

# غولهای هوش مصنوعی - دیپ لرینگ

*backpropagation,  
boltzmann machines*



Geoff Hinton  
Google

*convolution*



Yann Lecun  
Facebook

*stacked auto-  
encoders*



Yoshua Bengio  
U. of Montreal

*GPU utilization*



Andrew Ng  
Baidu

# مراكز اصلى تحقیق و توسعه

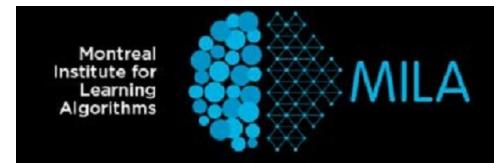
---



Google



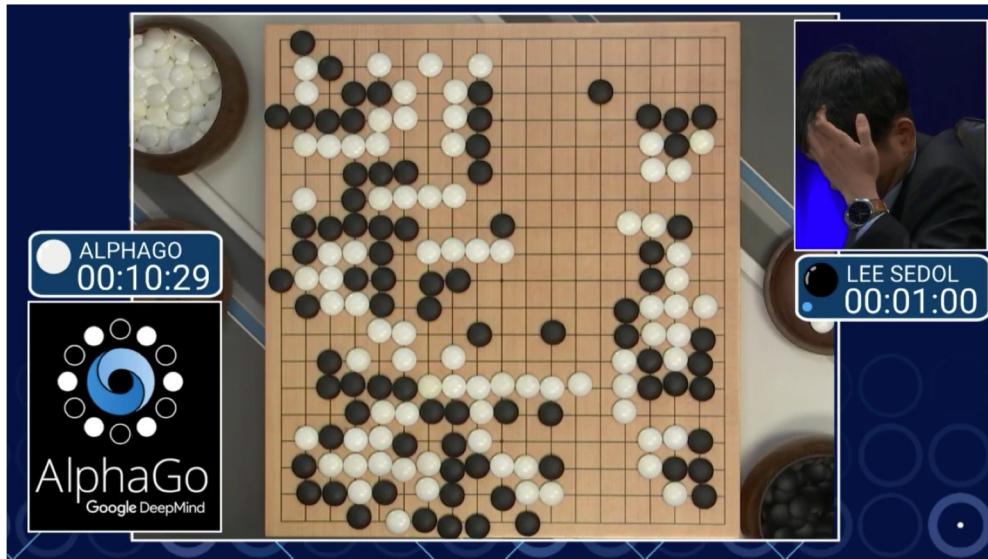
DeepMind



OpenAI

# AlphaGo - DeepMind

---



# OpenAI vs Humans – DOTA 2



## OpenAI

# OpenAI vs Humans – DOTA 2

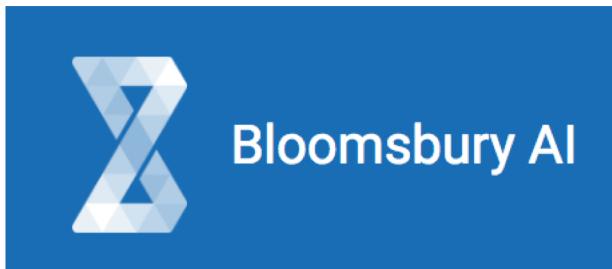


video

# اخبار اخیر

---

Facebook buys British artificial intelligence company Bloomsbury



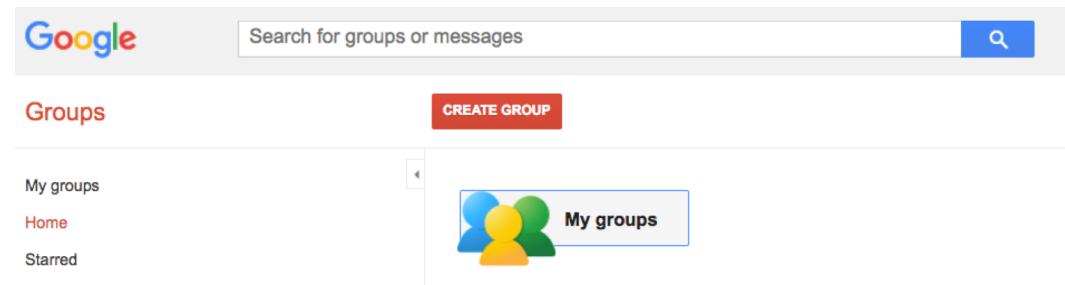
# اخبار اخیر



# جلسه‌ی قبل

صفحه گروه درس

- <https://groups.google.com/forum/#!forum/ai97> (sign up please)



# وبسایت درس

iust-courses.github.io/ai97/



Iran University of Science and Technology  
دانشگاه علم و صنعت ایران

HOME SCHEDULE LECTURES COURSE MATERIALS

## Introduction to Artificial Intelligence (1397)

### Announcements

- New Lecture is up: Introduction to Artificial Intelligence [\[slides\]](#)

### Course Description

This course introduces students to the basic knowledge representation, problem solving, and learning methods of artificial intelligence. You will learn the foundational principles that drive AI applications and practice implementing some of these systems.

Register to our Google groups page to get course notifications via email.

### Course instructor



Mohammad  
Taher  
Pilehvar

### Teaching Assistants



Amirhosein  
Kazemnejad



Soroush  
Gholami



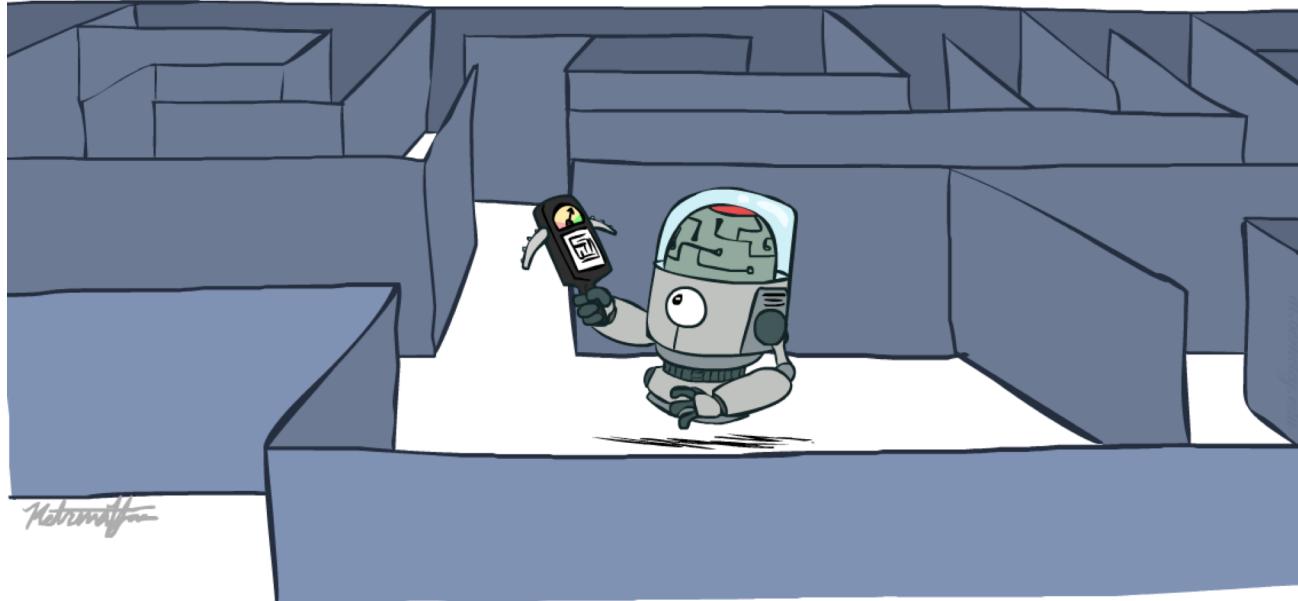
Mostafa  
MohammadAli  
Ebrahim



Kiarash  
Aghakasiri

# هوش مصنوعی

## جستجوی آگاهانه – Informed Search



محمد طاهر پیلهور

[These slides were mostly created by Dan Klein and Pieter Abbeel for CS188 Intro to AI at UC Berkeley]

# امروز

---



جستجوی آگاهانه

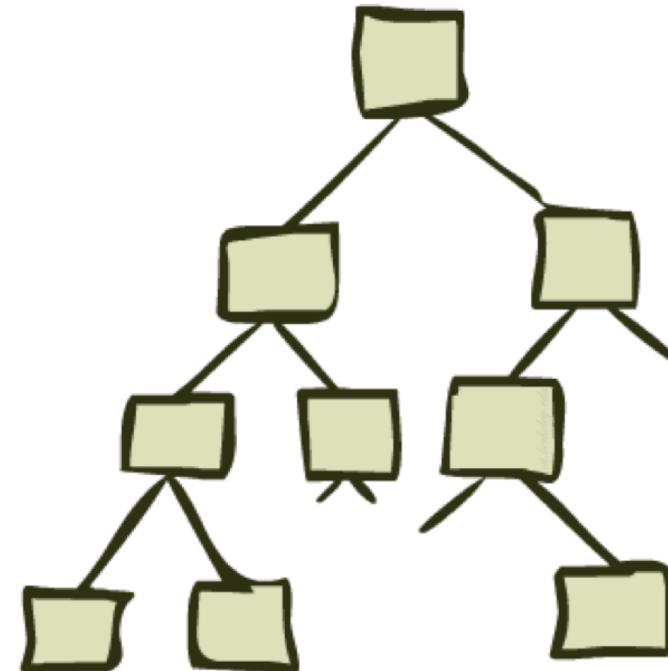
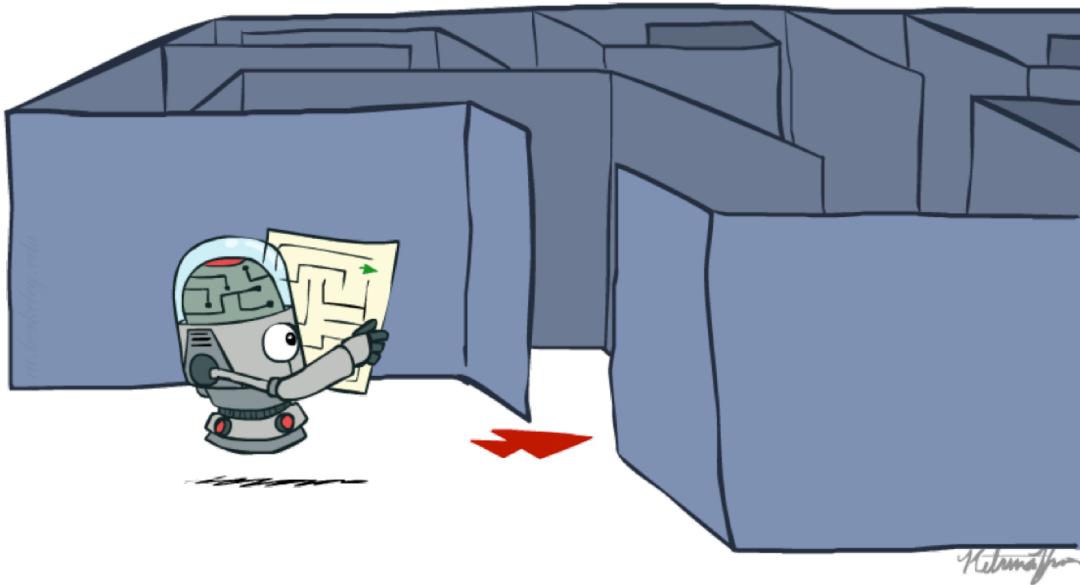
هیوریستیک (اکتشاف)

جستجوی حریصانه

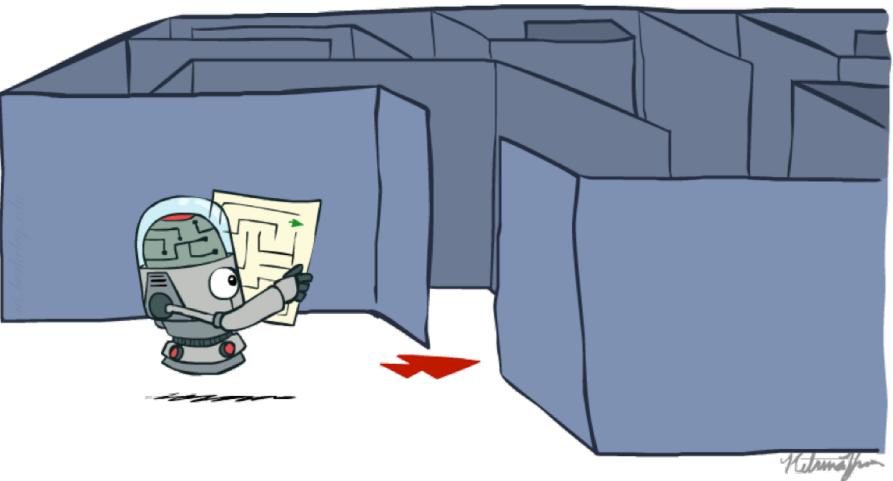
الگوریتم A\*

جستجوی گرافی

# خلاصه - جستجو



# خلاصه - جستجو

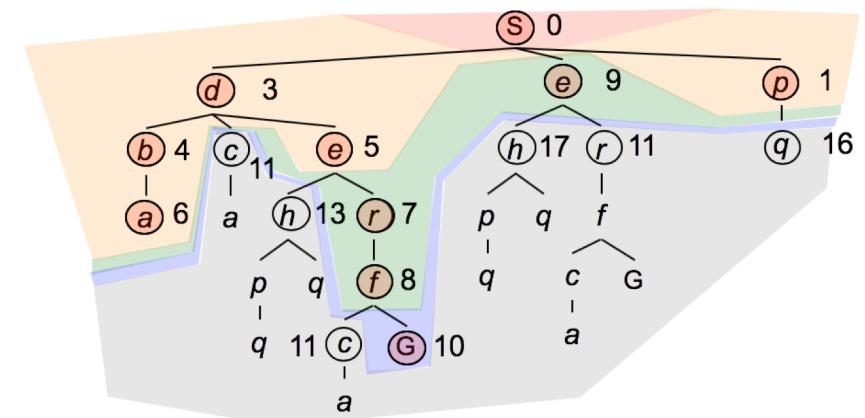
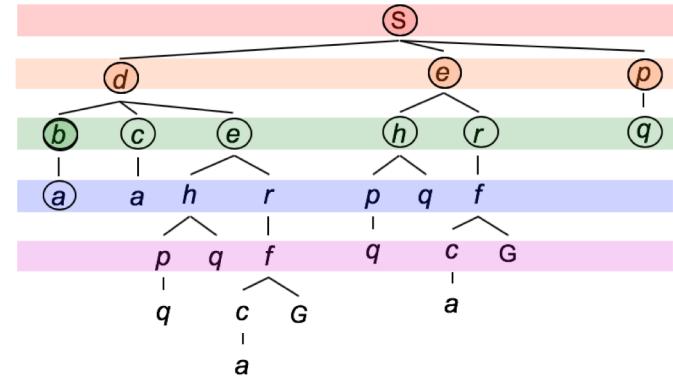
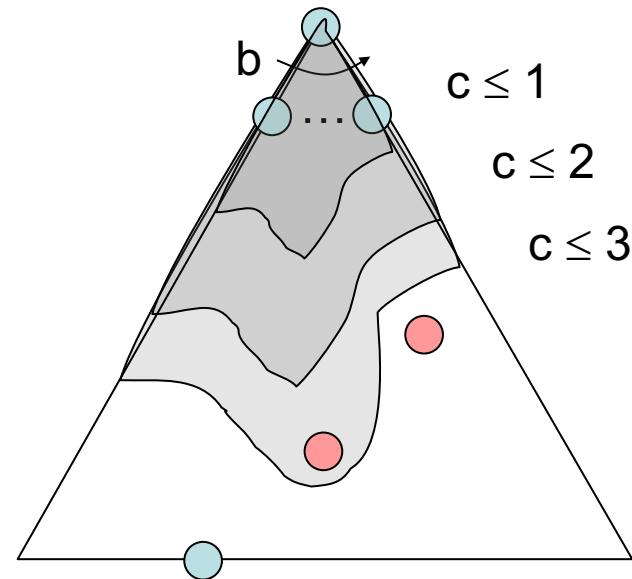
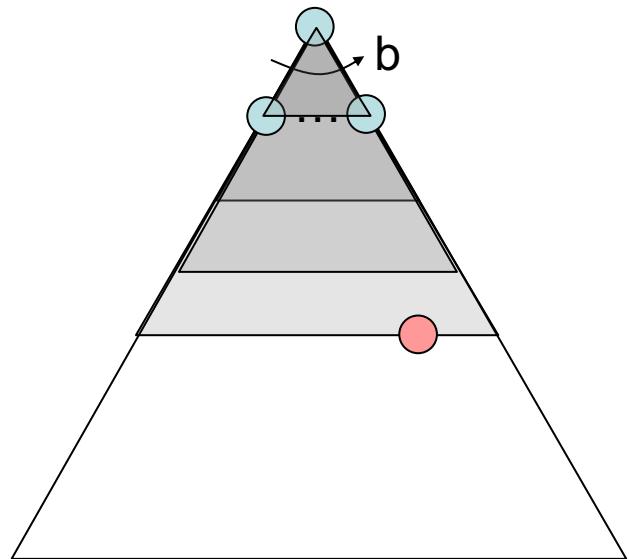
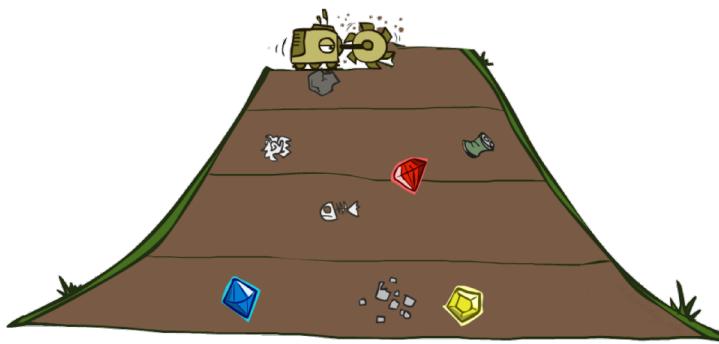


