به نام پروردگار دانایی

هوش مصنوعی

درس ششم بخش اول: جستجوهای آگاهانه – الگوریتم‌ها

سید کاوه احمدی

مشکلات جستجوی ناآگاهانه

- انتخاب گره بعدی برای گسترش فقط بر مبنای عمق گره است.
- از ساختار (صورت) مسئله بهره نمی‌برد.
- درخت جستجو بر اساس یک ساختار از پیش مشخص شده گسترش داده می‌شود.
  - از اطلاعات دریافتی از مسئله (دانش خاص مسئله) هیچ بهره‌ای نمی‌گیرد.

جستجوی آگاهانه – جستجوی اکتشافی

- جستجوی اول بهترین حریصانه
- الگوریتم A*

جستجوی آگاهانه – جستجوی اکتشافی

- در جستجوهای آگاهانه (informed) اطلاعاتی درباره هزینه رسیدن به هدف (نگاه به آینده) در اختیار عامل قرار می‌گیرد.
- ایده این است که گره‌ای اول بسط پیدا کند که فکر می‌کنیم شانس بیشتری برای رسیدن به هدف دارد (نگاه به آینده).
  - جستجوی اکتشافی (heuristic)
    - هدایت جستجو برپایه ساختار مسئله.
    - معمولا ما را به جواب نزدیک می‌کند.
    - از تخمین میزان هزینه رسیدن به هدف از وضعیت فعلی بهره می‌گیرد.

جستجوی آگاهانه – جستجوی اکتشافی

- در این الگوریتم‌ها یک تابع ارزیابی (Evaluation Function) تولید می‌شود که توضیحاتی در مورد میزان مطلوب بودن بسط یک گره ارائه می‌دهد.
  - چقدر مطلوب است که گره را بسط بدهیم؟
- گره‌ای که برای بسط انتخاب می‌شود بر اساس نتایج تابع ارزیابی بهترین خواهد بود.
- پیاده‌سازی از طریق صف اولویت
  - گره‌ها براساس مطلوب بودن مرتب شده‌اند

تابع ارزیابی

- با `f(n)` نمایش داده می‌شود و قرار است تخمینی از هزینه‌ی کوتاه‌ترین مسیر تا هدف که از گره `n` عبور می‌کند باشد.
- به زبان ساده از این تابع برای مشخص کردن اولویت گره در صف اولویت بهره گرفته می‌شود.
- در جستجوی آگاهانه، این تابع از دانش اضافی پیرامون مسئله برای ارزیابی بهره می‌گیرد.

تابع ارزیابی

- در ایجاد تابع ارزیابی در جستجوهای مختلف، از توابع زیر بهره گرفته می‌شود:
  - `g(n)` - تابع هزینه مسیر: هزینه مسیر از گره اولیه تا گره `n`
  - `h(n)` - تابع اکتشافی (`Heuristic Function`): هزینه تخمینی ارزان ترین مسیر از گره `n` به گره هدف
- چند نکته:
  - تابع اکتشافی (ابتکاری) باید قابل پیاده‌سازی باشد و خروجی آن یک کمیت قابل مقایسه. شرح ابتکاری ما برای تخمین فاصله تا هدف اگر قابل پیاده‌سازی نباشد به درد ما نخواهد خورد.
  - در جستجوی با هزینه یکسان (ناآگاهانه)، در عمل `f(n) = g(n)` است.
    - هزینه یکسان به این مفهوم است که `h(n)` تمامی گره‌ها یکسان در نظر گرفته می‌شود.

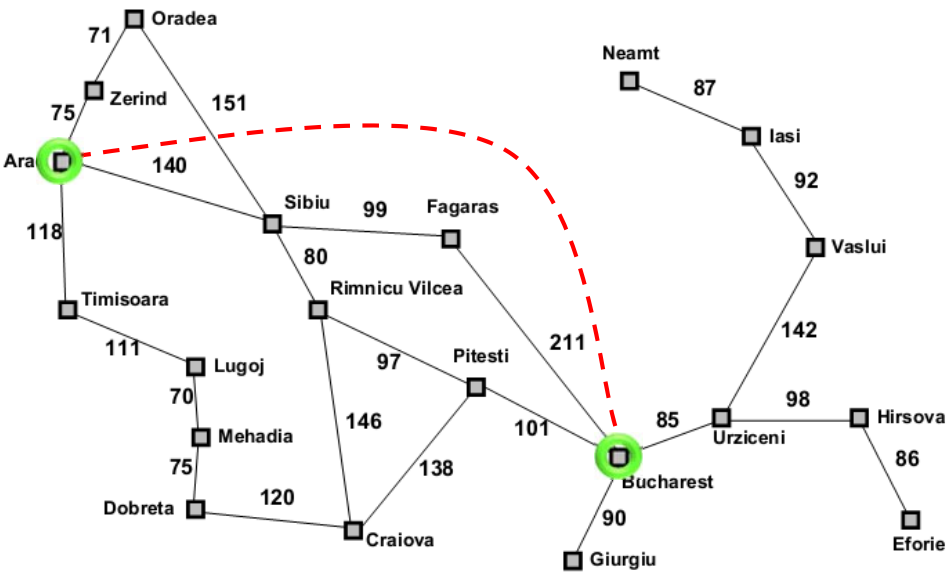
تابع ارزیابی

- توابع زیر چه چیزی را مشخص می‌کنند؟
  - $g^*(n)$: کوتاه‌ترین فاصله ممکن از مبدا تا گره جاری `n`.
  - $h^*(n)$: کوتاه‌ترین فاصله از گره جاری `n` تا نزدیک‌ترین هدف.
  - $f^*(n)$: کوتاه‌ترین فاصله از مبدا تا هدف به شرط عبور از گره `n`.
- $f^*(n) = g^*(n) + h^*(n)$
  - آیا $f^*(n)$ مسیر بهینه از مبدا تا هدف را مشخص می‌کند؟
    - فقط در صورتی که `n` روی مسیر بهینه باشد.
- C^*: همان $f^*(n)$ است که `n` گره هدف باشد (طول مسیر بهینه).

یک تابع اکتشافی در مسئله نقشه رومانی

- فاصله مستقیم شهر n تا بخارست: $h_{SLD}(n)$
  - فاصله روی نقشه نیست.
  - فاصله مستقیم به ما گرهی را معرفی می‌کند که به نظر می‌رسد به هدف نزدیک‌تر است.
  - به جدول فاصله مستقیم شهرها نیاز خواهیم داشت.
    - کدام شهرها به بخارست نزدیکترند و کدام دورتر.
- h_{SLD} نباید از روی توصیفات مسئله محاسبه شود.

یک تابع اکتشافی در مسئله نقشه رومانی



فاصله مستقیم هر شهر تا هدف (بخارست)

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

h(n)

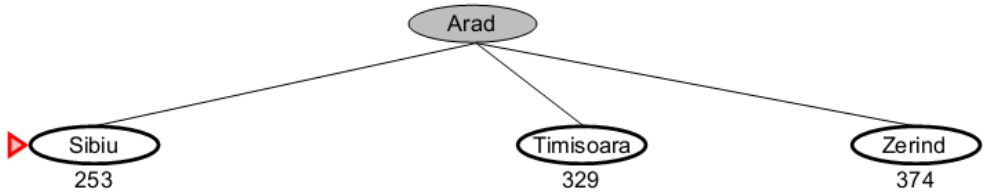
جستجوی اول بهترین حریصانه

- گره‌ای را بسط می‌دهد که به نظر می‌رسد (appears) نزدیکترین گره به هدف است و احتمالاً سریعتر به هدف می رسد.
  - گره‌ها را تنها بر مبنای تابع اکتشافی ارزیابی می‌کند: $f(n) = h(n)$
  - در واقع صف اولویت بر اساس $h(n)$ مرتب می‌شود.
- مانند الگوریتم اول عمق برای رسیدن به یک هدف یک مسیر را دنبال می‌کند و اگر به بن بست رسید به بالا بر می‌گردد.

جستجوی اول بهترین حریصانه

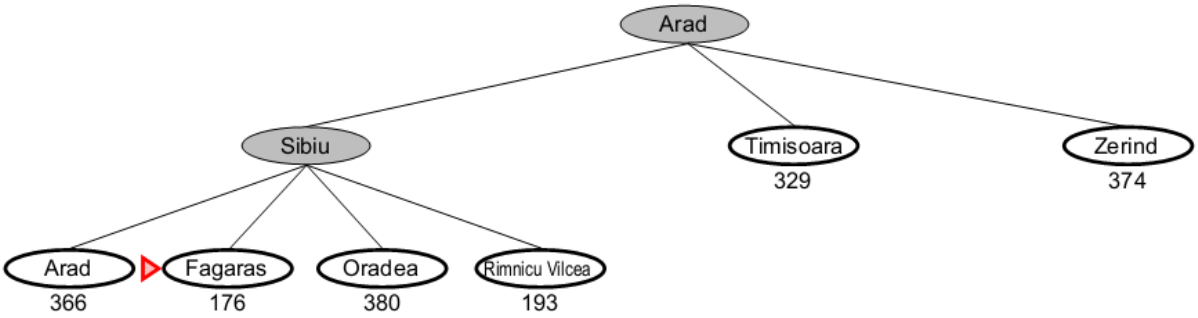
- در مثال رومانی

جستجوی اول بهترین حریصانه

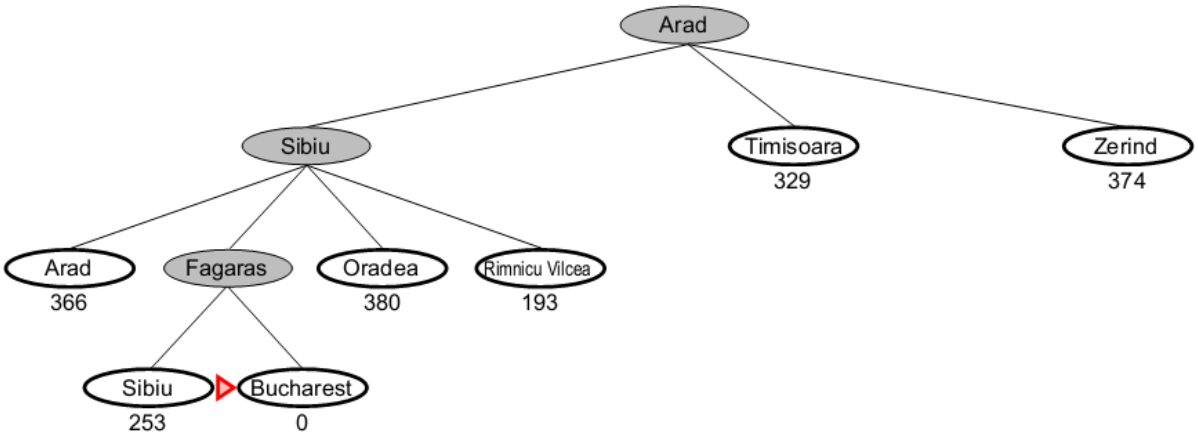


گره‌های خاکستری بسط داده شده‌اند و گره‌های سفید گره‌های frontier است.

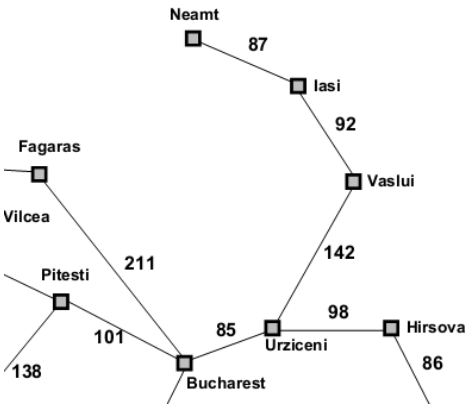
جستجوی اول بهترین حریصانه



جستجوی اول بهترین حریصانه



جستجوی اول بهترین حریصانه



- کامل نیست: ممکن است در یک حلقه گیر کند:
  - هدف: رفتن از lasi به faragas
    - تابع اکتشافی گفته شده پیشنهاد می‌کند neamt اول بسط داده شود چون به هدف نزدیکتر است (در خط مستقیم).
  - → neamt → lasi→ neamt →lasi
  - …
- در فضای محدود با چک کردن گره‌های تکراری کامل است.

جستجوی اول بهترین حریصانه

- بهینه نیست.
  - فقط تلاش می‌کند که در هر مرحله به هدف نزدیکتر شود (حریصانه).
  - در مثال رومانی هم مسیر پیدا شده بهینه نیست.
- از نظر دنبال کردن یک مسیر ویژه در تمام طول جستجو همانند جستجوی عمقی است.
  - با مشکلات جستجوی اول عمق نیز مواجه است!
- پیچیدگی زمانی در بدترین حالت: $O(b^m)$
  - m حداکثر عمق فضای جستجو
  - با استفاده از تابع اکتشافی مناسب، می‌توان این مقدار را به شدت کاهش داد.
- پیچیدگی حافظه همانند پیچیدگی زمانی
  - چون تمام گره‌ها را در حافظه نگه می‌دارد.

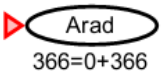
جستجوی اول بهترین حریصانه

میزان کاهش پیچیدگی به مسئله و کیفیت تابع h بستگی دارد و در صورتیکه تابع مناسبی انتخاب شود، هزینه می‌تواند بسیار بهبود یابد

جستجوی A*

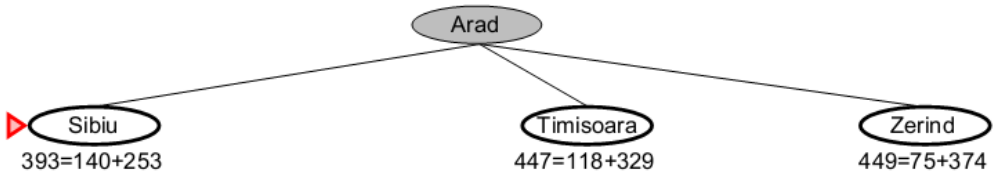
- تابع ارزیابی:
  - $f(n) = g(n) + h(n)$
- ترکیب مزایای جستجو با هزینه یکسان و حریصانه
- پرهیز از بسط گره‌هایی که تا کنون پرهزینه بودن آنها اثبات شده است.

