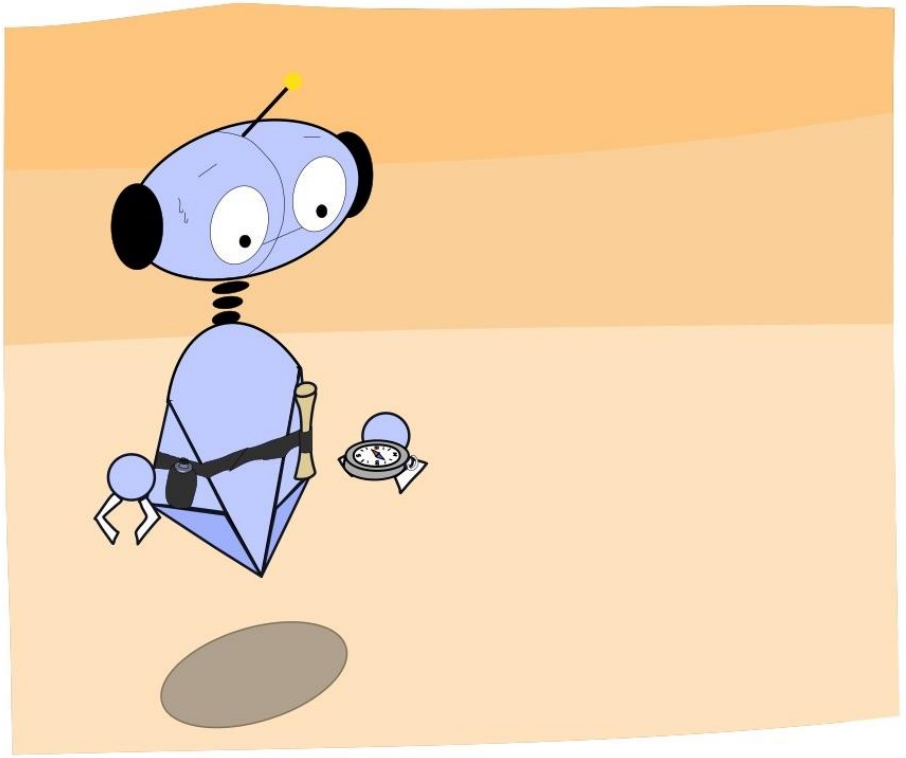
مبانی و کاربردهای هوش مصنوعی

جستجو آگاهانه (فصل 3.5 الی 3.6)



مدرس: مهدی جوانمردی

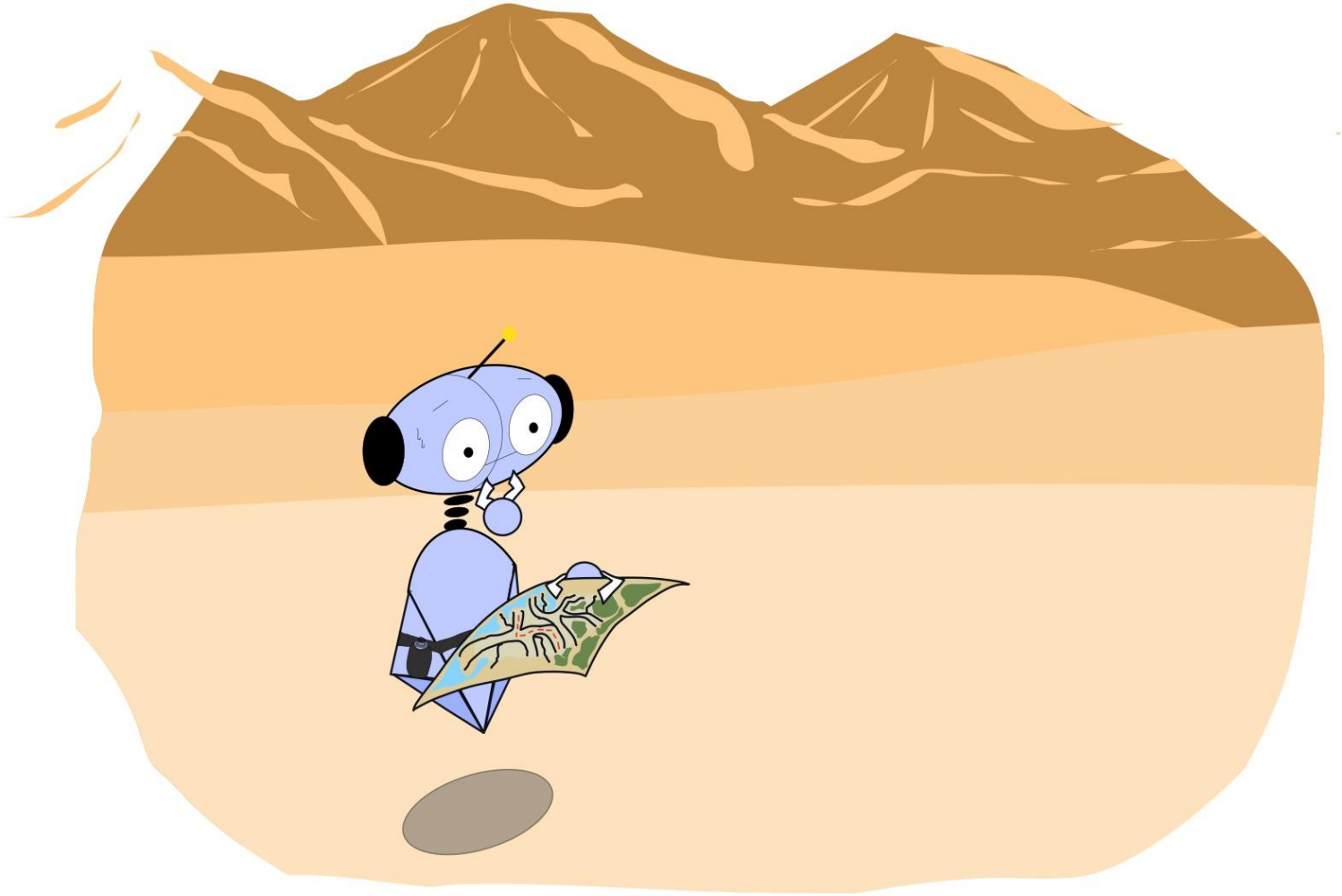
دانشگاه صنعتی امیرکبیر

الهام گرفته از محتوای CS188 دانشگاه برکلی

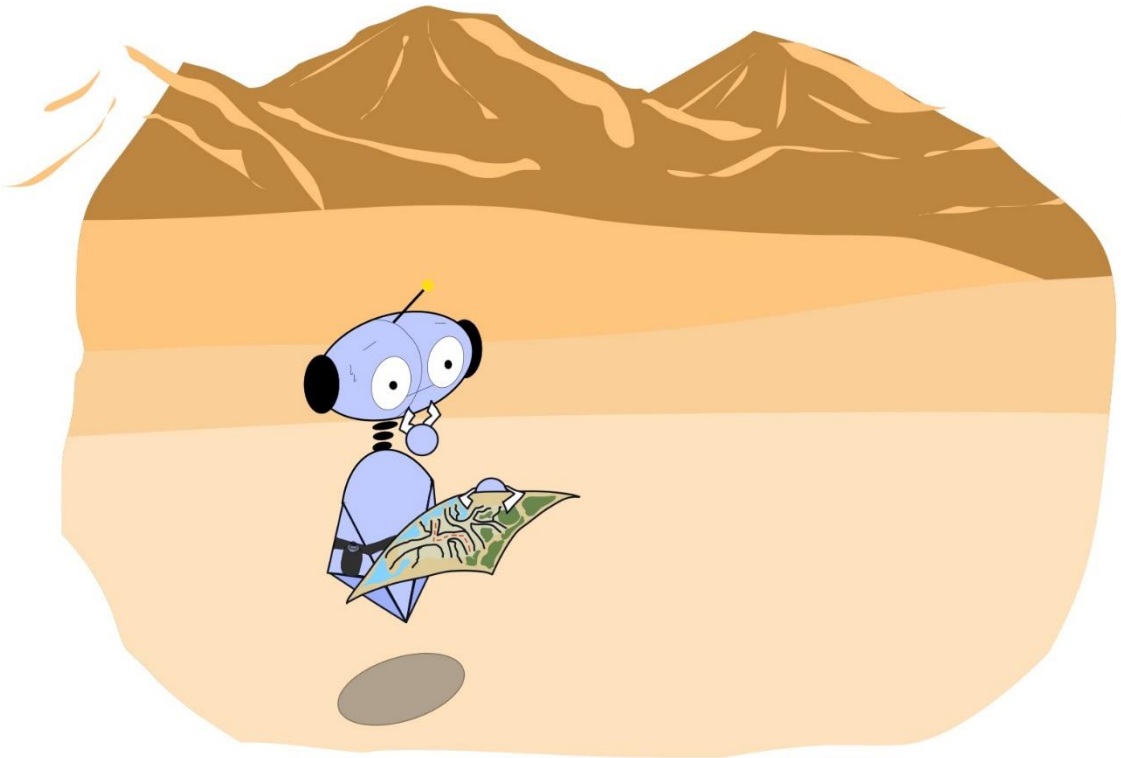
رئوس مطالب

- جستجوی آگاهانه
  - هیوریستیک‌ها
  - جستجوی حریصانه
  - جستجوی A*
- جستجوی گرافی

خلاصه: جستجو

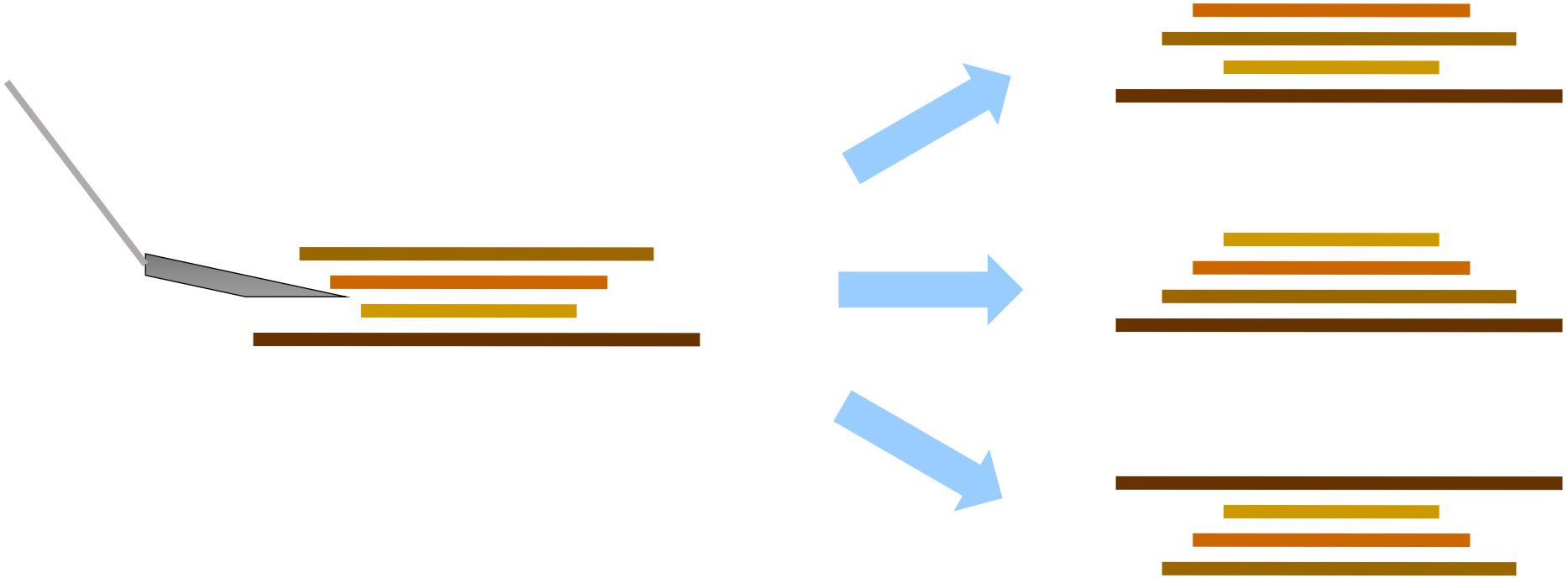


خلاصه: جستجو



- مسائل جستجو
  - حالت‌ها (پیکربندی جهان) states
  - اعمال و هزینه‌ها actions and costs
  - تابع پسین successor function
  - حالت شروع و حالت هدف start state and goal test
- درخت جستجو
  - گره‌ها: نشان دهنده برنامه برای رسیدن به حالت‌ها
  - برنامه‌ها دارای هزینه هستند (مجموع هزینه‌های اعمال)
- الگوریتم جستجو
  - به طور سیستماتیک یک درخت جستجو می‌سازد
  - انتخاب اولویت انتخاب از لبه (گره‌های ناشناخته)
  - بهینه: پیدا کردن کم‌هزینه‌ترین برنامه‌ها

مثال: مساله‌ی پَن‌کیک



هزینه: تعداد پن‌کیک‌های پشت و رو شده

مثال: مساله‌ی پَن‌کیک

BOUNDS FOR SORTING BY PREFIX REVERSAL

William H. GATES

Microsoft, Albuquerque, New Mexico

Christos H. PAPADIMITRIOU*†

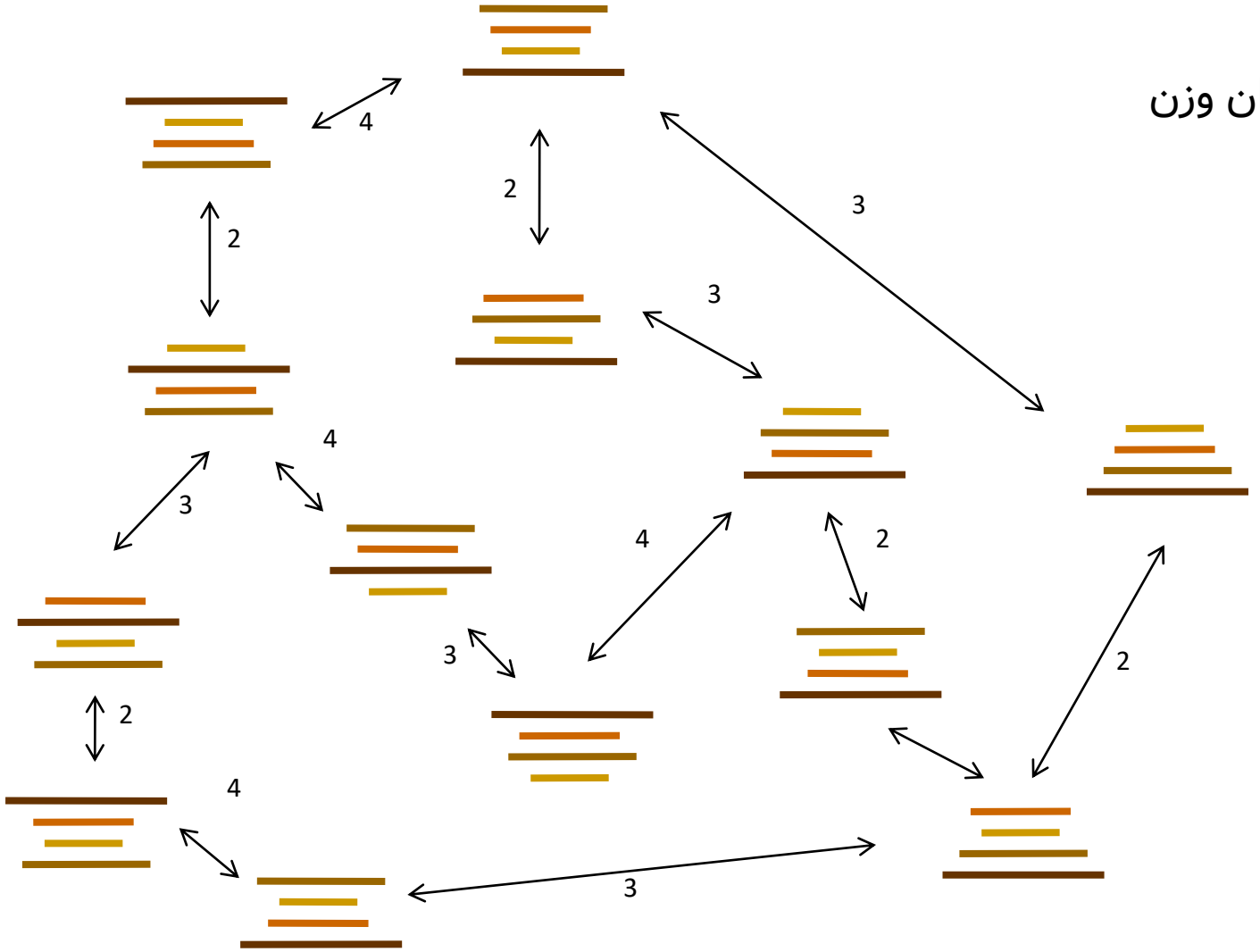
Department of Electrical Engineering, University of California, Berkeley, CA 94720, U.S.A.

Received 18 January 1978
Revised 28 August 1978

For a permutation σ of the integers from 1 to n, let $f(\sigma)$ be the smallest number of prefix reversals that will transform σ to the identity permutation, and let $f(n)$ be the largest such $f(\sigma)$ for all σ in (the symmetric group) S_n. We show that $f(n) \leqslant (5n+5)/3$, and that $f(n) \geqslant 17n/16$ for n a multiple of 16. If, furthermore, each integer is required to participate in an even number of reversed prefixes, the corresponding function $g(n)$ is shown to obey $3n/2 - 1 \leqslant g(n) \leqslant 2n+3$.

