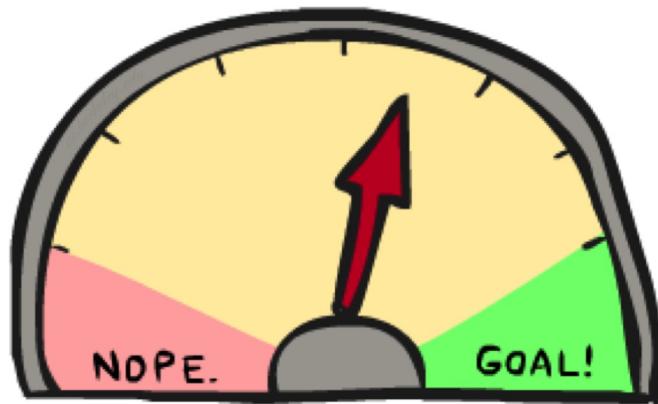


روش‌های جستجوی آگاهانه

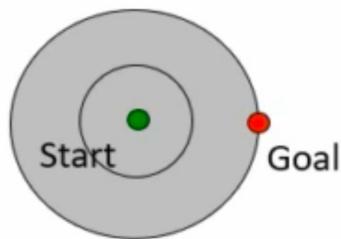
(Informed Search Methods)



.1 جستجوی حریصانه (Greedy Search)

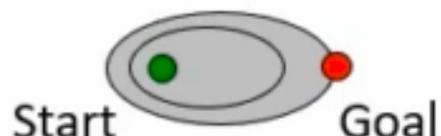
.2 جستجوی A^*

جستجوی آگاهانه



جستجوی ناآگاهانه

- تنها از اطلاعات موجود در تعریف مسئله استفاده می‌کند.
- تنها می‌تواند حالت‌های هدف را از حالت‌های غیرهدف تشخیص دهد.



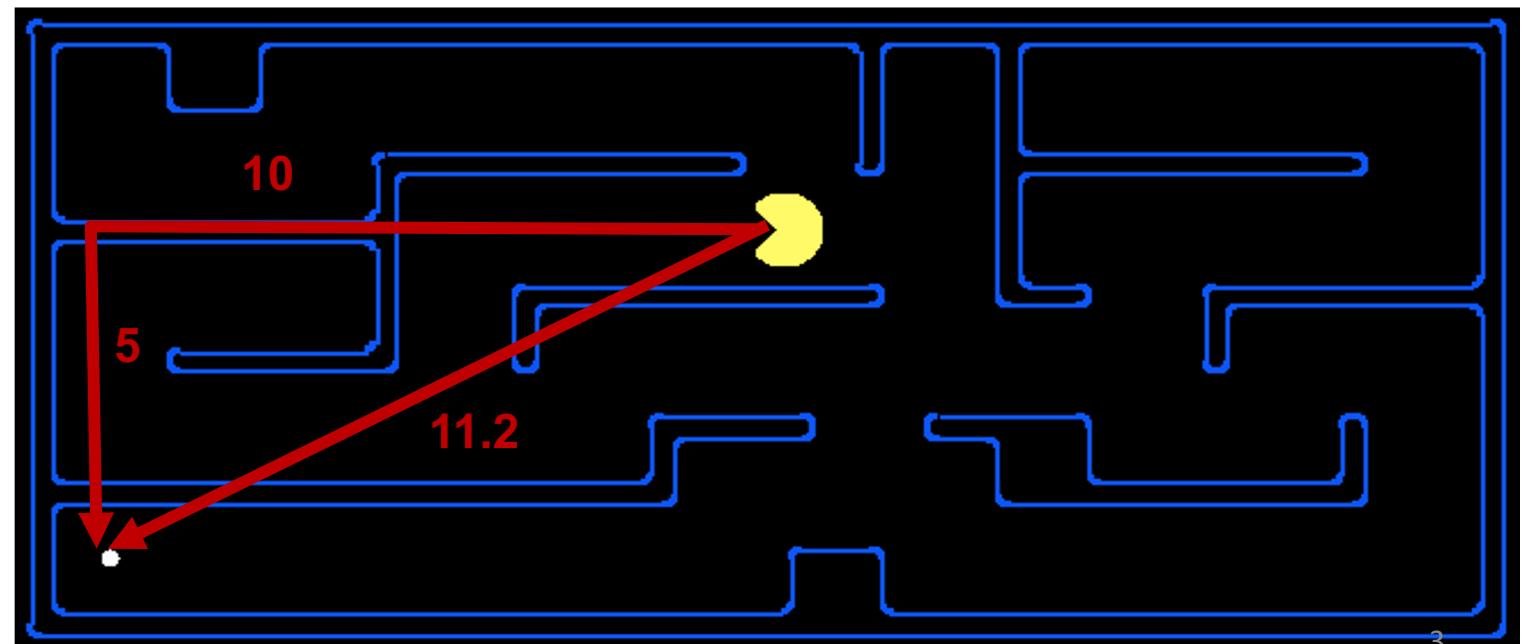
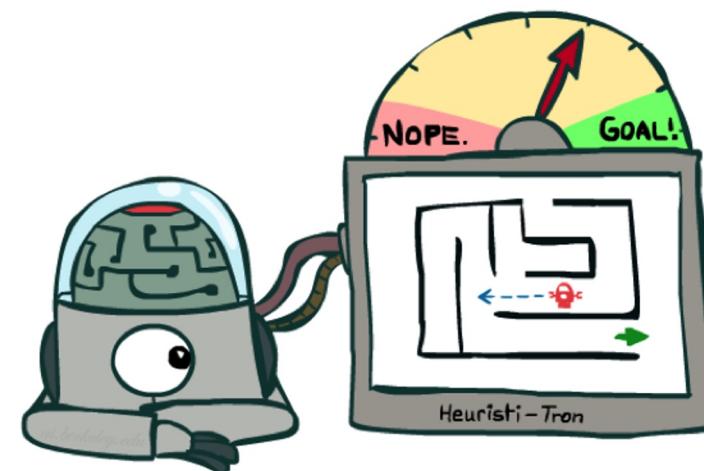
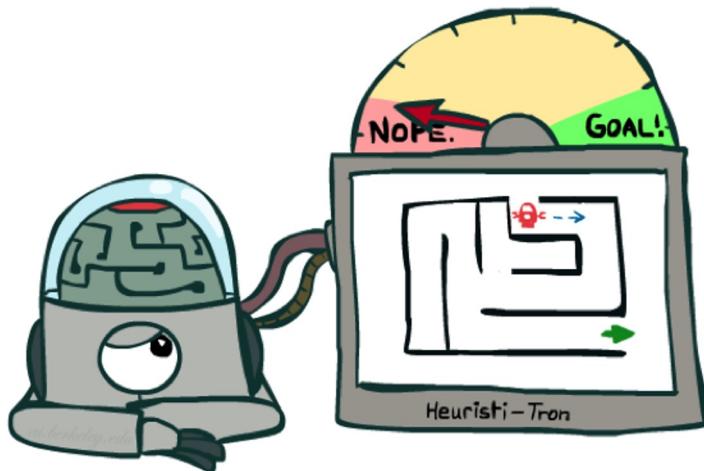
جستجوی آگاهانه

- علاوه بر اطلاعات موجود در تعریف مسئله، از اطلاعات خاص مسئله بهره می‌برد.
- بین حالت‌های غیرهدف (بسته به فاصله آن‌ها تا هدف) تمایز قائل می‌شود.
- هدایت جستجو به سمت هدف به جای پراکنده و بی‌هدف بودن.