جستجوی A*

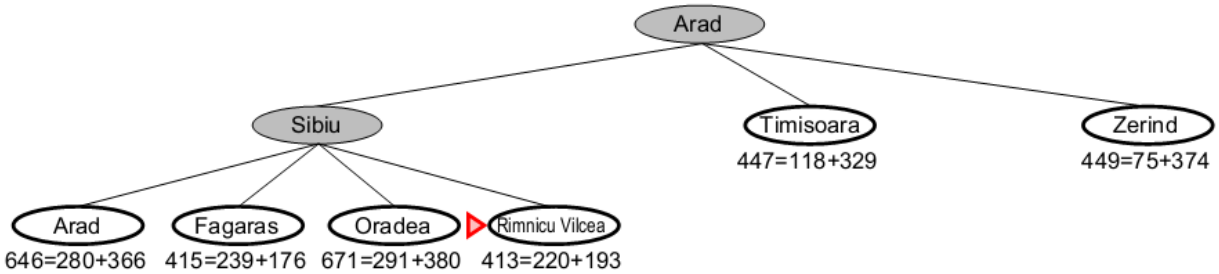


f(Arad) = g(Arad)+h(Arad)=0+366=366

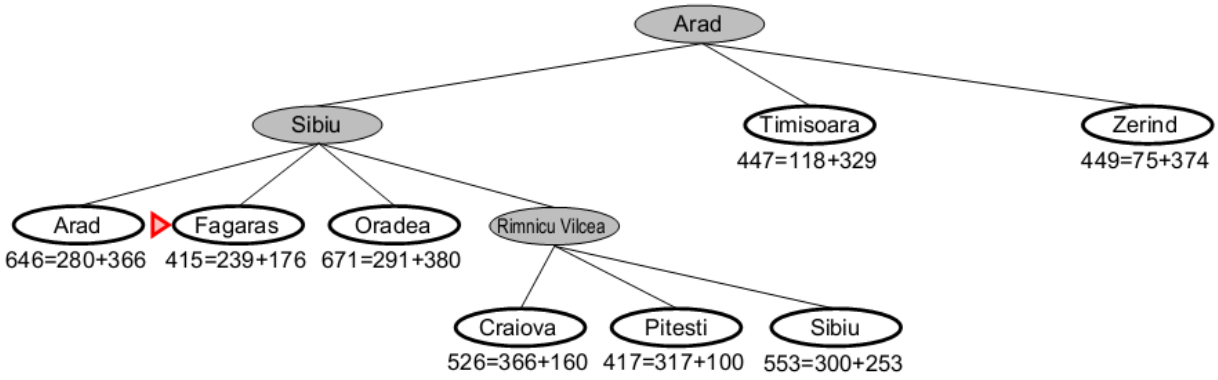
جستجوی A*



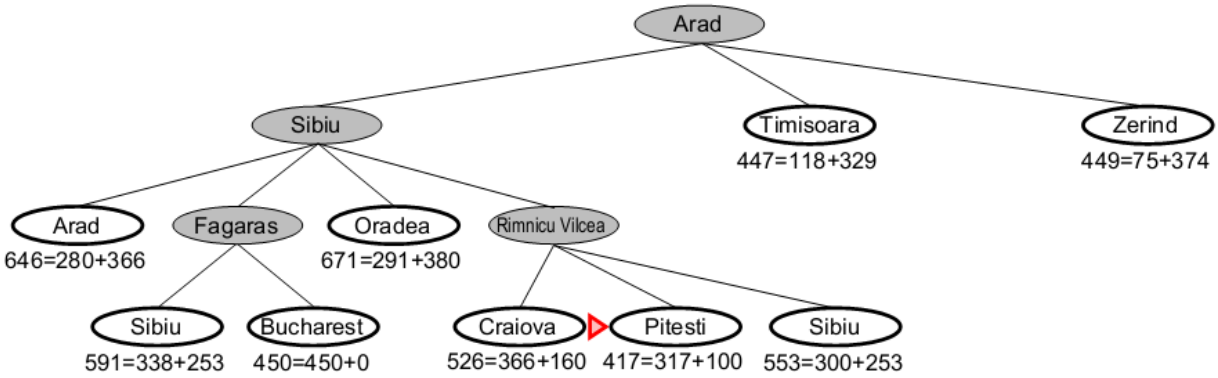
جستجوی A*



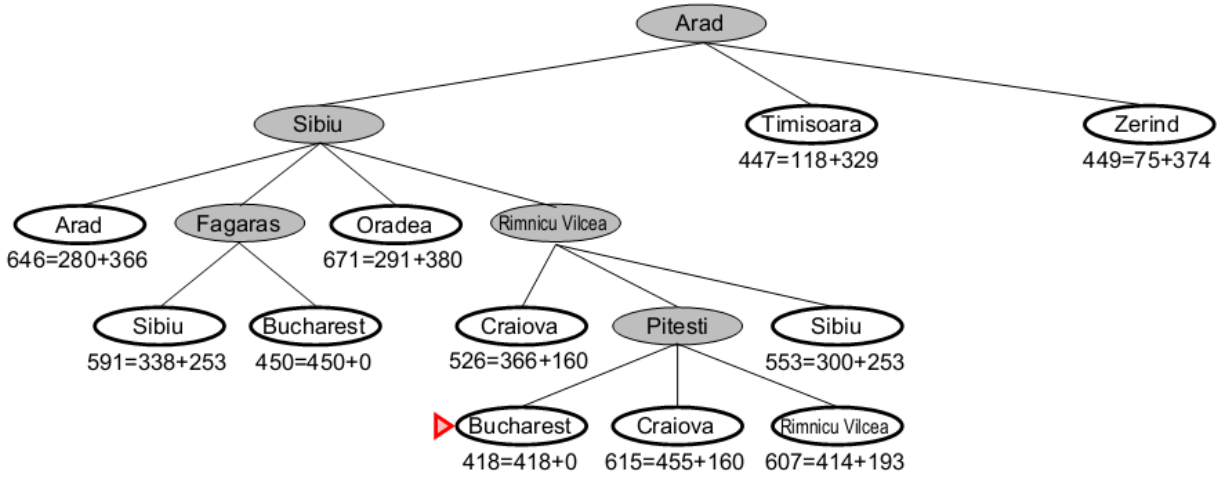
جستجوی A*



جستجوی A*



جستجوی A*

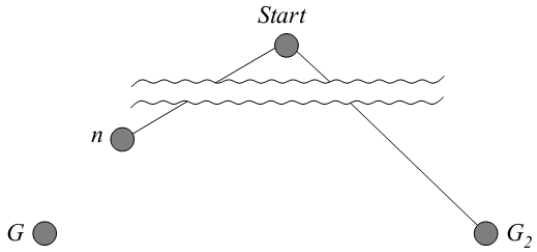


بهینگی A*

- A* بهینه است اگر از یک تابع اکتشاف قابل قبول (Admissible Heuristic) استفاده کند.
- یک تابع اکتشاف قابل قبول است اگر برای هر گره n داشته باشیم:
  - $h(n) \leq h^*(n)$ *where* $h^*(n)$ *is the true cost of* n
  - $h(n) \geq 0,\ so\ h(G) = 0\ for\ any\ goal\ G$
- فارسی: یک تابع اکتشاف قابل قبول هیچگاه هزینه رسید به هدف از یک گره را بیشتر از مقدار واقعی تخمین نمی‌زند.
  - تابع اکتشافی h_{SLD} (مثال رومانی) قابل قبول است چراکه هرگز هزینه مسیر واقعی نمی‌تواند از هزینه خط مستقیم تا هدف کمتر باشد (خط مستقیم بین دو نقطه کوتاه‌ترین مسیر بین آن دو نقطه است).

اثبات بهینگی A*

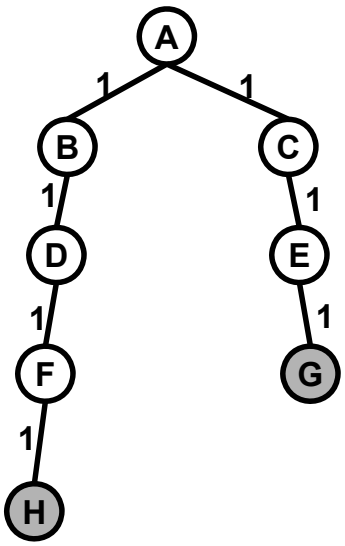
- فرض کنید G2 یک هدف زیربهینه (suboptimal) است که ایجاد شده و در صف قرار دارد.
- n یک گره بسط داده نشده در کوتاه‌ترین مسیر به هدف بهینه G است.



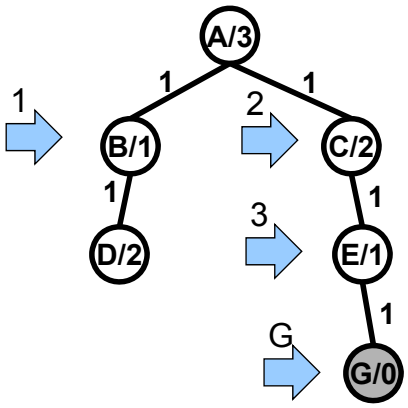
$$f(G_2) = g(G_2) \quad since\ h(G_2) = 0$$
$$> f(G) \quad since G_2\ is\ suboptimal$$
$$\geq f(n) \quad since\ h\ is\ admissible$$

مادامی که f(G2) > f(n)، جستجوی A* هیچگاه G2 را برای بسط انتخاب نمی کند

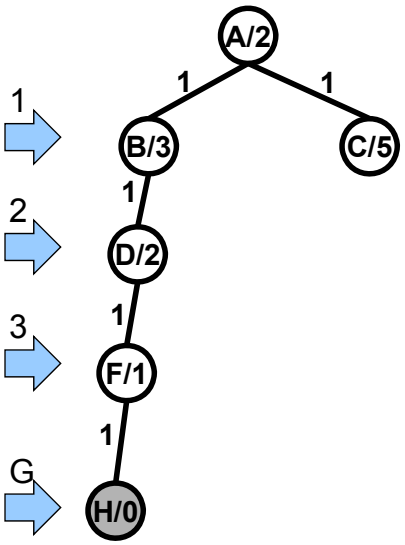
رفتار A* در برابر هیوریستیک‌های متفاوت



رفتار A* در برابر هیوریستیک‌های متفاوت

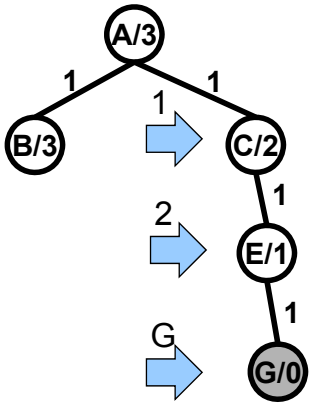


h ≤ h*



h ≥ h*

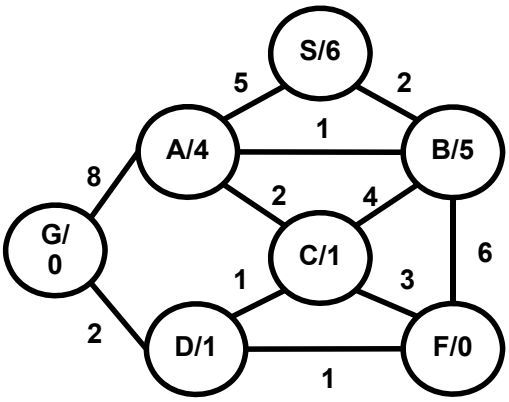
رفتار A* در برابر هیوریستیک‌های متفاوت



h = h*

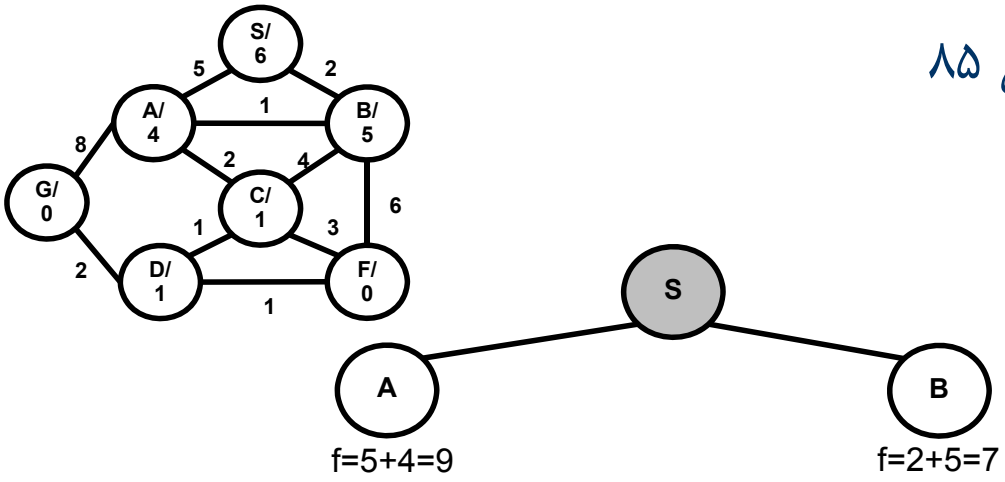
هوش مصنوعی ۸۵

- در گراف زیر حاصل جستجو با روش A* چیست؟ (S نقطه شروع است و اعداد روی یال ها هزینهٔ واقعی و اعداد داخل دایره ها مقدار h گره مورد نظر است)