حالات: وضعیت‌های ممکن  
تصمیمات و هزینه‌ها  
تابع پیش‌بر  
حالت شروع و پایان

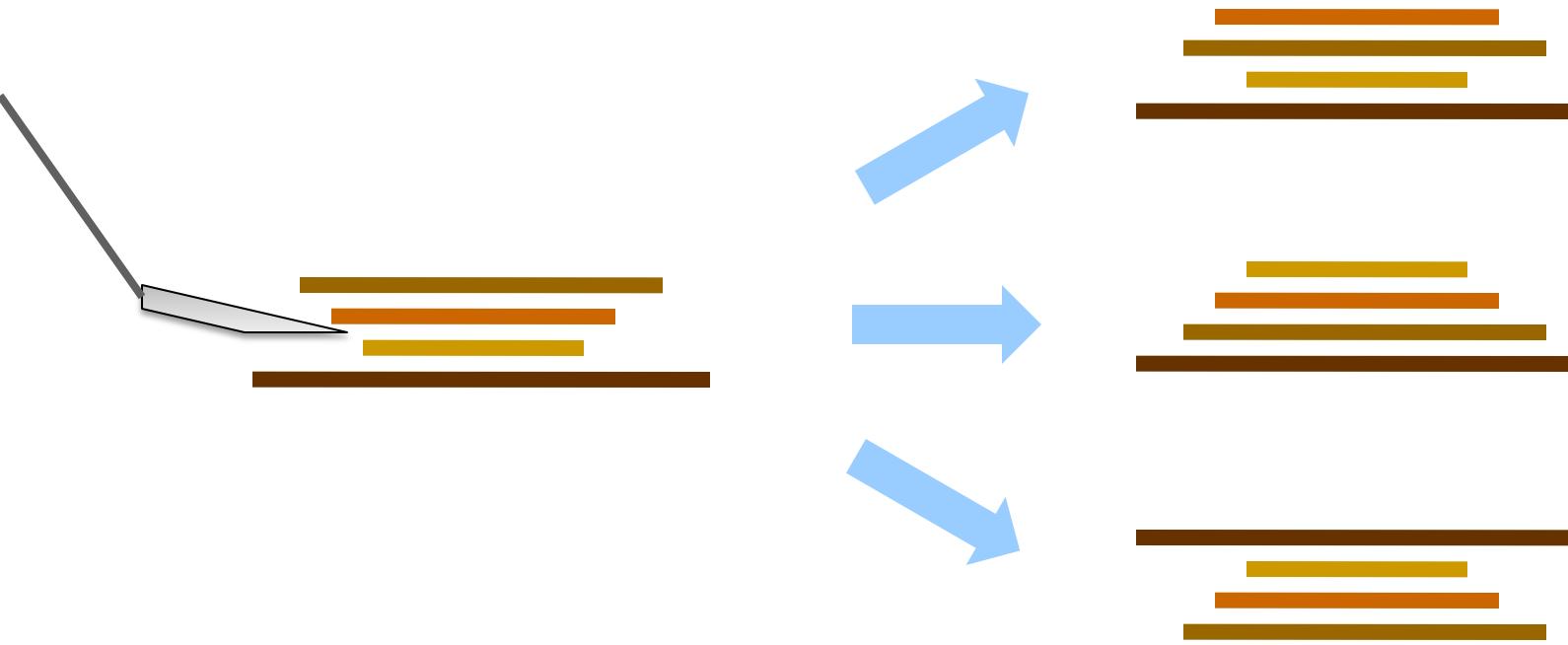
**درخت جستجو**  
گره‌ها تمامی تصمیمات گرفته شده تاکنون را نشان می‌دهند

**الگوریتم جستجو**  
یک درخت جستجو ایجاد می‌کند  
گره‌های موجود در فرینج (که هنوز بسط داده نشده‌اند) را مرتب می‌کند  
بهینه: کم‌هزینه‌ترین راه حل

# خلاصه - جستجو



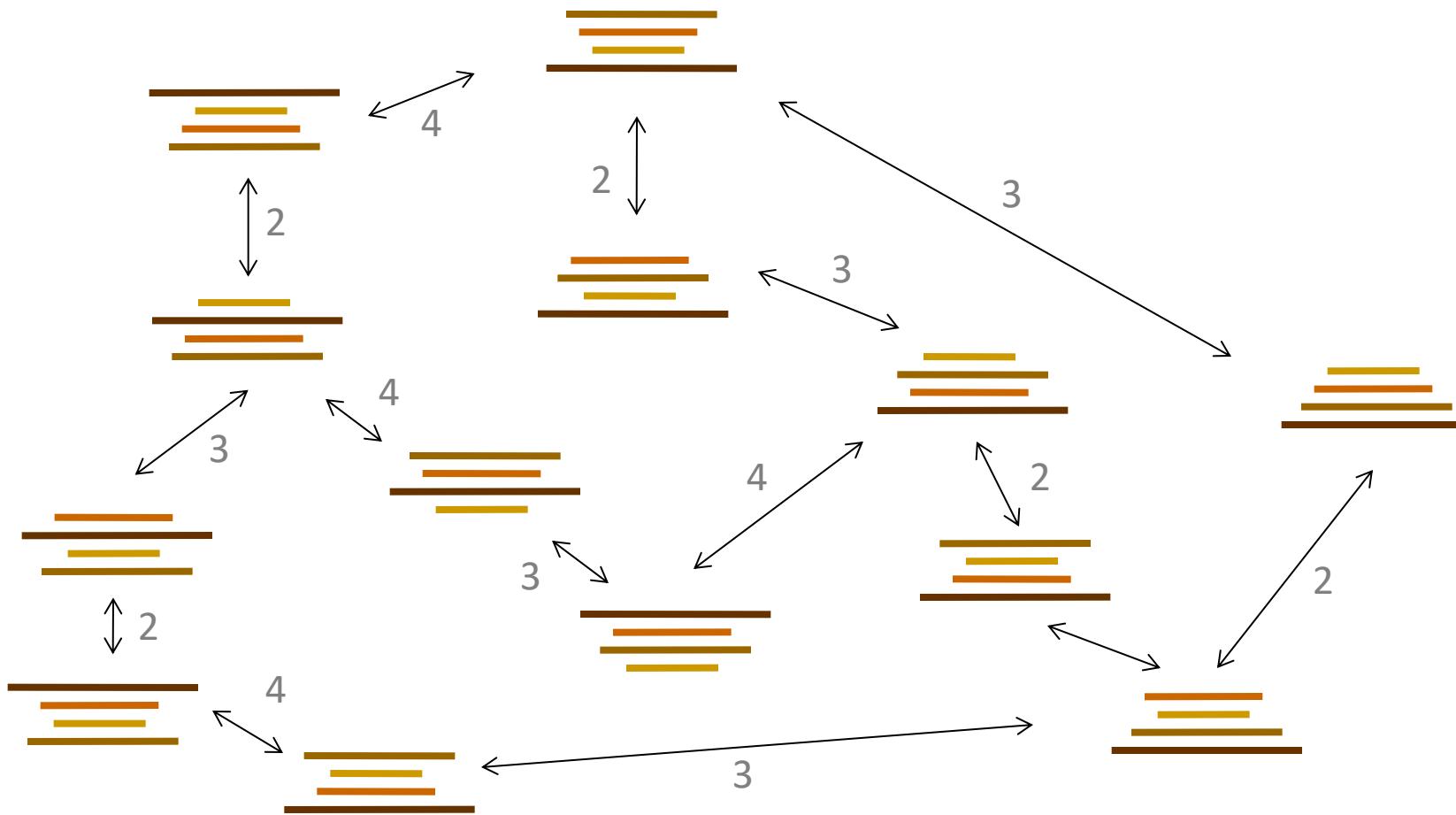
# مثال: پنکیک



هزینه: تعداد پنکیک هایی که زیر و رو می شوند

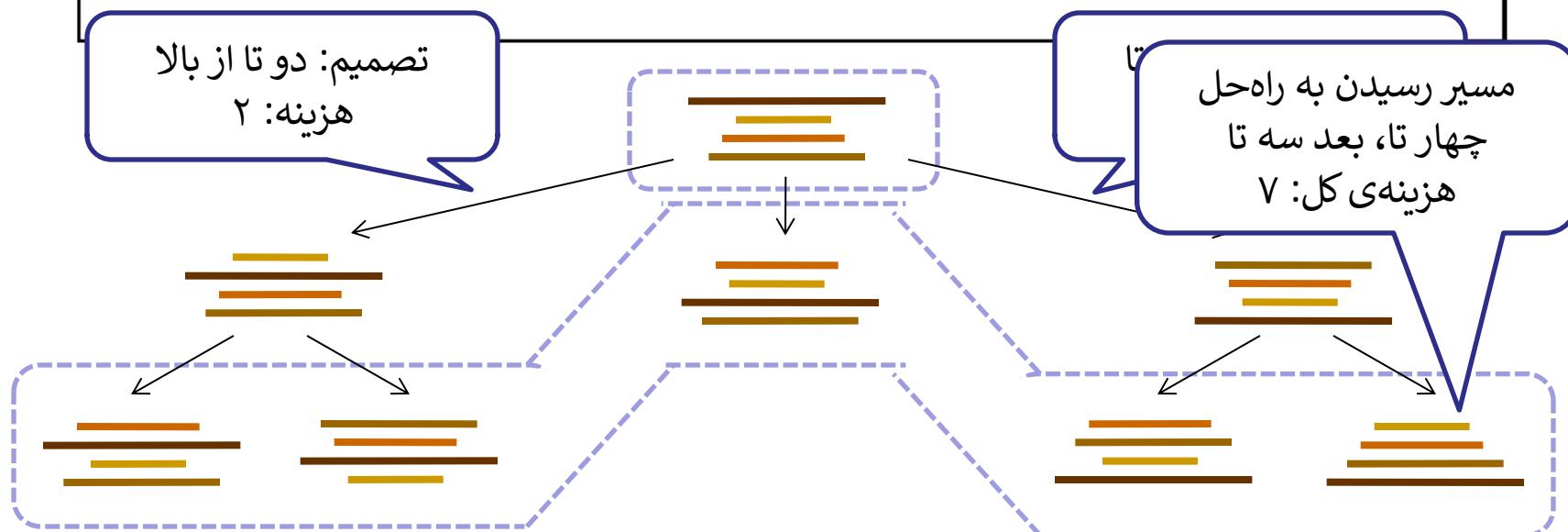
# مثال: پنکیک

فضای حالت با هزینه‌های هر تصمیم

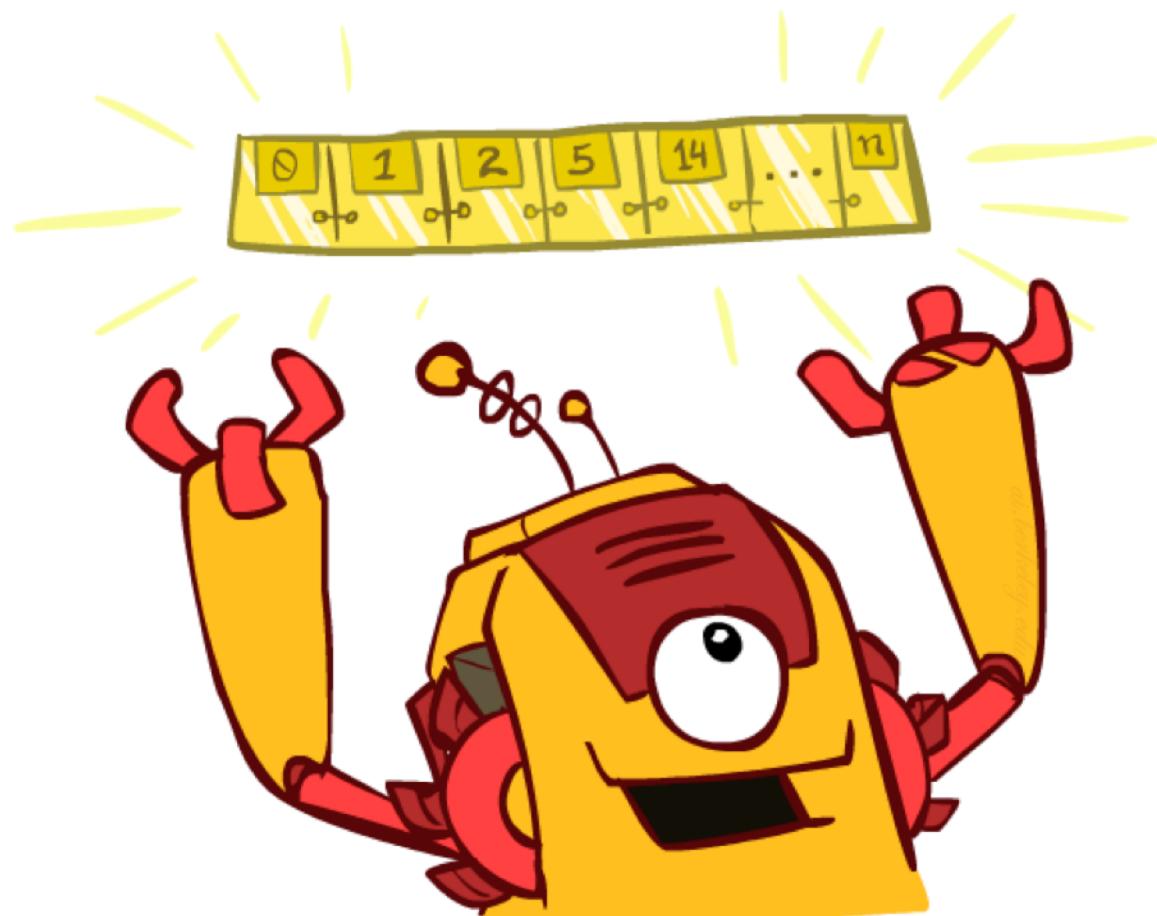


# جستجوی درخت - حالت کلی

```
function TREE-SEARCH(problem, strategy) returns a solution, or failure
    initialize the search tree using the initial state of problem
    loop do
        if there are no candidates for expansion then return failure
        choose a leaf node for expansion according to strategy
        if the node contains a goal state then return the corresponding solution
        else expand the node and add the resulting nodes to the search tree
    end
```



# صف (دنباله)



تنها تفاوت این الگوریتم‌های جستجو در نوع  
صف به کار رفته است

به طور کلی می‌توان برای تمامی این الگوریتم‌ها «صف اولویت‌دار» استفاده کرد (با اولویت‌دهی‌های متفاوت)

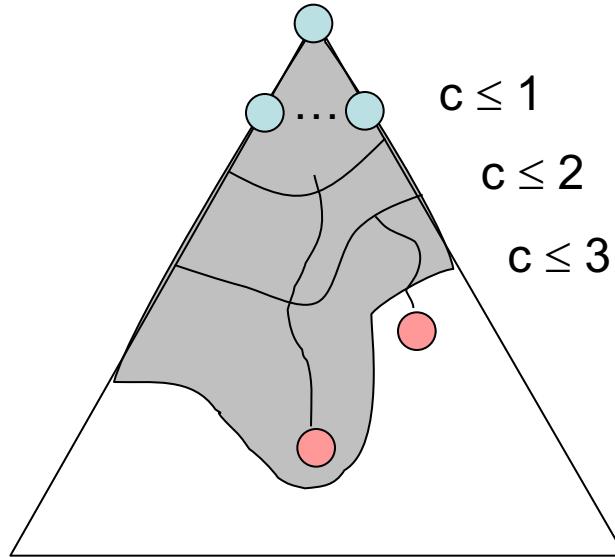
برای اول-سطح و اول-عمق می‌توان «صف» و «استک» (پشته) به کار برد

# جستجوی ناگاهانه

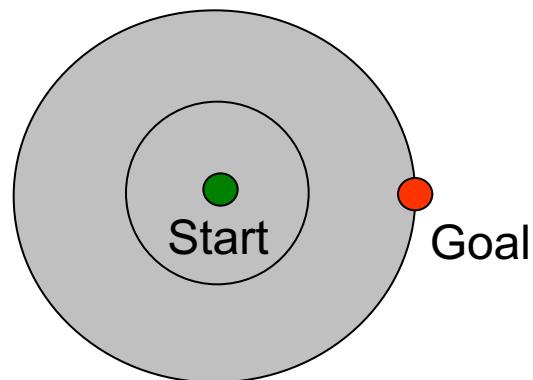
---



# الگوریتم هزینه یکسان



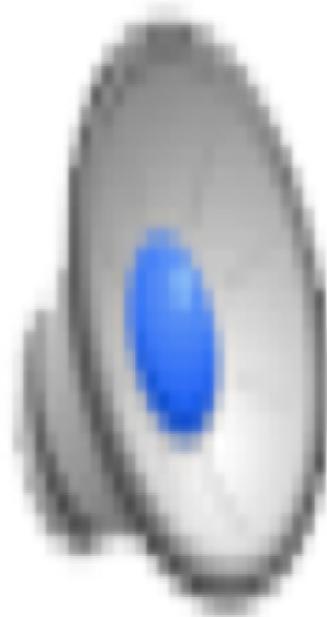
استراتژی: کم‌هزینه‌ترین گره را بسط بد  
کامل و بهینه!



تمامی گره‌ها را در هر جهتی که باشند بسط می‌دهد و  
هیچ ایده‌ای از اینکه راه حل کجا می‌تواند باشد ندارد

# مثال: هزینه یکسان

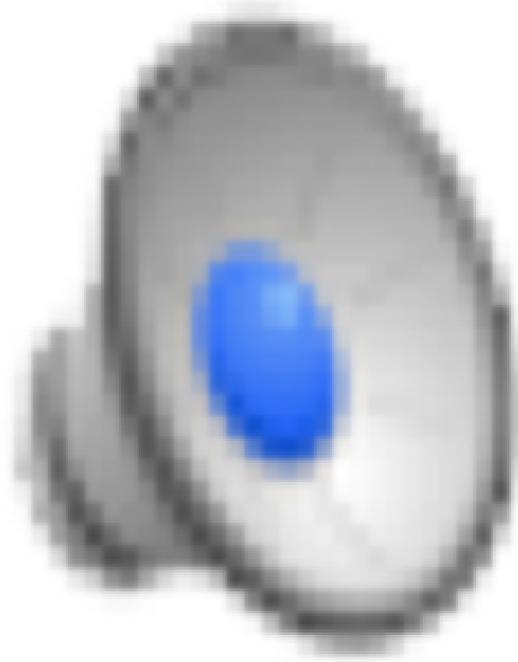
---



video

# مثال: هزینه یکسان (پکمن)

---



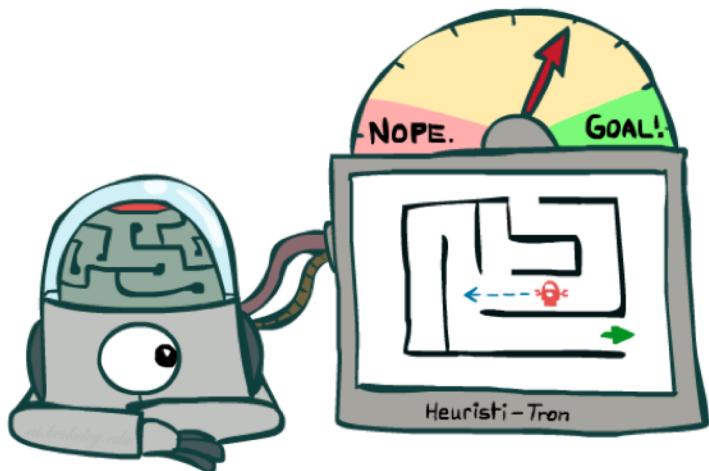
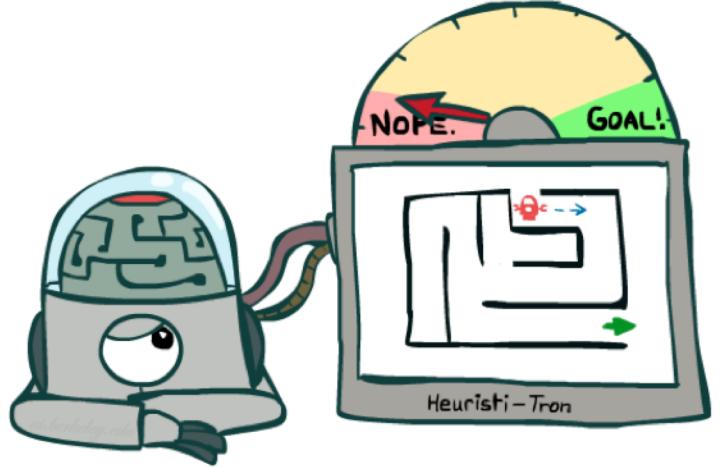
video