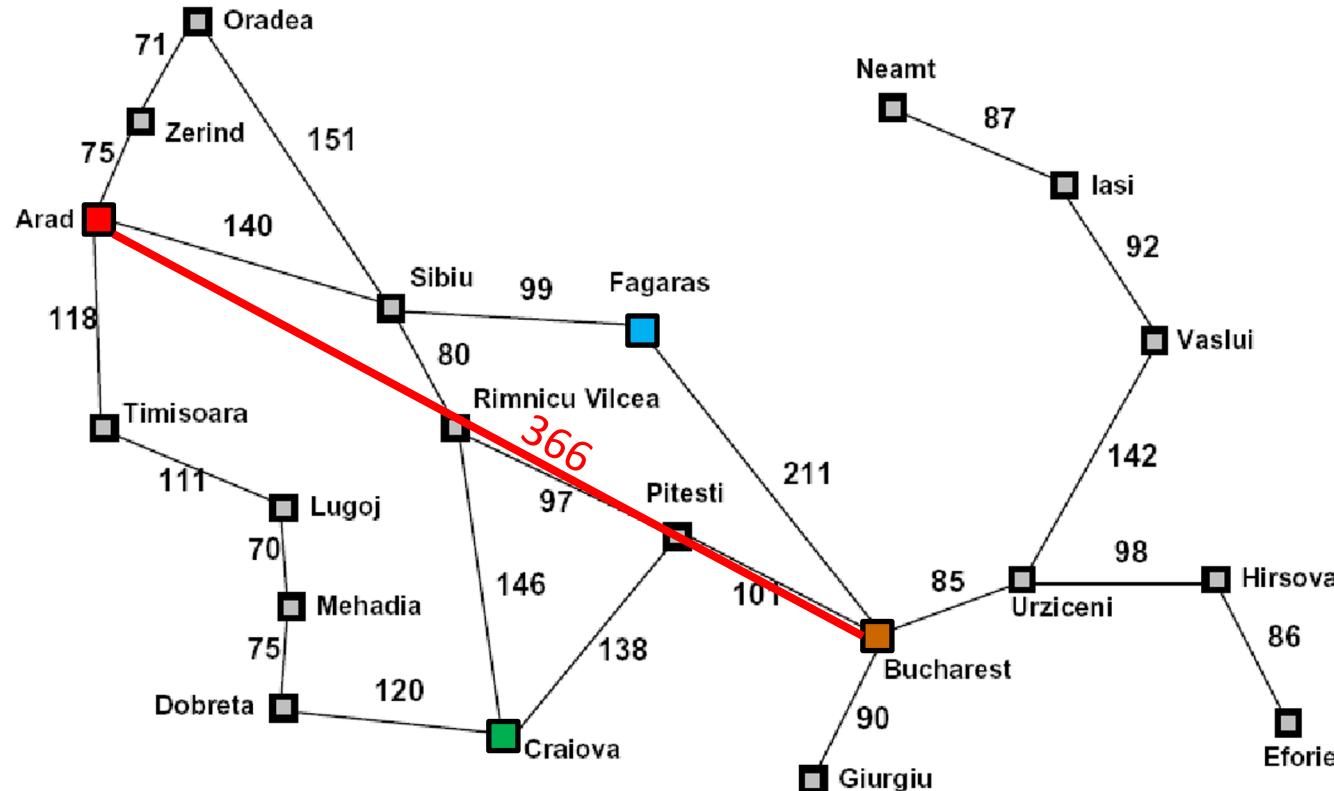
تابع هیوریستیک (Heuristics)

هیوریستیک

- تابعی است که فاصله هر حالت را تا هدف تخمین می‌زند.
- برای هر مسئله جستجو خاص آن طراحی می‌شود.
- مثال: تابع هیوریستیک در مسائل مسیریابی می‌تواند فاصله منهتن و فاصله اقلیدسی باشد.



مثال: تابع هیوریستیک در مسیریابی در رومانی



Straight-line distance to Bucharest	
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

$h(x)$

جستجوی حریصانه (Greedy Search)



جستجوی حریصانه

- الگوریتم جستجوی حریصانه از یک تابع هیوریستیک $h(n)$ استفاده می‌کند که مقدار آن نشان می‌دهد چقدر یک گره به هدف نزدیک است.
- هر گره‌ای که مقدار $h(n)$ آن کمتر باشد، زودتر بررسی و گسترش داده می‌شود.

مراحل اجرای الگوریتم:

۱. شروع از گره اولیه (Start Node):

- گره شروع را در صفات اولویت قرار می‌دهیم.
- مقدار هیوریستیک آن را محاسبه می‌کنیم $h(Start)$.

۲. بررسی گره‌های مجاور (Expansion of Nodes):

- از صفات اولویت، گره‌ای را که کمترین مقدار $h(n)$ دارد، انتخاب می‌کنیم.
- اگر این گره هدف باشد، جستجو متوقف می‌شود.
- در غیر این صورت، همهٔ فرزندان این گره را تولید و مقدار $h(n)$ آن‌ها را محاسبه می‌کنیم.

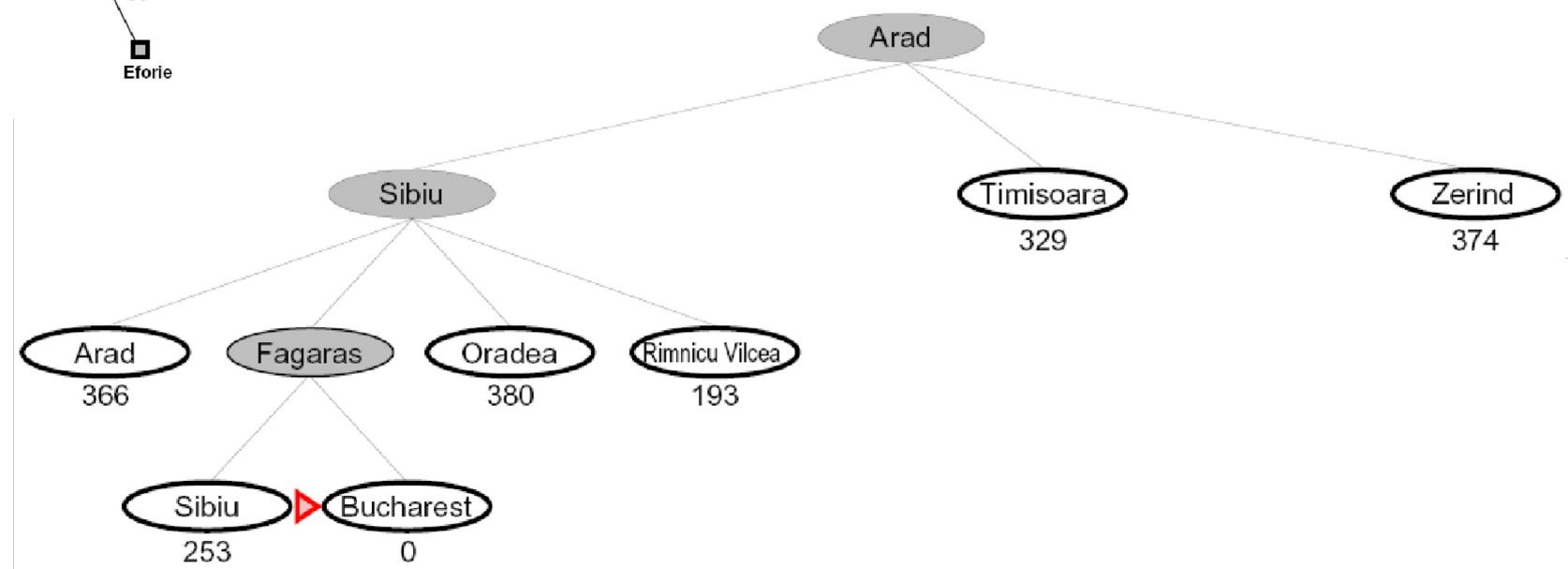
۳. مرتب‌سازی مجدد صفات اولویت:

- همهٔ گره‌های جدید را در صفات اولویت قرار می‌دهیم.
- گره‌ای که کمترین مقدار $h(n)$ دارد، در اولویت انتخاب قرار می‌گیرد.

۴. تکرار مراحل ۲ و ۳ تا رسیدن به هدف یا خالی شدن صفات اولویت

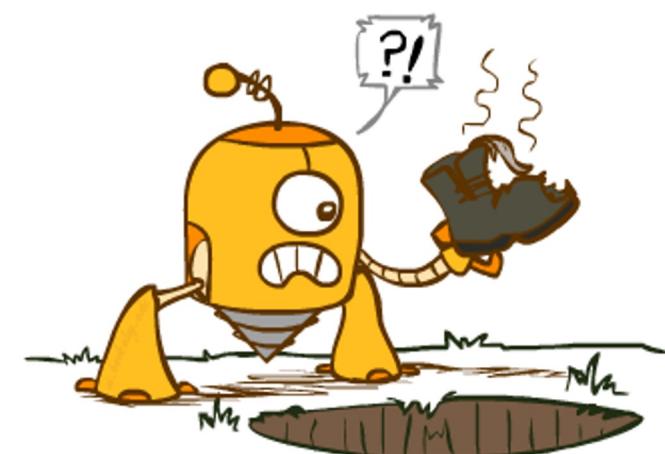
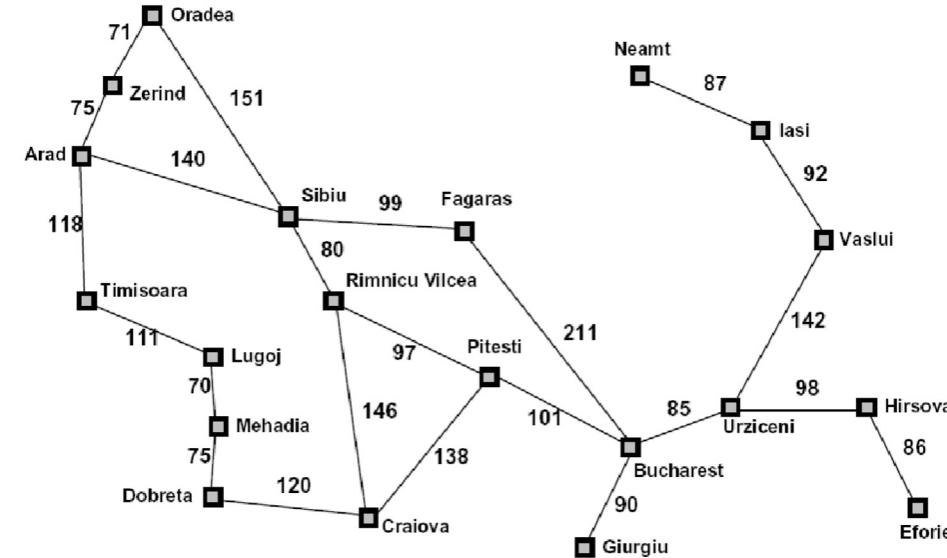
مثال: جستجوی حریصانه

هدف: رفتن از Arad به Bucharest



آیا بهینه است؟

▪ خیر. مسیر به دست آمده به بخارست کوتاهترین مسیر نیست!



