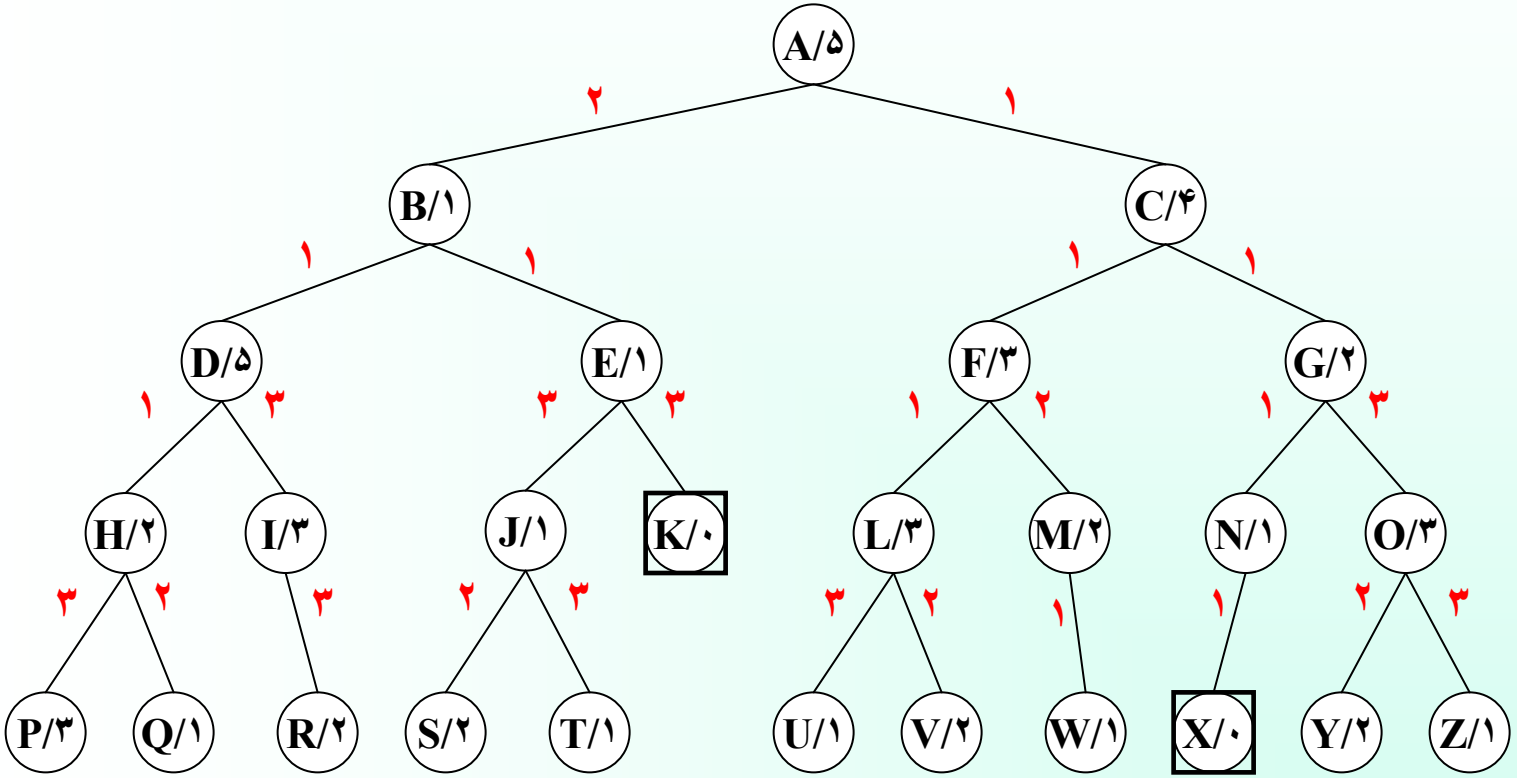
Email: ComputerCollege_Fam@Yahoo.Com

جستجوی A*



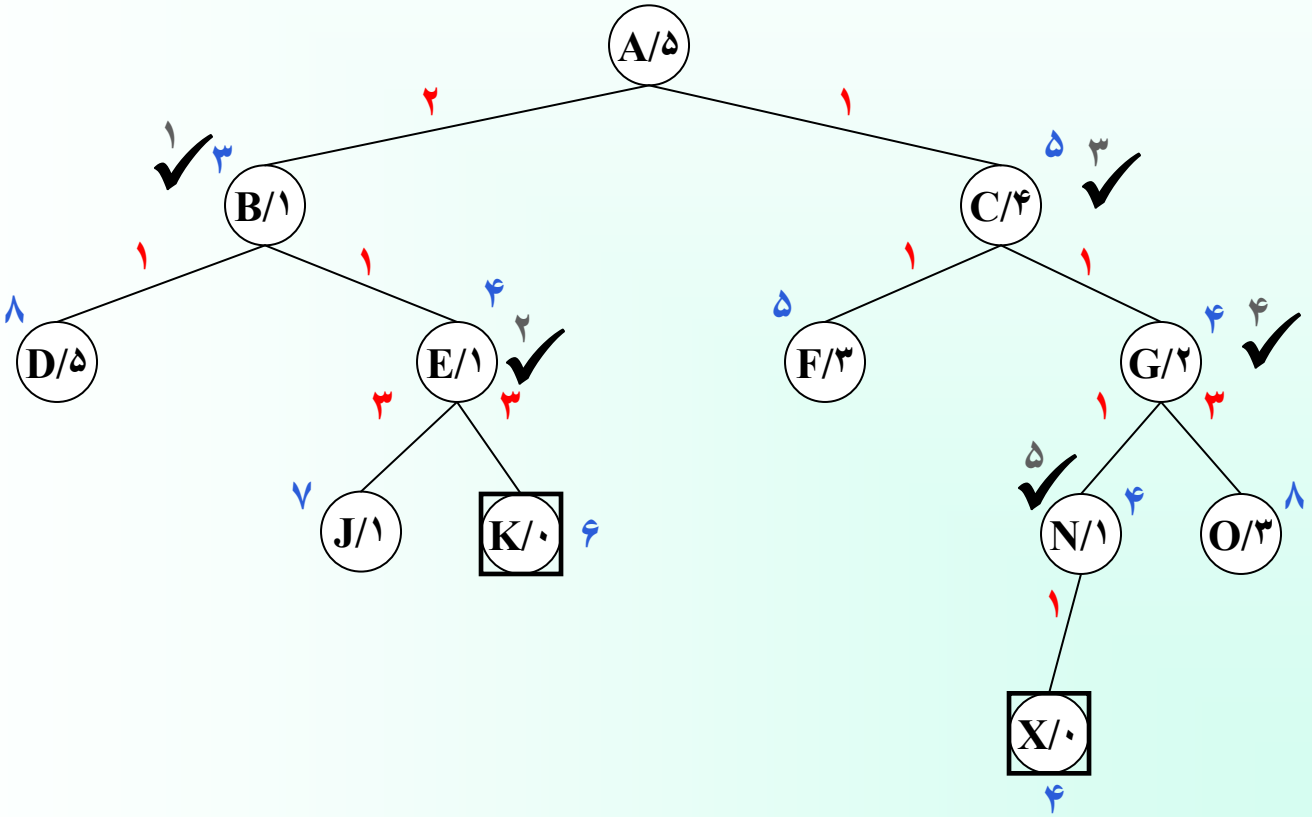
Email: ComputerCollege_Fam@Yahoo.Com

جستجوی A*



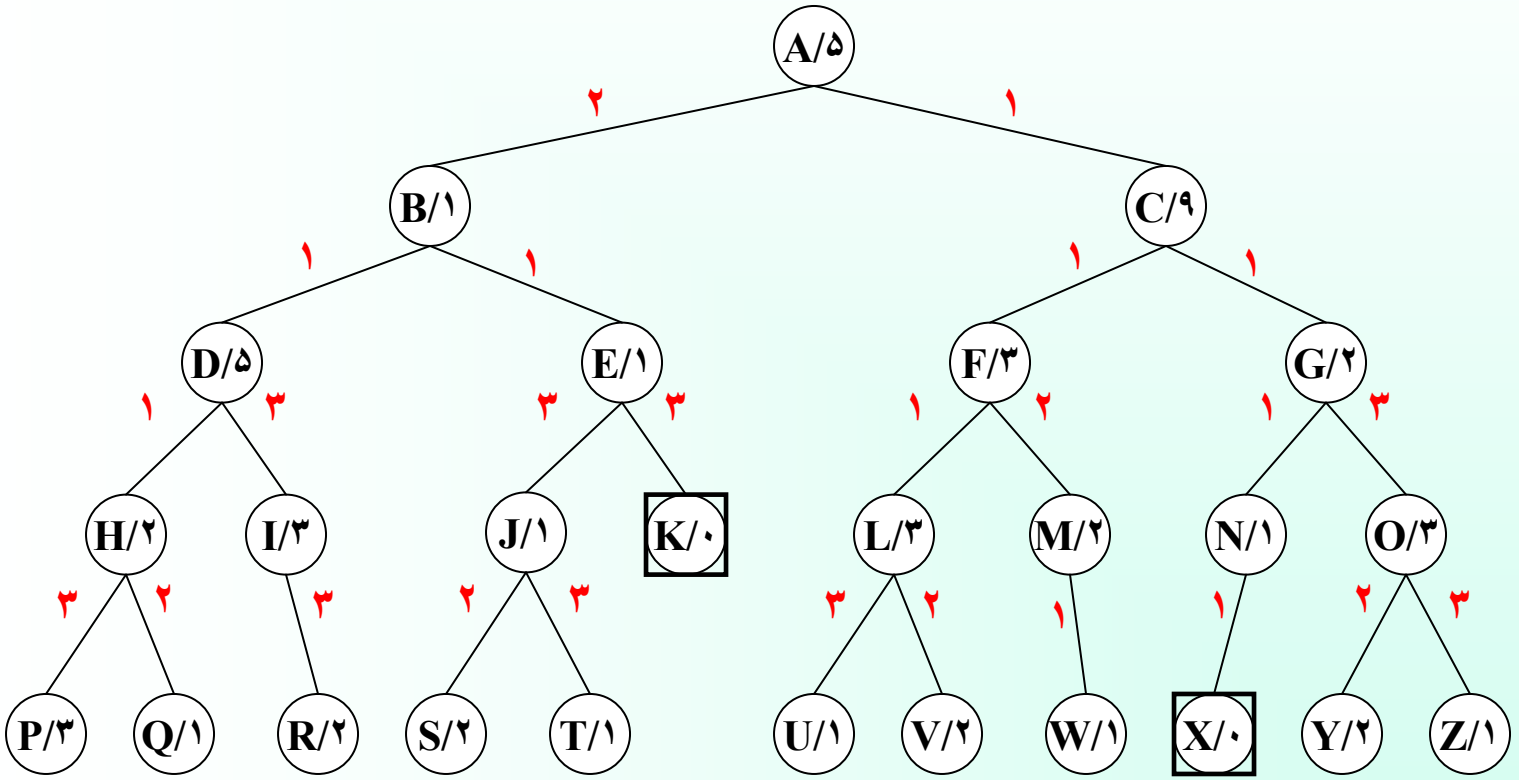
Email: ComputerCollege_Fam@Yahoo.Com

جستجوی A*



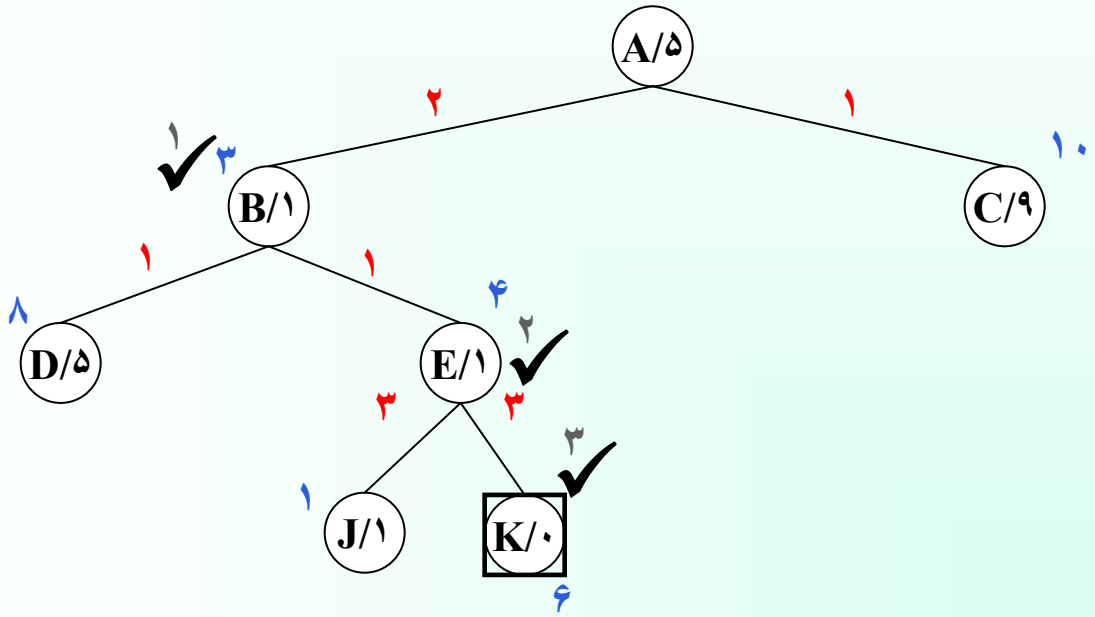
Email: ComputerCollege_Fam@Yahoo.Com

جستجوی A*



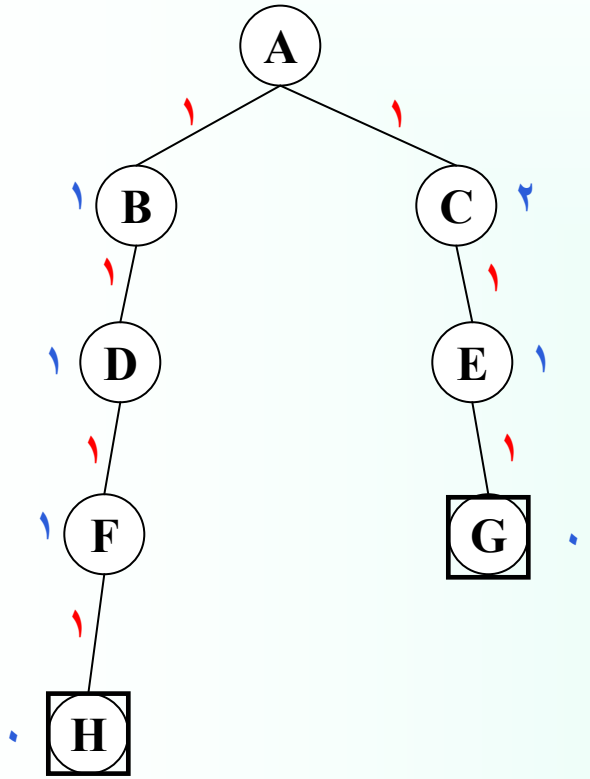
Email: ComputerCollege_Fam@Yahoo.Com

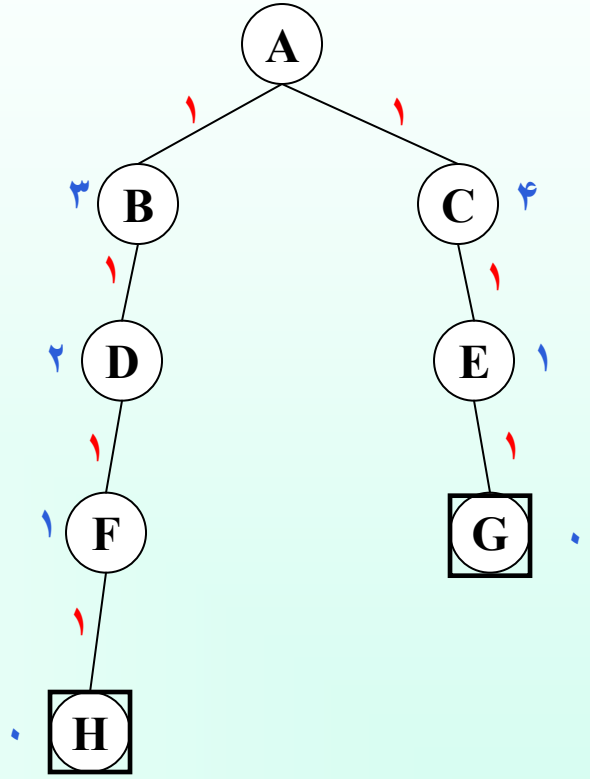
جستجوی A*



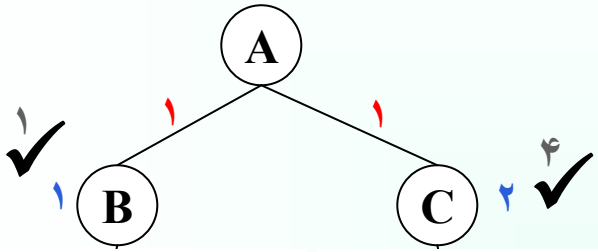
Email: ComputerCollege_Fam@Yahoo.Com

جستجوی A*

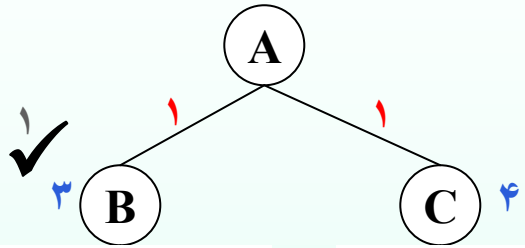




جستجوی A*



$h \leq h^*$



$h \not\leq h^*$

ویژگی های جستجوی A*

- A* بهینه می باشد به شرطیکه تابع هیوریستیک h(n) مورد استفاده در آن یک هیوریستیک قابل قبول (Admissible) باشد
- در یک هیوریستیک قابل قبول همواره شرط $h(n) \leq h^*(n)$ برقرار است
  - $h^*(n)$ هزینه واقعی مسیر از n تا هدف می باشد
  - $h(n) \geq 0$
  - $h(G)=0$ (که G یک هدف است)
  - $0 \leq h(n) \leq h^*(n)$

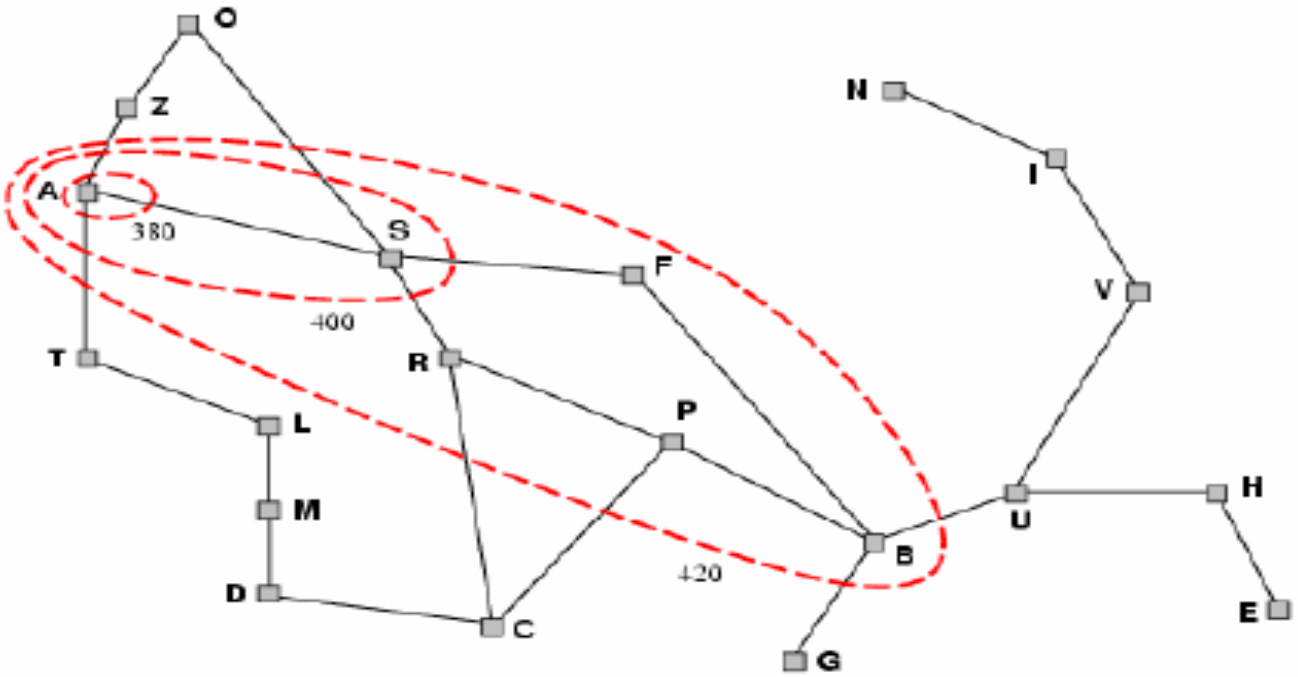
ویژگی های A*

- A* معمولاً قبل از اینکه دچار کمبود زمان شود، دچار کمبود فضا می‌شود. زیرا این جستجو تمام گره‌های تولید شده را در حافظه ذخیره می‌کند.
- A* کامل است (در گرافهایی با فاکتور انشعاب محدود)
- پیچدگی حافظه $O(b^d)$ (چراکه گره های تولید شده در حافظه می مانند)
- پیچیدگی زمانی : نمایی و بر حسب |خطای نسبی h * طول راه حل|
  - فقط در شرایطی پیچیدگی زمانی، نمایی نخواهد بود که رشد خطا در تابع هیوریستیک، رشدی سریعتر از لگاریتم هزینه واقعی مسیر نداشته باشد
- $|h(n)-h^*(n)| \leq O(\log h^*(n))$

Email: ComputerCollege_Fam@Yahoo.Com

خاصیت یکنوایی Monotonicity

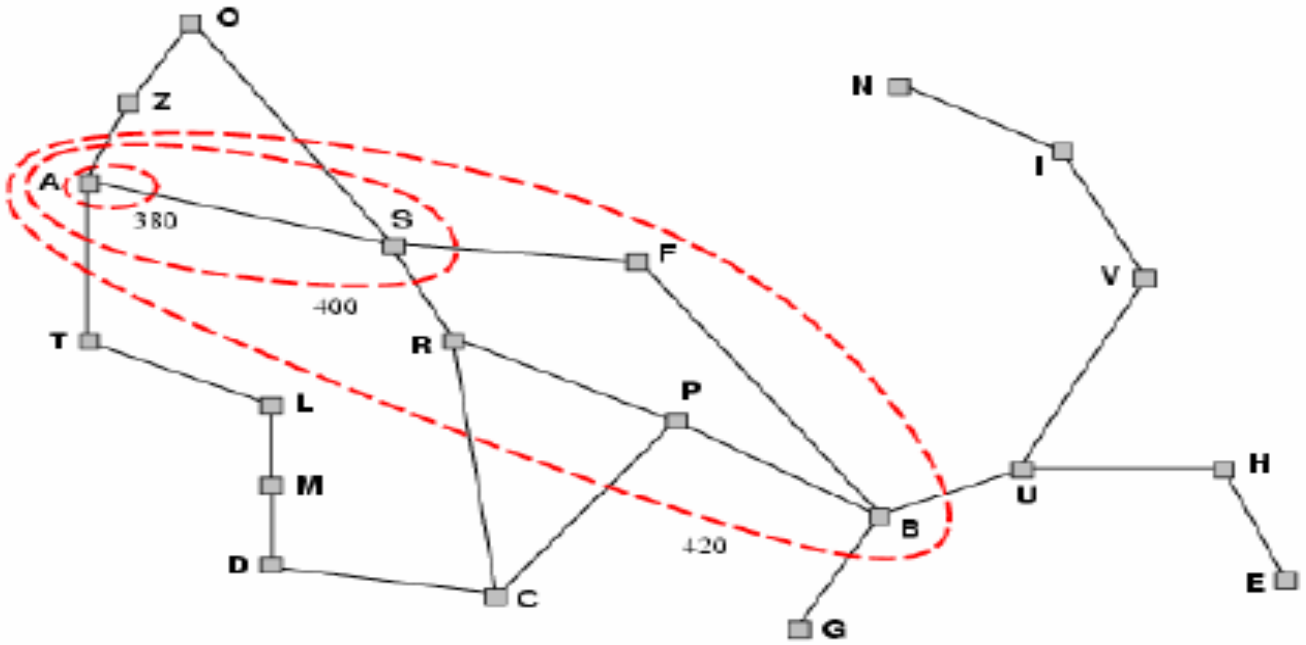
- تابع هیوریستیکی یکنوا است که در طول هر مسیری از ریشه، هزینه f حاصل از آن هرگز کاهش نیابد.
- اگر یکنوا نباشد، با ایجاد یک اصلاح جزئی آن را یکنوا می‌کنیم.
- با داشتن شرط فوق می توان کانتورهای (Contour) در فضای حالت کشید.
- بعنوان مثال کانتور با مقدار 400 شامل تمام گره هایی است که مقدار f آنها کمتر از 400 باشد.



Email: ComputerCollege_Fam@Yahoo.Com

خاصیت یکنوایی Monotonicity

- در جستجو با هزینه یکسان (در A* با h=0) این نواحی (کانتورها) به صورت دایره وار در اطراف فضای حالت خواهند بود.
- هر چه تابع هیوریستیک صحیح تر باشد نواحی به سمت هدف کشیده خواهد شد و در اطراف مسیر بهینه باریک تر خواهند شد.



Email: ComputerCollege_Fam@Yahoo.Com

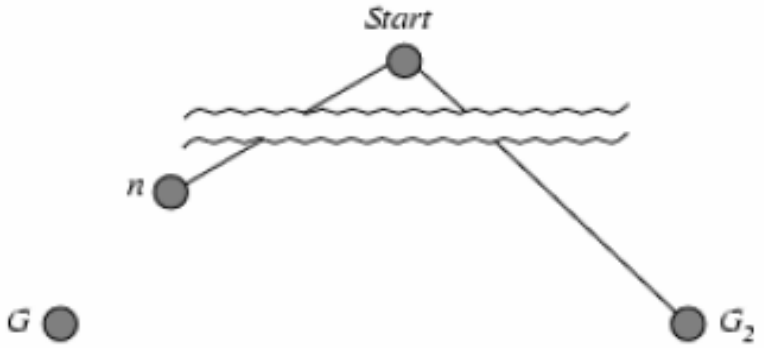
ویژگی های A*

- اگر f^* هزینه واقعی مسیر بهینه از شروع تا هدف تعریف شود، می توان گفت :
  - A* تمام گره ها با $f(n)<f^*$ را بسط خواهد داد.
  - A* ممکن است تعدادی از گره هایی که دقیقا روی « کانتور هدف» (یعنی $f(n)=f^*$) قرار دارند را قبل از اینکه هدف انتخاب شود بسط دهد.
  - A* هرگز گرهی را با $f(n) > f^*$ هرگز گسترش نمی دهد.
- A* برای هر تابع هیوریستیک قابل قبول، دارای کارایی بهینه می باشد. (Optimally efficient) می باشد. یعنی نسبت به دیگر الگوریتمهای بهینه کمترین تعداد گره را گسترش می دهد.

Email: ComputerCollege_Fam@Yahoo.Com

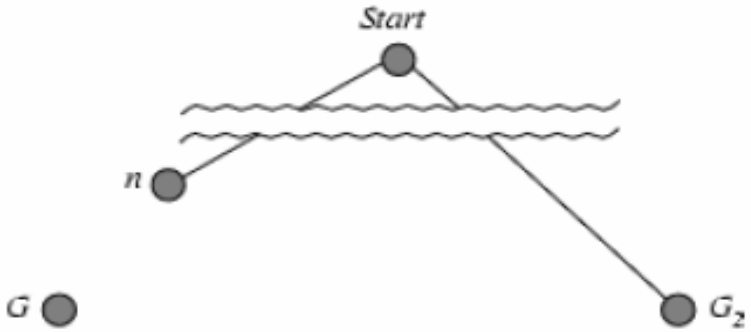
اثبات بهینه بودن A*

- اگر G حالت هدف بهینه با هزینه مسیر h* باشد.
- فرض کنید G2 یک هدف زیر بهینه (SubOptimal)می باشد. که هزینه مسیر g(G2)>h*
- نشان میدهیم که بسط G2 و پایان الگوریتم با یافتن این هدف غیر ممکن است
- اگر n یک گره بسط داده نشده از صف باشد که بر روی کوتاهترین مسیر به سمت هدف بهینه G باشد.



Email: ComputerCollege_Fam@Yahoo.Com

اثبات بهینه بودن A*



چون، $h(G2) = 0$

چون، $G2$ زیر بهینه است

چون، h قابل قبول است

$f(G_2) = g(G_2)$

$> g(G)$

$\geq f(n)$

- $f(G_2) > f(n)$ در نتیجه A^* هرگز $G2$ را برای بسط انتخاب نمی کند.

