بسم الله الرحمن الرحیم

دانشگاه فنی و حرفه‌ای

هوش مصنوعی(فصل چهارم)

میرمحمود ابوالحسنی

جستجوهای آگاهانه

فهرست مطالب

- استراتژی‌های آگاهانه
  - جستجوی حریصانه
  - جستجوی A*
- کیفیت توابع هیوریستیک
- ابداع توابع هیوریستیک
- جستجوی محلی
  - تپه نوردی
  - SA
  - پرتوی محلی
  - ژنتیک

جستجوهای آگاهانه

- جستجوی ناآگاهانه.
  - تنها از اطلاعات موجود در تعریف مسئله استفاده می‌کند.
  - تنها می‌تواند حالت‌های هدف را از حالت‌های غیرهدف تشخیص دهد.
- جستجوی آگاهانه.
  - علاوه بر اطلاعات موجود در تعریف مسئله، از اطلاعات خاص مسئله بهره می‌برد.
  - بین حالت‌های غیرهدف (بسته به میزان نزدیک بودن آنها به هدف) تمایز قایل می‌شود.

جستجوهای آگاهانه

یاد آوری شبه کد جستجوی درختی

```
function TREE-SEARCH(problem , fringe) return a solution or failure
    fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
    loop do
        if EMPTY?(fringe) then return failure
        node ← REMOVE-FIRST(fringe)
        if GOAL-TEST[problem] applied to STATE[node] succeeds
                then return SOLUTION(node)
        fringe ← INSERT-ALL(EXPAND(node, problem), fringe)
```

جستجوهای آگاهانه

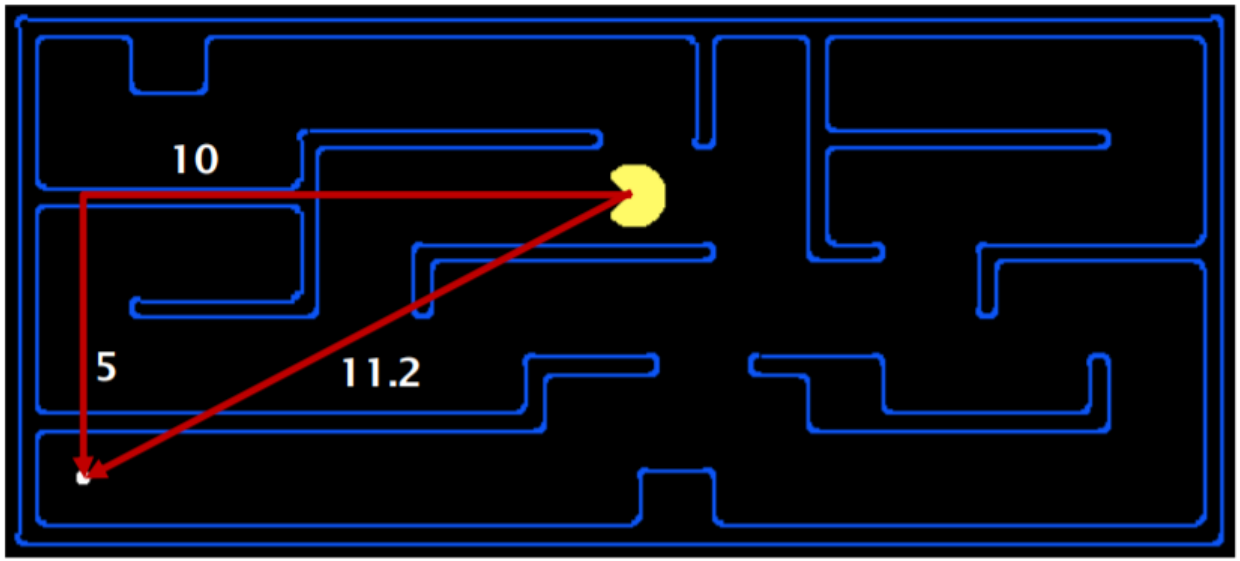
جستجوی سطحی

- روش کلی برای جستجوی آگاهانه
- جستجوی سطحی: گره برای بسط دادن بر اساس تابع ارزیابی *f(n)* انتخاب می‌شود.
- ایده: تابع ارزیابی فاصله تا هدف را اندازه‌گیری می‌کند. یعنی گره‌ایی را برای بسط دادن انتخاب می‌کند که به نظر بهترین باشد.
- پیاده سازی: *fringe* یک صف مرتب بر اساس *f(n)*

یعنی گره ای که کمترین مقدار f(n) را دارد در اول صف قرار می گیرد.

جستجوهای آگاهانه

- هیوریستیک.
  - یک تابع که فاصله هر حالت را تا هدف تخمین می‌زند.
  - وابسته به مسئله است.
  - مثال: فاصله مانهاتانی و فاصله اقلیدسی در مسایل مسیریابی



جستجوهای آگاهانه

- ایده.
  - هر گره‌ای که تولید می‌شود را ارزیابی کن. [تابع ارزیابی]
  - هر بار بهترین گره گسترش نیافته را انتخاب و گسترش بده.

- پیاده‌سازی. با استفاده از یک صف اولویت که در آن گره‌ها بر اساس میزان مطلوب بودنشان مرتب شده‌اند.

- انواع خاص از جستجوی اول بهترین.
  - جستجوی حریصانه
  - جستجوی A*

تابع ارزیابی

$$f(n) = h(n)$$

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

جستجوهای آگاهانه

جستجوی حریصانه

- تابع ارزیابی.

$$f(n) = h(n)$$

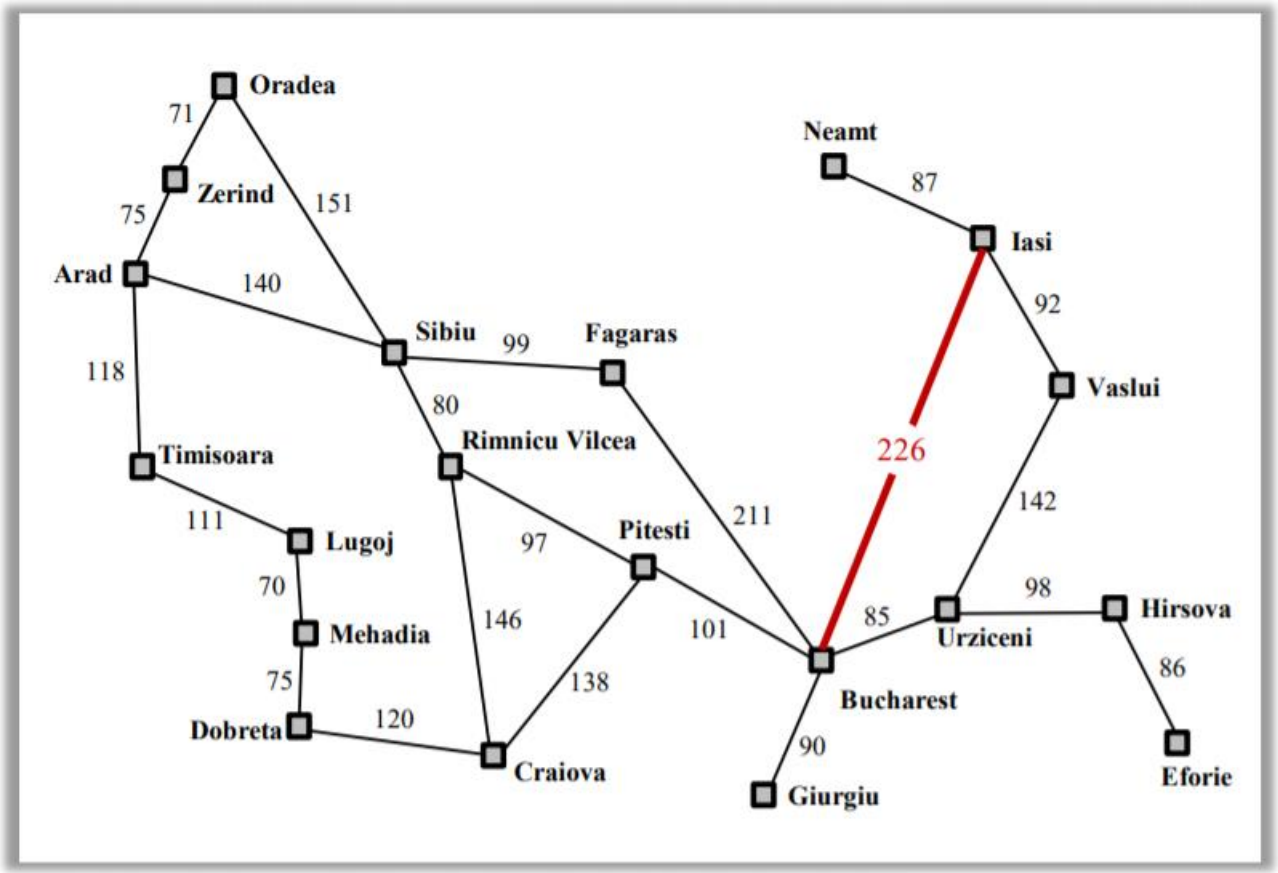
تابع ارزیابی — تابع هیوریستیک

- تابع هیوریستیک.
  - $h(n)$ = فاصله تخمینی از گره n تا هدف.

- جستجوی حریصانه.
  - اول گره‌ای را گسترش بده که به نظر می‌رسد به هدف نزدیک‌تر است.

جستجوهای آگاهانه

مساله رومانی (آراد - بخارست)



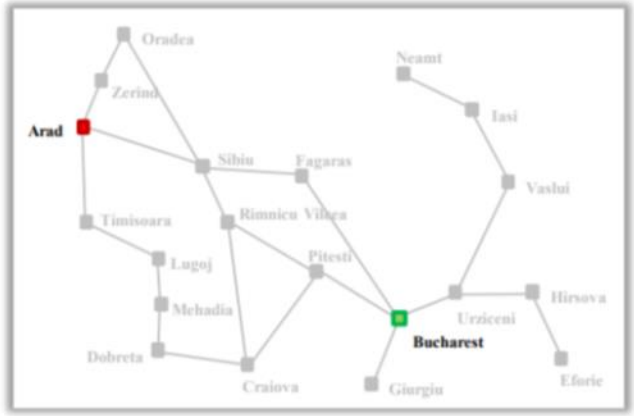
فاصله مستقیم تا بخارست	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

جدول فاصله مستقیم هوایی بین شهرها، اطلاعات اضافی برصورت مساله است که در جستجوی آگاهانه به آن نیاز داریم.

جستجوهای آگاهانه

مساله رومانی (آراد – بخارست)

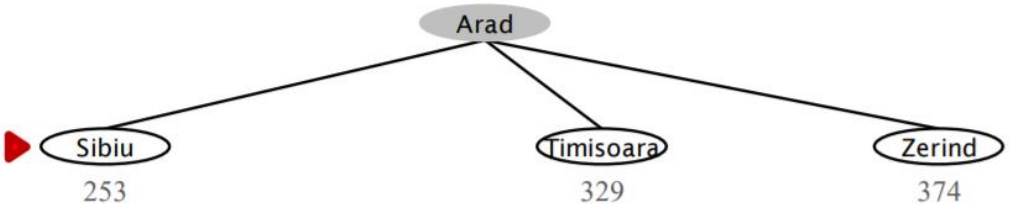
- جستجوی حریصانه. گره‌ای را گسترش بده که به نظر می‌رسد به هدف نزدیک‌تر است.

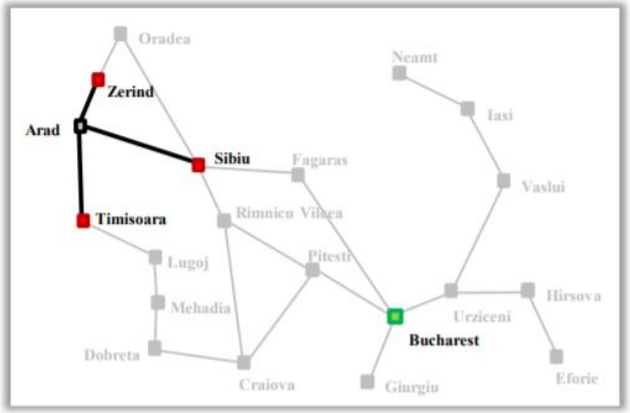



```
frontier: [ Arad (366) ]
```

جستجوهای آگاهانه

- جستجوی حریصانه. گره‌ای را گسترش بده که به نظر می‌رسد به هدف نزدیک‌تر است.