# جستجوی آگاهانه!

---

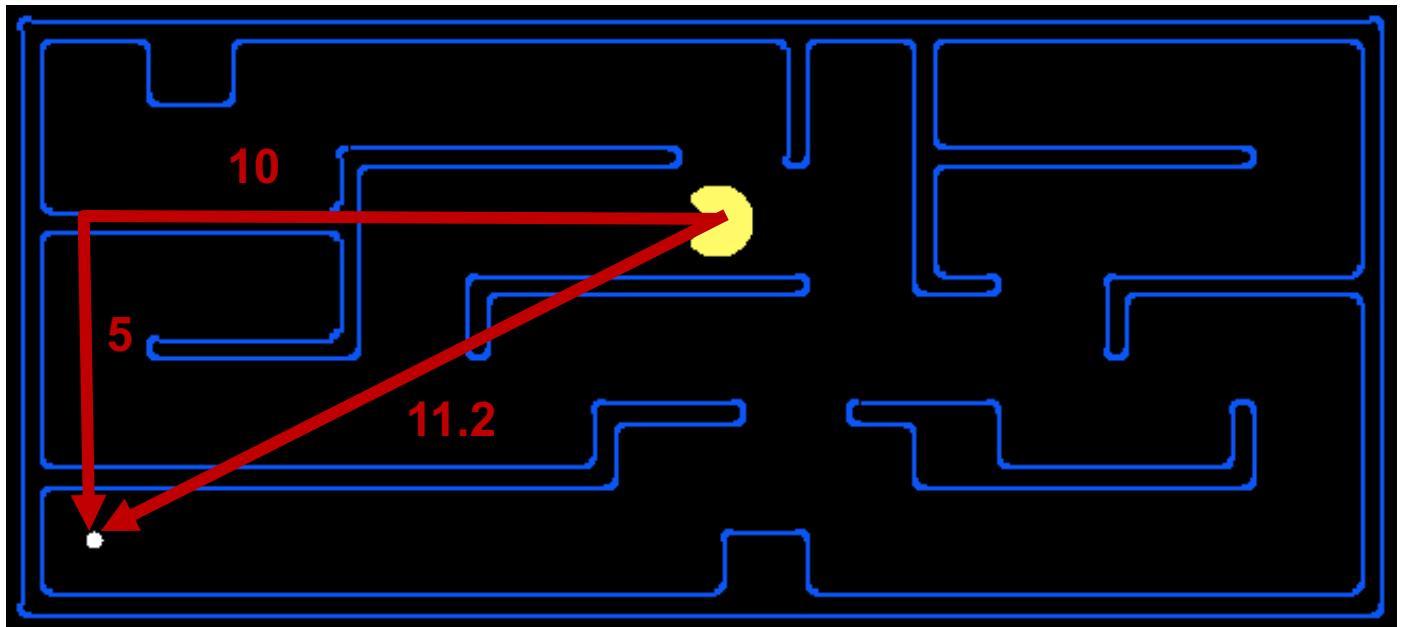


# جستجوی هیوریستیک (اکتشافی)

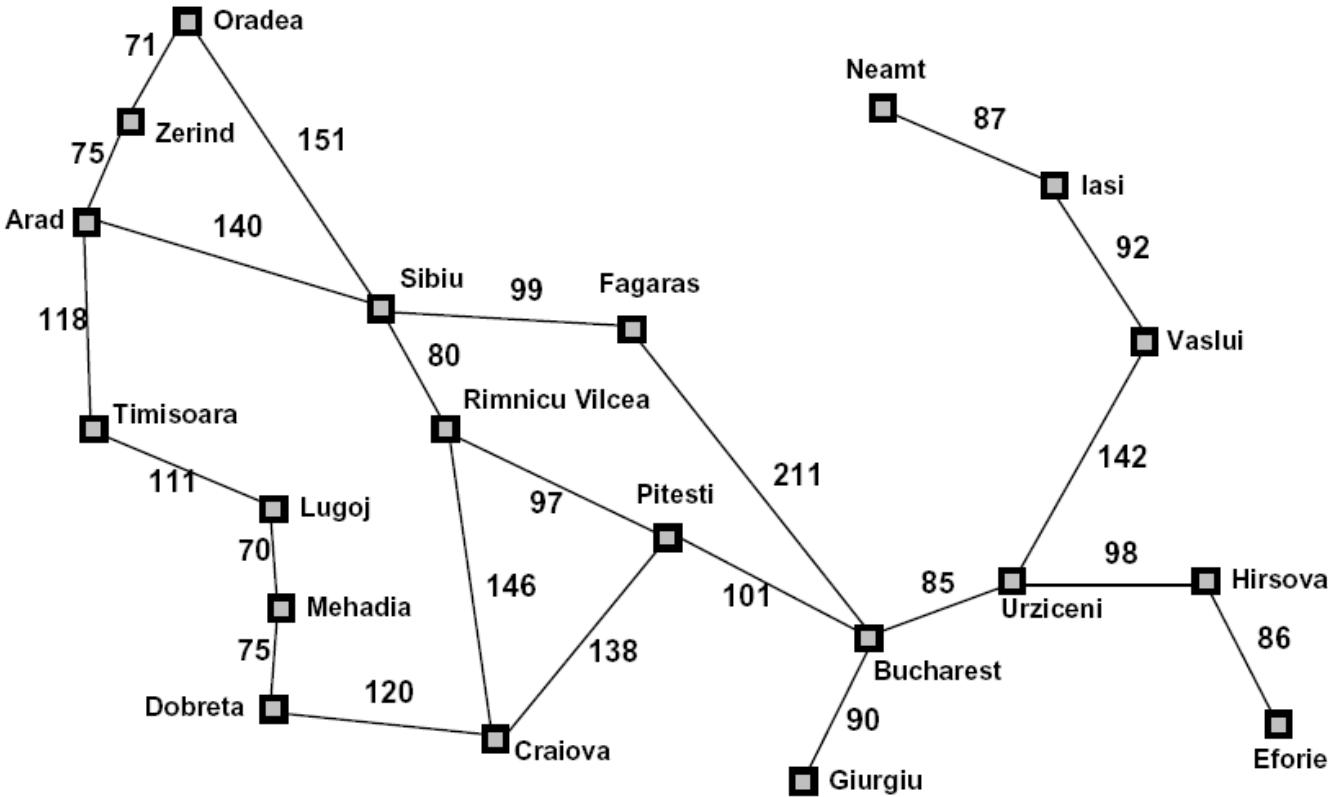


هیوریستیک - heuristic

تابعی که حدس (تخمین) می‌زند چقدر به راه حل نزدیک هستیم  
با توجه به مسئله‌ی جستجو تعیین می‌شود  
مثال: فاصله‌ی اقلیدسی یا منهتن



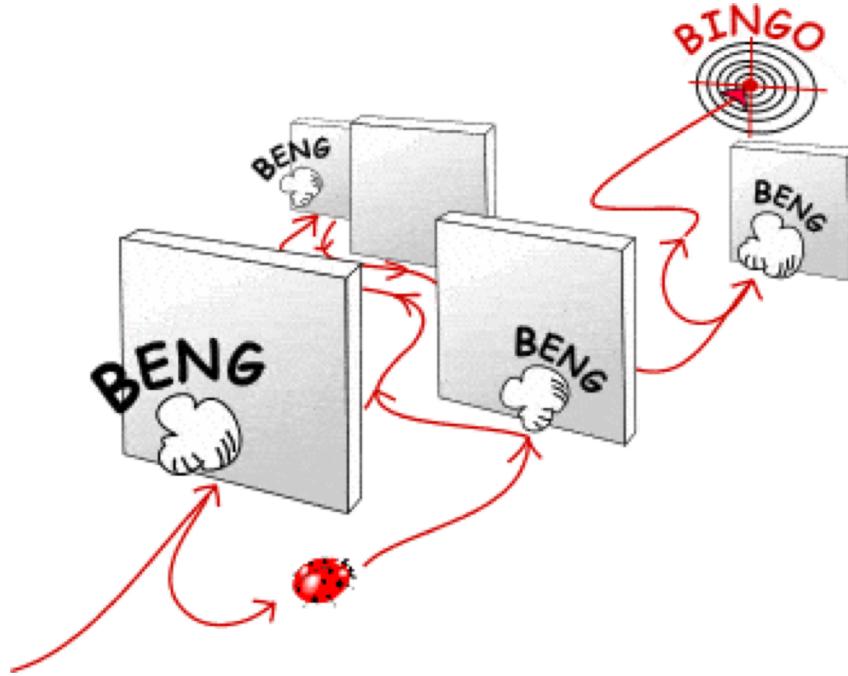
# مثال: تابع هيورنستيك



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

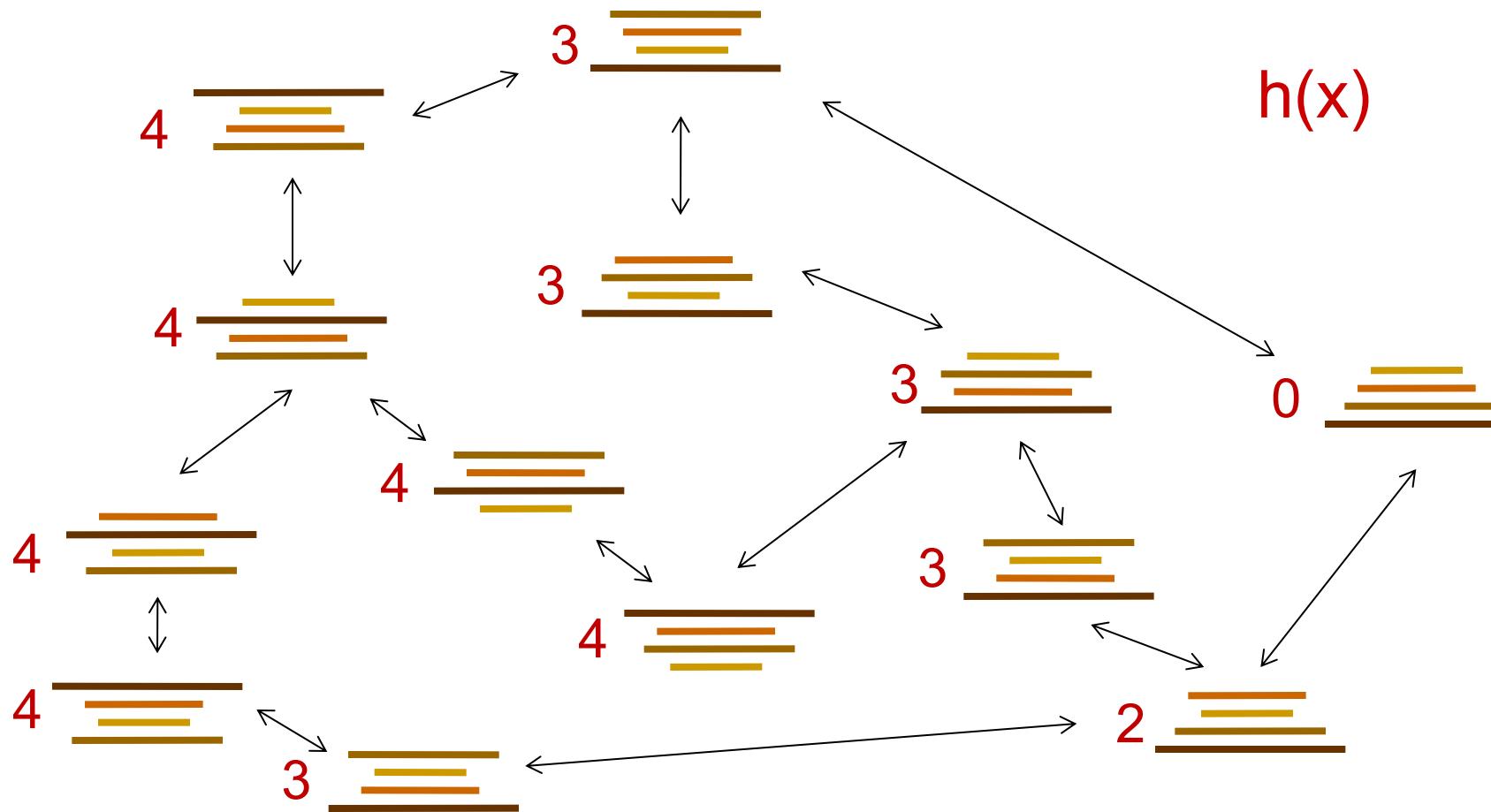
$h(x)$

# تابع هیوریستیک



# مثال: تابع هیوریستیک

هیوریستیک: تعداد پنکیک‌هایی که هنوز در جای خود نیستند

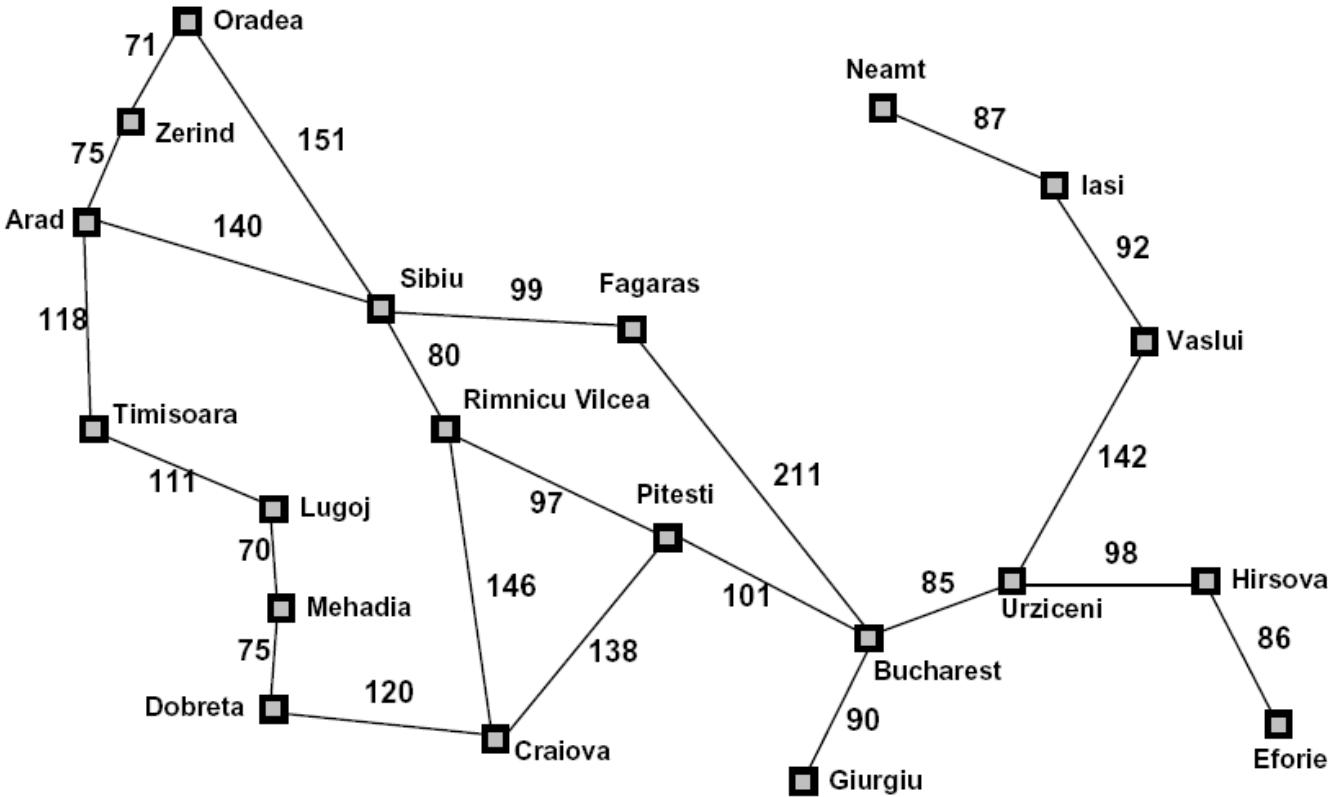


# جستجوی حریصانه

---



# مثال: تابع هيورنستيك



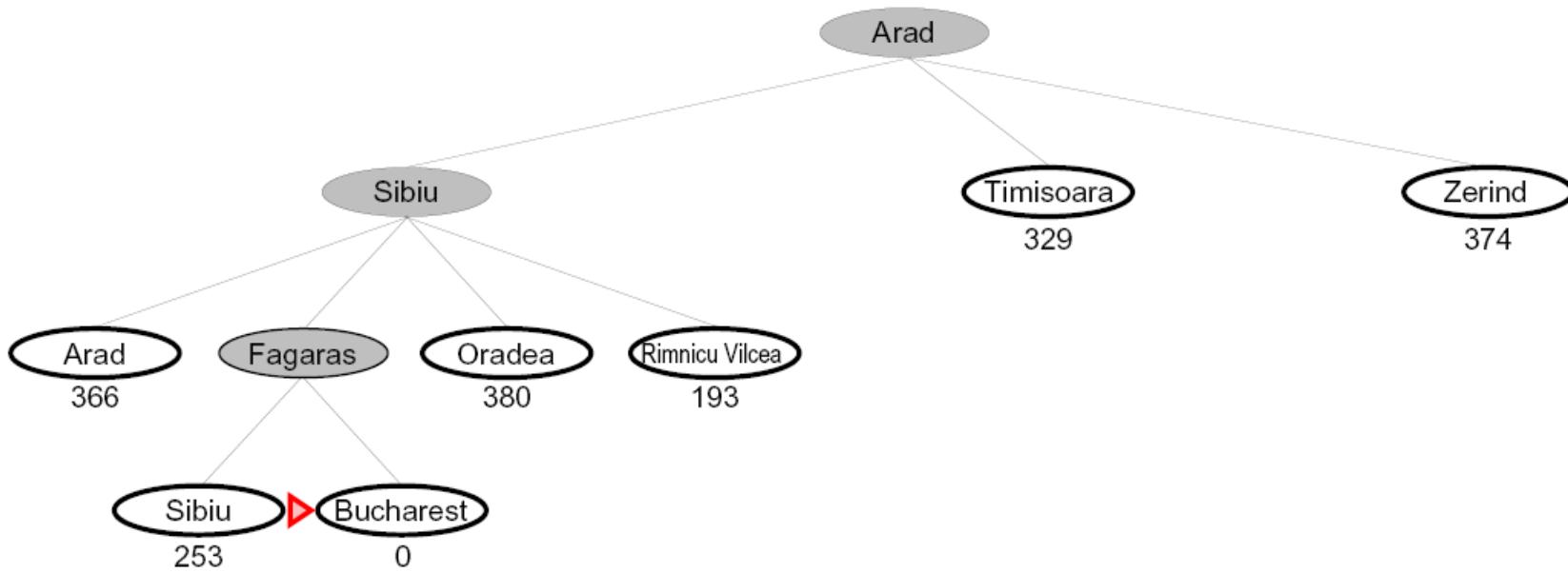
Straight-line distance  
to Bucharest

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

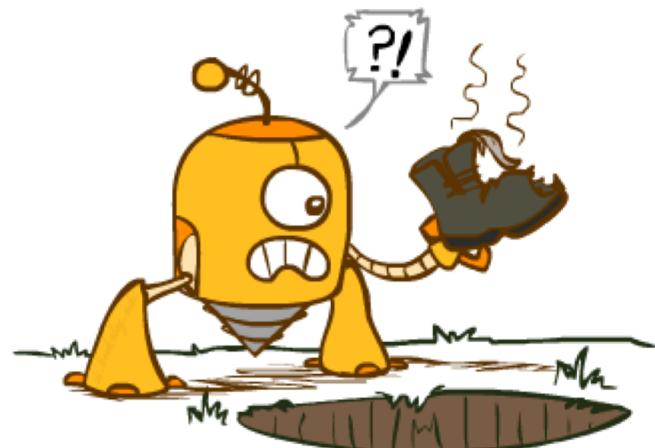
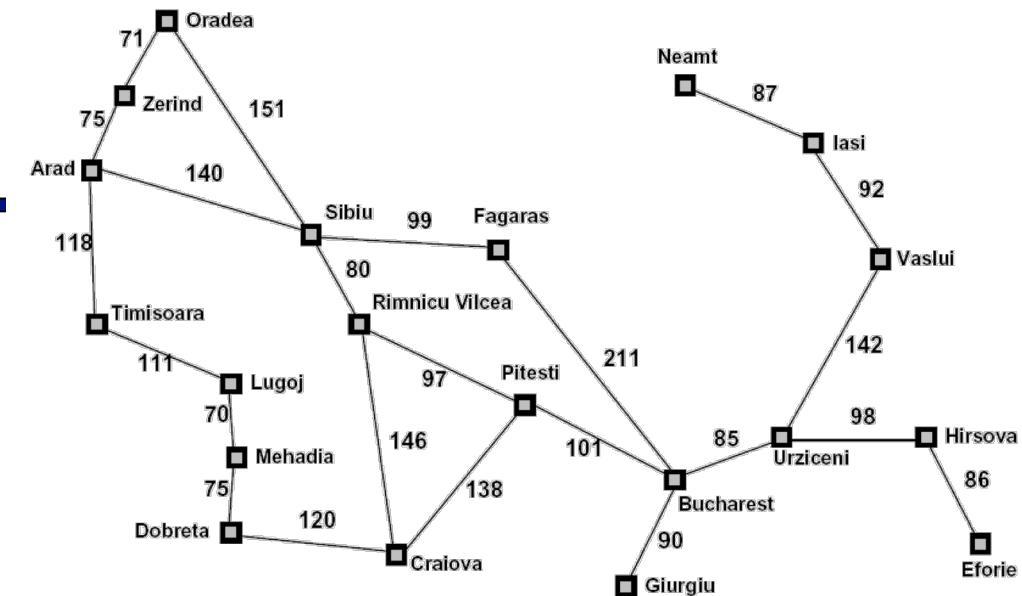
$h(x)$