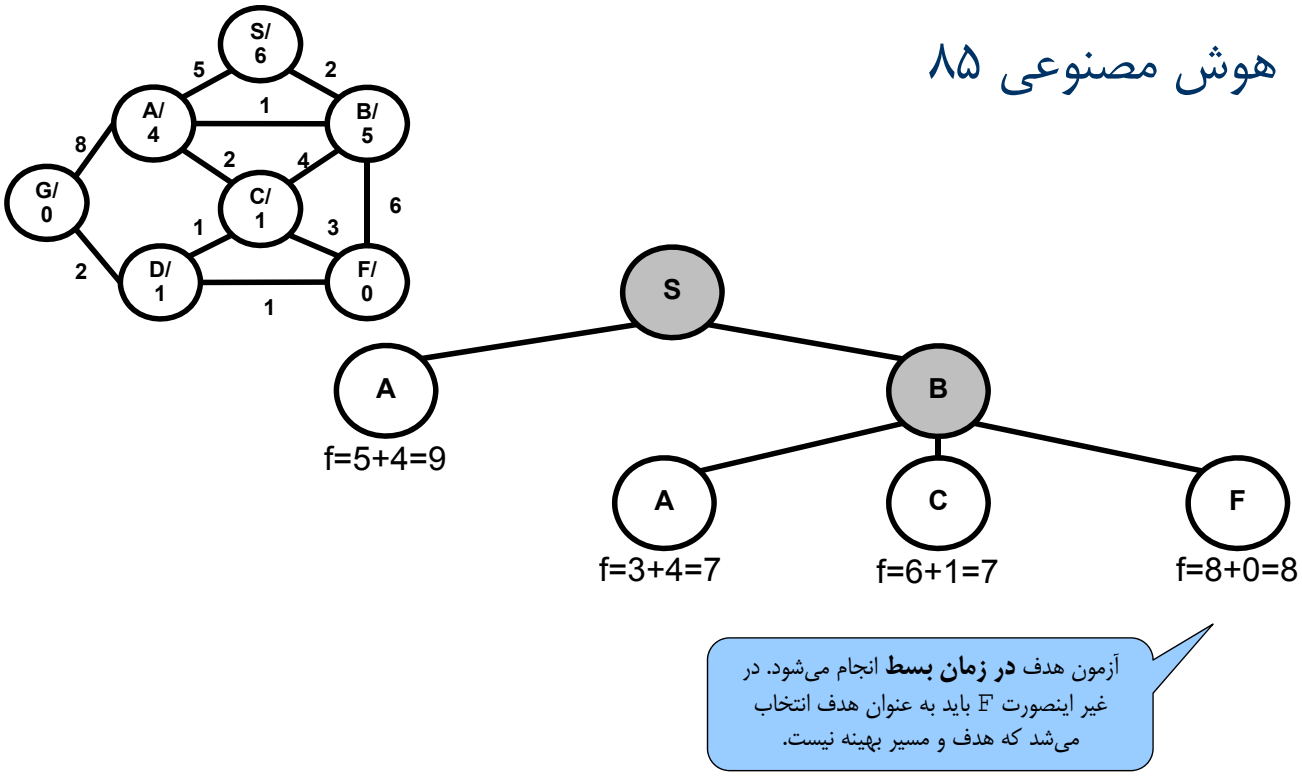


1. SBF
2. SBAG
3. SBCDG
4. SBACDF

هوش مصنوعی ۸۵

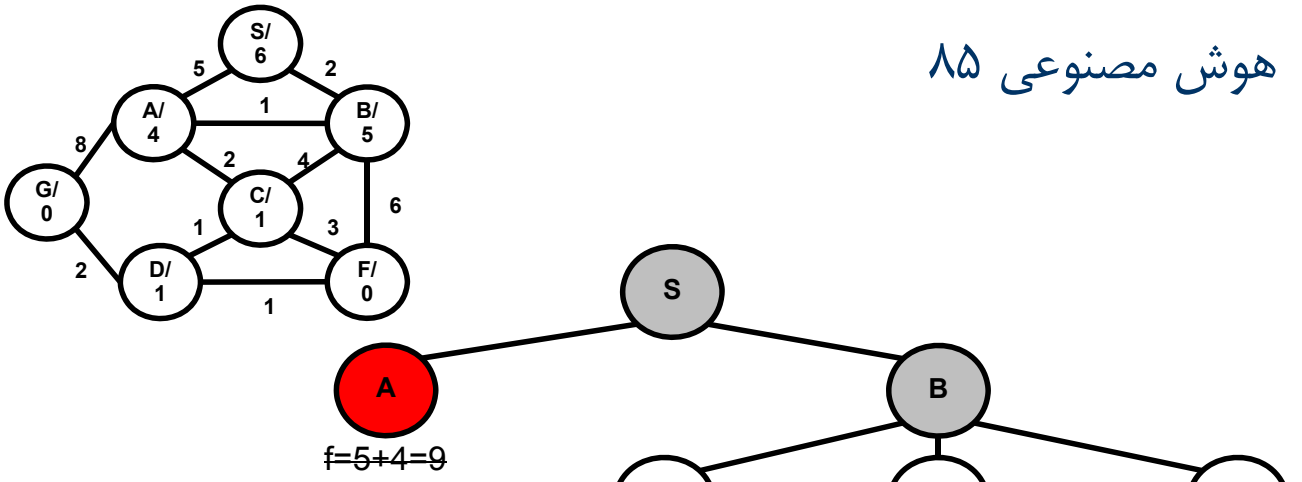


هوش مصنوعی ۸۵



آزمون هدف **در زمان بسط** انجام می‌شود. در غیر اینصورت F باید به عنوان هدف انتخاب می‌شد که هدف و مسیر بهینه نیست.

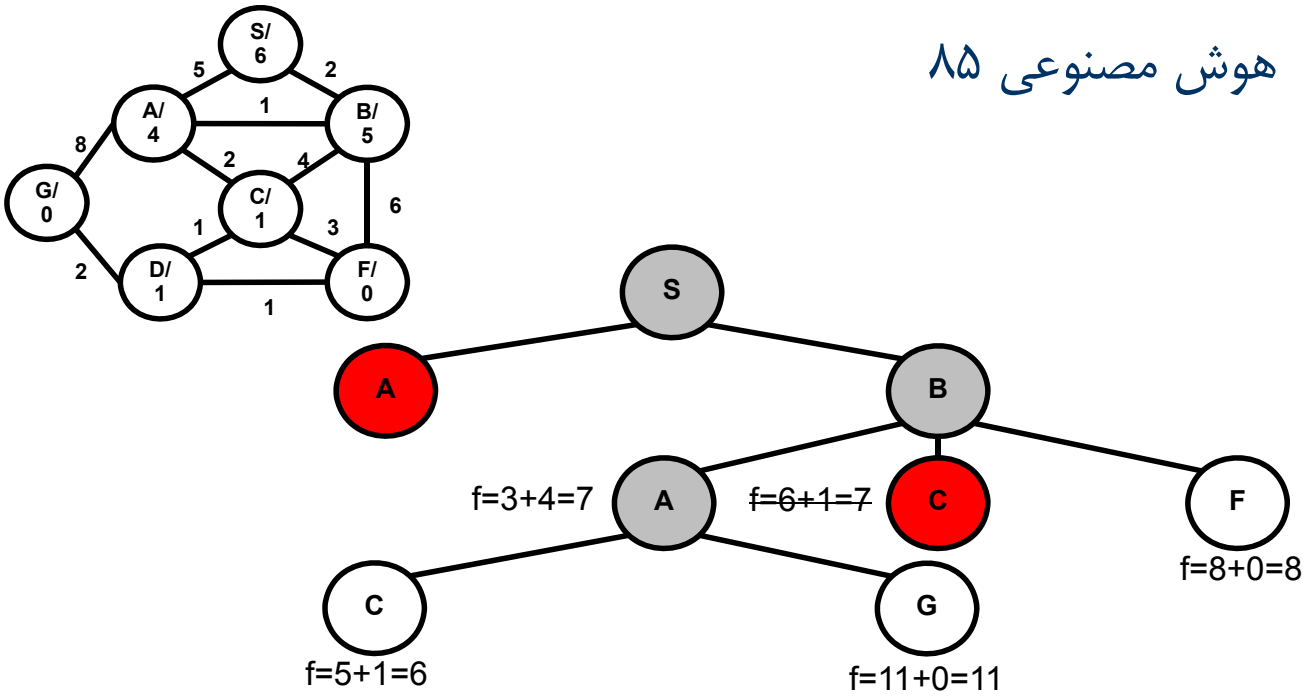
هوش مصنوعی ۸۵



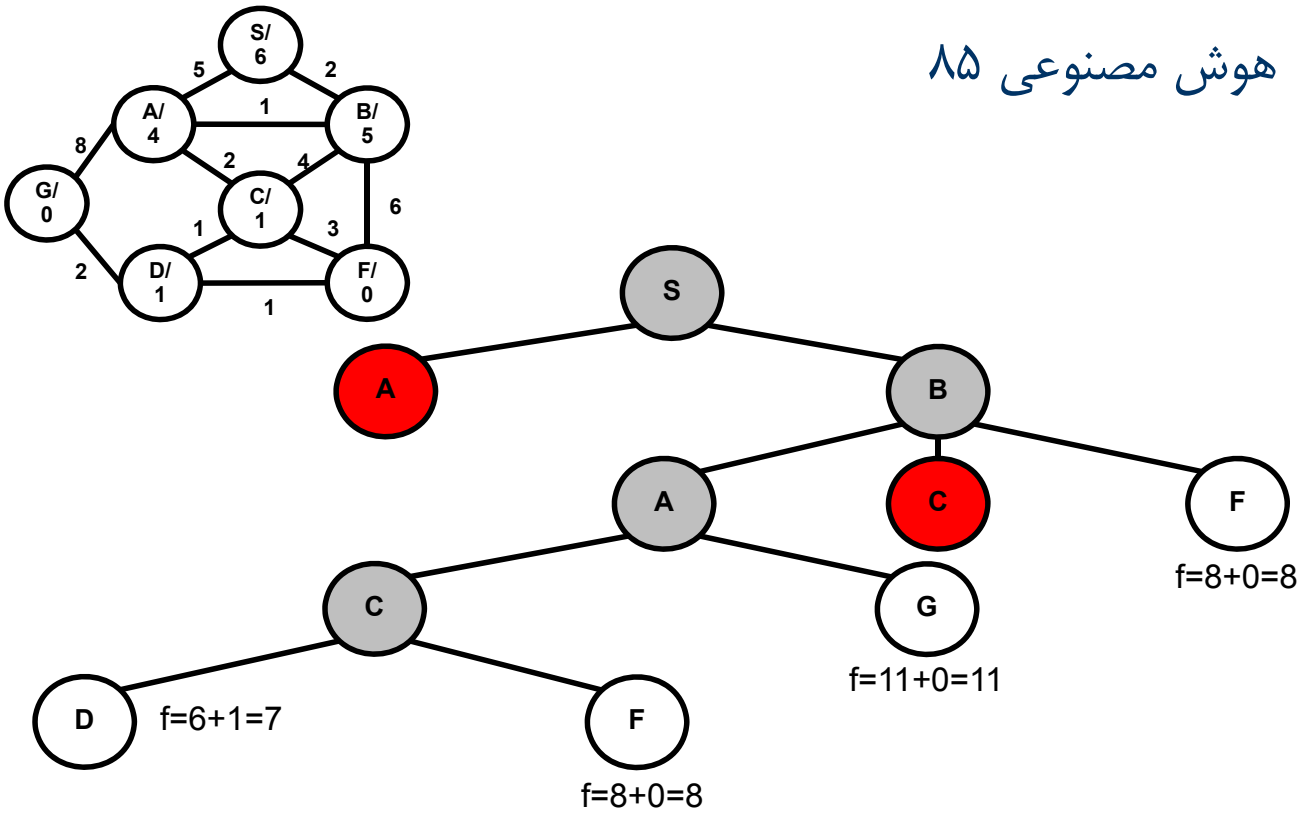
A در frontier وجود دارد و دوباره به آن اضافه نمی‌شود. اما از آنجا که با f کمتر می‌توانیم به آن برسیم، f و پدر آن به روز می‌شود. (مشابه آنچه در UCS دیدیم). بنابراین Aیی که از طریق B به آن می‌رسیم در frontier باقی می‌ماند.

A و C دارای f یکسان هستند. در تست‌ها گره‌ها را به ترتیب الفبا پیمایش می‌کنیم. مگر آنکه روش دیگری ذکر شده باشد.