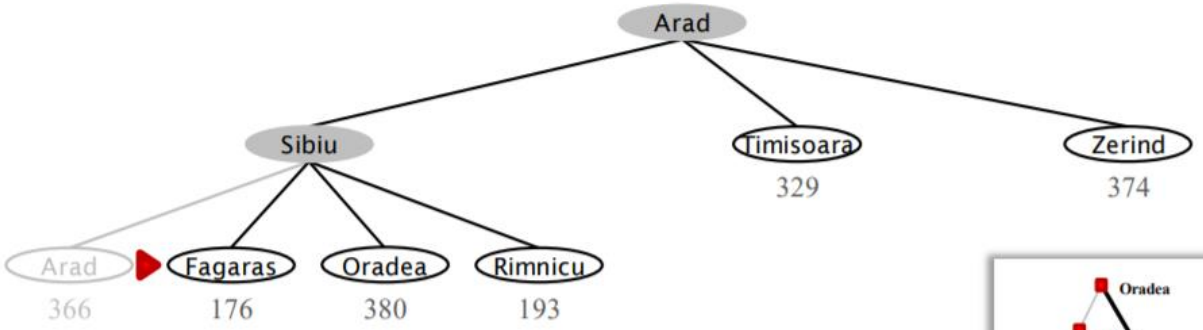


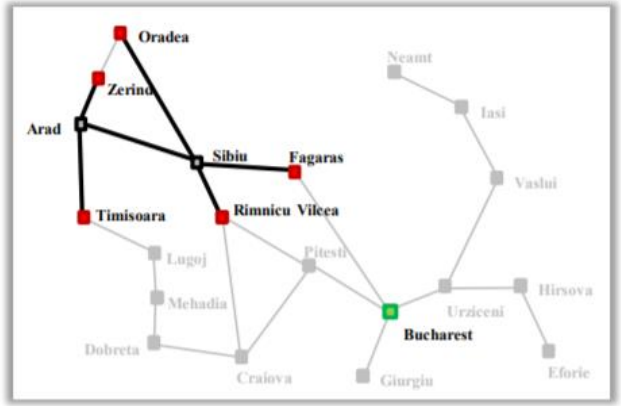

```
frontier: [ Sibiu (253), Timisoara (329), Zerind (374) ]
```

Firing اکنون ۳ عضو دارد که باید بر اساس **f(n)** باید به ترتیب به صف اضافه شوند

جستجوهای آگاهانه

- جستجوی حریصانه. گره‌ای را گسترش بده که به نظر می‌رسد به هدف نزدیک‌تر است.



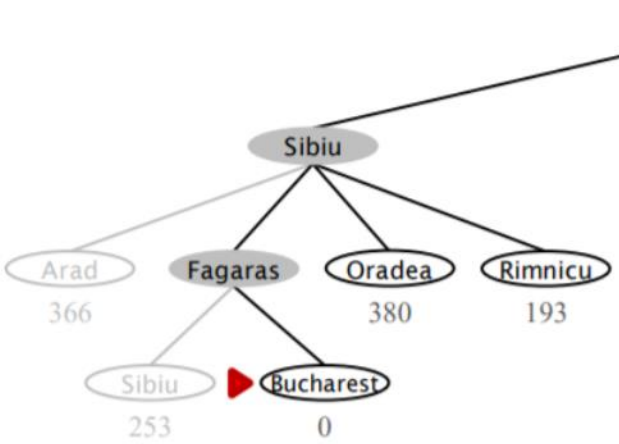


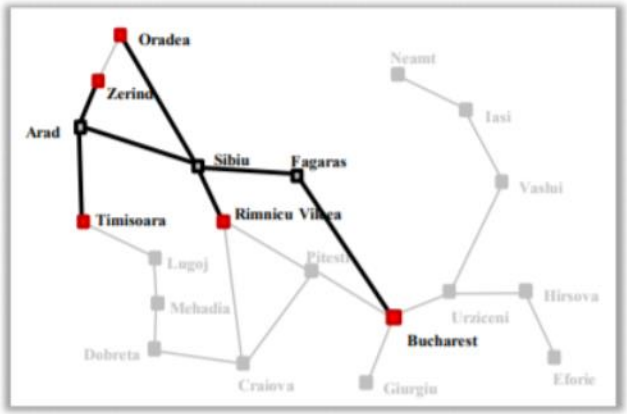

```
frontier: [ Fagaras (176), Rimnicu (193), Timisoara (329),
            Zerind (374), Oradea (380) ]
```

چون f(n) برای گره fagaras از همه کمتر است پس برای بسط دادن انتخاب می شود.

جستجوهای آگاهانه

- جستجوی حریصانه. گره‌ای را گسترش بده که به نظر می‌رسد به هدف نزدیک‌تر است.

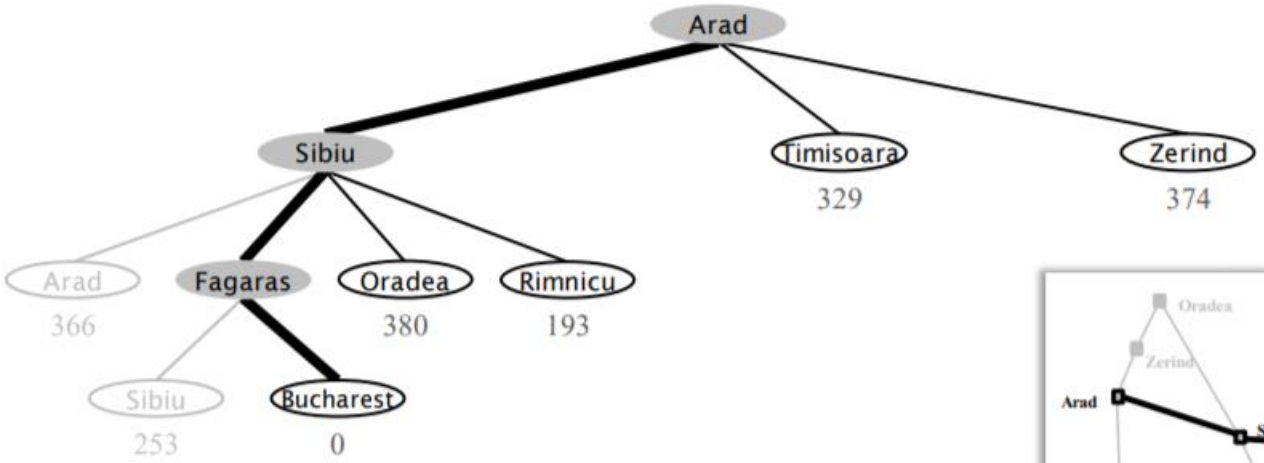


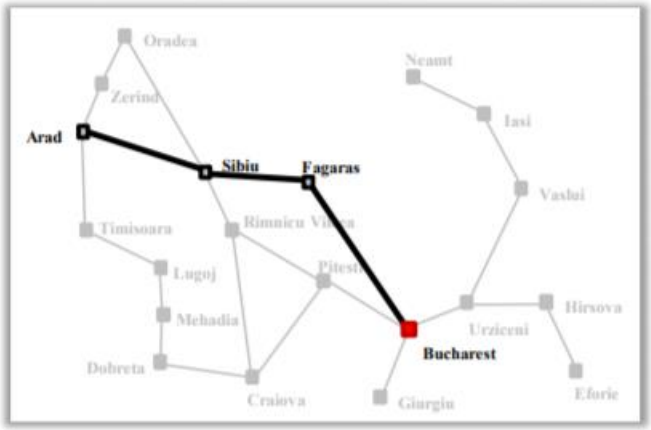



```
frontier: [ Bucharest (0), Rimnicu (193), Timisoara (329),
            Zerind (374), Oradea (380) ]
```

جستجوهای آگاهانه

- جستجوی حریصانه. گره‌ای را گسترش بده که به نظر می‌رسد به هدف نزدیک‌تر است.





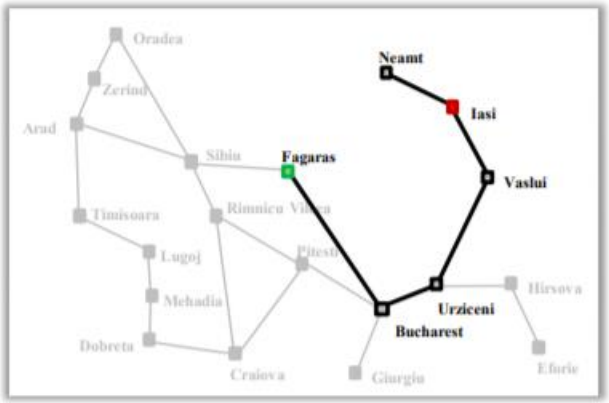
Firing اکنون بر اساس **f(n)** باید به ترتیب صف مرتب شود. و چون **f(n)** گره **Bucharest** مساوی صفر است پس برای بسط دادن انتخاب می‌شود. ولی چون می‌بینیم هدف است پس جستجو متوقف می‌شود.

پس جواب: **Arad-Sibiu-Fagaras- Bucharest**

مجموع مقادیر **f(n)** آنها برابر هزینه کل مسیر است.

جستجوهای آگاهانه

- کامل؟ **خیر** [ممکن است در یک حلقه‌ی بی نهایت گیر کند]
  - مثال: جستجو از **Iasi --> Fagaras**



- بهینه؟ **خیر** [مثال رومانی]

- پیچیدگی زمانی؟ **نمایی** [مانند جستجوی عمقی]

$$b^1 + b^2 + b^3 + \cdots + b^m \in O(b^m)$$

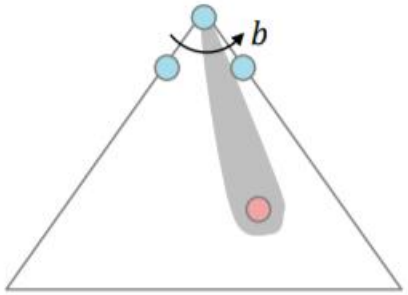
- پیچیدگی حافظه؟ **نمایی**

$$O(b^m)$$

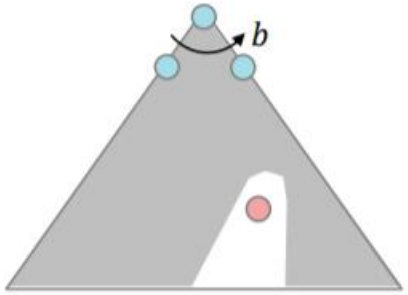
نکته: تمام الگوریتم هایی که کامل نیستند مرتبه زمانی آنها بدترین حالت ممکن است.

جستجوهای آگاهانه

- جستجوی حریصانه. گره‌ای را گسترش بده که به نظر می‌رسد به هدف نزدیک‌تر است.



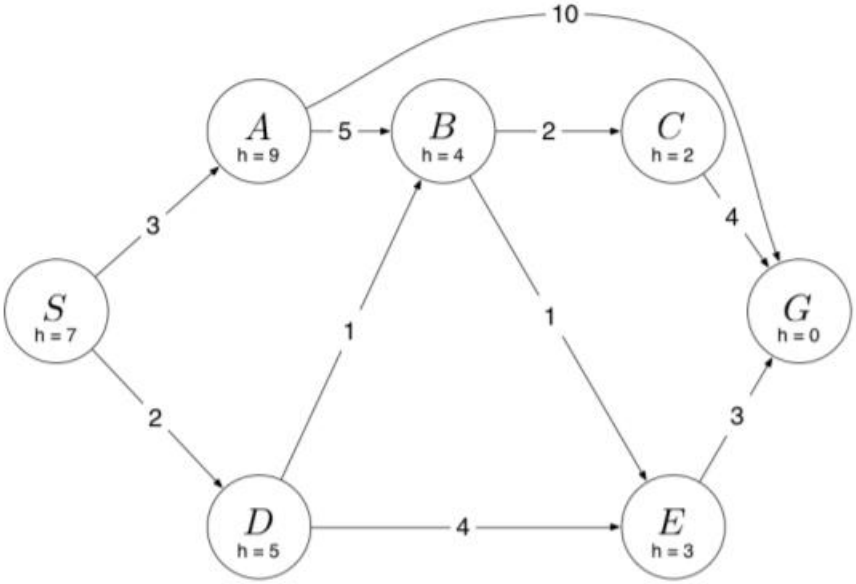
- تحلیل بهترین حالت. [زمان و حافظه خطی]
  - استراتژی حریصانه در بهترین حالت، مسیر جستجو را مستقیماً به سمت یک حالت هدف (که معمولاً بهینه نیست) هدایت می‌کند.
  - در بهترین حالت، اولین مسیر بررسی شده شامل گره هدف است.



- تحلیل بدترین حالت. [زمان و حافظه نمایی]
  - استراتژی حریصانه در بدترین حالت همانند یک جستجوی عمقی که خیلی بد هدایت شده است، عمل می‌کند.
  - در بدترین حالت، آخرین مسیر بررسی شده شامل گره هدف است.