ارزیابی جستجوی حریصانه

1. کامل بودن (Complete)

- خیر. ممکن است در یک حلقه بی نهایت گیر کند

2. بهینه بودن (Optimal)

- خیر

3. پیچیدگی زمانی (Time complexity)

- نمایی-مانند جستجوی عمقی

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

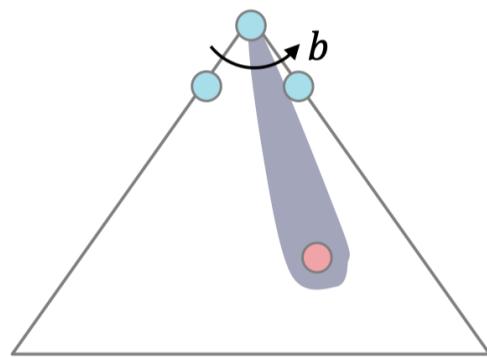
4. پیچیدگی حافظه (Space complexity)

$$O(b^m)$$

- نمایی

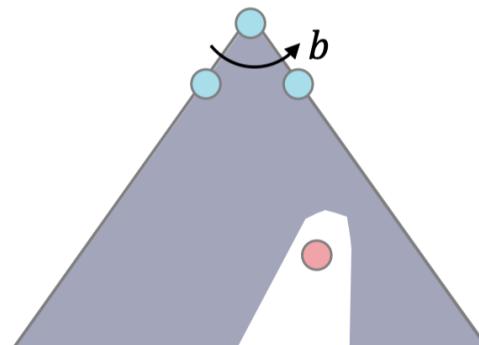
جستجوی حریصانه: تحلیل کارایی

□ **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]

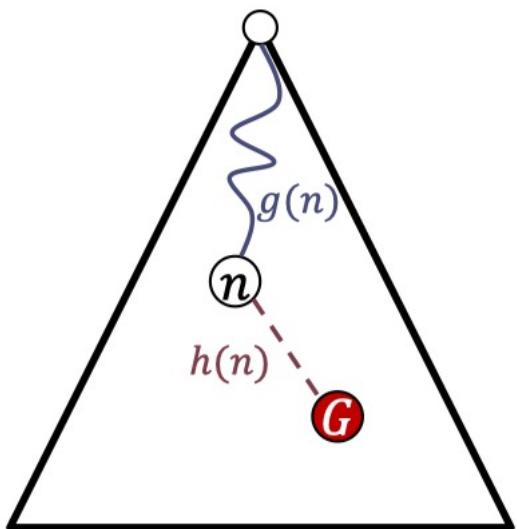
- استراتژی حریصانه در بدترین حالت همانند یک جستجوی عمیق که خیلی بد هدایت شده است، عمل می‌کند.
- در بدترین حالت، آخرین مسیر بررسی شده شامل گره هدف است.

جستجوی A^*



جستجوی A^*

□ جستجوی A^* ترکیبی از دو الگوریتم زیر است:



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

▪ جستجوی هزینه یکنواخت:

- کم هزینه ترین گره تا الان را گسترش می دهد.
- تابع هزینه: $g(n)$ (هزینه واقعی از شروع تا گره n)

▪ جستجوی حریصانه:

- گره ای را گسترش می دهد که از الان تا آینده کم هزینه تر به نظر می رسد.
- تابع هزینه: $h(n)$ (تخمین هزینه باقیمانده تا هدف)

□ تابع ارزیابی، یک تابع هزینه ترکیبی است

▪ $g(n)$: هزینه‌ی واقعی مسیر پیموده شده از ریشه تا گره n .

▪ $h(n)$: هزینه تخمینی کوتاهترین مسیر از گره n تا هدف

▪ $f(n)$: هزینه تخمینی کوتاهترین مسیر راه حلی که از گره n می گذرد.

جستجوی A^*

- الگوریتم جستجوی A^* علاوه بر تابع هیوریستیک $h(n)$ ، هزینه واقعی طی شده تا گره (n) را نیز استفاده می‌کند. $(f(n) = g(n) + h(n))$
- هر گره‌ای که مقدار $f(n)$ آن کمتر باشد، زودتر بررسی و گسترش داده می‌شود.

مراحل اجرای الگوریتم:

۱. شروع از گره اولیه (Start Node):

- گره شروع را در صفات اولویت قرار می‌دهیم.
- مقدار تابع $f(n) = g(n) + h(n)$ را برای گره محاسبه می‌کنیم.

۲. بررسی گره‌های مجاور (Expansion of Nodes):

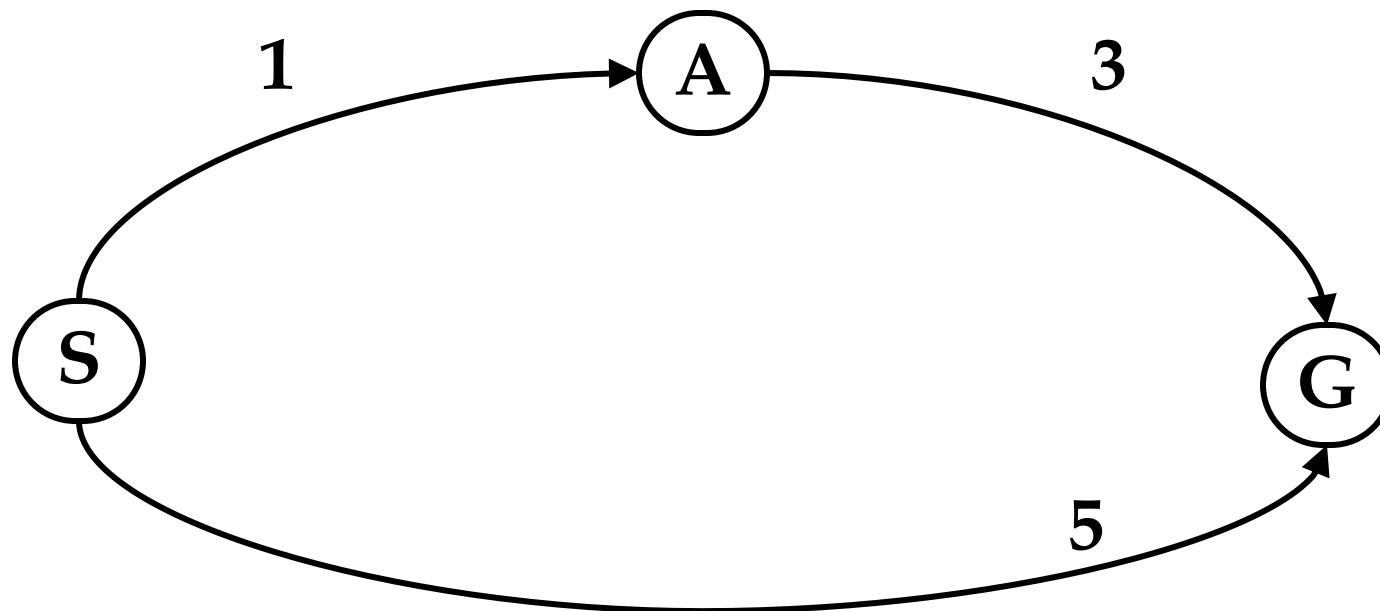
- از صفات اولویت، گره‌ای را که کمترین مقدار $f(n)$ دارد، انتخاب می‌کنیم.
- اگر این گره هدف باشد، جستجو متوقف می‌شود.
- در غیر این صورت، همه‌ی فرزندان این گره را تولید و مقدار $f(n)$ آن‌ها را محاسبه می‌کنیم.