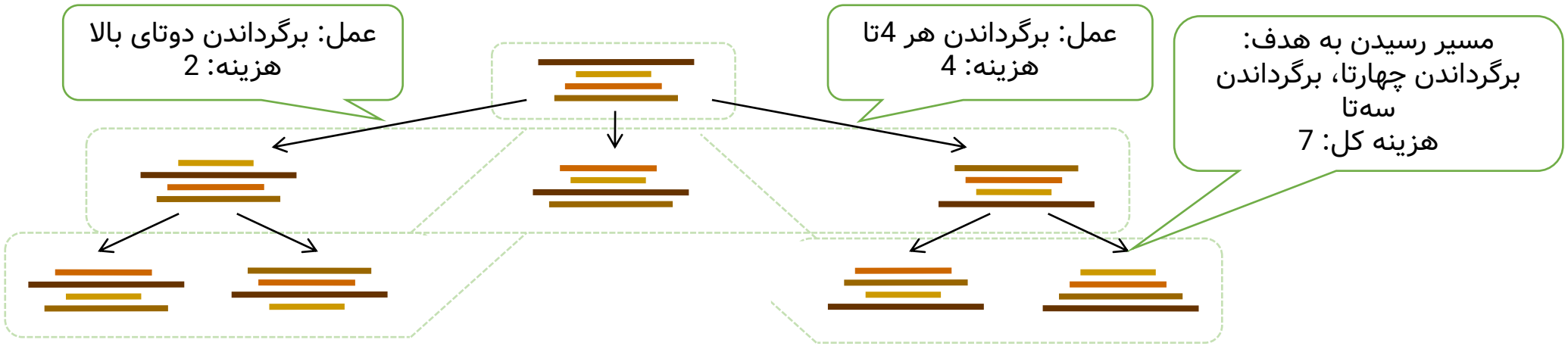
مثال: مساله‌ی پَن‌کیک

گراف فضای حالت با هزینه‌ها به عنوان وزن



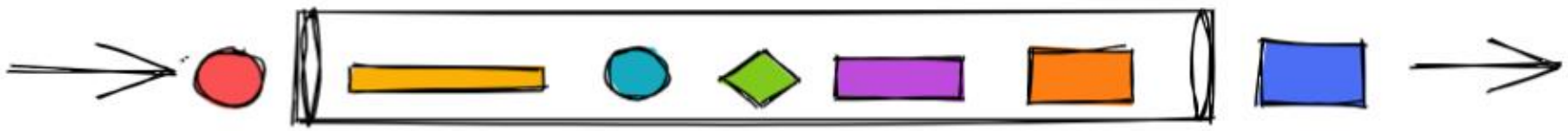
جستجوی درختی متداول

```
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
```

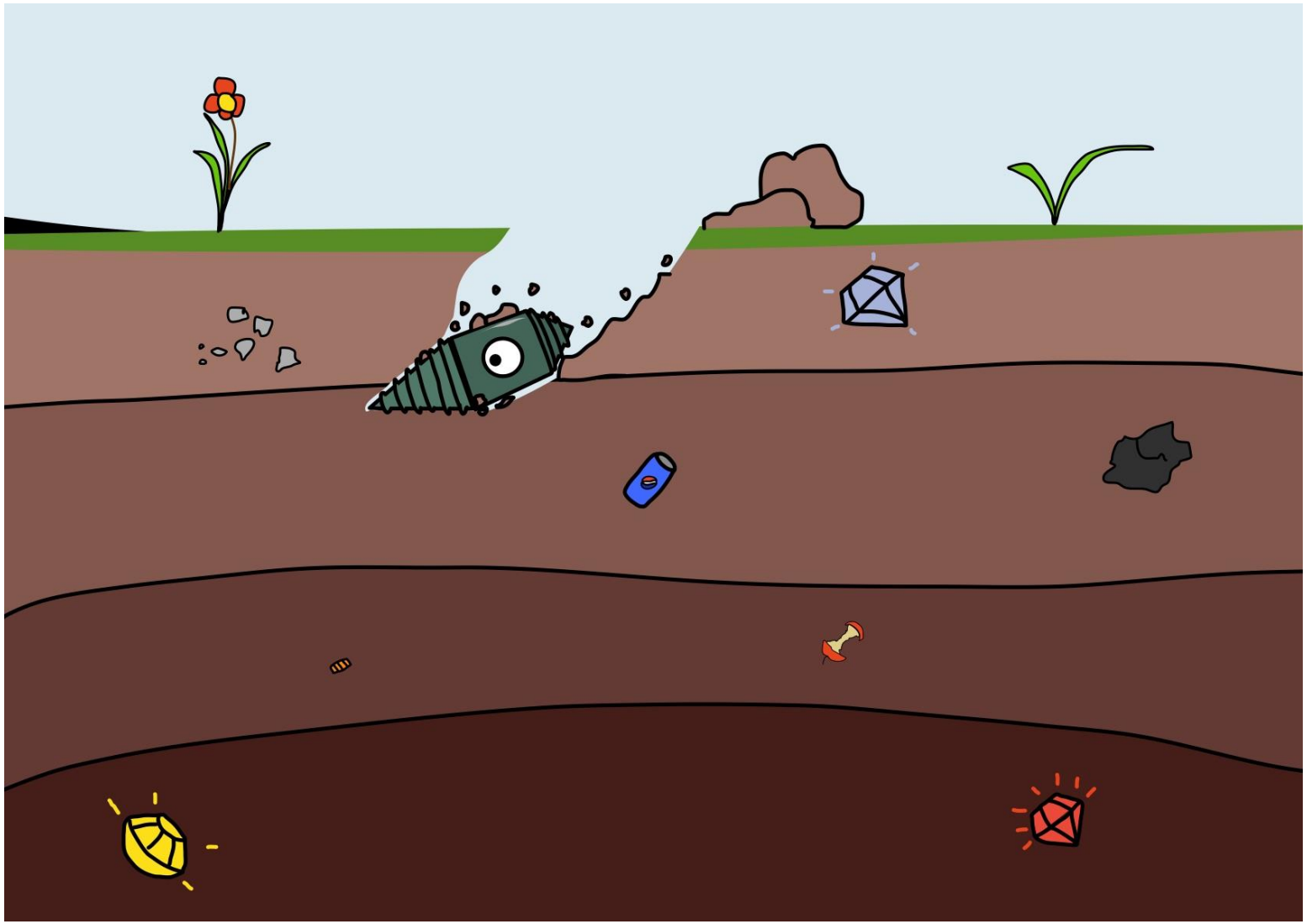


یک صف

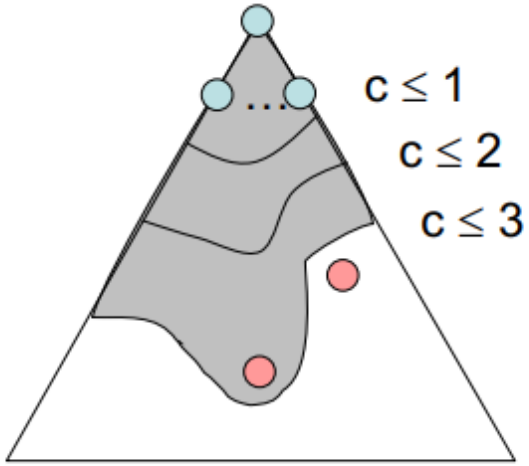
- همه این الگوریتم‌های جستجو به جز در استراتژی‌های لبه یکسان هستند
  - از نظر مفهومی، تمام لبه‌ها صف‌های اولویت هستند. (مثلا مجموعه‌ای از گره‌ها با اولویت‌های پیوست شده)
  - عملاً، برای DFS و BFS، می‌توانید با استفاده از پشته‌ها و صف‌ها، از سربار log(n) از یک صف اولویت اجتناب کرد
  - حتی می‌توان کد ثابتی برای یک پیاده‌سازی نوشت

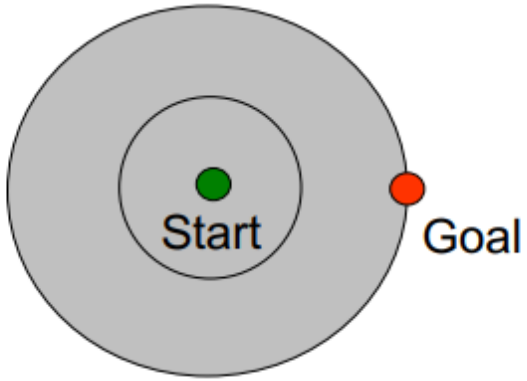


جستجوی ناآگاهانه



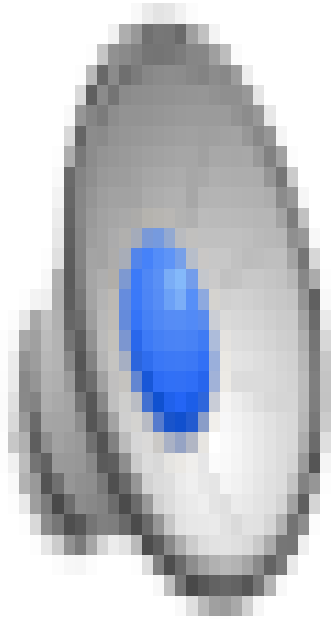
جستجوی هزینه یکنواخت



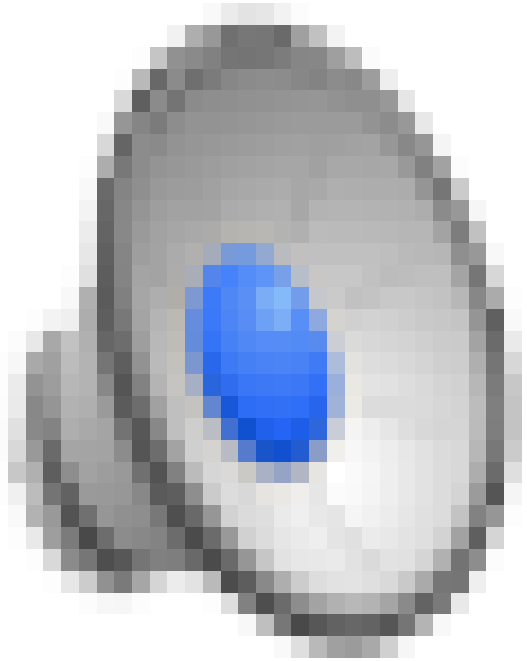


- استراتژی: مسیر با کمترین هزینه را گسترش می‌دهد
- خوبی: UCS کامل و بهینه است
- موارد بد:
  - گزینه‌ها را در همه جهت بررسی می‌کند
  - اطلاعاتی در مورد مکان هدف وجود ندارد

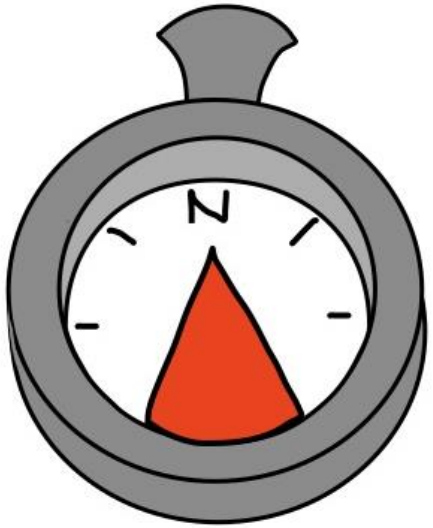
ویدیوی دموی کانتور UCS خالی

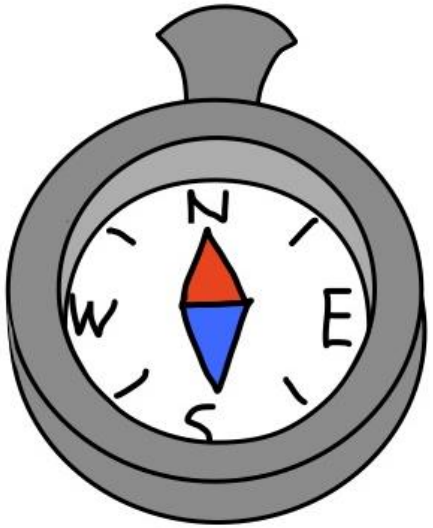


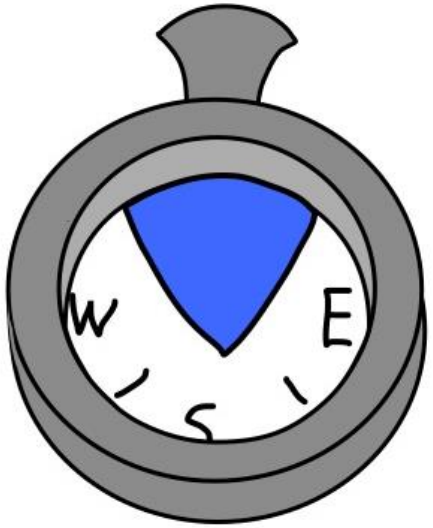
ویدیوی دموی کانتور UCSماز کوچک pacman



جستجوی آگاهانه

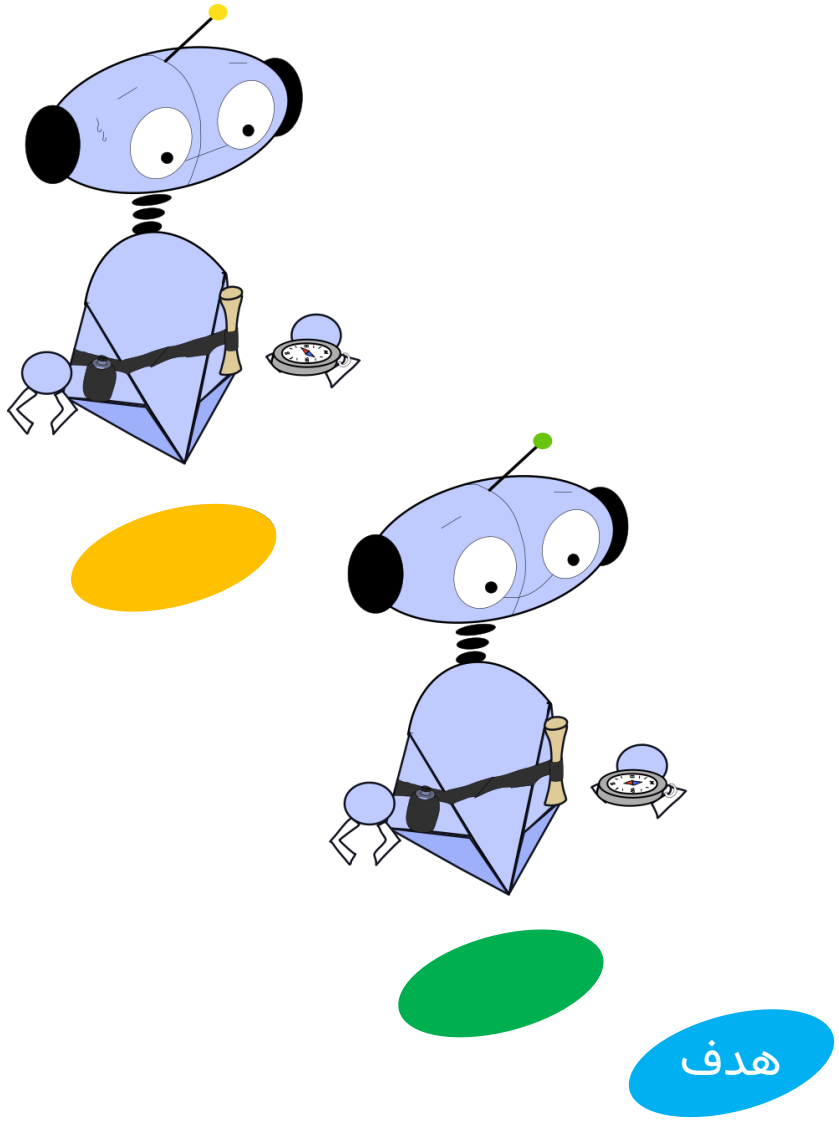


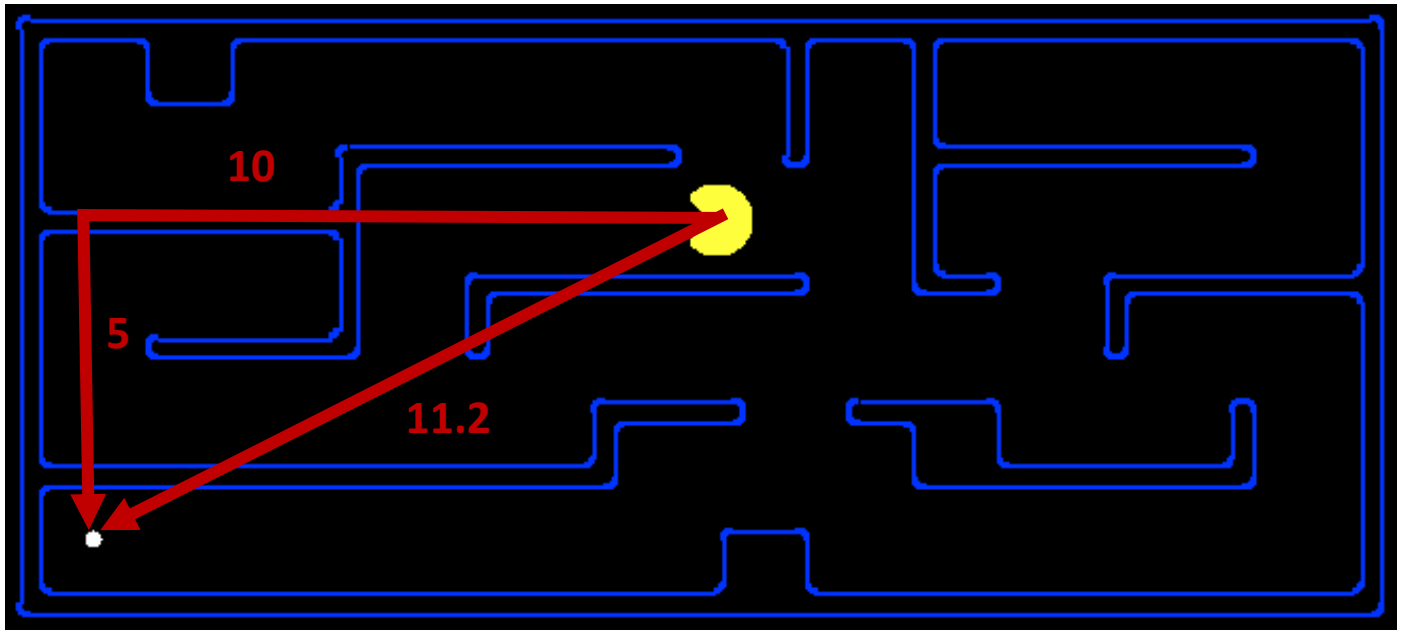




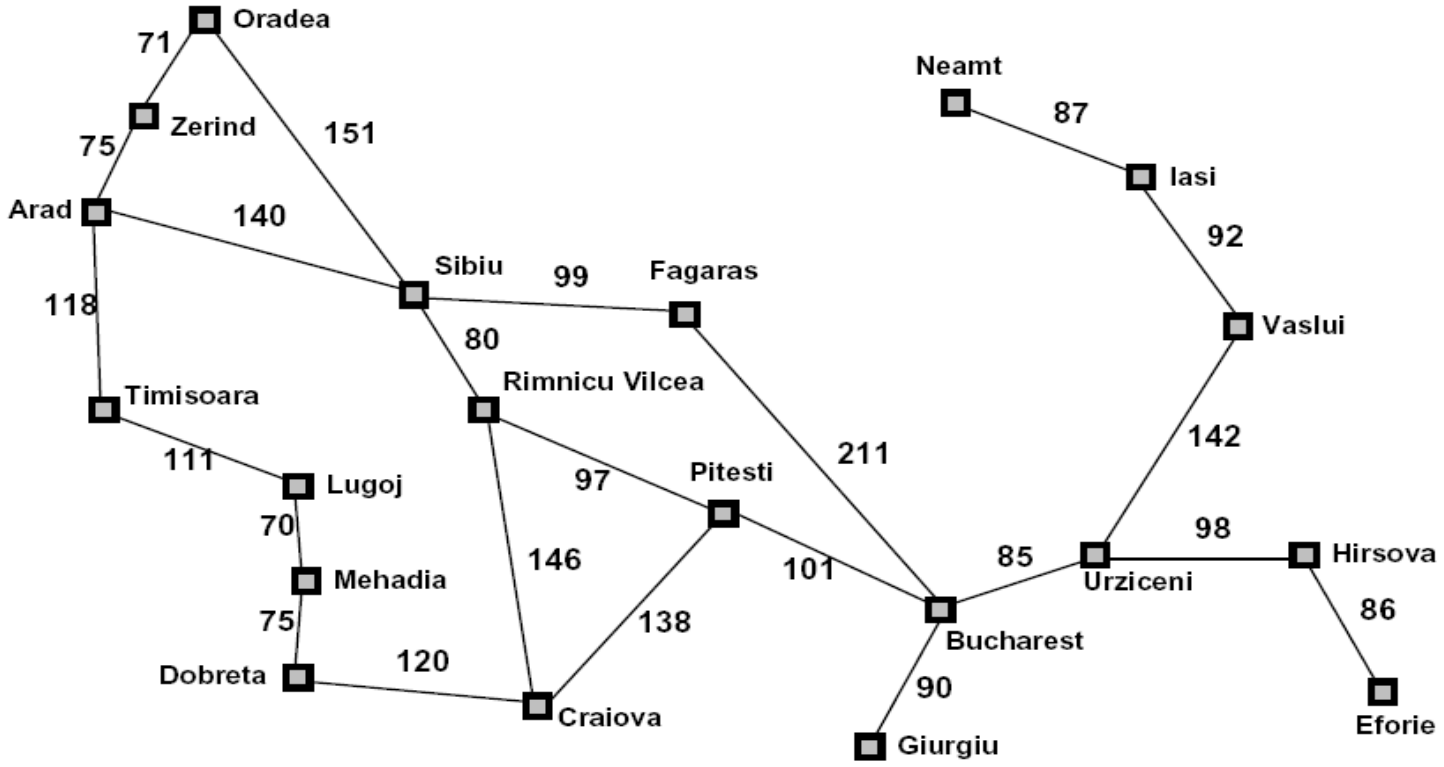
هیوریستیک‌های جستجو

- یک هیوریستیک عبارت است از:
  - تابعی که تخمین می‌زند که یک حالت چقدر به هدف نزدیک است
  - برای یک مساله جستجوی خاص طراحی شده است
  - مثال: فاصله منهتن و فاصله اقلیدسی برای مسئله مسیریابی