جستجوهای آگاهانه

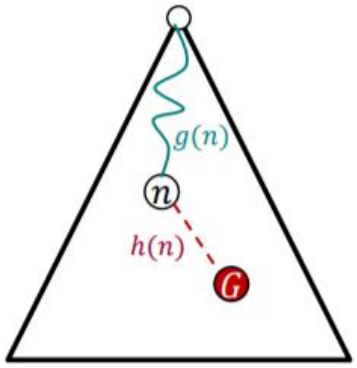
تمرین: اگر بر روی گراف زیر از جستجوی حریصانه استفاده شود، مسیر یافته شده کدام است؟



جستجوهای آگاهانه

جستجوی A*

- ایده. از ادامه دادن مسیرهایی که می‌دانی پرهزینه هستند، اجتناب کن.
- جستجوی A*. ترکیب مزایای جستجوی هزینه یکنواخت و جستجوی حریصانه.
  - جستجوی هزینه یکنواخت: کامل و بهینه است، اما می‌تواند بسیار زمان‌بر و حافظه‌بر باشد.
  - جستجوی حریصانه: نه کامل است و نه بهینه، اما می‌تواند بسیار سریع و کم حافظه باشد.
- تابع ارزیابی.
  - **g (n)**: هزینه‌ی واقعی مسیر پیموده شده از ریشه تا گره **n**.
  - **h (n)**: هزینه‌ی تخمینی کوتاه‌ترین مسیر از گره **n** تا هدف.
  - **f (n)**: هزینه‌ی تخمینی کوتاه‌ترین مسیر راه‌حلی که از گره **n** می‌گذرد.

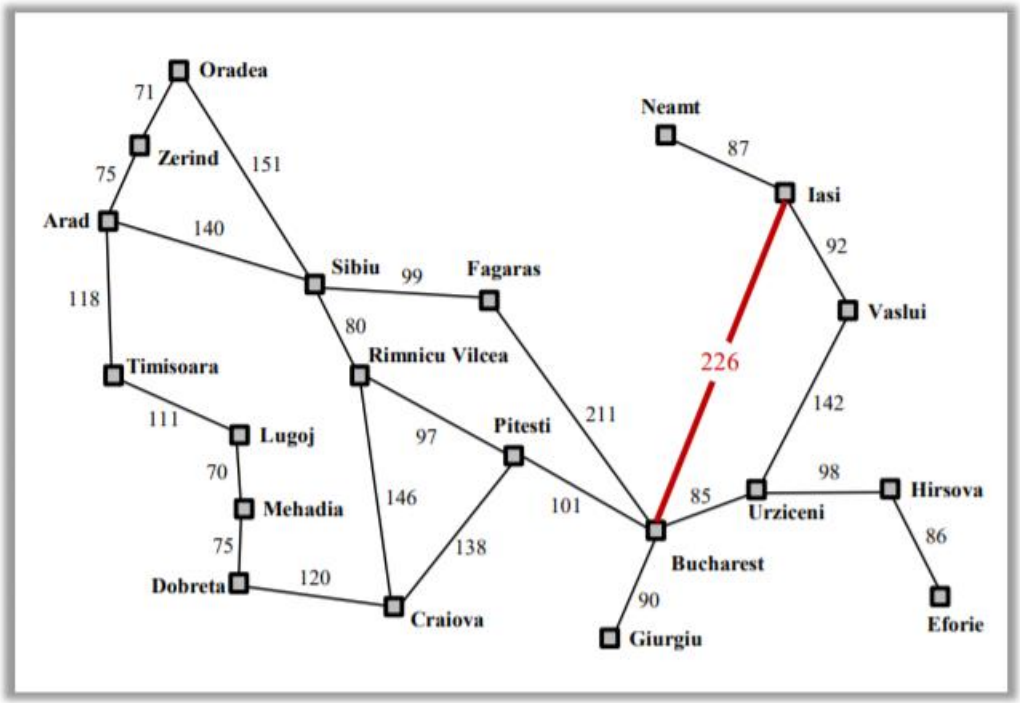


$$f(n) = g(n) + h(n)$$

جستجوهای آگاهانه

جستجوی A*

مثال: پیدا کردن مسیر بهینه از آراد به بخارست



فاصله مستقیم تا 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

جستجوهای آگاهانه

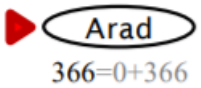
جستجوی A*

مثال: پیدا کردن مسیر بهینه از آراد به بخارست

g(Arad,Arad)=0

h(Arad,Bucharest)=366

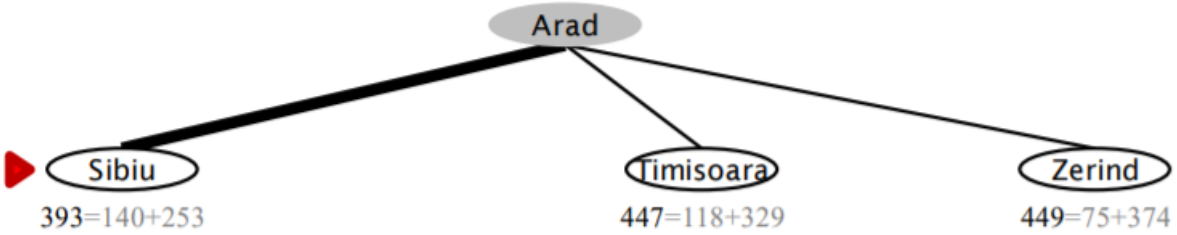
f(Arad)=0+366



جستجوهای آگاهانه

جستجوی A*

مثال: پیدا کردن مسیر بهینه از آراد به بخارست

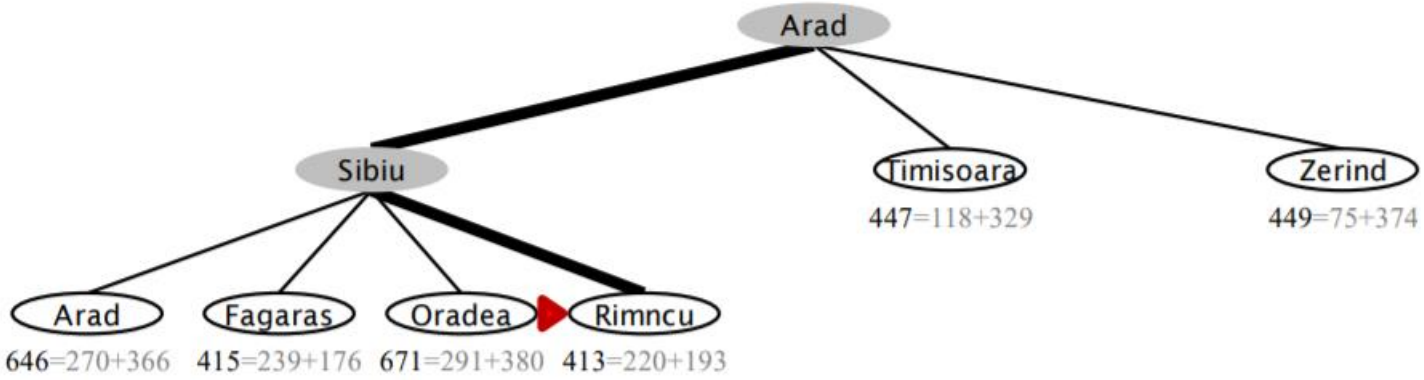


در این مساله (آراد – بخارست) مقدار تابع **g(n)** حاصل جمع اعدادی است از گره مبدا تا گره **n** روی یالها نوشته شده است و **h(n)** فاصله هوایی از روی جدول بدست می‌آید.

جستجوهای آگاهانه

جستجوی A*

مثال: پیدا کردن مسیر بهینه از آراد به بخارست

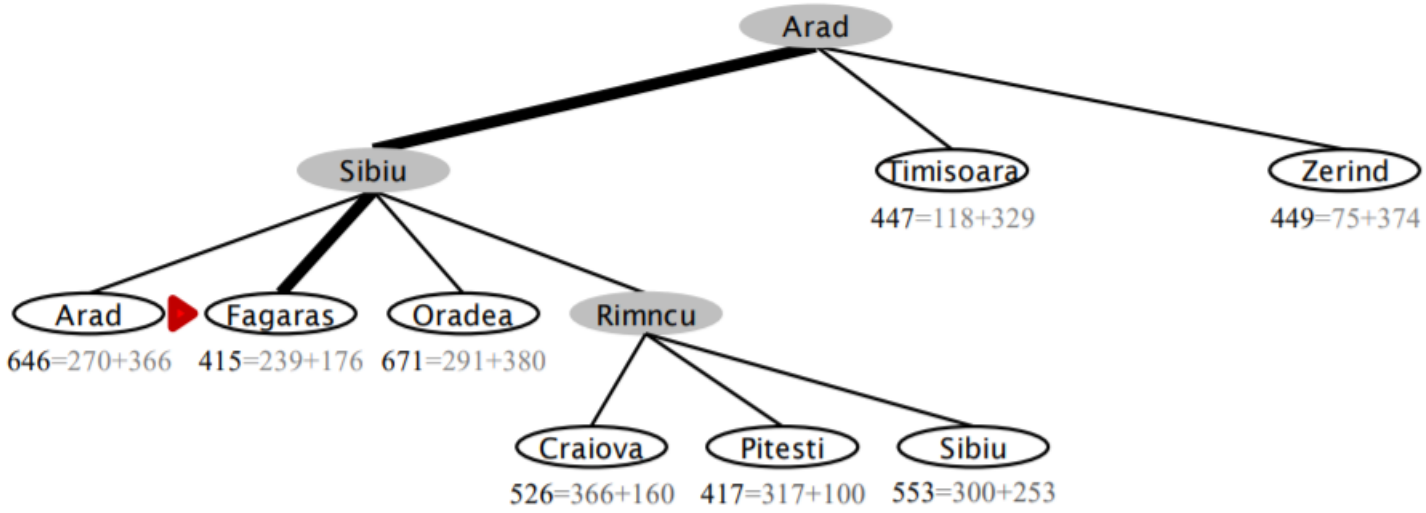


نکته: فرق جستجوی حریصانه با جستجوی A^* در این است که در حستجوی حریصانه فقط فاصله تخمینی گره تا هدف معیار انتخاب بود ولی در این روش معیار انتخاب مجموع فاصله تخمینی تا هدف بعلاوه فاصله واقعی گره مورد نظر از گره شروع در نظر گرفته می‌شود.

جستجوهای آگاهانه

جستجوی A*

مثال: پیدا کردن مسیر بهینه از آراد به بخارست

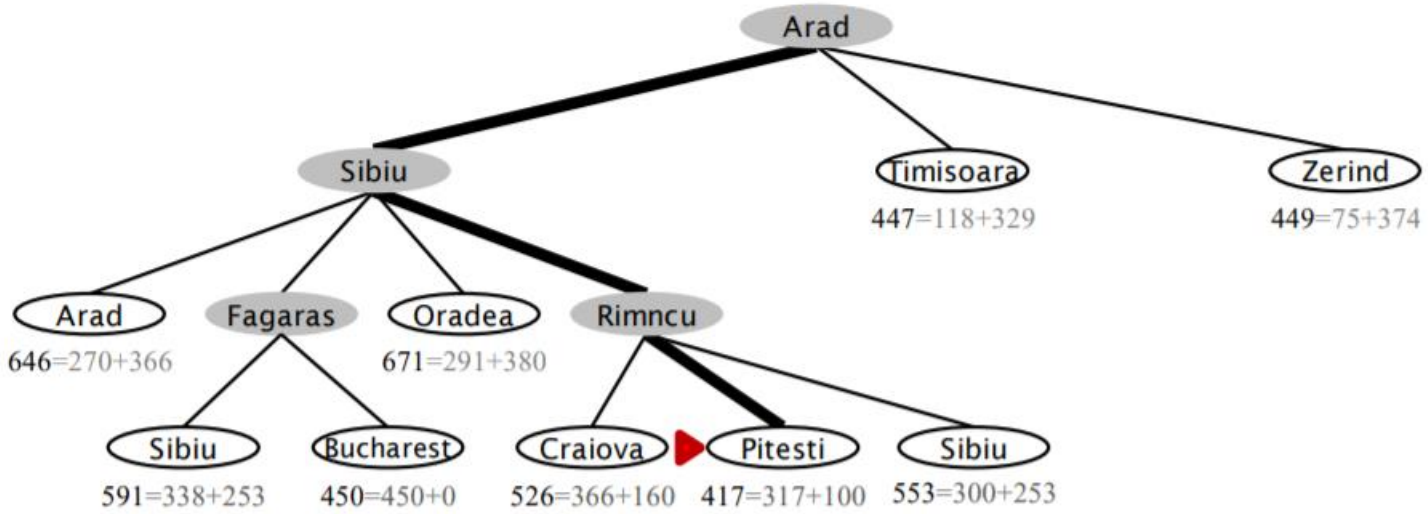


نکته: در این الگوریتم دوباره گره آراد شانس انتخاب شدن ندارد، زیرا هزینه آن بیشتراز بقبه گره‌ها می‌شود لذا این الگوریتم هیچ وقت در حلقه تکرار نمی‌افتد.