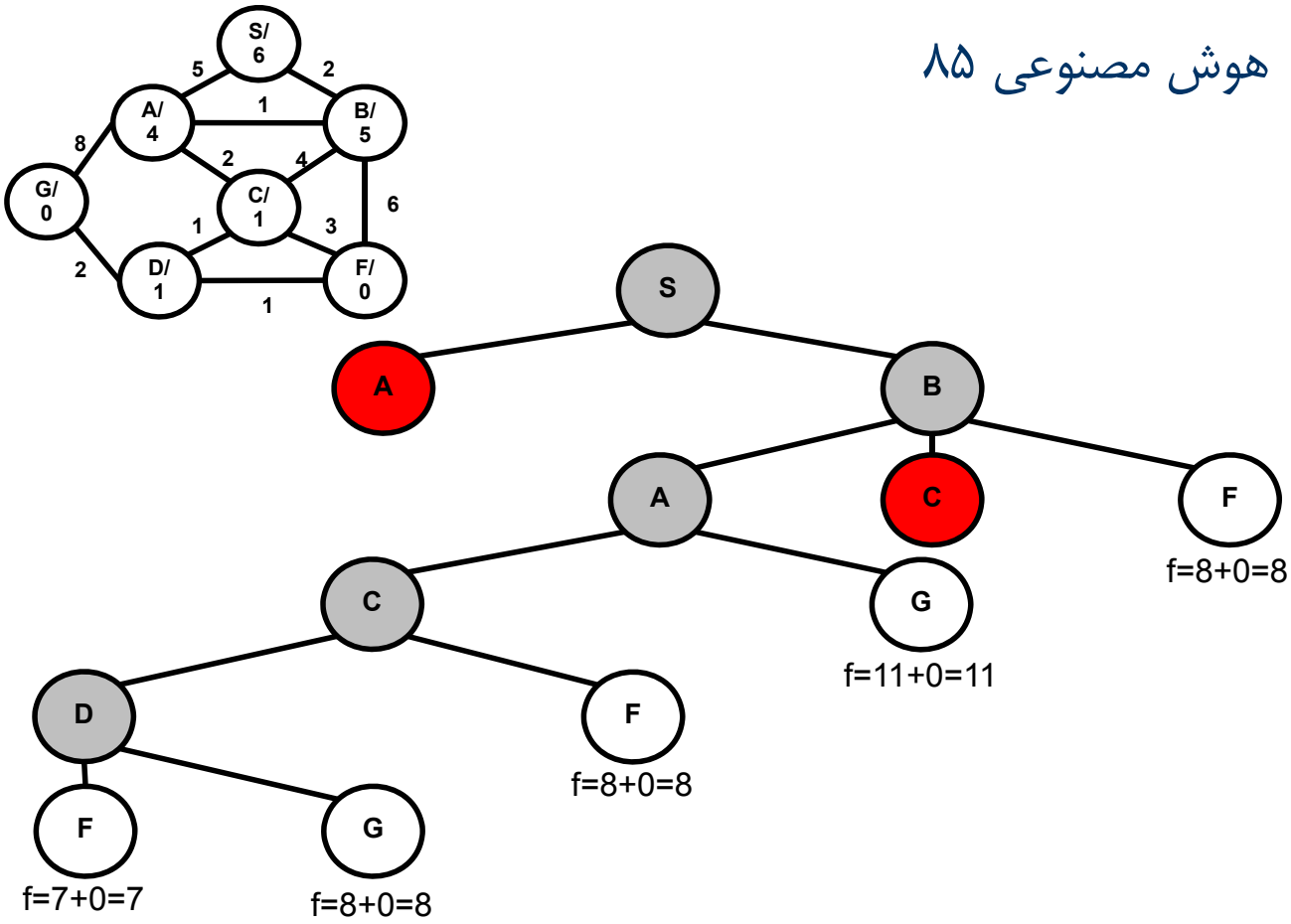
هوش مصنوعی ۸۵



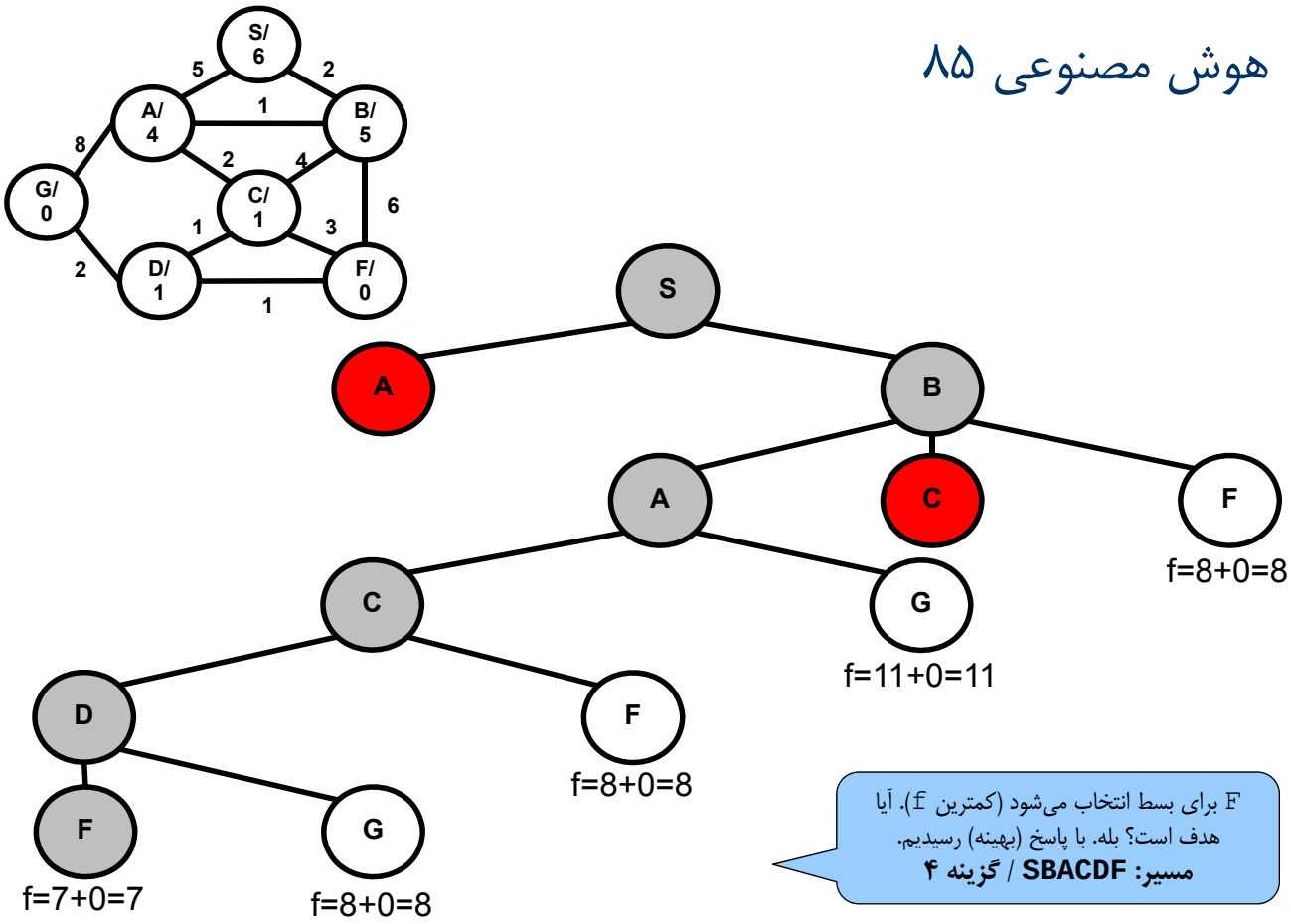
هوش مصنوعی ۸۵



هوش مصنوعی ۸۵



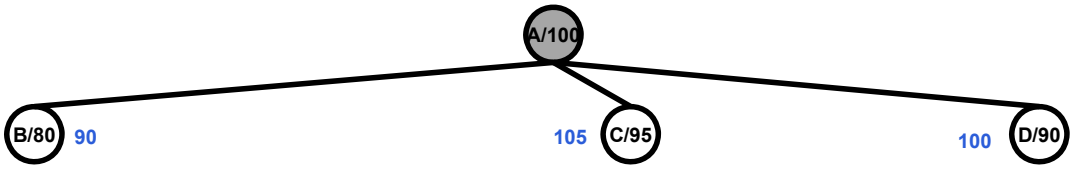
هوش مصنوعی ۸۵



جستجوی A* در فضای حالت گرافی

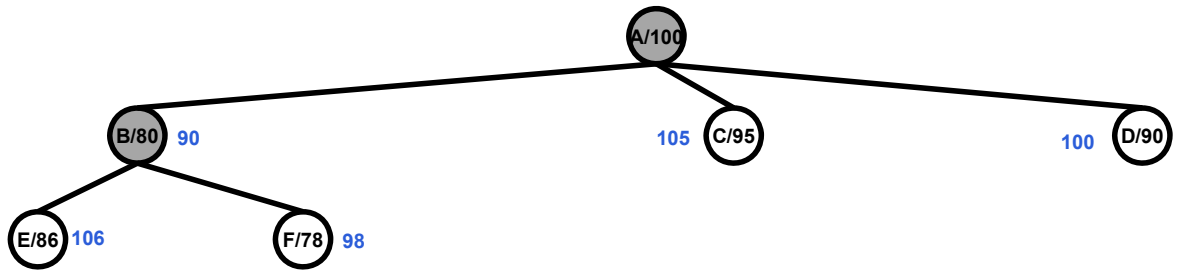
- در حالتی که فضای جستجو گرافی است هریک از موقعیت‌های زیر ممکن است پیش بیاید:
  - اگر پس از شروع از یک وضعیت در نهایت به همان وضعیت بازگردیم، یک حلقه به وجود می‌آید.
  - ممکن است مسیرهای متعددی از یک وضعیت به وضعیت دیگری وجود داشته باشد.
- استفاده از `Open-List` به تنهایی در این حالت ممکن است باعث پیدا نشدن مسیر بهینه شود. چراکه مسیر جایگزین موجود ممکن است هزینه کمتری داشته باشد.
- راه حل عملی:
  - نگهداری `Open-List` و `Close-List`
  - چک کردن گره‌های جدید تولید شده از جهت امکان تکراری بودن
  - در نظر گرفتن مجدد گره‌های تکراری در صورت نیاز (در صورتی که با هزینه کمتر به آن برسیم)
    - در مورد گره‌های درون `Open-List` به روز کردن `f` و مسیر
    - در مورد گره‌های درون `Close-List` عملیات `Reopen or Update`