۳. مرتب‌سازی مجدد صفات اولویت:

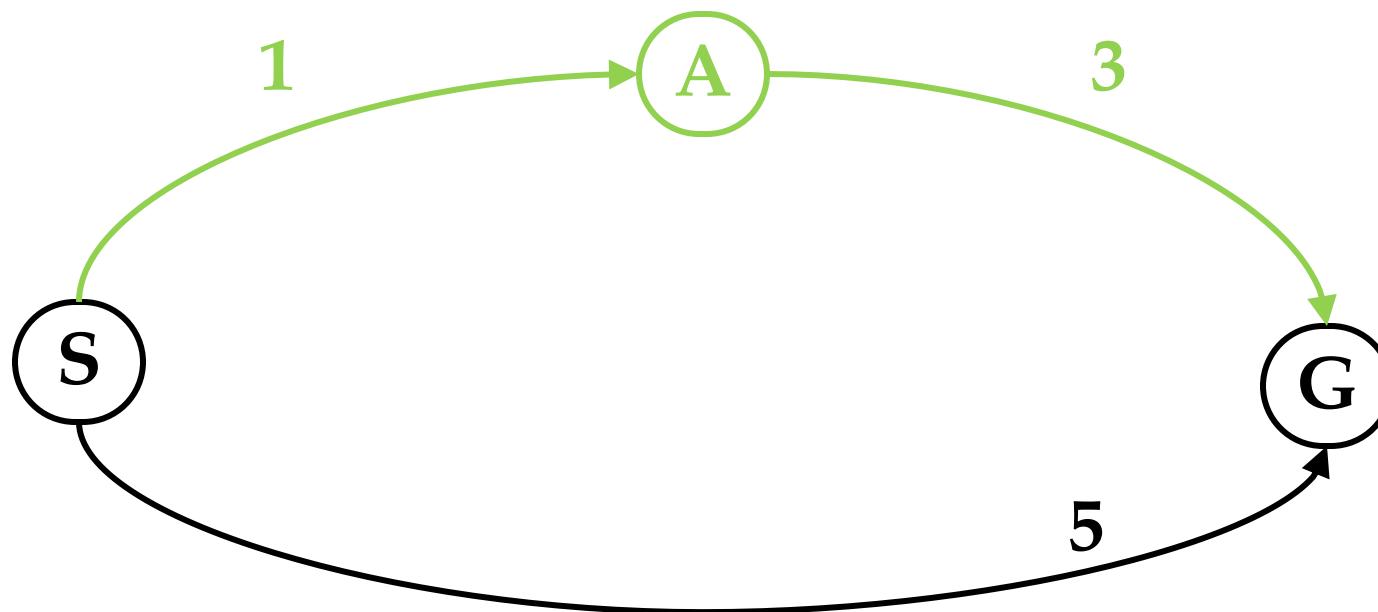
- همه‌ی گره‌های جدید را در صفات اولویت قرار می‌دهیم.
- گره‌ای که کمترین مقدار $f(n)$ دارد، در اولویت انتخاب قرار می‌گیرد.

۴. تکرار مراحل ۲ و ۳ تا رسیدن به هدف یا خالی شدن صفات اولویت

بررسی بهینگی A^*

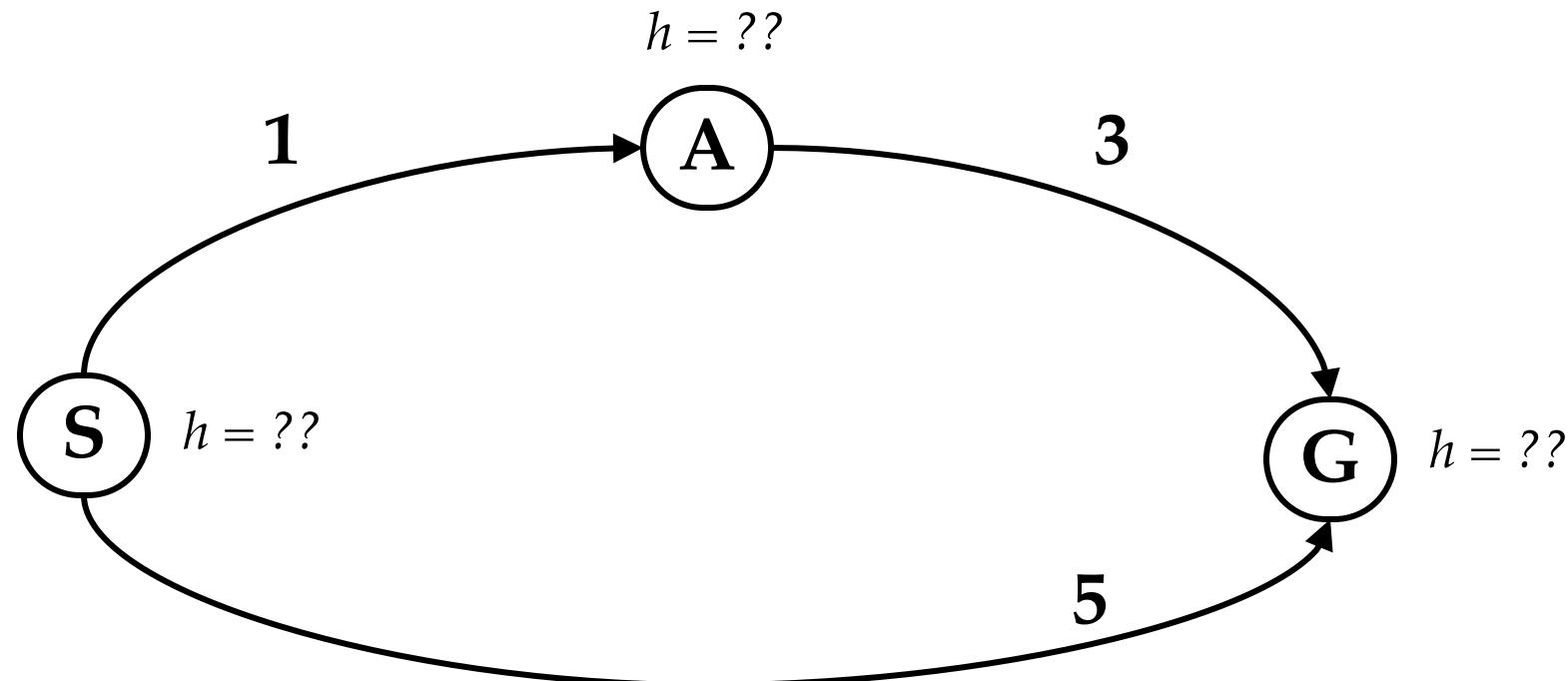


بررسی بھینگی A^*



بررسی بهینگی A^*

g: هزینه‌ی پس‌رو (backward cost): هزینه‌ای که تا الان از گره شروع (S) تا گره فعلی طی شده است.
h: هزینه‌ی پیش‌رو (forward cost): هزینه‌ی تخمینی از الان تا آینده از گره فعلی تا گره هدف.



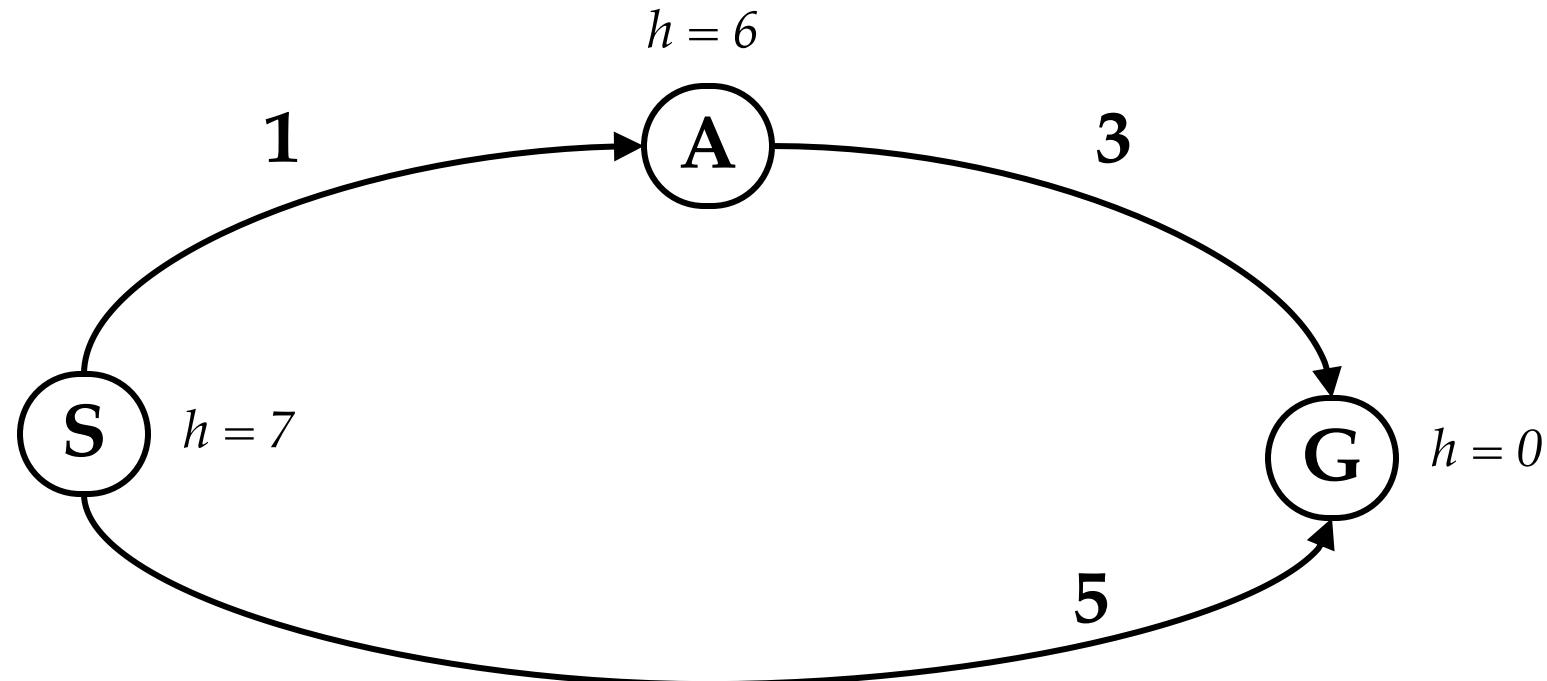
بررسی بهینگی A*

سوال: هیوریستیک‌ها از کجا می‌آیند؟

پاسخ: ما باید آن‌ها را ایجاد کنیم!

g: هزینه‌ی پس‌رو (backward cost): هزینه‌ای که تا الان از گره شروع (S) تا گره فعلی طی شده است.

h: هزینه‌ی پیش‌رو (forward cost): هزینه‌ی تخمینی از الان تا آینده از گره فعلی تا گره هدف.