جستجوهای آگاهانه

جستجوی A*

مثال: پیدا کردن مسیر بهینه از آراد به بخارست

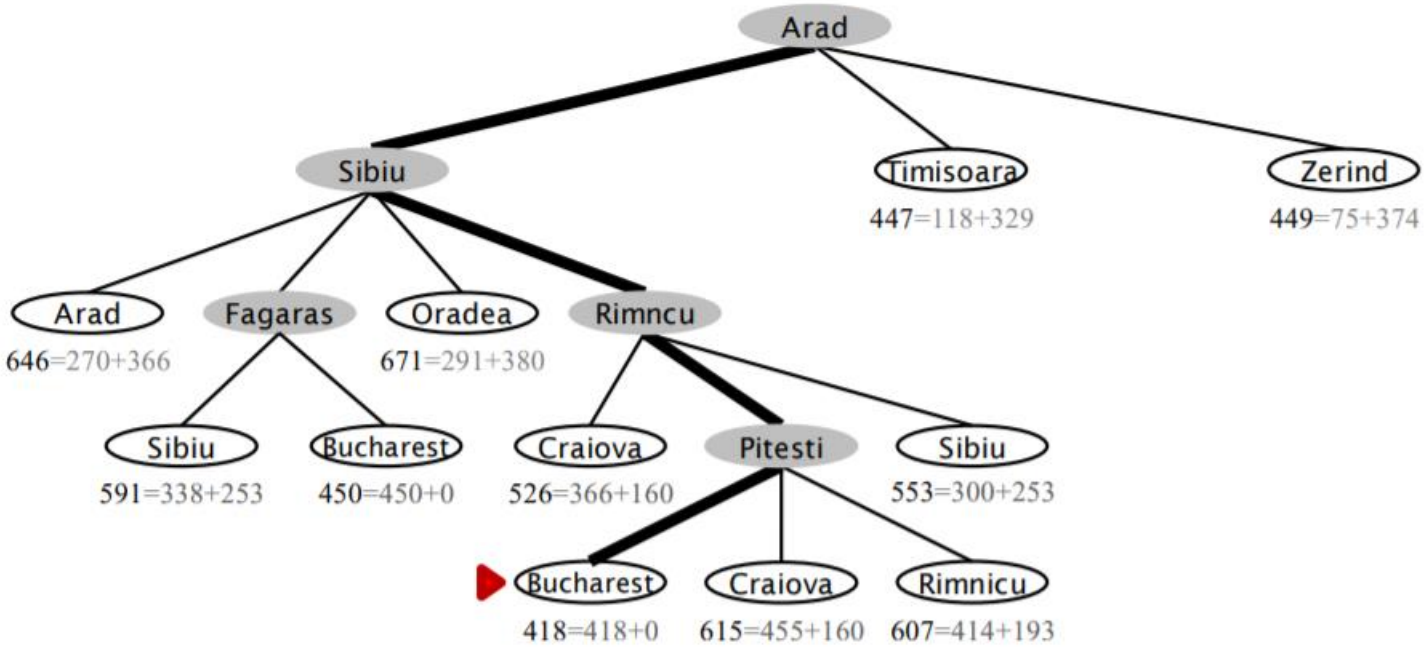


نکته: در گراف فوق می‌بینیم که **Bucharest** که گره هدف است تولید شده ولی چون مقدار هزینه مسیر آن نسبت به **Pitesti** بیشتر است لذا در این مرحله **Bucharest** انتخاب نمی‌شود.

جستجوهای آگاهانه

جستجوی A*

مثال: پیدا کردن مسیر بهینه از آراد به بخارست



جواب بدست آمده: Arad-Sibiu-Rimnicu -Pitesti-Bucharest

جستجوهای آگاهانه

تابع هیوریستیک قابل پذیرش

- یک تابع هیوریستیک قابل پذیرش است:
  1. هیچگاه تخمین اضافی از هزینه تا هدف ارایه ندهد.
  2. خوش بینانه باشد.

Formally:

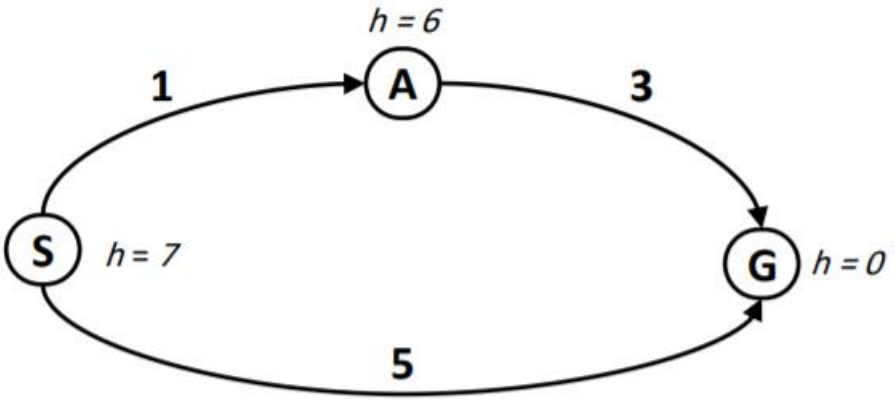
1. $h(n) <= h^*(n)$ where $h^*(n)$ is the true cost from n
2. $h(n) >= 0$ so $h(G)=0$ for any goal G.

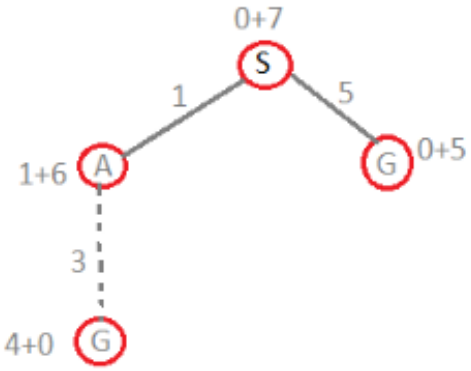
- آیا $h_{SLD}(n)$ قابل پذیرش است؟

بلی، چون فاصله هوایی بین دو شهر کوچکتر یا مساوی فاصله جاده‌ای بین آنهاست.

جستجوهای آگاهانه

مثال: تابع هیورستیک غیر قابل پذیرش.





- آیا **A*** در گراف بالا مسیر بهینه را برمی‌گرداند؟ ایراد کار در کجاست؟
- هزینه‌ی واقعی مسیر بد > هزینه‌ی تخمینی مسیر خوب
- بنابراین تخمین‌ها باید از مقدار واقعی کمتر باشند.

مشاهده می کنیم که مسیر بهینه را برنمی‌گرداند چون طبق شکل بجای مسیر S به A ، مسیر S به G را انتخاب می کند.

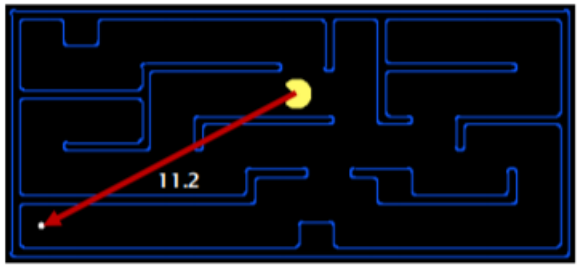
جستجوهای آگاهانه

- آیا جستجوی A* بهینه است؟
  - بله، به شرطی که تابع هیوریستیک قابل قبول باشد.
- هیوریستیک قابل قبول. تابع هیوریستیک h(n) قابل قبول است اگر به ازای هر گره مانند n همواره داشته باشیم:

$$0 \leq h(n) \leq h^*(n)$$

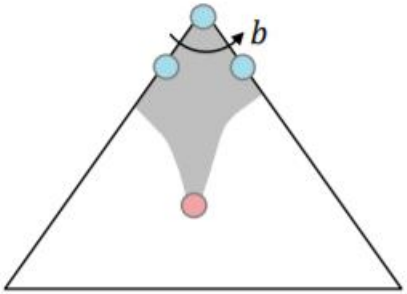
به طوری که h*(n) بیانگر هزینه‌ی واقعی کوتاه‌ترین مسیر از n تا هدف است.

- مثال. هیوریستیک فاصله مستقیم در مسایل مسیریابی.



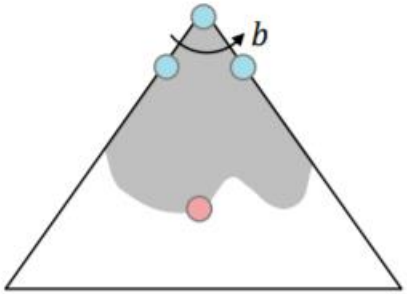
جستجوهای آگاهانه

مقایسه جستجوی A* با جستجوی با هزینه یکنواخت

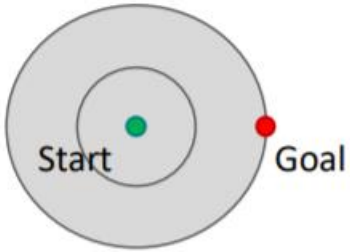


جستجوی A*

جستجوی A* عمدتا گره‌ها را در جهت هدف گسترش می‌دهد، اما برای تضمین بهینگی گاهی اوقات نیز به سمت اطراف حرکت می‌کند.



جستجوی هزینه یکنواخت



جستجوی هزینه یکنواخت بدون توجه به هدف، در همه جهت‌ها به طور یکسان جستجو می‌کند.

جستجوهای آگاهانه

مقایسه جستجوی A* با جستجوی حریصانه و جستجوی با هزینه یکنواخت

- جستجوی A* هم از هزینه‌ی پیموده شده استفاده می‌کند و هم از هزینه‌ی باقیمانده.
- جستجوی A* با توابع هیوریستیک قابل قبول یا سازگار بهینه است.
- طراحی هیوریستیک یک نکته کلیدی است: اغلب به روش راحت‌سازی!

جستجوهای آگاهانه

جستجوی A*

- کامل بودن؟
- بلی
- بهینه بودن؟
- بلی

نکته: در این الگوریتم حداقل گره های ممکن بسط داده می‌شود لذا بهینه موثر است.

C^* هزینه واقعی مسیر است.

- A* expands all nodes with $f(n) < C^*$
- A* expands some nodes with $f(n) = C^*$
- A* expands no nodes with $f(n) > C^*$

- بنابراین A* بهینه موثر نیز می‌باشد.

جستجوهای آگاهانه

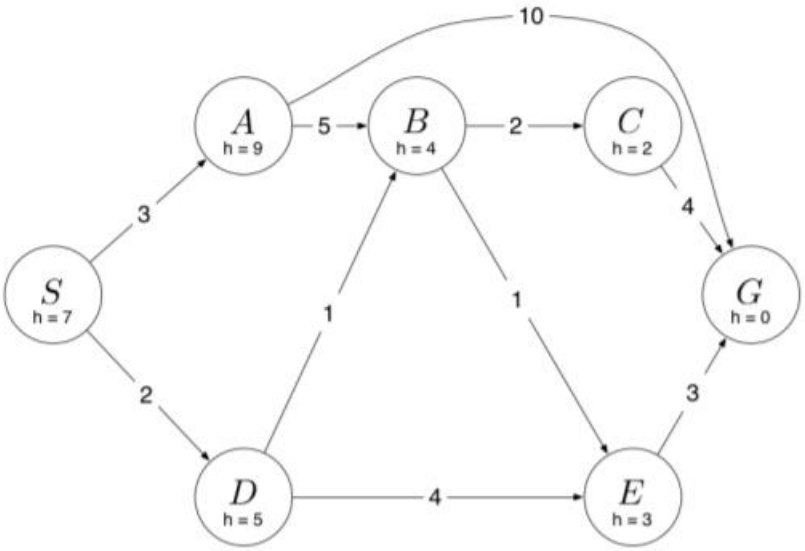
جستجوی A*

- مرتبه زمانی؟
- کماکان از مرتبه نمایی می‌باشد.
- مرتبه حافظه؟
- هم مرتبه با پیچیدگی زمانی است.

! حافظه به مراتب مشکل بزرگتری است.