مثال: درخت جستجوی A* در فضای حالت گرافی

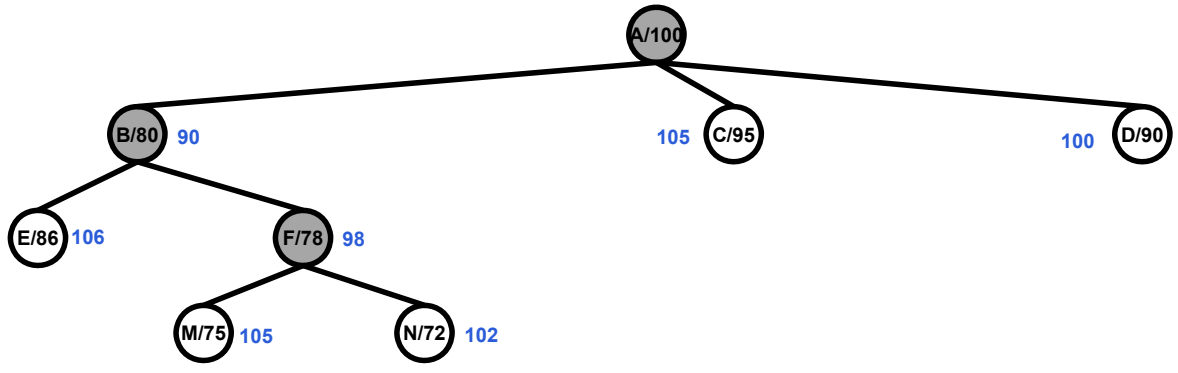


هزینه هر مرحله ۱۰ میباشد

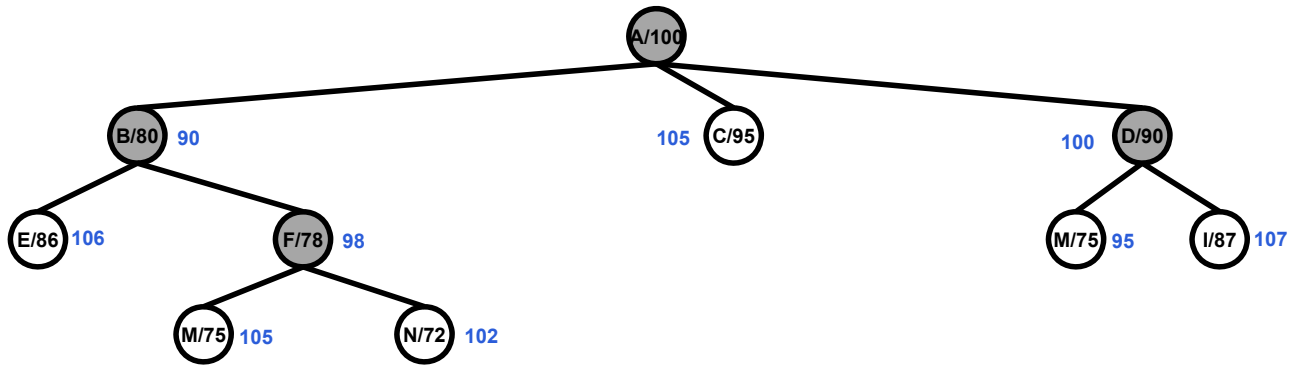
مثال: درخت جستجوی A* در فضای حالت گرافی



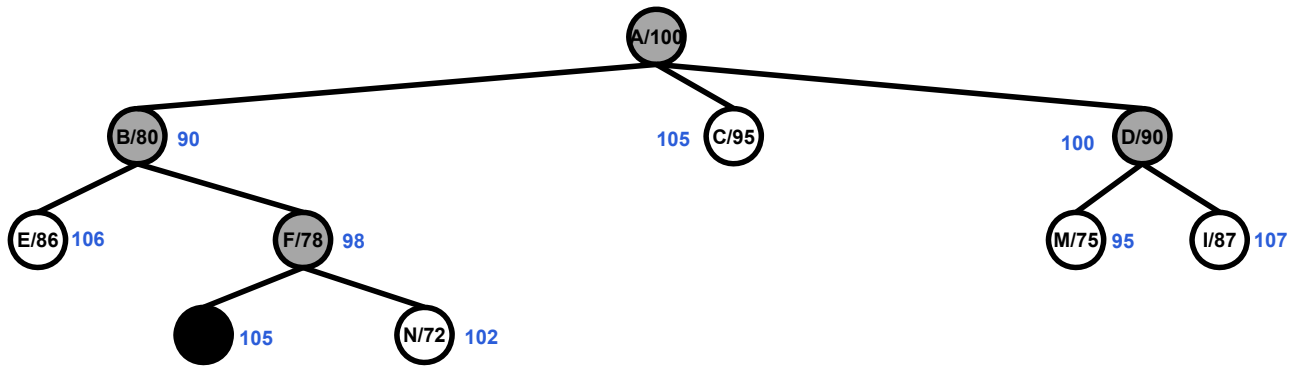
مثال: درخت جستجوی A* در فضای حالت گرافی



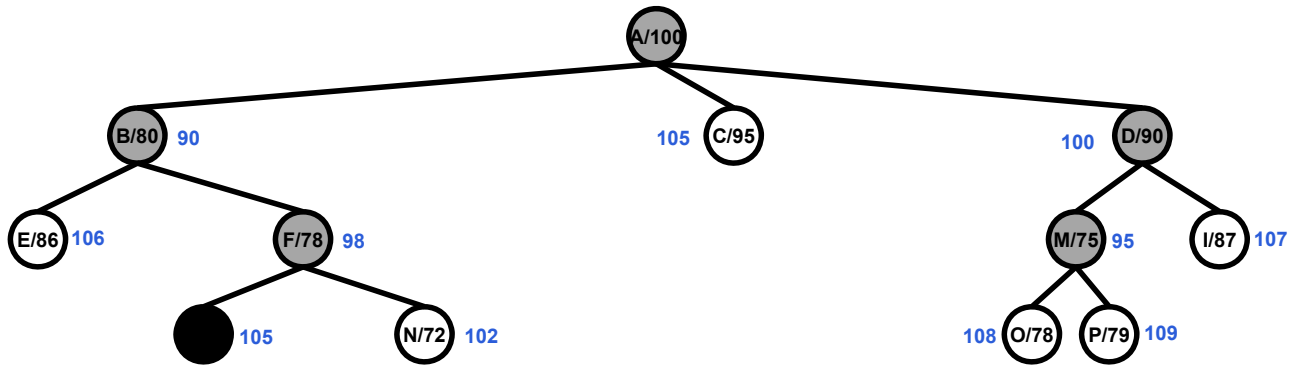
مثال: درخت جستجوی A* در فضای حالت گرافی



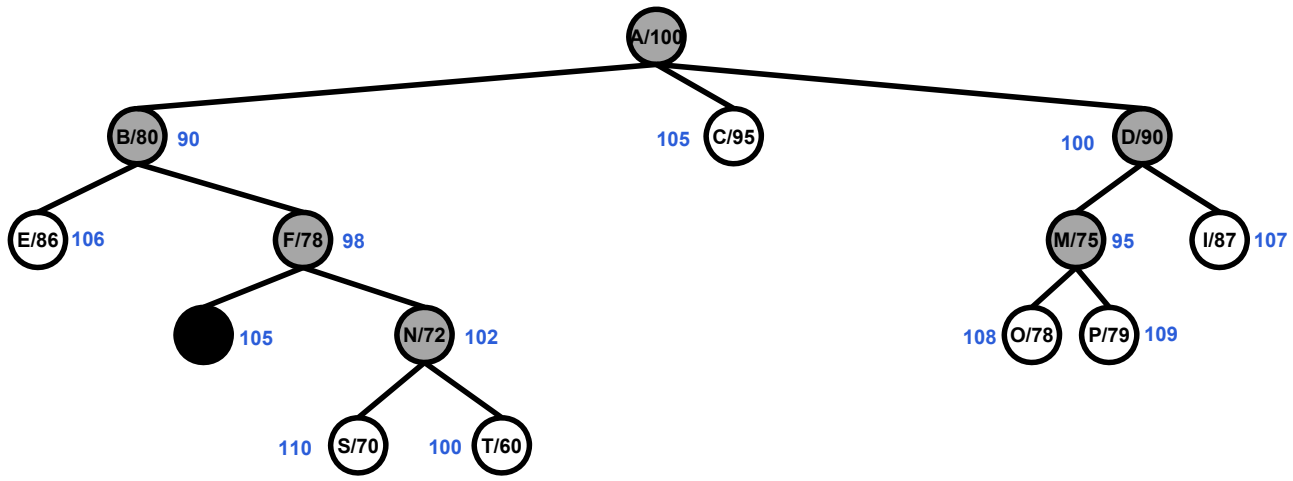
مثال: درخت جستجوی A* در فضای حالت گرافی



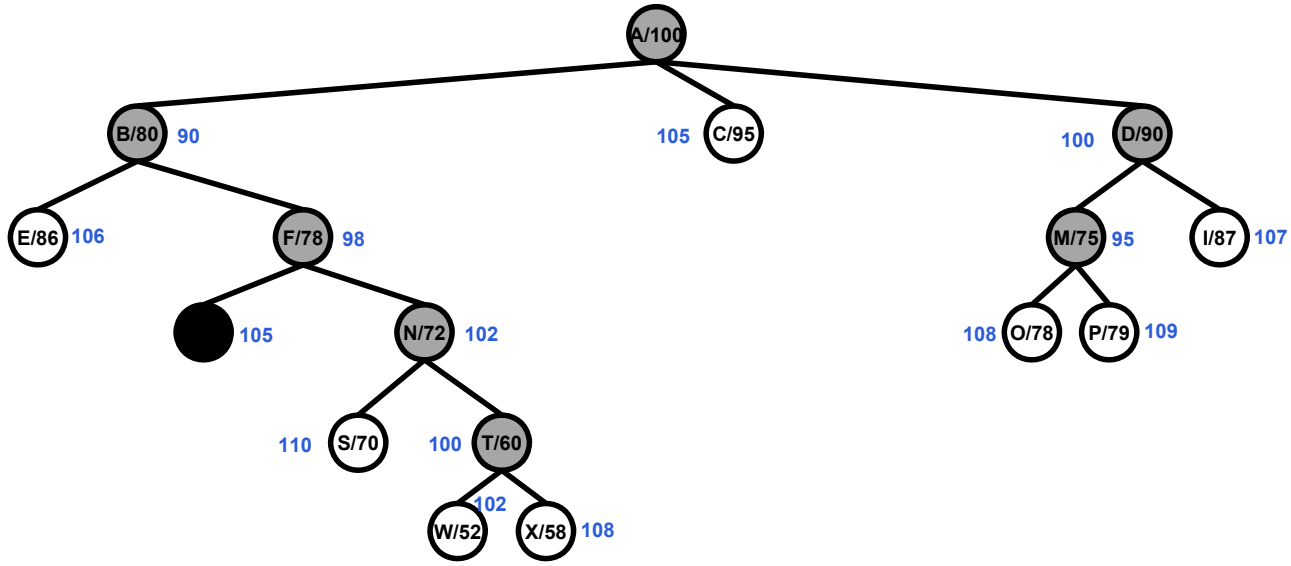
مثال: درخت جستجوی A* در فضای حالت گرافی



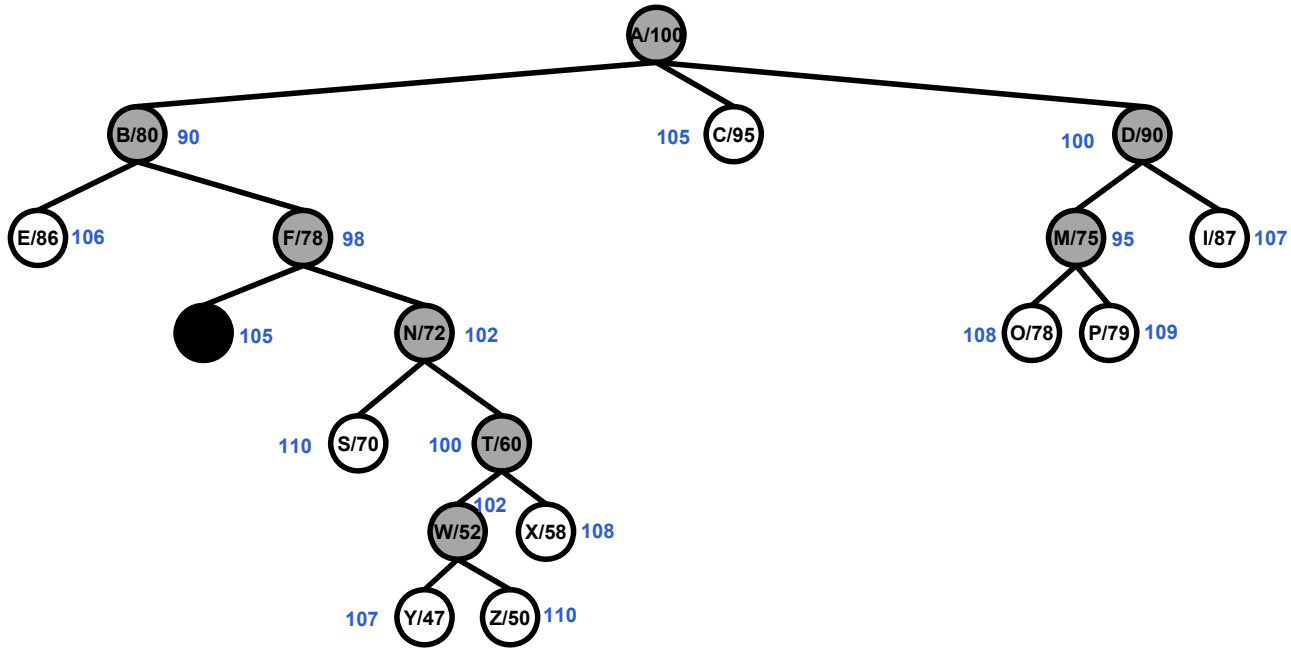
مثال: درخت جستجوی A* در فضای حالت گرافی



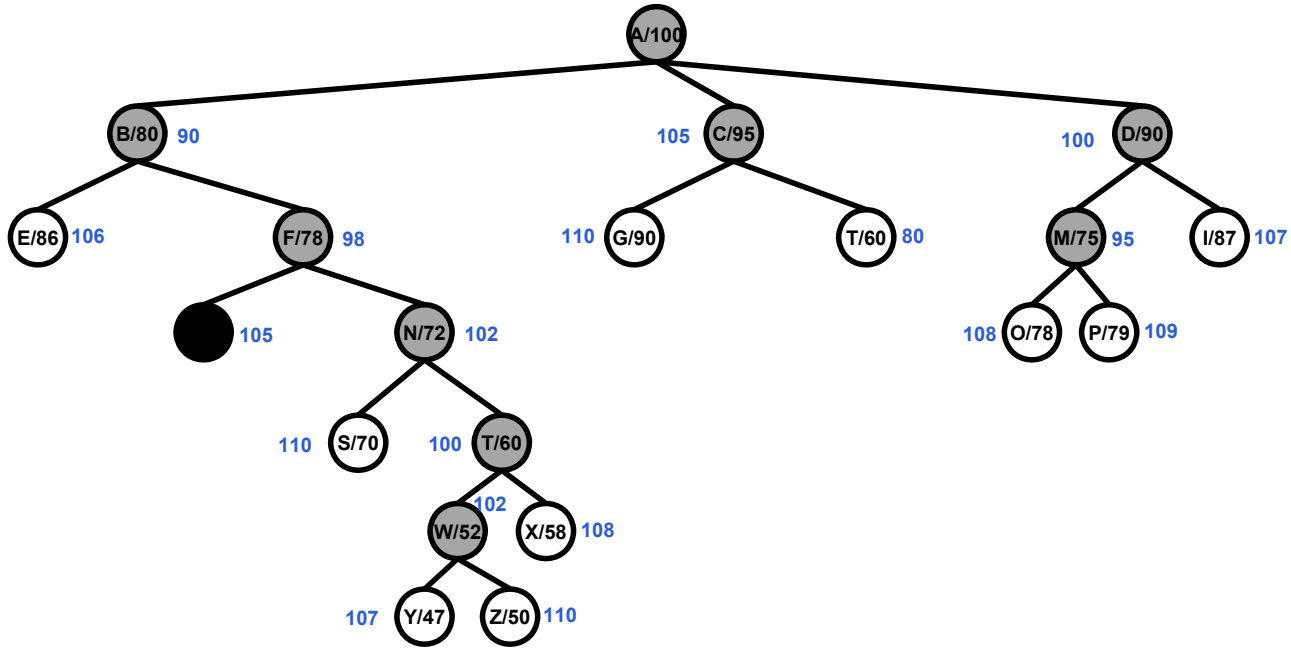
مثال: درخت جستجوی A* در فضای حالت گرافی



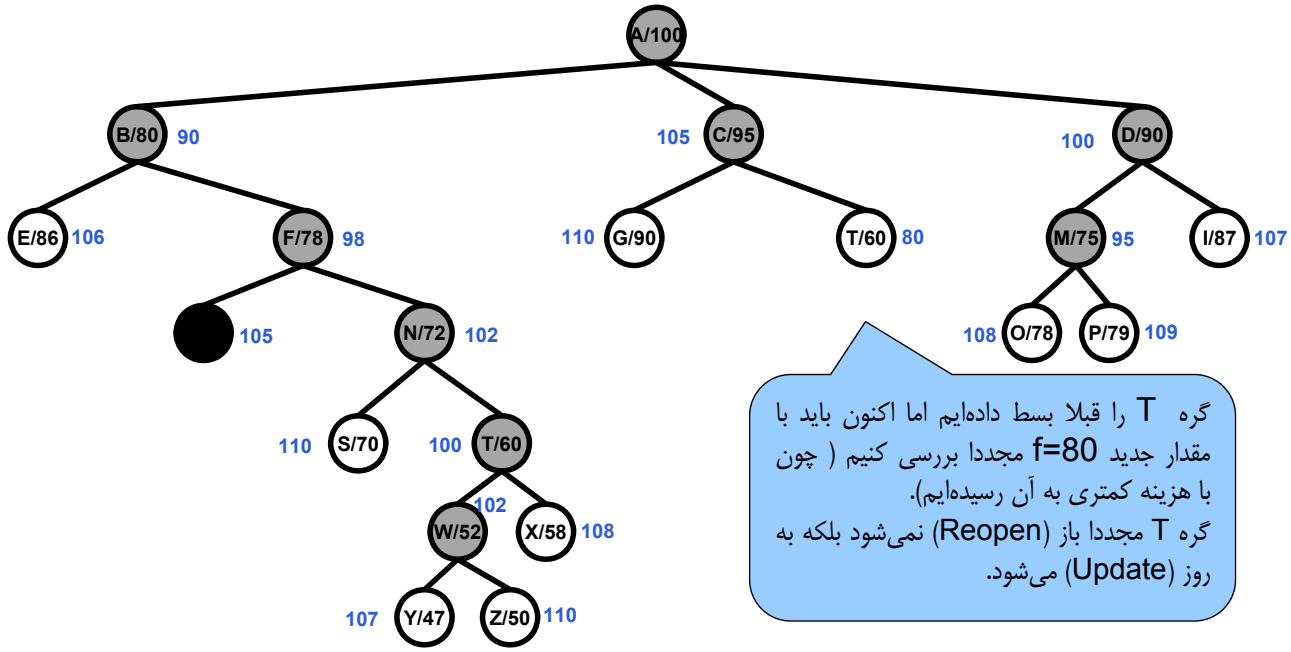
مثال: درخت جستجوی A* در فضای حالت گرافی



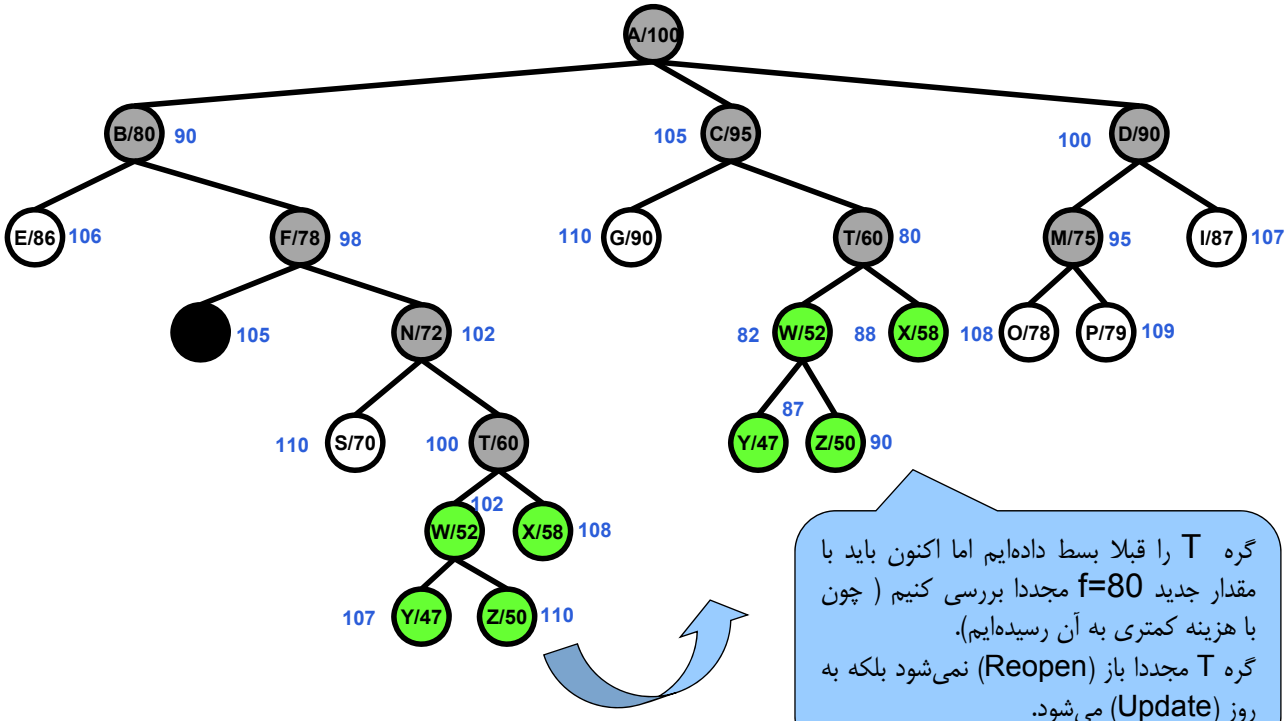
مثال: درخت جستجوی A* در فضای حالت گرافی



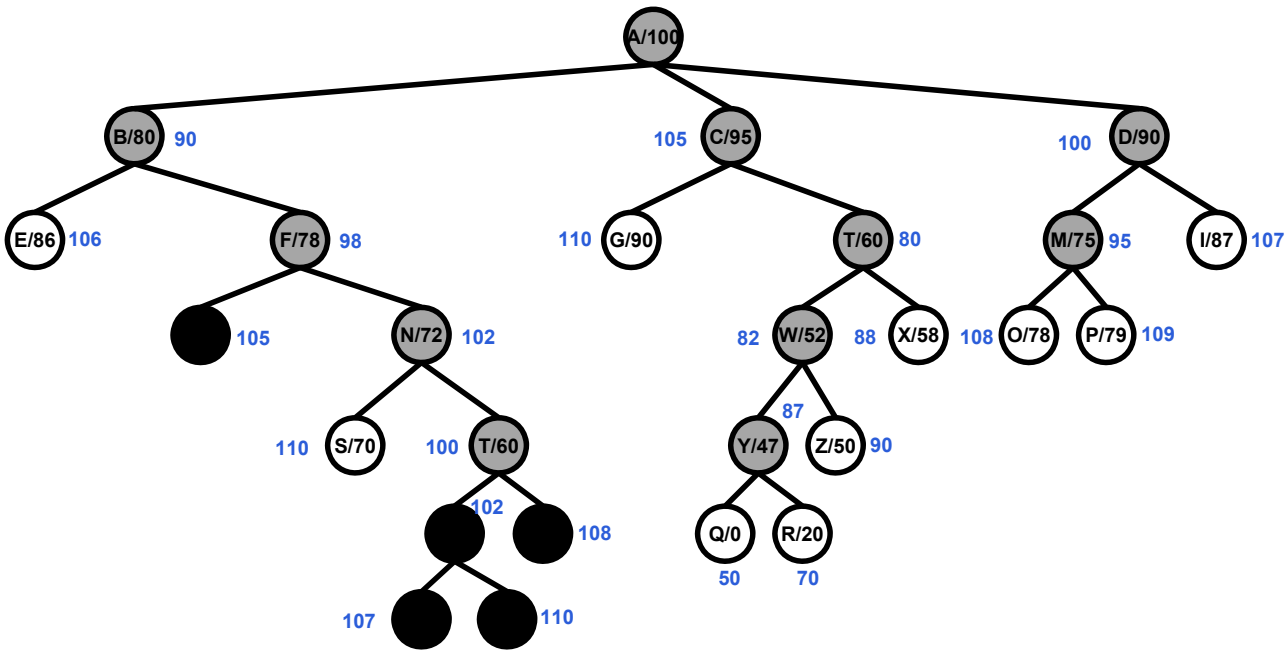
مثال: درخت جستجوی A* در فضای حالت گرافی



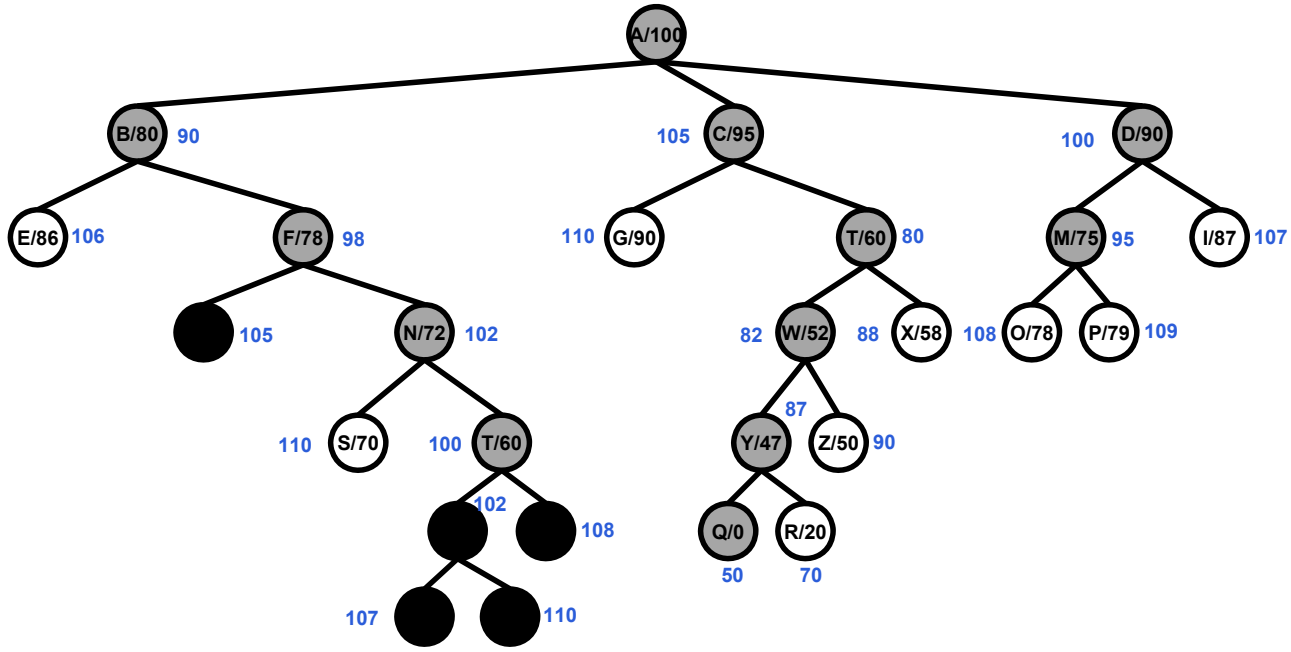
مثال: درخت جستجوی A* در فضای حالت گرافی



مثال: درخت جستجوی A* در فضای حالت گرافی

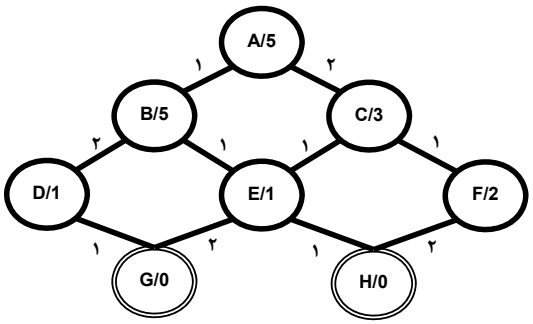


مثال: درخت جستجوی A* در فضای حالت گرافی



هوش مصنوعی ۸۸

- در گراف مقابل اگر در جستجو با روش A* تست هدف یک بار در لحظهٔ تولید و بار دیگر در لحظهٔ بسط صورت گیرد به ترتیب چه مسیری یافت خواهد شد؟ اعداد روی لبه ها هزینهٔ واقعی مسیر، اعداد داخل دایره ها هزینهٔ تخمینی گره تا هدف است. (ترتیب بسط فرزندان هر گره به ترتیب حروف الفباست.)



1. تست در لحظه تولید ACEG در لحظهٔ بسط ACEH