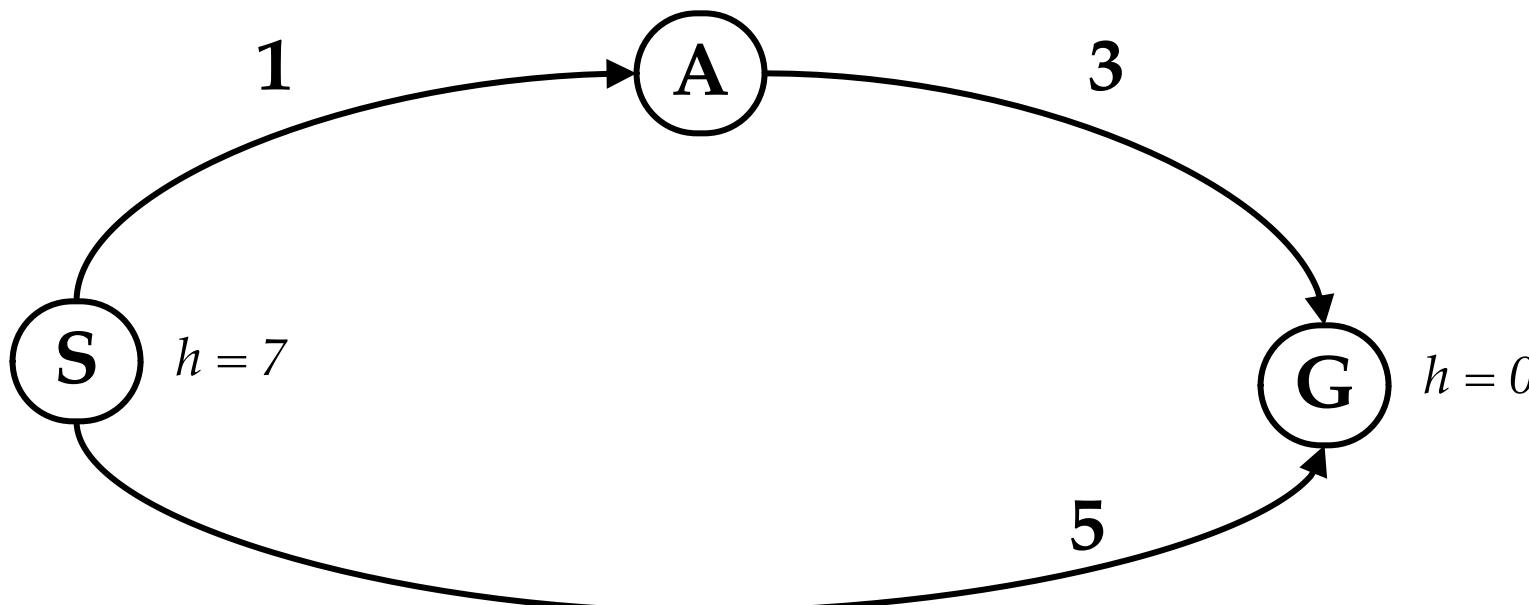
Not the best heuristic...

بررسی بهینگی A^*

g: هزینه‌ی پس رو (backward cost): هزینه‌ای که تا الان از گره شروع (S) تا گره فعلی طی شده است.

$$h = 6$$

h: هزینه‌ی پیش رو (forward cost): هزینه‌ی تخمینی از الان تا آینده از گره فعلی تا گره هدف.



□ چه مشکلی پیش آمد؟

■ نیاز داریم که تخمین‌ها کمتر یا مساوی از هزینه‌های واقعی باشند!

$7 \cancel{=} < 5$

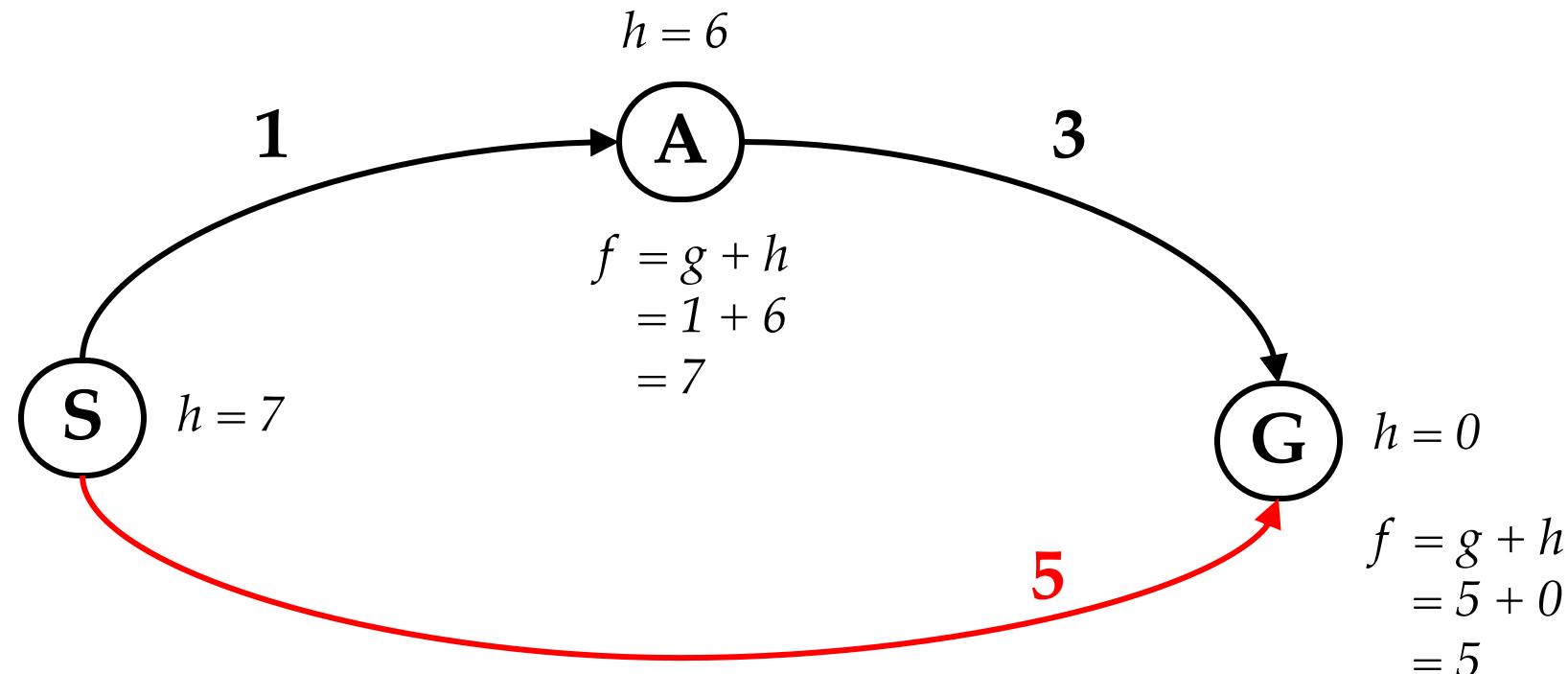
■ در اینجا تخمین 7 و هزینه واقعی 5 بود!

بررسی بهینگی A*

سوال: هیوریستیک‌ها از کجا می‌آیند؟

پاسخ: ما باید آن‌ها را ایجاد کنیم!

g: هزینه‌ی پس‌رو (backward cost): هزینه‌ای که تا الان از گره شروع (S) تا گره فعلی طی شده است.
 h: هزینه‌ی پیش‌رو (forward cost): هزینه‌ی تخمینی از الان تا آینده از گره فعلی تا گره هدف.



Not the best heuristic...