جستجوهای آگاهانه

تمرین: اگر بر روی گراف زیر از جستجوی درختی A* استفاده شود، مسیر یافته شده کدام است؟



جستجوهای آگاهانه

کاربردهای جستجوی A*

- کاربردها.
  - مسیریابی
  - بازی های کامپیوتری
  - برنامه ریزی منابع
  - برنامه ریزی حرکت روبات
  - تحلیل زبانی
  - ترجمه ماشینی
  - شناسایی گفتار
  - ...

جستجوهای آگاهانه

روش‌های بهبود جستجوی A*

- یادآوری. پیچیدگی زمانی و پیچیدگی حافظه‌ی A* در اغلب مسائل نمایی است، مگر آن که برای آن مسئله یک تابع هیوریستیک بسیار دقیق وجود داشته باشد.
- ایراد اصلی A*. مصرف حافظه‌ی نمایی
- روش‌های کاهش مصرف حافظه در A*.
  - جستجوی IDA* (مصرف حافظه خطی)
  - جستجوی RBFS (مصرف حافظه خطی)
  - جستجوی SMA* (مصرف حافظه به اندازه حافظه موجود)

جستجوهای آگاهانه

جستجوی IDA* (Iterative deepening A*)

جستجوی عمیق کننده تکراری، تکنیکی مفید برای کاهش درخواست حافظه‌است. حال می‌توانیم از این تکنیک استفاده نموده و هر بار یک جستجوی عمقی تا هزینه f-limitرا جستجو کنیم. اگر به جواب نرسیدیم هزینه f-limit را افزایش داده و از اول به بسط فضای حالت می‌پردازیم. (f=g+h)

نکته ۱: محدودیت هزینه در هر مرحله به گونه‌ای انتخاب می‌شود که در مراحل قبلی ثابت شده‌است که جوابی با هزینه کمتر از این مقدار وجود ندارد.

نکته ۲: می‌توان هزینه مرحله جدید را برابر با کمترین هزینه گره‌ی که در مرحله قبلی بسط داده نشده قرار داد از آنجا که این روش به صورت عمقی است پس پیچیدگی فضایی آن در بدترین حالت (O(bd است.

نکته ۳: پیچیدگی زمانی بستگی به تابع هیورستیک دارد.
نکته ٤: این روش نیز مشابه A* کامل و بهینه‌است.

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.

جستجوهای آگاهانه

جستجوی اول – بهترین بازگشتی

الگوریتم RBFS

1. بهترین گره برگ و بهترین جانشین برای آن انتخاب شود.
2. اگر مقدار بهترین گره برگ از جانشین آن بیشتر شد، آنگاه به مسیر جانشین عقبگرد شود.
3. در حین عقبگرد، مقدار f(n) بروزرسانی شود.
4. گره جانشین بسط داده شود.

نکته: منظور از بهترین مقدارگره ، مینیمم f(n)=g(n)+h(n) است.