# جستجوی حریصانه

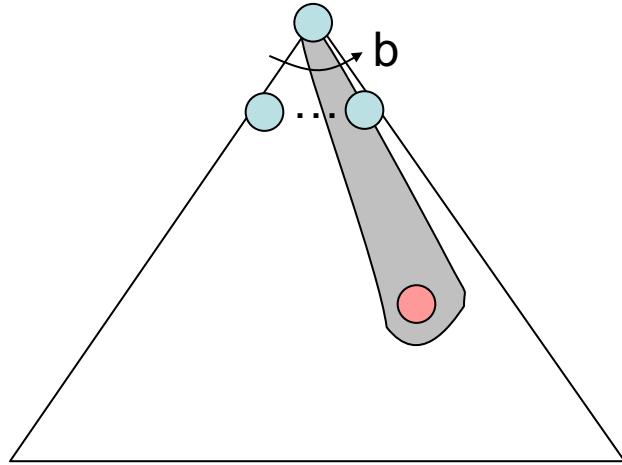
استراتژی: گره‌ای را بسط بده که به نظر (هیوریستیک) از همه به راه حل (مقصد) نزدیکتر است می‌کند



چه مشکلی ممکن است پیش آید؟

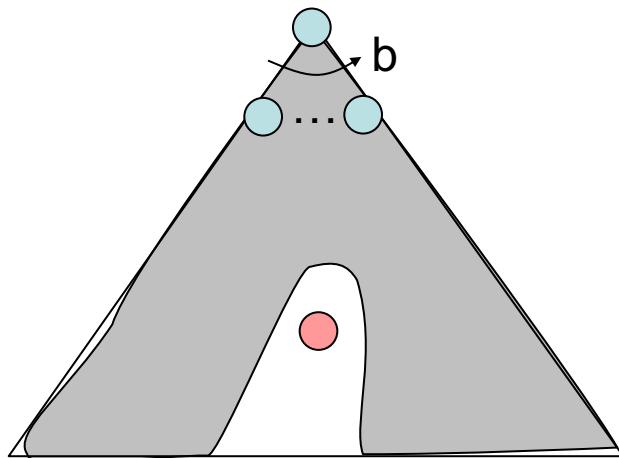


# جستجوی حریصانه



استراتژی:  
گره‌ای را بسط بده که به نظر (هیوریستیک) از همه به راه حل (مقصد) نزدیکتر است می‌کند

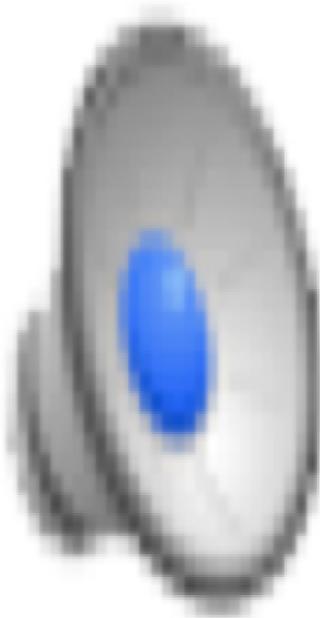
هیوریستیک: فاصله‌ی (مستقیم-هوایی) تا راه حل را تخمین بزن



بدترین حالت: اول-عمقی که راهنمایی نادرست شده

# جستجوی حریصانه

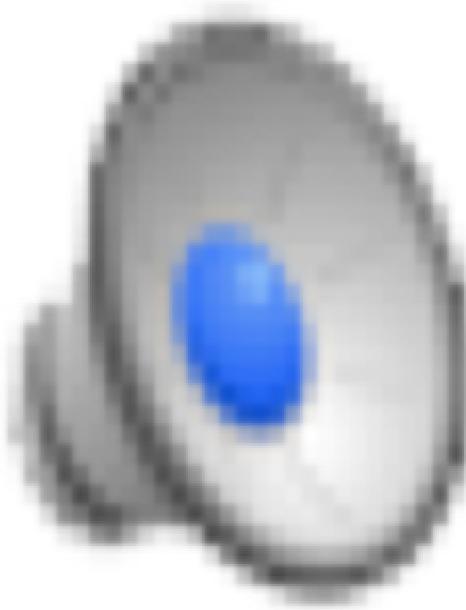
---



video

# جستجوی حریصانه (پکمن)

---



video

# جستجوی A\*



# جستجوی A\*

---



هزینه یکسان

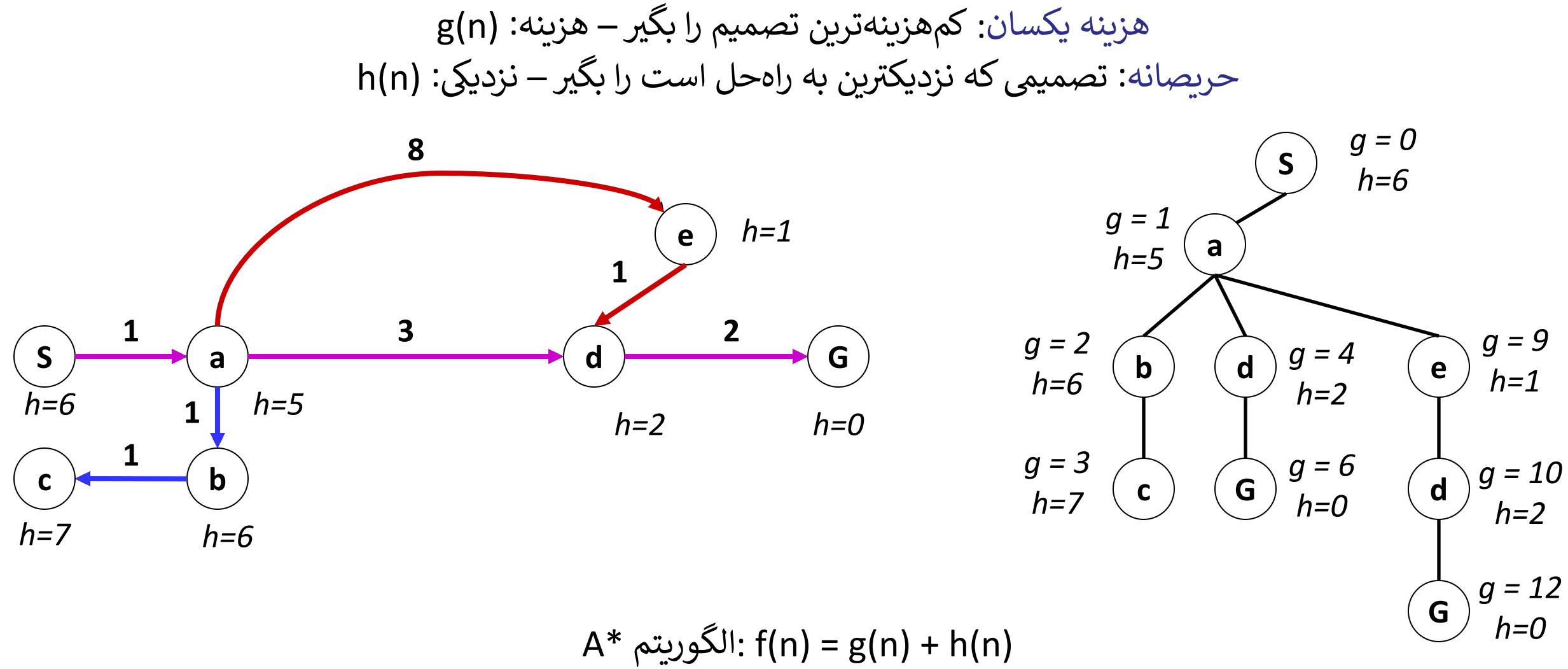


حریصانه

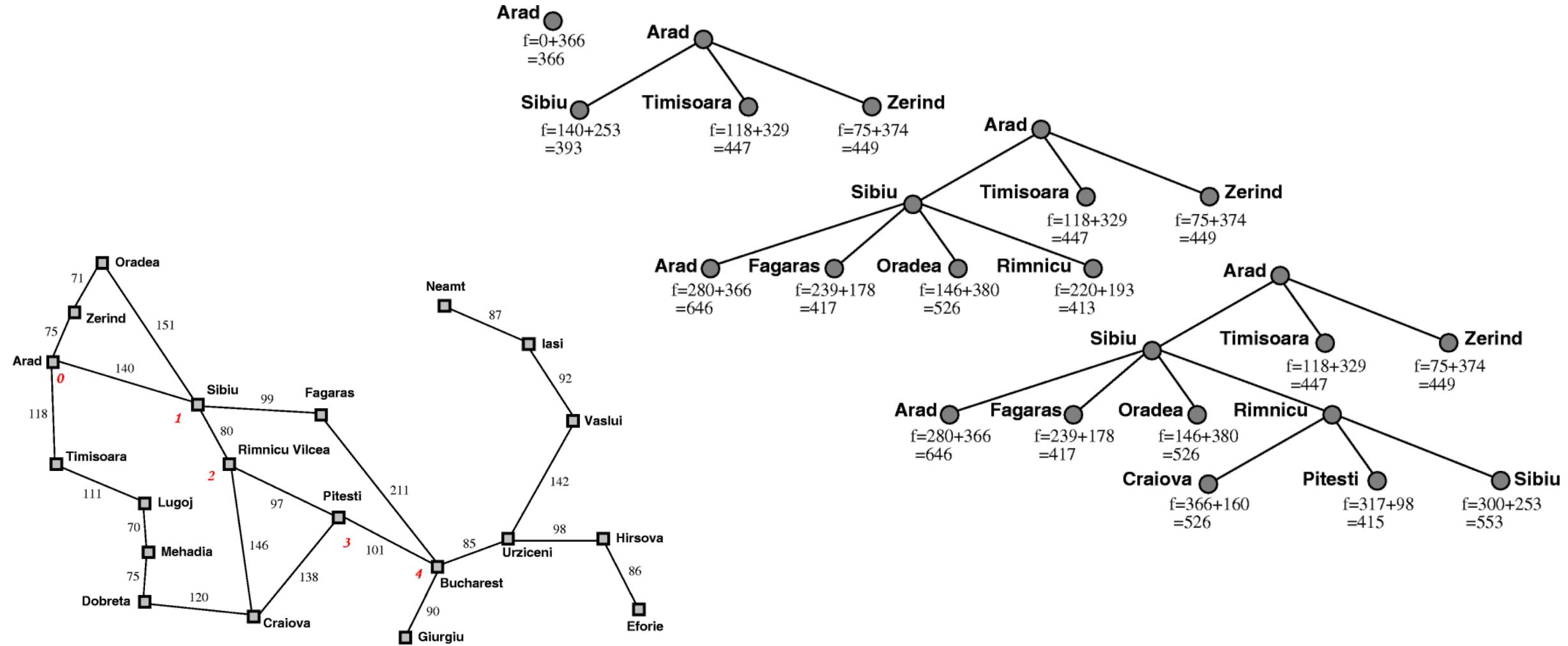


A\*

# ترکیب الگوریتم‌های حریصانه و هزینه یکسان



# A\* – نقشه رومانی



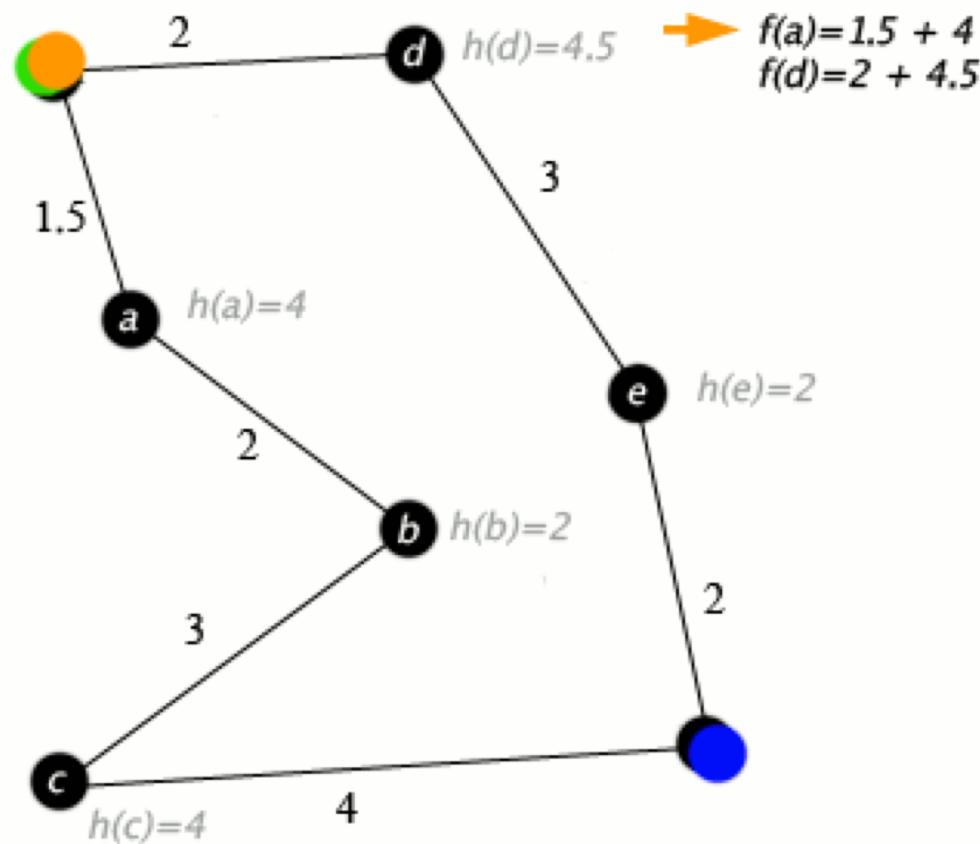
# A\* مثال



[video](#)

Illustration from Wikipedia

# A\* مثال

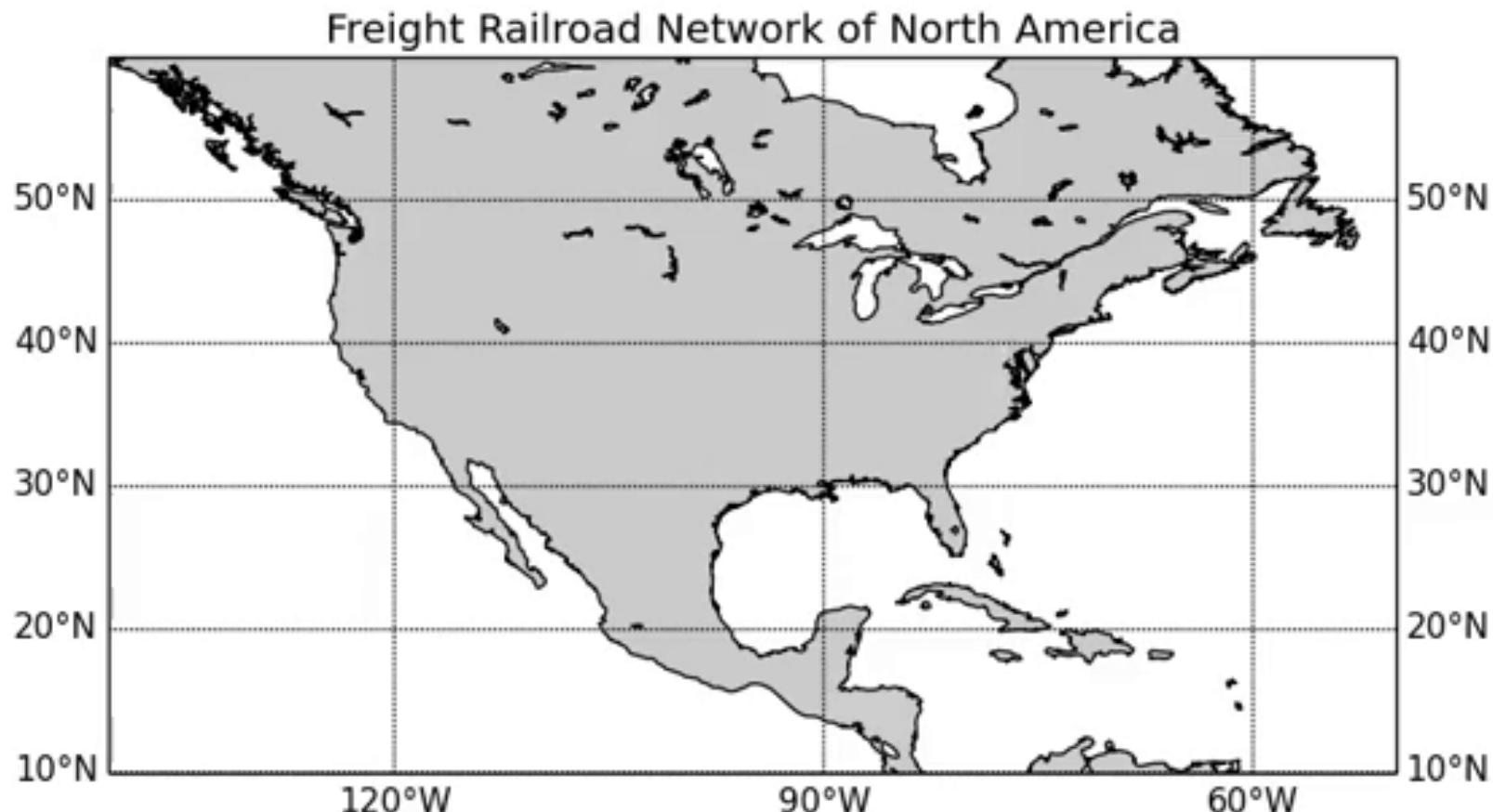


video

Illustration from Wikipedia

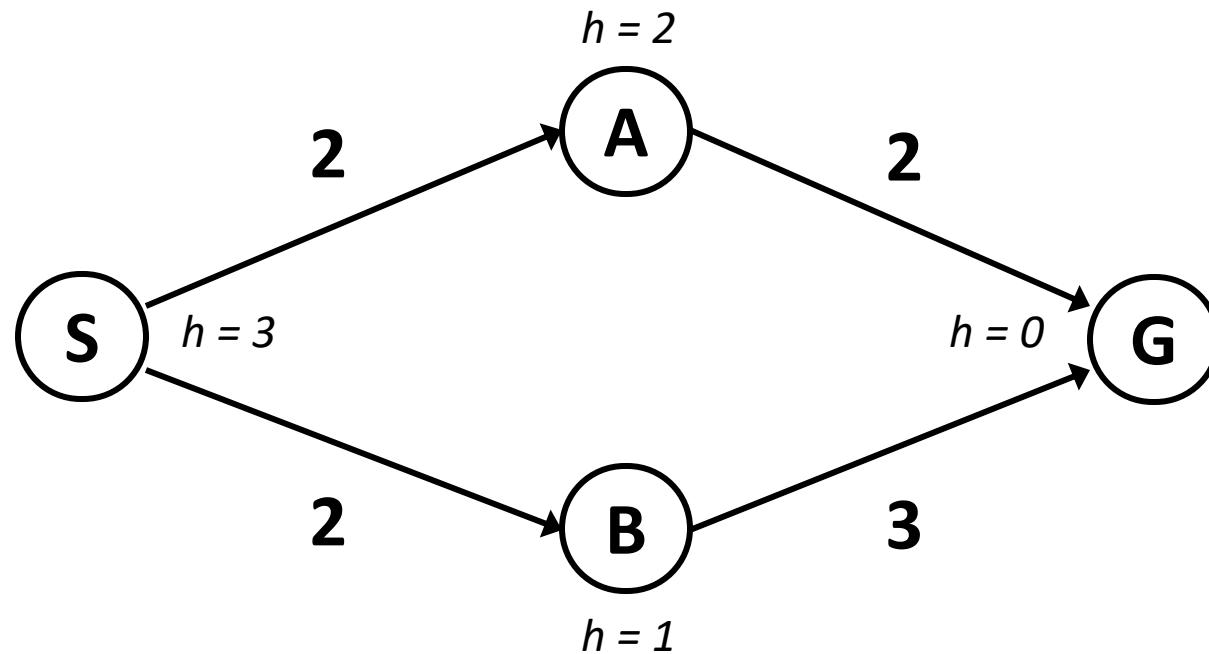
# A\* مثال

مسیر واشنگتن دی سی به لس آنجلس  
لبه‌ها خط راه آهن و هیوریستیک کوتاه‌ترین فاصله‌ی مستقیم (بر روی یک کره)