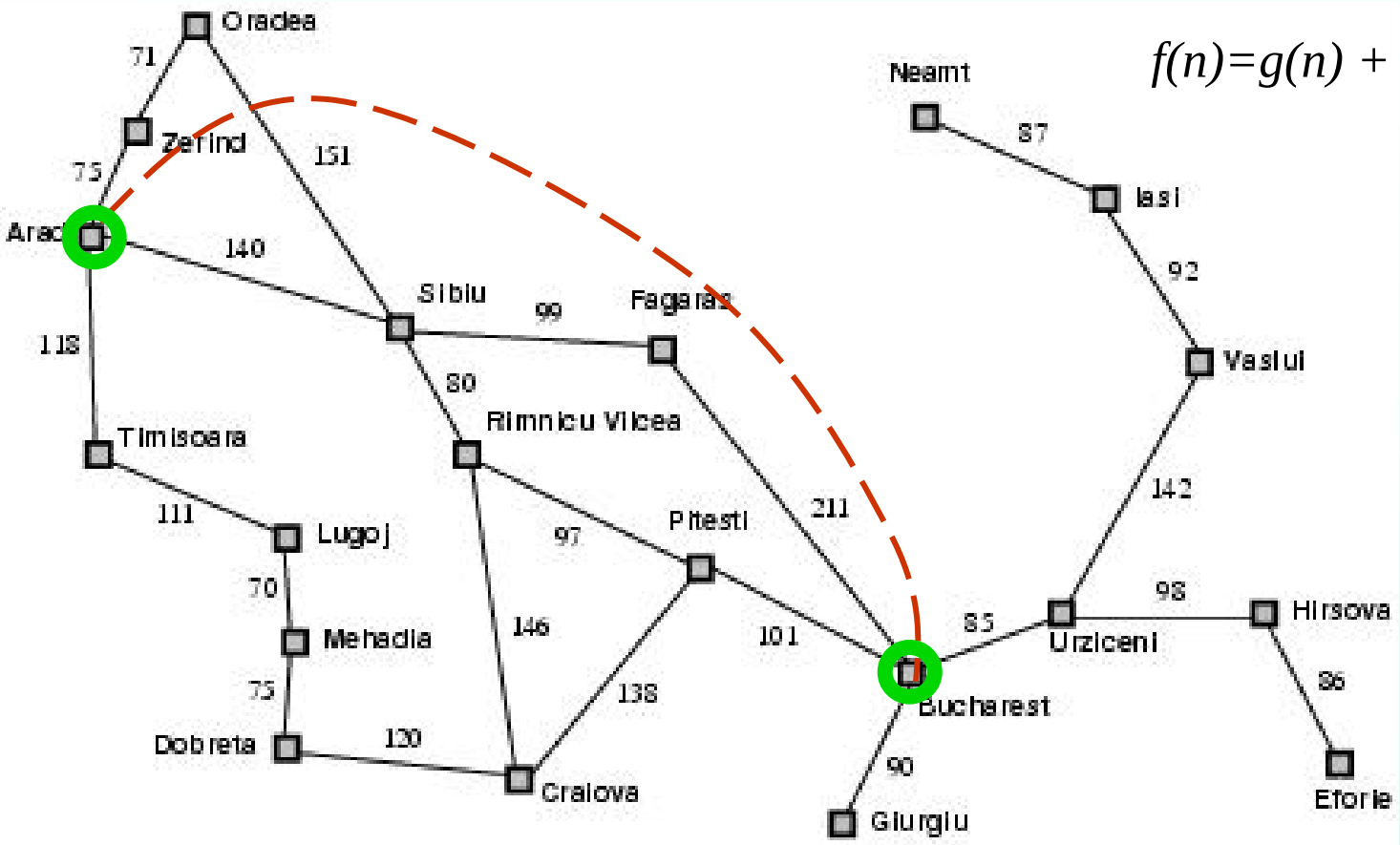
جستجوی A*

- 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 $O(bd)$

Email: ComputerCollege_Fam@Yahoo.Com

مثال دیگر از جستجوی A*

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



Email: ComputerCollege_Fam@Yahoo.Com

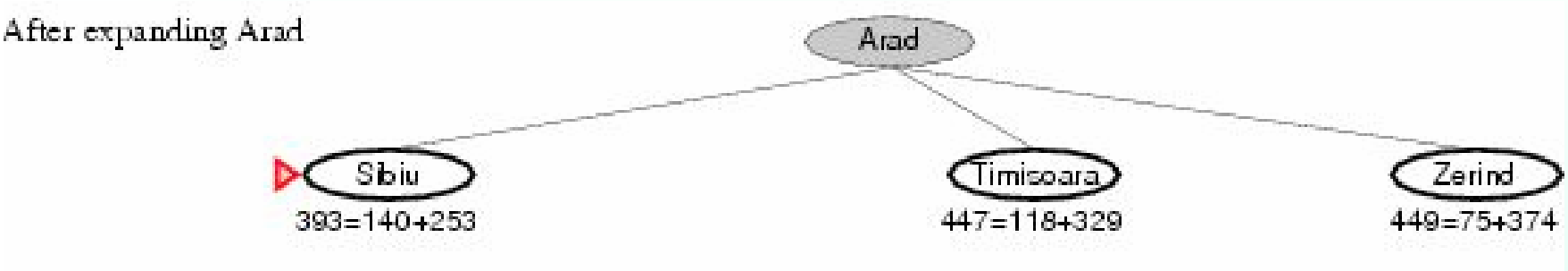
جستجوی A* در نقشه رومانی

(a) The initial state

جستجوی Bucharest با شروع از Arad

f(Arad) = g(Arad)+h(Arad)=۰+۳۶۶=۳۶۶

جستجوی A* در نقشه رومانی



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

f(Sibiu)=c(Arad,Sibiu)+h(Sibiu)=۱۴۰+۲۵۳=۳۹۳

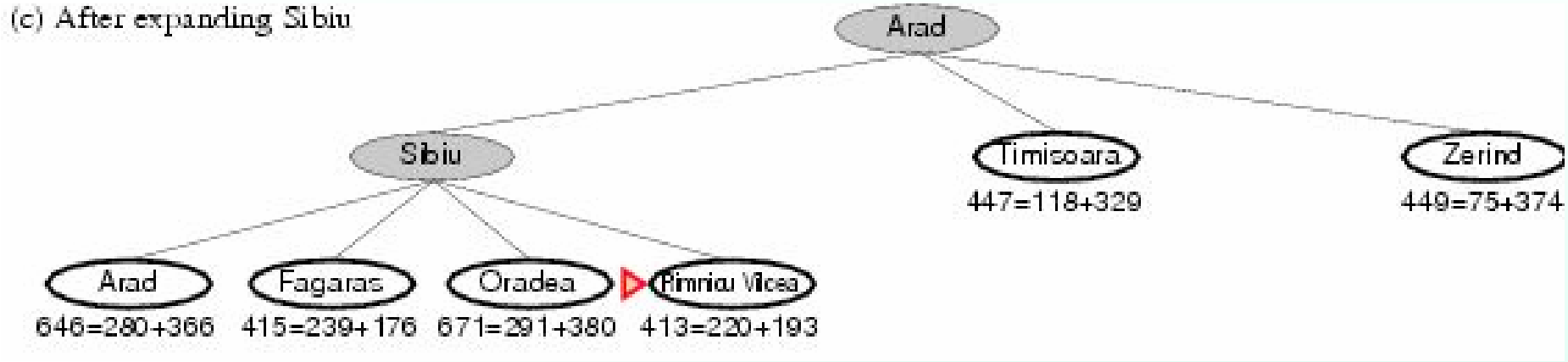
f(Timisoara)=c(Arad,Timisoara)+h(Timisoara)=۱۱۸+۳۲۹=۴۴۷

f(Zerind)=c(Arad,Zerind)+h(Zerind)=۷۵+۳۷۴=۴۴۹

بهترین انتخاب شهر Sibiu است

Email: ComputerCollege_Fam@Yahoo.Com

جستجوی A* در نقشه رومانی



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

f(Arad)=c(Sibiu,Arad)+h(Arad)=۲۸۰+۳۶۶=۶۴۶

f(Fagaras)=c(Sibiu,Fagaras)+h(Fagaras)=۲۳۹+۱۷۹=۴۱۵

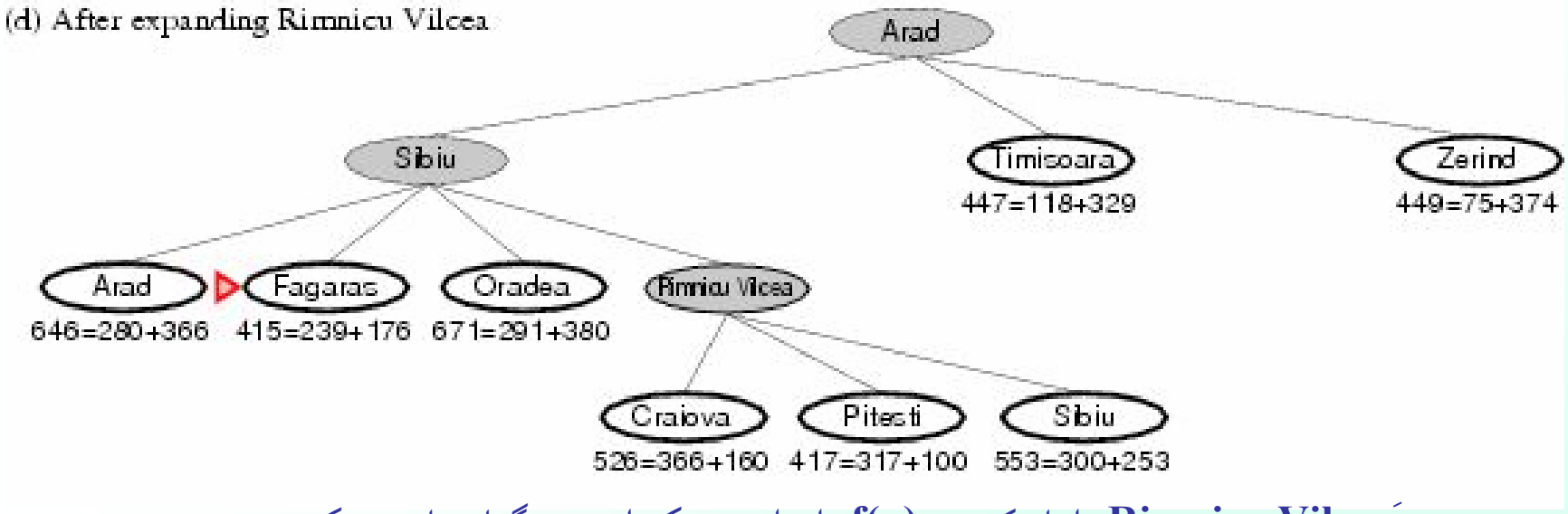
f(Oradea)=c(Sibiu,Oradea)+h(Oradea)=۲۹۱+۳۸۰=۶۷۱

f(Rimnicu Vilcea)=c(Sibiu,Rimnicu Vilcea)+ h(Rimnicu Vilcea)=۲۲۰+۱۹۲=۴۱۳

بهترین انتخاب شهر **Rimnicu Vilcea** است

Email: ComputerCollege_Fam@Yahoo.Com

جستجوی A* در نقشه رومانی



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

f(Craiova)=c(Rimnicu Vilcea, Craiova)+h(Craiova)=۳۶۰+۱۶۰=۵۲۶

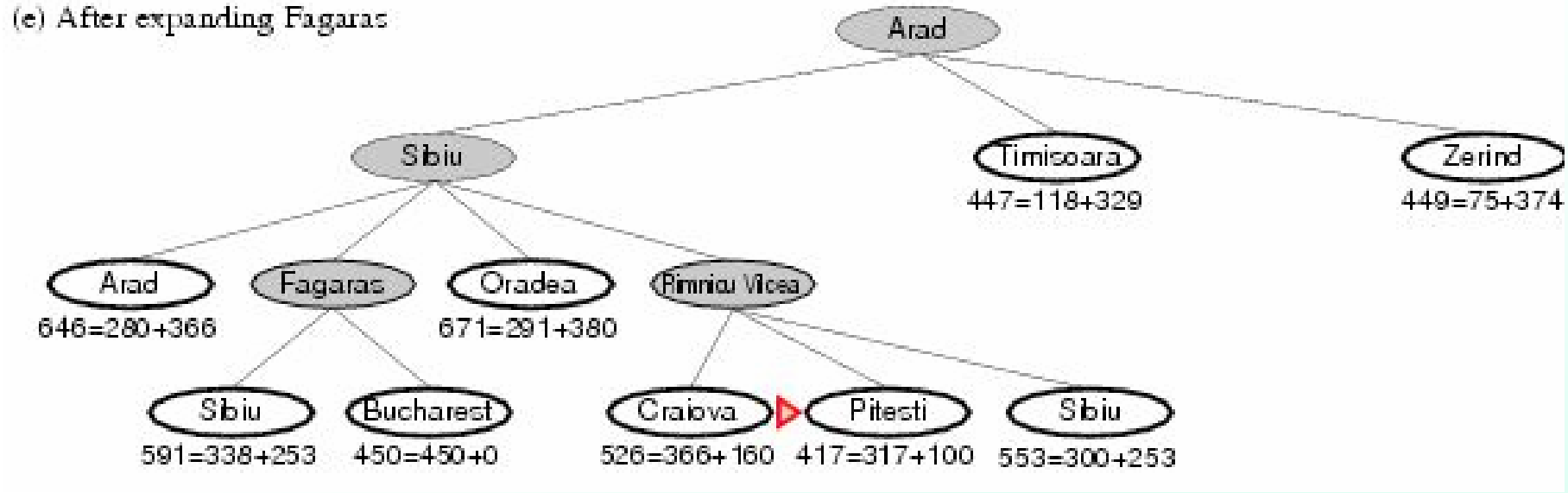
f(Pitesti)=c(Rimnicu Vilcea, Pitesti)+h(Pitesti)=۳۱۷+۱۰۰=۴۱۷

f(Sibiu)=c(Rimnicu Vilcea,Sibiu)+h(Sibiu)=۳۰۰+۲۵۳=۵۵۳

بهترین انتخاب شهر Fagaras است

Email: ComputerCollege_Fam@Yahoo.Com

جستجوی A* در نقشه رومانی



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

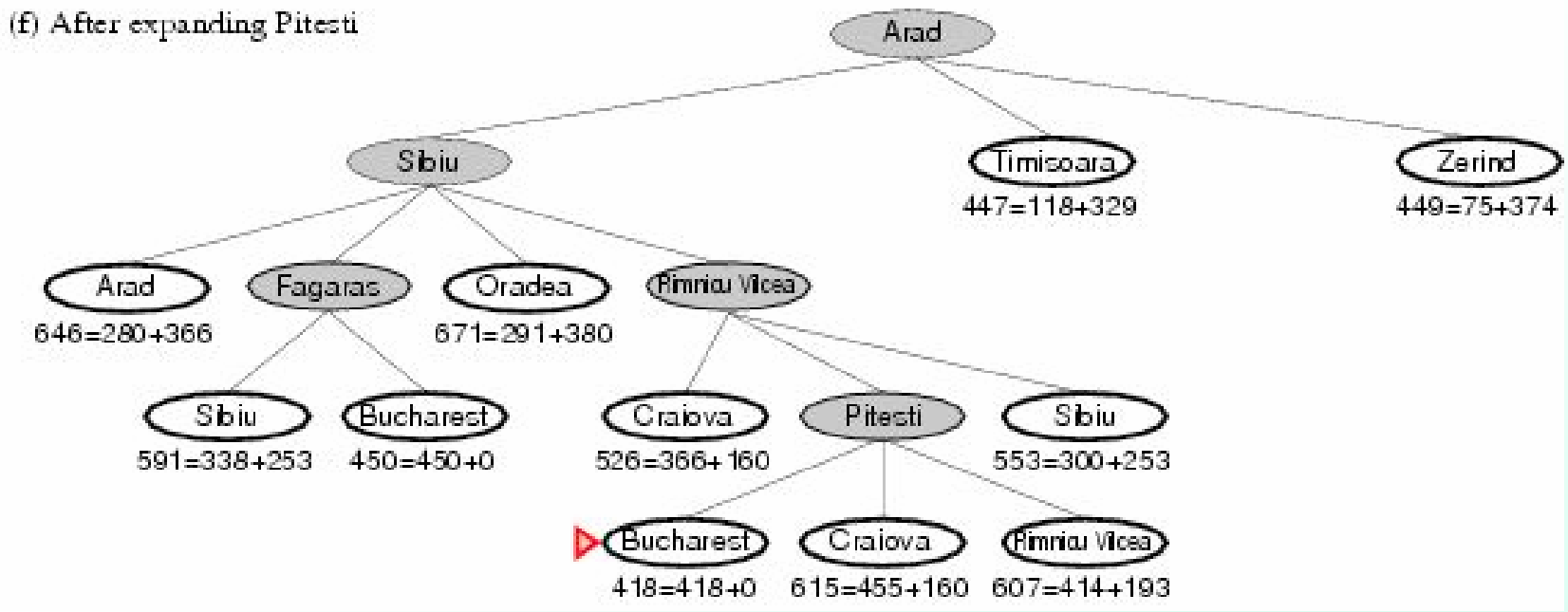
f(Sibiu)=c(Fagaras, Sibiu)+h(Sibiu)=۳۳۸+۲۵۳=۵۹۱

f(Bucharest)=c(Fagaras,Bucharest)+h(Bucharest)=۴۵۰+۰=۴۵۰

بهترین انتخاب شهر **Pitesti !!!** است

Email: ComputerCollege_Fam@Yahoo.Com

جستجوی A* در نقشه رومانی



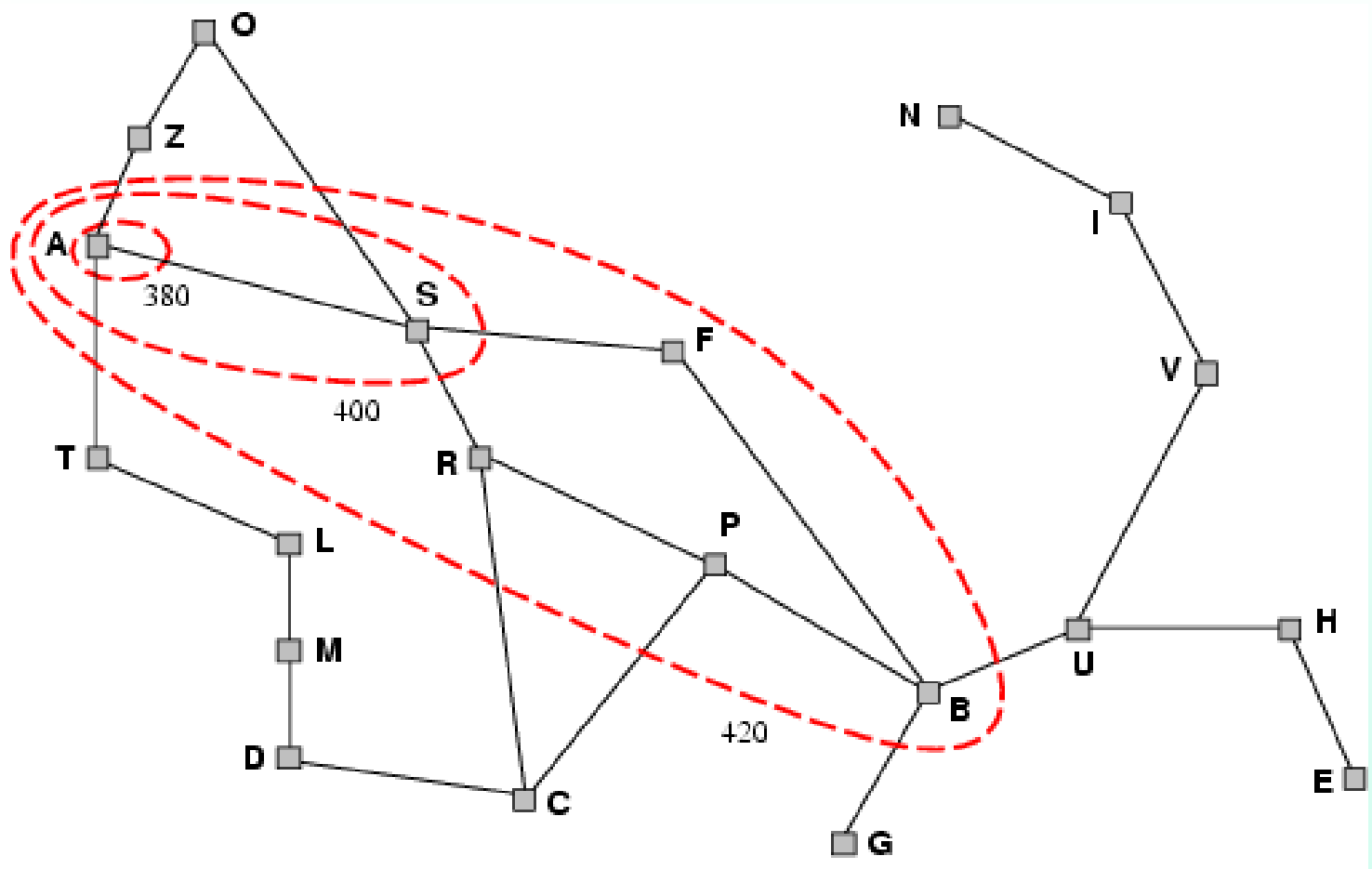
Pitesti را باز کرده و **f(n)** را برای هر یک از زیربرگها محاسبه میکنیم:

f(Bucharest)=c(Pitesti,Bucharest)+h(Bucharest)=۴۱۸+۰=۴۱۸

بهترین انتخاب شهر **Bucharest !!!** است

Email: ComputerCollege_Fam@Yahoo.Com

جستجوی A* در نقشه رومانی



Email: ComputerCollege_Fam@Yahoo.Com

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.
- IDA* برای اغلب مسئله های با هزینه های مرحله ای، مناسب است و از سربار ناشی از نگهداری صف مرتبی از گره ها اجتناب میکند

Email: ComputerCollege_Fam@Yahoo.Com

بهترین جستجوی بازگشتی RBFS

- ساختار آن شبیه جست و جوی عمقی بازگشتی است، اما به جای اینکه دائما به طرف پایین مسیر حرکت کند، مقدار f مربوط به بهترین مسیر از هر جد گره فعلی را نگهداری می کند، 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.

Email: ComputerCollege_Fam@Yahoo.Com

بهترین جستجوی بازگشتی RBFS

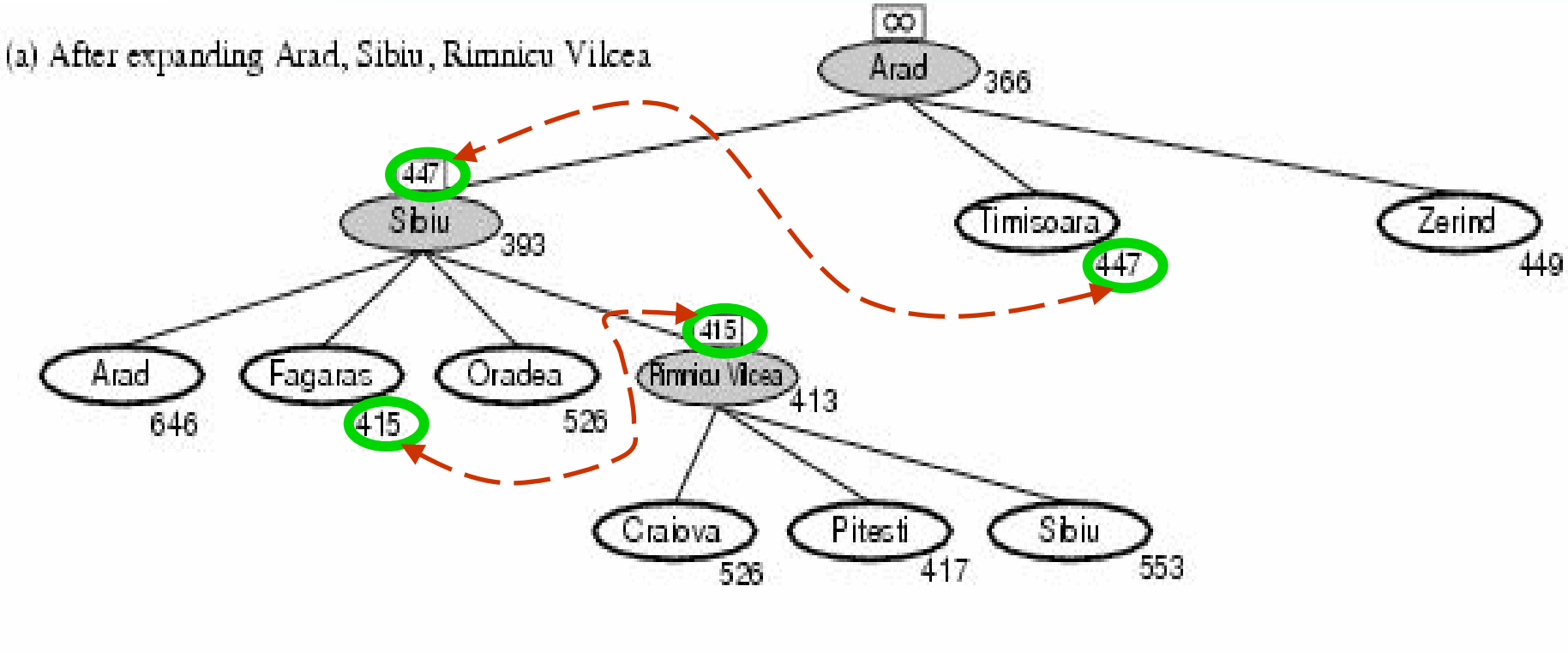
- RBFS تا حدی از IDA* کارآمدتر است، اما گره های زیادی تولید می کند.

- IDA* و RBFS در معرض افزایش توانی پیچیدگی قرار دارند که در جست و جوی گراف ها مرسوم است، زیرا نمی توانند 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 می کنند که به آنها آسیب می رساند. IDA* بین هر تکرار فقط یک عدد را نگهداری می کند که فعلی هزینه f است. RBFS اطلاعات بیشتری در حافظه نگهداری می کند.