جستجوهای آگاهانه

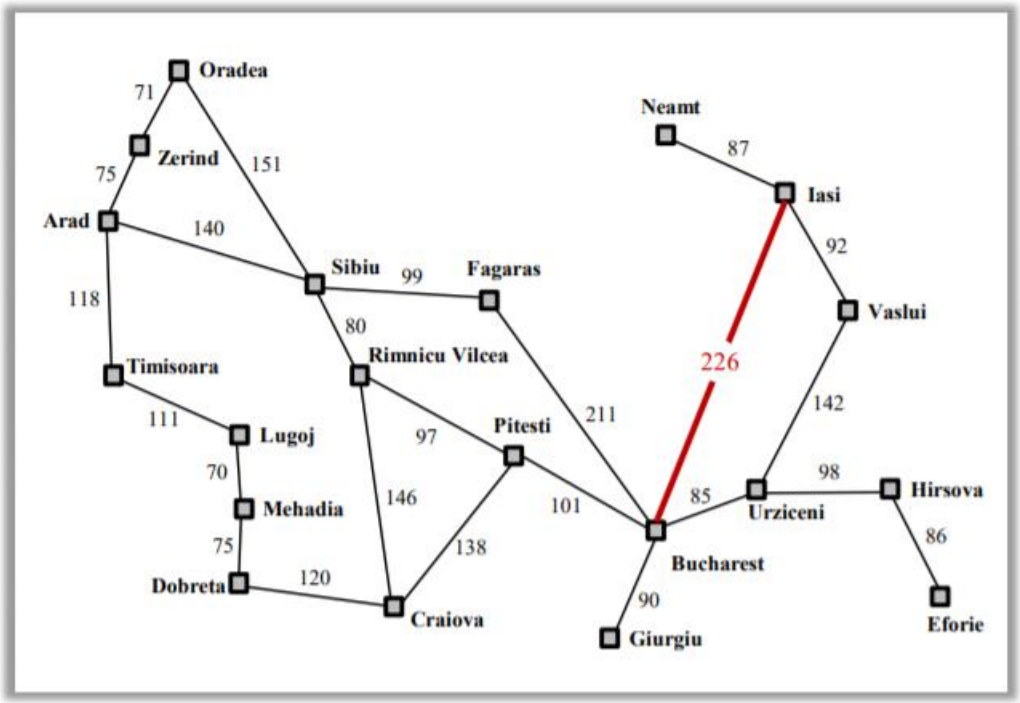
```
function RECURSIVE-BEST-FIRST-SEARCH(problem) return a solution or failure
    return RBFS(problem, MAKE-NODE(problem.INITIAL-STATE,∞)

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

جستجوهای آگاهانه

جستجوی RBFS

مثال: پیدا کردن مسیر بهینه از آراد به بخارست

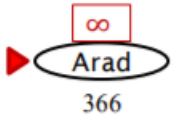


فاصله مستقیم تا بخارست	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

جستجوهای آگاهانه

جستجوی RBFS

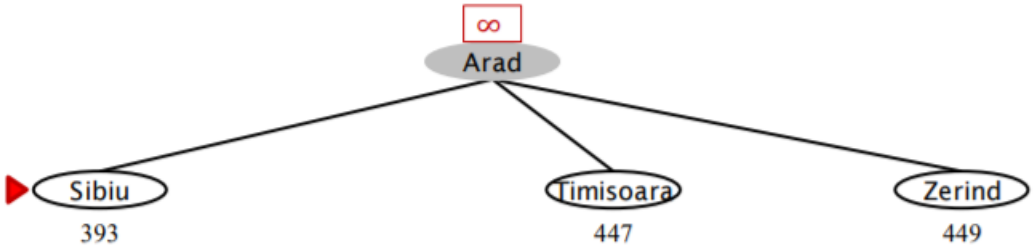
مثال: پیدا کردن مسیر بهینه از آراد به بخارست



جستجوهای آگاهانه

جستجوی RBFS

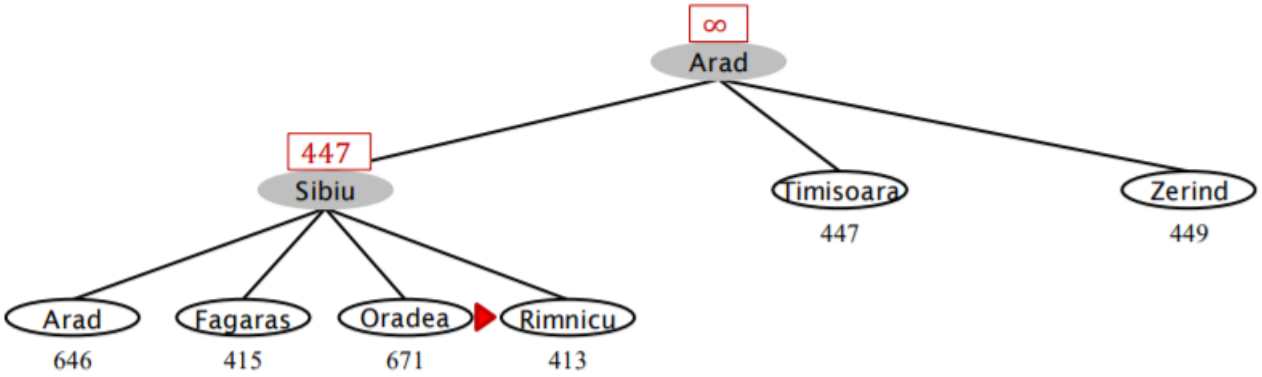
مثال: پیدا کردن مسیر بهینه از آراد به بخارست



جستجوهای آگاهانه

جستجوی RBFS

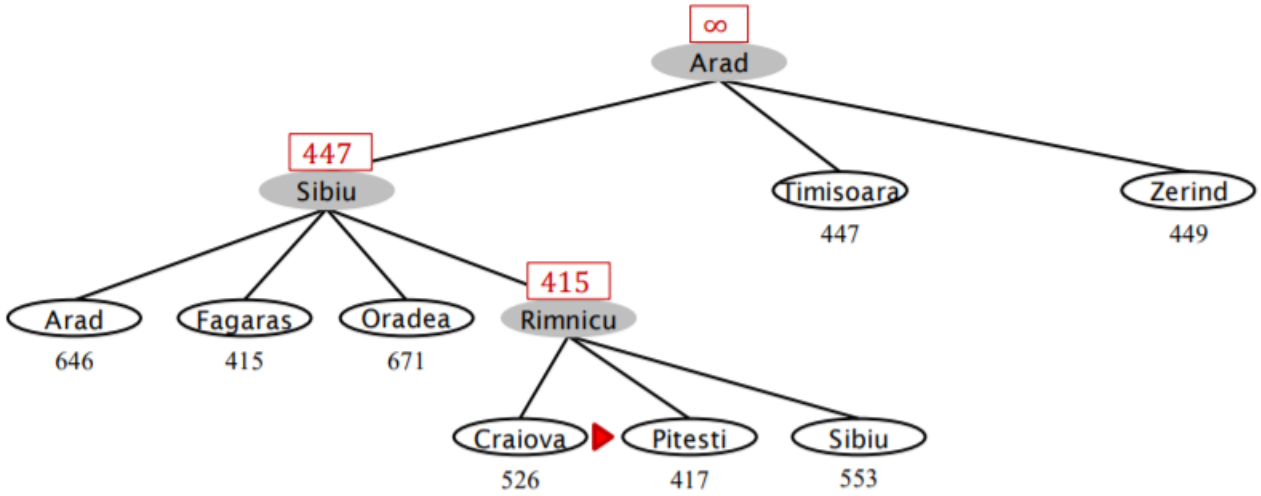
مثال: پیدا کردن مسیر بهینه از آراد به بخارست



جستجوهای آگاهانه

جستجوی RBFS

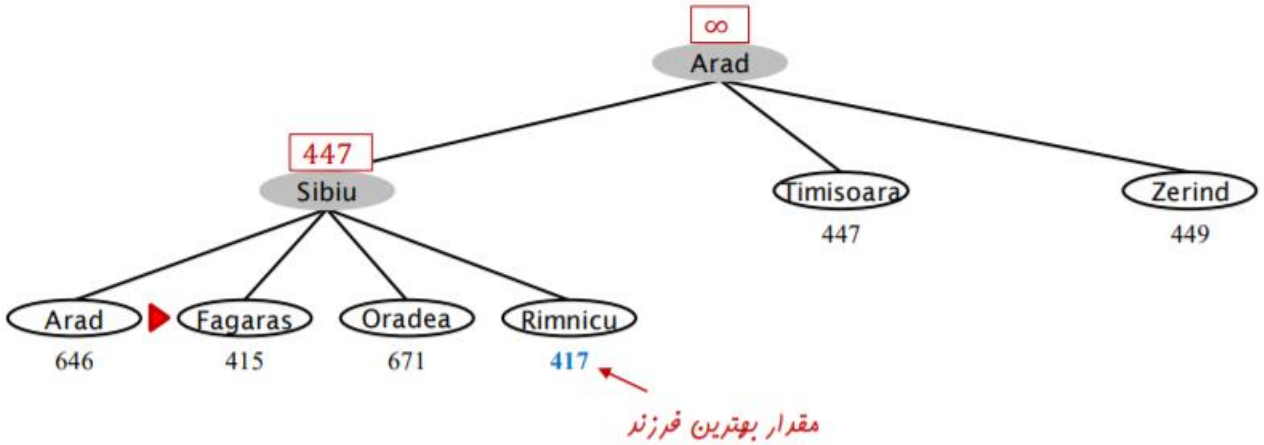
مثال: پیدا کردن مسیر بهینه از آراد به بخارست



جستجوهای آگاهانه

جستجوی RBFS

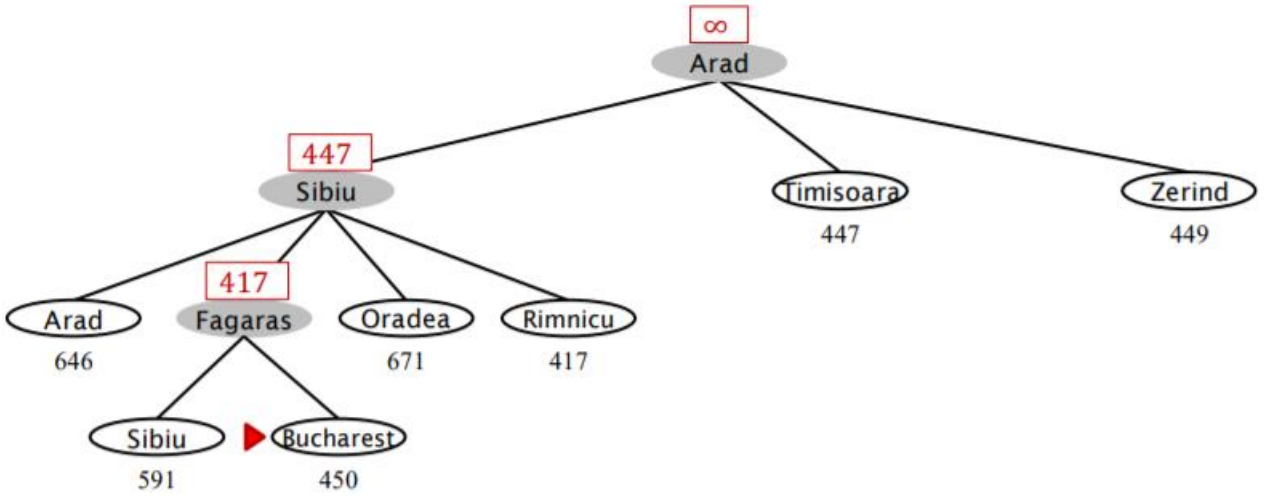
مثال: پیدا کردن مسیر بهینه از آراد به بخارست



جستجوهای آگاهانه

جستجوی RBFS

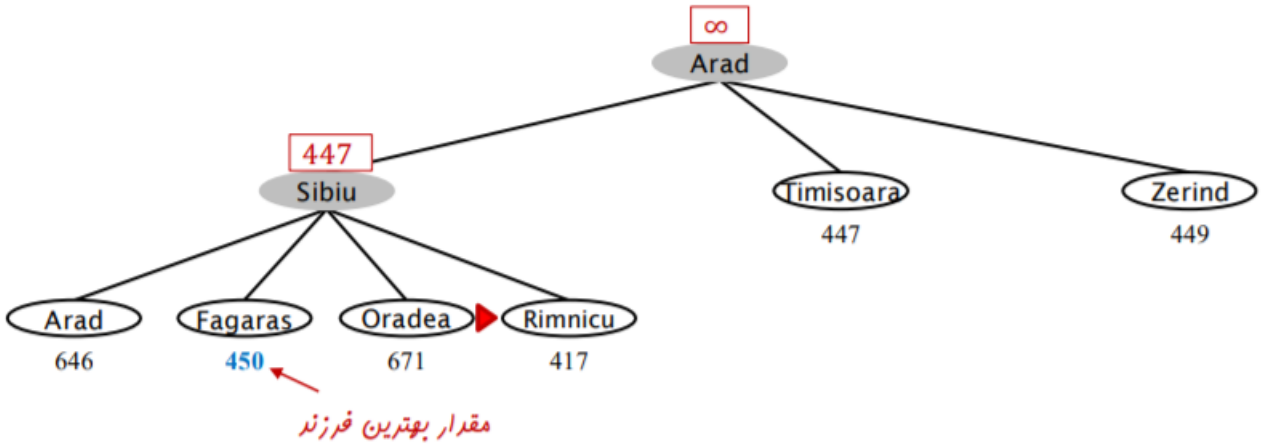
مثال: پیدا کردن مسیر بهینه از آراد به بخارست



جستجوهای آگاهانه

جستجوی RBFS

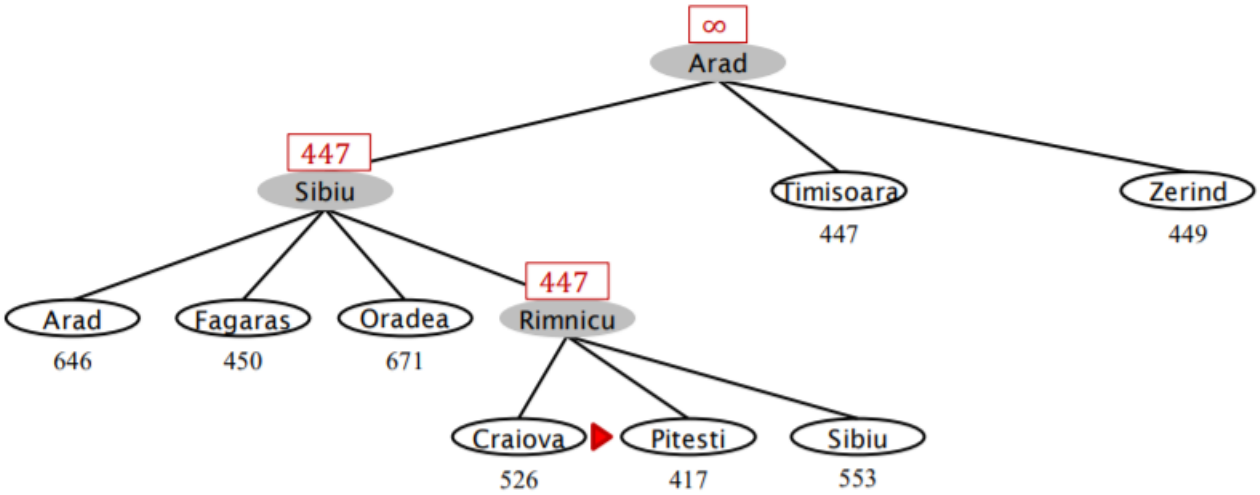
مثال: پیدا کردن مسیر بهینه از آراد به بخارست



جستجوهای آگاهانه

جستجوی RBFS

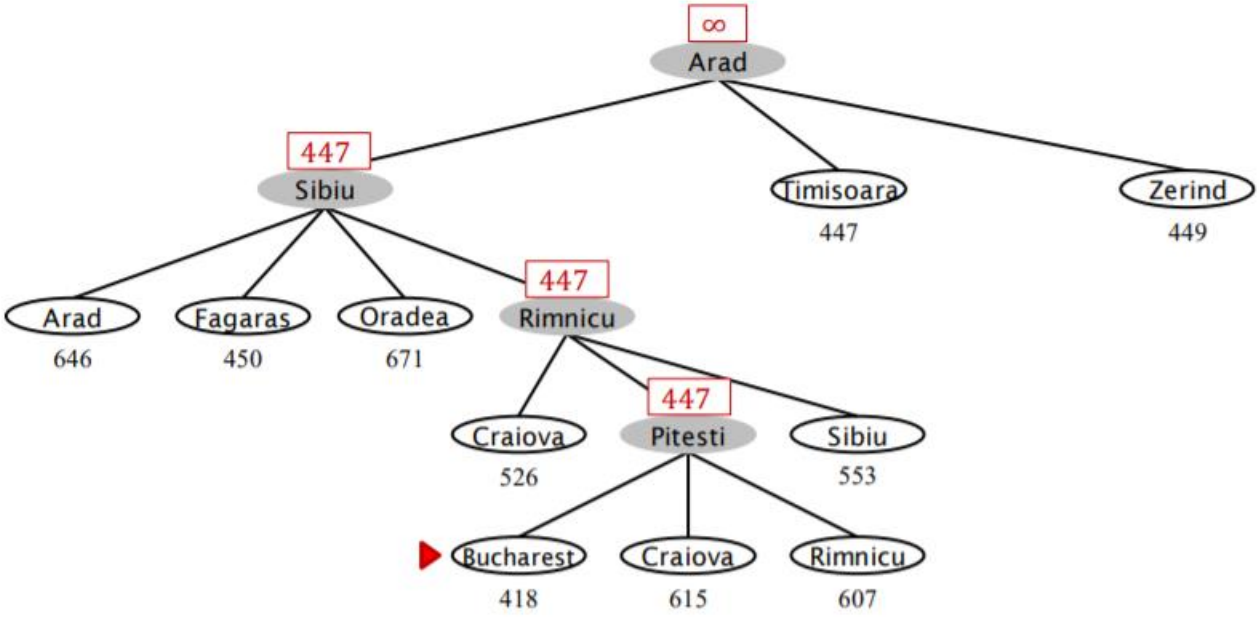
مثال: پیدا کردن مسیر بهینه از آراد به بخارست



جستجوهای آگاهانه

جستجوی RBFS

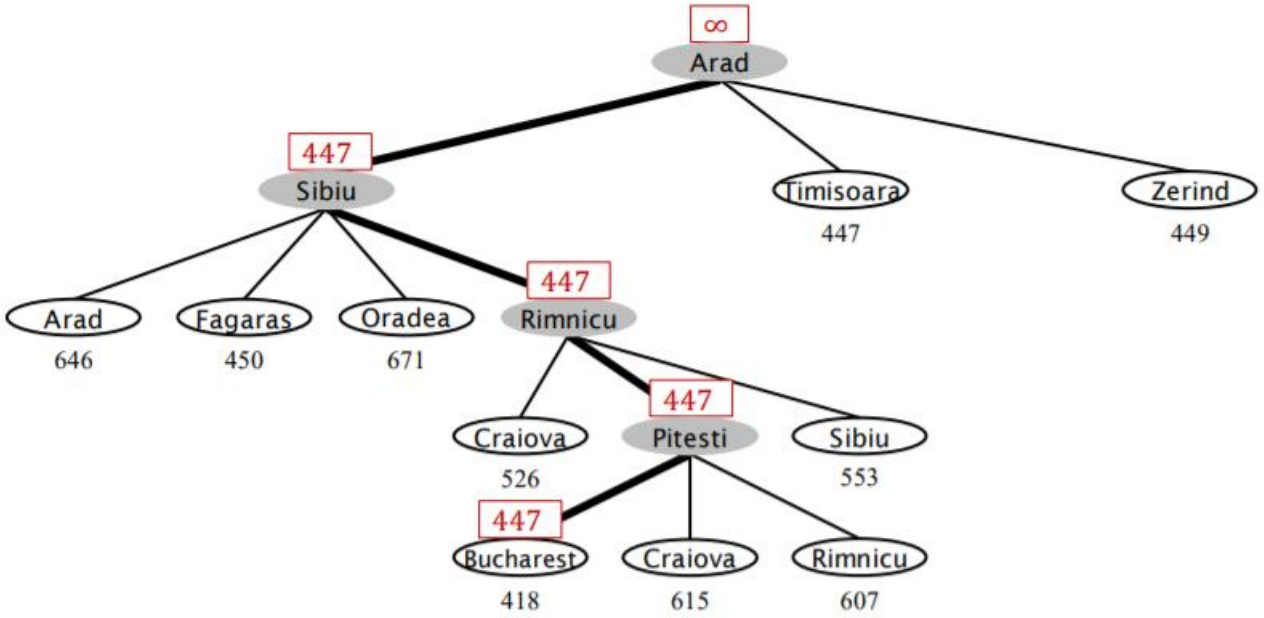
مثال: پیدا کردن مسیر بهینه از آراد به بخارست



جستجوهای آگاهانه

جستجوی RBFS

مثال: پیدا کردن مسیر بهینه از آراد به بخارست



جستجوهای آگاهانه

RBFS

- RBFS جستجوی به مراتب موثرتری از ID A* است.
- کماکان از تولید تعداد بسیار زیادی گره به دلیل **تغییر عقیده** رنج می برد.
- مانند A* اگر h(n) قابل پذیرش باشد، بهینه است.
- پیچیدگی حافظه o(bd) است.
- پیچیدگی زمانی به کیفیت تابع هیوریستیک و میزان تغییر عقیده بستگی دارد.

نکته: جواب RBFS با جواب A* یکی است ولی RBFS حافظه کمتری نیاز دارد.

جستجوهای آگاهانه

جستجوی SMA*

- ایده. استفاده از کل حافظه‌ی موجود!
  - گسترش بهترین گره برگی تا زمانی که حافظه پر نشده است.
  - حذف بدترین گره برگی در صورت پر بودن حافظه و سپس گسترش بهترین گره برگی
- اگر هزینه‌ی تمام برگ‌ها برابر باشند چه می‌شود؟
  - ممکن است یک گره هم برای گسترش و هم برای حذف انتخاب شود.
  - گسترش: بهترین گره برگی که از همه جدیدتر است.
  - حذف: بدترین گره برگی که از همه قدیمی‌تر است.
- کامل بودن و بهینگی.
  - SMA* کامل است اگر حداقل یک راه‌حل قابل دستیابی وجود داشته باشد.
  - SMA* بهینه است اگر راه‌حل بهینه قابل دستیابی باشد.

جستجوهای آگاهانه

جستجوی SMA*

الگوریتم **SMA***، حافظه محدود **A*** ساده شده **Simplified-Memory-Bounded A*** می‌باشد.

این الگوریتم، قادر است تا از تمام حافظه موجود برای اجرای جستجو استفاده کند. استفاده از حافظه بیشتر کارایی جستجو را وسعت می‌بخشد.

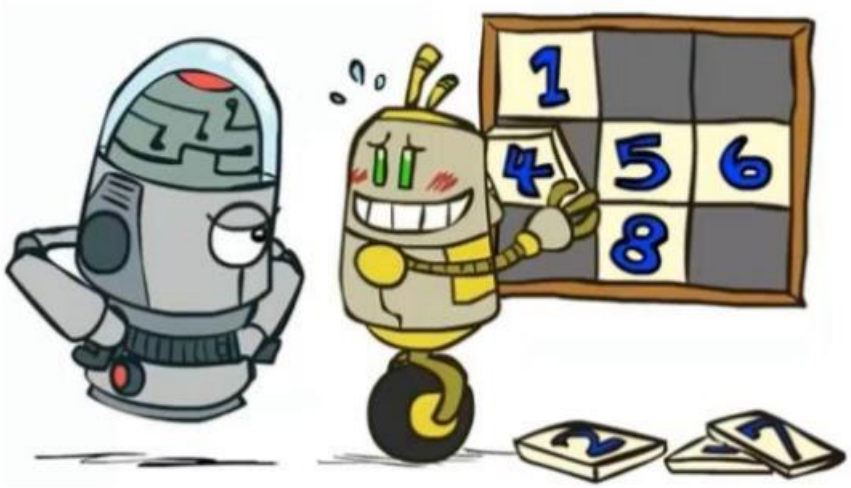
طراحی **SMA*** ساده است:

زمانی که نیاز به تولید فرزند داشته باشد ولی حافظه‌ای نداشته باشد، نیاز به ساختن فضا بر روی صف دارد. برای انجام این امر، یک گره را حذف می‌کند. گره‌هایی که به این طریق از صف حذف می‌شوند، گره‌های فراموش‌شده یا (**forgotten nodes**) نامیده می‌شوند.

برای اجتناب از جستجوی مجدد زیردرخت‌هایی که از حافظه حذف شده‌اند، در گره‌های اجدادی، اطلاعاتی در مورد کیفیت بهترین مسیر در زیر درخت فراموش شده، نگهداری می‌شود.

جستجوهای آگاهانه

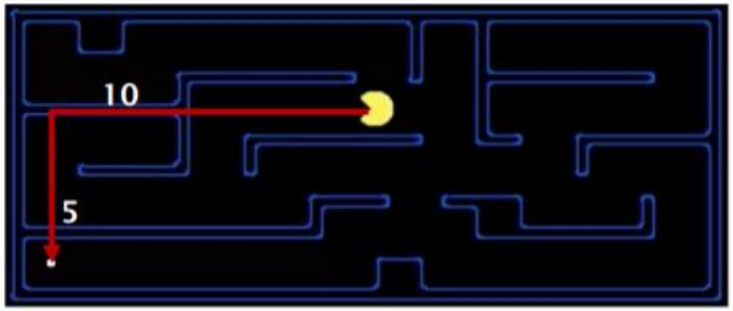
ابداع تابع هیوریستیک



جستجوهای آگاهانه

ابداع تابع هیوریستیک

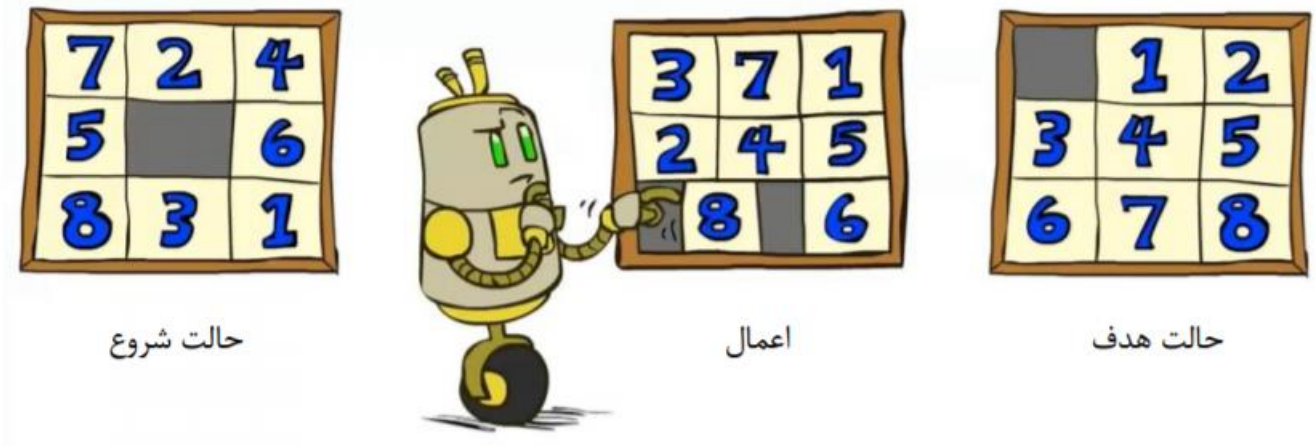
- حل مسایل سخت به صورت بهینه، بستگی به وجود توابع هیوریستیک قابل قبول دارد.
- روش راحت‌سازی.
  - در اغلب موارد، توابع هیوریستیک با حل کردن مسایل راحت شده به دست می‌آیند.
- مسئله‌ی راحت شده. نسخه‌ای از مسئله‌ی اصلی که در آن یک یا چند محدودیت حذف شده‌اند.



فاصله مانهاتانی

جستجوهای آگاهانه

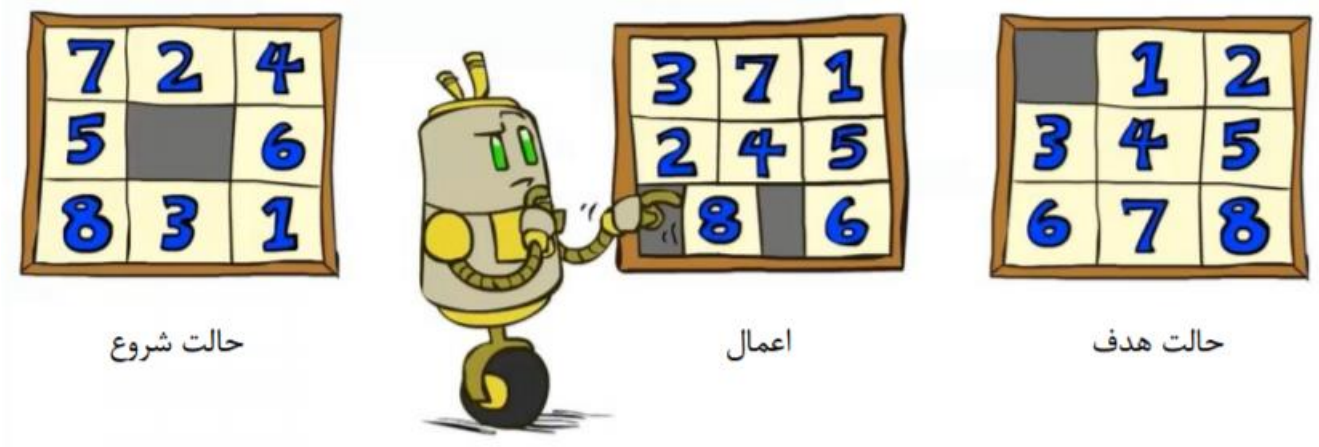
ابداع تابع هیوریستیک



- حالت ها؟
- تعداد حالت ها؟
- اعمال؟
- هزینه؟

جستجوهای آگاهانه

ابداع تابع هیوریستیک

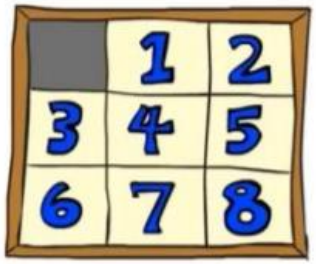


- حالت ها؟ هر چیدمان از ۸ کاشی بر روی صفحه
- تعداد حالت ها؟ ۹!
- اعمال؟ حرکت دادن یک کاشی به یک خانه مجاور و خالی
- هزینه؟ ۱ به ازای هر عمل

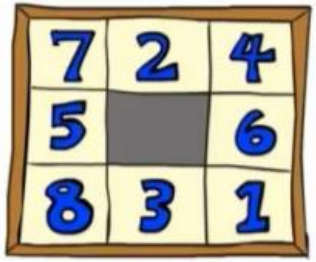
جستجوهای آگاهانه

ابداع تابع هیوریستیک

- محدودیت ها. [برای حرکت دادن یک کاشی از خانه‌ی A به خانه‌ی B]
  I. خانه‌ی B باید خالی باشد.
  II. خانه‌ی B باید همسایه‌ی خانه‌ی A باشد.



حالت هدف



حالت شروع

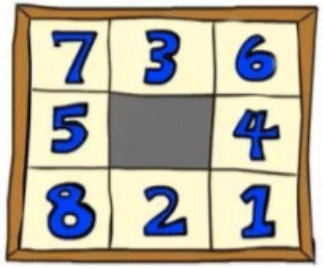
- مسایل راحت شده.
  - نسخه‌ی ۱. هر دو محدودیت حذف شوند.
  - نسخه‌ی ۲. محدودیت I حذف شود.
  - نسخه‌ی ۳. محدودیت II حذف شود.

- نکته اصلی. هزینه‌ی دقیق یک مسئله راحت شده را می‌توان به عنوان تخمینی از هزینه‌ی واقعی مسئله‌ی اصلی در نظر گرفت.

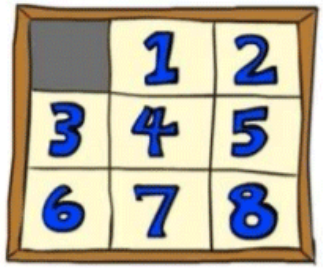
جستجوهای آگاهانه

راحت سازی ۱

- مسایل راحت شده.
  - نسخه ۱. هر دو محدودیت حذف شوند.
  - نسخه ۲. محدودیت I حذف شود.
  - نسخه ۳. محدودیت II حذف شود.



$h_1(start) = 8$



$h_1(goal) = 0$

- هزینه‌ی حل مسئله راحت شده = تعداد کاشی‌هایی که در مکان نادرست قرار دارند.

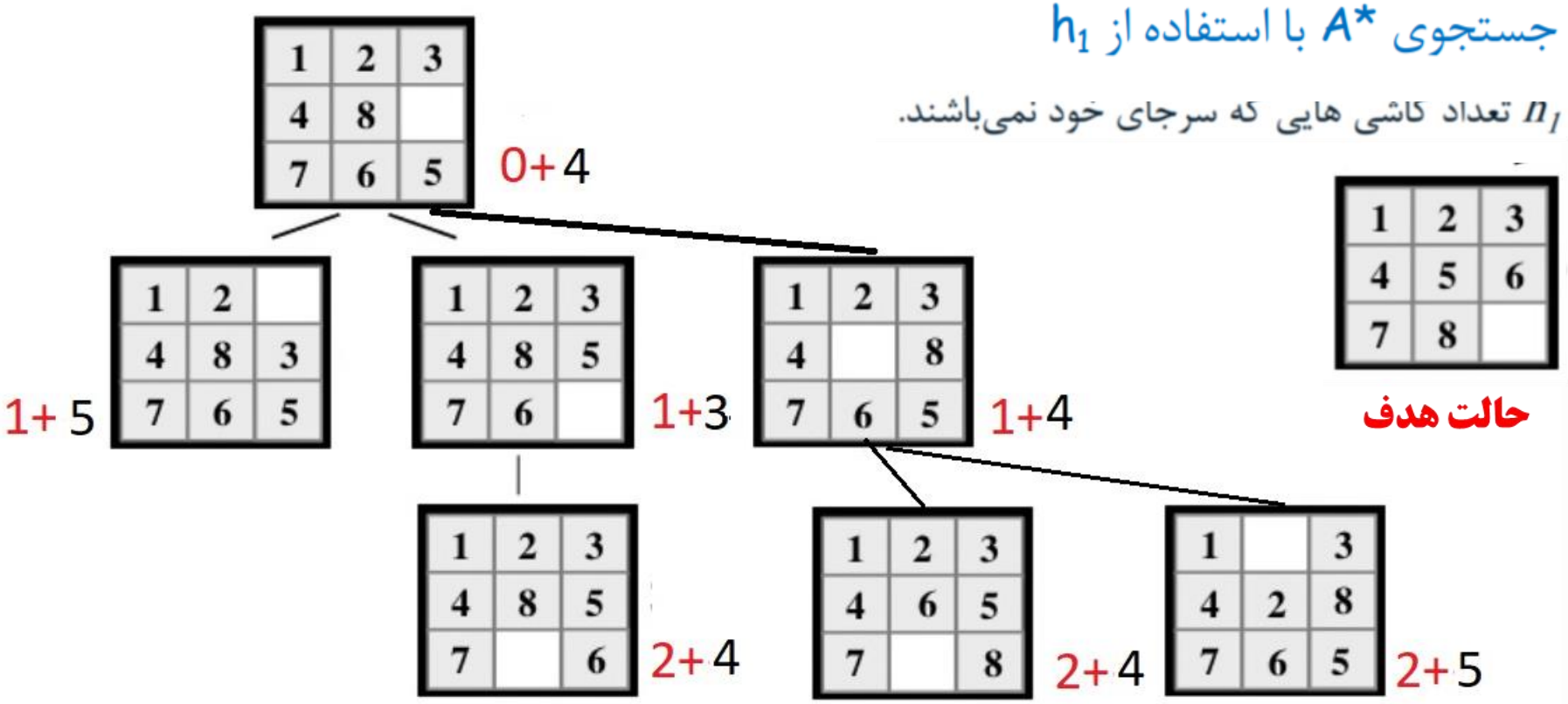
	عمق ۴	عمق ۱۴	عمق ۲۴
IDS	۱۱۲	۳،۴۷۳،۹۴۱	۵/۴ × ۱۰^۱۰
h_1	۱۳	۵۳۹	۳۹،۱۳۵

مقایسه‌ی تعداد گره‌های تولید شده به وسیله‌ی جستجوی عمیق کننده‌ی تکراری و جستجوی A*

جستجوهای آگاهانه

جستجوی A^* با استفاده از h_1

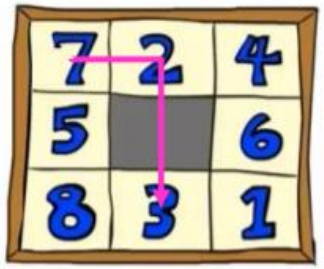
h_1 تعداد کاشی هایی که سرجای خود نمی‌باشند.



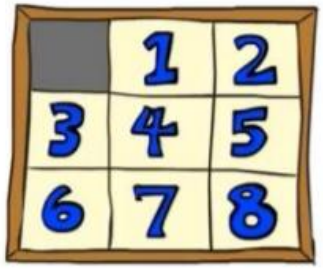
جستجوهای آگاهانه

راحت سازی ۲

$h_2(start) = 18$



$h_2(goal) = 0$



- مسایل راحت شده.
  - نسخه ۱. هر دو محدودیت حذف شوند.
  - نسخه ۲. محدودیت I حذف شود.
  - نسخه ۳. محدودیت II حذف شود.

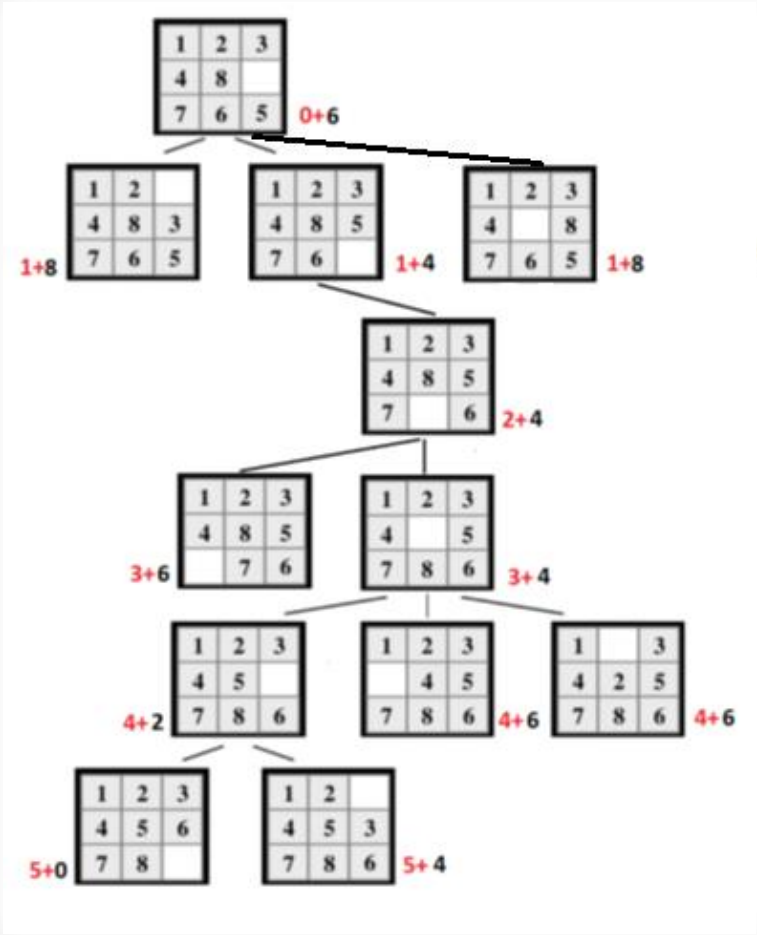
$h_2(start) = 3+1+2+2+2+3+3+2$

- هزینه‌ی حل مسئله راحت شده = مجموع فواصل مانهاتانی کاشی‌ها تا مکان هدف.

	عمق ۴	عمق ۱۴	عمق ۲۴
IDS	۱۱۲	۳.۴۷۳.۹۴۱	۵/۴ × ۱۰^۱۰
h_1	۱۳	۵۳۹	۳۹.۱۳۵
h_2	۱۲	۱۱۳	۱.۶۴۱

مقایسه‌ی تعداد گره‌های تولید شده به وسیله‌ی جستجوی عمیق کننده‌ی تکراری و جستجوی A*