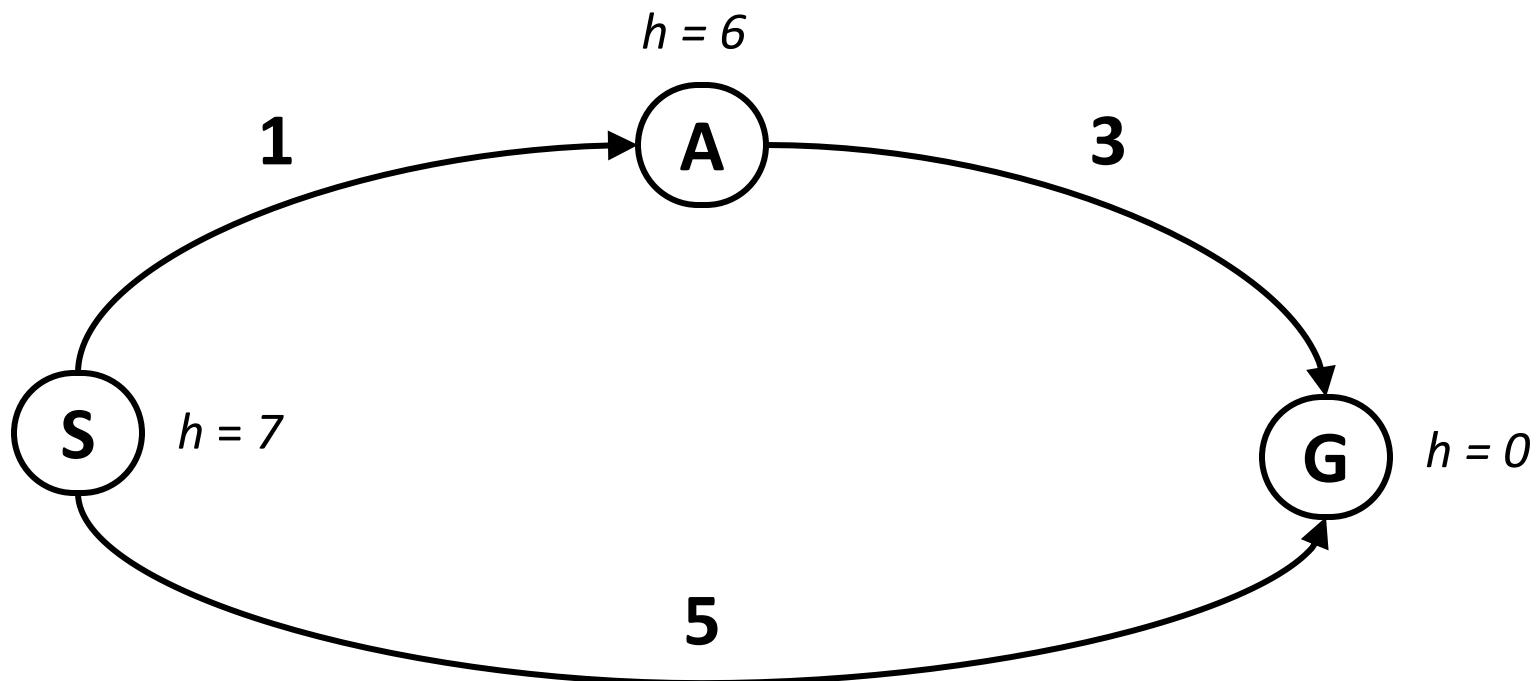


# چه موقع جستجو را متوقف کنیم؟

مادامی که گرهی بهتری هست ادامه بده، حتی اگر به راه حل رسیده‌ای



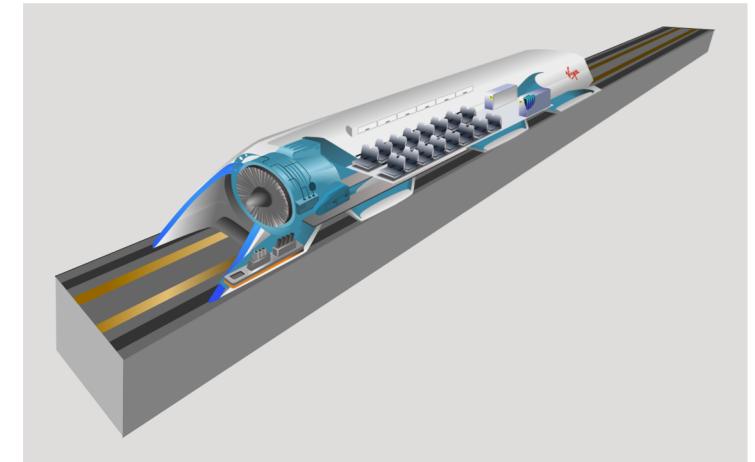
# بهینه؟ - A\*



مشکل کجاست؟  
تخمین ما باید از مقدار واقعی هزینه کمتر باشد!

# Elon Musk

---



# ماشین خودران تسلی

Auto TV

THE PERSON IN THE DRIVER'S SEAT  
IS ONLY THERE FOR LEGAL REASONS.

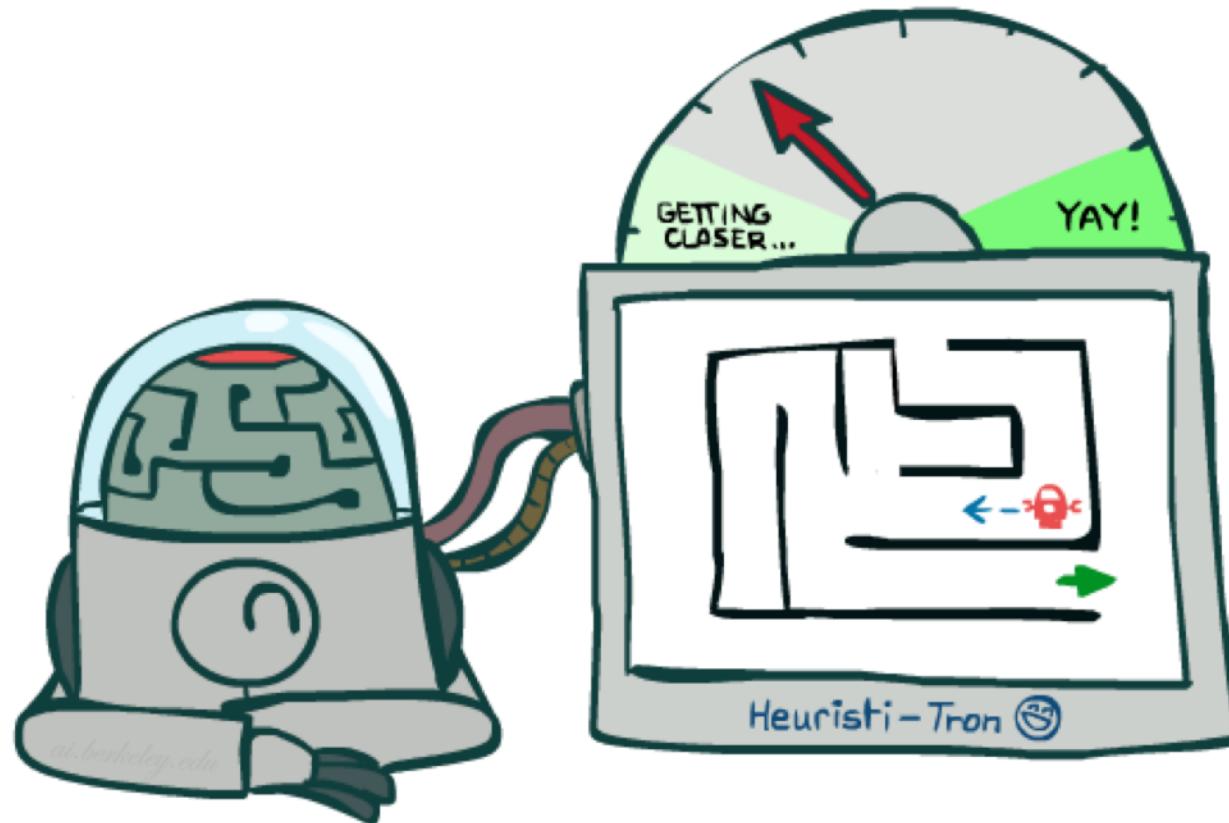
HE IS NOT DOING ANYTHING.  
THE CAR IS DRIVING ITSELF.

# ماشین خودران تسلا

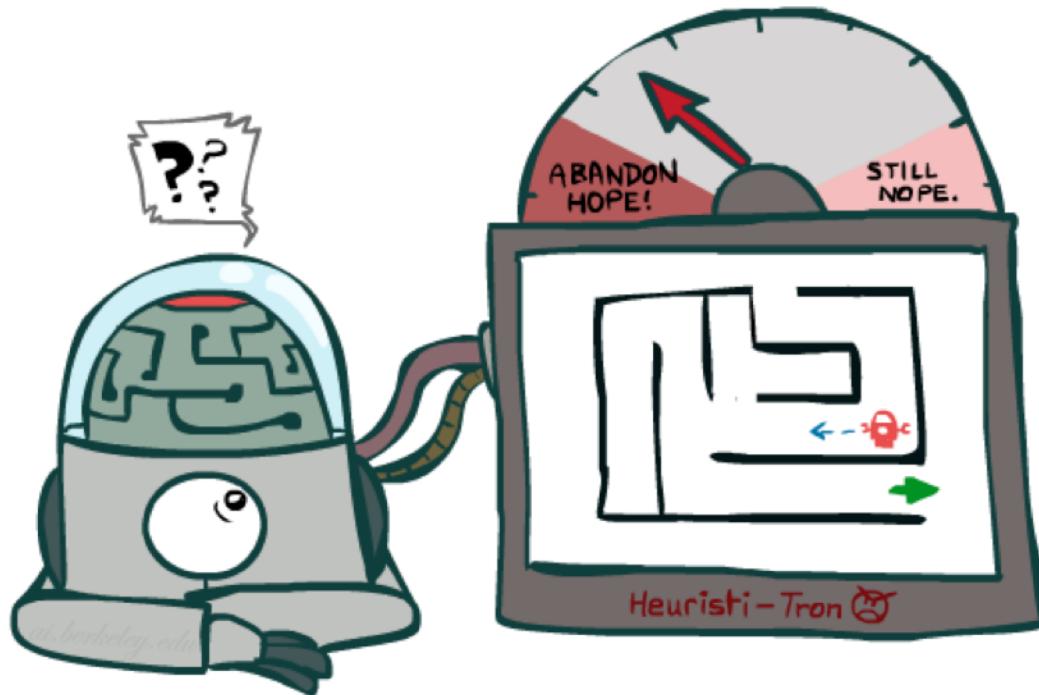


video

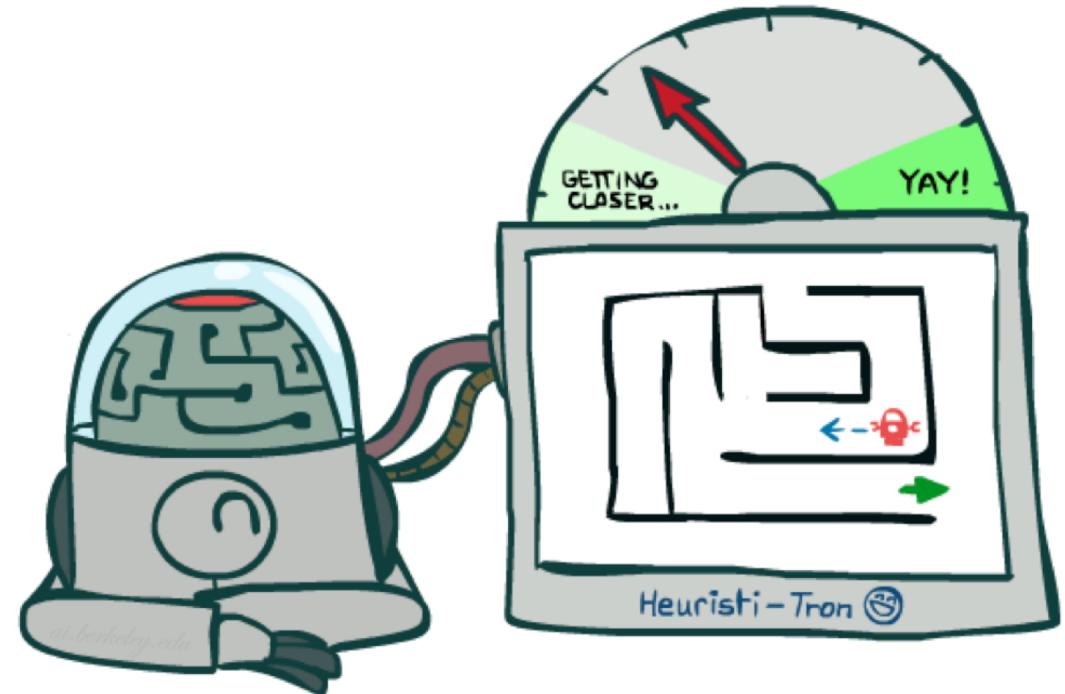
# هیوریستیک «قابل قبول» – Admissible



# ویژگی قابل قبول بودن



هیوریستیک غیرقابل قبول بدین است! هزینه را  
بیشتر از آنچه که هست تخمین می‌زند  
را حل بهینه ممکن است که فدای بدینی شود!



هیوریستیک قابل قبول خوبیان است! هزینه را  
کمتر از آنچه که هست تخمین می‌زند  
تصمیماتی که به نظر خوب نمی‌آیند را از اولویت کنار  
می‌گذارد ولی آنها را حذف نمی‌کند

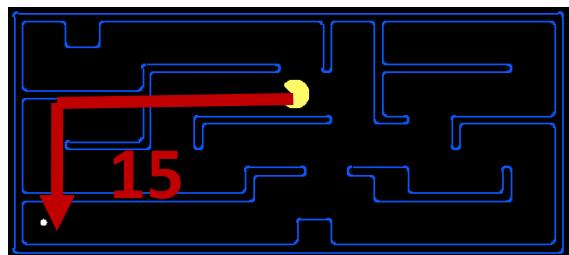
# هیوریستیک قابل قبول

شرط قابل قبول بودن:

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

هزینه‌ی واقعی گرهی  $n$ :

مثال:



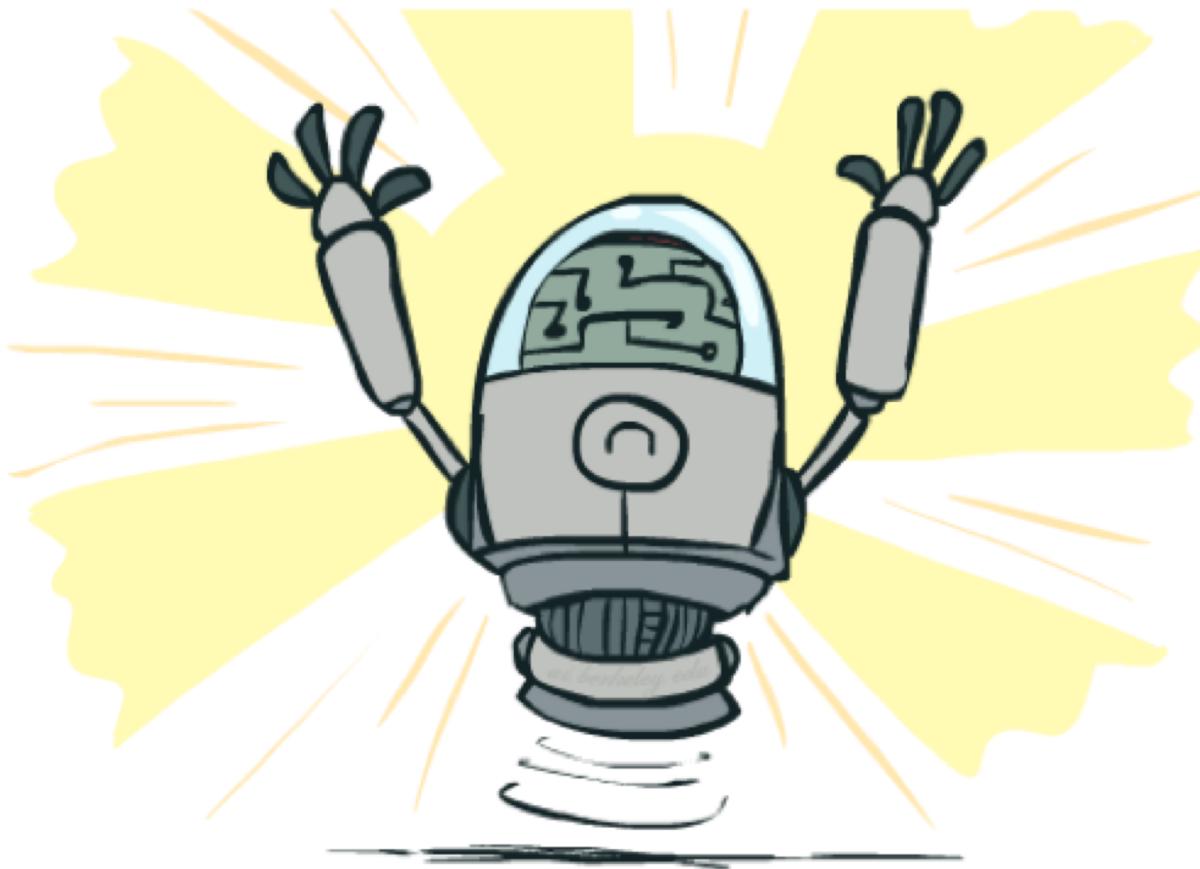
4



کار اصلی ما در طراحی الگوریتم  $A^*$  پیدا کردن یک هیوریستیک قابل قبول برای مساله