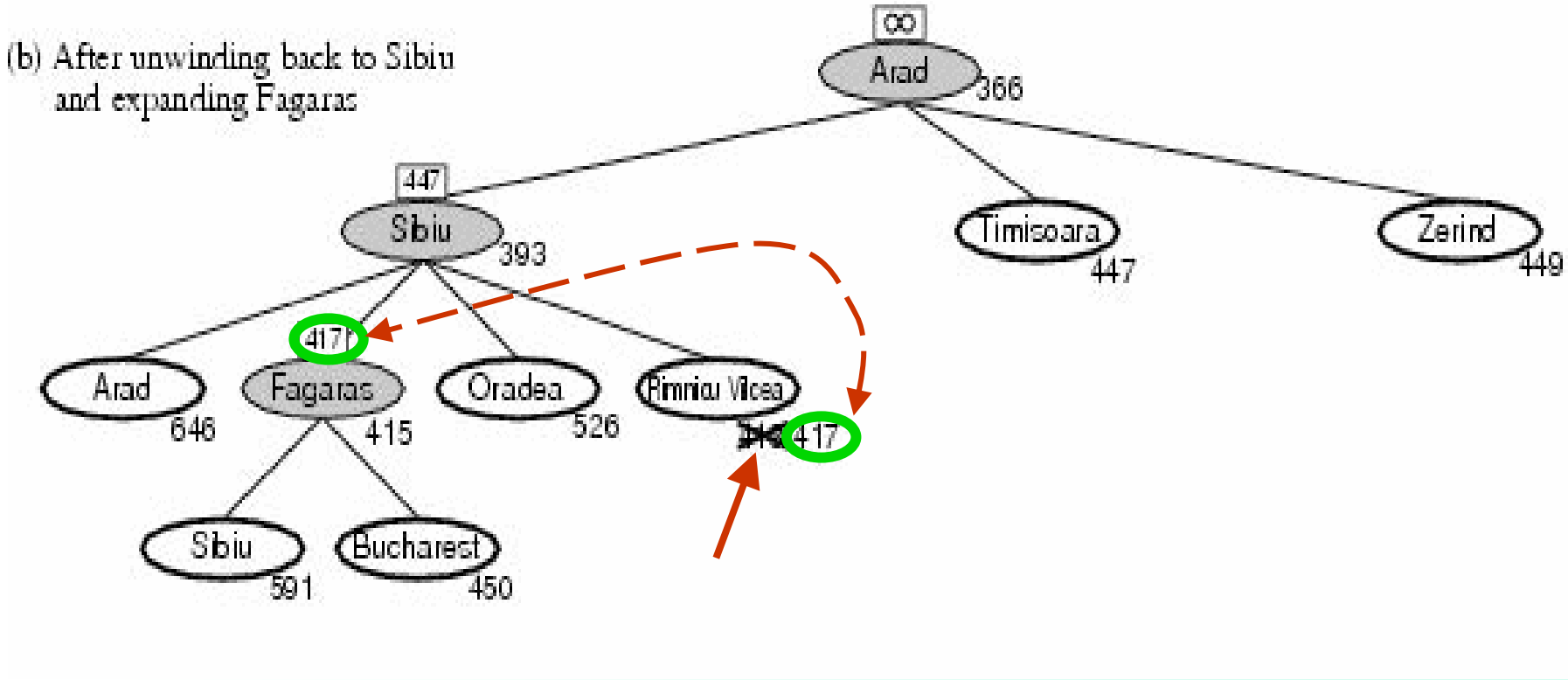
Email: ComputerCollege_Fam@Yahoo.Com

بهترین جستجوی بازگشتی در نقشه رومانی



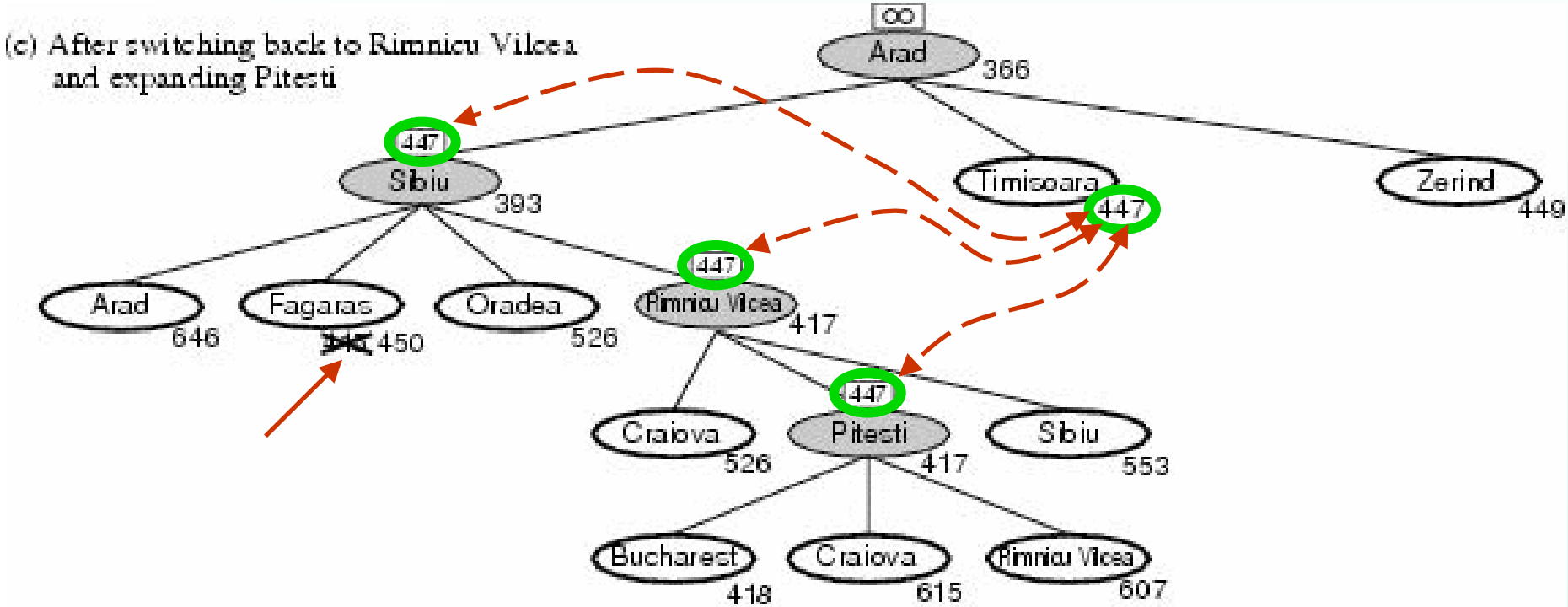
Email: ComputerCollege_Fam@Yahoo.Com

بهترین جستجوی بازگشتی در نقشه رومانی



Email: ComputerCollege_Fam@Yahoo.Com

بهترین جستجوی بازگشتی در نقشه رومانی



Email: ComputerCollege_Fam@Yahoo.Com

```
function RECURSIVE-BEST-FIRST-SEARCH(problem) returns a solution, or failure
    RBFS(problem, MAKE-NODE(INITIAL-STATE[problem]), ∞)

function RBFS(problem, node, f_limit) returns a solution, or failure and a new f-cost limit
    if GOAL-TEST[problem](STATE[node]) then return node
    successors ← EXPAND(node, problem)
    if successors is empty then return failure, ∞
    for each s in successors do
        f[s] ← max(g(s) + h(s), f[node])
    repeat
        best ← the lowest f-value node in successors
        if f[best] > f_limit then return failure, f[best]
        alternative ← the second-lowest f-value among successors
        result, f[best] ← RBFS(problem, best, min(f_limit, alternative))
        if result ≠ failure then return result
```

الگوریتم جستجوی اول- بهترین بازگشتی

Email: ComputerCollege_Fam@Yahoo.Com

جستجوی حافظه محدود ساده SMA*

- SMA* بهترین برگ را بسط می دهد تا حافظه پر شود. در این نقطه بدون از بین بردن گره های قبلی نمی تواند گره جدیدی اضافه کند
- SMA* همیشه بدترین گره برگ را حذف می کند و سپس از طریق گره فراموش شده به والد آن بر می گردد. پس جد زیر درخت فراموش شده، کیفیت بهترین مسیر را در آن زیر درخت می داند
- اگر عمق سطحی ترین گره هدف کمتر از حافظه باشد، کامل است.
- SMA* بهترین الگوریتم همه منظوره برای یافتن حل های بهینه می باشد

Email: ComputerCollege_Fam@Yahoo.Com

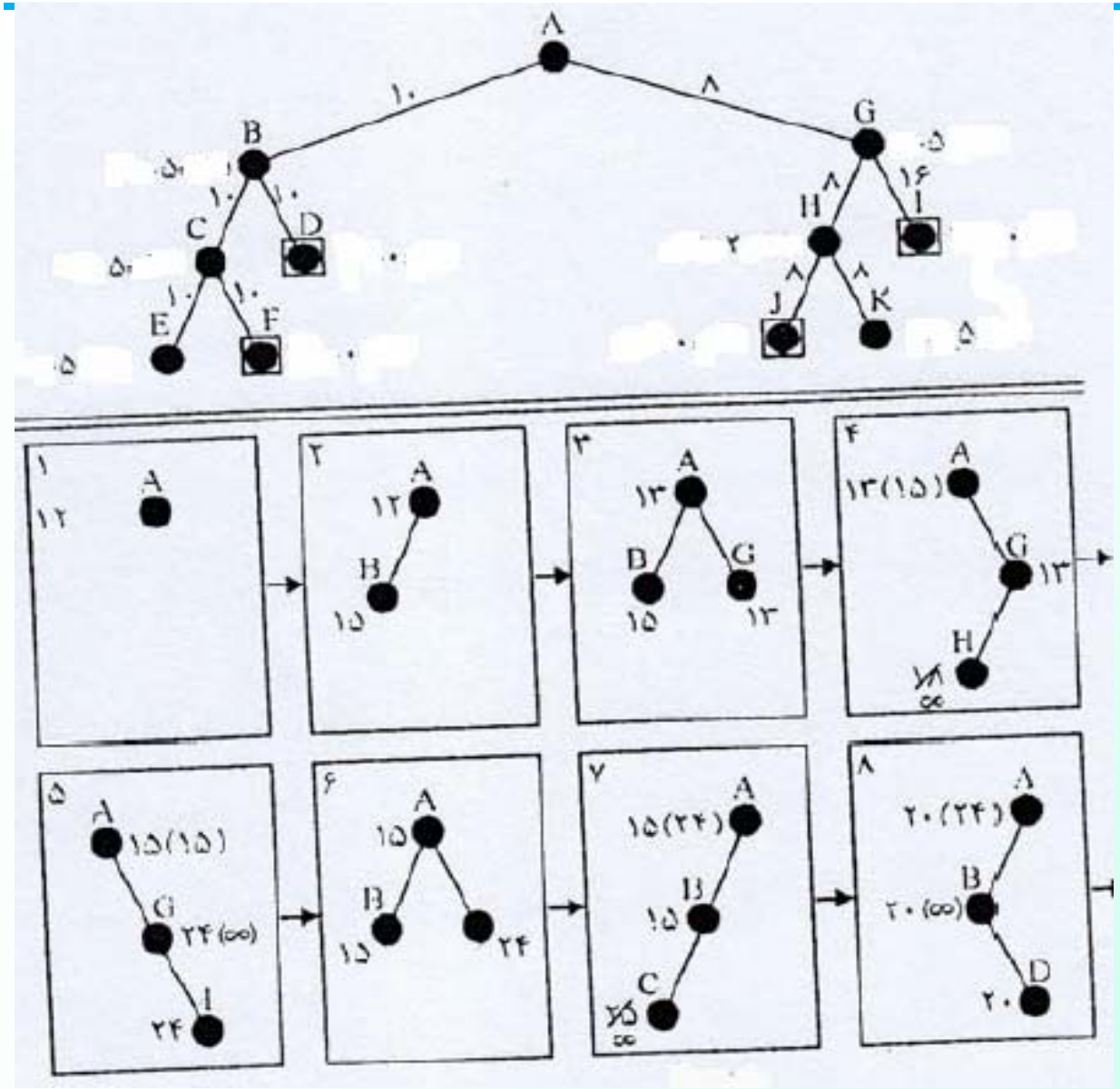
جستجوی حافظه محدود ساده SMA*

- اگر مقدار f تمام برگ ها 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 می کند

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

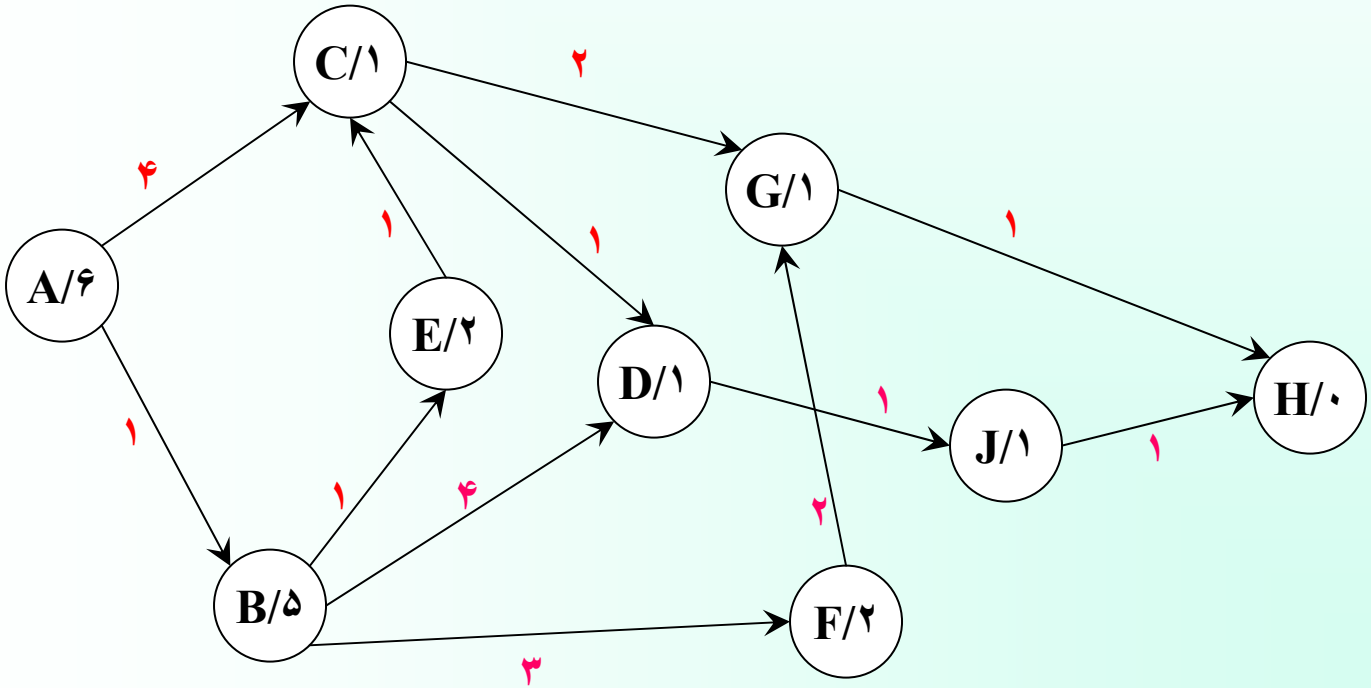
- محدودیت های حافظه ممکن است مسئله ها را از نظر زمان محاسباتی، غیر قابل حل کند.