مثال: تابع هیوریستیک

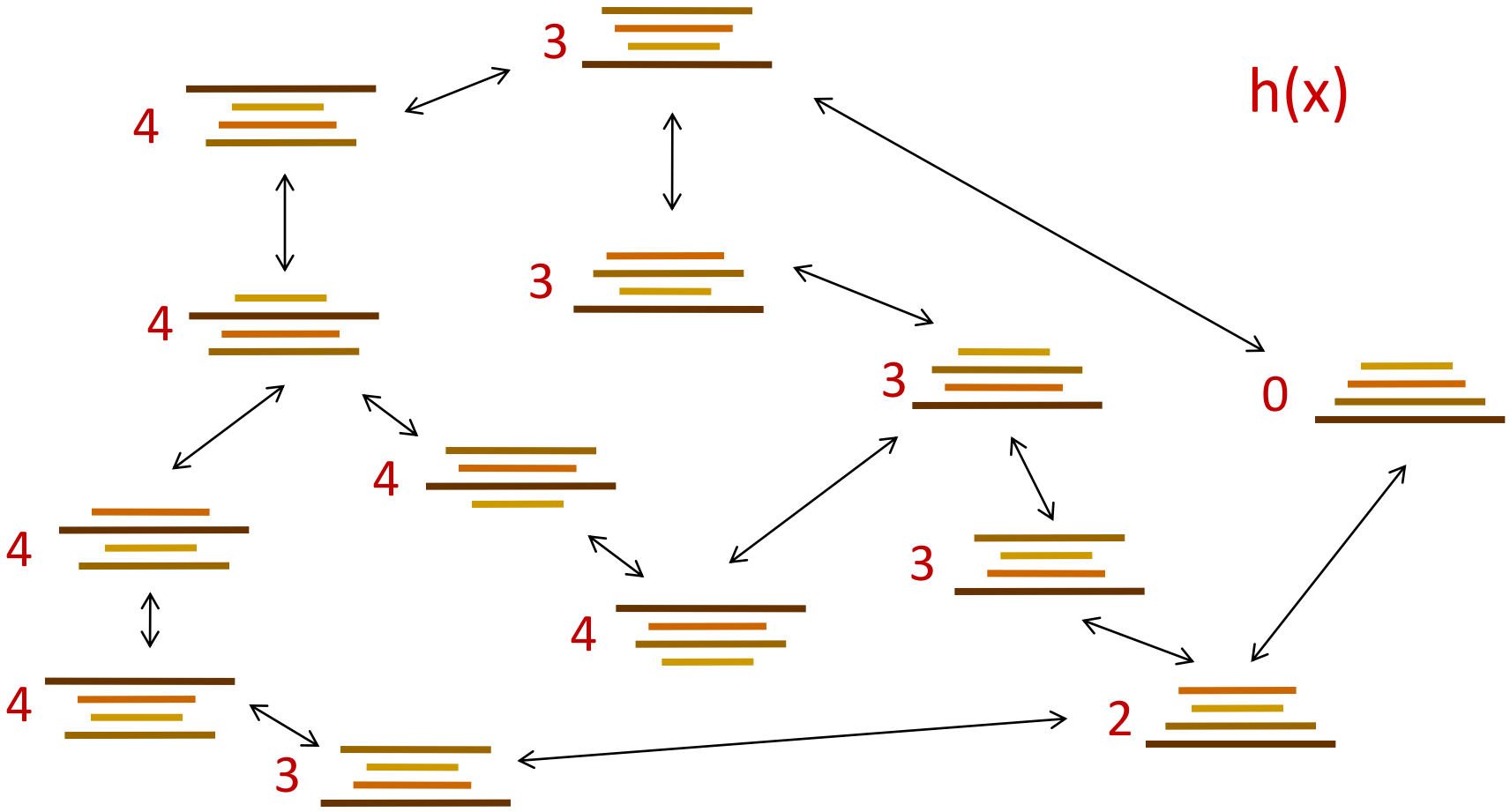


Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

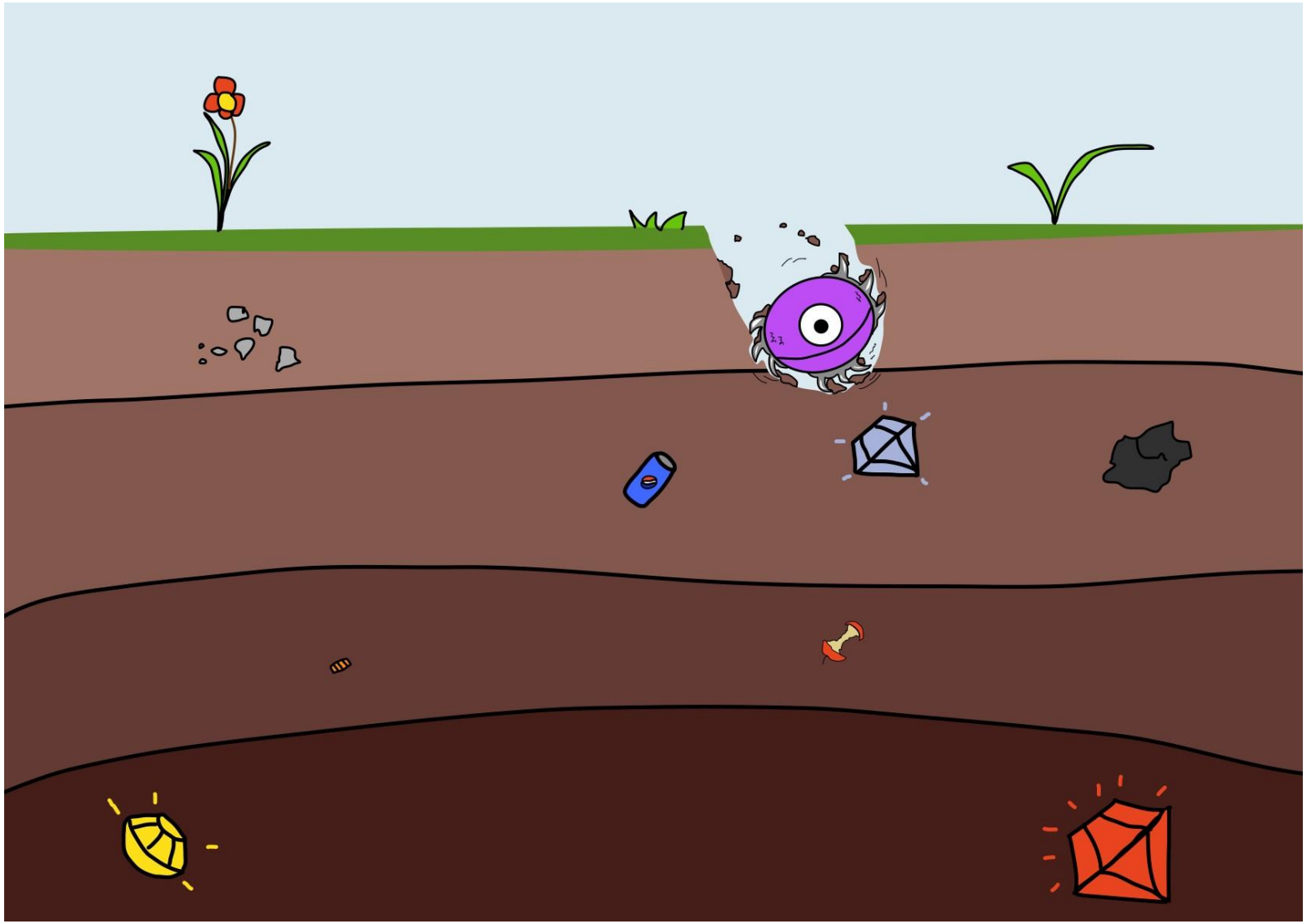
h(x)

مثال: تابع هیوریستیک

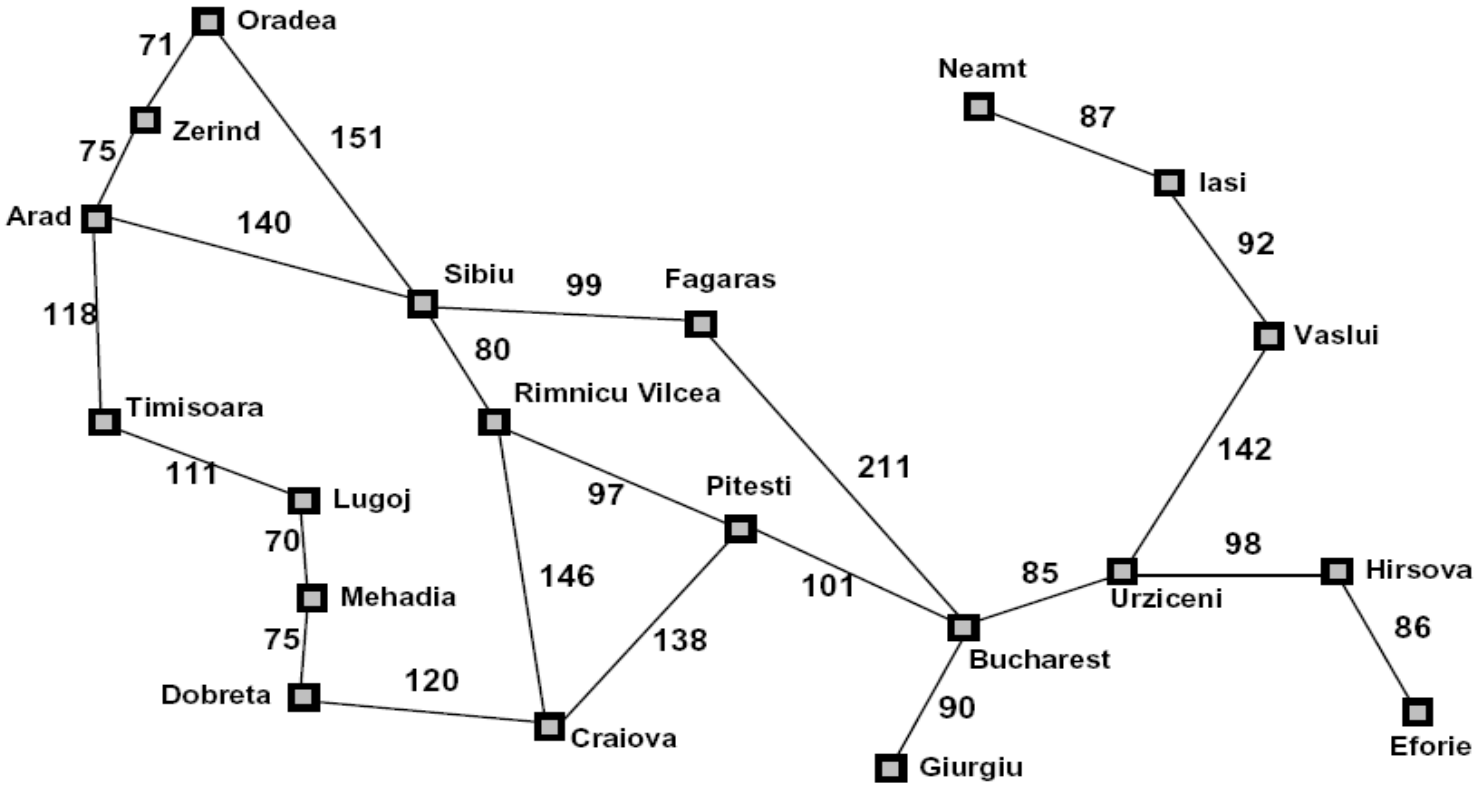
هیوریستیک: تعداد بزرگترین پنکیک که هنوز در جای خود نیست.



جستجوی حریصانه



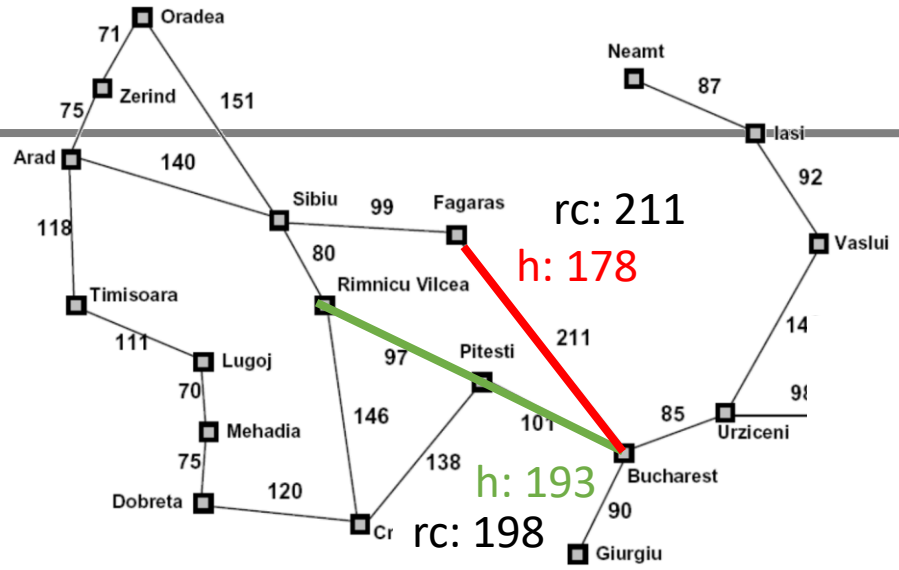
جستجوی حریصانه



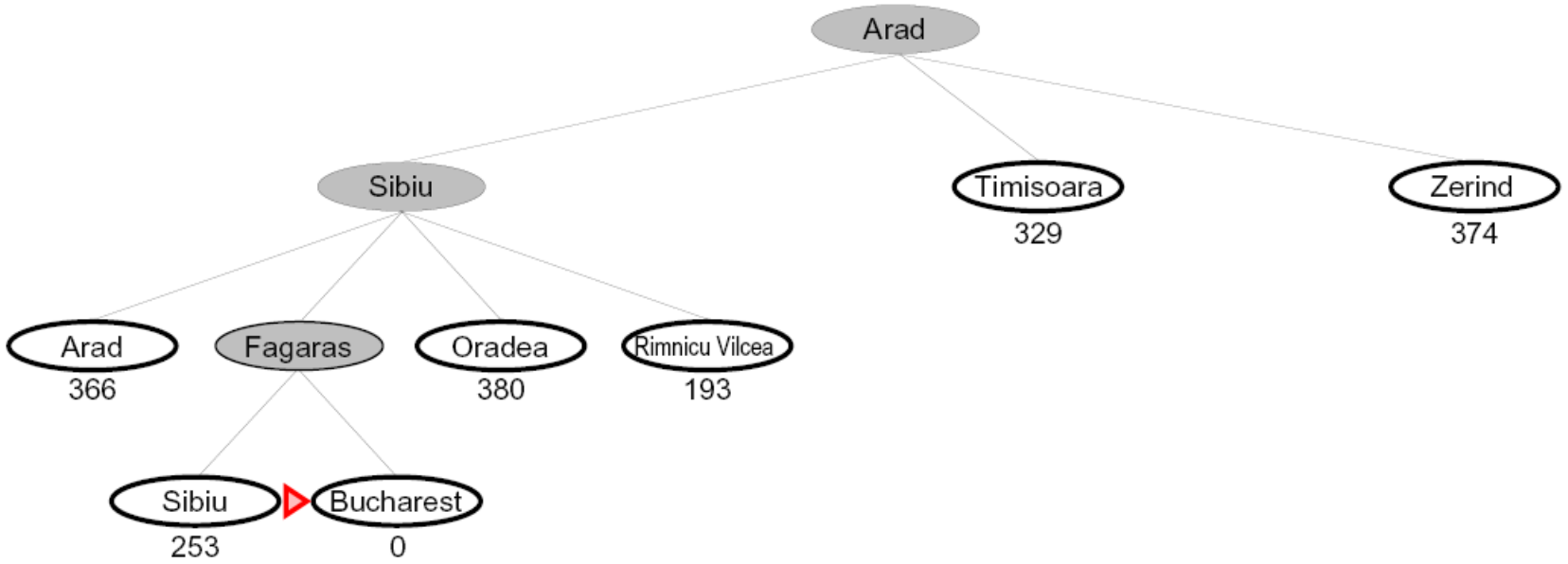
Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

h(x)

مثال: تابع هیوریستیک

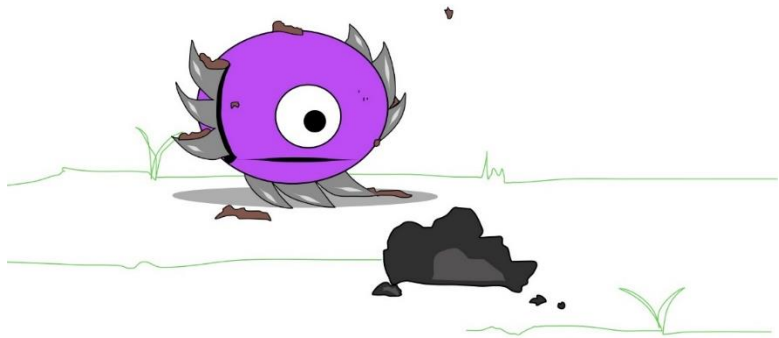


- گره‌ای را که نزدیک‌تری به هدف بنظر می‌رسد گسترش دهید...



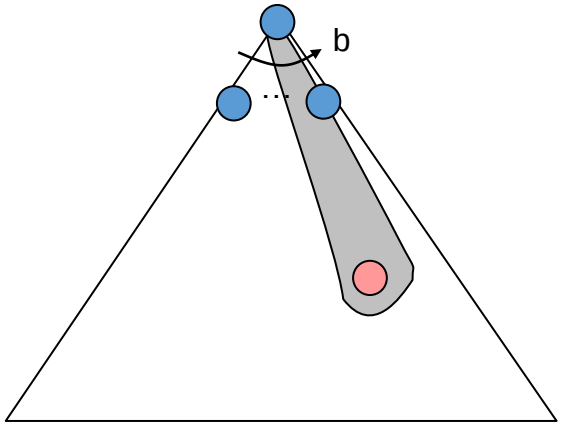
Straight−line distance to Bucharest

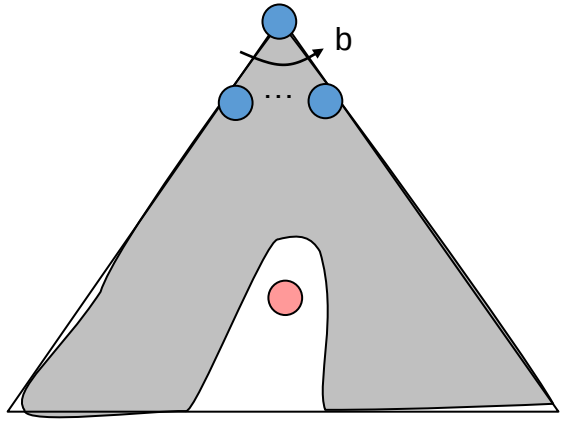
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374



- کجای کار می‌تواند خطا برود؟

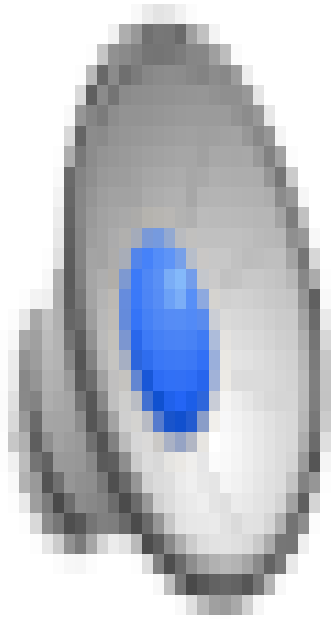
جستجوی حریصانه



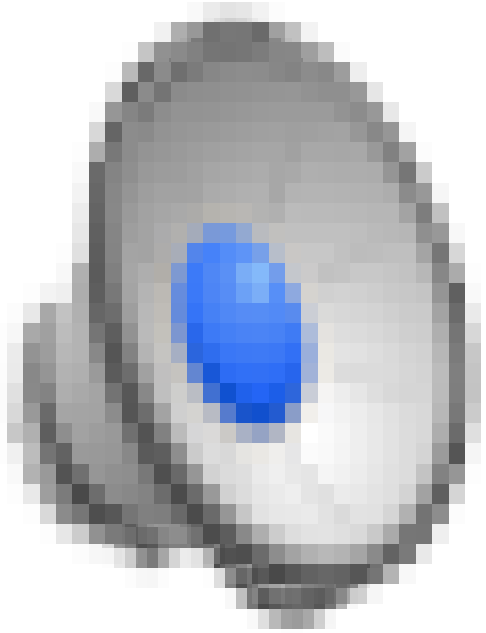


- استراتژی: گره‌ای را که فکر می‌کنید به یک حالت هدف نزدیک‌تر است، گسترش دهید
  - هیوریستیک: تخمین فاصله تا نزدیکترین هدف برای هر حالت
- حالت رایج و مورد انتظار:
  - بهترین/اولین حدس شما را مستقیماً به سمت هدف (اشتباه) می‌برد
- بدترین حالت: مثل یک DFS با هدایت خیلی بد