# بھینگی روشن جستجوی A\*

---



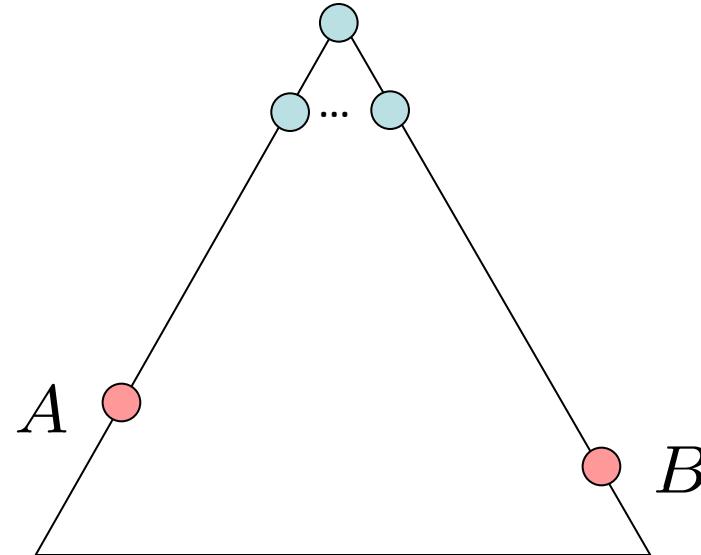
# بھینگی روشن جستجوی A\*

Assume:

- A is an optimal goal node
- B is a suboptimal goal node
- h is admissible

Claim:

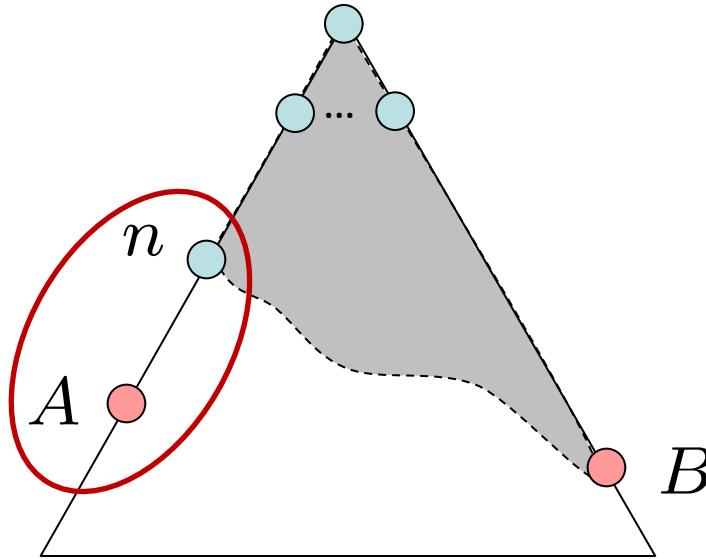
- A will exit the fringe before B



# بھینگی روشن جستجوی $A^*$

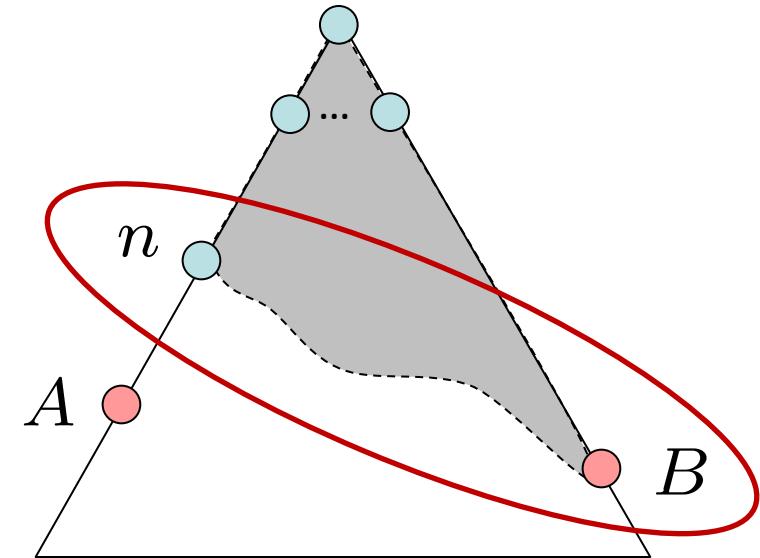
Proof:

- Imagine B is on the fringe
- Some ancestor  $n$  of A is on the fringe, too (maybe A!)
- Claim:  $n$  will be expanded before B  
 $f(n)$  is less or equal to  $f(A)$



# A\* جستجوی روش بھینگی

$f(B) = g(B)$	since $h(B) = 0$
$f(A) = g(A)$	since $h(A) = 0$
$g(A) < g(B)$	since $B$ is suboptimal
$f(A) < f(B)$	
$h(n) \leq h^*(n)$	since $h$ is admissible
$g(n) + h(n) \leq g(n) + h^*(n)$	
$f(n) \leq f(A)$	
$f(n) < f(B)$	since $f(A) < f(B)$



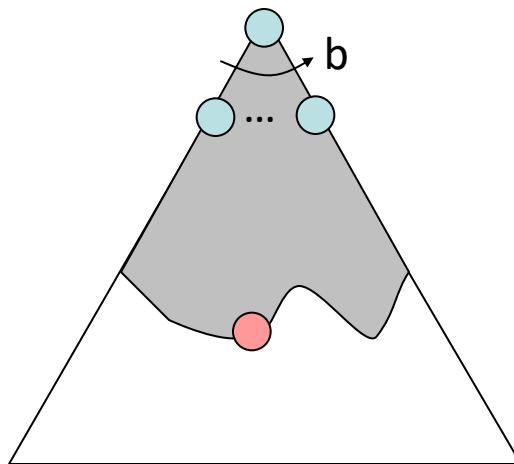
- All ancestors of A expand before B
- A expands before B
- A\* search is optimal

ویژگی‌های  $A^*$

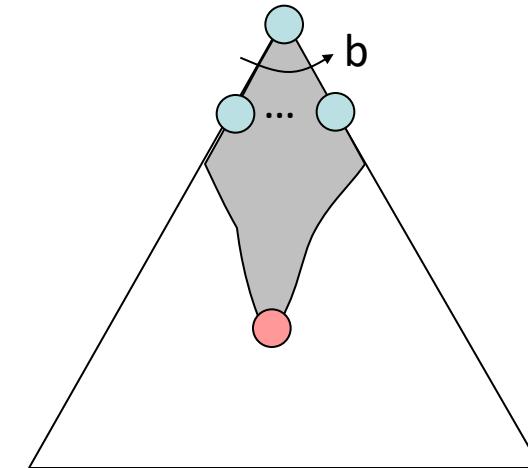
# ویژگی‌های $A^*$

---

هزینه-یکسان

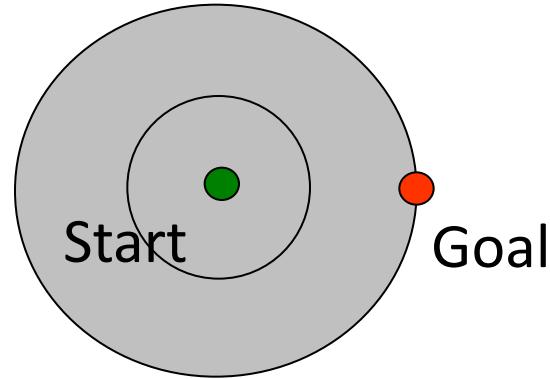


$A^*$

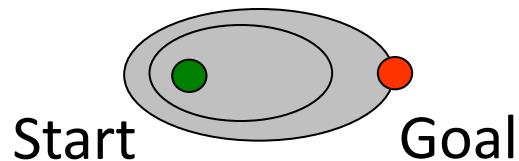


# هزینه-یکسان در برابر $A^*$

---



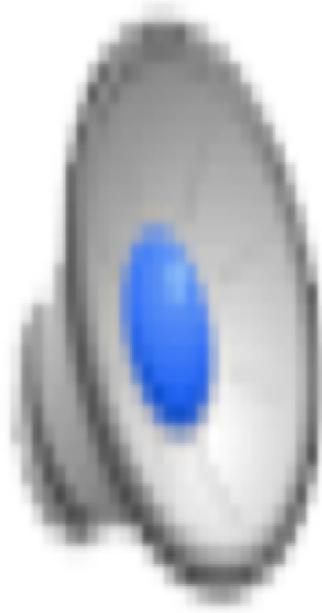
هزینه-یکسان در تمام جهات پیش می‌رود



به سمت هدف متمایل می‌شود

# جستجوی هزینه-یکسان

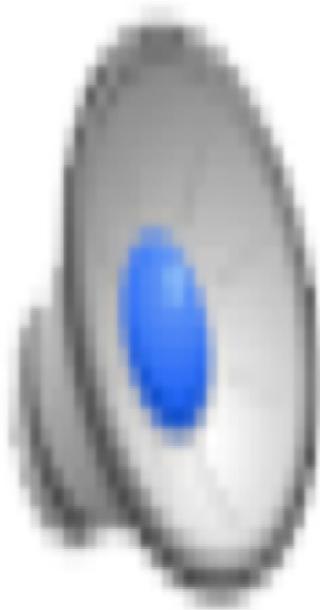
---



video

# جستجوی حریصانه

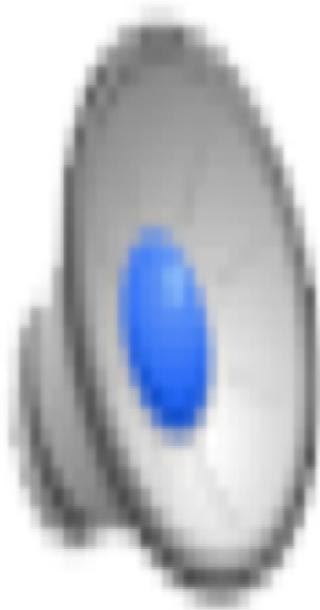
---



video

# جستجوی A\*

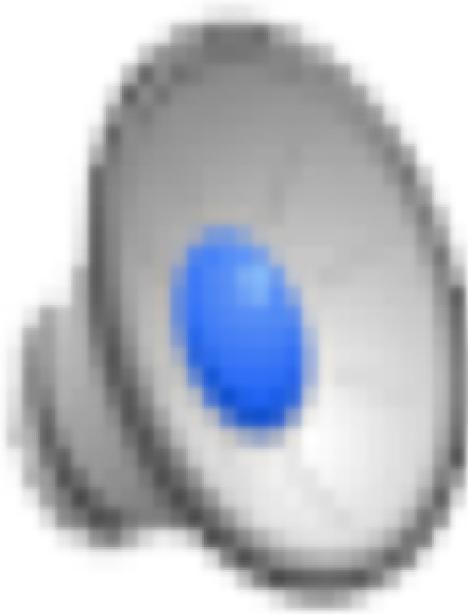
---



video

# جستجوی $A^*$ - پکمن

---

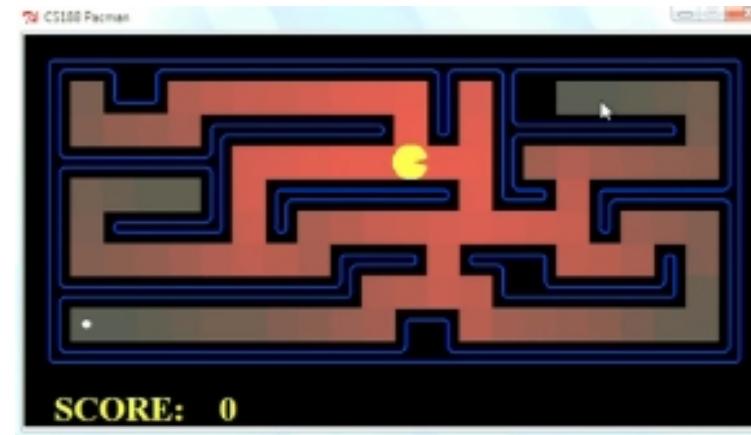


video

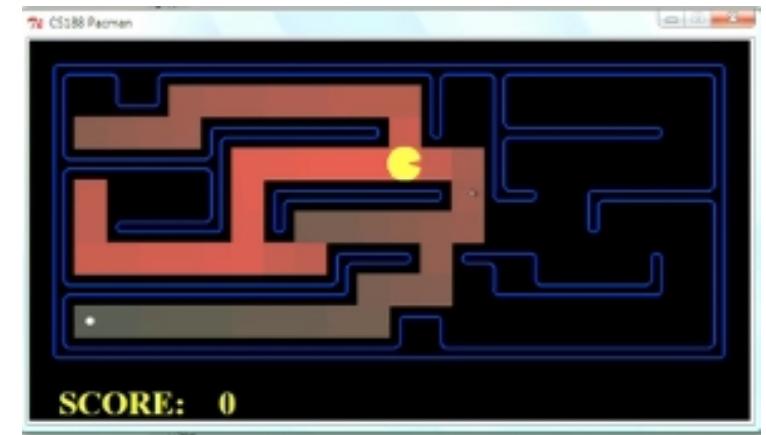
# مقایسه روش‌های جستجو



Greedy



Uniform Cost



A\*

# کاربردهای $A^*$

---

بازی‌های کامپیوتری

مسائل مسیریابی

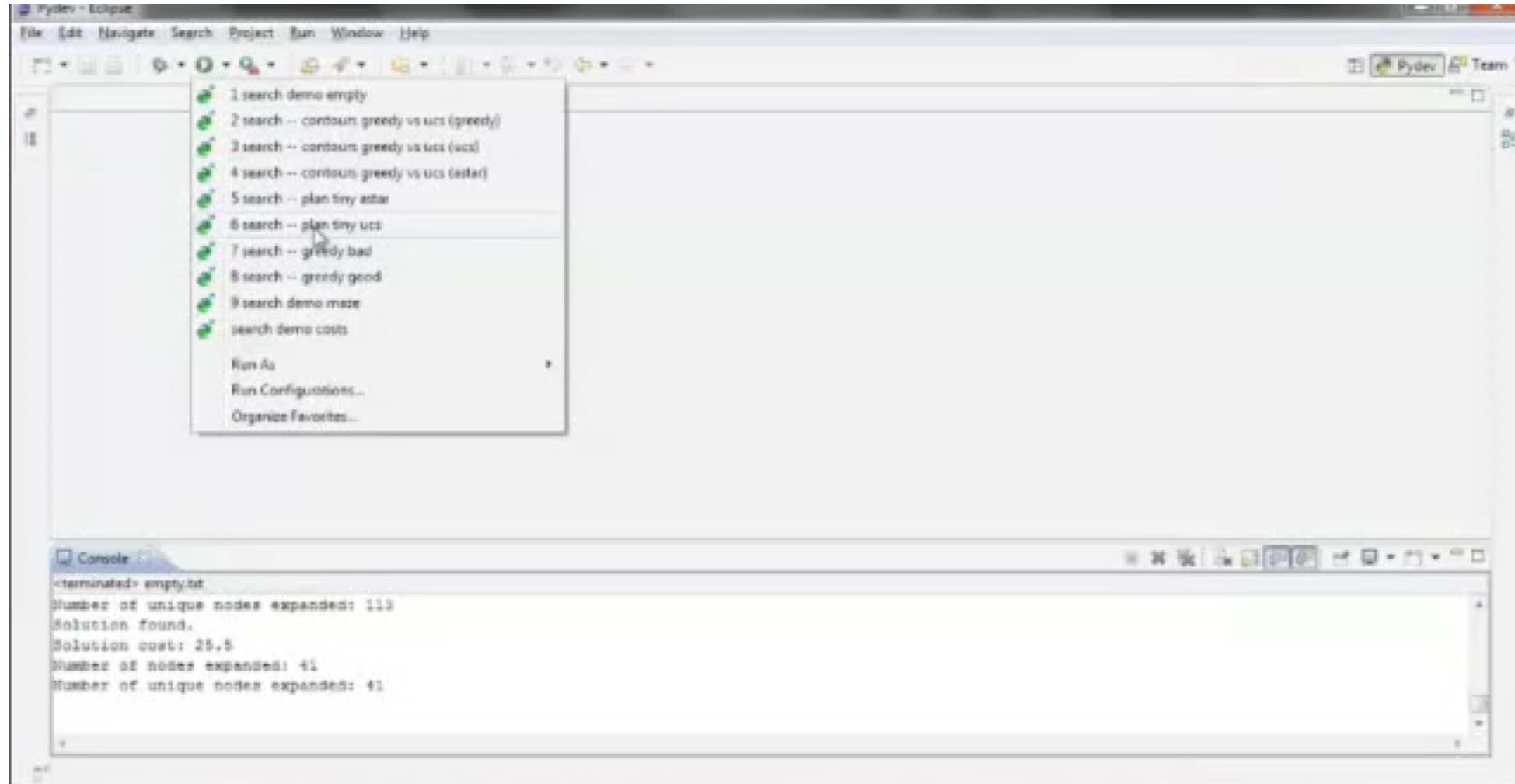
حرکت‌یابی روبات‌ها

ترجمه ماشینی

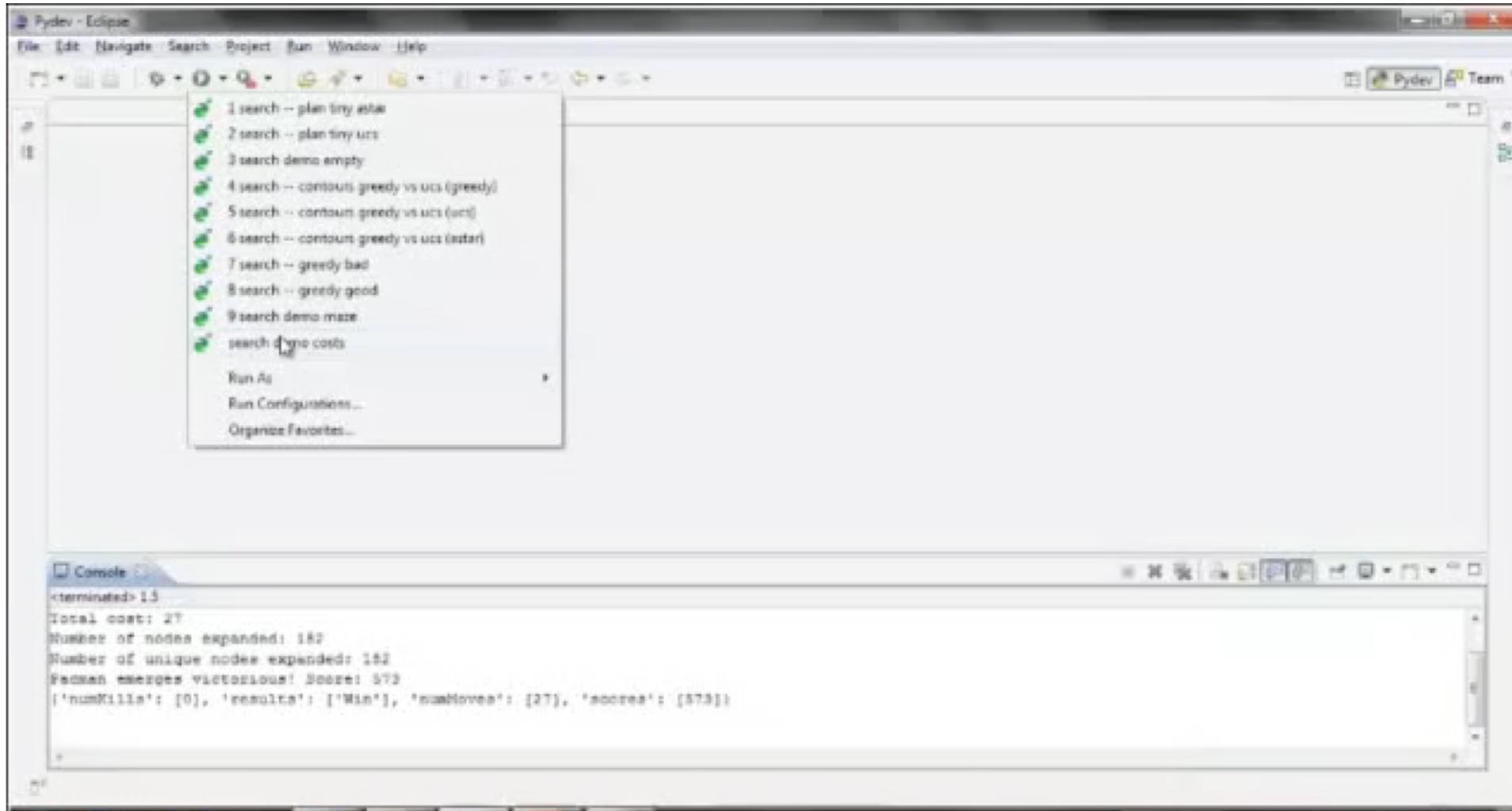
تشخیص گفتار

...