Email: ComputerCollege_Fam@Yahoo.Com

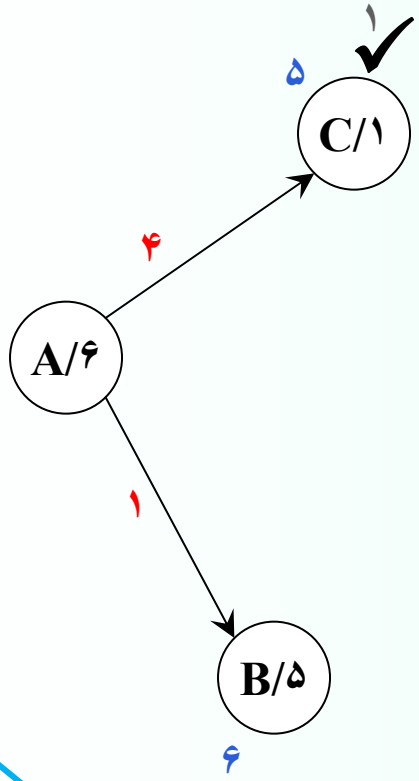


Email: ComputerCollege_Fam@Yahoo.Com

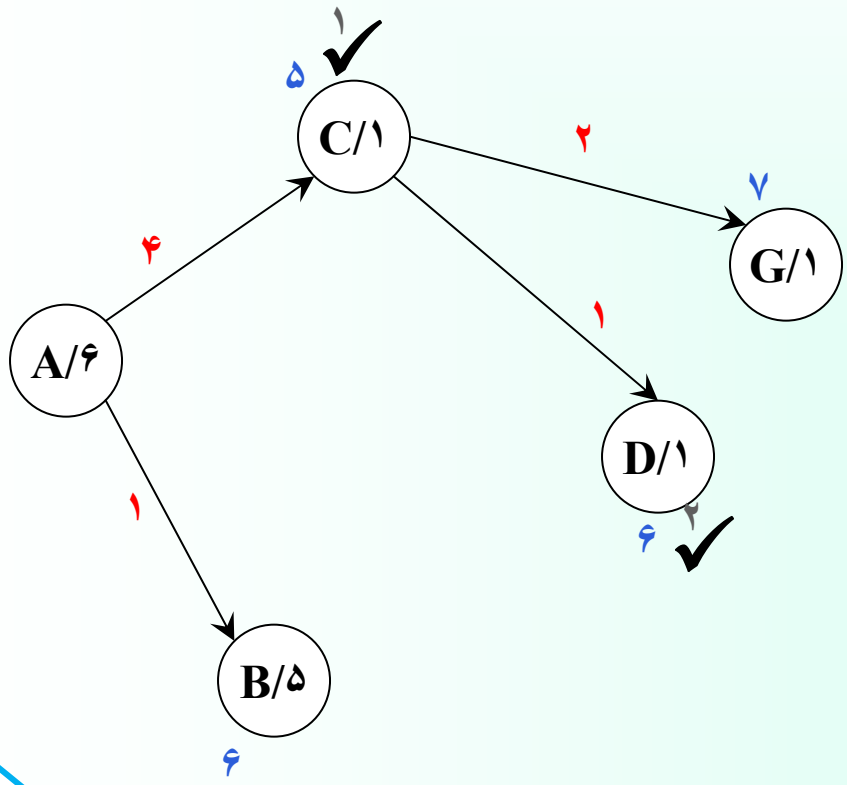
جستجوی گراف با A*



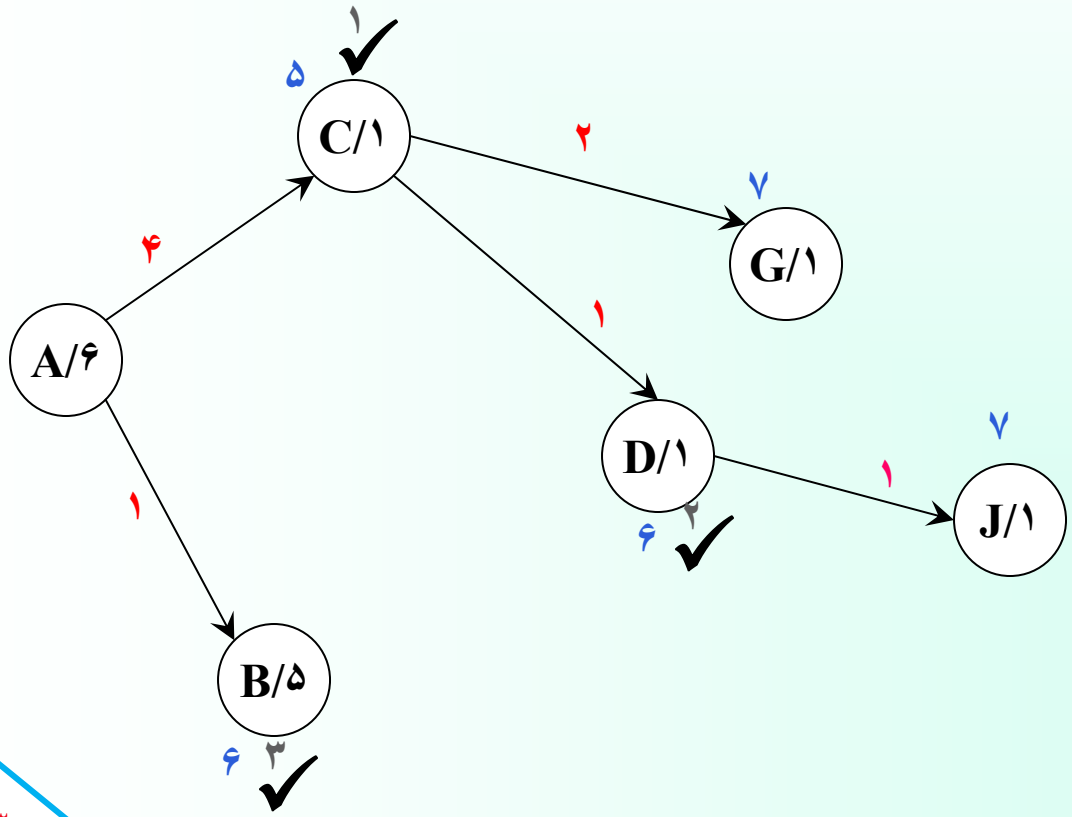
جستجوی گراف با A*



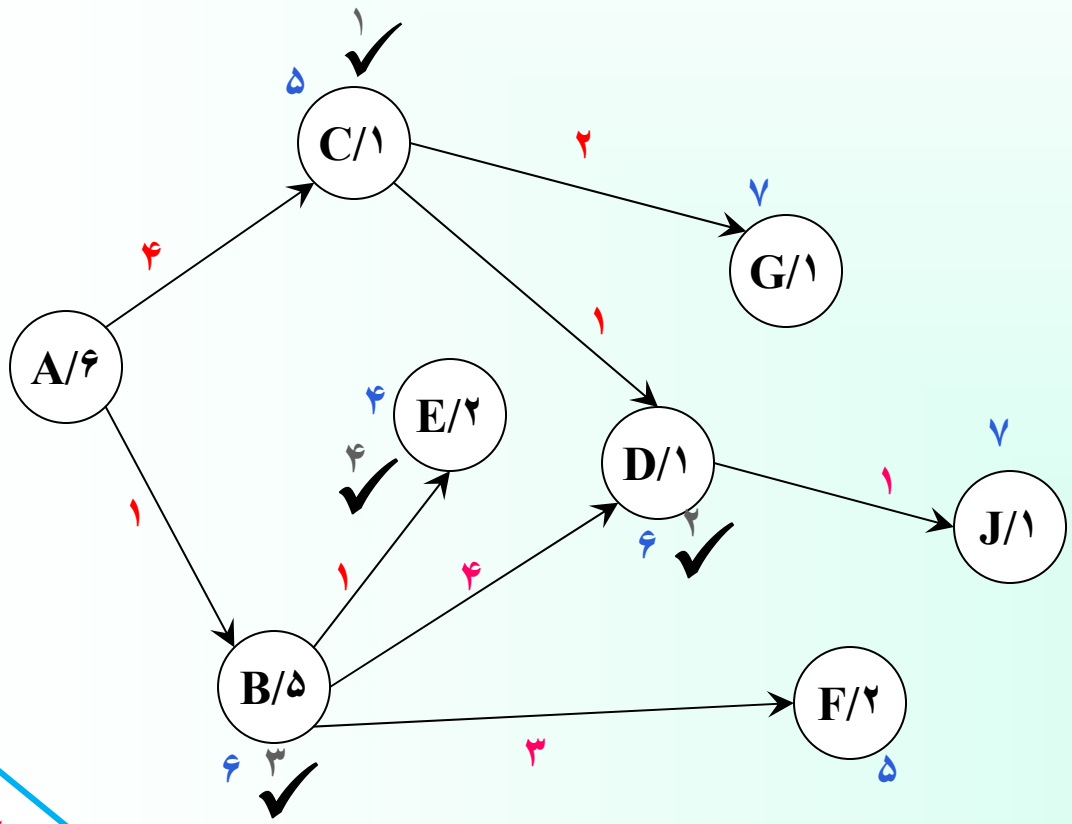
جستجوی گراف با A*



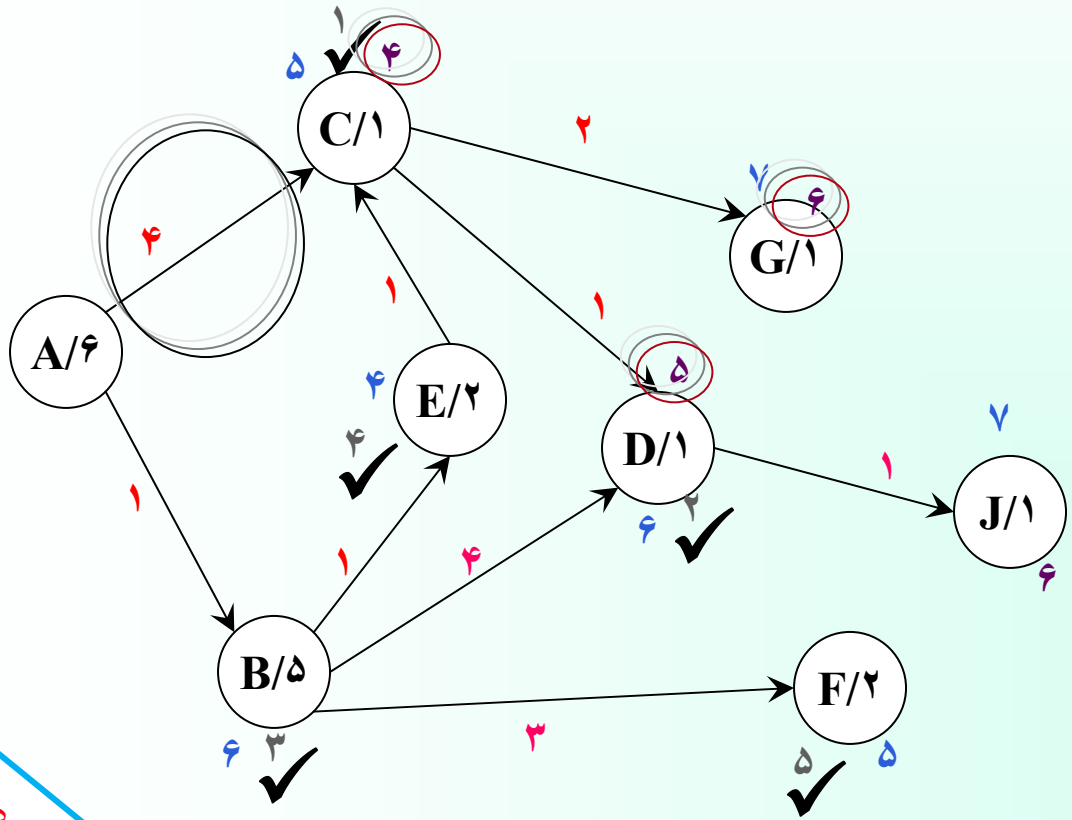
جستجوی گراف با A*



جستجوی گراف با A*

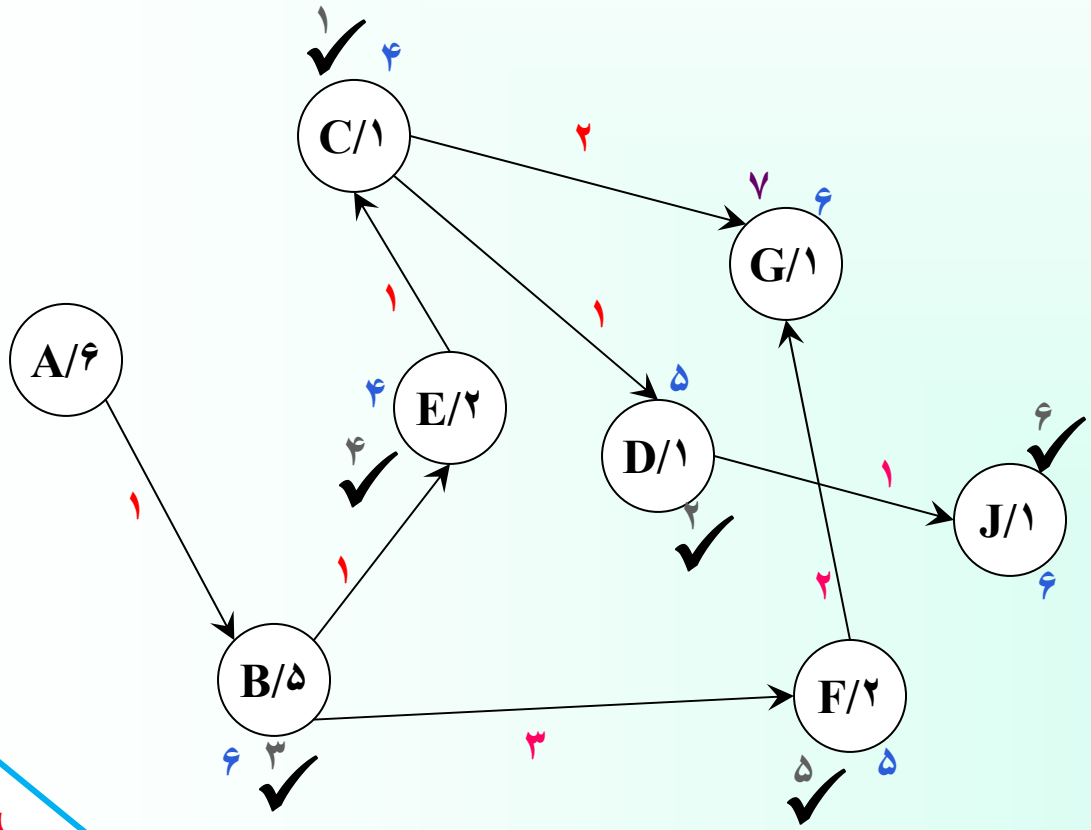


جستجوی گراف با A*



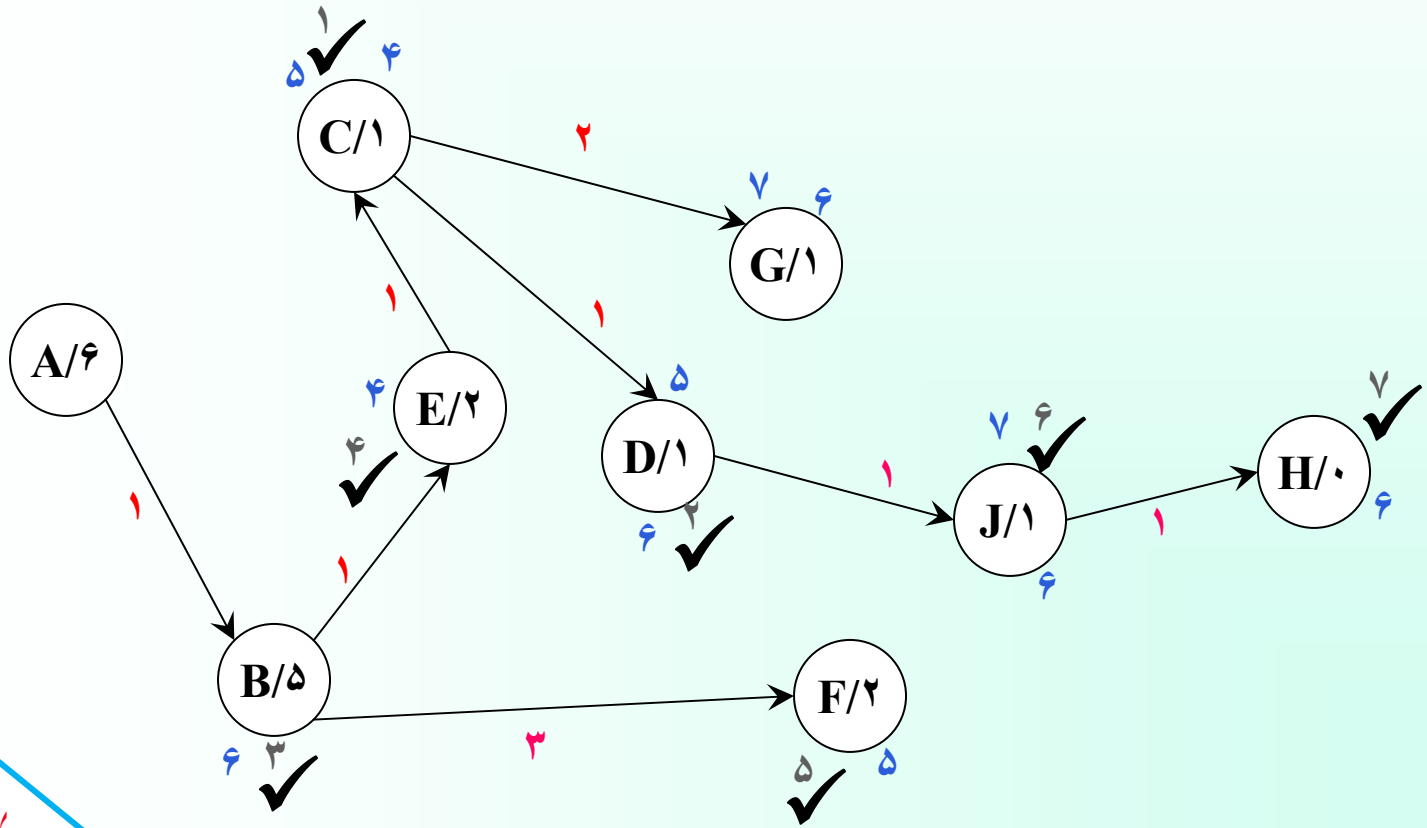
Email: ComputerCollege_Fam@Yahoo.Com

جستجوی گراف با A*



Email: ComputerCollege_Fam@Yahoo.Com

جستجوی گراف با A*

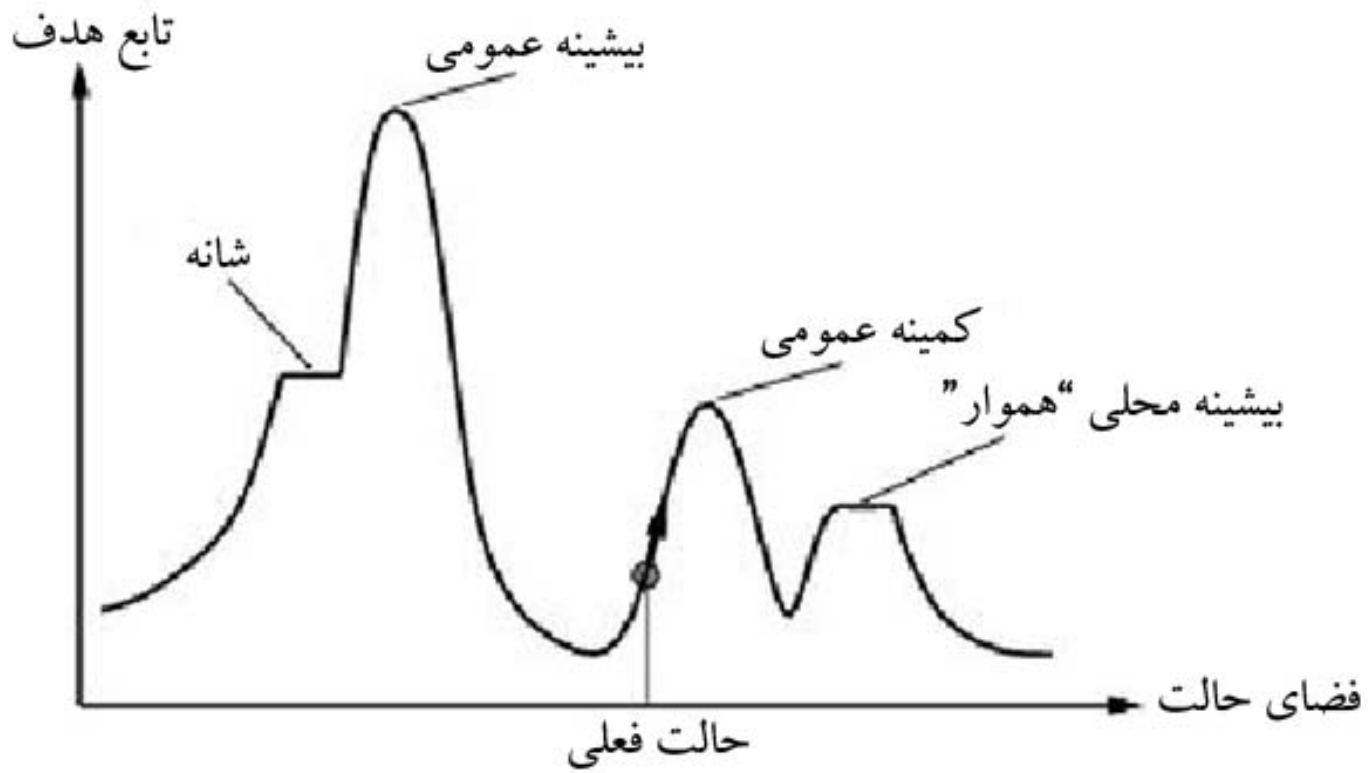


الگوریتم های جست و جوی محلی و بهینه سازی

- 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 ۸ وزیر
- دو امتیاز عمده جست و جوهای محلی
  - استفاده از حافظه کمکی
  - ارائه راه حلهای منطقی در فضاهای بزرگ و نامتناهی
- این الگوریتم ها برای حل مسائل بهینه سازی نیز مفیدند
  - یافتن بهترین حالت بر اساس تابع هدف

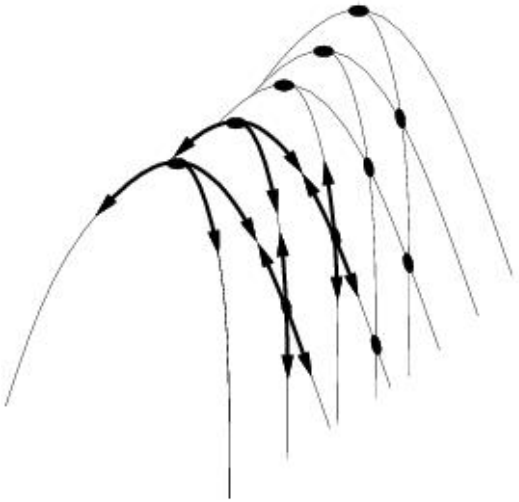
Email: ComputerCollege_Fam@Yahoo.Com

الگوریتم های جست و جوی محلی و بهینه سازی



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 حریصانه نیز نام دارد
  - بدون فکر قبلی حالت همسایه خوبی را انتخاب میکند
- تپه نوردی به دلایل زیر میتواند متوقف شود:
  - بیشینه محلی
  - برآمدگی ها
  - فلات



جست و جوی تپه نوردی

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

مثال: مسئله ۸ وزیر

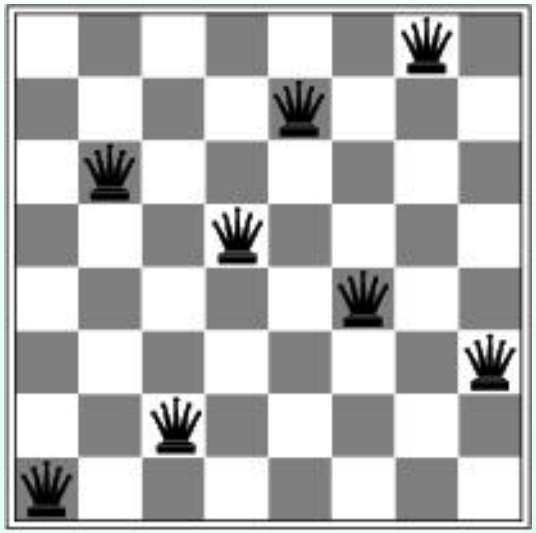
- مسئله ۸ وزیر با استفاده از فرمولبندی حالت کامل
  - در هر حالت ۸ وزیر در صفحه قرار دارند
- تابع جانشین: انتقال یک وزیر به مربع دیگر در همان ستون
- تابع اکتشاف: جفت وزیرهایی که نسبت به هم گارد دارند
  - مستقیم یا غیر مستقیم

Email: ComputerCollege_Fam@Yahoo.Com

مثال جست و جوی تپه نوردی

الف

ب



الف- حالت با هزینه h=۱۷ که مقدار h را برای هر جانشین نشان میدهد

ب- کمینه محلی در فضای حالت ۸ وزیر؛ h=۱

Email: ComputerCollege_Fam@Yahoo.Com

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 می یابد

Email: ComputerCollege_Fam@Yahoo.Com

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 فرایند موازی مبادله میشود

Email: ComputerCollege_Fam@Yahoo.Com

هوش مصنوعی

فصل پنجم

مسائل ارضای محدودیت

Email: ComputerCollege_Fam@Yahoo.Com

فهرست

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

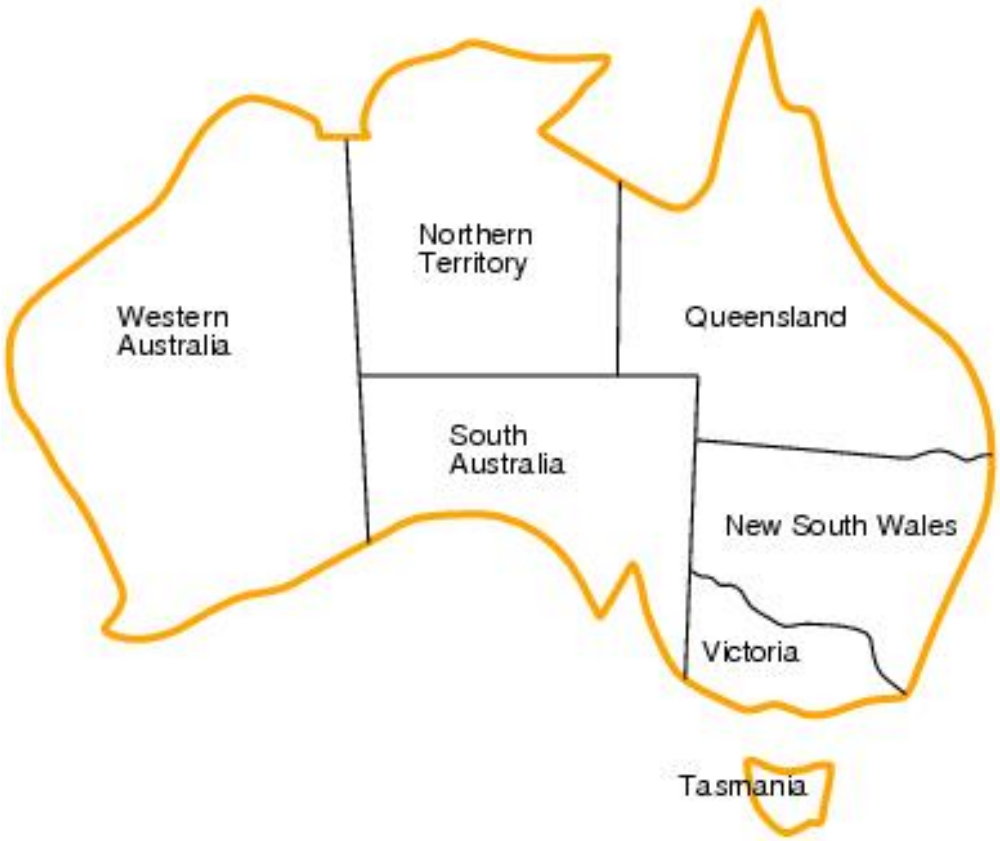
ارضای محدودیت (CSP-Constraint Satisfaction Problems) چیست؟

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

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

Email: ComputerCollege_Fam@Yahoo.Com

مثال CSP: رنگ آمیزی نقشه



متغیرها: WA, NT, Q, NSW, V, SA, T

دامنه: D_i = {آبی، سبز، قرمز}

محدودیت ها: دو منطقه مجاور، همرنگ نیستند

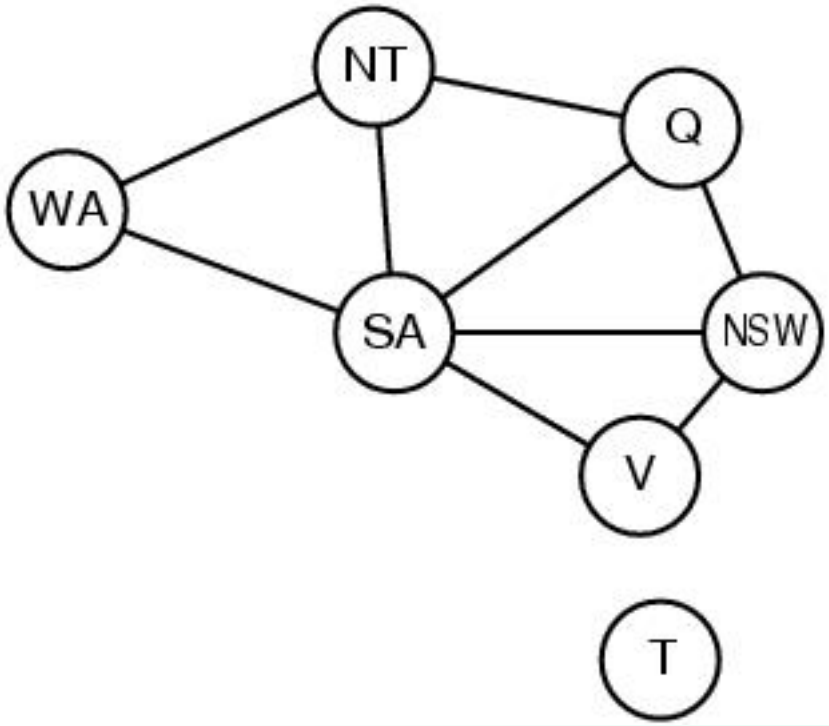
مثال: WA ≠ NT یعنی (WA,NT) عضو

{(قرمز,سبز),(قرمز,آبی),(سبز,قرمز),
(سبز,آبی),(آبی,قرمز),(آبی,سبز)}

Email: ComputerCollege_Fam@Yahoo.Com

راه حل انتساب مقادیری است که محدودیتها را ارضا کند

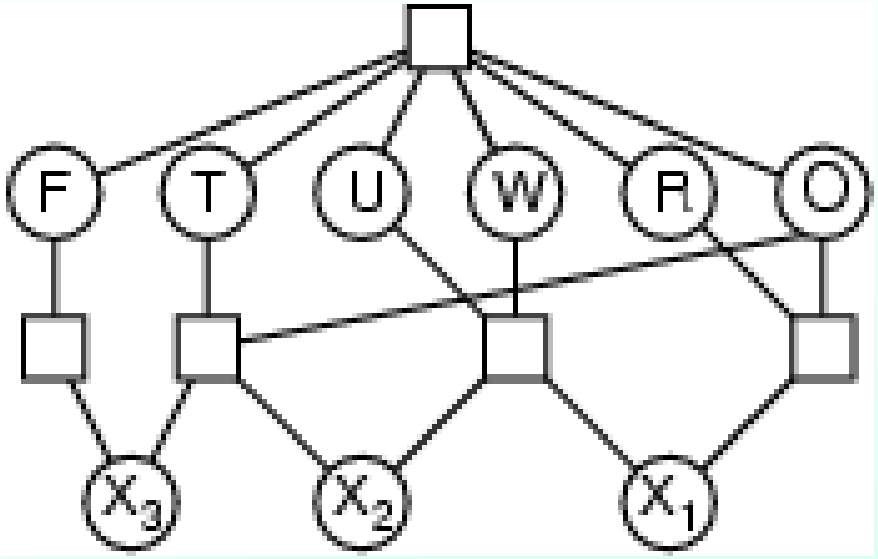
گراف محدودیت



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

مثال CSP: رمزنگاری

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



متغیرها: F,T,U,W,R,O,X۱,X۲,X۳ دامنه: {۰و۱و۲و۳و۴و۵و۶و۷و۸و۹}

محدودیتها: F,T,U,R,O,W مخالفند – O+O=R+۱۰.X۱ – ...

Email: ComputerCollege_Fam@Yahoo.Com

- نمایش حالتها در CSP از الگوی استانداردی 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

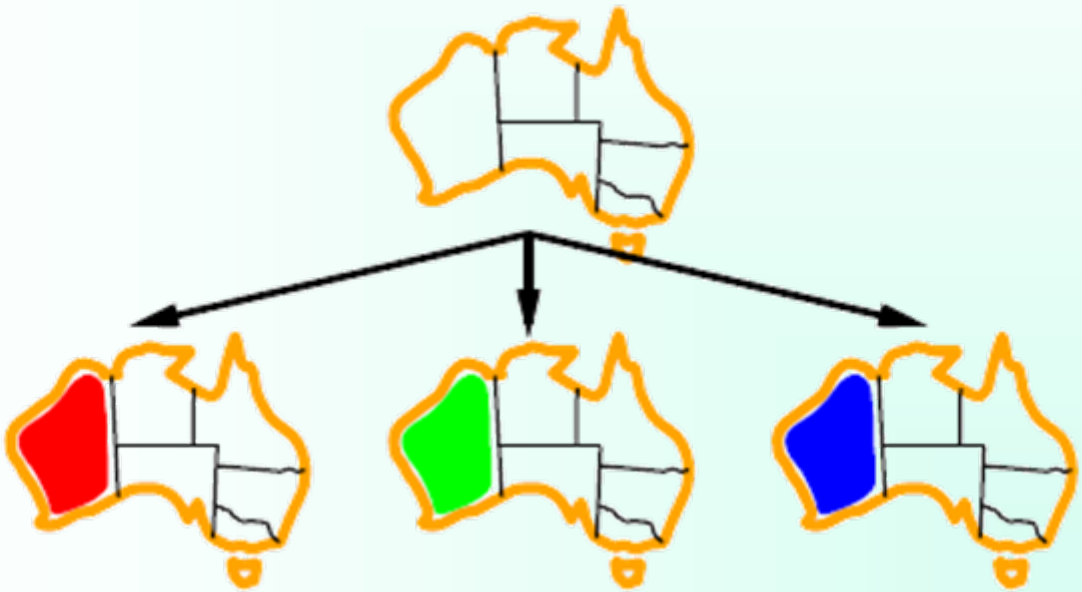
جست و جوی عقبگرد برای CSP

- 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

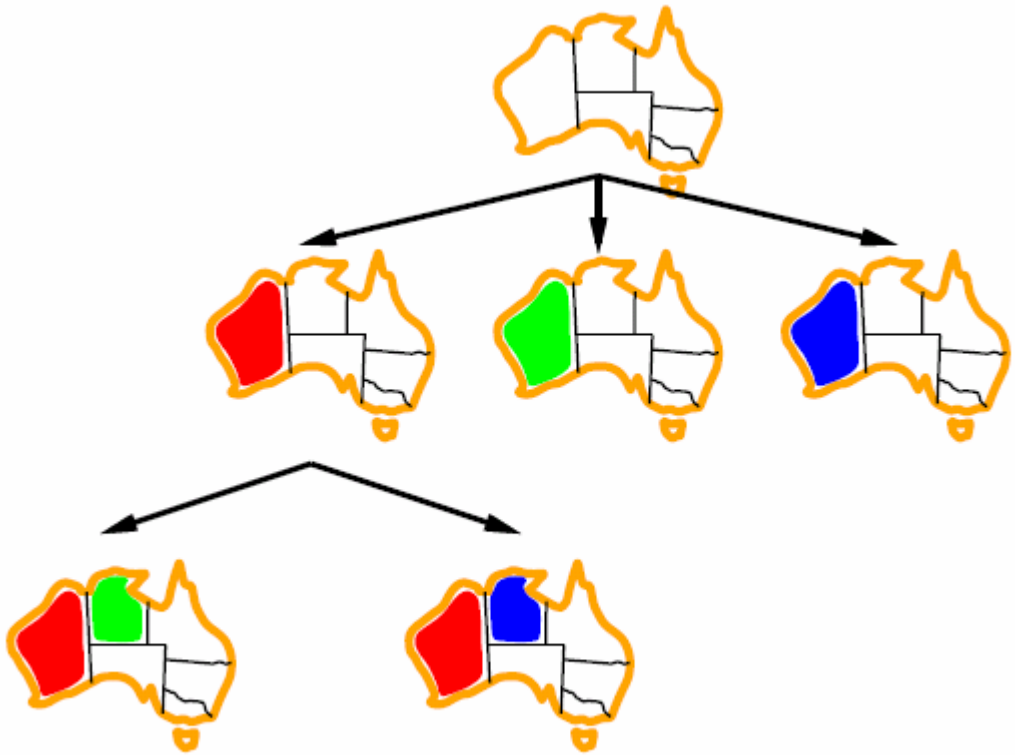
Email: ComputerCollege_Fam@Yahoo.Com

مثال جست و جوی عقبگرد برای CSP

مثال جست و جوی عقبگرد برای CSP



مثال جست و جوی عقبگرد برای CSP



Email: ComputerCollege_Fam@Yahoo.Com

هوش مصنوعی

فصل ششم

جستجوی خصمانه

تئوری بازی ها

Email: ComputerCollege_Fam@Yahoo.Com

فهرست

- بازی ها چیستند و چرا مطالعه می شوند؟
- انواع بازی ها
- الگوریتم **minimax**
- بازی های چند نفره
- هرس آلفا-بتا
- بازی های قطعی با اطلاعات ناقص
- بازی هایی که حاوی عنصر شانس هستند

بازی ها 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 مسئله های خصمانه میشود که به عنوان بازی شناخته میشوند
- چرا مطالعه میشوند؟
  - قابلیتهای هوشمندی انسانها را به کار میگیرند
  - ماهیت انتزاعی بازی ها
  - حالت بازی را به راحتی میتوان نمایش داد و عاملها معمولا به مجموعه کوچکی از فعالیتها محدود هستند که نتایج آنها با قوانین دقیقی تعریف شده اند

Email: ComputerCollege_Fam@Yahoo.Com

انواع بازی ها

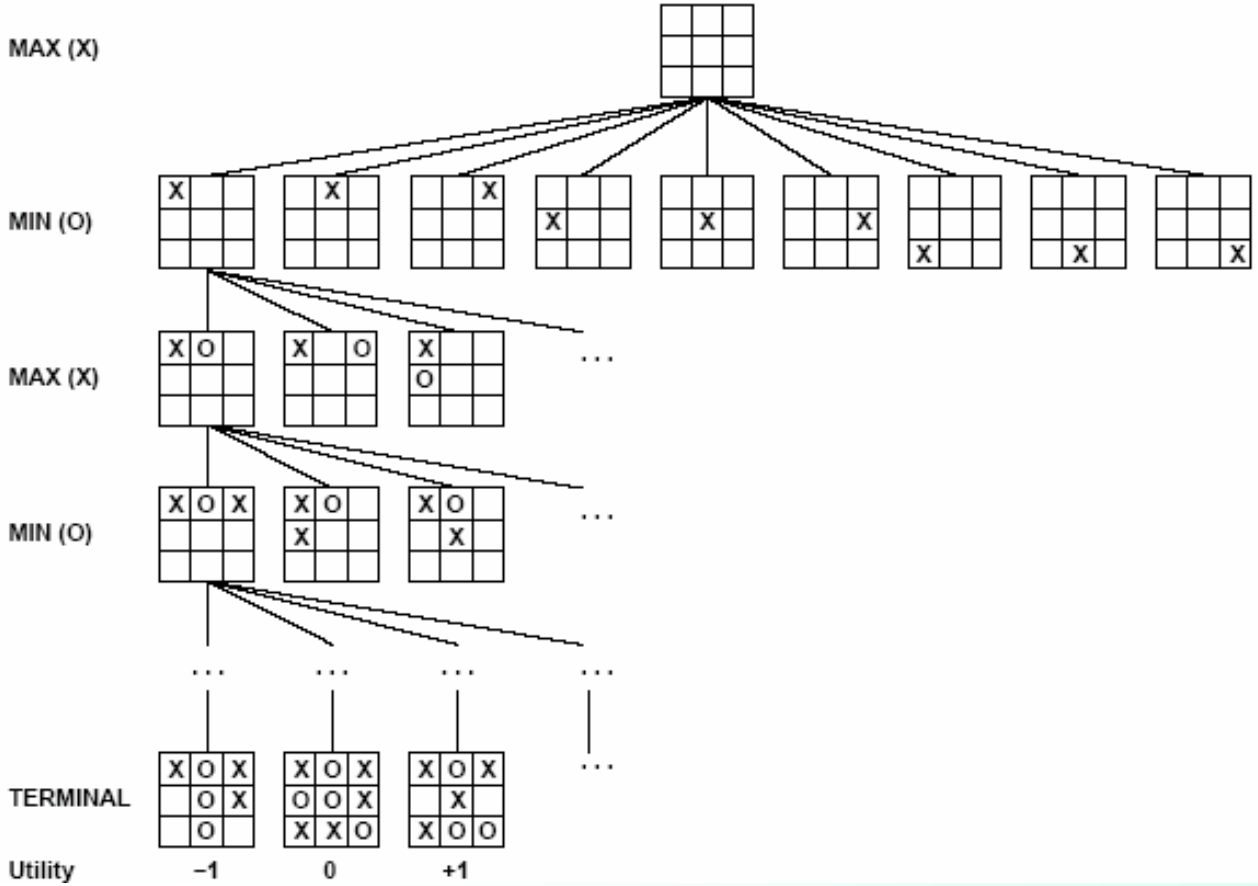
	تصادفی /شانسی	قطعی
اطلاعات کامل	تخته نرد	شطرنج ریورسی
اطلاعات ناقص	پوکر	

یک نمونه بازی

- بازی دو نفره: Min و Max
  - اول Max حرکت میکند و سپس به نوبت بازی میکنند تا بازی تمام شود
  - در پایان بازی، برنده جایزه و بازنده جریمه میشود
- بازی به عنوان یک جستجو:
  - حالت اولیه: موقعیت صفحه و شناسه های قابل حرکت
  - تابع جانشین: لیستی از (حالت,حرکت) که معرف یک حرکت معتبر است
  - آزمون هدف:پایان بازی چه موقع است؟(حالت های پایانه)
  - تابع سودمندی: برای هر حالت پایانه یک مقدار عددی را ارائه میکند. مثلا برنده(+۱) و بازنده(−۱)
- حالت اولیه و حرکات معتبر برای هر بازیکن، **درخت بازی** را برای آن بازی ایجاد می کند