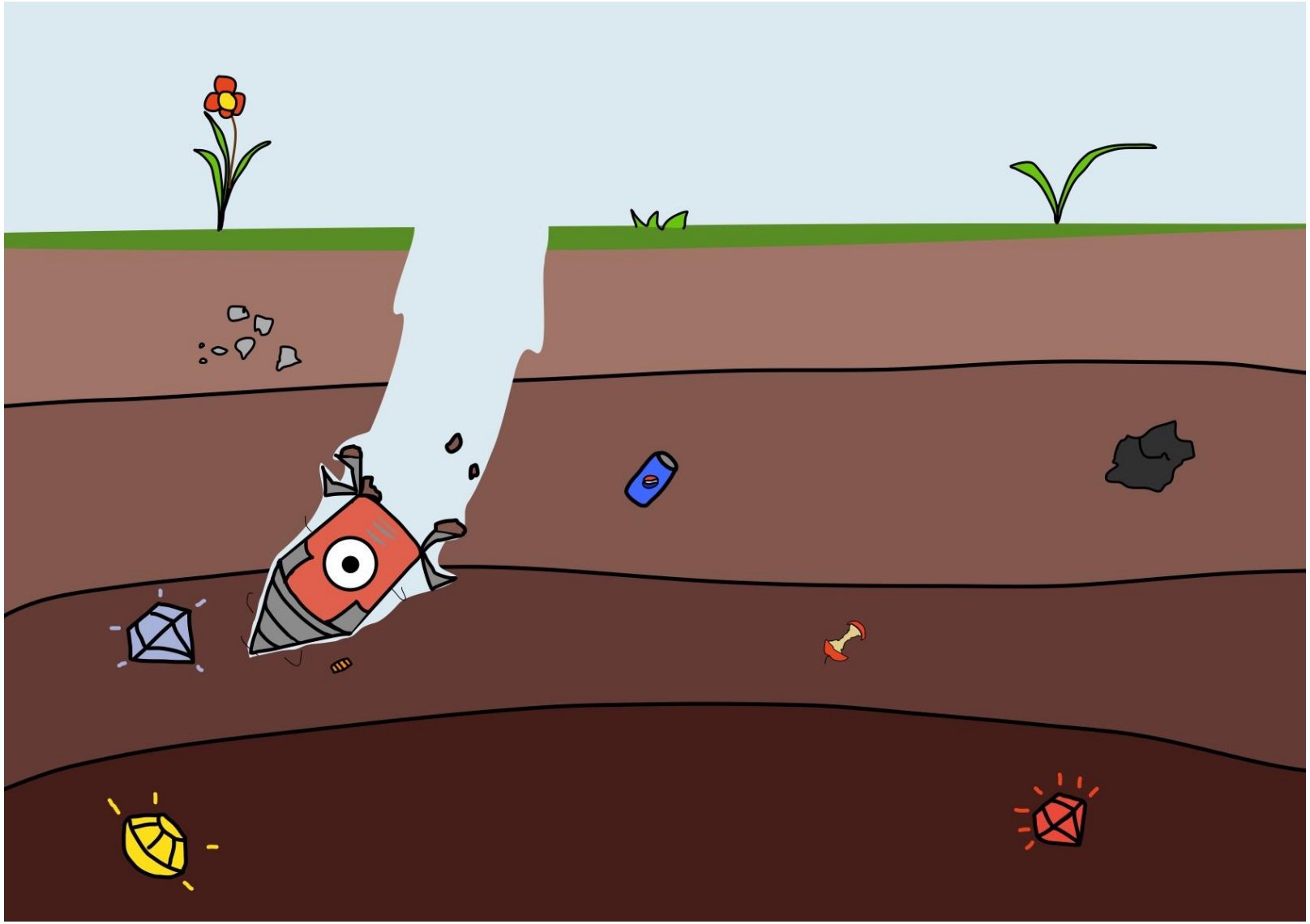
ویدیوی دموی کانتور حریصانه (صحنه خالی)



ویدیوی دموی کانتور حریصانه (مارپیچ کوچک pacman)



جستجوی A*



جستجوی A*

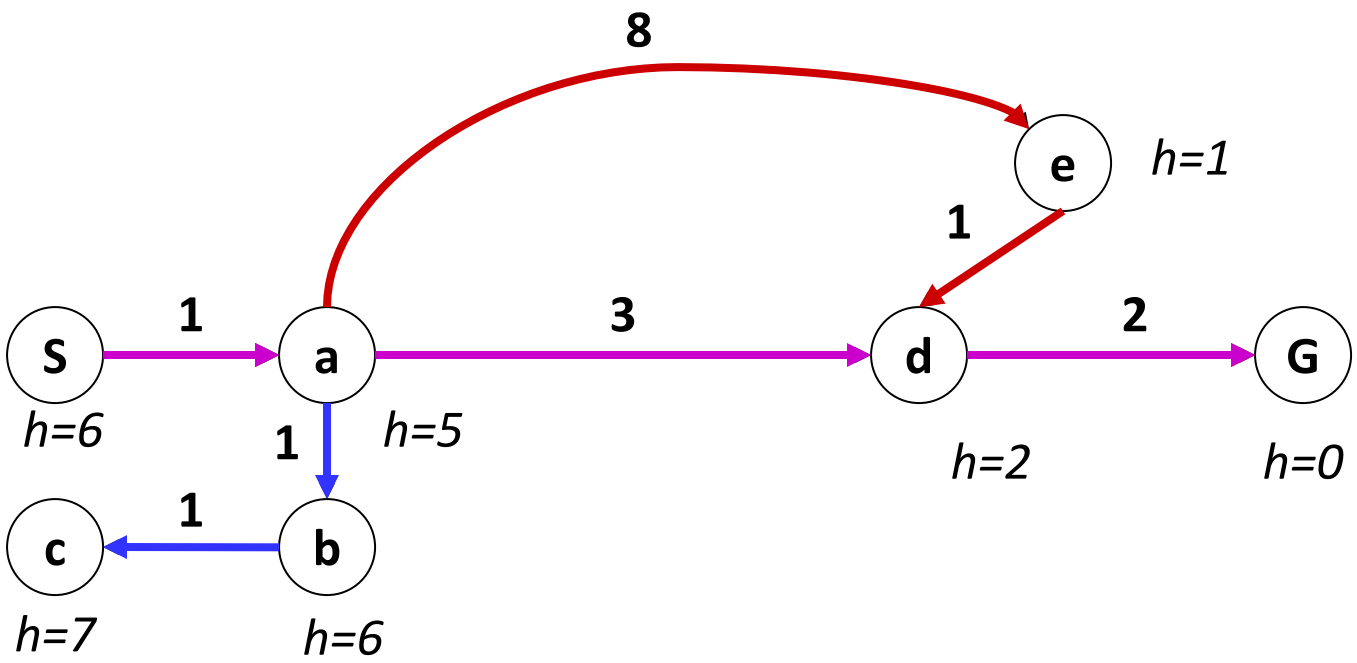
UCS

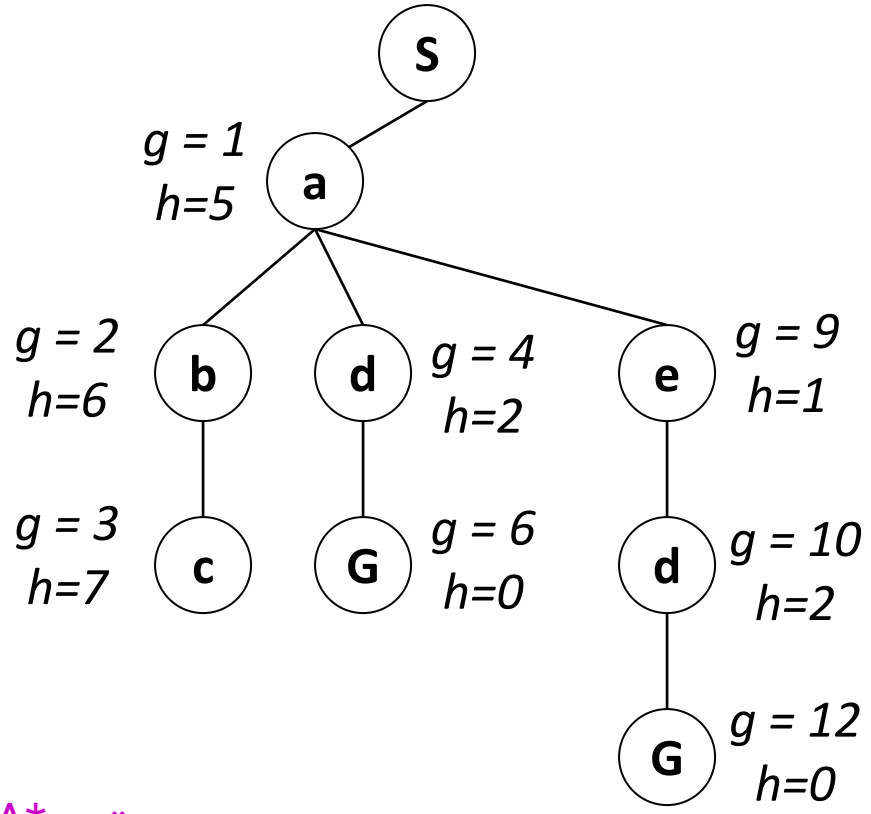
A*

حریصانه

ترکیب کردن UCS و حریصانه

- جستجوی هزنیه یکنواخت بر اساس هزینه‌ی مسیر یا هزینه رو به عقب **g(n)** اولویت‌بندی می‌کند *backward cost*
- جستجوی حریصانه بر اساس نزدیکی به هدف یا هزینه پیش‌رو **h(n)** اولویت‌بندی می‌کند *forward cost*



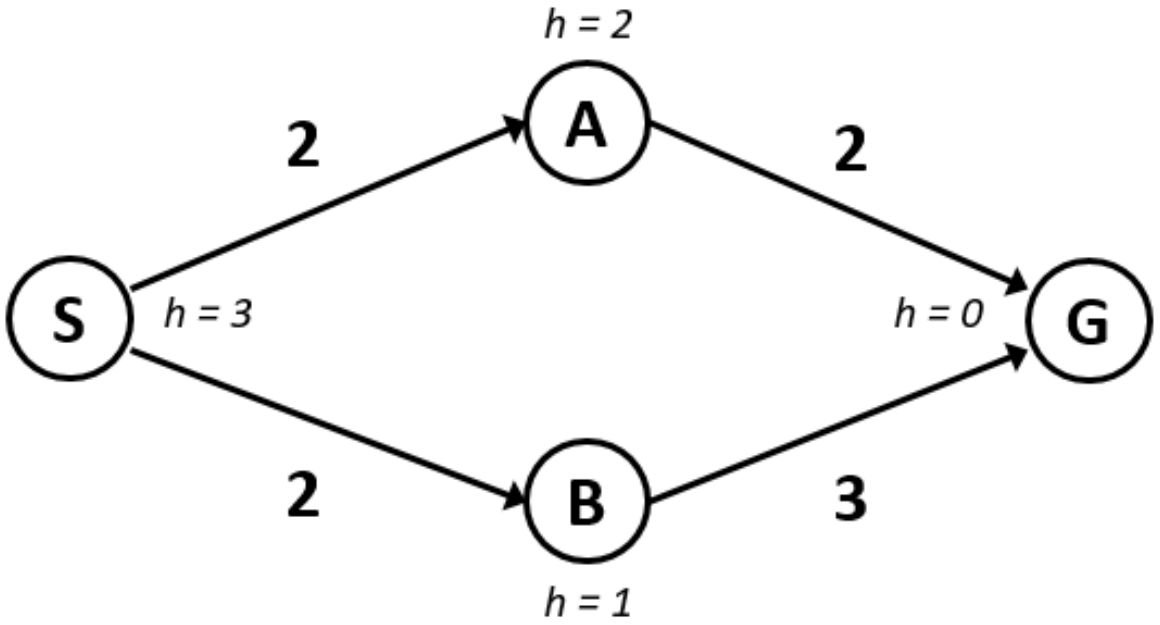


- جستجو A* بر اساس مجموع: f(n)= g(n) + h(n) اولویت‌بندی می‌کند

Example: Teg Grenager

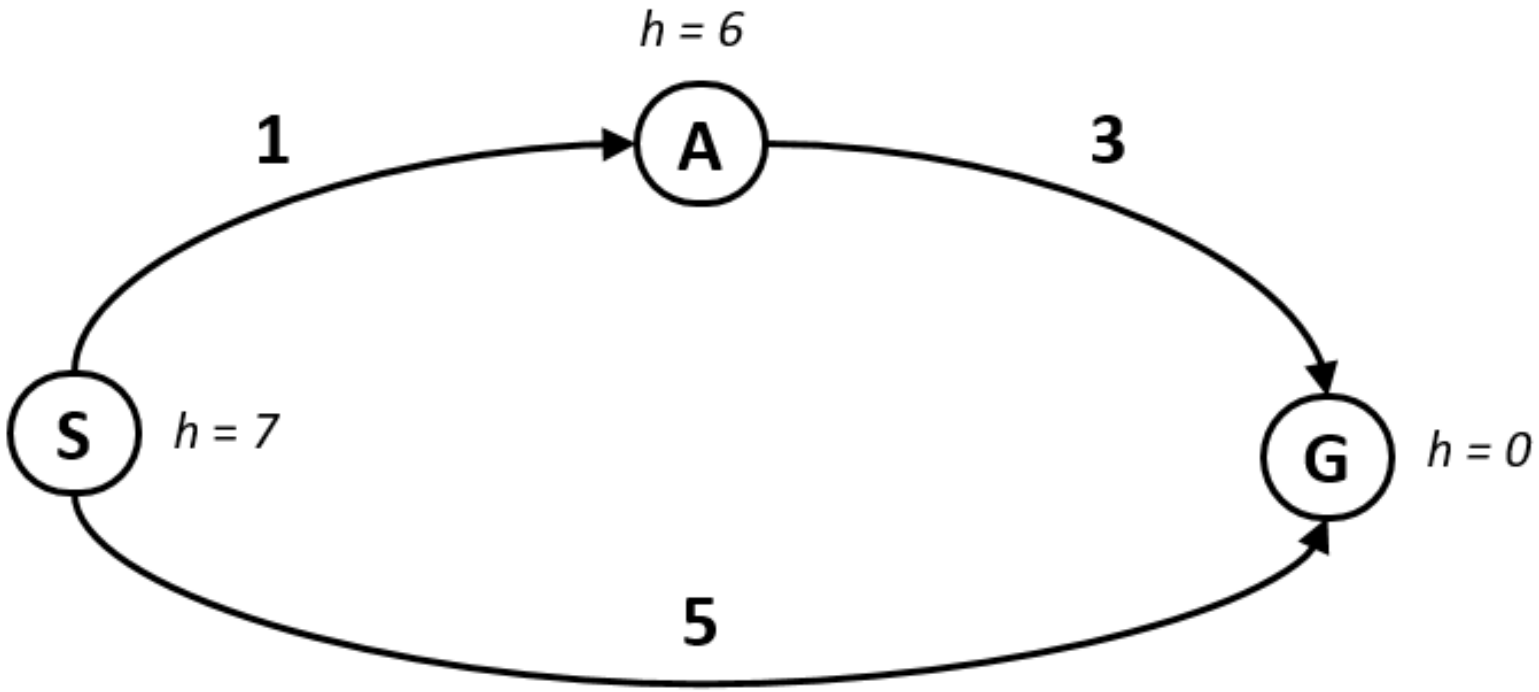
چه زمانی باید A* خاتمه یابد؟

- آیا وقتی هدفی را در صف قرار می‌دهیم باید توقف کنیم؟



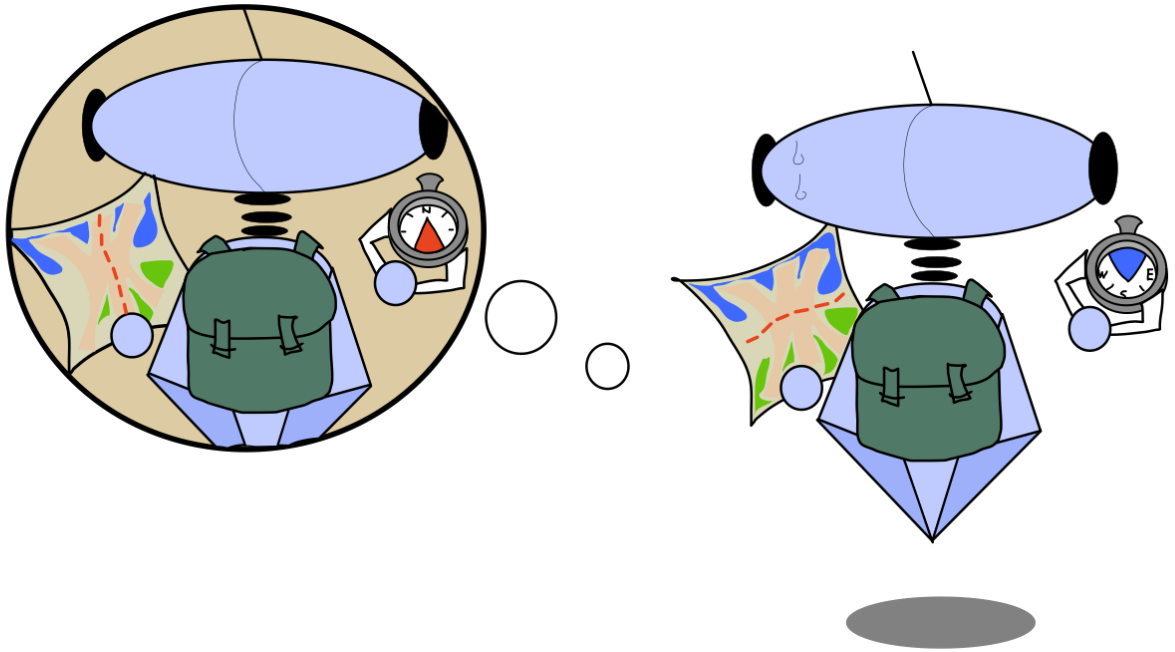
- خیر: فقط زمانی متوقف می‌شویم که هدفی را از صف خارج کنیم

آیا A* بهینه است؟

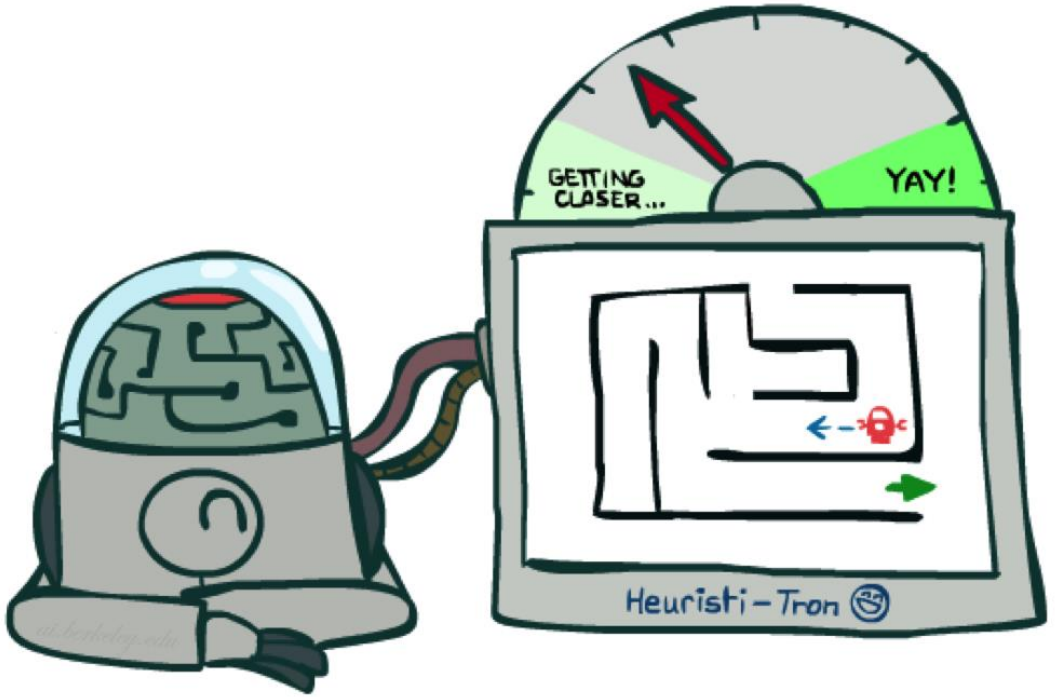


- کجای کار اشتباه پیش رفت؟
- هزینه واقعی هدف بد < برآورد هزینه هدف خوب
- ما به برآوردهایی نیاز داریم که کمتر از هزینه‌های واقعی باشد!