2. تست در لحظه تولید ABEH در لحظهٔ بسط ABEH
3. تست در لحظه تولید ACEH در لحظهٔ بسط ABEH
4. تست در لحظه تولید ACEG در لحظهٔ بسط ABEG

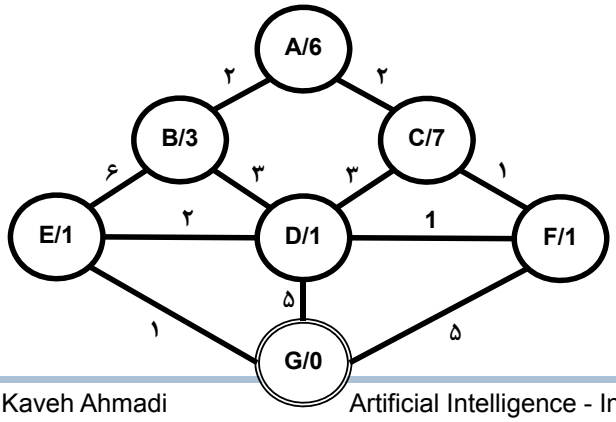
هوش مصنوعی ۸۸

- پاسخ گزینه ۱ – در لحظه بسط همان A* معمولی ولی در لحظه تولید با تولید G به عنوان هدف انتخابش می‌کند.

- نکته:
  - اگر هیوریستیک قابل قبول بود، با یک نگاه می‌توانست گفت که مسیر بهینه تا هدف ABEH است. بنابراین یا گزینه ۲ درست بود یا گزینه ۳. در اینجا ذکر نشده که هیوریستیک قابل قبول است و قابل قبول هم نیست! به این موارد توجه کنید و فریب طراح را نخورید!

هوش مصنوعی ۸۹ – نوبت شماست

- گراف مقابل را با روش A* در دو حالت جستجو نموده‌ایم. حالت اول حالت معمولی جستجوی A* است و حالت دوم که آن را روش کهنه‌گرا می‌نامیم، حالتی است که برای هر گره اولین مسیر رسیدن به آن را به عنوان تنها مسیر رسیدن به گره حفظ می‌کنیم. (گره‌های تکراری در لحظه تولید حذف می‌شوند.) مسیر پاسخ در دو حالت معمولی و کهنه‌گرا به ترتیب کدام خواهند بود؟ اعداد روی لبه‌ها هزینه واقعی مسیر، اعداد داخل دایره‌ها هزینه تخمینی گره تا هدف است. (ترتیب بسط فرزندان هر گره به ترتیب حروف الفباست.)



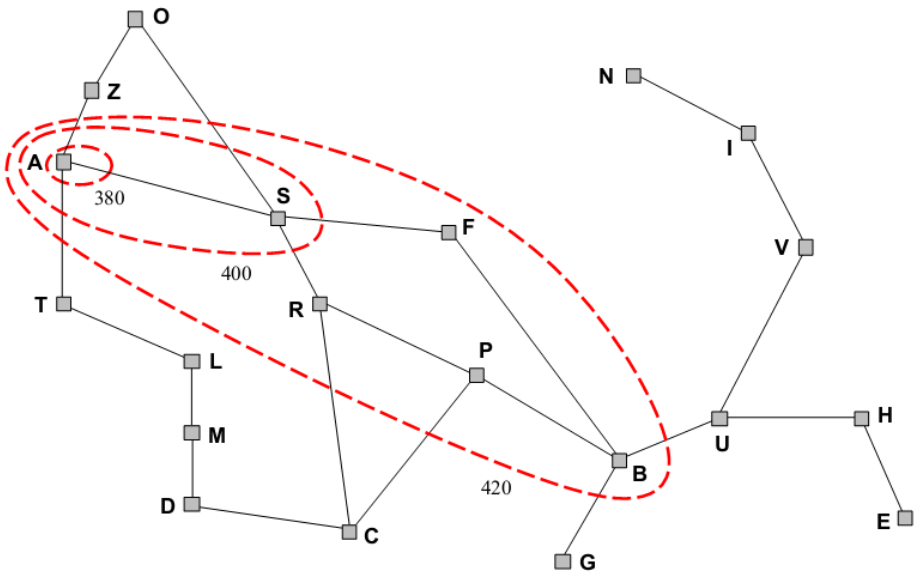
1. ABDEG و ABDG
2. ACFDEG و ABDG
3. ACFDEG و ABDEG
4. ABDG و ACFDEG

هوش مصنوعی ۸۹

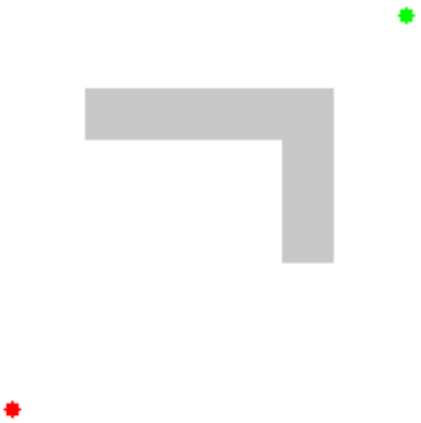
- پاسخ: گزینه ۱

بهینگی A*: کانتورها

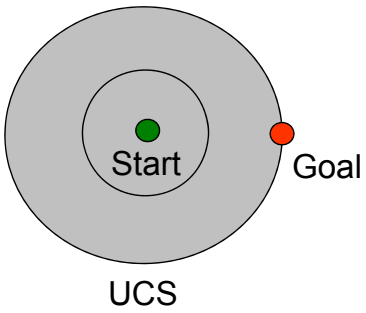
- با یک h صحیح باندها به تدریج به سمت کانتور هدف کشیده می‌شوند و به سمت هدف بهینه باریک‌تر می‌شوند.
  - کانتور i شامل تمام گره‌هایی است که $f = f_i$ و $f_i < f_{i+1}$