Email: ComputerCollege_Fam@Yahoo.Com

یک نمونه بازی



الگوریتم؛

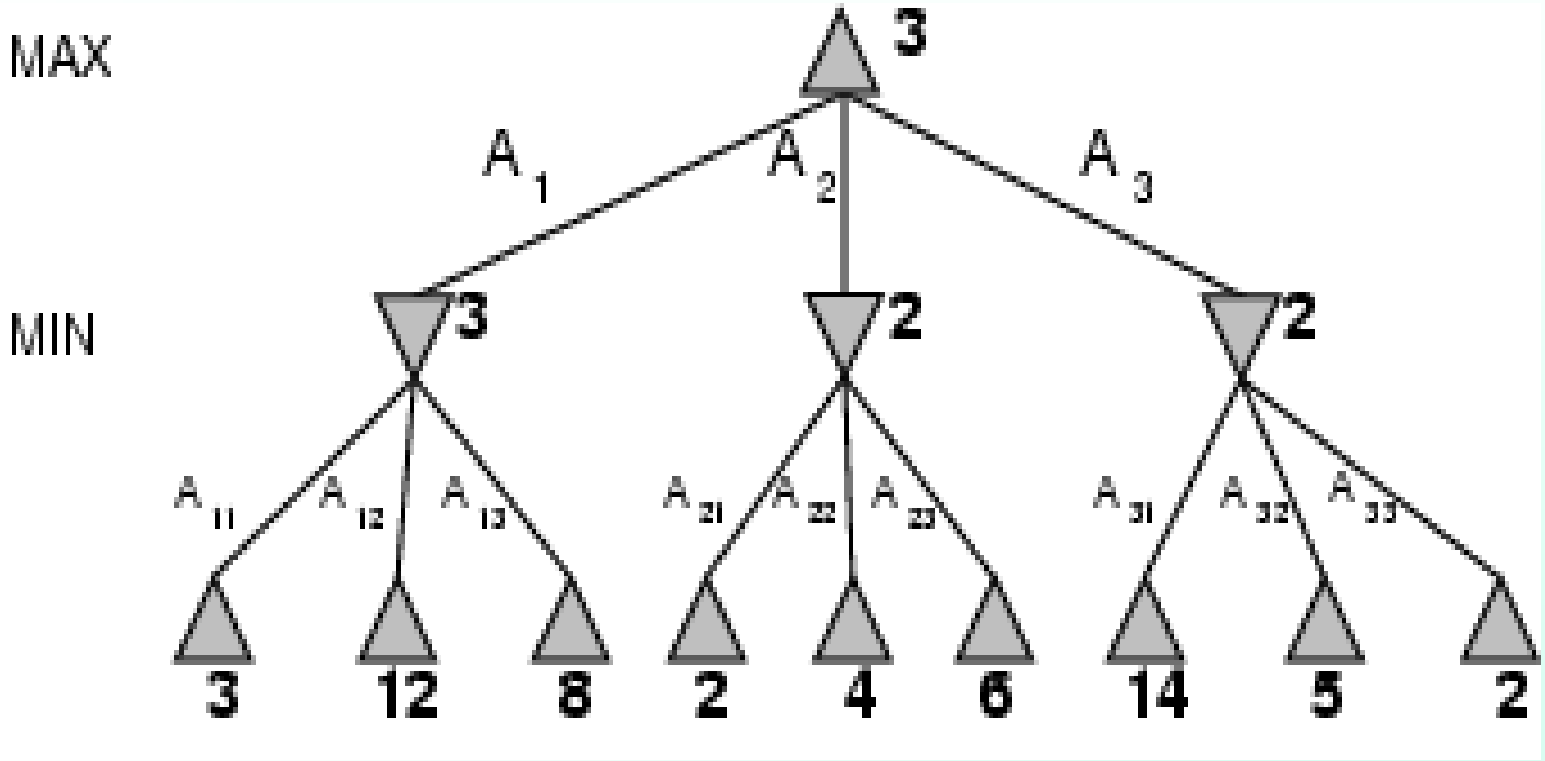
- بازیکن: انتخاب بهترین حالت
- حریف: انتخاب بهترین موقعیت برای خودش یا بدترین وضعیت برای بازیکن

بازیکن: ماکزیمم حالت

حریف: مینیمم حالت

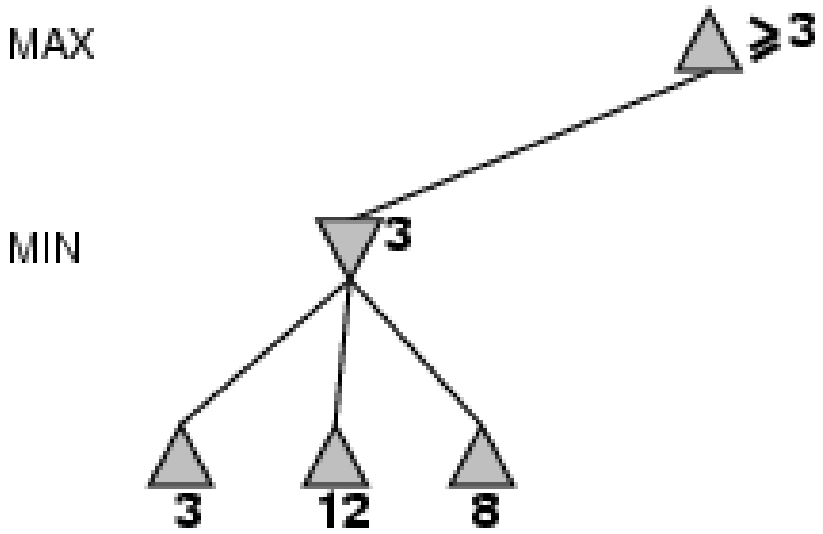
Email: ComputerCollege_Fam@Yahoo.Com

الگوریتم minimax

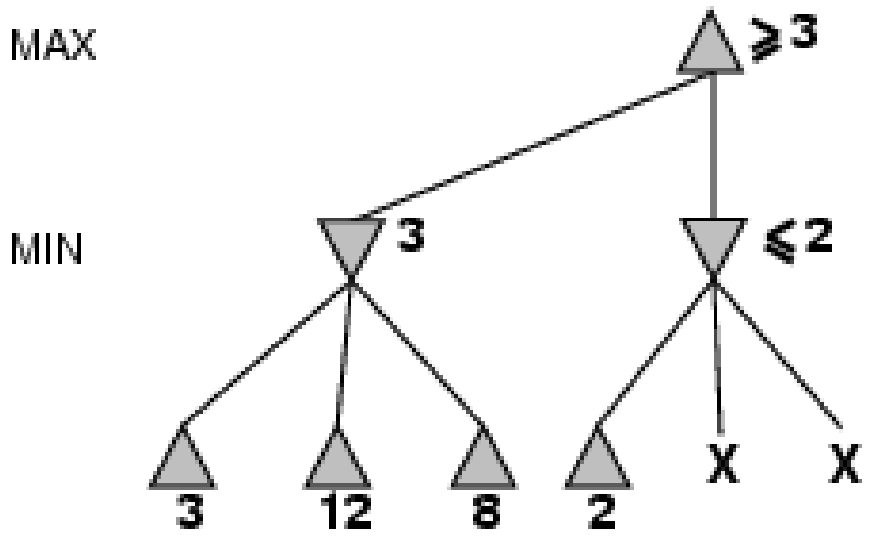


Email: ComputerCollege_Fam@Yahoo.Com

یک نمونه بازی

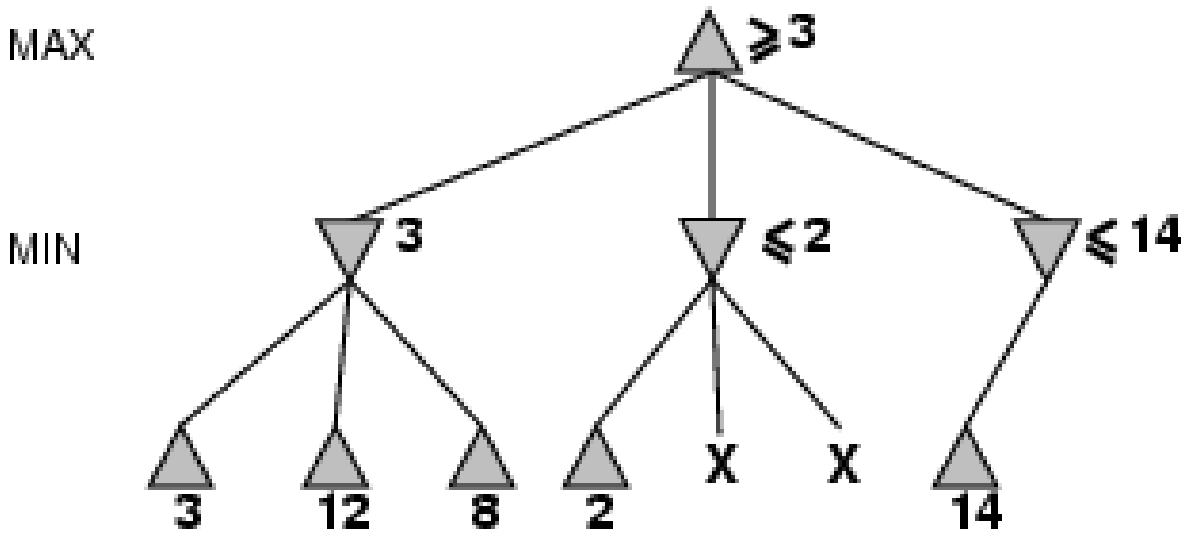


یک نمونه بازی



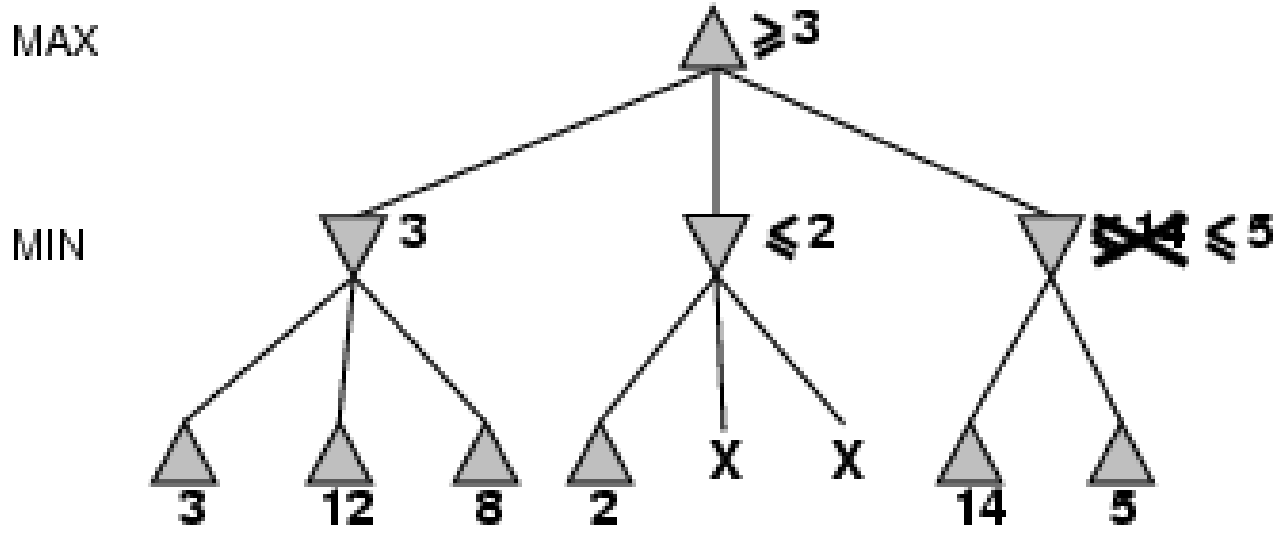
Email: ComputerCollege_Fam@Yahoo.Com

یک نمونه بازی



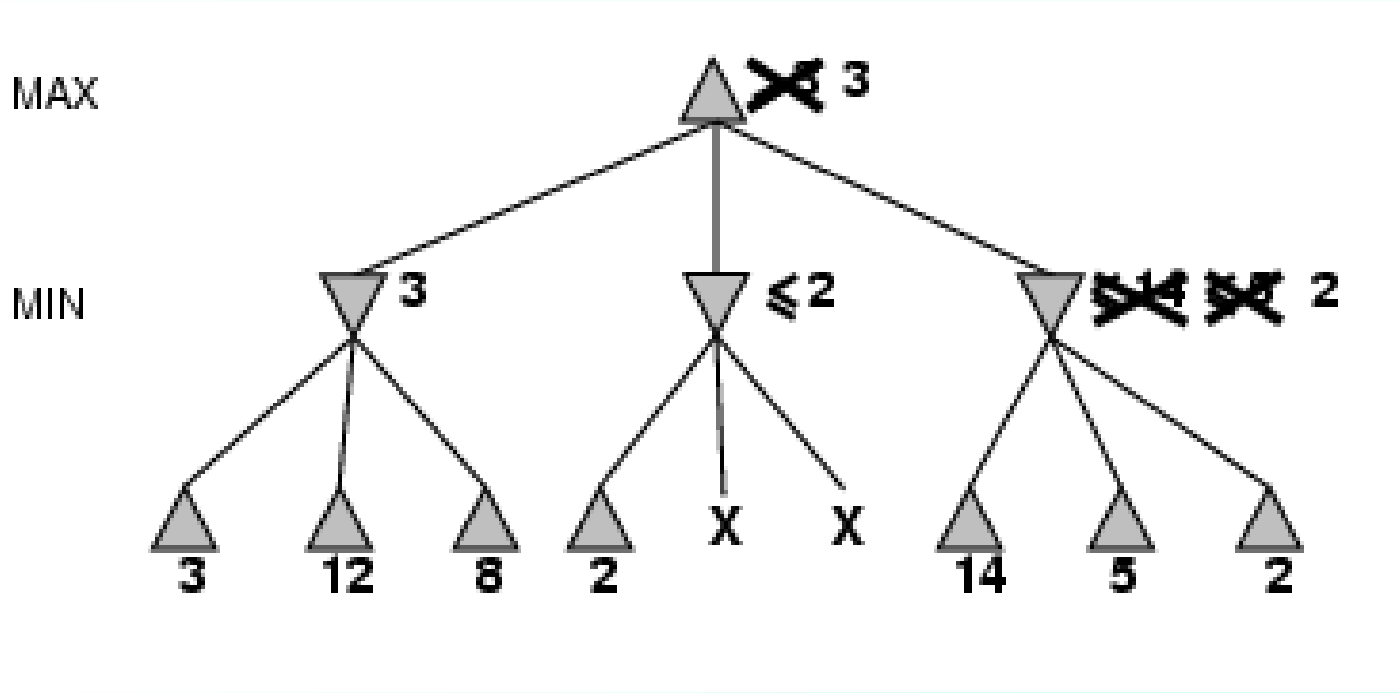
Email: ComputerCollege_Fam@Yahoo.Com

یک نمونه بازی



Email: ComputerCollege_Fam@Yahoo.Com

یک نمونه بازی



الگوریتم minimax

کامل بودن: بله (اگر درخت محدود باشد)

بهینگی: بله

پیچیدگی زمانی: $O(b^m)$

پیچیدگی فضا: $O(bm)$

Email: ComputerCollege_Fam@Yahoo.Com

هرس آلفا-بتا

در الگوریتم MaxMin:

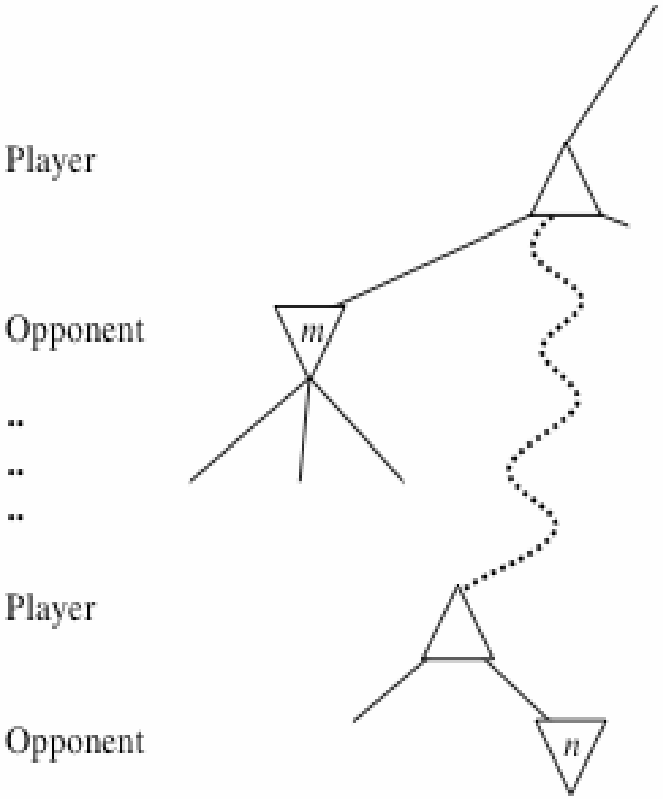
- تعداد حالتهای بازی که باید بررسی شوند، بر حسب تعداد حرکتها، توانی است
  - راه حل: محاسبه تصمیم الگوریتم، بدون دیدن همه گره ها امکانپذیر است

هرس آلفا-بتا:

- انشعاب هایی 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 گره هایی 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

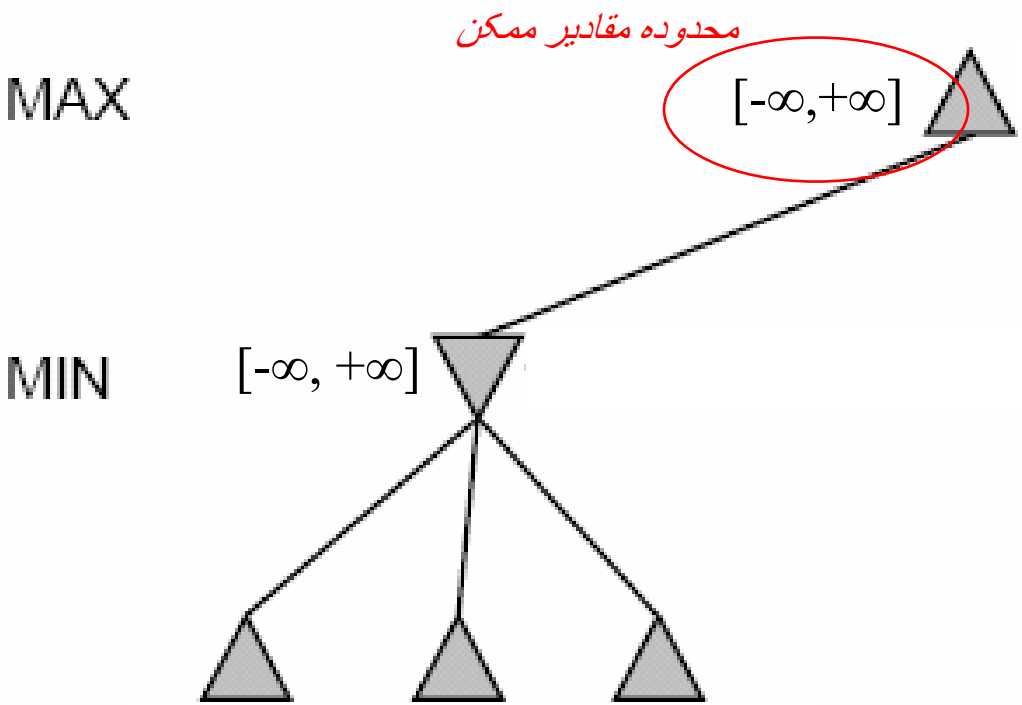
Email: ComputerCollege_Fam@Yahoo.Com

هرس آلفا-بتا

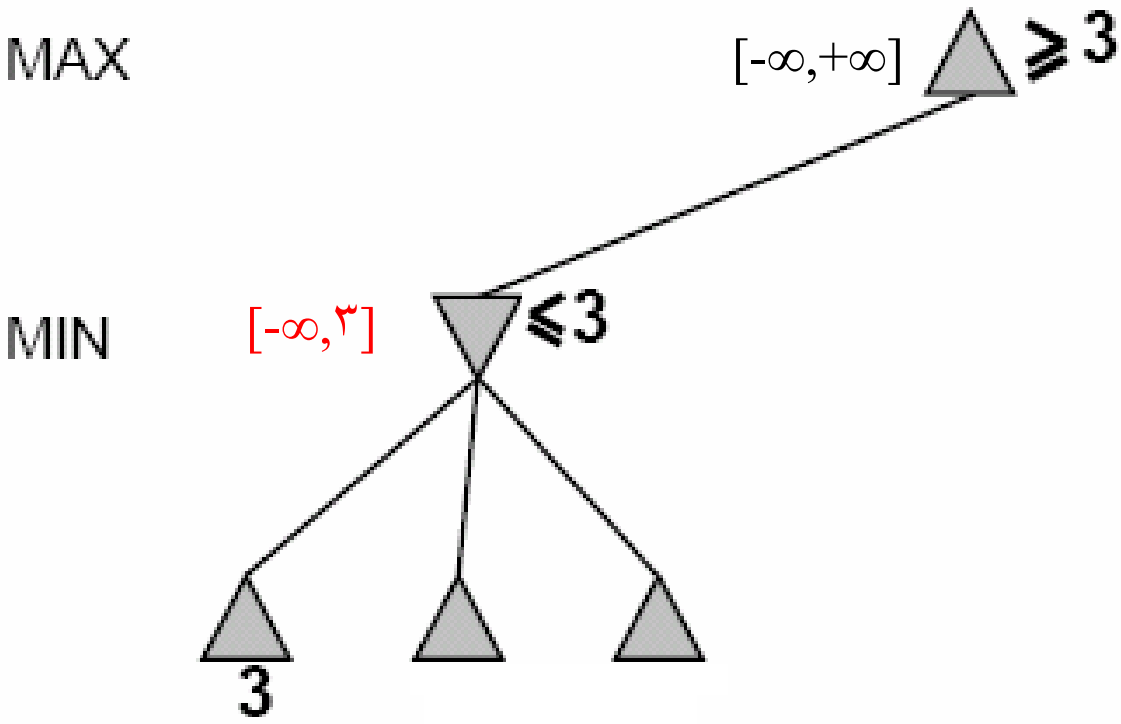


- گره n که هر جای درخت می تواند باشد، بررسی می شود.
- اگر بازیکن انتخاب بهتری داشته باشد
  - در گره والد n
  - یا هر انتخاب بهتری تا کنون
- n هیچوقت در بازی واقعی قابل دسترس نخواهد بود
- در نتیجه n هرس می شود