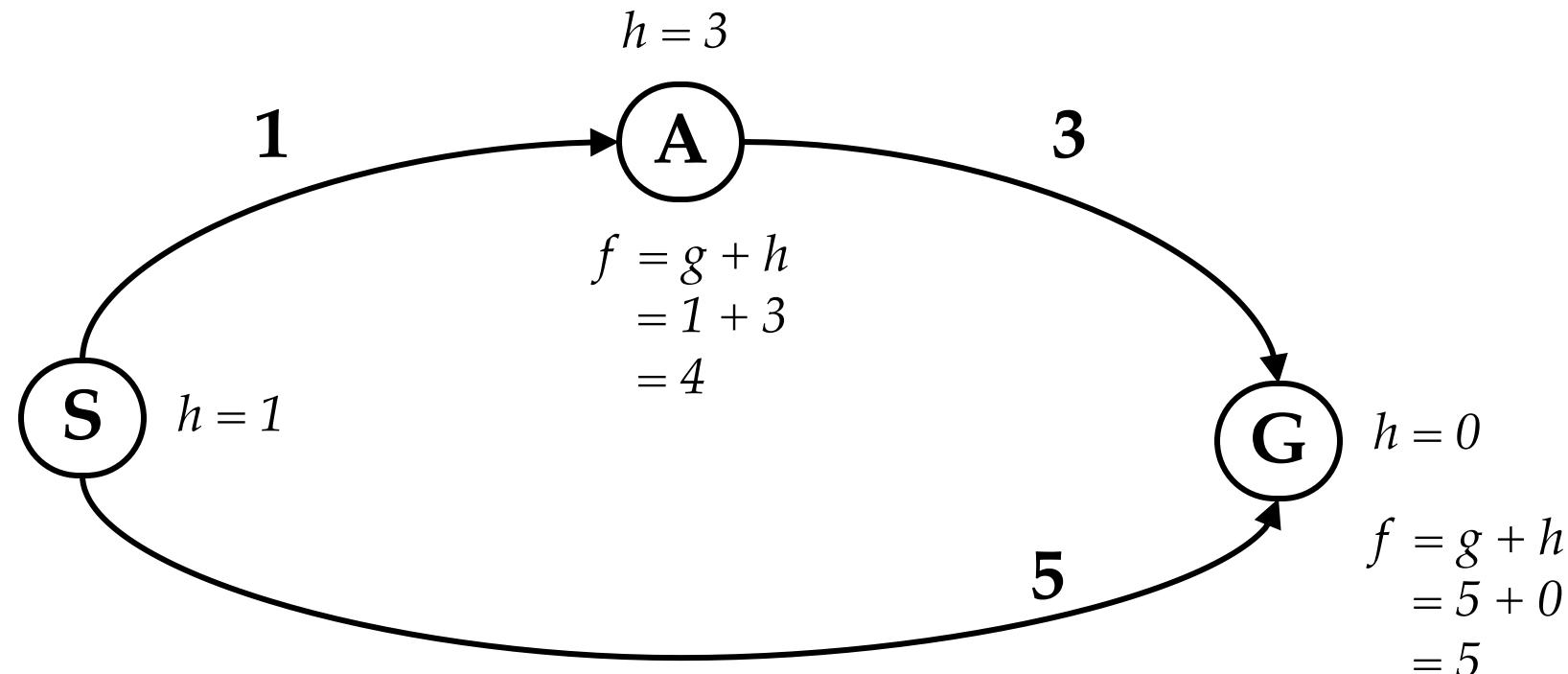
بررسی بهینگی A*

سوال: هیوریستیک‌ها از کجا می‌آیند؟

پاسخ: ما باید آن‌ها را ایجاد کنیم!

g: هزینه‌ی پس‌رو (backward cost): هزینه‌ای که تا الان از گره شروع (S) تا گره فعلی طی شده است.

h: هزینه‌ی پیش‌رو (forward cost): هزینه‌ی تخمینی از الان تا آینده از گره فعلی تا گره هدف.



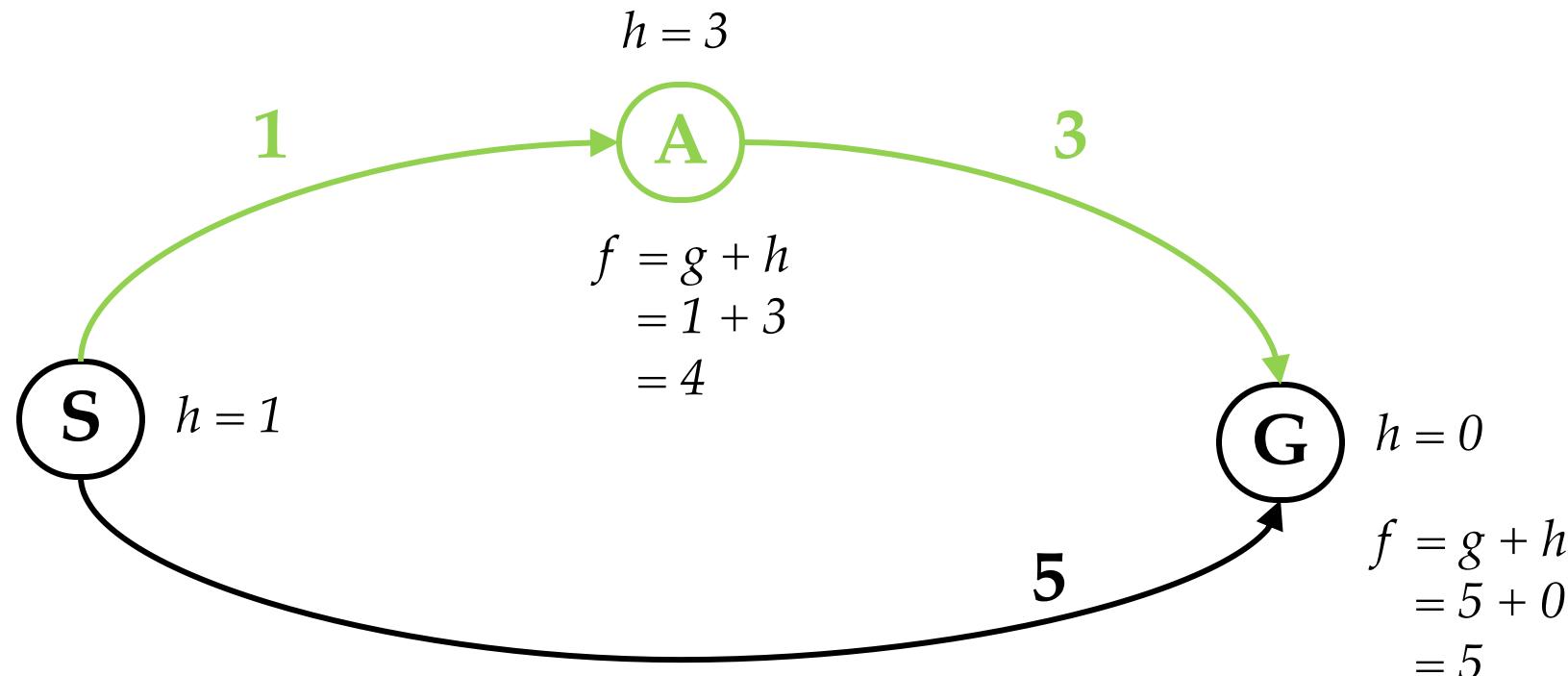
What's a better heuristic?

بررسی بهینگی A*

سوال: هیوریستیک‌ها از کجا می‌آیند؟

پاسخ: ما باید آن‌ها را ایجاد کنیم!

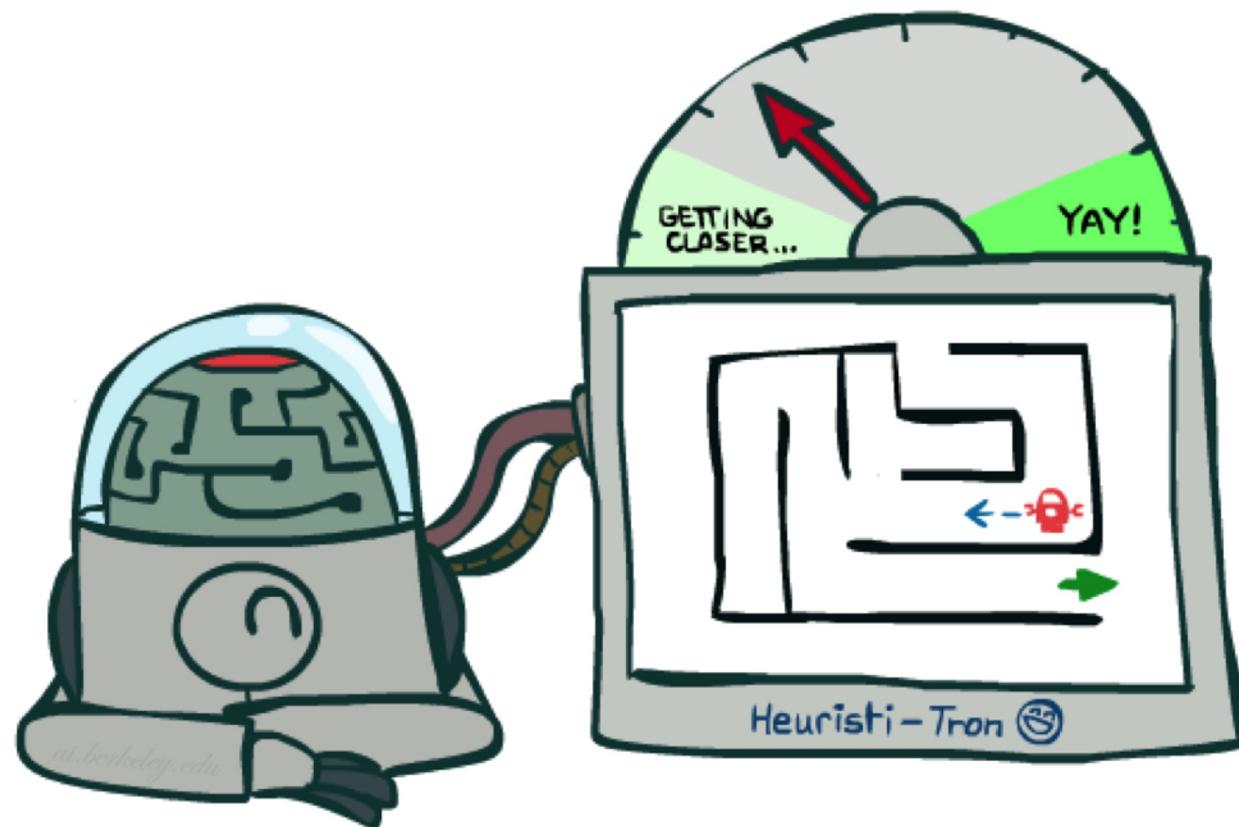
g: هزینه‌ی پس‌رو (backward cost): هزینه‌ای که تا الان از گره شروع (S) تا گره فعلی طی شده است.
h: هزینه‌ی پیش‌رو (forward cost): هزینه‌ی تخمینی از الان تا آینده از گره فعلی تا گره هدف.