# A\* هزینه-یکسان و



# جستجوی پشت صحنه را حدس بزنید



# طراحی هیوریستیک‌ها

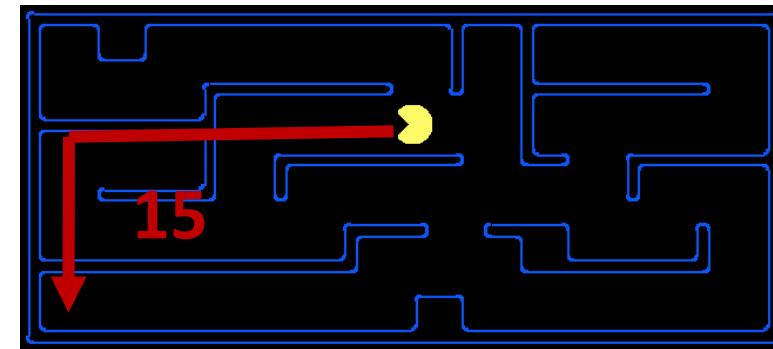
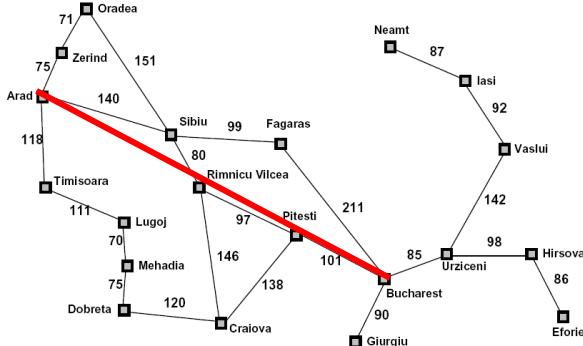


# هیوریستیک قابل قبول

کار اصلی در حل مسائل پیدا کردن یک هیوریستیک «قابل قبول» است

یک روش: از محدودیت‌های مساله کم کن و راه حل را به عنوان هیوریستیک در نظر بگیر

366



# 8-puzzle – مثال

7	2	4
5		6
8	3	1

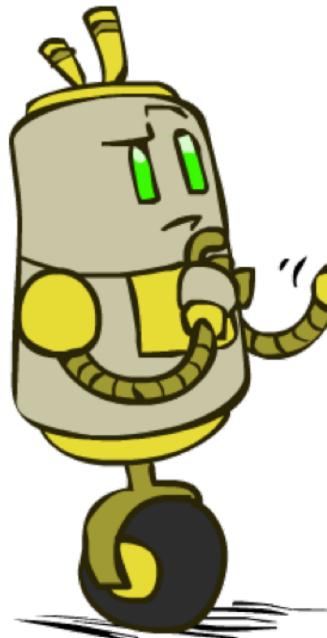
حالت شروع

3	7	1
2	4	5
8		6

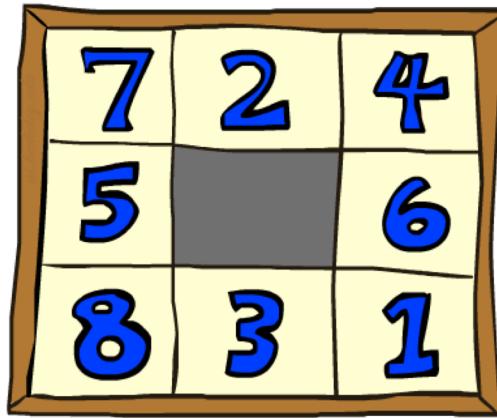
Actions

	1	2
3	4	5
6	7	8

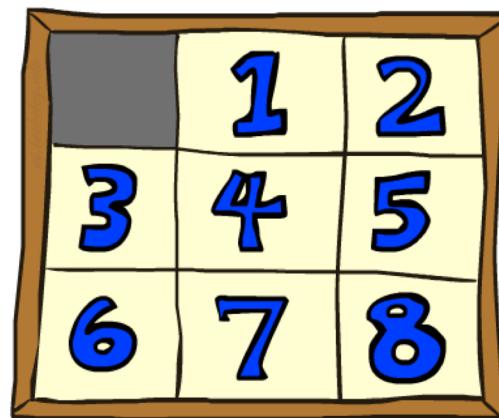
هدف



# مثال – 8-puzzle



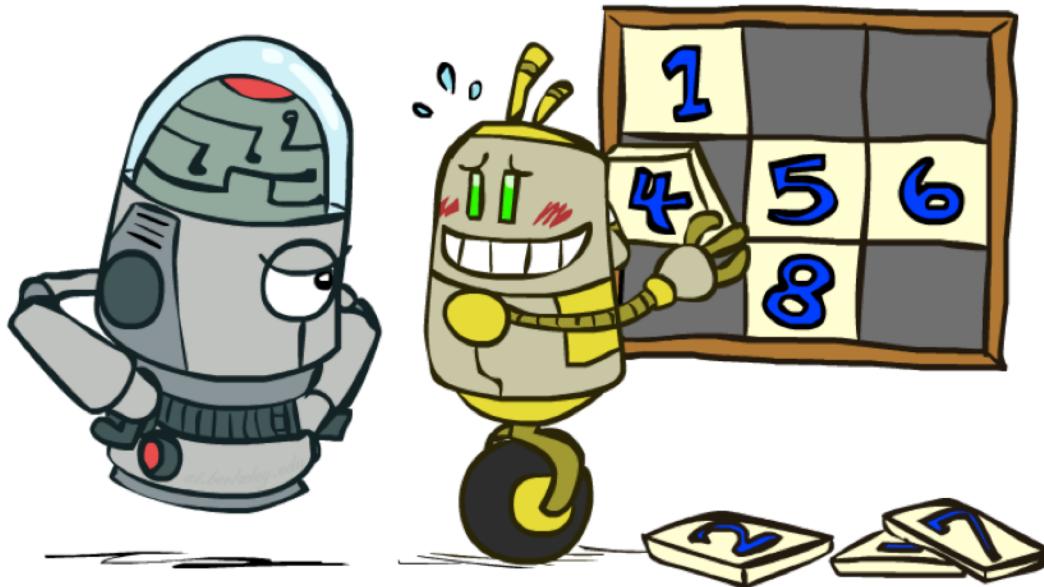
Start State



Goal State

هیوریستیک ۱ : تعداد کاشی‌های که در جای خود نیستند چرا «قابل قبول» است؟

$$h(\text{start}) = 8$$



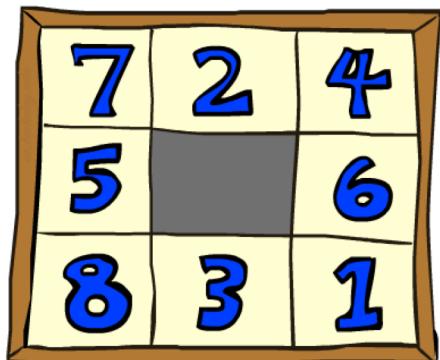
Average nodes expanded when the optimal path has...			
	...4 steps	...8 steps	...12 steps
UCS	112	6,300	$3.6 \times 10^6$
TILES	13	39	227

# مثال – 8-puzzle

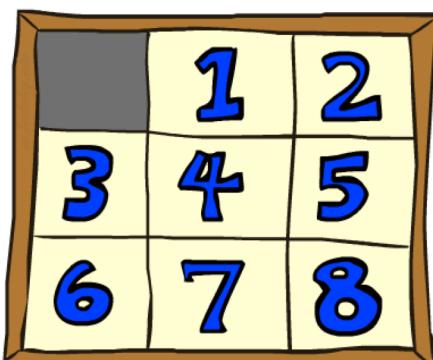
هیوریستیک ۲: کاشی‌ها آزادانه حرکت کنند

چرا «قابل قبول» است؟

$$h(\text{start}) = 3 + 1 + 2 + \dots = 18$$



Start State



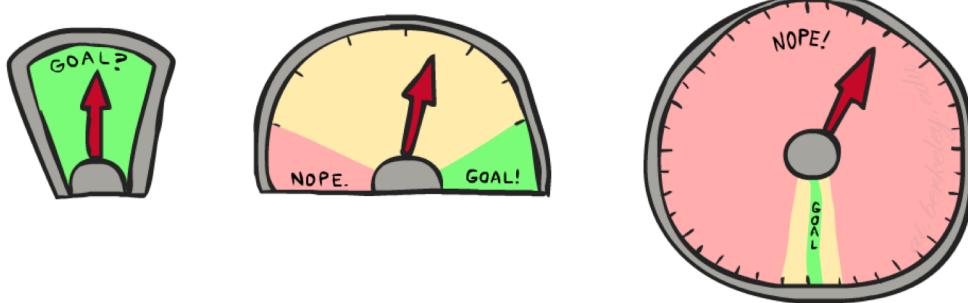
Goal State

Average nodes expanded  
when the optimal path has...

	...4 steps	...8 steps	...12 steps
TILES	13	39	227
MANHATTAN	12	25	73

# مثال – 8-puzzle

از هزینه‌ی واقعی به عنوان هیوریستیک استفاده کنیم؟  
قابل قبول؟

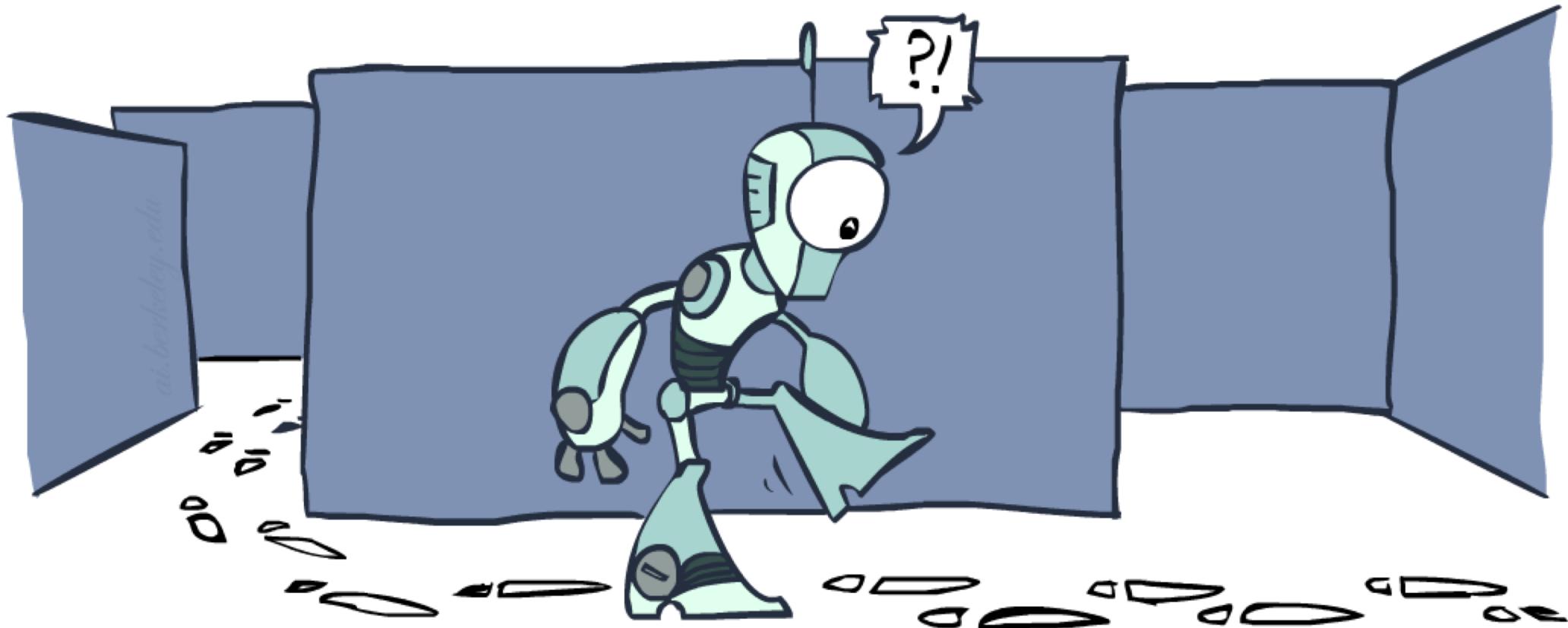


آیا در تعداد گره‌های بسط داده شده بهبودی می‌دهد؟  
مشکل کجاست؟

بدهستان (trade-off) بین تخمین خوب و دشواری رسیدن به آن  
هر چه هیوریستیک تخمین بهتری از هزینه‌ی واقعی بزند، احتمالاً سختی محاسبه‌ی آن بیشتر می‌شود ولی  
گره‌های کمتری بسط داده می‌شوند

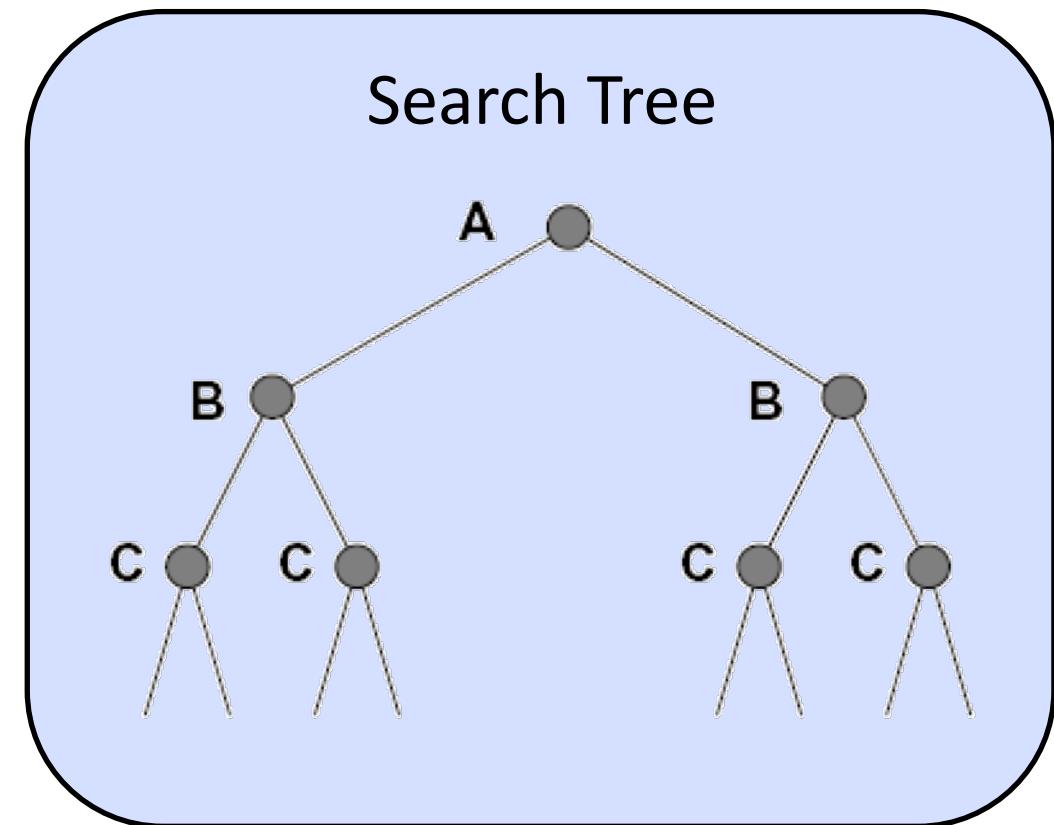
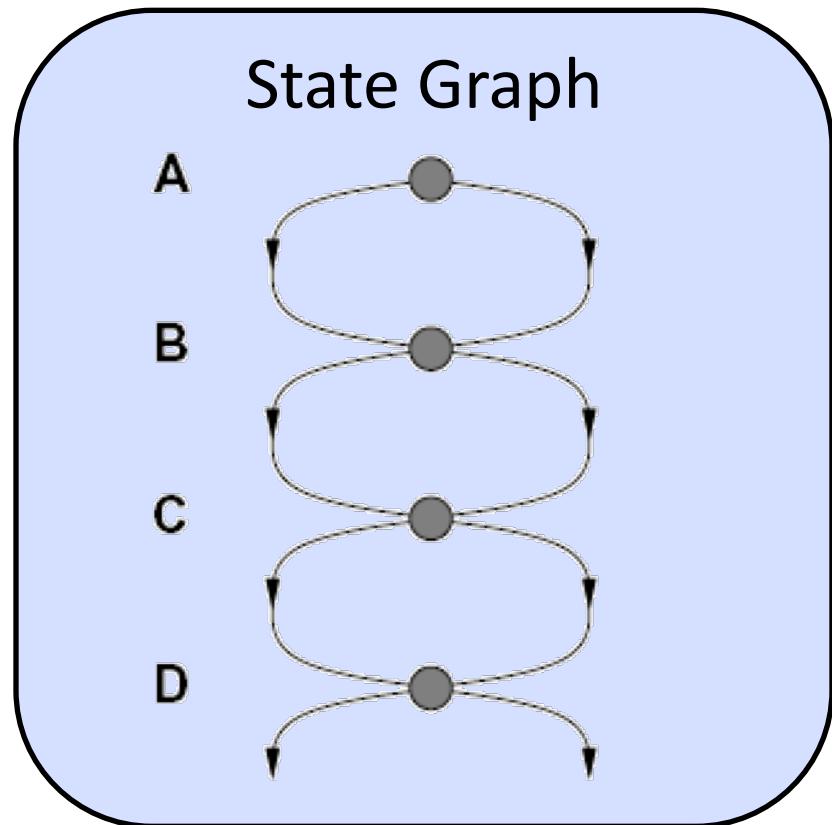
# جستجوی گرافی

---



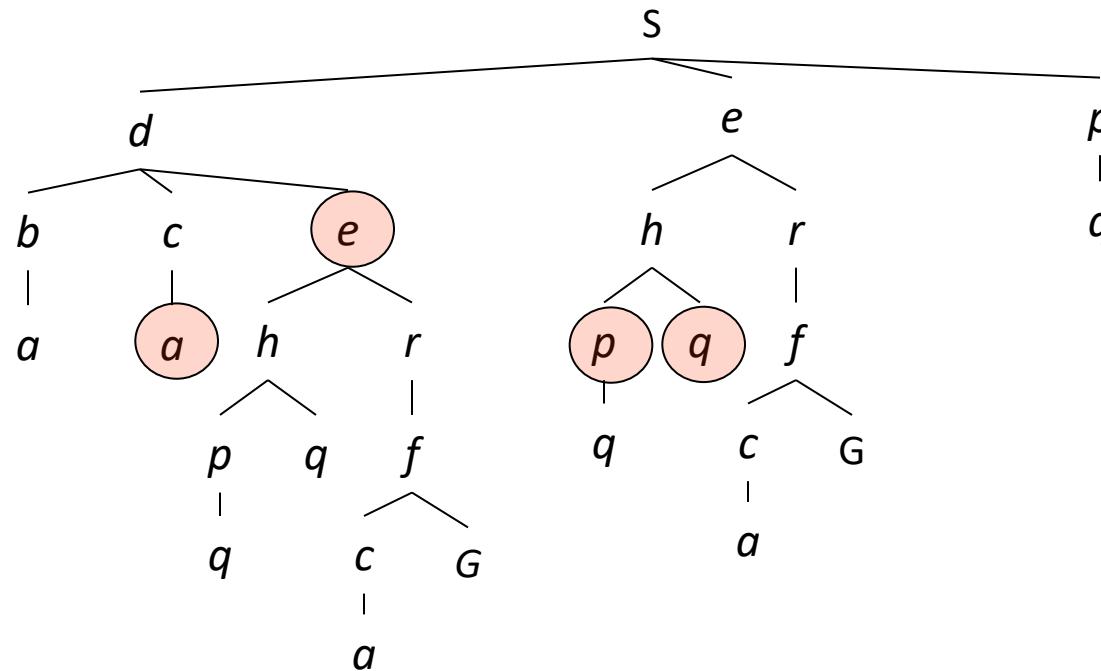
# جستجوی درختی: کار اضافه!