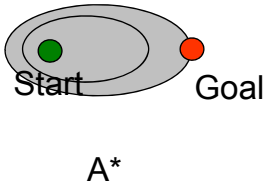


بهینگی A*: کانتورها



کانتورهای A* در برابر کانتورهای UCS





هوش مصنوعی ۸۱

- شرایط قابل پذیرش بودن یک الگوریتم برای یافتن جواب مسئله کدام است؟

۱. $\exists n : h(n) \le h^*(n), g(n) \ge g^*(n)$

۲. $\exists n : h(n) \ge h^*(n), g(n) \ge g^*(n)$

۳. $\forall n : h(n) \le h^*(n), g(n) \ge g^*(n)$

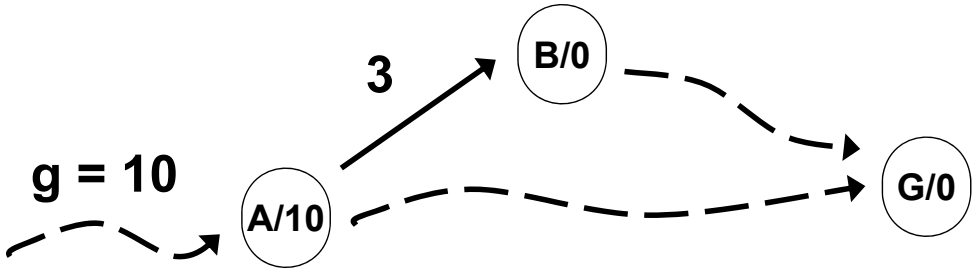
۴. $\forall n : h(n) \le h^*(n), g(n) \le g^*(n)$

هوش مصنوعی ۸۱

- پاسخ: گزینه ۴
  - مطابق لم گفته شده و کانتورهای A*

خاصیت یکنوایی A*

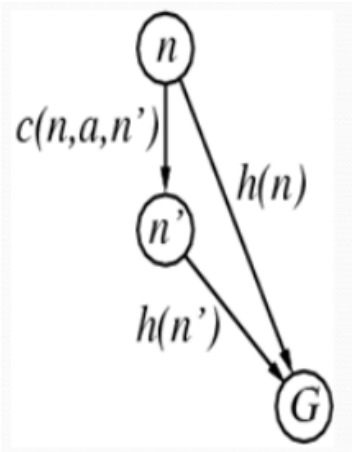
- آیا همیشه **f**ها به ترتیب میزان مقدارشان و از کوچک به برزگ بسط پیدا می‌کنند؟
- آیا ممکن است گره n را بسط دهیم و سپس فرزند n′ آنرا پیدا کنیم که f کمتری داشته باشد؟



- زمانی که گراف فضای مسئله واقعا یک گراف باشد (جستجوی گرافی) برای بهینه بودن جستجو لازم است علاوه برای قابل قبول بودن، تابع اکتشافی یکنوا یا سازگار (consistent, monotone) هم باشد.
  - به این ترتیب کانتورها به شکل گفته بسط پیدا خواهد کرد.

خاصیت یکنوایی A*

- تابع اکتشافی h یکنوا است اگر به ازای هر گره n و مابعد بلافصل n′ و عمل a داشته باشیم:
  - $h(n) \leq c(n, a, n') + h(n')$



- اگر h سازگار باشد، زمان بسط یک گره n در A* مطمئن خواهیم بود که $g(n) = g^*(n)$
- یک تابع اکتشاف یکنوا حتما قابل قبول نیز خواهد بود.

هوش مصنوعی ۹۰

- فرض کنید فضای جستجویی دارای پنج گره A، B، C، D و E باشد. جدول زیر فواصل واقعی این گره‌ها را از هم نشان می‌دهد. (وجود عدد در هر خانه جدول نشان دهنده این است که از گره مربوط به سطر به سمت گره مربوط به ستون مسیری به طول عدد وجود دارد). اگر گره A گره شروع، گره E گره هدف و تابع h تابع مکاشفه‌ای تخمین فاصله گره تا هدف باشد، کدام یک از گزینه‌های زیر صحیح است؟

	A	B	C	D	E
A		10	8	2	
B	10		2		2
C	8			2	6
D	2		2		9
E		2	6	9	

1. اگر h(B)=3، h(C)=3 و h(D)=8 آنگاه تابع h یک تابع یکنواخت (monotonic) است ولی قابل قبول (admissible) نیست
2. اگر h(B)=1، h(C)=5 و h(D)=8 آنگاه تابع h یک تابع یکنواخت (monotonic) و قابل قبول (admissible) است
3. اگر h(B)=1، h(C)=5 و h(D)=8 آنگاه تابع h یک تابع یکنواخت (monotonic) نیست ولی قابل قبول (admissible) است
4. اگر h(B)=3، h(C)=6 و h(D)=9 آنگاه تابع h یک تابع یکنواخت (monotonic) و قابل قبول (admissible) است

هوش مصنوعی ۹۰

- از آنجا که E گره هدف است، هزینه بهینه رسیدن به هدف با توجه به جدول به شکل زیر است:
  - $h^*(B) = 2,\ h^*(C) = 6,\ h^*(D) = 10$
- گزینه ۱ غلط است چون یک تابع هیوریستیک یکنوا لزوما قابل قبول نیز هست.
- گزینه ۴ غلط است چون $H(B) > H^*(B)$ است بنابراین تابع هیوریستیک قابل قبول نیست.
- تابع هیوریستیک گزینه‌های ۲ و ۳ قابل قبولند اما این تابع یکنواخت نیست زیرا در یک تابع یکنواخت باید به ازای هر گره n و مابعد بلافصل n' داشته باشیم:
  - $h(n) \leq c(n, a, n') + h(n')$

این شرط برای گره D برقرار نیست بنابراین گزینه ۳ صحیح است.

- $h(D) \geq c(D \rightarrow C) + h(C) \quad : \quad 8 \geq 2 + 5$

ویژگی‌های جستجوی A*

- لم:
  - در هر مرحله قبل از اتمام الگوریتم A*، همواره گره‌ی همانند n‘ در لیست frontier با خواص سه گانه زیر قرار دارد (مشروط به قابل قبول بودن تابع اکتشاف):
    - n‘ روی مسیر بهینه تا هدف قرار دارد: الگوریتم مسیر بهینه را گم نمی‌کند.
    - A* مسیر بهینه تا n‘ را یافته است: وقتی به n‘ می‌رسیم، کوتاه‌ترین مسیر را تا آن یافته‌ایم.
    - f(n')<C*: به مسیر بهینه نرسیده‌ایم (از آن رد نشده‌ایم) و n‘ پیش از گره هدف بسط پیدا می‌کند.

ویژگی‌های جستجوی A*

- اگر C* هزینه مسیر پاسخ باشد و با فرض یکنوا بودن تابع اکتشاف:
  - A* تمامی گره‌های با f(n) < C* را بسط می‌دهد.
  - همچنین ممکن است برخی گره‌های روی کانتور هدف (f(n) = C*) را قبل از انتخاب حالت هدف توسعه دهد.
- شهودا مشخص است که اولین پاسخ یافت شده **بهینه است**.
  - گره‌های هدف در کانتورهای بعدی، f بالاتری دارند.
- شهودا مشخص است که **کامل است**. (در گراف‌های با فاکتور انشعاب محدود)
  - با افزودن باندهای افزایش یابنده f باید به باندی که f برابر با هزینه مسیر به گره هدف است برسیم.

ویژگی‌های جستجوی A*

- A* هیچ گره‌ای با $f(n) > C^*$ را بسط نمی‌دهد. هرجای درخت جستجو که باشند.
  - آنها را هرس می‌کند!
- اگر الگوریتمی تمام گره‌های با $f(n) < C^*$ را بسط ندهد ممکن است با خطر از دست دادن پاسخ بهینه مواجه شود.
  - A* تقریبا فقط گره‌هایی را بسط می‌دهد که باید بسط دهد (با فرض وجود تابع اکتشاف کارا)

ویژگی‌های جستجوی A*

- پیچیدگی حافظه $O(b^d)$
- تعداد گره‌های درون کانتور هدف نسبت به طول راه‌حل نمایی است.
  - مگر آنکه خطا در تابع اکتشافی از لگاریتم هزینه مسیر واقعی سریع‌تر رشد نکند.
    - وابستگی به تابع هیوریستیک کاملا مشهود است.

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

هوش مصنوعی ۹۰

- چند مورد از موارد زیر درست می‌باشد؟
- ۱) اگر برای مسئله خاصی حداقل یک راه حل کامل آگاهانه وجود داشته باشد، آنگاه حداقل یک راه حل کامل ناآگاهانه نیز برای آن مسئله وجود دارد.
- ۲) اگر برای مسئله خاصی یک راه حل کامل ناآگاهانه وجود داشته باشد، آنگاه حداقل یک راه حل کامل آگاهانه نیز برای آن مسئله وجود دارد.
- ۳) اگر برای مسئله خاصی هیچ راه حل ناآگاهانه وجود نداشته باشد، آنگاه هیچ راه حل کامل آگاهانه‌ای نیز برای آن مسئله وجود ندارد.
- ۴) اگر برای مسئله خاصی هیچ راه حل کامل ناآگاهانه‌ای وجود نداشته باشد، آنگاه هیچ راه حل کامل آگاهانه‌ای نیز برای آن مسئله وجود ندارد.

۱. یک مورد ۲. دو مورد ۳. سه مورد ۴. چهار مورد

هوش مصنوعی ۹۰

- گزینه ۴ صحیح است.

هوش مصنوعی ۸۱

- روش جستجوی A* تحت چه شرایطی یافتن پاسخ بهینه را تضمین می‌کند؟

1. اصولا روش‌های ابتکاری از جمله A* قادر به یافتن پاسخ بهینه نیستند.
2. شرایط لازم برای اینکه روش A* یافتن پاسخ بهینه را تضمین کند به دامنه مسأله بستگی دارد.
3. در صورتی که تابع ابتکاری مورد استفاده، فاصله وضعیت‌های مختلف تا وضعیت هدف را هرگز بیشتر از مقدار واقعی تخمین نزند.
4. در صورتی که تابع ابتکاری مورد استفاده، فاصله وضعیت‌های مختلف تا وضعیت هدف را حداکثر به مقدار کوچک دلتا و بیشتر از مقدار واقعی تخمین بزند.

هوش مصنوعی ۸۱

- پاسخ: گزینه ۳

هوش مصنوعی ۸۳

- می‌خواهیم با استفاده از روش جستجوی A* پاسخ بهینه مساله‌ای را بیابیم. با فرض اینکه هر سه تابع ابتکاری h_1، h_2 و h_3 برای این منظور قابل استفاده باشند، کدام یک از توابع ترکیبی زیر نیز برای یافتن حل بهینه مسئله قابل استفاده خواهد بود؟

۱. $h_1 + h_2 + h_3$

۲. $\frac{h_1 + h_2 + h_3}{3}$

۳. $h_1 \times h_2 \times h_3$

۴. $\sqrt{h_1 \times h_2 \times h_3}$

هوش مصنوعی ۸۳

- پاسخ: گزینه ۲

آی تی ۸۵

- برای حل مسأله حرکت یک مسافر از شهری به شهر دیگر، الگوریتم جستجوی A* انتخاب شده است. اگر تابع هیورستیک فاصله مانهاتان برای این الگوریتم انتخاب شود، آنگاه:

1. این تابع هیورستیک پذیرفتنی است اما همنوا نیست، زیرا قاعدهٔ مثلث را نقض می کند.
2. این تابع هیورستیک پذیرفتنی و مقبول است اما همنوا نیست، زیرا همواره هزینه را کمتر از مقدار واقعی آن تخمین می زند.
3. این تابع هیورستیک پذیرفتنی و مقبول نیست، زیرا ممکن است هزینه را بیش از مقدار واقعی آن تخمین بزند.
4. این تابع همنوا است، اما پذیرفتنی و مقبول نیست، زیرا قاعدهٔ مثلث را نقض می‌کند.

آی تی ۸۵

- پاسخ: گزینه ۳

آی تی ۸۸

- در جستجوی روی گراف، کدام خصوصیت تابع مکاشفه‌ای $h(n)$ در معادله‌ی $f(n)=h(n)+g(n)$ قابل قبول بودن آن را تضمین می‌کند؟

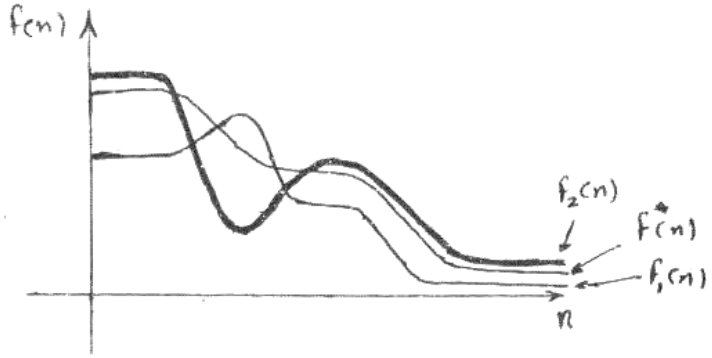
1. $h(n)$ همواره از $g(n)$ کوچکتر باشد.
2. $h(n)$ تابعی یکنواخت باشد.
3. $h(n)$ در گره‌های متوالی غیرنزولی باشد.
4. $h(n)$ تابعی غیرصفر و همواره از $g(n)$ بزرگتر باشد.

آی تی ۸۸

- پاسخ: گزینه ۲
  - یک تابع اکتشافی یکنوا لزوما قابل قبول بودن آنرا نیز تضمین می‌کند.

آی تی ۹۰

- در جستجوی A* در صورت استفاده از کدام تابع مکاشفه‌ای تضمین پیدا کردن جواب بهینه وجود دارد؟



1. $f_1(n)$
2. $f_2(n)$
3. $\frac{f_1(n)+f_2(n)}{2}$
4. $f_1(n) + f_2(n)$

آی تی ۹۰

- پاسخ: گزینه ۳
  - f_1 در برخی نقاط تخمینی بیشتر از f^* دارد اما در آن نقاط f_2 تخمین به مراتب کمتری دارد و برعکس. به نظر می‌رسد میانگین مقادیر این دو تابع همواره و در همه نقاط کمتر از مقدار f^* در آن نقطه باشد.

- نکته:
  - هیورستیک‌ها می‌توانند قابلیت تصحیح داشته باشند به این معنی که در گره‌هایی که خاصیت یکنوایی تابع هیورستیک نقض می‌شود از مقدار تابع در گره قبلی‌اش استفاده کنیم.