What's a better heuristic?

هیوریستیک قابل قبول (Admissible) = هزینه را از هر گره تا هدف کمتر یا مساوی از مقدار واقعی تخمین می‌زند.

هیوریستیک‌های قابل قبول

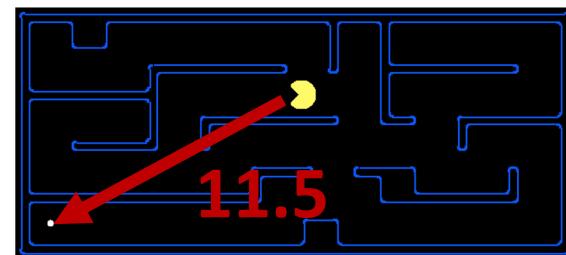
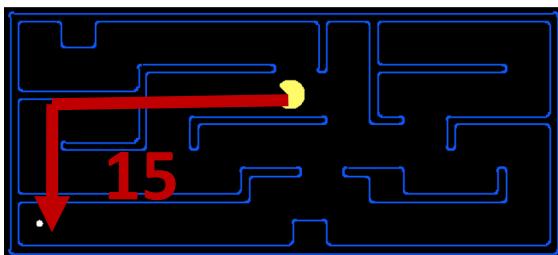


هیوریستیک‌های قابل قبول

- جستجوی A^* به شرطی بهینه است که تابع هیوریستیک قابل قبول باشد.
- تابع هیوریستیک h قابل قبول (خوش‌بینانه) است اگر و فقط اگر:

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

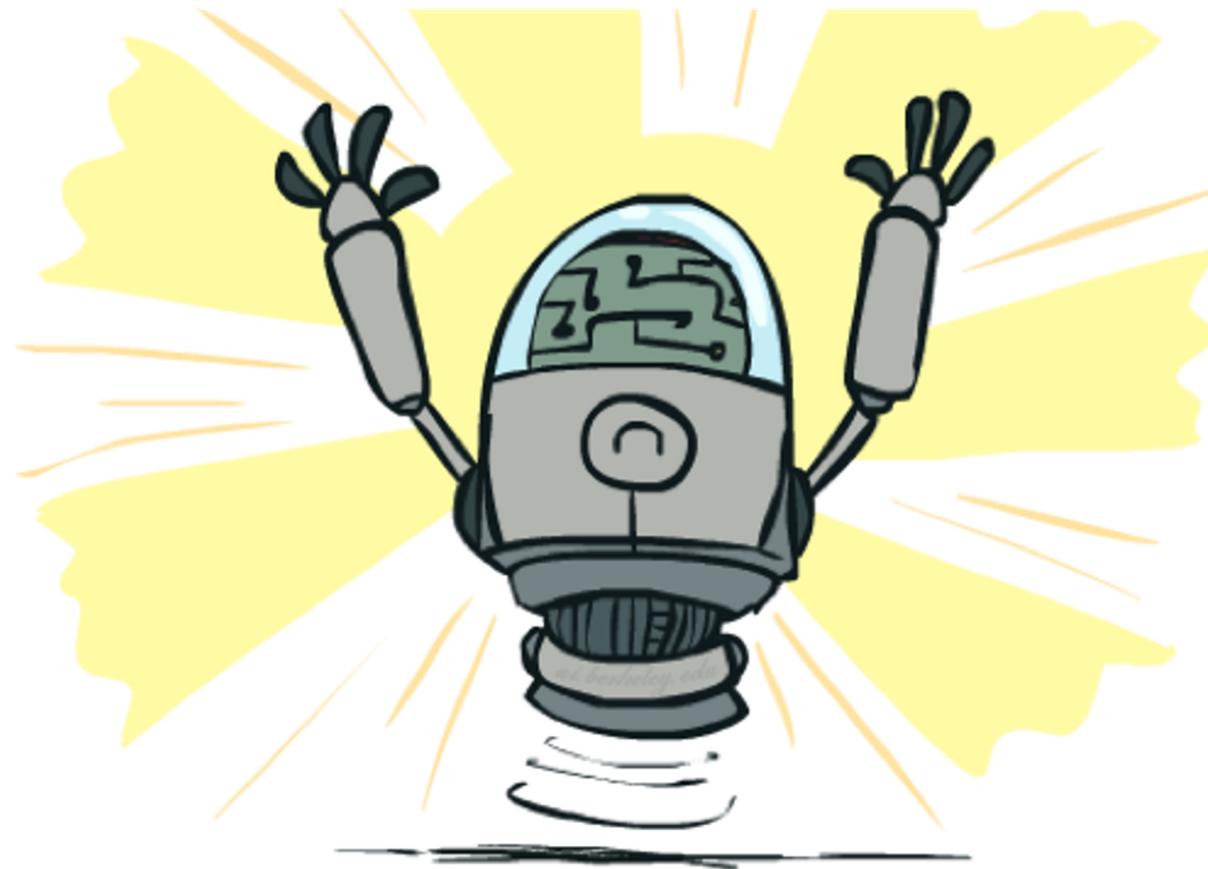
وقتی $h^*(n)$ بیانگر هزینه واقعی کوتاهترین مسیر از n تا هدف است.



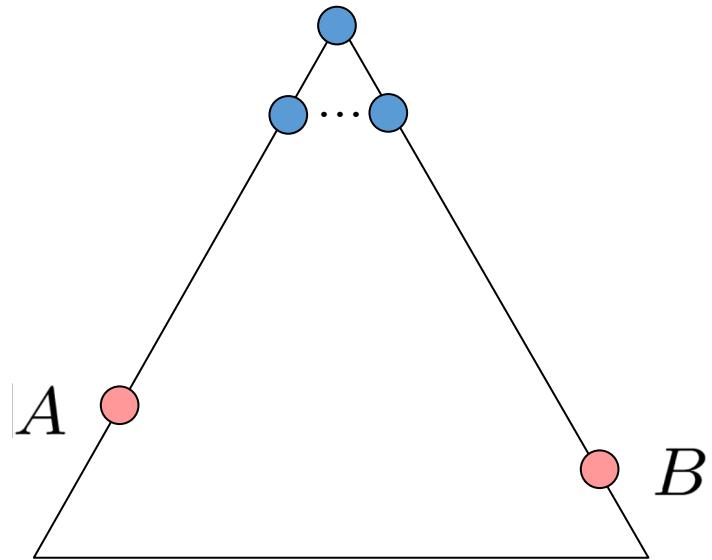
□ مثال:

- پیدا کردن هیوریستیک‌های قابل قبولی که هم خوش بینانه باشد و هم نزدیک به h^* کلید موفقیت است.
- (بیش از حد خوش بینانه هم خوب نیست!)

اثبات برهینگی * A^* در جستجوی درختی



اثبات برهینگی A^* در جستجوی درختی



□ قضیه:

- اگر تابع هیوریستیک h قابل قبول باشد، آنگاه A^* در جستجوی درختی برهینه است.

□ فرضیات:

- A یک گره هدف برهینه است.
- B یک گره هدف غیر برهینه (SubOptimal) است.
- h یک تابع هیوریستیک قابل قبول است.

□ ادعا

- قبل از B از حاشیه جستجو (fringe) خارج خواهد شد

اثبات بهینگی $*A$ در جستجوی درختی

□ اثبات:

- فرض کنید B در حاشیه (fringe) قرار دارد.

- همچنین فرض کنید که یک جدای A که آن را n می‌نامیم، نیز در حاشیه باشد

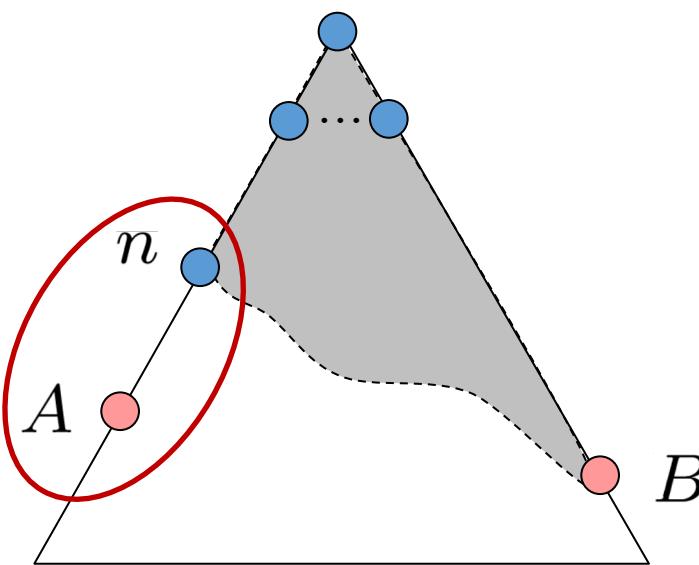
□ ادعا:

- n قبل از B بسط داده خواهد شد.