به دلیل وجود حالت‌های تکراری



# کار تکراری در جستجوی درختی

به عنوان مثال، نیاز به بسط دادن گره‌های نشان داده شده نداریم



# جستجوی گرافی

ایده: همچ گره‌ای را بیش از یک بار بسط نده!

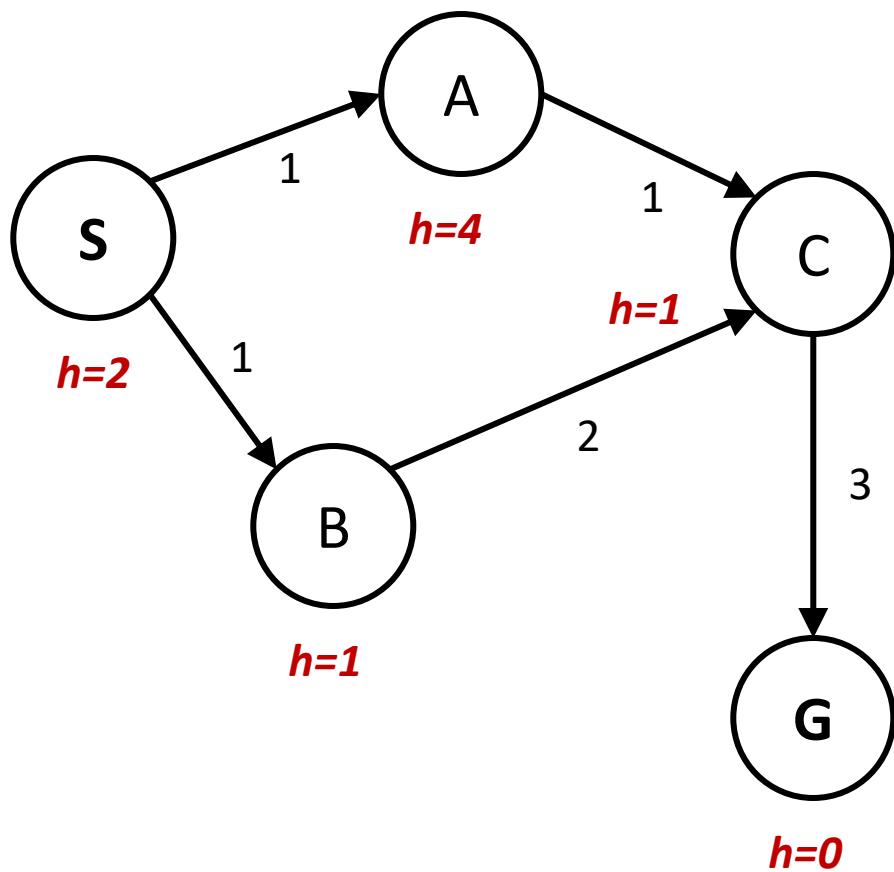
پیاده سازی:

جستجوی درختی + حافظه‌ای از گره‌های بسط داده شده  
قبل از بسط دادن هر گره چک کن که آیا قبل از بسط داده شده یا نه؟  
اگر جدید است، بسط بده و به حافظه اضافه کن  
در غیر این صورت، آن را در نظر نگیر

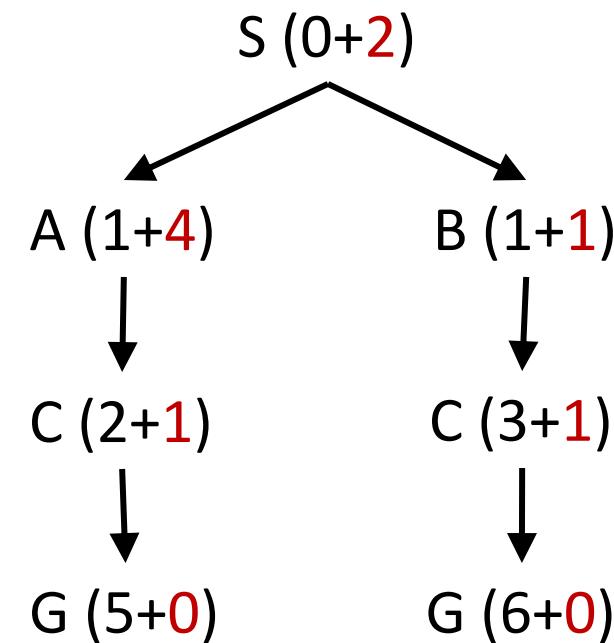
آیا جستجوی گرافی می‌تواند در «کامل» بودن تاثیری بگذارد?  
در بهینگی چطور؟

# جستجوی گرافی

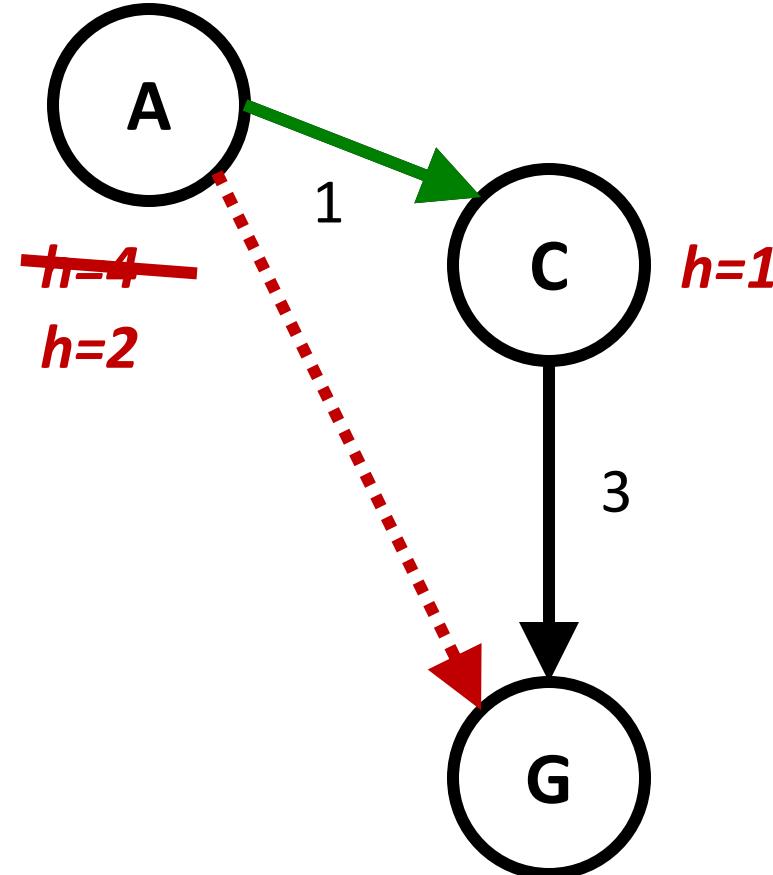
گراف فضای حالت



جستجوی درختی



# سازگاری هیوریستیک - Consistency -



قابل قبول بودن: تخمین هیوریستیک کوچکتر از هزینه‌ی واقعی باشد

$$h(A) \leq \text{actual cost from A to G}$$

سازگاری: تخمین از گرهی فعلی همیشه کمتر از (یا مساوی) تخمین از گرهی بعدی  
+ هزینه رسیدن به آن گره

$$h(A) \leq \text{cost}(A \text{ to } C) + h(C)$$

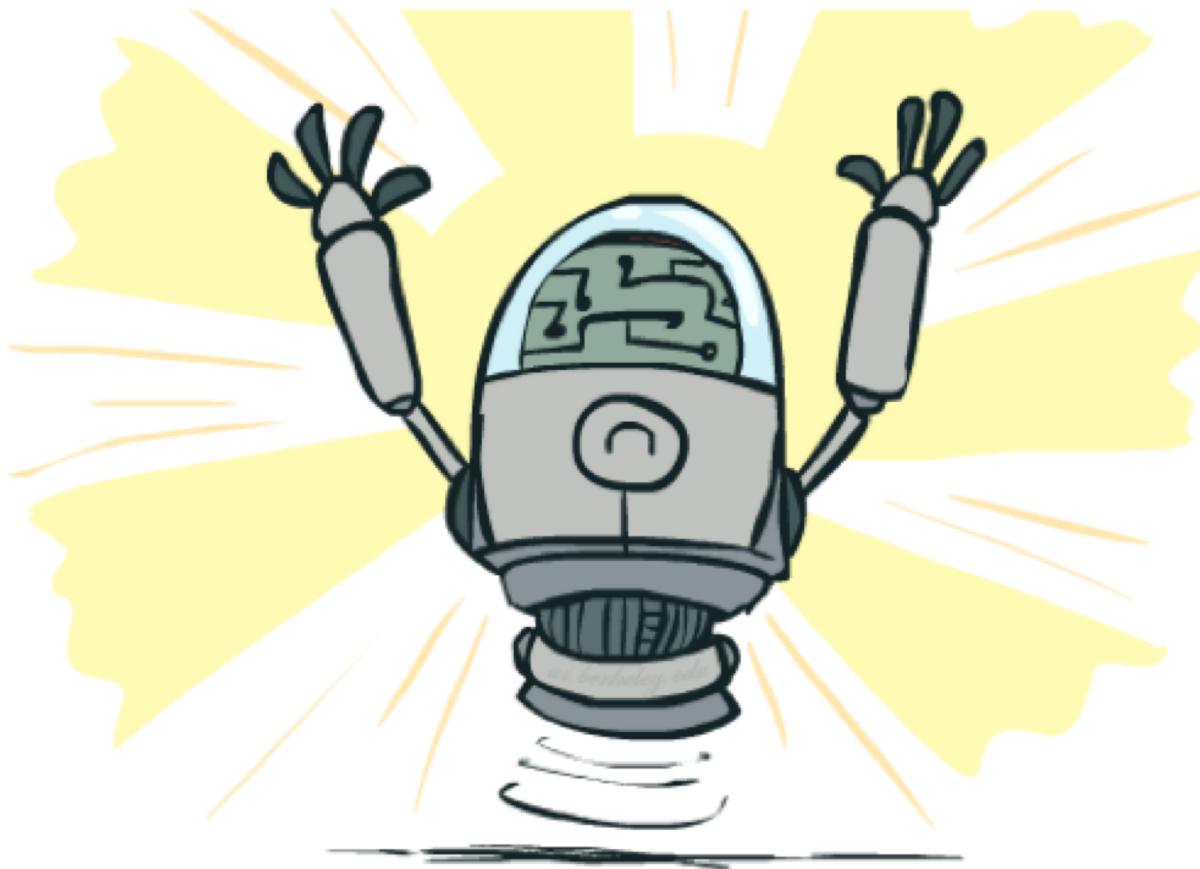
تبعات سازگاری؟

در طول مسیر، از مقدار هزینه هیچ وقت کاسته نمی‌شود

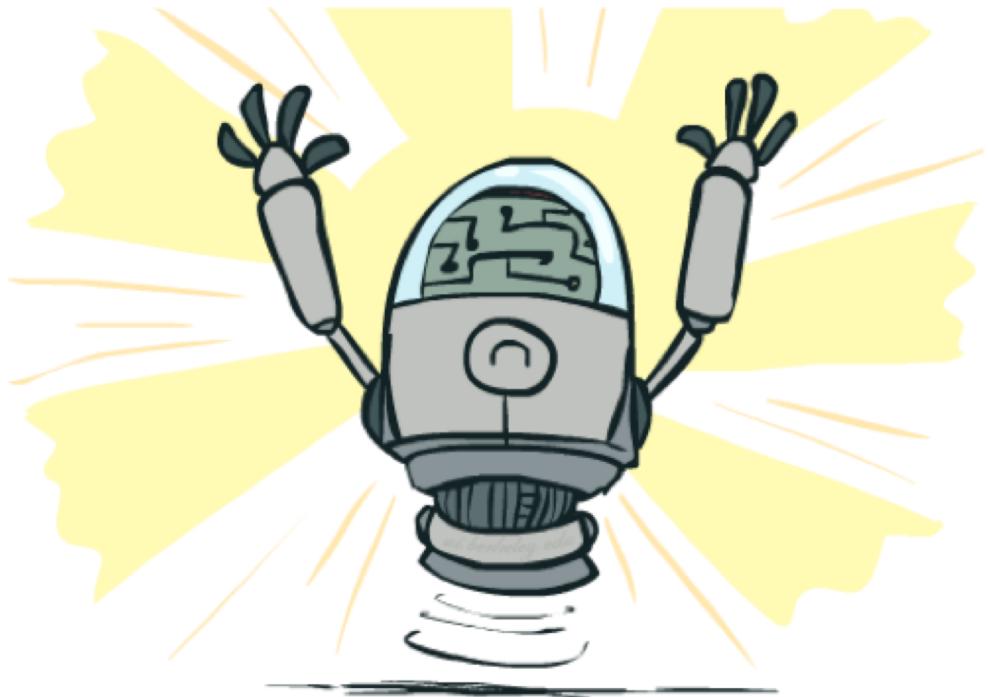
جستجوی گرافی  $A^*$  را بهینه می‌کند!

# بھینگی جستجوی گرافی A\*

---



# بهینگی A\*



## جستجوی درختی

بهینه: در صورت «قابل قبول» بودن هیوریستیک هزینه-یکسان بهینه است، چون  $h=0$  قابل قبول است

## جستجوی گرافی

بهینه: در صورت «سازگار» بودن هیوریستیک هزینه-یکسان بهینه است، چون  $h=0$  سازگار است

هیوریستیک سازگار قابل قبول است

# A\*: خلاصه:

---

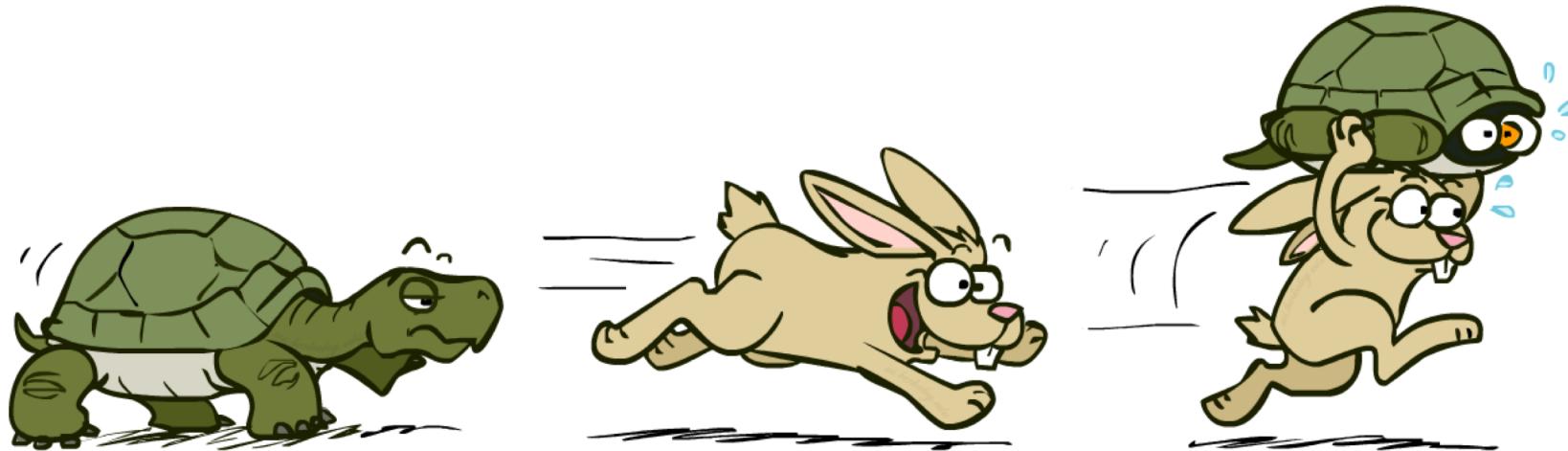


# A\*: خلاصه:

A\* همیشه تخمینی از هزینه‌های پیش رو را در ذهن دارد (علاوه بر هزینه‌های گذشته)

بهینه است، در صورتی که هیوریستیک قابل قبول/سازگار باشد

طراحی هیوریستیک کلیدی است



# شبکه جستجوی درختی

```
function TREE-SEARCH(problem, fringe) return a solution, or failure
  fringe  $\leftarrow$  INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
  loop do
    if fringe is empty then return failure
    node  $\leftarrow$  REMOVE-FRONT(fringe)
    if GOAL-TEST(problem, STATE[node]) then return node
    for child-node in EXPAND(STATE[node], problem) do
      fringe  $\leftarrow$  INSERT(child-node, fringe)
    end
  end
```

# شبکه جستجوی گرافی

```
function GRAPH-SEARCH(problem, fringe) return a solution, or failure
  closed ← an empty set
  fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
  loop do
    if fringe is empty then return failure
    node ← REMOVE-FRONT(fringe)
    if GOAL-TEST(problem, STATE[node]) then return node
    if STATE[node] is not in closed then
      add STATE[node] to closed
      for child-node in EXPAND(STATE[node], problem) do
        fringe ← INSERT(child-node, fringe)
    end
  end
```

# سوال؟

---



هزینه یکسان



حریصانه



A\*