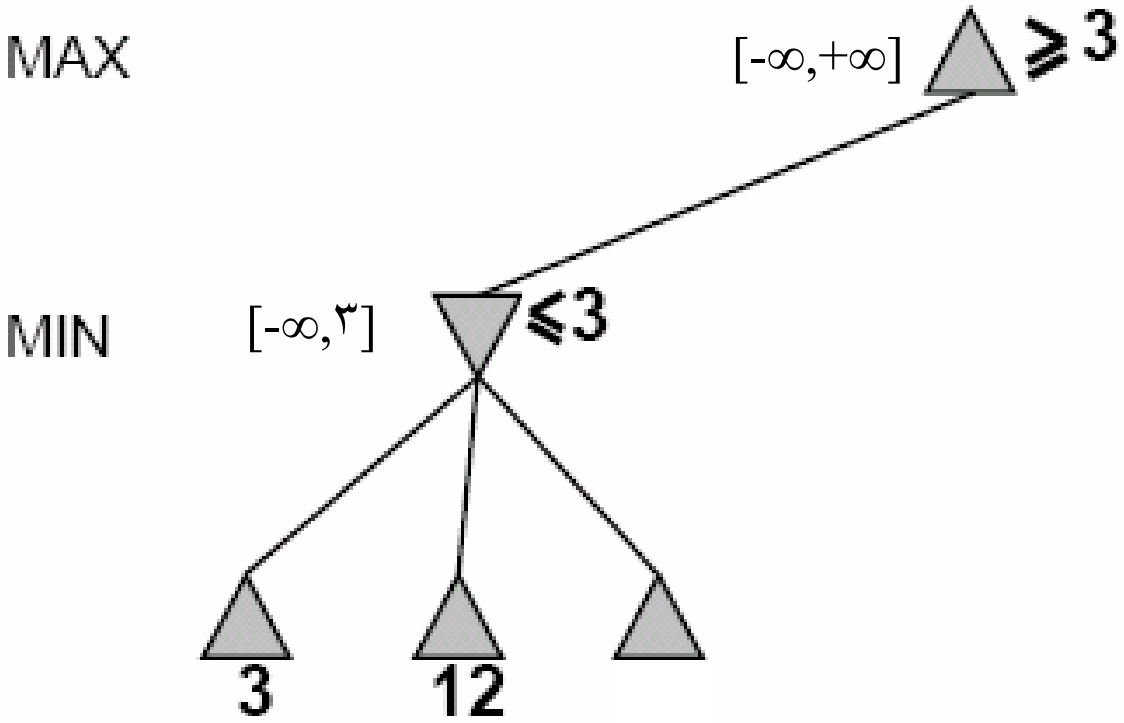
مثال: هرس آلفا-بتا

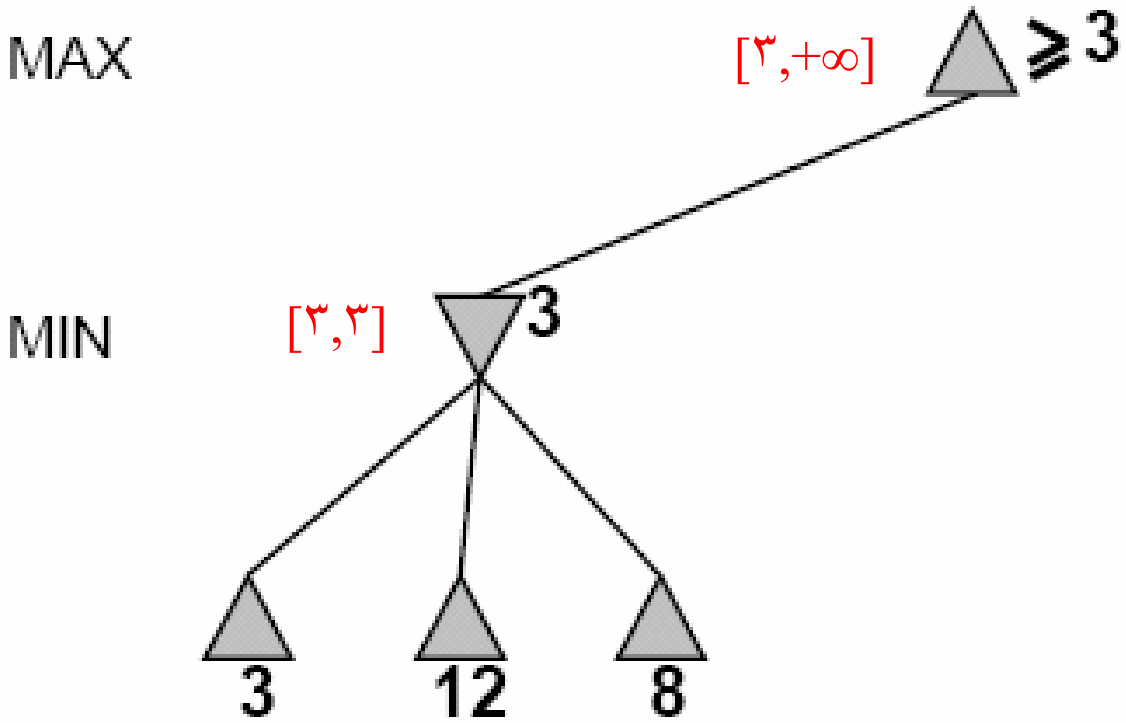


مثال: هرس آلفا-بتا

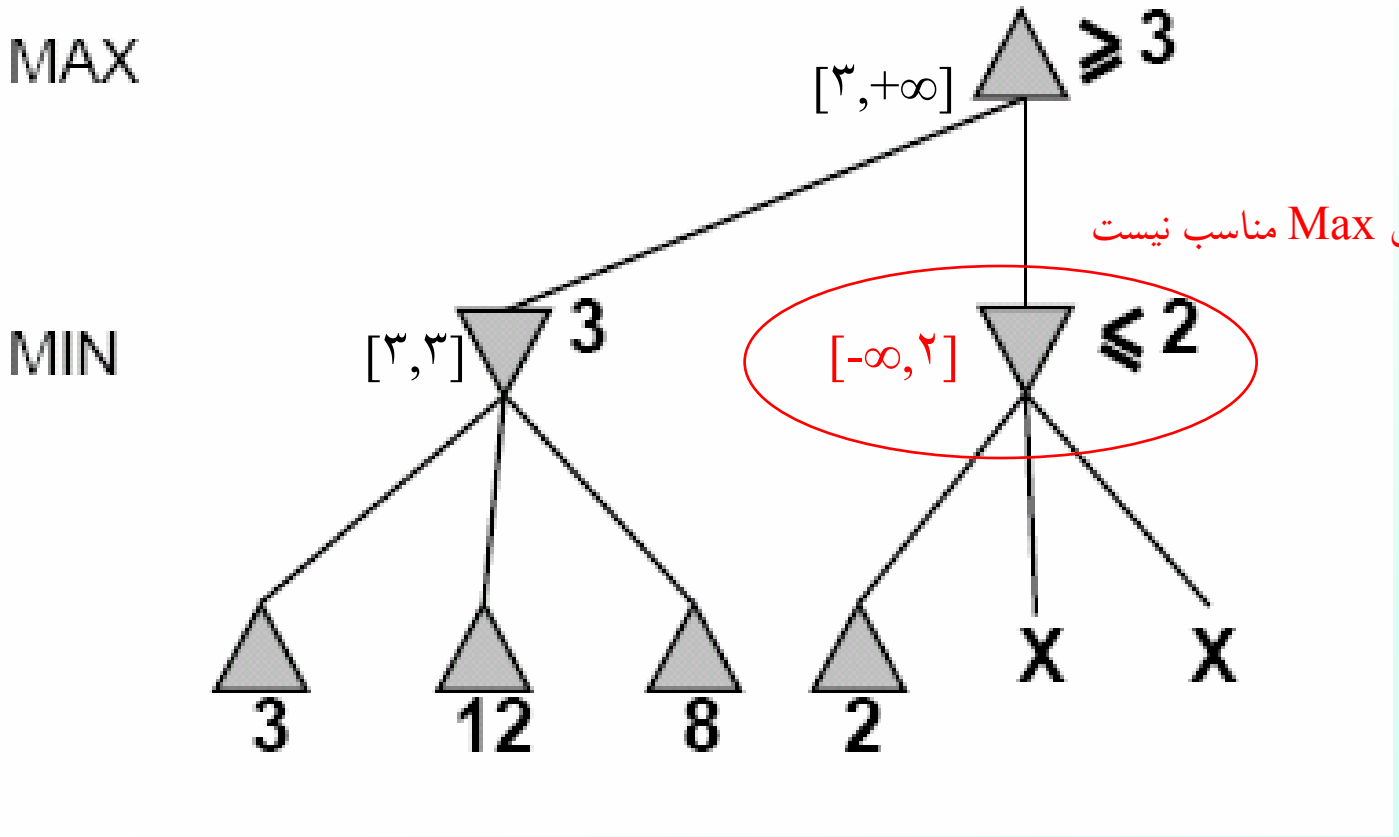


مثال: هرس آلفا-بتا

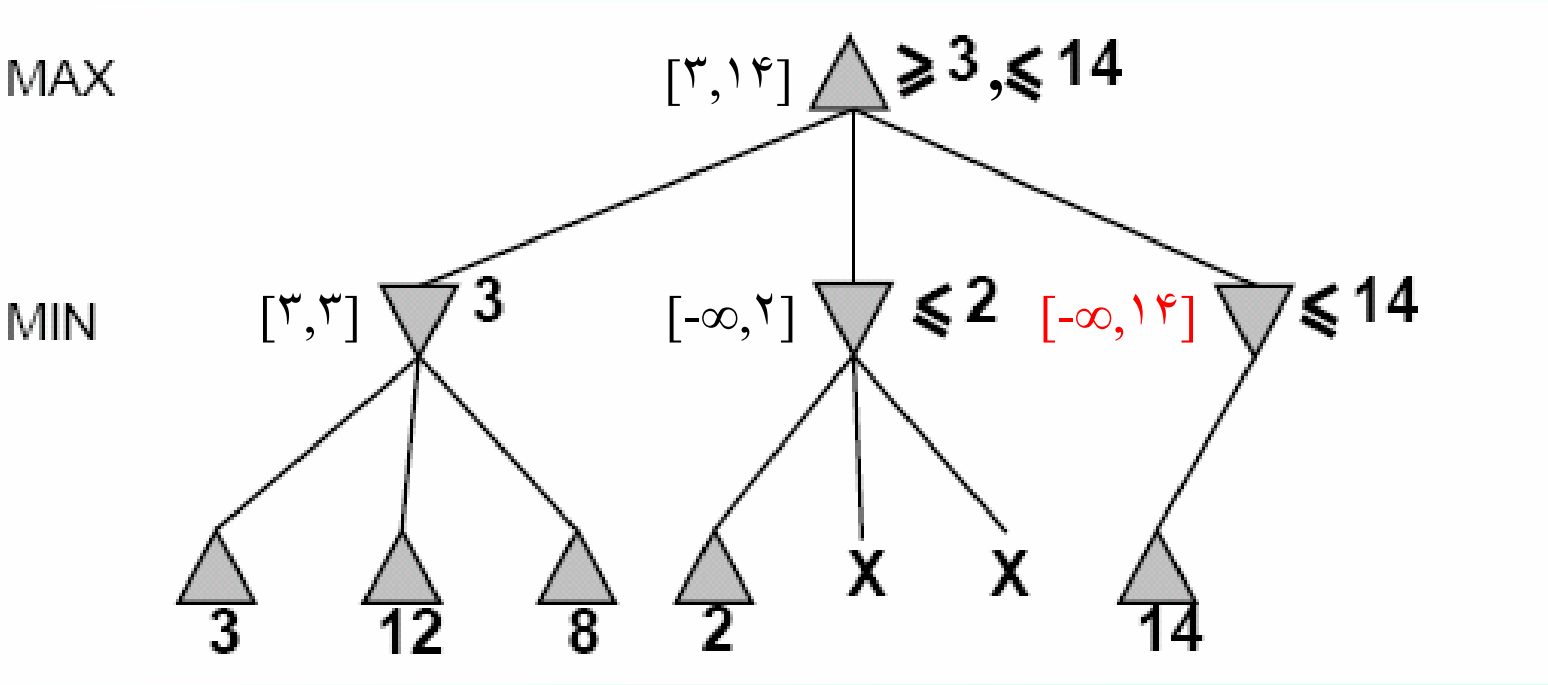



MAX
[۳,+∞]
≥ 3
MIN
[۳,۳]
3
3
12
8

مثال: هرس آلفا–بتا

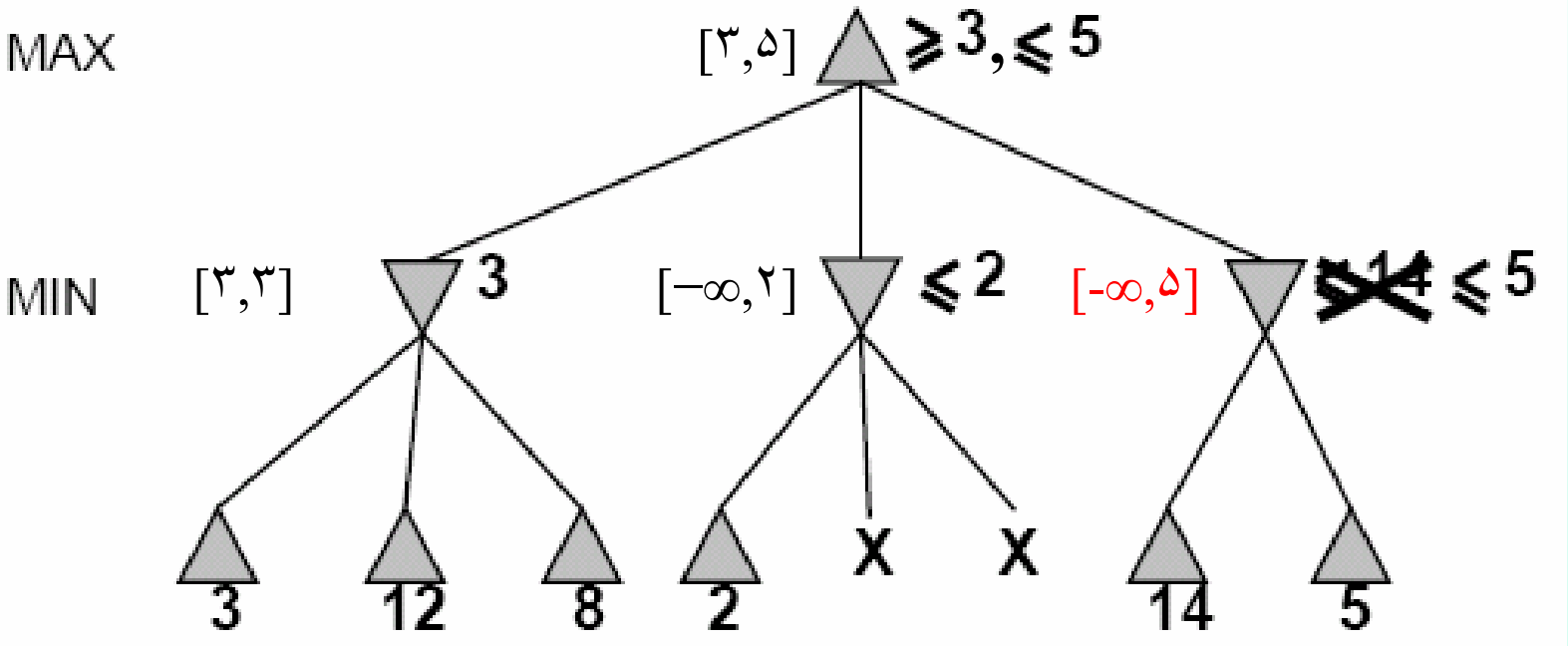


مثال: هرس آلفا-بتا



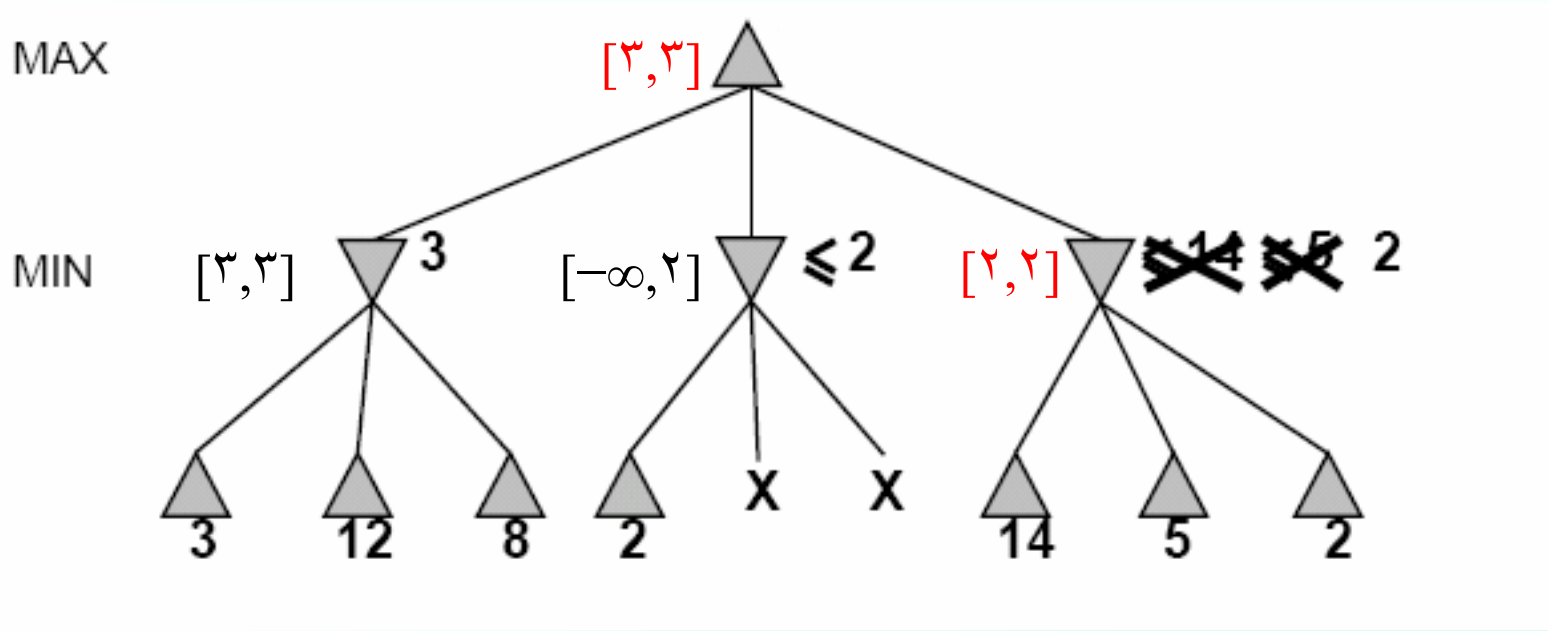
جستجوی خصمانه

مثال: هرس آلفا-بتا

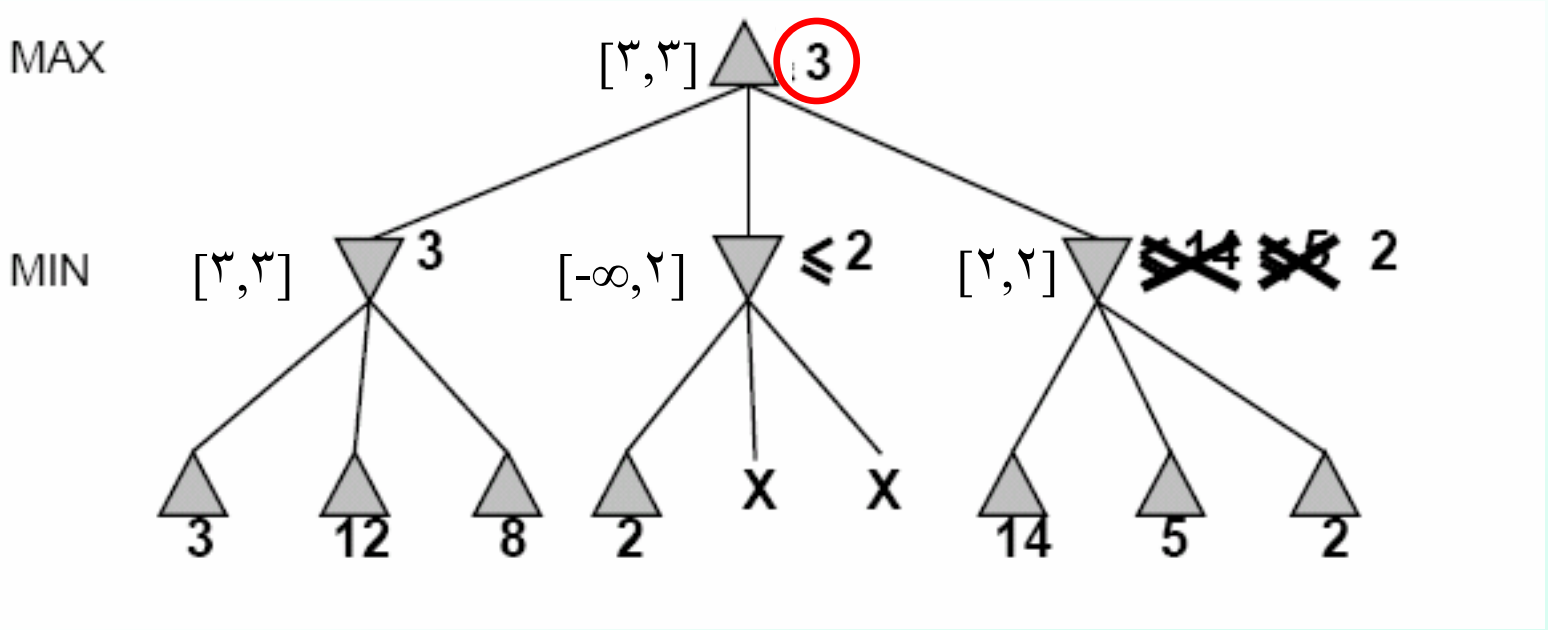


Email: ComputerCollege_Fam@Yahoo.Com

مثال: هرس آلفا–بتا



مثال: هرس آلفا-بتا



بازی های قطعی با اطلاعات ناقص

معایب الگوریتم های پیشین

- الگوریتم minimax کل فضای جست و جوی بازی را تولید می کند
- الگوریتم آلفا-بتا با وجود هرس درخت، اما کل مسیر حالت های 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(۱۹۵۰)

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

بازی های قطعی با اطلاعات ناقص

- در شانون, minimax و آلفا–بتا به دو روش بطور متناوب عمل می کنند
- جایگزینی تابع سودمندی با تابع ارزیابی اکتشافی بنام EVAL
  - تخمینی از سودمندی موقعیت ارائه می کند
- جایگزین تست پایانه با تست توقف
  - تصمیم می گیرد EVAL چه موقع اعمال شود

Email: ComputerCollege_Fam@Yahoo.Com

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 می گرداندند
- اغلب توابع ارزیابی، خواص گوناگونی از حالت ها را محاسبه می کنند
  - خواص روی هم رفته، کلاس های هم ارزی یا دسته های مختلفی از حالت ها را تعریف می کنند
  - حالت های 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 نمی داند کدام حالت منجر به چه چیزی می شود، اما می تواند مقداری برگرداند که تناسب حالت ها را با هر نتیجه نشان دهد

Email: ComputerCollege_Fam@Yahoo.Com

مثال: تابع EVAL

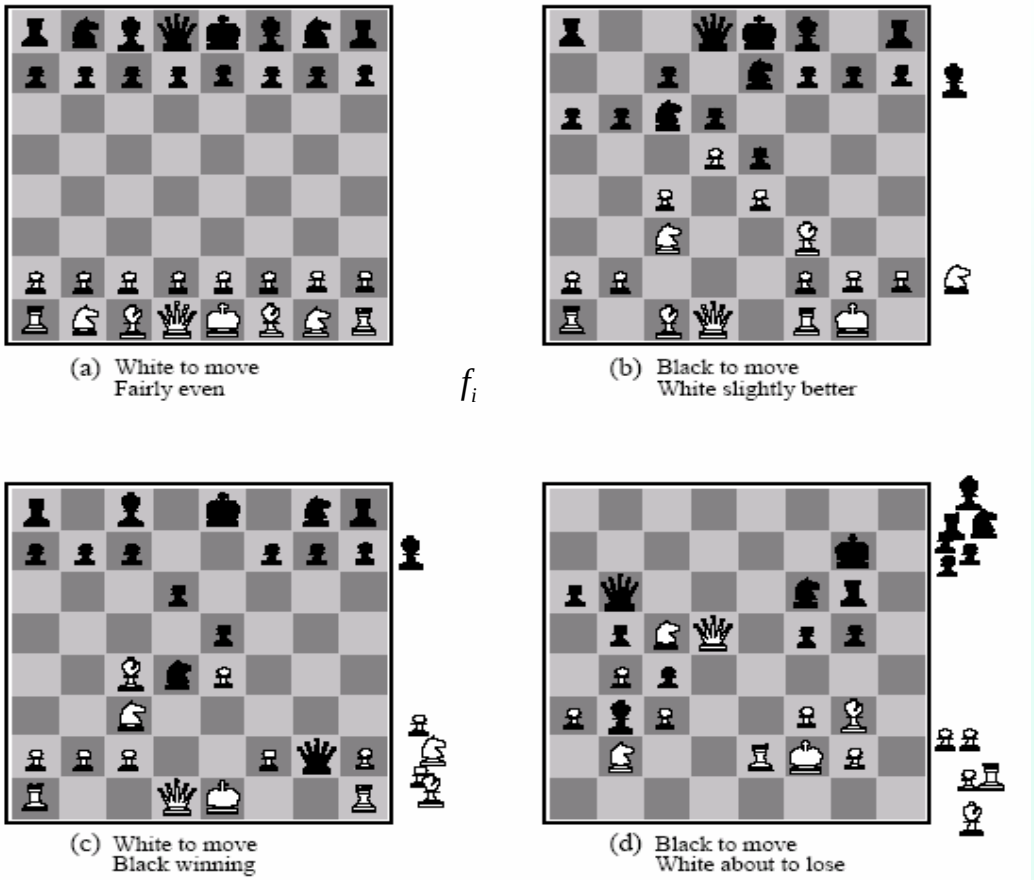
اغلب توابع ارزیابی, مقدار عددی جداگانـه ای برای هر خاصیت محاسبه، سپس آنهـا را ترکیـب میکنند تا مقدار کل بدست آید

مثال در تابع بازی شطرنج:

f_i تعداد هر نوع قطعه در صفحه

w_i مقادیر آن قطعات(۱ برای پیاده، ۳ برای اسب یا فیل،۵ برای رخ و ...)

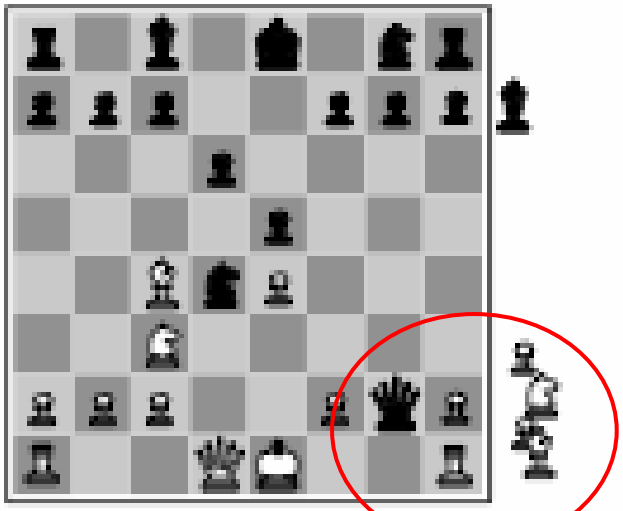
S وضعیت صفحه بازی



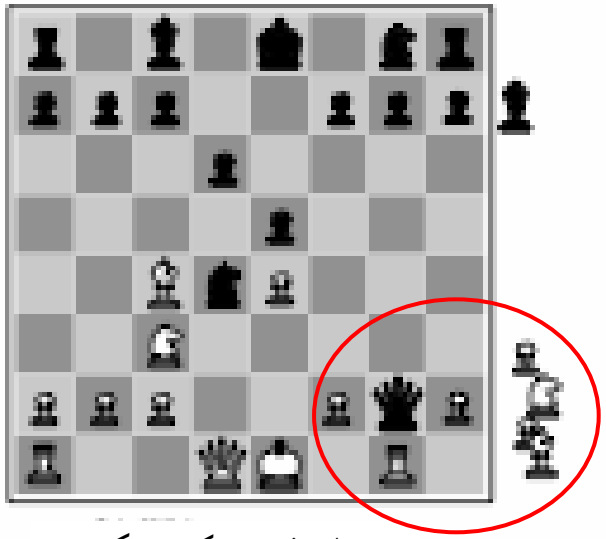
$$Eval(s) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s)$$

مثال: تابع EVAL

ارزیابی تابع EVAL از مقدار پیروزی در دو موقعیت کاملا متفاوت



الف) سفید حرکت میکند



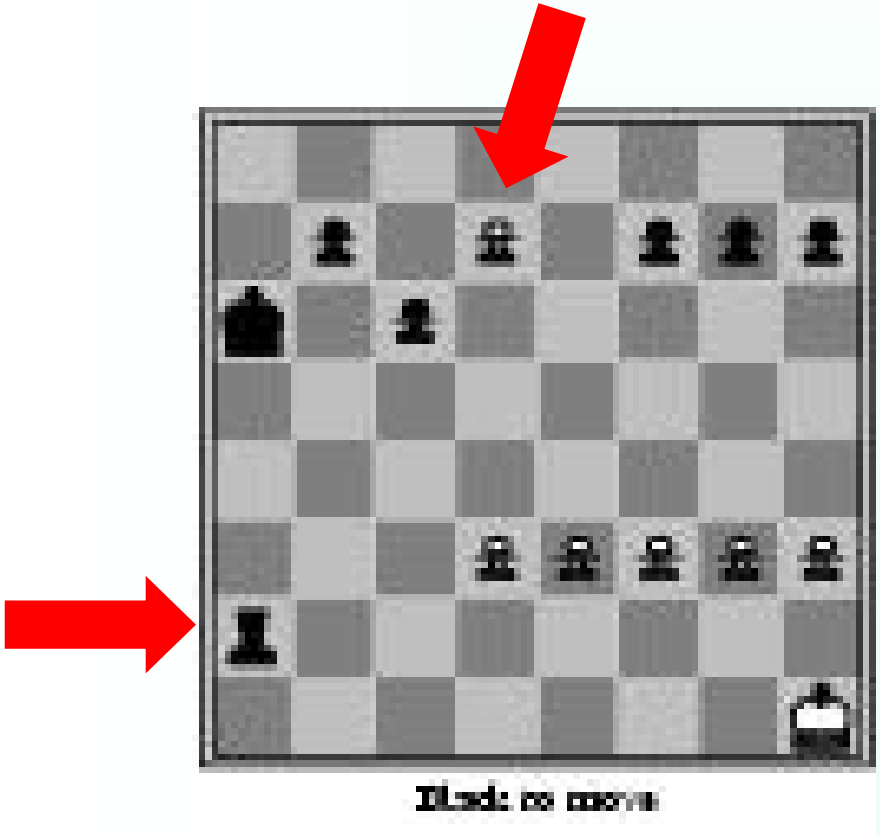
ب) سفید حرکت میکند

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

ب) پس از اینکه سفید، وزیر را در اختیار می گیرد، سیاه می بازد

Email: ComputerCollege_Fam@Yahoo.Com

اثر افق



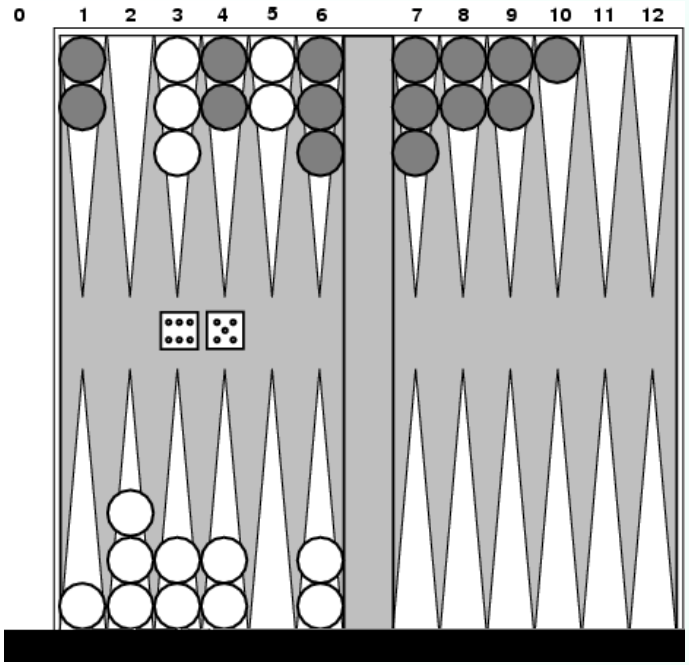
وقتی بوجود می آید که برنامه با اثری از رقیب مواجه شود که منجر به خرابی جدی گشته و اجتناب پذیر است

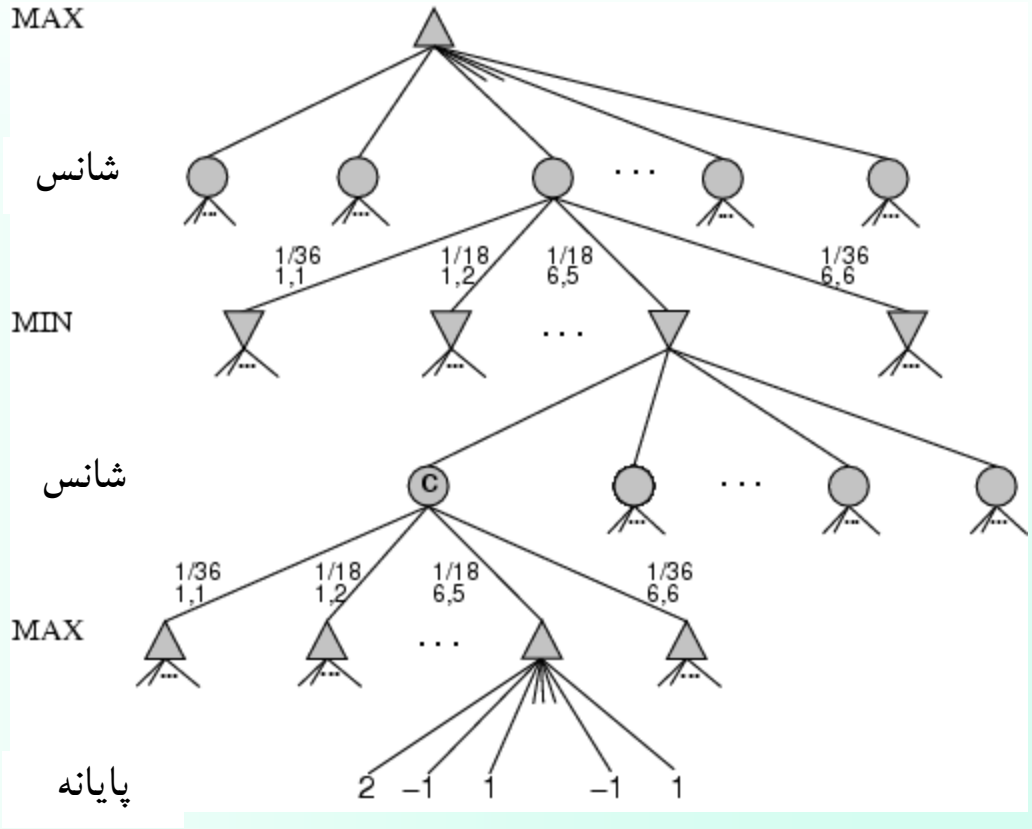
مثال: شکل مقابل؛

سیاه در اصل جلوست، اما اگر سفید پیاده اش را از سطر هفتم به هشتم ببرد، پیاده به وزیر تبدیل می شود و موقعیت برد برای سفید بوجود می آید

Email: ComputerCollege_Fam@Yahoo.Com

بازی هایی که حاوی عنصر شانس هستند

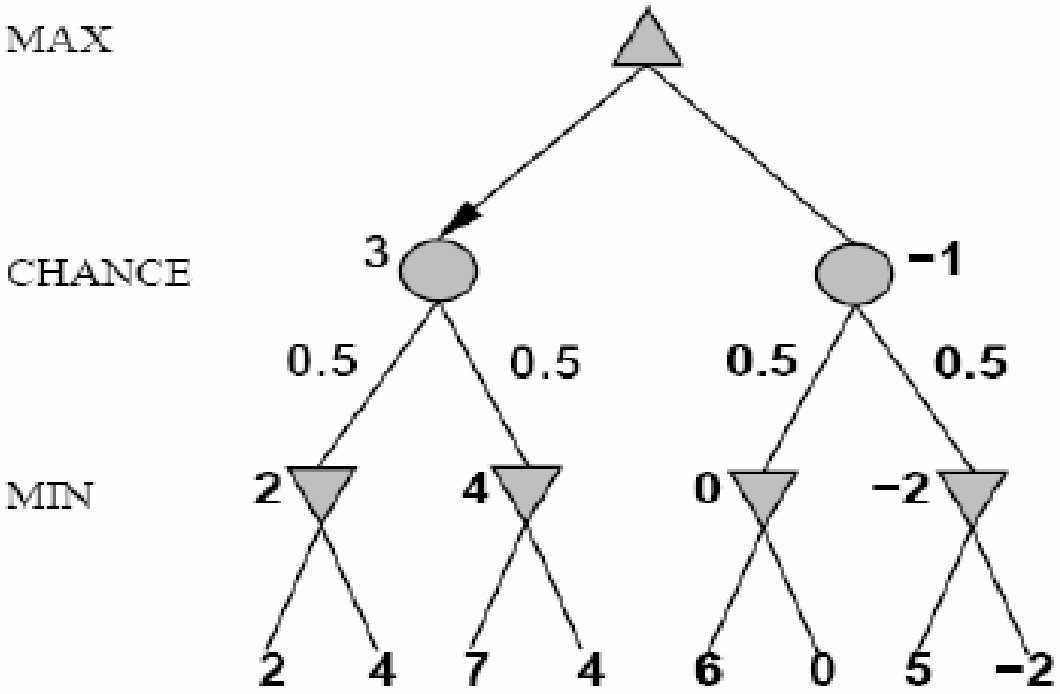




Email: ComputerCollege_Fam@Yahoo.Com

بازی هایی که حاوی عنصر شانس هستند

مثال ساده شده با سکه ی دورویه :



هوش مصنوعی

فصل هفتم

عامل های منطقی

Email: ComputerCollege_Fam@Yahoo.Com

فهرست

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

انواع منطق

- منطق گزاره ای
- منطق مرتبه اول
- منطق مرتبه دوم
- منطق مرتبه های بالاتر

منطق یک نوع زبان است که از دو بخش مهم گرامر و معنا تشکیل شده است.