اثبات می‌کنیم $f(n) \leq f(A)$ کوچکتر یا مساوی است.

اثبات می‌کنیم $f(A) < f(B)$ کوچکتر از $f(B)$ است.

$$f(n) \leq f(A) < f(B)$$



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

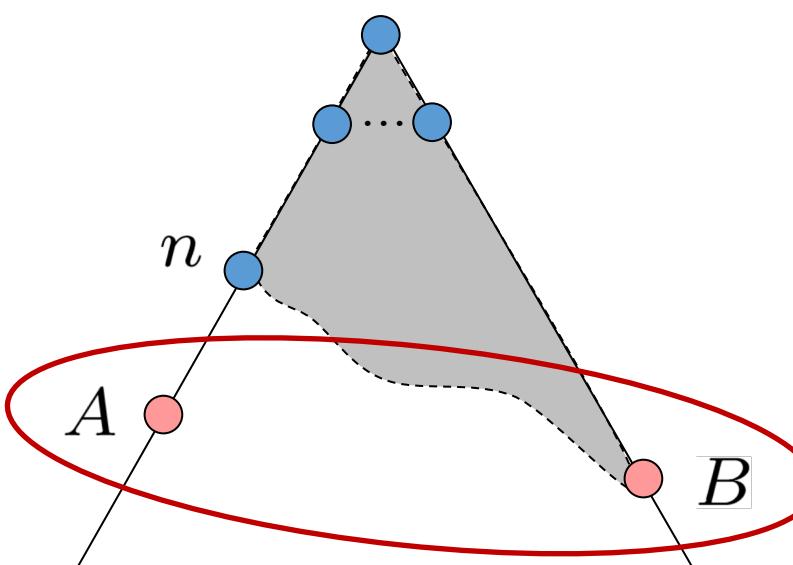
$$f(A) = g(A) + h(A) \stackrel{0}{=} g(A)$$

$$= g(n) + h^*(n)$$

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

$$g(n) + h(n) \leq g(n) + h^*(n)$$

$$f(n) \leq f(A)$$



$$g(A) < g(B)$$

$$h(A) + g(A) < h(B) + g(B)$$

$$f(A) < f(B)$$

$$f(n) < f(A)$$

$$f(n) \leq f(A)$$

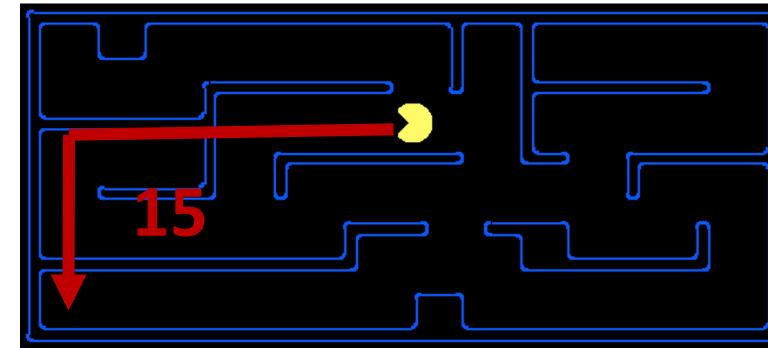
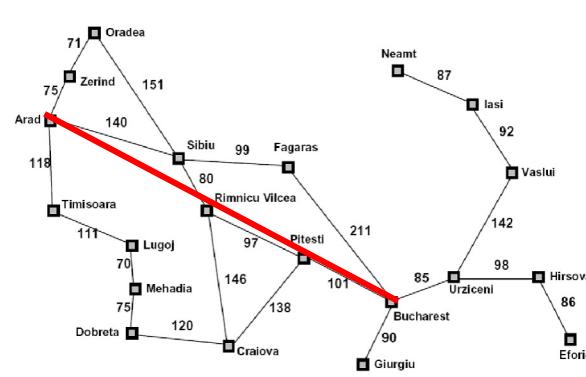
$$f(A) < f(B)$$

ایجاد هیوریستیک‌های قابل قبول

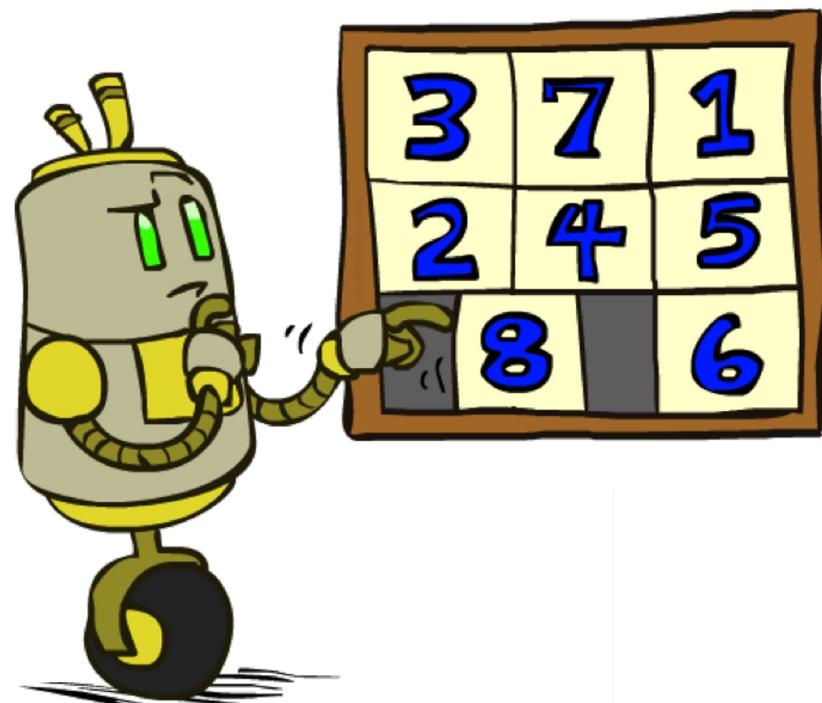
□ در اغلب موارد، توابع هیوریستیک قابل قبول با حل کردن مسائل ساده شده به دست می‌آیند.

□ مسئله ساده‌شده (Relaxed Problem): نسخه‌ای از مسئله اصلی که در آن یک یا چند محدودیت حذف شده‌اند.

366



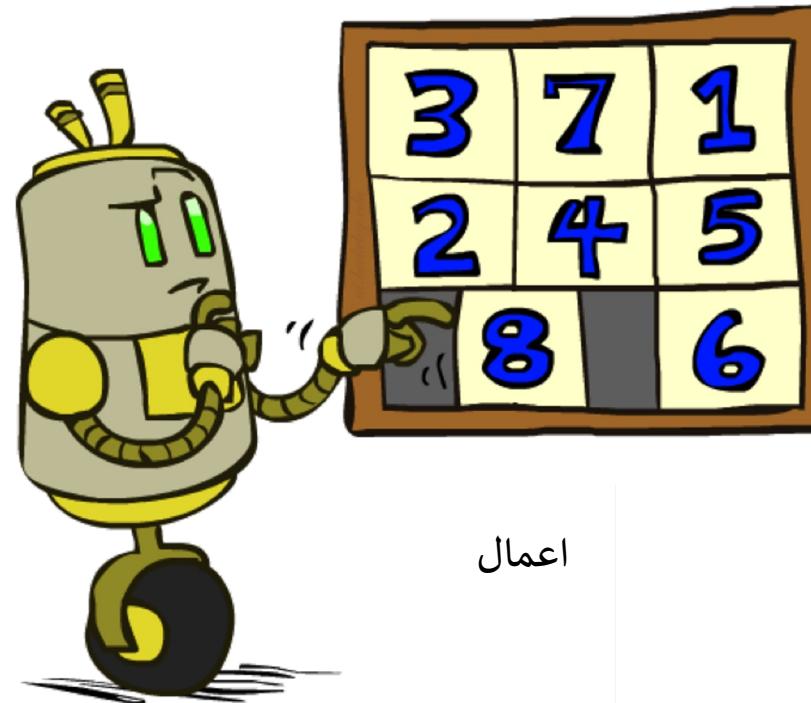
ایجاد هیوریستیک‌های قابل قبول برای پازل ۸ تایی



مثال: پازل ۸ تایی

7	2	4
5		6
8	3	1

حالت شروع



اعمال

	1	2
3	4	5
6	7	8

حالت هدف

تابع هیوریستیک
قابل قبول؟

□ فرموله کردن مسئله پازل ۸ تایی:

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