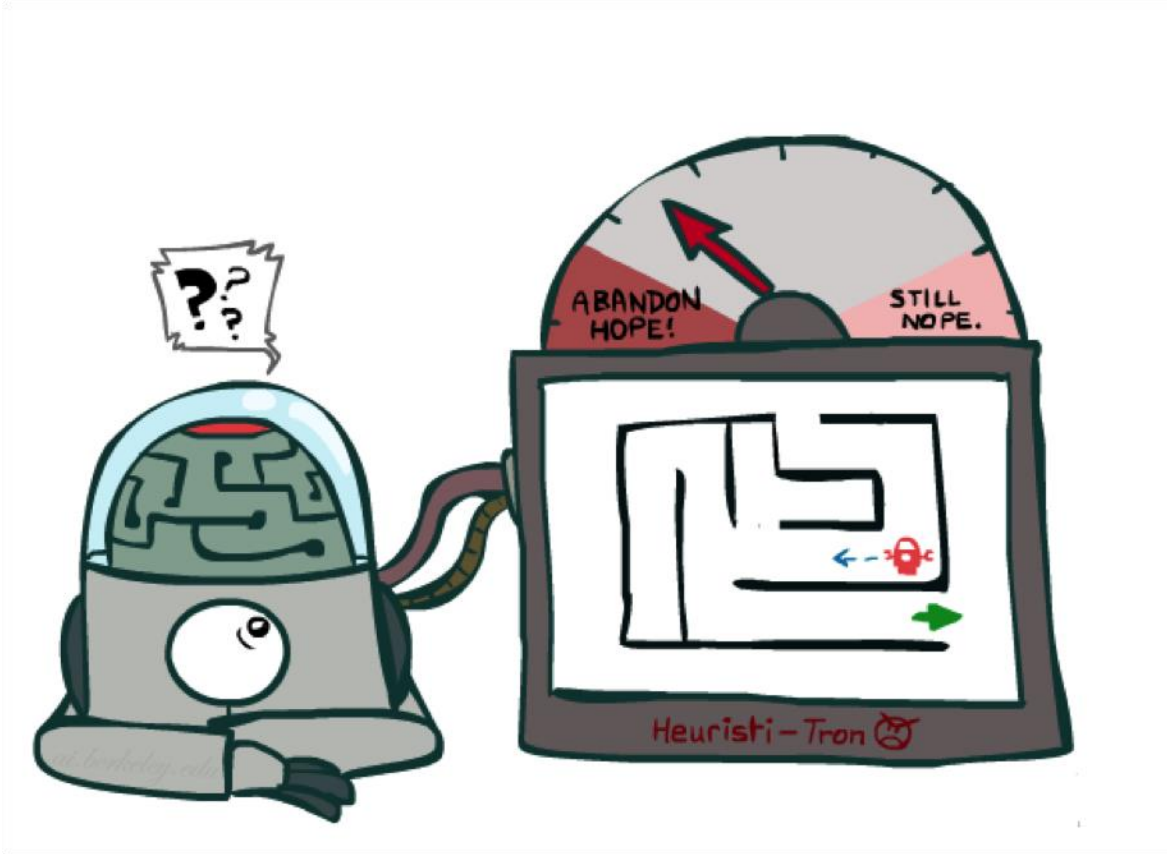
هیوریستیک قابل قبول Admissible heuristics



ایده: قابل قبول بودن Admissibility



- هیوریستیک قابل قبول (خوشبینانه) برنامه‌های بد را کند می‌کند اما هرگز از هزینه‌های واقعی بیشتر نمی‌شود



- هیوریستیک‌های غیرقابل قبول (بدبینانه) با به دام انداختن برنامه‌های خوب در لبه، بهینگی را از بین می‌برد

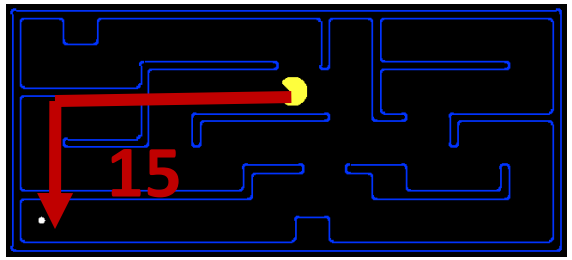
هیوریستیک قابل قبول

- هیوریستیک h قابل قبول (خوشبینانه) است، اگر:

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

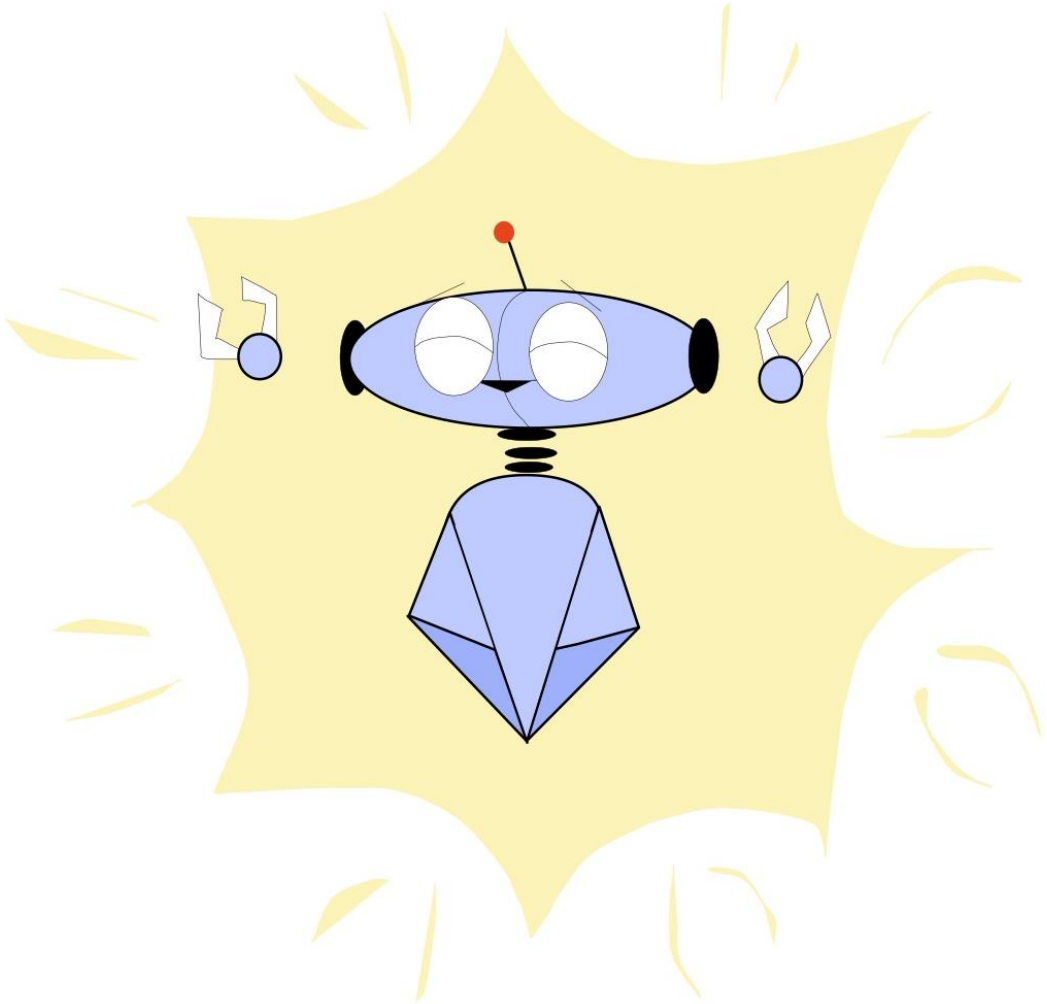
که در آن، $h^*(n)$ هزینه واقعی نزدیک‌ترین هدف است

- مثال‌ها:

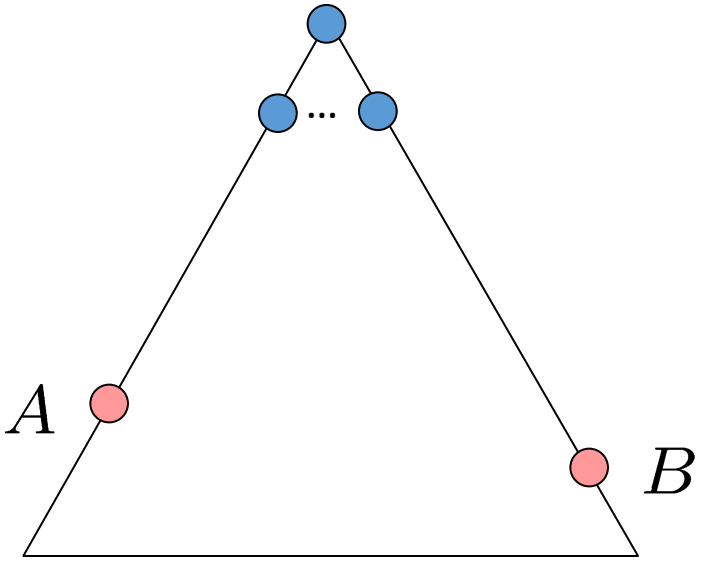


- به دست آوردن یک هیوریستیک قابل قبول مهم‌ترین کاری است که برای استفاده از A* در عمل نیاز داریم

بهینه بودن جستجوی درختی A*

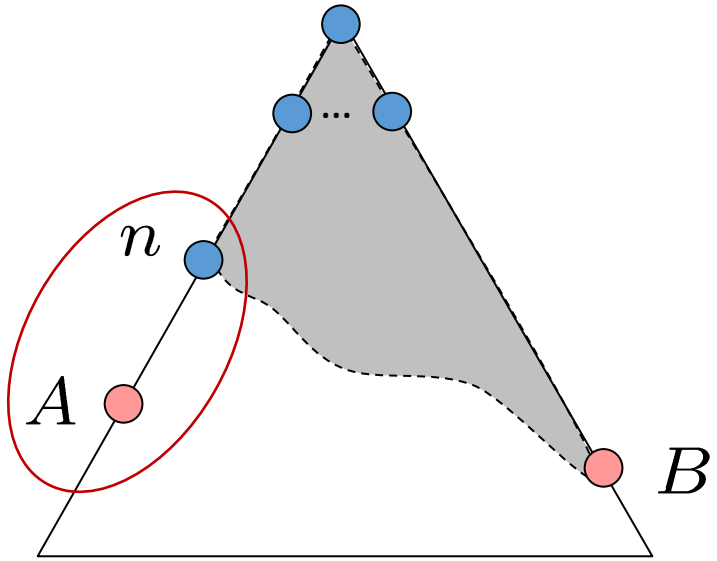


بهینه بودن جستجوی درختی A*



- فرض کنید:
  - A یک گره هدف بهینه است
  - B یک گره هدف غیربهینه است
  - h قابل قبول است
- ادعا: A قبل از B از لیست لبه خارج می‌شود

بهینه بودن جستجوی درختی A*: مسدود کردن



اثبات:

- تصور کنید B در لبه است
- یکی از اجداد A بنام n نیز در لبه هست (شاید خود A!)
- ادعا : n قبل از b بسط داده خواهد یافت (خروج از لبه)

1. f(n) کمتر یا مساویست با f(A)

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

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

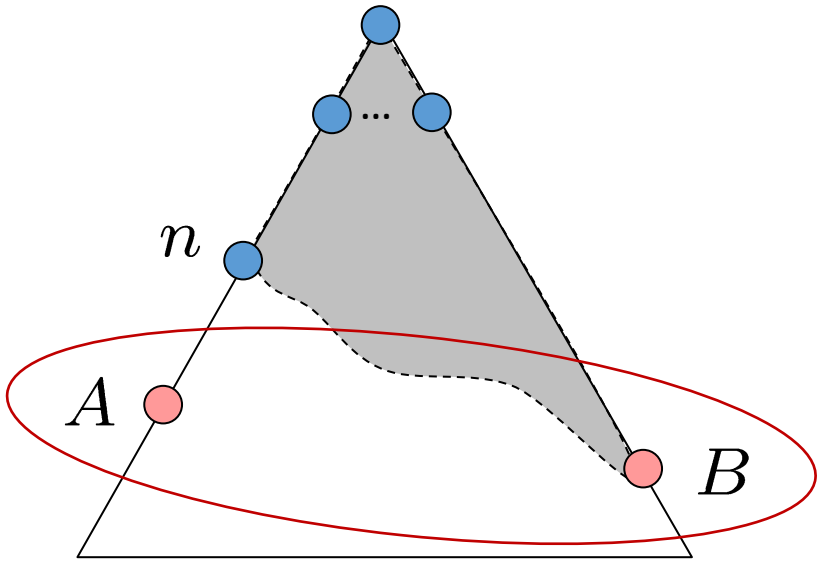
$$g(A) \ge g(n) + h(n)$$

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

$$f(n) \le g(A)$$ قابل قبول بودن h

$$g(A) = f(A)$$ h = 0 در هدف

بهینه بودن جستجوی درختی A*: مسدود کردن



اثبات:

- تصور کنید B در لبه است
- یکی از اجداد A بنام n نیز در لبه هست (شاید خود A!)
- ادعا : n قبل از b بسط داده خواهد یافت (خروج از لبه)

1. f(n) کمتر یا مساویست با f(A)
2. f(A) کمتر است از f(B)

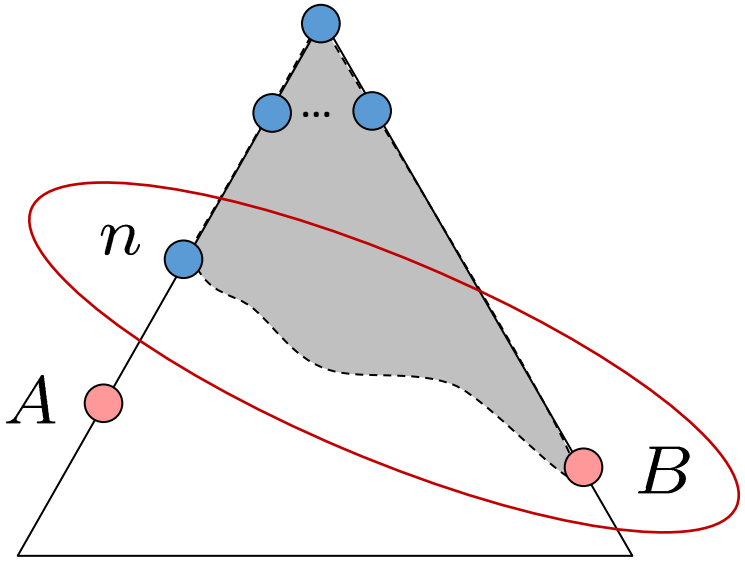
B هدف غیر بهینه است

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

h = 0 در هدف

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

بهینه بودن جستجوی درختی A*: مسدود کردن



اثبات:

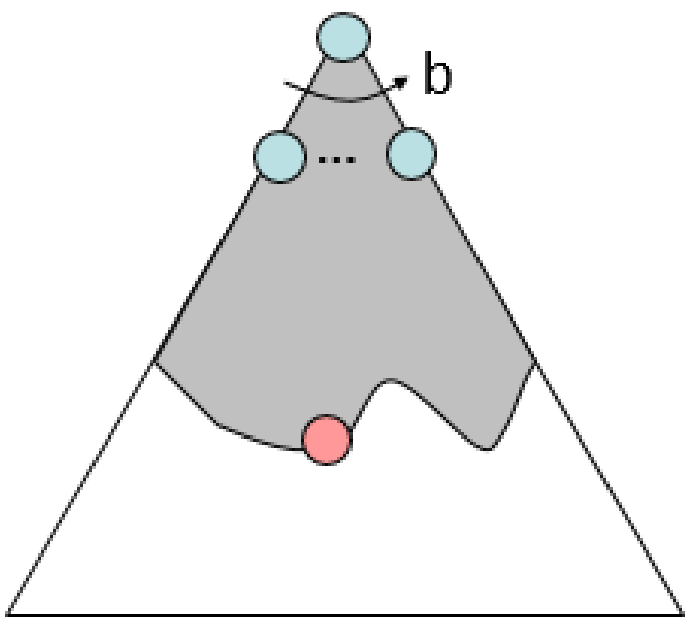
- تصور کنید B در لبه است
- یکی از اجداد A بنام n نیز در لبه هست (شاید خود A!)
- ادعا : n قبل از b بسط داده خواهد یافت (خروج از لبه)
  1. f(n) کمتر یا مساویست با f(A)
  2. f(A) کمتر است از f(B)
  3. n قبل از B بسط می‌یابد
- همه اجداد A قبل از B بسط می‌یابد
- A قبل از B بسط می‌یابد
- جستجوی A* بهینه است

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

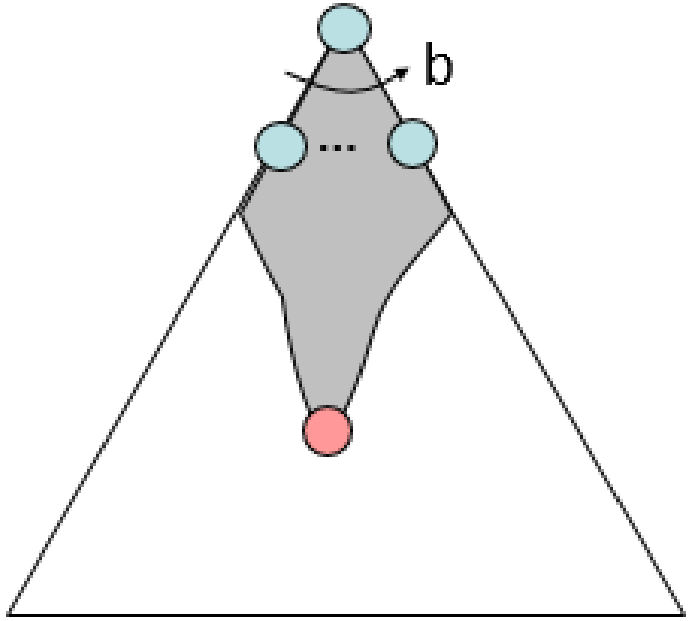
ویژگی‌های A*

ویژگی‌های A*

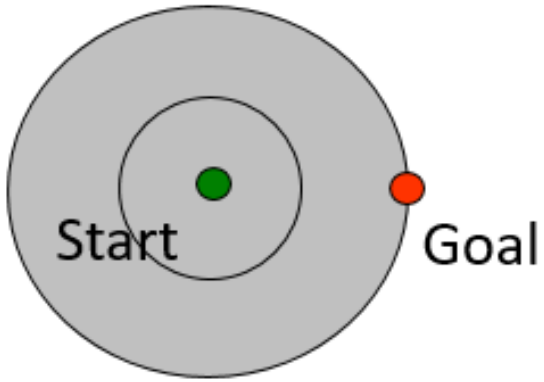
هزینه یکنواخت



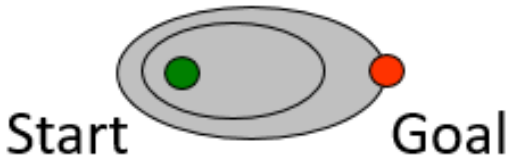
A*



کانتورهای UCS در مقابل A*

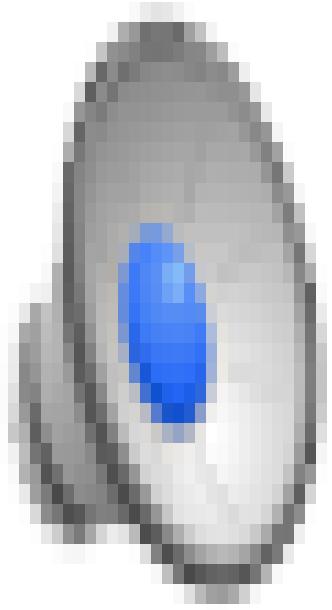


- هزینه یکنواخت به طور یکسان در همه "جهت‌ها" بسط می‌یابد

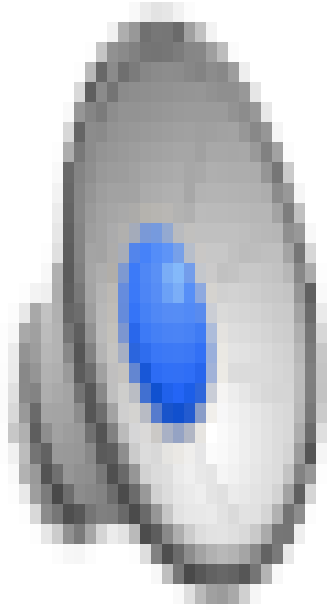


- A* عمدتاً به سمت هدف بسط می‌یابد، اما برای اطمینان از بهینه بودن، با احتیاط عمل می‌کند