معیارهای فاصله عمدتا مورد استفاده

- Minkowski metric (L_k norm)

$$\|x-y\|_k = \left(\sum_{i=1}^{D}|x_i-y_i|^k\right)^{1/k}$$

  - The choice of an appropriate value of k depends on the amount of emphasis that you would like to give to the larger differences between dimensions

- Manhattan or city-block distance (L_1 norm)

$$\|x-y\|_{c-b} = \sum_{i=1}^{D}|x_i-y_i|$$

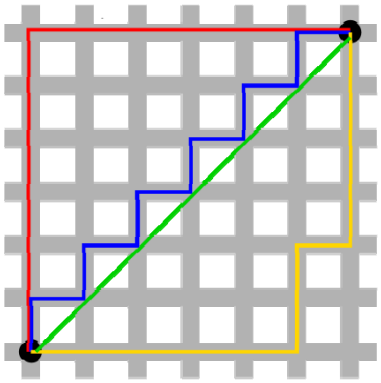
  - When used with binary vectors, the L1 norm is known as the Hamming distance

- Euclidean norm (L_2 norm)

$$\|x-y\|_e = \left(\sum_{i=1}^{D}|x_i-y_i|^2\right)^{1/2}$$

- Chebyshev distance (L_∞ norm)

$$\|x-y\|_c = \max_{1\le i\le D}|x_i-y_i|$$



خط سبز رنگ نشان‌دهنده فاصله اقلیدسی و خطهای قرمز، آبی و رزد نشان‌دهنده فاصله منتهتن است (تصویر از ویکیپدیا).
در فاصله چیبشف می‌توان همانند فاصله منهتن حرکت روی لبه‌ها را داشت + حرکت‌های قطری (همانند حرکت مهره‌ی شاه در صفحه شطرنج)

اثر تابع اکتشاف در کارایی

- مثال معمای ۸

7	2	4
5		6
8	3	1

حالت فعلی

1	2	3
8		4
7	6	5

حالت هدف

- میانگِین هزینه حل تقریبا ۲۲ مرحله و فاکتور انشعاب در حدود ۳ است.
- جست و جوی جامع تا عمق ۲۲ تقریبا ۳^۲۲ حالت دارد.
- با انتخاب یک تابع اکتشافی مناسب می‌توان مراحل جستجو را کاهش داد.

اثر تابع اکتشاف در کارایی

- مثال معمای ۸
  - $h_1(n) =$ تعداد خانه‌هایی که در محل خود قرار ندارند
  - $h_2(n) =$ مجموع فاصله خانه‌ها از محل‌های نهایی

7	2	4
5		6
8	3	1

حالت فعلی

1	2	3
8		4
7	6	5

حالت هدف

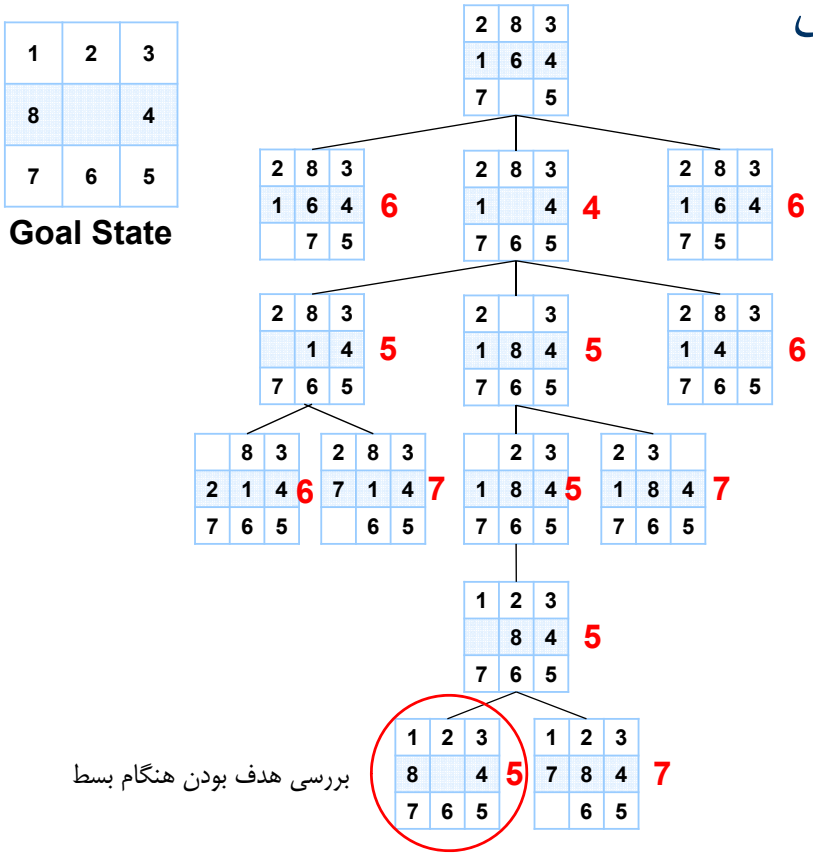
$$h_1(n) = 7$$
$$h_2(n) = 4 + 0 + 3 + 1 + 3 + 2 + 2 + 1 = 16$$

اثر تابع اکتشاف در کارایی

- مثال معمای ۸
  - h_1 یک اکتشاف قابل قبول است، زیرا هر کاشی که در جای نامناسب قرار دارد، حداقل یکبار باید جابجا شود
    - مسئله خیلی راحت (relax) شده است. و محدودیت‌های مسئله در نظر گرفته نشده است (فرض شده مهره‌ها با یک پرش می‌توانند به محل اصلی خود بروند).
  - h_2 قابل قبول است، زیرا هر جابجایی که می‌تواند انجام گیرد، به اندازه یک مرحله به هدف نزدیک می‌شود.
    - در h_2 چون کاشی‌ها نمی‌توانند در امتداد قطر جابجا شوند، فاصله‌ای که محاسبه می‌کنیم مجموع فواصل افقی و عمودی است. این فاصله را فاصله بلوک شهر (City block distance) یا فاصله منهتن (manhatan distance) می‌نامند.
    - مسئله باز راحت شده اما محدودیت‌های بیشتری از مسئله را در نظر گرفته است.
  - هیچ کدام از این برآوردها، هزینه واقعی راه حل نیست

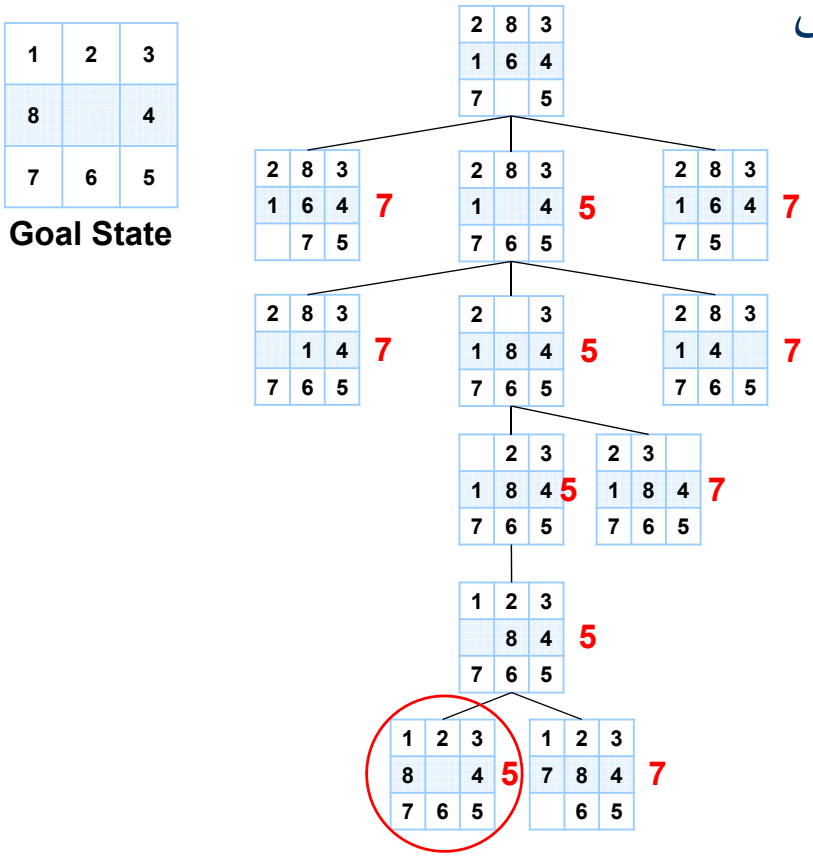
اثر تابع اکتشاف در کارایی

- جستجوی A* با h1



اثر تابع اکتشاف در کارایی

- جستجوی A* با h2



تسلط (Dominance)

- اگر برای دو هیوریستیک قابل قبول و به ازای همه گره‌های n داشته باشیم $h_2(n) > h_1(n)$ آنگاه می‌توان گفت h_2 بر h_1 تسلط دارد و برای استفاده در جستجو بهتر است.
  - تعداد گره هایی که A^* با بکارگیری h_2 بسط می‌دهد، هرگز بیشتر از A^* با بکارگیری h_1 نیست.
    - هر گره‌ای که بوسیله‌ی A^* و h_2 بسط داده می‌شود، با h_1 نیز بسط داده می‌شود و h_1 ممکن است منجر به بسط گره‌های دیگری نیز شود.
  - غالب بودن مستقیما به کارایی ترجمه می‌شود.
  - همیشه بهتر است از تابع اکتشافی با مقادیر بزرگ (و قابل قبول) استفاده کرد، به شرطی که زمان محاسبه اکتشاف، خیلی بزرگ نباشد.

تسلط (Dominance)

- مثال: تعداد گره‌های بسط داده شده برای رسیدن به هدف اگر هدف در عمق‌های ۱۲ یا ۲۴ باشد:
  - d = 12
    - $A^*(h_1)$ = 227 nodes
    - $A^*(h_2)$ = 73 nodes
  - d = 24
    - $A^*(h_1)$ = 39,135 nodes
    - $A^*(h_2)$ = 1,641 nodes

هوش مصنوعی ۸۴

- فرض کنید می خواهیم الگوریتم A^* را برای جستجو بکار ببریم و سه هیورستیک h_1,h_2,h_3 موجودند که همگی قابل پذیرش هستند و برای تمام وضعیت‌ها داریم $h_2>h_1>h_3$. کدام یک از عبارات زیر درست است؟

1. مسیر بهینه فقط از بکارگیری h_2 حاصل می‌شود اما h_1 از h_3 بهتر است.
2. با هرکدام از هیورستیک‌های فوق مسیر بهینه حاصل می‌شود، اما h_2 مسیر ارزانتر را پیدا می‌کند.
3. با هرکدام از هیورستیک‌های فوق مسیر بهینه حاصل می‌شود، اما h_2 آن را با کمترین بسط پیدا می‌کند.
4. راه حل‌های حاصل از این هیورستیک‌ها از نظر بهینگی به همان ترتیب $h_2>h_1>h_3$ هستند.

هوش مصنوعی ۸۴

- پاسخ: گزینه ۳

آی تی ۸۹

- کدام عبارت درمورد جستجوی A* نادرست است؟

1. در جستجوی A* باید h = h* باشد.
2. جستجوی A* کامل و بهینه است.
3. در جستجوی A* هزینه تا گره جاری و تخمین از گره جاری تا هدف باید مشخص باشد.
4. درصورتی که h = h* باشد، پیچیدگی زمانی برابر با حاصلضرب عمق جواب در تعداد متوسط شاخه‌ها است.

آی تی ۸۹

- گزینه ۱

فاکتور انشعاب موثر (Effective Branching Factor) – b*

- یک راه برای تشخیص کیفیت کشف‌کنندگی، فاکتور انشعاب مؤثر b^* است.
- اگر مجموع تعداد گره‌های بسط داده شده توسط A^* برای یک مسئله ویژه N باشد و عمق راه حل d، آنگاه b^* فاکتور انشعابی است که یک درخت یکنواخت با عمق d خواهد داشت تا گره‌های N را نگه دارد.

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

- b^* نشان می‌دهد که روش اکتشافی پیشنهادی تا چه حد قادر است میزان فاکتور انشعاب را کاهش دهد. هرچه میزان b^* به یک نزدیک‌تر باشد، تابع اکتشافی به h^* نزدیک‌تر است

فاکتور انشعاب موثر (Effective Branching Factor) – b*

	Search Cost		Effective Branching Factor	
Depth	$A^*(h_1)$	$A^*(h_2)$	$A^*(h_1)$	$A^*(h_2)$
2	6	6	1.79	1.79
4	13	12	1.48	1.45
6	20	18	1.34	1.30
8	39	25	1.33	1.24
10	93	39	1.38	1.22
12	227	73	1.42	1.24
14	539	113	1.44	1.23
16	1301	211	1.45	1.25
18	3056	363	1.46	1.26
20	7276	676	1.47	1.27
22	18094	1219	1.48	1.28
24	39135	1641	1.48	1.26

مقایسه هزینه جستجو و فاکتور انشعاب موثر برای الگوریتم A^* هنگام استفاده از توابع اکتشاف h_1 و h_2. داده‌ها میانگین مقادیر ۱۰۰ اجرای متفاوت روی مسئله معمای ۸ برای پاسخ‌های با طول‌های مختلف است.

بدیهیات

- اگر در A* تابع اکتشافی قابل قبول نباشد، الگوریتم همان اول بهترین حریصانه خواهد بود.
- در اول بهترین حریصانه اگر h(n)=g(n) باشد، الگوریتم همان جستجو با هزینه یکسان خواهد بود.
- در جستجو با هزینه یکسان اگر هزینه همه لبه‌ها یکسان باشد، الگوریتم همان اول سطح خواهد بود.

هوش مصنوعی ۸۴

- در مسئله حرکت در راه پیچ در پیچ (Maze) فرض کنید عامل بتواند در یک جدول n*m که بعضی از خانه‌ها سیاه (غیر قابل عبور) است به ۴ جهت افقی و عمودی (نه قطری) حرکت کند. در این مساله برای یافتن کوتاه‌ترین مسیر کدام یک از جملات زیر صحیح نیست؟

1. جستجوی A* سریعتر از عرض اول به جواب می‌رسد.
2. جستجوی A* تعداد گره کمتری نسبت به جستجوی عمق اول بسط می‌دهد.
3. جستجوی عمق اول سریعتر از جستجوی عرض اول به جواب می‌رسد.
4. جستجوی عرض اول جواب بهتری نسبت به جستجوی عمق اول می‌یابد.

هوش مصنوعی ۸۴

- گزینه ۳ صحیح است.
  - چون هزینه ها ثابت اول سطح نیز بهینه است ولی A* سریعتر است
  - اول عمق لزوما بهینه نیست (در یک شاخه تا بینهایت پیش رود)