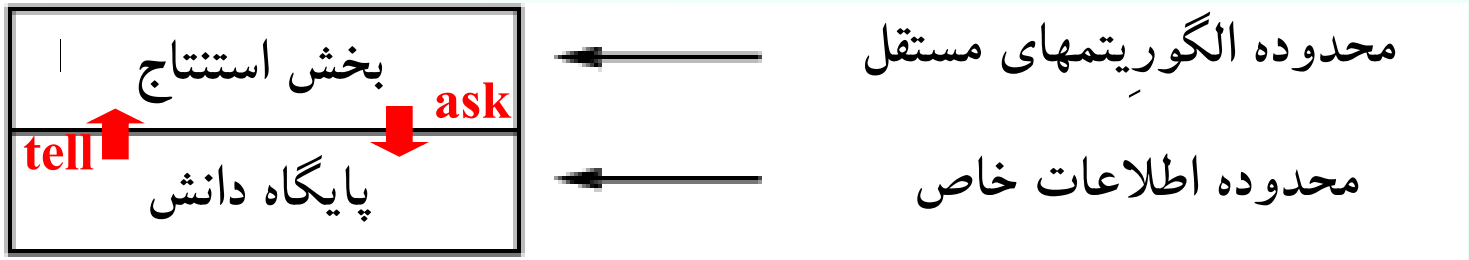
- گزاره چیست؟ (یک جمله)
- باران می بارد
  - یک گزاره است که می توان آنرا به r نشان داد.
- هوا آفتابی است
  - یک گزاره است که می توان آنرا به s نشان داد.
- نمادهای به کار گرفته در منطق گزاره ای
  - ثابت ها: true,false
  - نمادهای گزاره ای: r,s
  - رابطهای منطقی: ¬ Λ V→ ↔
  - پرانتزها

- Sentence→Atomic Sentence| Complex Sentence
- Atomic Sentence→True | False | P | Q|...
- Complex Sentence→(Sentence) | Sentence Connective Sentence | ¬ Sentence
- Connective→ Λ | V|$\rightarrow$| $\leftrightarrow$

Email: ComputerCollege_Fam@Yahoo.Com

عامل های 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



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

عامل های مبتنی بر دانش

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

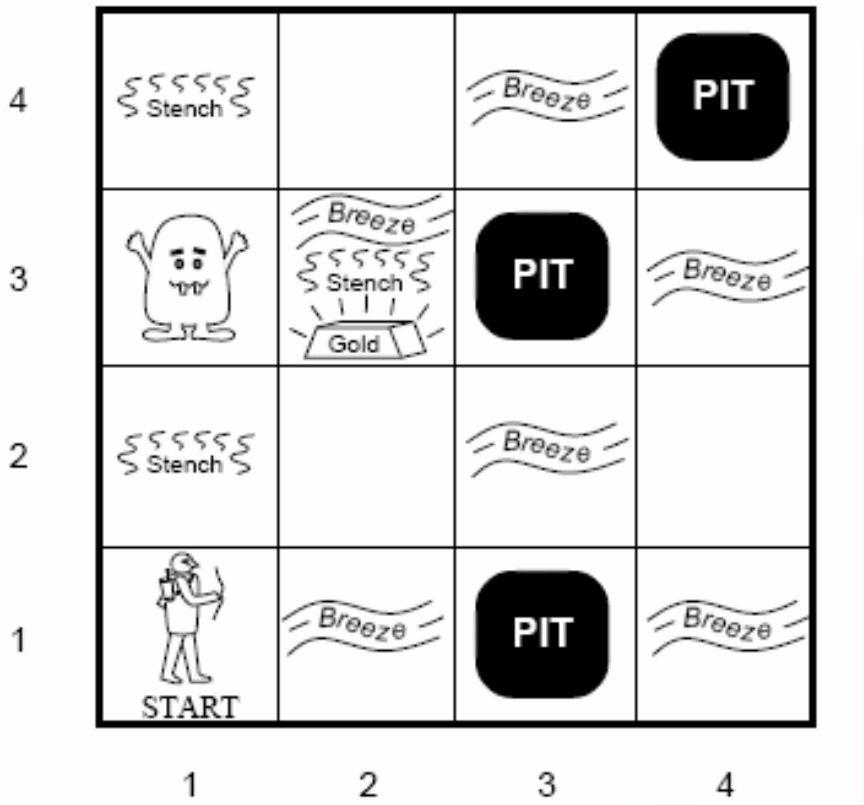
Email: ComputerCollege_Fam@Yahoo.Com

دنیای Wumpus

Wumpus ، نام یک بازی کامپیوتری است ؛ در این بازی ، عامل یک غار را که از اتاق های متصل شده توسط راهرو تشکیل شده کشف می کند . کمین گاه ، جایی است که در آن Wumpus قرار دارد و Wumpus ، جانوری است که هر عاملی را که به اتاقش وارد شود می خورد . همچنین ، برخی از اتاق ها دارای چاله های بدون کف هستند که هر عاملی را که در اتاق سرگردان است را به تله می اندازند . گاهی از وقت ها کپه ای از طلا هم در اتاق هست . **هدف ما در بازی این است که طلا ها را جمع کنیم و بدون اینکه خورده شویم خارج شویم .**

Email: ComputerCollege_Fam@Yahoo.Com

جهان WUMPUS



- معیار کارایی:
  - ۱۰۰۰+ انتخاب طلا، ۱۰۰۰- افتادن در گودال یا خورده شدن، ۱- هر مرحله، ۱۰- برای استفاده از تیر
- محیط:
  - بوی تعفن در مربع های همجوار WUMPUS
  - نسیم در مربع های همجوار گودال
  - درخشش در مربع حاوی طلا
  - کشته شدن WUMPUS با شلیک در صورت مقابله
  - تیر فقط مستقیم عمل می کند
  - برداشتن و انداختن طلا
- حسگرها:
  - بوی تعفن، نسیم، تابش، ضربه، جیغ زدن
- محرک ها:
  - 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 دقیقا مشخص است

رویدادی: خیر، ترتیبی از فعالیت هاست

ایستا: بله, WUMPUS و گودال ها حرکت ندارند

گسسته: بله

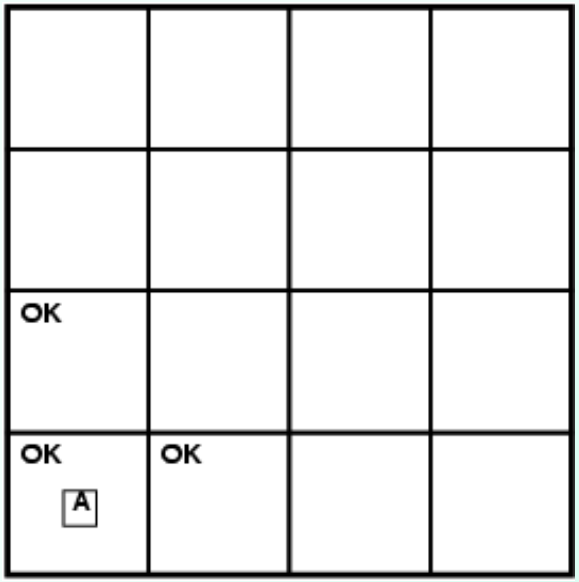
تک عامله: بله، WUMPUS در اصل یک خصوصیت طبیعی است

Email: ComputerCollege_Fam@Yahoo.Com

کاوش در جهان WUMPUS

- با توجه به اینکه هیچ بوی بد یا نسیمی در [۱,۱] وجود ندارد، عامل نتیجه می گیرد [۱,۲] و [۲,۱] امن هستند. خانه های امن را با ok نشان می دهد.

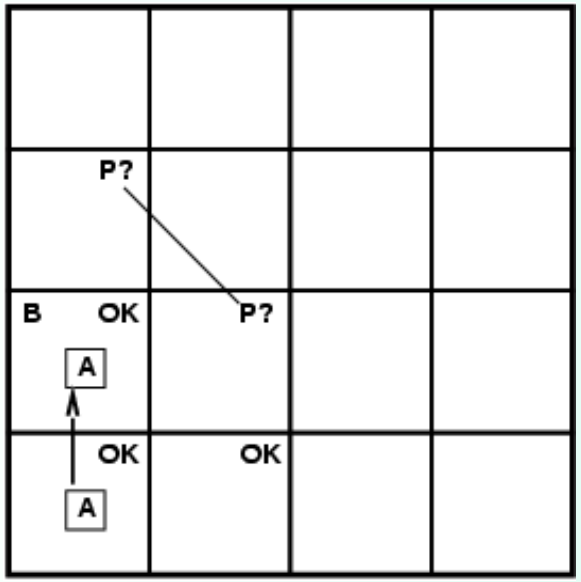
A = عامل
B = نسیم
G = درخشش،طلا
OK = مربع امن
P = گودال
S = تعفن
V = ملاقات شده
W = Wumpus

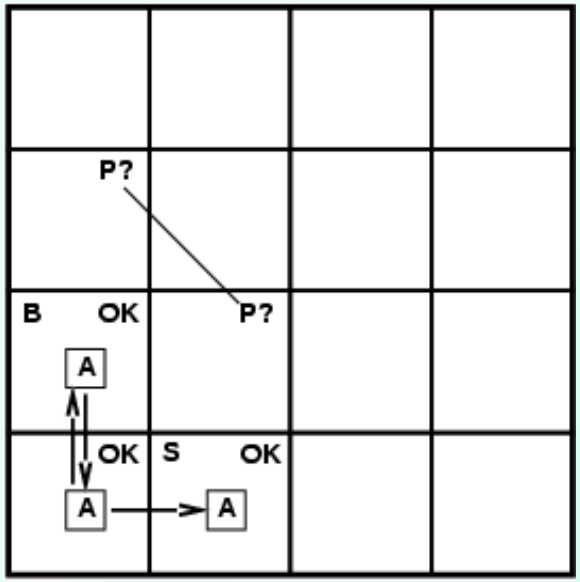


کاوش در جهان WUMPUS

برود. در این موقعیت عامل نسیم را احساس می کند. بنابراین [۲,۱]فرض کنید عامل به خانه باشد بنابراین یا در [۱,۱]گودالی باید در یکی از مربع های مجاور باشد. گودال نمی تواند در رفته و از [۱,۱] یا در هر دو گودال وجود دارد.بنابراین عامل مجددا به [۳,۱] یا در [۲,۲] می رود.[۱,۲]آنجا به

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

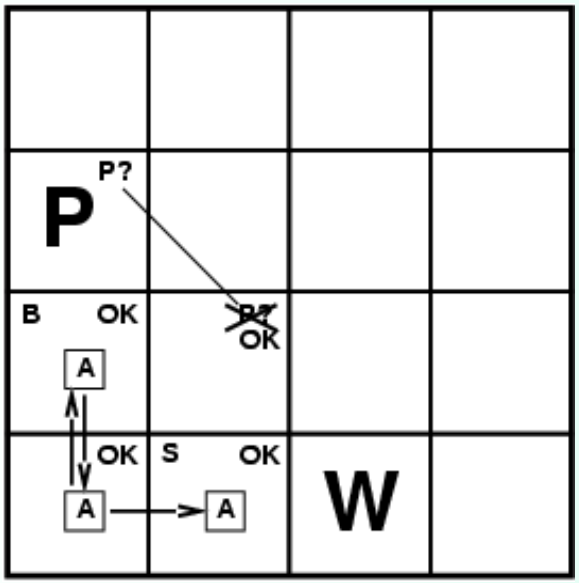


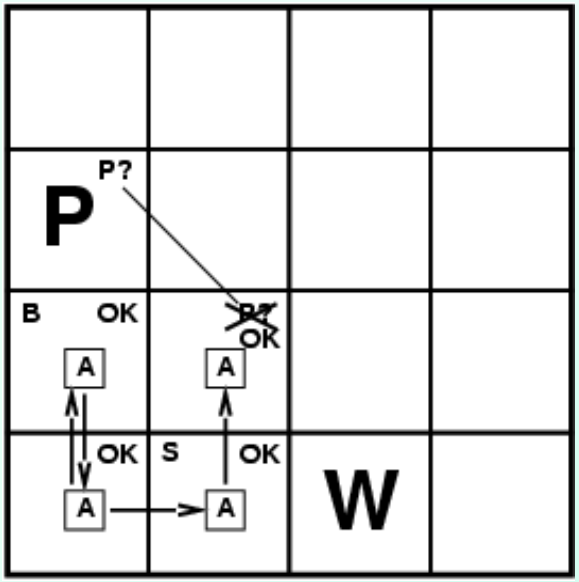


کاوش در جهان WUMPUS

•در [۱,۲] عامل بوی بد را دریافت می کند و متوجه می شود wampus در یکی از همسایه ها است. W در [۱,۱] نمی باشد در [۲,۲] نیز نمی تواند باشد زیرا هنگامی که عامل در [۲,۱] بود بوی بدی حس نکرد. لذا نتیجه می گیرد که W در [۱,۳] قرار دارد. همچنین با توجه به اینکه نسیمی احساس نکرد نتیجه می گیرد که در [۲,۲] گودال وجود ندارد لذا گودال در [۳,۱] است.

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





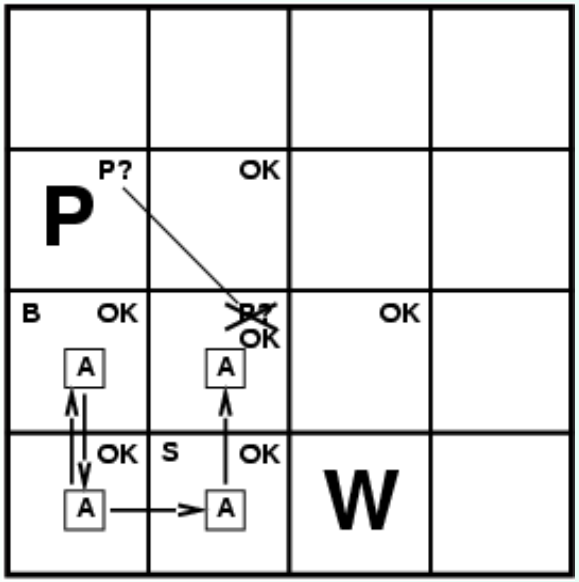
Email: ComputerCollege_Fam@Yahoo.Com

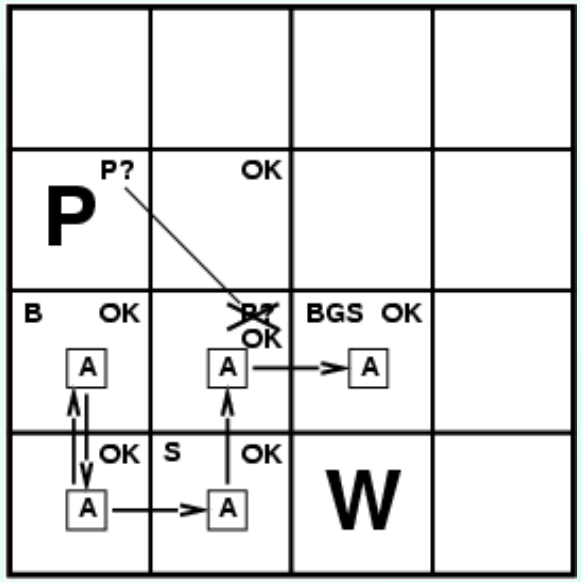
کاوش در جهان WUMPUS

می رود و با توجه به اینکه در این خانه بو یا نسیم دریافت نمی کند نتیجه می [۲,۲]عامل به گیرد که همسایه ها امن هستند.

برود و در آنجا بوی بد، نسیم و تشعشع را حس کند پس طلا [۲,۳]فرض کنید عامل از آنجا به در همان خانه است.

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 می کند
- مثال، در زبان ریاضیات
  - $X+۲ >= y$ یک جمله اما $x۲+y$ جمله نیست
  - $X+۲ >= y$ در جهان درست است اگر $x=۷$ و $y=۱$
  - $X+۲ >= y$ در جهان غلط است اگر $x=۰$ و $y=۶$

Email: ComputerCollege_Fam@Yahoo.Com

استلزام

- استلزام منطقی بین جملات این است که جمله ای بطور منطقی از جمله دیگر پیروی می کند

$$a \models b$$

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

- مثال: جمله $x+y=۴$ مستلزم جمله $x+y=۴$ است

منطق گزاره ای

- نحو منطق گزاره ای، جملات مجاز را تعریف می کند
- 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 می کنند
- جملات پیچیده با استفاده از رابط های منطقی، از جملات ساده تر ساخته می شوند:
- ¬ (not) جمله ای مثل $W_{1,3}$ ¬ $W_{1,3}$ نقیض $W_{1,3}$ است
  - لیترال یک جمله اتمیک(لیترال مثبت)، یا یک جمله اتمیک منفی(لیترال منفی) است
  - ^ (and) مثل $W_{1,3}$ ^ $P_{1,3}$ ترکیب عطفی نام دارد.هر بخش آن یک عطف نامیده میشود
  - v (or) مثل $W_{2,2}$ v ($W_{1,3}$ ^ $P_{3,1}$) ترکیب فصلی مربوط به فصل های $W_{2,2}$ و $W_{1,3}$ ^ $P_{3,1}$
  - => (استلزام): $W_{2,2}$ ¬ v ($W_{1,3}$ ^ $P_{3,1}$) استلزام یا شرطی نامیده میشود. مقدمه یا مقدم آن $W_{1,3}$ و نتیجه یا تالی آن $W_{2,2}$ ¬ است
  - ⇔ جمله $W_{2,2}$ ⇔ $W_{1,3}$ دو شرطی نام دارد

Email: ComputerCollege_Fam@Yahoo.Com

منطق گزاره ای

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

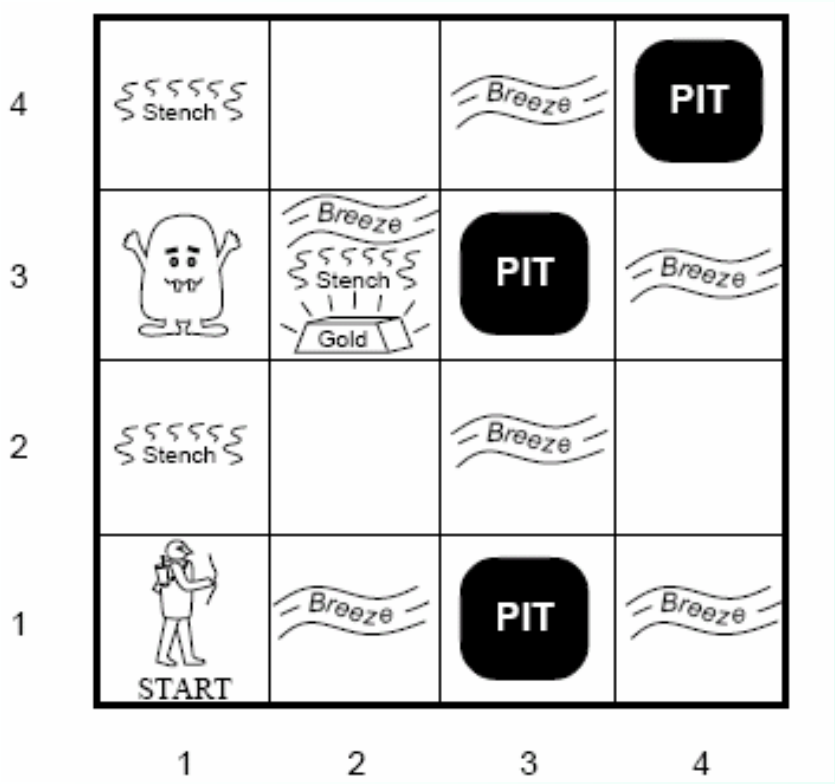
Email: ComputerCollege_Fam@Yahoo.Com

جدول درستی پنج رابطه منطقی

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

Email: ComputerCollege_Fam@Yahoo.Com

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

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

Email: ComputerCollege_Fam@Yahoo.Com

الگوهای استدلال در منطق گزاره ای

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

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

- می توان از

(WumpusAhead ^ WumpusAlive)

و

(WumpusAhead ^ WumpusAlive) => Shoot

Shoot را استنتاج کرد

Email: ComputerCollege_Fam@Yahoo.Com

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

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

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

خاصیت یکنواختی

مجموعه ای از جملات استلزامی که فقط می تواند 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 می کند

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

قانون resulotion واحد می تواند به قانون resulotion کامل تعمیم داد:

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

Email: ComputerCollege_Fam@Yahoo.Com

الگوریتم Resolution

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

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

- الگوریتم resolution:
  - برای اینکه نشان دهیم KB|=a, مشخص میکنیم (KB ^ ¬ a) ارضا کننده نیست
  - ابتدا (KB ^ ¬ a) را به CNF تبدیل میکنیم
  - سپس قانون resulotion به عبارات کوچک حاصل اعمال میشود
  - هر جفتی که شامل لیترال های مکمل باشد، resulotion می شود تا عبارت جدیدی ایجاد گردد
  - اگر این عبارت قبلا در مجموعه نباشد، به آن اضافه می شود
  - 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

Email: ComputerCollege_Fam@Yahoo.Com

صورت نرمال ربط دهنده [1] (CNF)

اغلب ، مفید است که فرمول ها را به صورت های نرمال بنویسیم . صورت های نرمال با آنچه که قبل از نرمال کردن بودند فرقی ندارند و تفاوت آن ها در ظاهر آن هاست و برای استدلال ، مناسب ترند . یک حرف [2] ، یک چیز تجزیه ناپذیر [3] یا نقیض آن است مثل ، P یا نقیض آن که $\neg P$ است . یک فرمول به صورت CNF است اگر به صورت $A_1 \wedge A_2 \wedge \ldots A_k$ باشد که A_i تشکیل شده از یا (OR) گزاره ها یا نقیض آن ها . به عنوان مثال ؛

$(p \vee q) \wedge r \wedge (\neg p \vee \neg r \vee s)$ ، به صورت CNF است .

$\neg(p \vee q) \wedge r \wedge (\neg p \vee \neg r \vee s)$ ، به صورت CNF نمی باشد .

$(p \vee q) \wedge r \wedge (p \Rightarrow (\neg r \vee s))$ ، به صورت CNF نمی باشد .

نکته : *هر عبارت منطق گزاره ای می تواند به فرم نرمال ربط دهنده تبدیل شود .*

Email: ComputerCollege_Fam@Yahoo.Com

مثال :

فرمول نهایی (به صورت CNF)	فرمول اوّلیه
$(A \vee B) \wedge (A \vee C)$	$A \vee (B \wedge C)$
$(B \vee A) \wedge (C \vee A)$	$(B \wedge C) \vee A$

Email: ComputerCollege_Fam@Yahoo.Com

مزیّت های منطق گزاره ای

- اعلانی یا اظهاری بودن این روش – دانش می تواند از استنتاج ، مجزّا باشد .
- می تواند اطّلاعات جزیی را به کار بگیرد .
- می تواند عبارت های پیچیده تر را به صورت ساده تر تولید نماید .
- مکانیزم های صحیح و کامل دارد (برای عبارت های شیپوری ، کارآمد می باشد)

Email: ComputerCollege_Fam@Yahoo.Com

معایب منطق گزاره ای

- افزایش تشریحی در تعداد لفظ ها
- راهی برای تشریح ارتباط های میان اشیا وجود ندارد .
- راهی برای بیان کیفیّت ، در مورد اشیا وجود ندارد .

منطق مرتبه ی اوّل ، روشی برای رسیدگی کردن به این مسایل می باشد .

Email: ComputerCollege_Fam@Yahoo.Com

زنجیر 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 می شوند
  - لیترال مثبت را **رأس** و لیترال های منفی را **بدنه عبارت** گویند
  - عبارت معینی که فاقد لیترال های منفی باشد، گزاره ای بنام **حقیقت** نام دارد
  - **عبارات معین** اساس **برنامه نویسی منطقی** را می سازد
- استنتاج با عبارات هورن، از طریق الگوریتم های **زنجیر پیشرو** و **زنجیر عقبگرد** انجام می گیرد

- برای هر خانه باید جداگانه قوانین و حقایق مربوط به آن نوشته شود نمی توان قوانین کلی نوشت.

مثلا

هر جای گودال باشد در همسایه های آن، نسیم وجود دارد با این منطق قابل نمایش نیست.

فقط در یک خانه wampus وجود دارد با این منطق قابل نمایش نیست.

Email: ComputerCollege_Fam@Yahoo.Com

- اگر بخواهیم جمله ”فقط در یک خانه wampus وجود دارد“ را با منطق گزاره ای نمایش دهیم باید به صورت زیر عمل کنیم.

¬W[۱,۱]	¬W[۱,۲]	W[۱,۳]	¬W[۱,۴]
¬W[۲,۱]	¬W[۲,۲]	¬W[۲,۳]	¬W[۲,۴]
¬W[۳,۱]	¬W[۳,۲]	¬W[۳,۳]	¬W[۳,۴]
¬W[۴,۱]	¬W[۴,۲]	¬W[۴,۳]	¬W[۴,۴]

- مشاهده می شود که باید ۱۶ قانون نوشته شود.

Email: ComputerCollege_Fam@Yahoo.Com

- یا مثلا در مورد گودال ها باید برای تک تک خانه ها قوانین گودال و نسیم را نوشت به صورت زیر

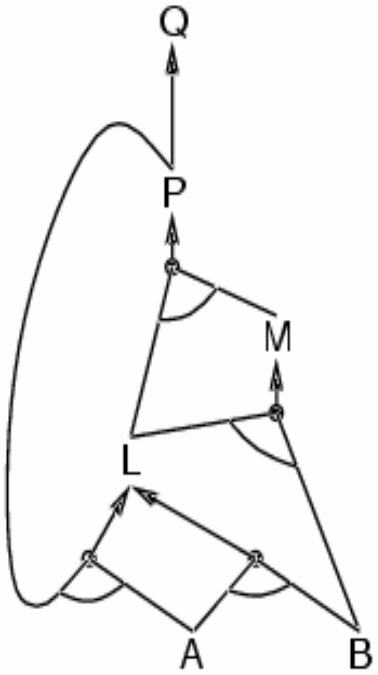
- $B[۱,۱] \leftrightarrow P[۱,۲] \vee P[۲,۱]$
- $B[۲,۱] \leftrightarrow P[۱,۱] \vee P[۲,۲] \vee P[۳,۱]$
- ...