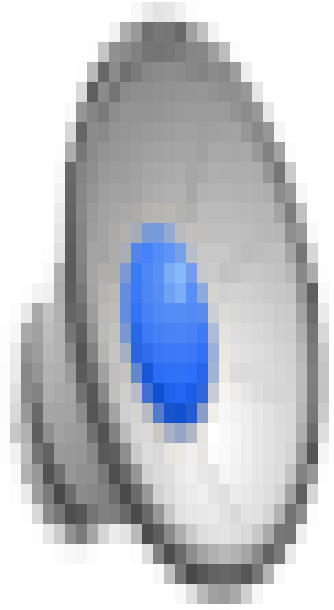
ویدیوی دموی کانتورها (خالی) – UCS



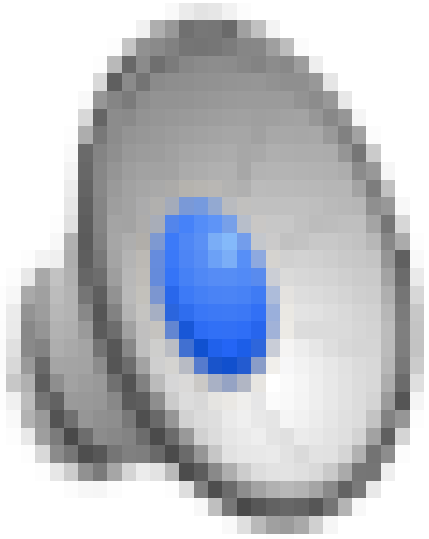
ویدیوی دموی کانتورها (خالی) - حریصانه



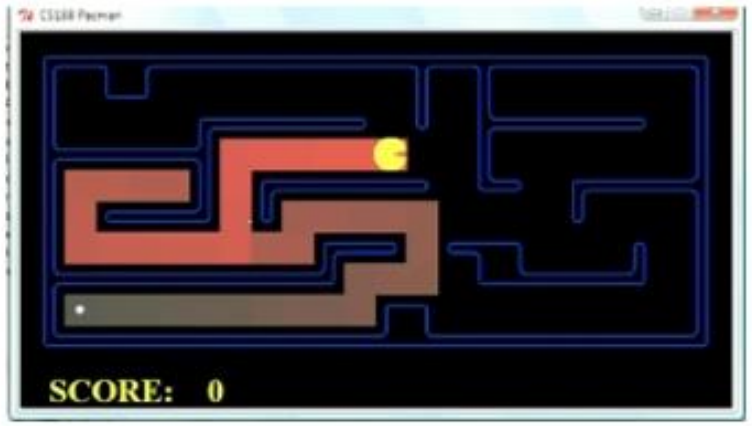
ویدیوی دموی کانتورها (خالی) – A*



ویدیوی دموی کانتورها (Pacman small maze) – A*

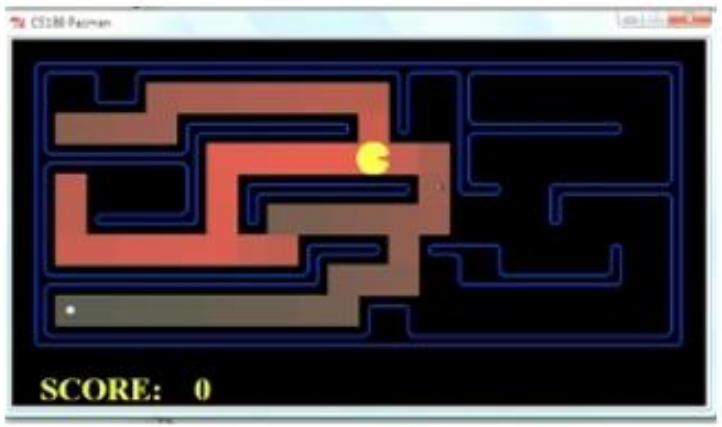


مقایسه



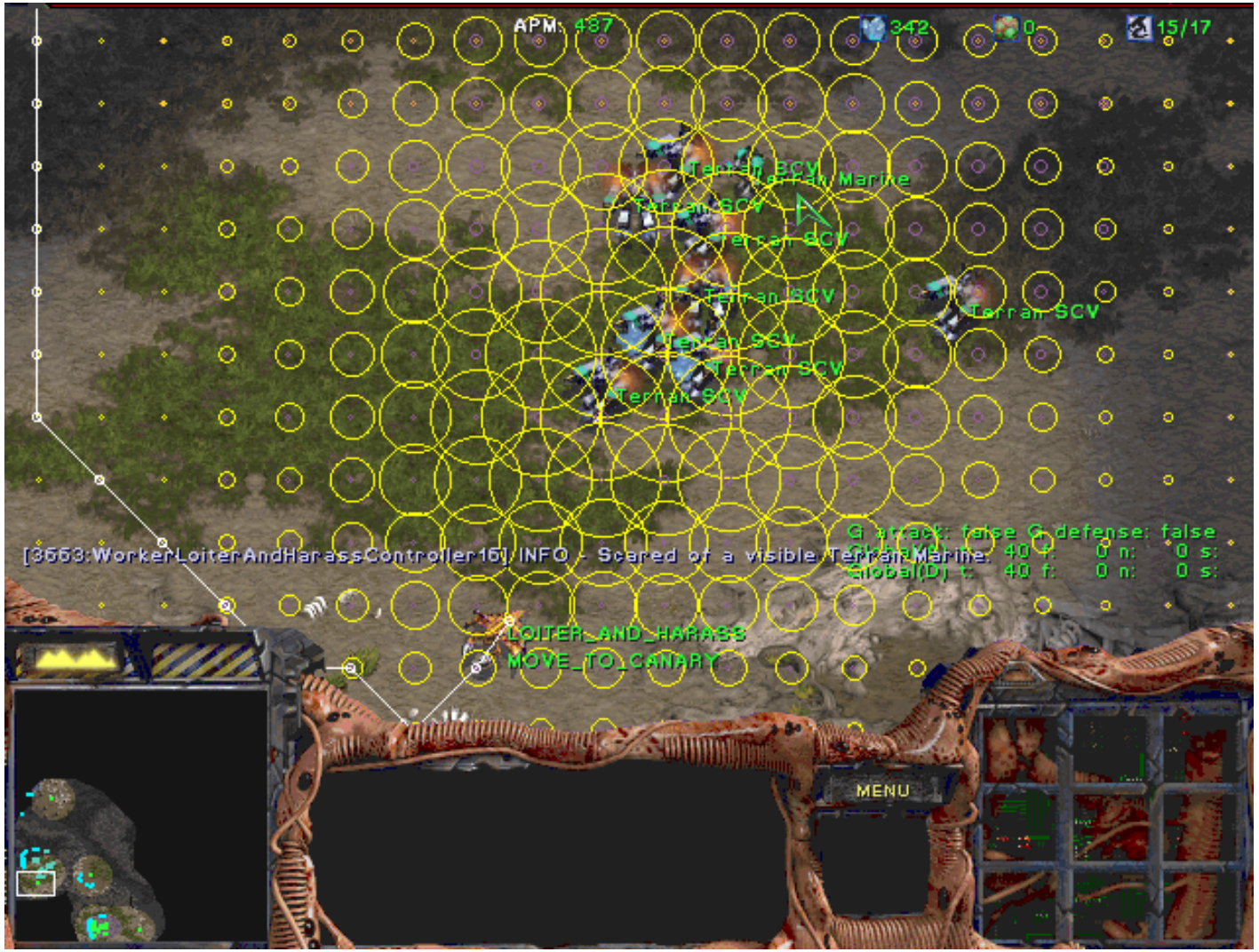
حریصانه

هزینه یکنواخت

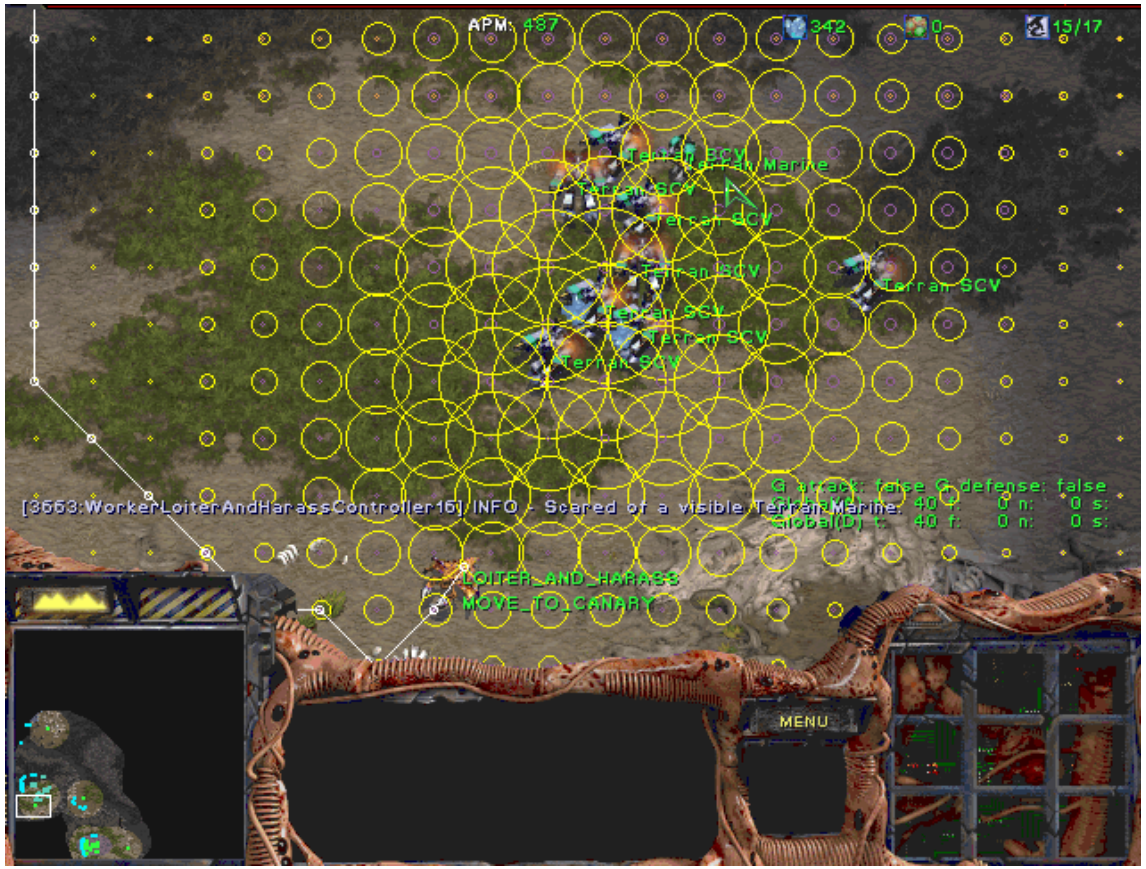


A*

کاربردهای A*

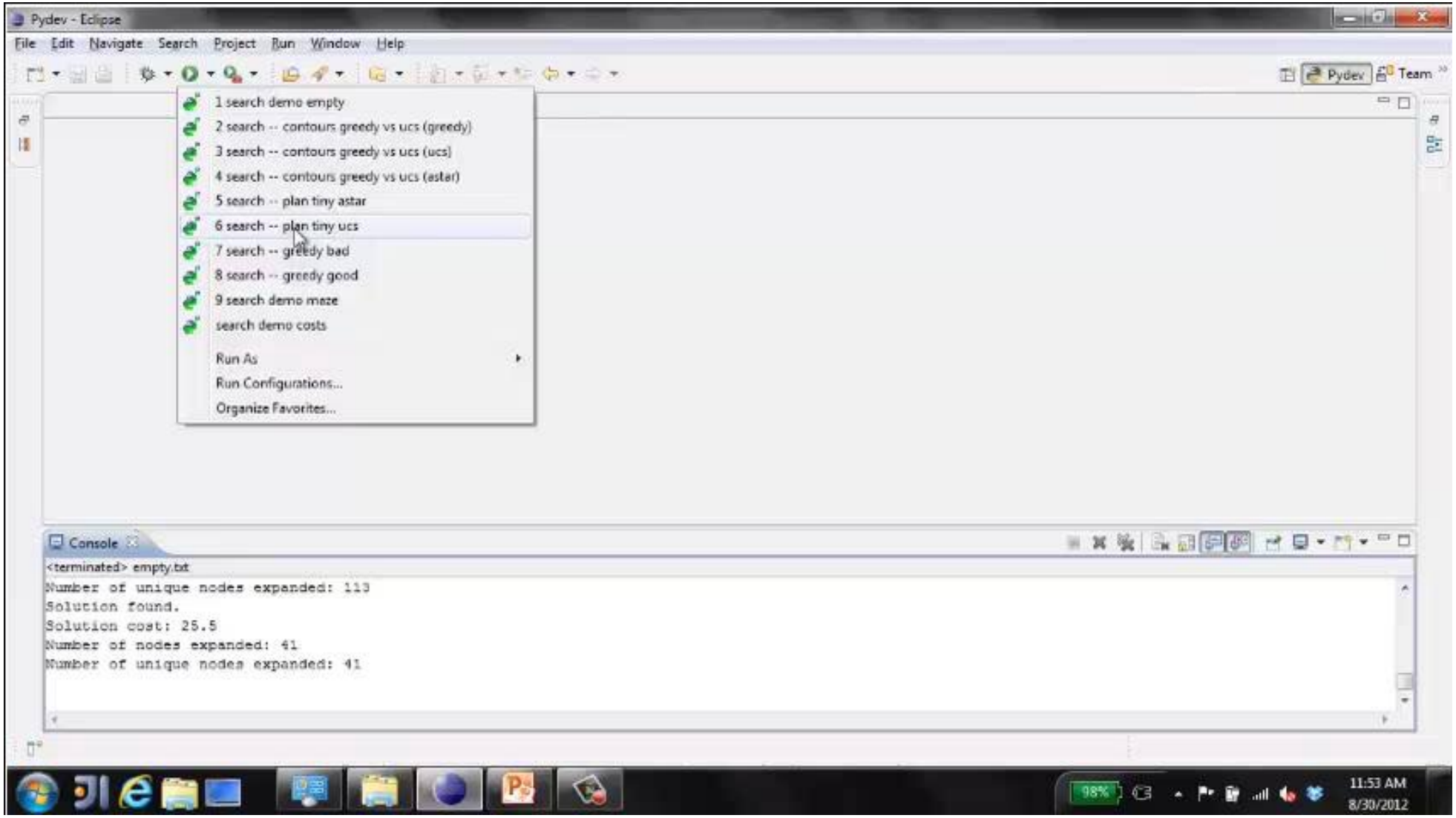

APM: 487
342
0
15/17
Terran SCV
Terran Marine
Terran SCV
Terran SCV
Terran SCV
Terran SCV
Terran SCV
Terran SCV
Terran SCV
[3663:WorkerLoiterAndHarassController16] INFO - Scared of a visible Terran Marine.
G attack: false G defense: false
Global(D) t: 40 f: 0 n: 0 s:
LOITER_AND_HARASS
MOVE_TO_CANARY
MENU

کاربردهای A*

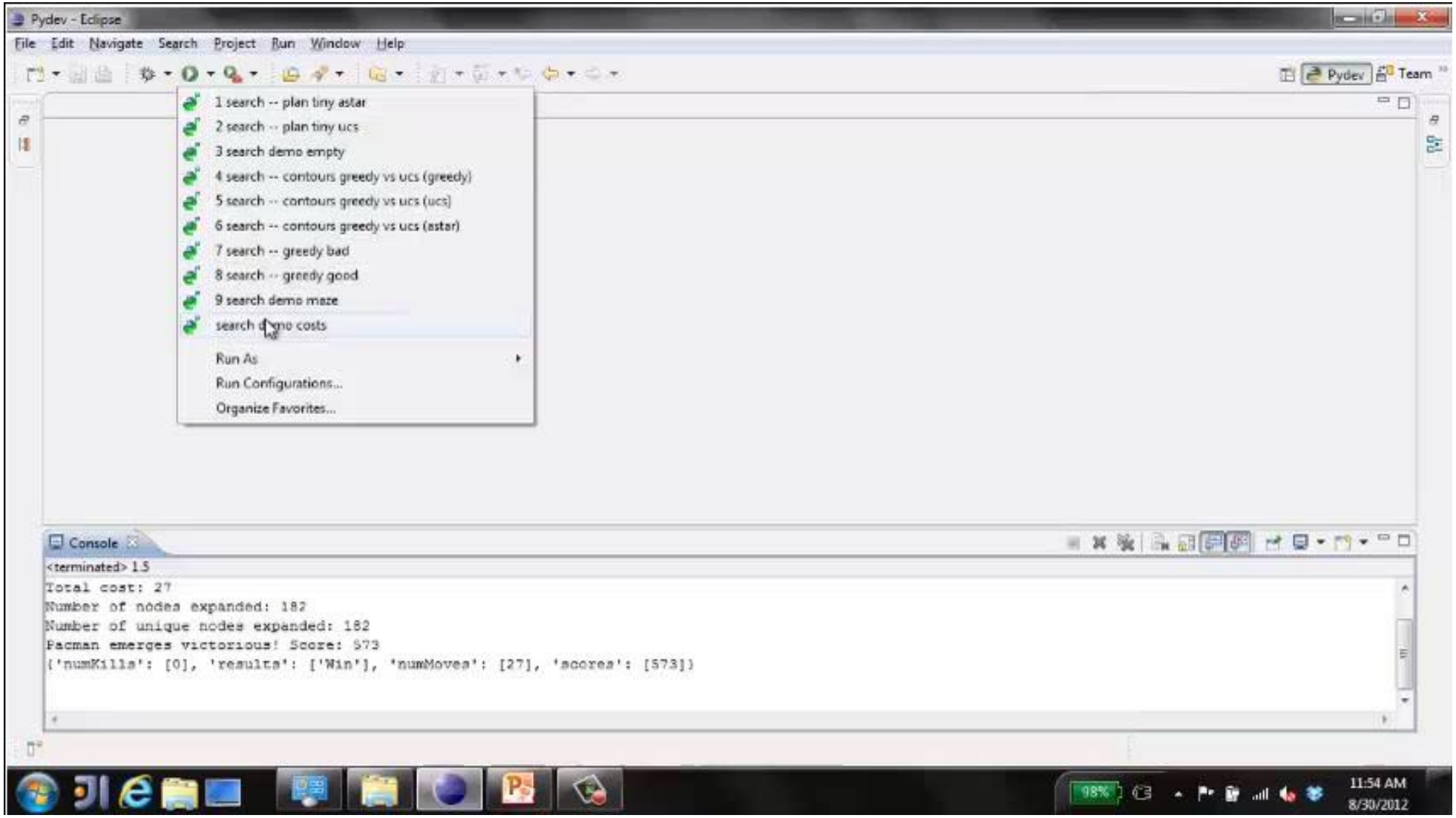


- بازی‌های ویدیویی
- مسائل مسیریابی / مسیرگزینی
- مسائل برنامه ریزی منابع
- برنامه‌ریزی حرکت ربات
- تحلیل زبان
- ...