جستجوهای آگاهانه



جستجوی A^* با استفاده از h_2

h_2 مجموع فاصله افقی – عمودی (منهتن) هر کاشی تا جای واقعی

1	2	3
4	5	6
7	8	

حالت هدف

جستجوهای آگاهانه

ابداع تابع هیوریستیک

- س. آیا می‌توان از هزینه‌ی واقعی به عنوان یک هیوریستیک استفاده نمود؟
  - آیا این هیوریستیک قابل قبول است؟
  - آیا این هیوریستیک باعث صرفه جویی در تعداد گره‌های تولید شده می‌شود؟
  - مشکل استفاده از این هیوریستیک چیست؟
- توجه. در ابداع توابع هیوریستیک، مسئله‌ی اصلی برقراری یک توازن مناسب میان کیفیت تابع هیوریستیک و هزینه‌ی محاسبه‌ی آن به ازای هر گره است.
  - هیوریستیک‌های دقیق‌تر تعداد گره‌های کمتری تولید می‌کنند، اما محاسبه‌ی آنها معمولا به زمان بیشتری نیاز دارد.

جستجوهای آگاهانه

مقایسه کیفیت تابع هیوریستیک

- س. آیا هیوریستیک h_2 همواره تعداد گره های کمتری نسبت به h_1 تولید می‌کند؟
- ج. بله، زیرا h_2 بر h_1 تسلط دارد.

- تسلط. اگر به ازای هر n داشته باشیم:
  - $h_2(n) >= h_1(n)$
  - و هر دو تابع هیوریستیک قابل قبول باشند،

آنگاه h_2 بر h_1 تسلط دارد.

	عمق ۴	عمق ۱۴	عمق ۲۴
h_1	۱۳	۵۳۹	۳۹،۱۳۵
h_2	۱۲	۱۱۳	۱،۶۴۱

جست‌وجوهای آگاهانه

ابداع تابع هیوریستیک (خلاصه)

- راحت‌سازی. می‌توان هیوریستیک‌های قابل قبول را برای یک مسئله، از هزینه‌ی دقیق راه‌حل یک نسخه‌ی راحت شده از مسئله به دست آورد.
- مثال. [معمای ۸] یک کاشی می‌تواند از خانه‌ی A به خانه‌ی B حرکت کند، اگر A همسایه B باشد و B خالی باشد.
  - اگر یک کاشی بتواند به هر خانه‌ای حرکت کند، آنگاه هیوریستیک h_1 کوتاه‌ترین راه‌حل را می‌دهد.
  - اگر یک کاشی بتواند به هر خانه‌ی همسایه حرکت کند، آنگاه هیوریستیک h_2 کوتاه‌ترین راه‌حل را می‌دهد.
- نکته‌ی کلیدی. هزینه‌ی راه‌حل بهینه در یک مسئله‌ی راحت، بیشتر از هزینه‌ی راه‌حل بهینه در مسئله‌ی اصلی نیست!

جستجوهای آگاهانه

ابداع تابع هیوریستیک (خلاصه)

سوال: کدام یک از توابع h_1 و h_2 تخمین بهتری است؟
جواب: هر کدام که محدودیت‌های کمتری را نقض می‌کند. ولذا h_2 بهتر است.
معمولا هر کدام که فضای حالت بزرگتری تخمین می‌زند بهتر است چون به مقدار واقعی نزدیکتر است.

روشهای دیگر

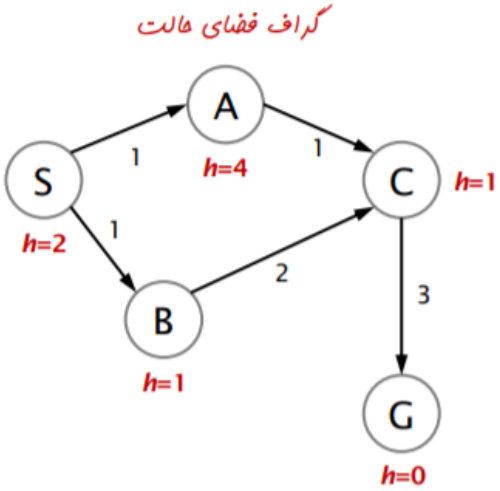
روش زیر مسئله:
در این روش تعدادی از مربع‌های پازل را نادیده می‌گیریم و مساله را برای باقیمانده حالت‌ها که ساده‌تر است حل می‌کنیم و از این تخمینی برای کل مساله بدست می‌آوریم.

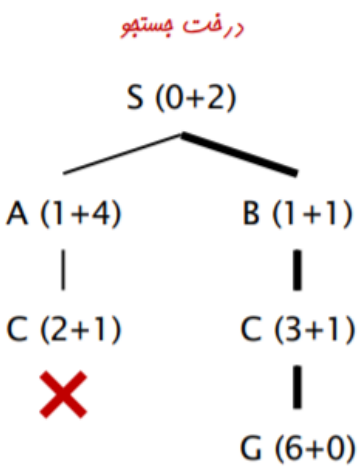
روش یاگیری ماشین:
سیستم پس حل تعداد زیاد مسائل مشابه، خودش تابع هیوریستیک را پیدا می‌کند.

جستجوهای آگاهانه

بهینگی A* در جستجوی گرافی

- مثال. یافتن مسیر از S به G.



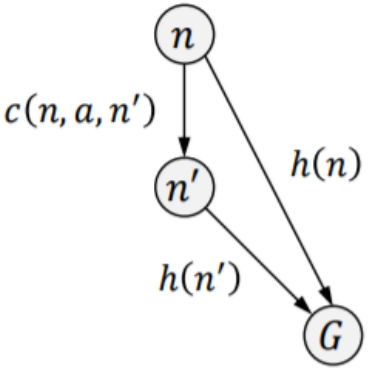


- توجه. در این مثال اگرچه تابع هیوریستیک قابل قبول است، اما A* قادر به یافتن راه‌حل بهینه نیست.

جستجوهای آگاهانه

توابع هیوریستیک سازگار

- هیوریستیک سازگار. یک هیوریستیک سازگار است اگر:



$$h(n) \leq c(n, a, n') + h(n')$$

- یکنوایی. اگر h سازگار باشد، آنگاه f در طول هر مسیری غیرکاهشی است.

$$\begin{aligned} f(n') &= g(n') + h(n') \\ &= g(n) + c(n, a, n') \\ &\geq g(n) + h(n) \\ &= f(n) \end{aligned}$$

- قضیه. اگر h سازگار باشد، آنگاه A* در جستجوی گرافی بهینه است.

جستجوهای آگاهانه

پایان فصل چهارم