Email: ComputerCollege_Fam@Yahoo.Com

زنجیر پیشرو

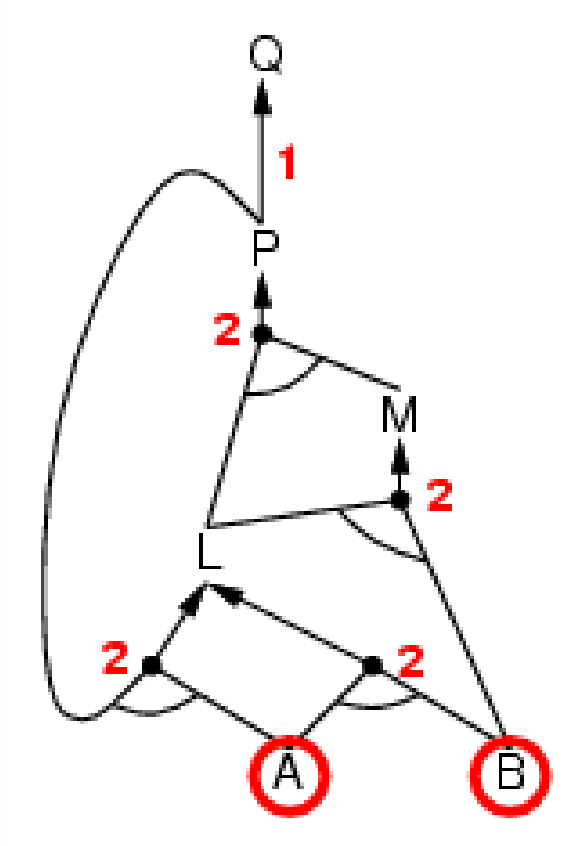
الگوریتم زنجیر پیشرو تعیین می کند آیا نماد گزاره ای q(تقاضا)، توسط پایگاه دانش عبارات هورن ایجاب می شود یا خیر

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

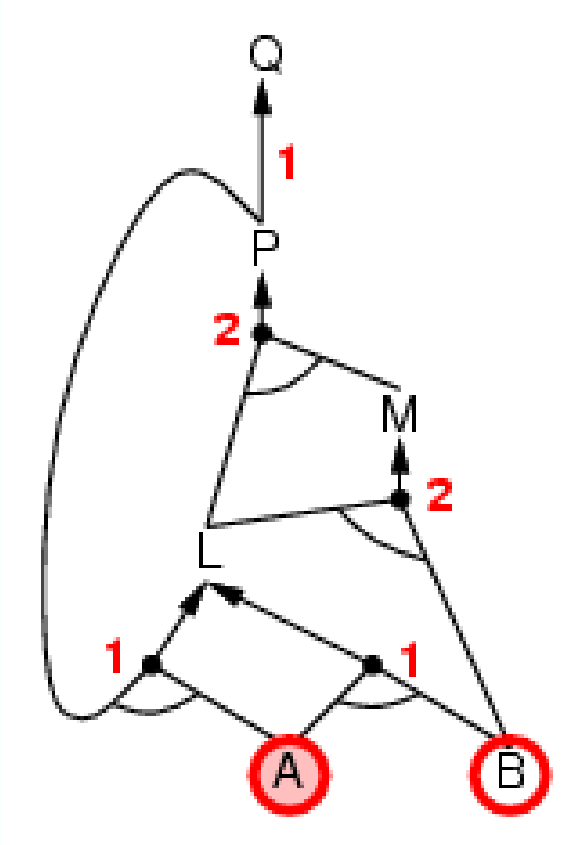


Email: ComputerCollege_Fam@Yahoo.Com

زنجیر پیشرو

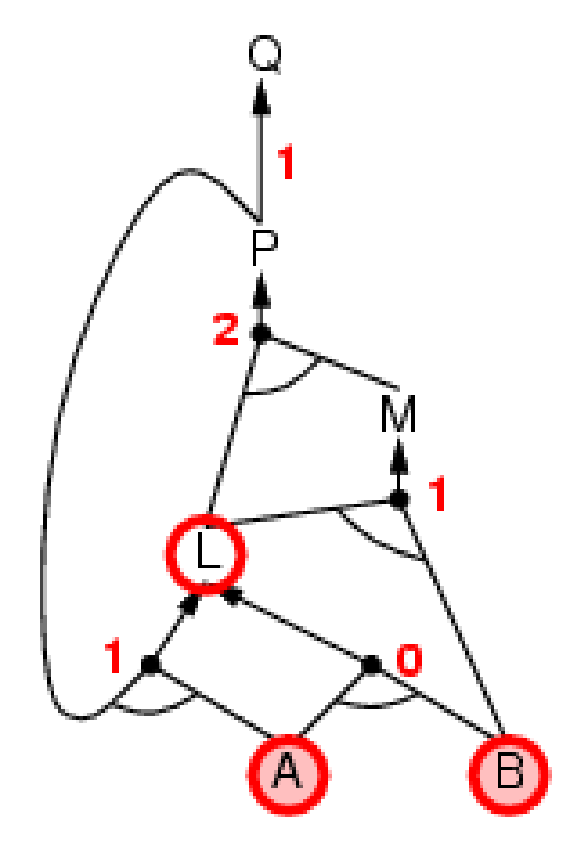


زنجیر پیشرو

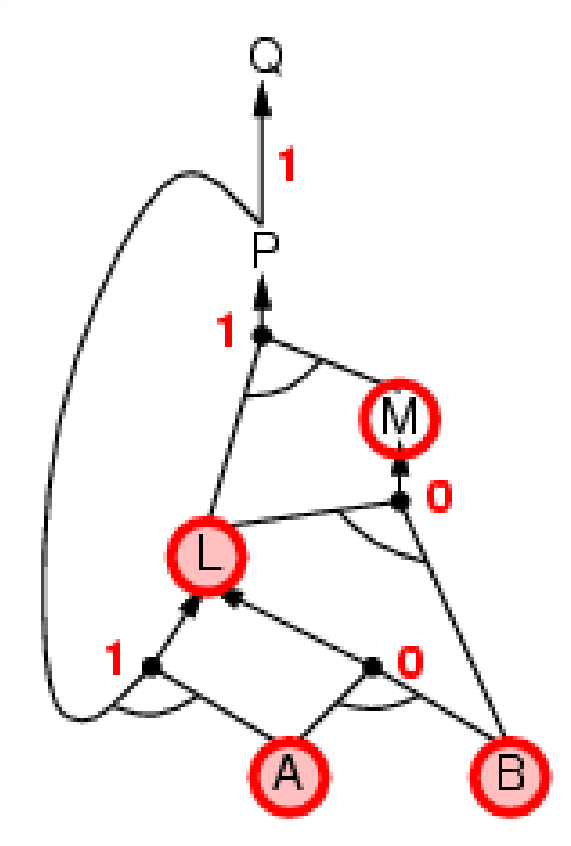


Email: ComputerCollege_Fam@Yahoo.Com

زنجیر پیشرو

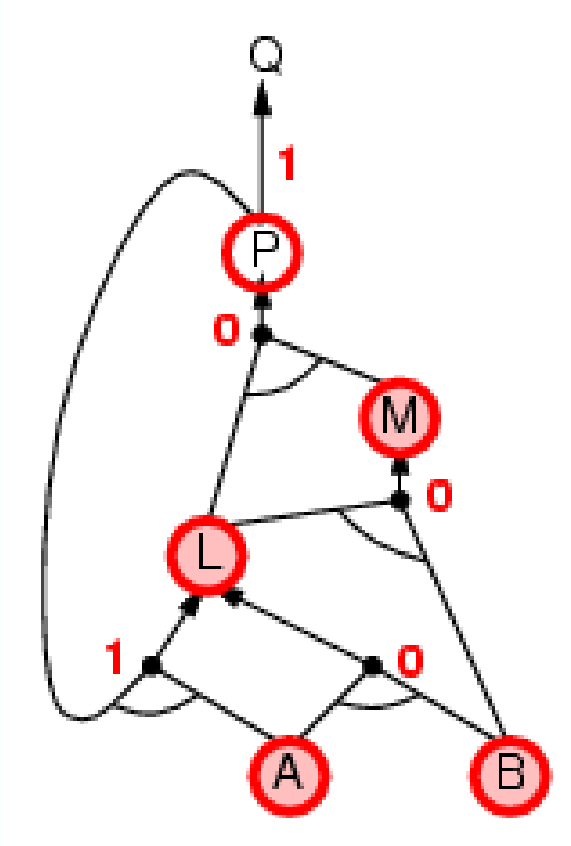


زنجیر پیشرو

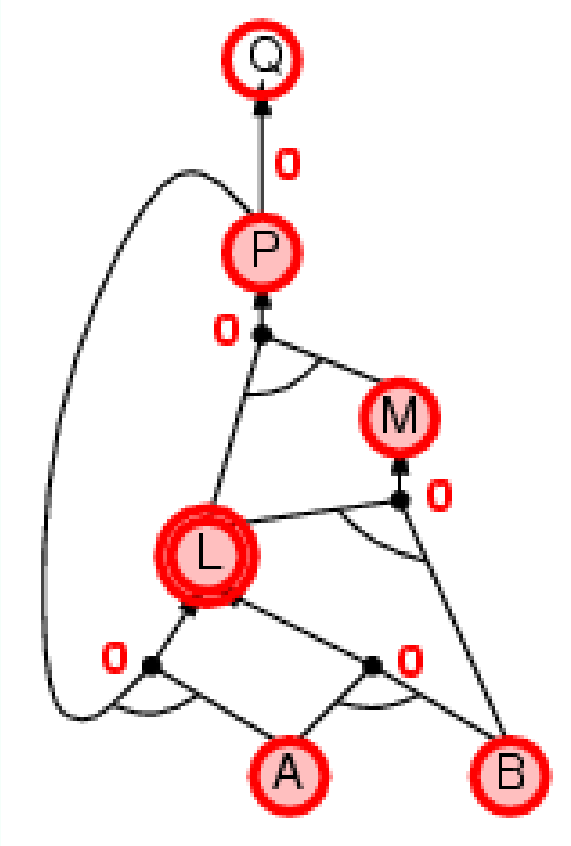


Email: ComputerCollege_Fam@Yahoo.Com

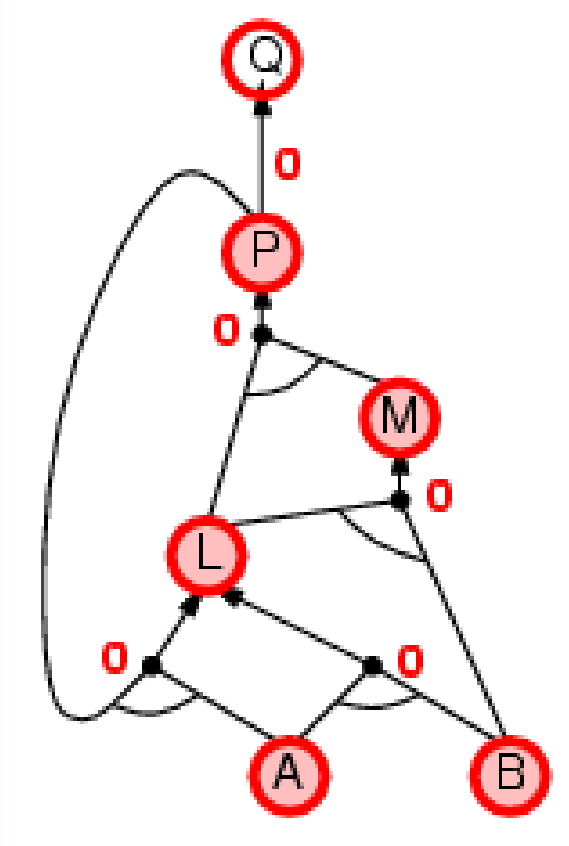
زنجیر پیشرو



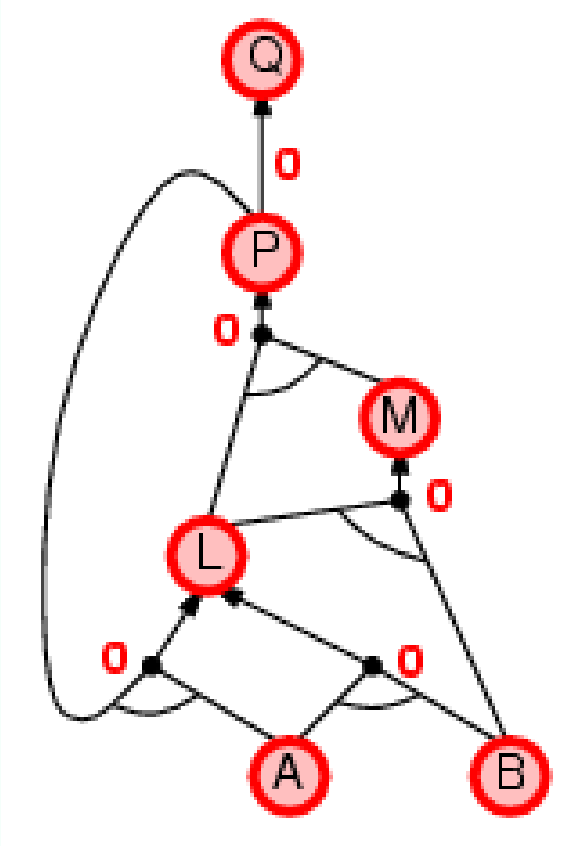
زنجیر پیشرو



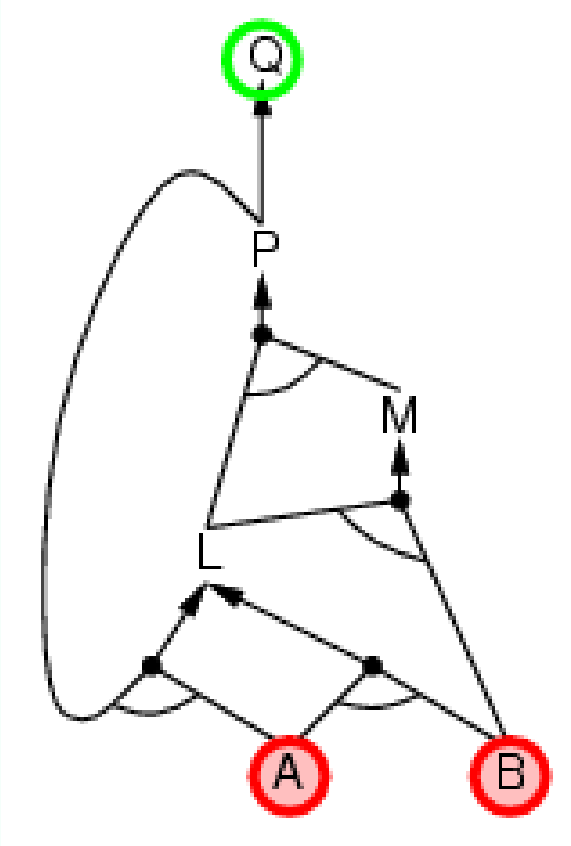
زنجیر پیشرو



زنجیر پیشرو



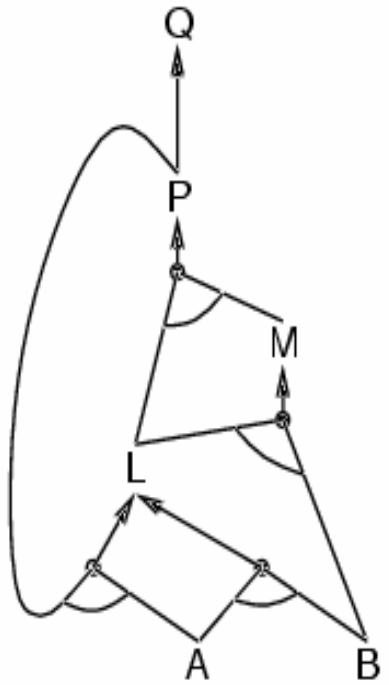
الگوریتم عقبگرد کامل



الگوریتم عقبگرد کامل

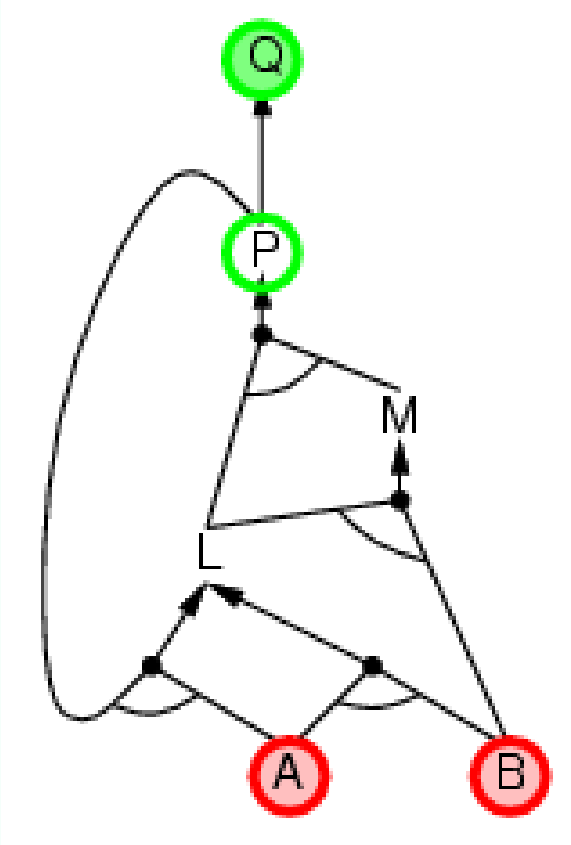
- تغییرات عمده: خاتمه زودرس، اکتشاف نماد محض، اکتشاف عبارت واحد

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

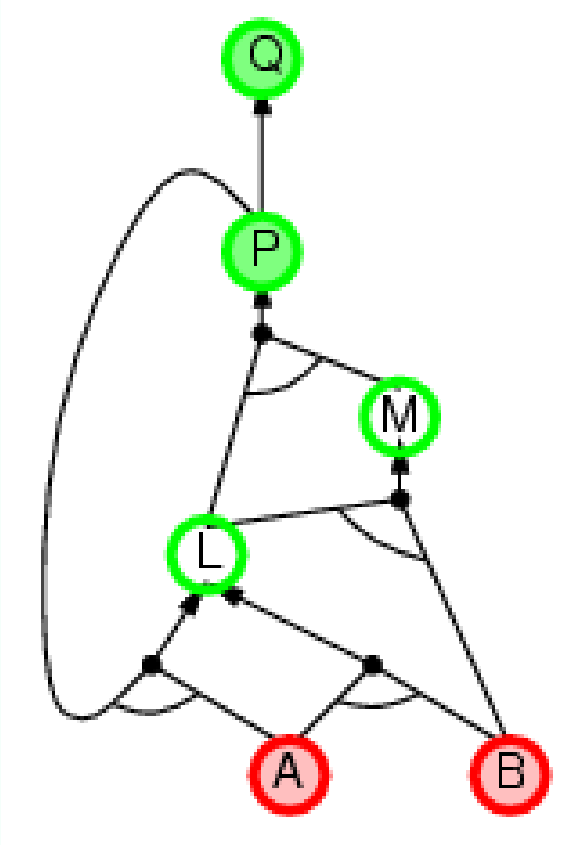


Email: ComputerCollege_Fam@Yahoo.Com

الگوریتم عقبگرد کامل

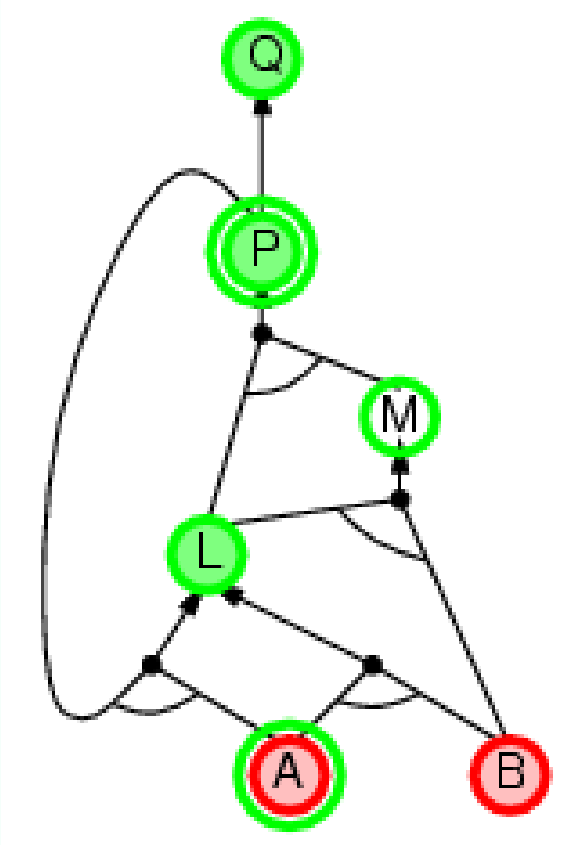


الگوریتم عقبگرد کامل

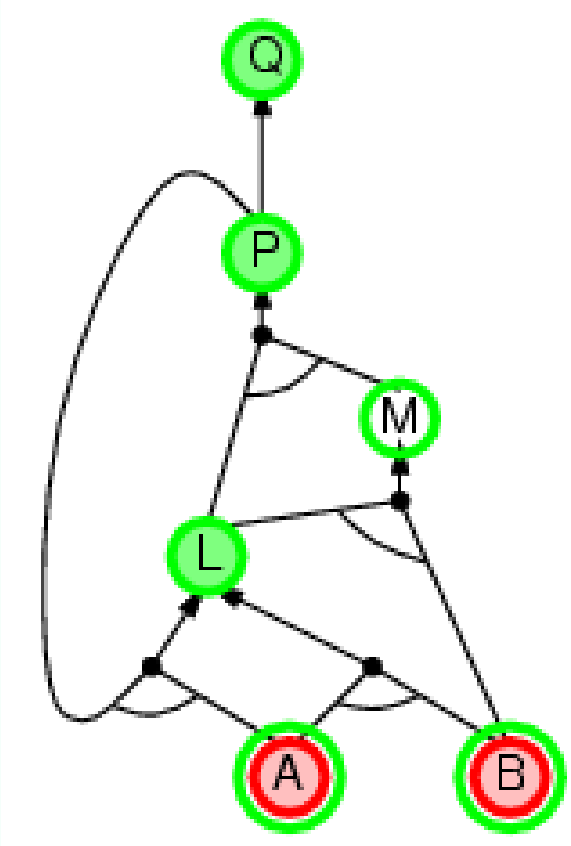


Email: ComputerCollege_Fam@Yahoo.Com

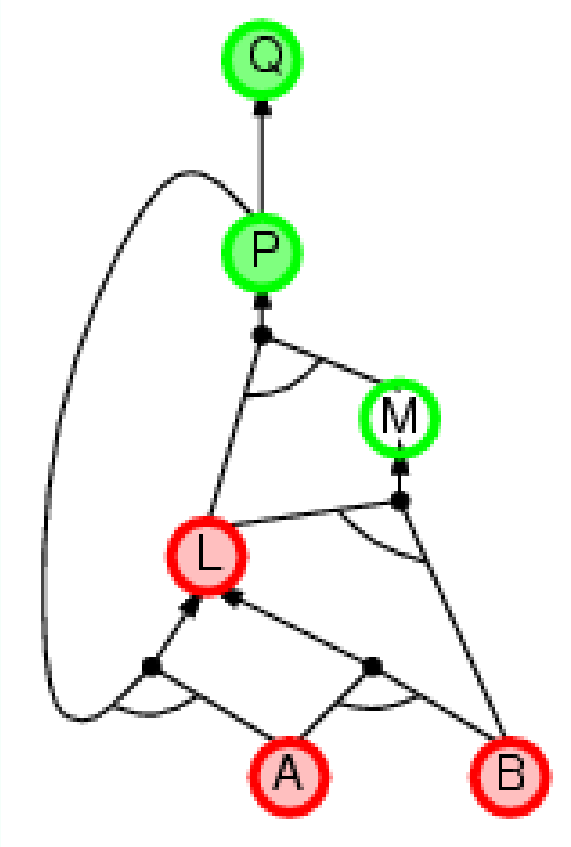
الگوریتم عقبگرد کامل



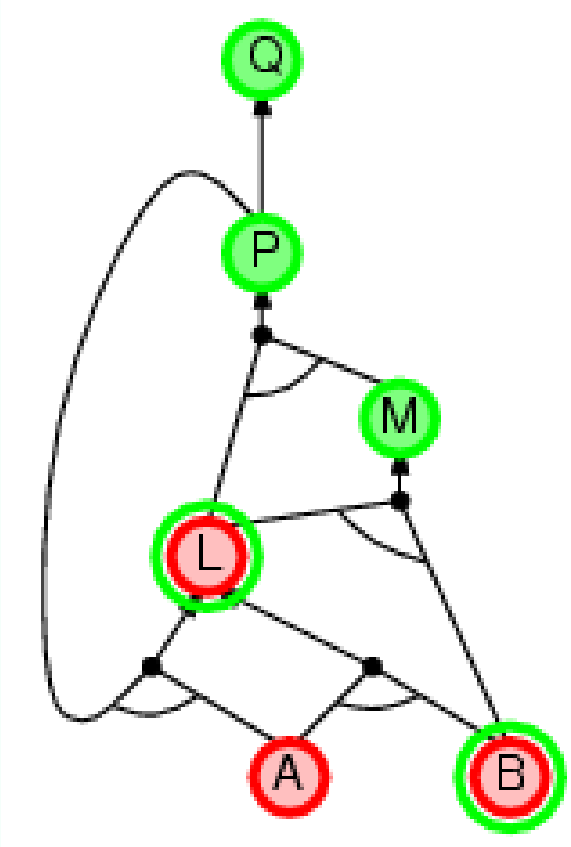
الگوریتم عقبگرد کامل



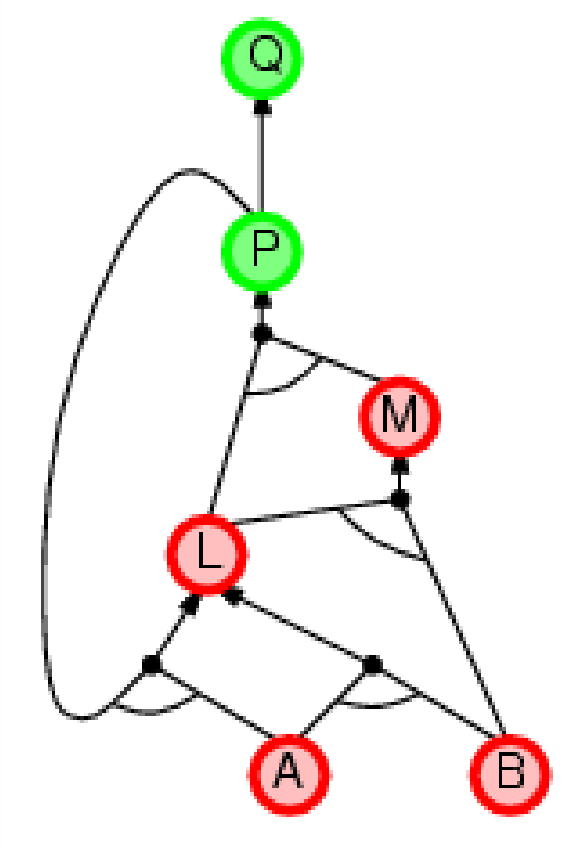
الگوریتم عقبگرد کامل



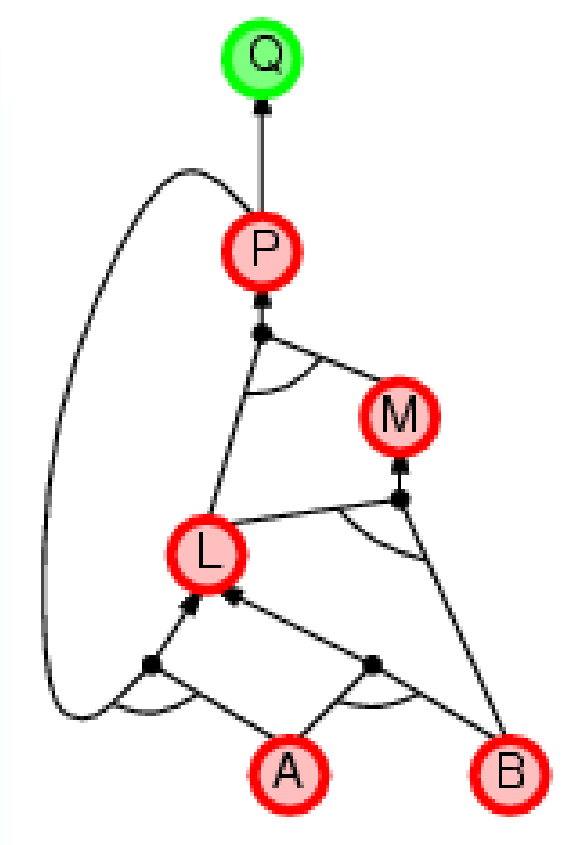
الگوریتم عقبگرد کامل



الگوریتم عقبگرد کامل



الگوریتم عقبگرد کامل



الگوریتم عقبگرد کامل