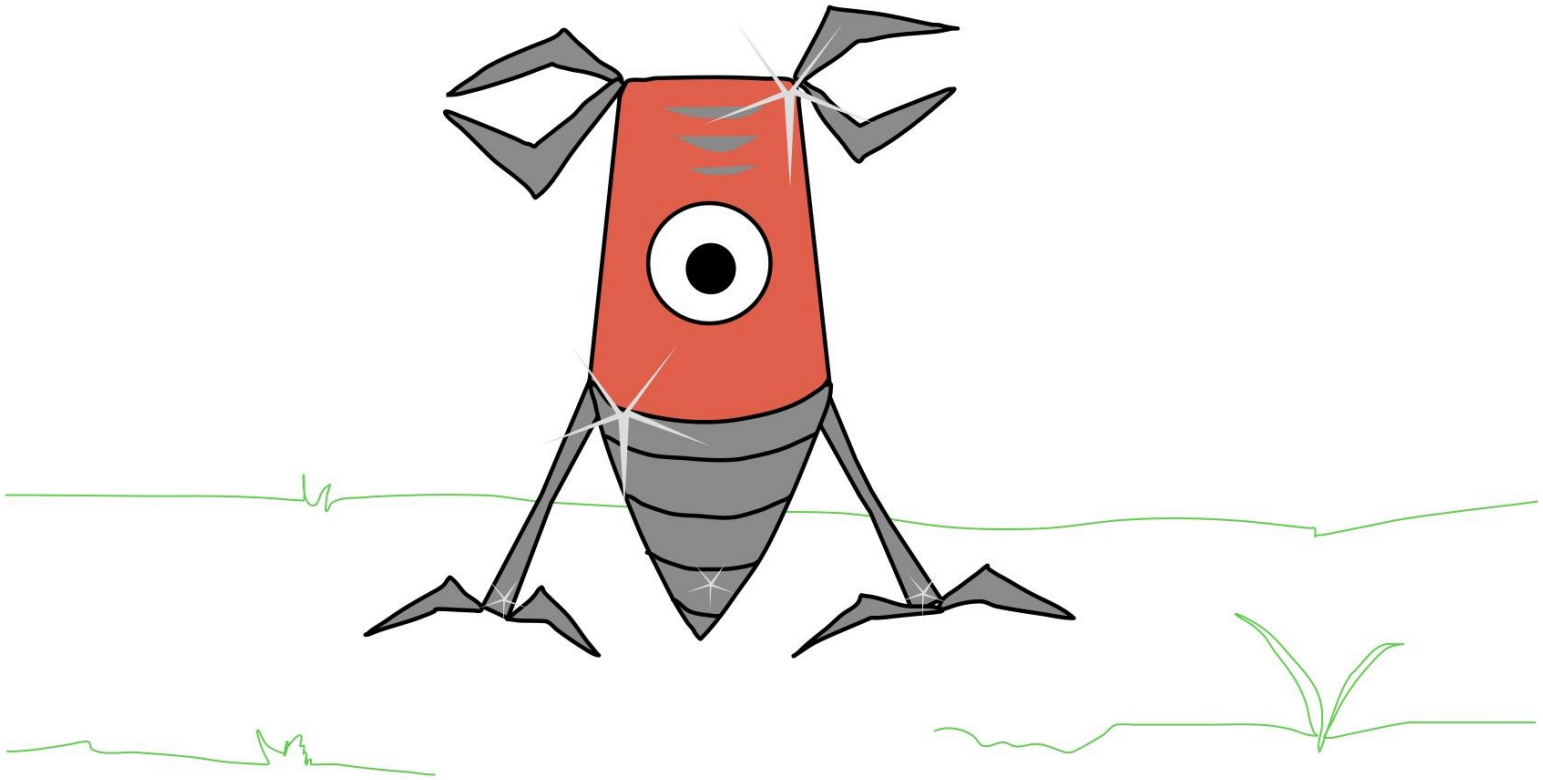
Pacman (Tiny Maze) – A*/UCS ویدیوی دموی



ویدیوی دمو آب خالی کم عمق/عمیق – الگوریتم را حدس بزنید

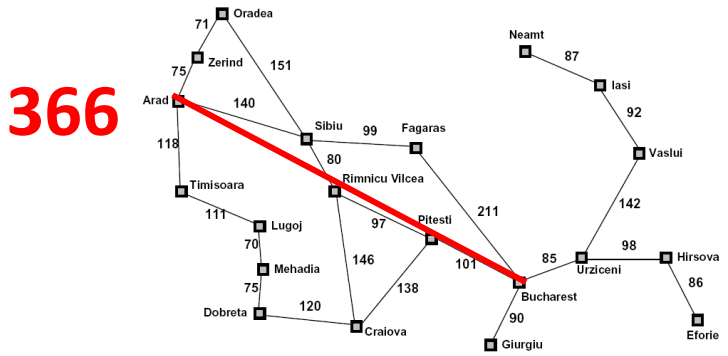


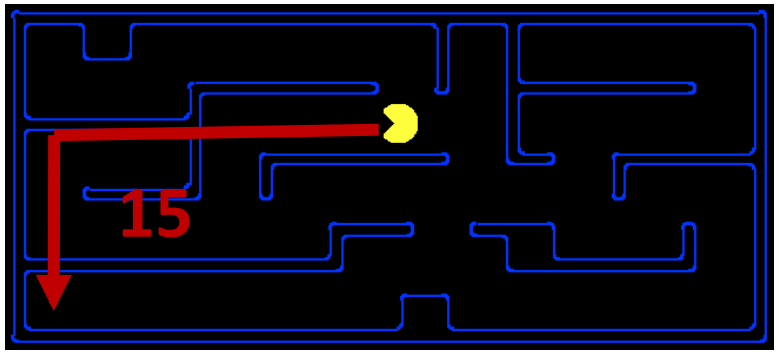
ساخت هیوریستیک‌ها



ساخت هیوریستیک‌های قابل قبول

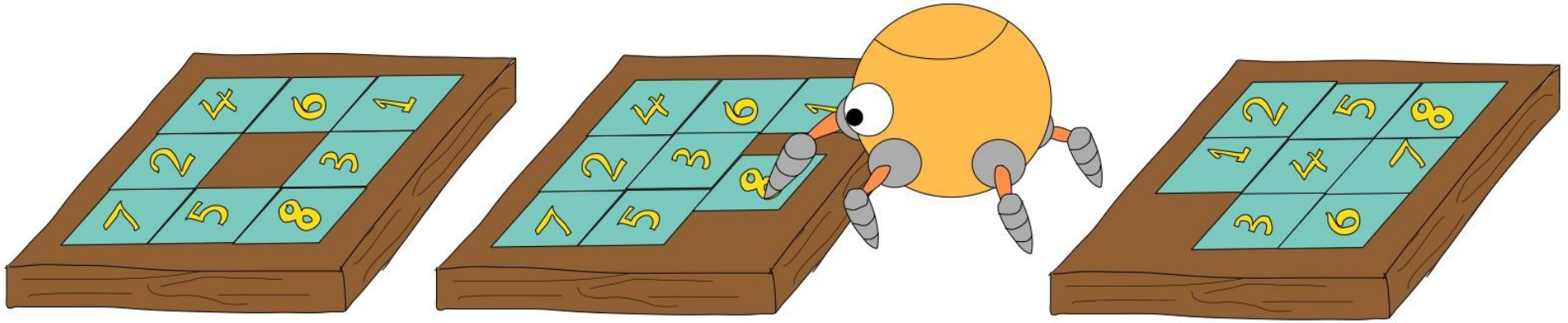
- اصلی‌ترین کار برای حل بهینه مسائل جستجوی سخت، دستیابی به هیوریستیک قابل قبول است
- اغلب وقت‌ها، هیوریستیک‌های قابل قبول راه حلی برای فرم ساده‌شده مسئله هستند، که در آن اعمال جدید در دسترس باشد





- هیوریستیک‌های غیرقابل قبول نیز خیلی اوقات مفید خواهد بود

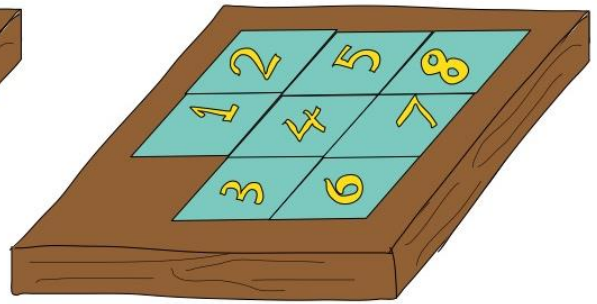
مثال: 8 پازل



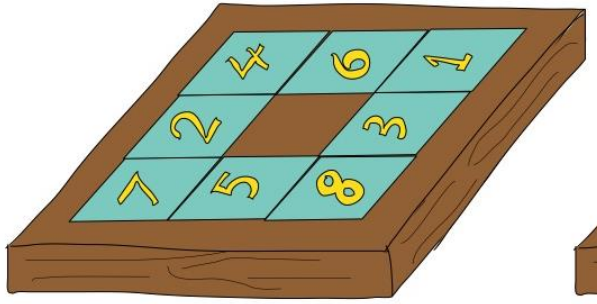
حالت شروع　　اعمال　　حالت هدف

- حالت‌ها چیست؟
- چند حالت وجود دارد؟
- اعمال چیست؟
- از حالت شروع چند حالت پسین وجود دارد؟
- هزینه چه باید باشد؟

۸ پازل - ۱



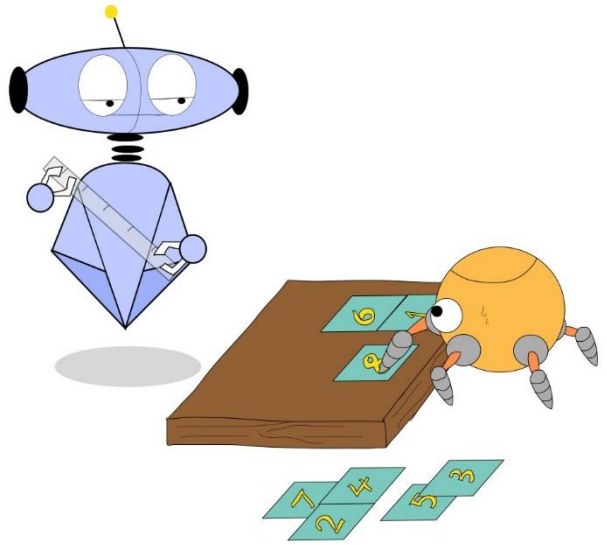
حالت هدف



حالت شروع

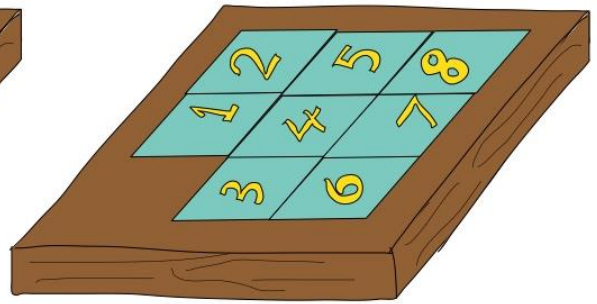
- هیوریستیک: تعداد کاشی‌های نابجا
- چرا قابل قبول است؟
- h(start) = 8
- این یک هیوریستیک برای مساله‌ی relaxed است.

	میانگین گره‌های گسترش داده‌شده زمانی که مسیر بهینه ... باشد:		
	4 مرحله	8 مرحله	12 مرحله
UCS	112	6,300	3.6×10^6
TILES	13	39	227

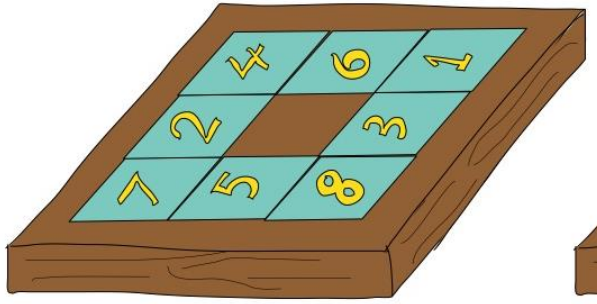


Statistics from Andrew Moore

۸ پازل - ۲



حالت هدف



حالت شروع

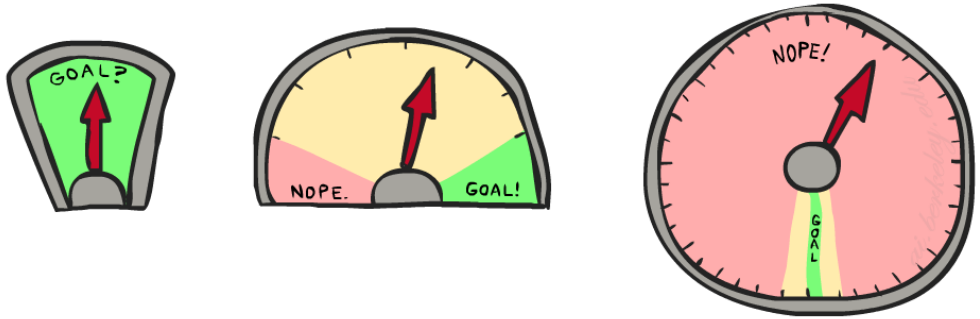
- چه می‌شود اگر یک 8 پازل ساده‌تر داشتیم که در آن هر کاشی می‌توانست در هر زمانی در هر جهتی بلغزد، بدون توجه به کاشی‌های دیگر؟

- مجموع مسافت منهتن
- چرا قابل قبول است؟
- h(start) = 3 + 2 + 1 + … = 18

	میانگین گره‌های گسترش داده‌شده زمانی که مسیر بهینه ... باشد:		
	4 مرحله	8 مرحله	12 مرحله
TILES	13	39	227
MANHATTAN	12	25	73

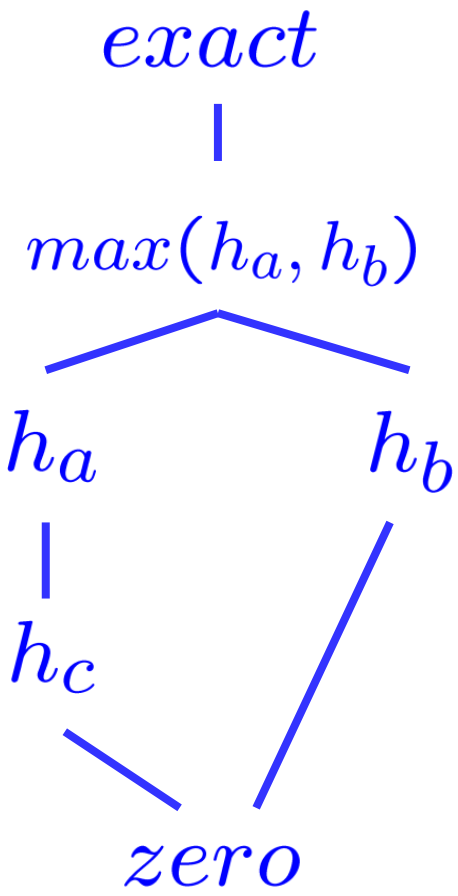
۸ پازل - ۳

- نظر شما در مورد استفاده از هزینه واقعی به عنوان هیوریستیک چیست؟
  - آیا قابل قبول خواهد بود؟
  - آیا در گره‌های گسترش یافته صرفه جویی می‌کنیم؟
  - مشکلش چیست؟



- A*: یک مبادله بین کیفیت برآورد هیوریستیک و کار مورد نیاز در هر گره وجود دارد
  - با نزدیک‌تر شدنِ هیوریستیک به هزینه واقعی، گره‌های کمتری را بسط می‌یابد، اما معمولاً در هر گره کار بیشتری برای محاسبه خود هیوریستیک انجام می‌دهید

هیوریستیک بدیهی، غلبه



- غلبه: ha ≥ hc اگر:

$$\forall n : h_a(n) \geq h_c(n)$$

- هیوریستیک یک نیمه شبکه (semi-lattice) را تشکیل می‌دهد:
  - حداکثر مقدار دو یا چند هیوریستیک قابل قبول، قابل قبول است.

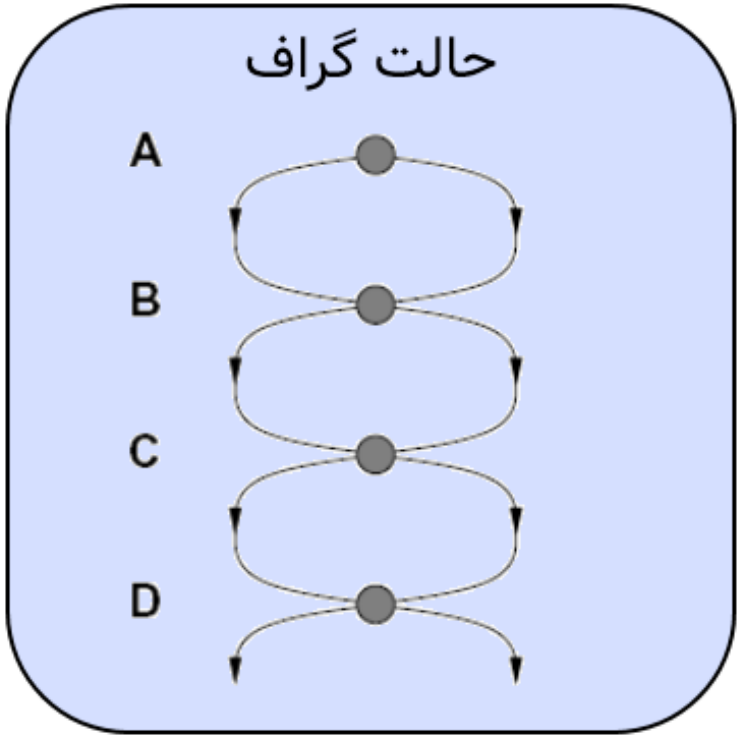
$$h(n) = max(h_a(n), h_b(n))$$

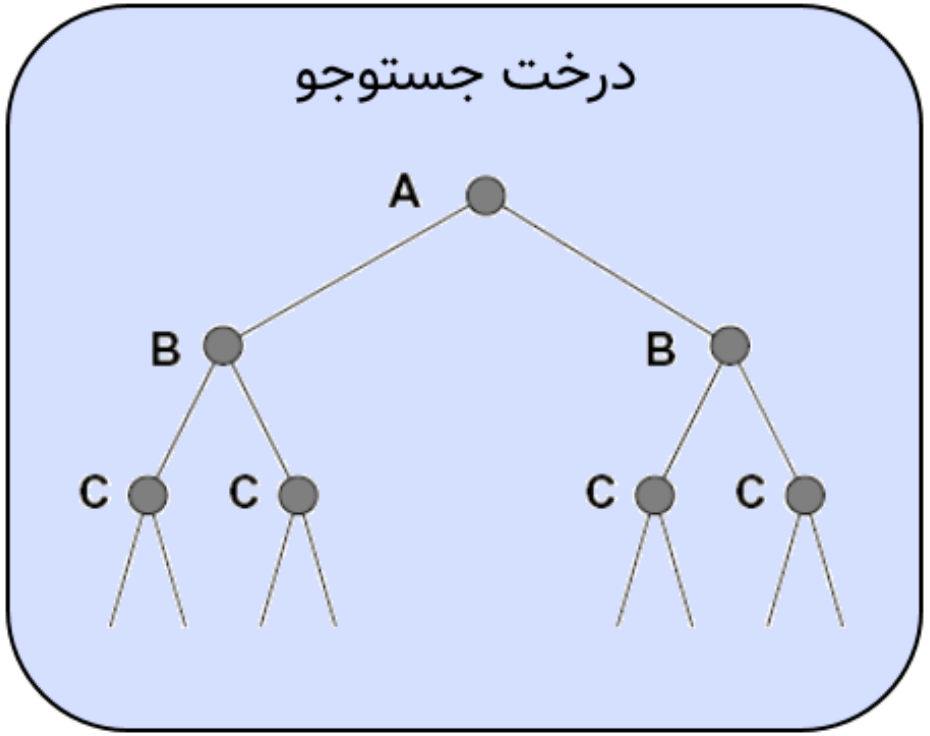
- هیوریستیک‌های بدیهی:
  - پایین شبکه، هیوریستیک صفر است (این به ما چه می دهد؟)
  - بالای شبکه، هیوریستیک دقیق است.

جستوجوی گرافی

جستجوی درختی: کار اضافی!

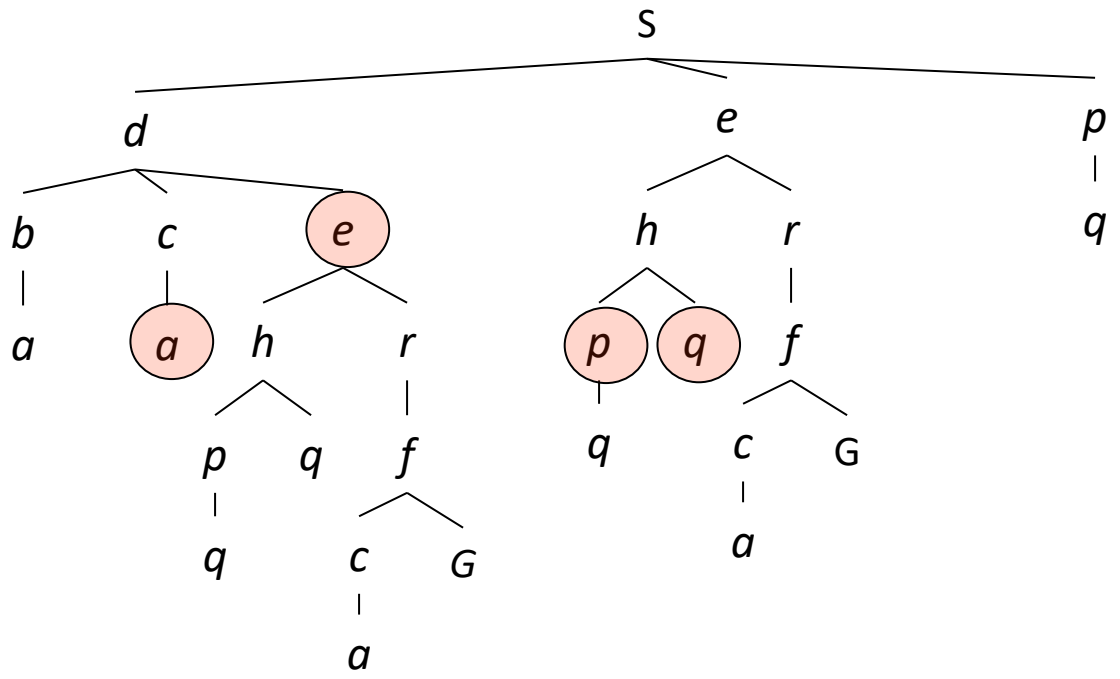
- عدم شناسایی حالت‌های تکرار شده می‌تواند باعث افزایش تصاعدی کار شود.





جستجوی گرافی

- به عنوان مثال، در BFS، ما لازم نیست خودمان را با گسترش گره‌های حلقه اذیت کنیم (چرا؟)



جستجوی گرافی

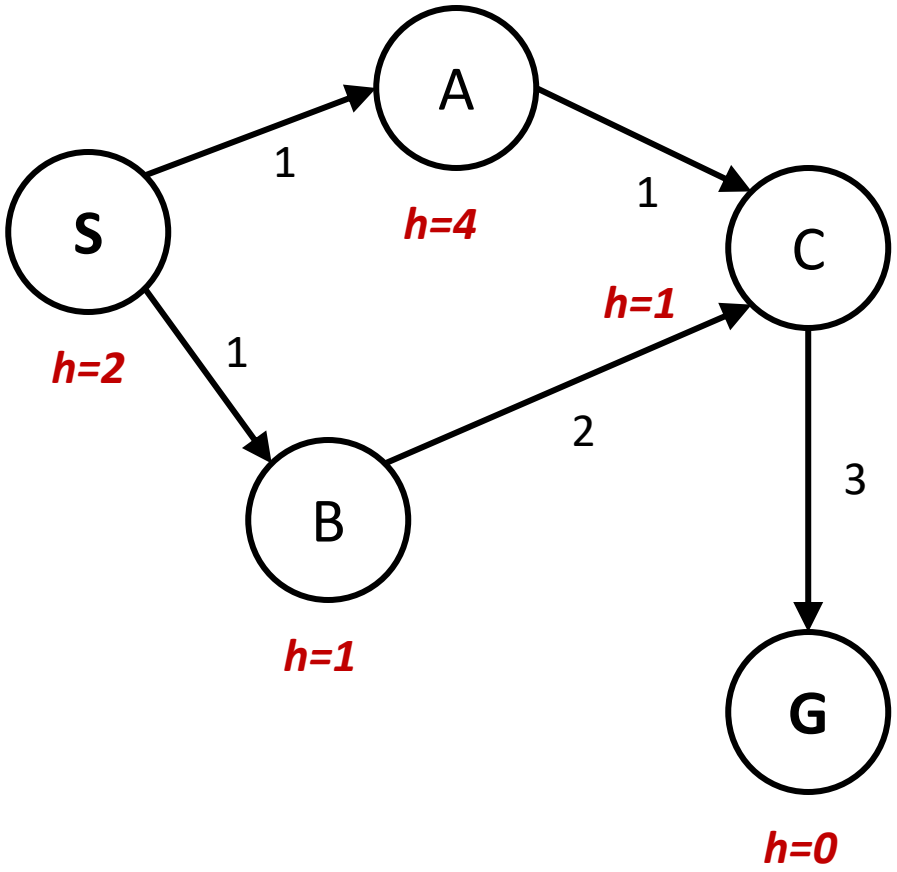
- ایده: هرگز یک حالت را دوبار گسترش (expand) ندهید.
- نحوه پیاده سازی:
  - جستجوی درختی + مجموعه ای از حالت‌های گسترش یافته ("مجموعه بسته")
  - درخت جستجو را گره به گره گسترش دهید، اما...
  - قبل از گسترش یک گره، بررسی کنید که حالت آن قبلاً هرگز گسترش نیافته باشد
  - اگر جدید نیست، آن را رد کنید، اگر جدید است به مجموعه بسته اضافه کنید
- مهم: مجموعه بسته را به عنوان یک مجموعه (set) ذخیره کنید، نه یک لیست (list)

- آیا جستجوی گرافی می‌تواند کامل بودن را از بین ببرد؟ چرا؟/ چرا نه؟

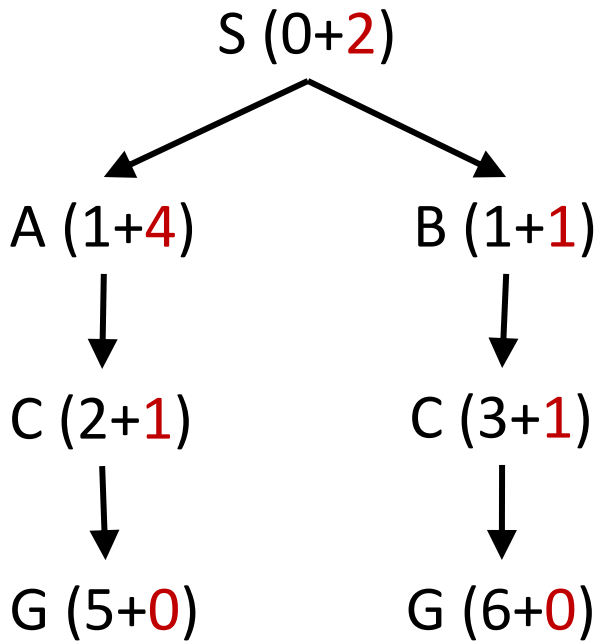
- بهینه بودن چطور؟

جستجوی گرافی A* اشتباه است؟

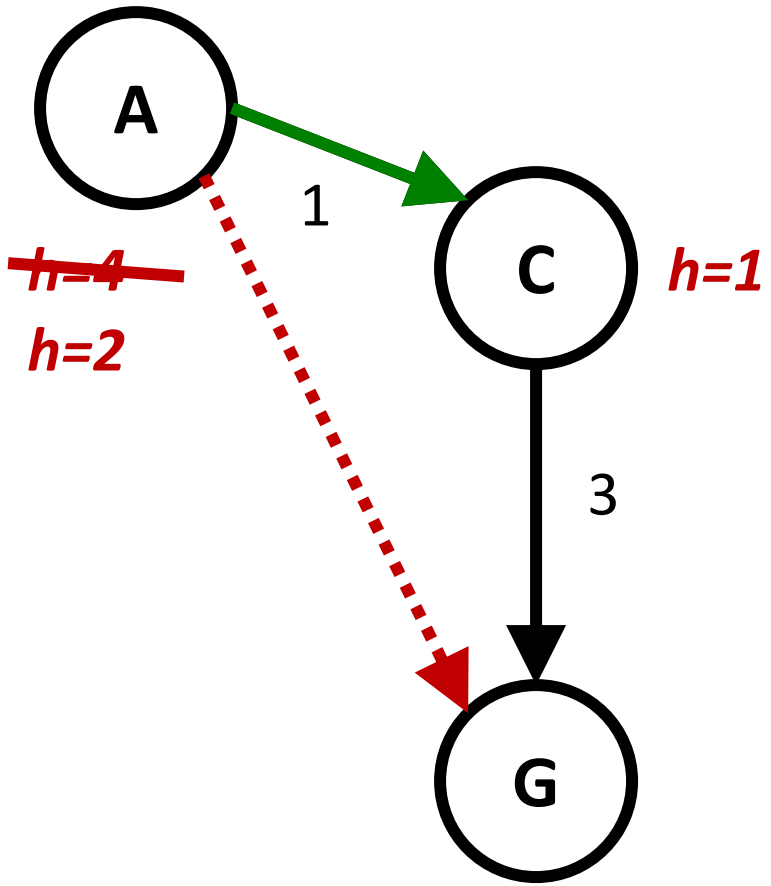
گراف فضای حالت



درخت جستجو



سازگاری (consistency) هیوریستیک



- ایده اصلی: هزینه‌های تخمینی هیوریستیک ≤ هزینه‌های واقعی
  - قابل قبول بودن: هزینه هیوریستیک ≤ هزینه واقعی برای هدف

    $h(A) \leq$ هزینه واقعی از A به C
  - سازگاری: هزینه "یال‌ها" هیوریستیک ≤ هزینه واقعی برای هر یال

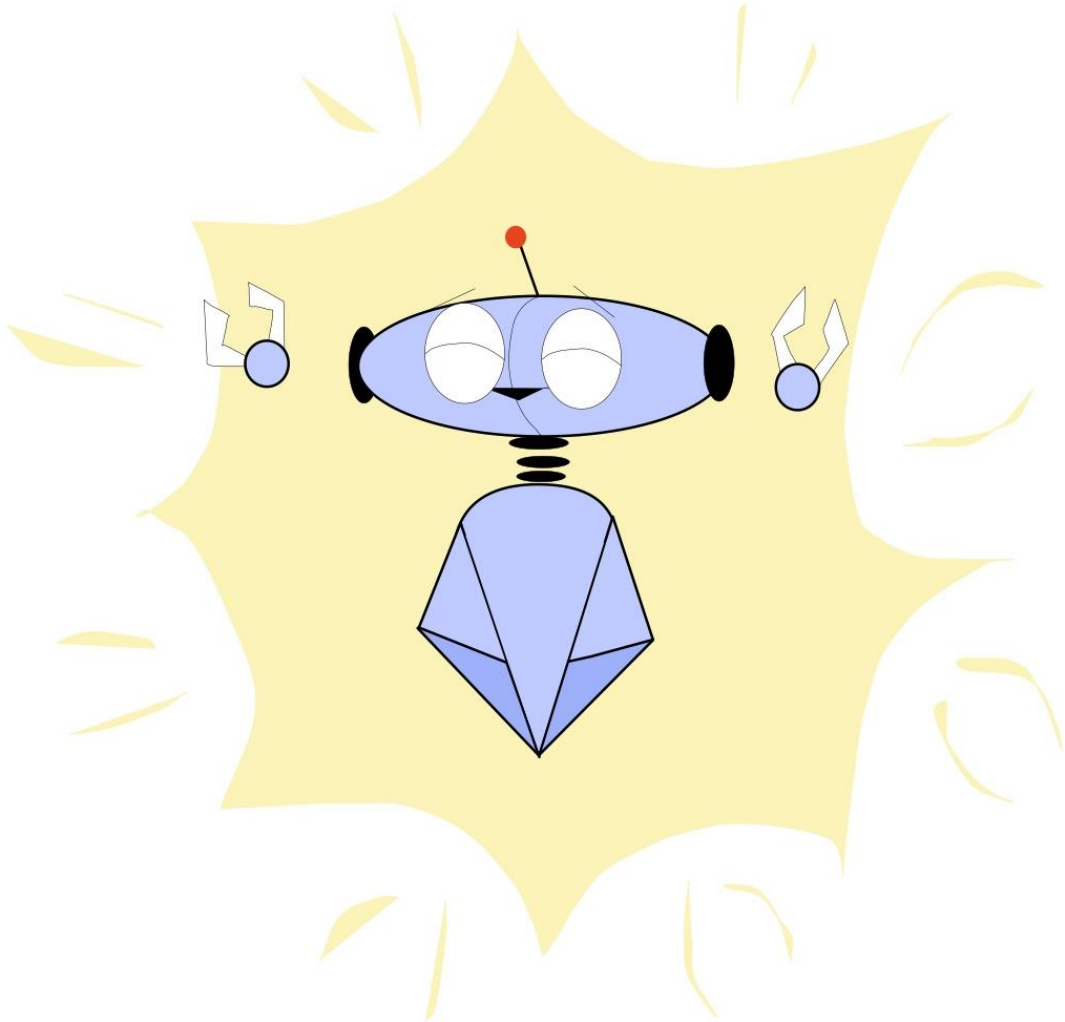
    $h(A) - h(C) \leq cost(A \text{ to } C)$

- پیامدهای سازگاری:
  - مقدار f در طول مسیر هرگز کاهش نمی یابد

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

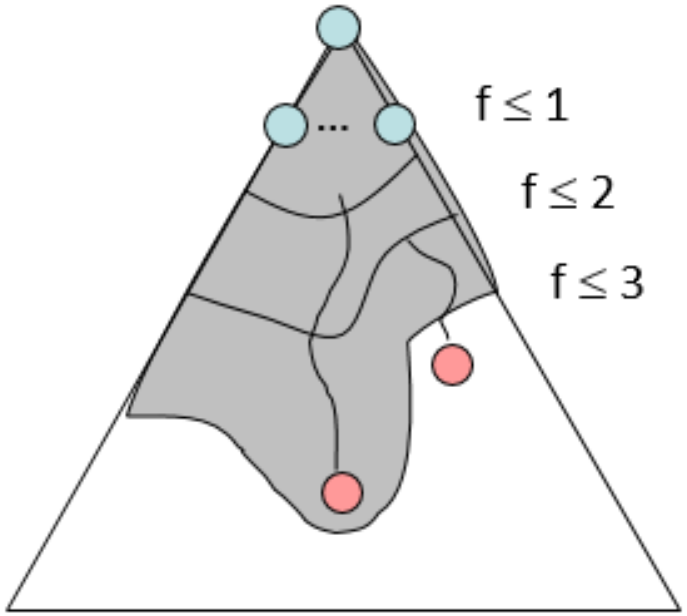
- جستجوی گرافی A* بهینه است

بهینه بودن جستجوی گرافی A*

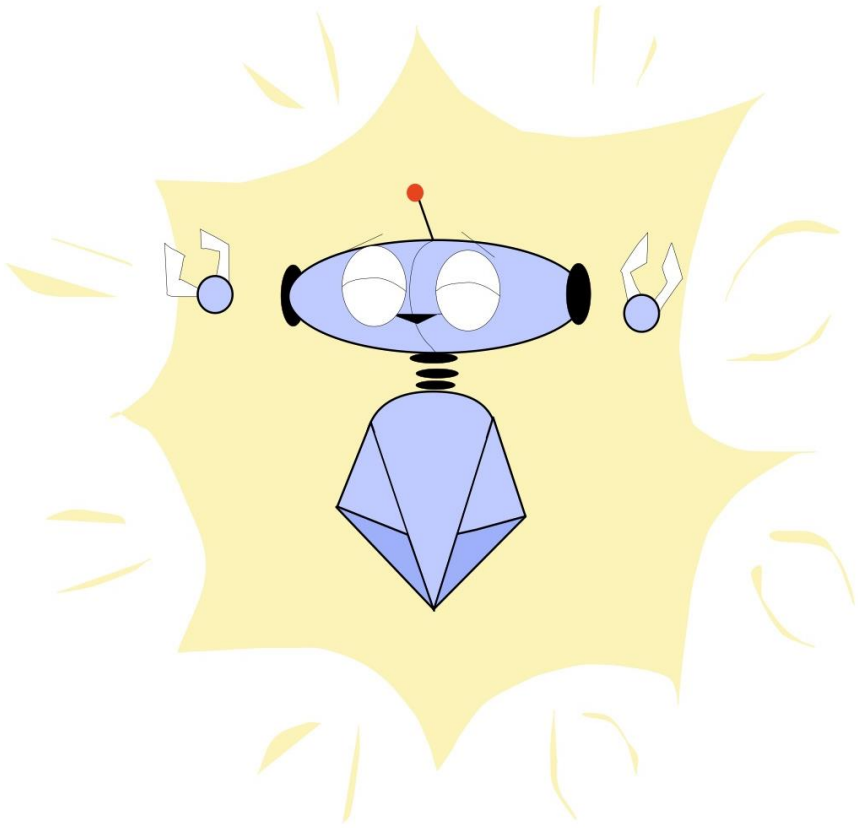


بهینه بودن جستجوی گرافی A*

- طراحی: در نظر بگیرید که A* با یک هیوریستیک سازگار چه می‌کند:
  - واقعیت 1: در جستجوی درختی، A* گره‌ها را با افزایش مقدار f یا (f-contours) گسترش خواهد داد
  - واقعیت 2: برای هر حالت s، گره‌هایی که به s بصورت بهینه می‌رسند، قبل از گره‌هایی که به s بصورت غیربهنیه می‌رسد گسترش می‌یابند
  - نتیجه: جستجوی گرافی A* بهینه است



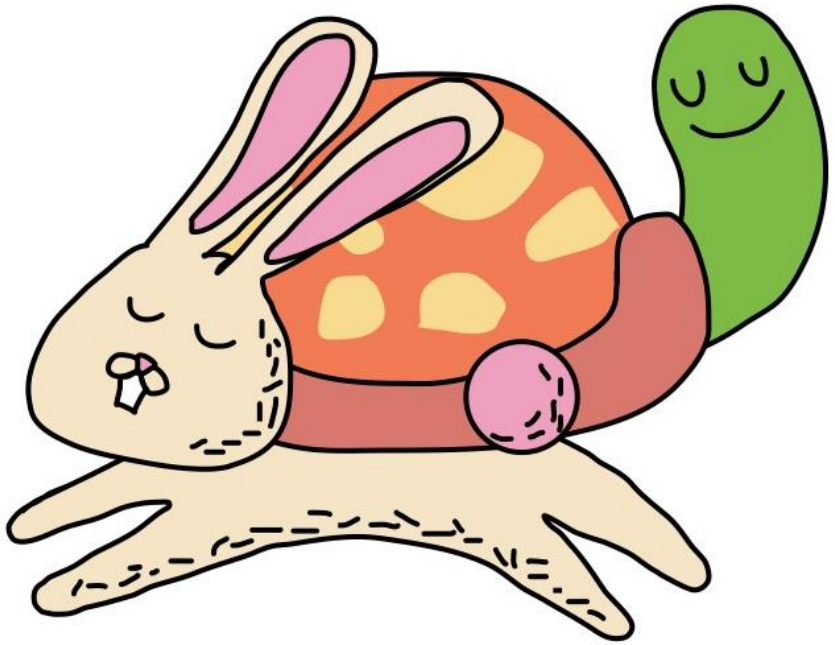
بهینه بودن



- جستجوی درختی
  - اگر هیوریستیک قابل قبول باشد A* بهینه است
  - UCS یک مورد خاص از A* است (h = 0)
- جستجوی گرافی
  - اگر هیوریستیک سازگار باشد A* بهینه است
  - UCS بهینه است (h = 0 سازگار است)
- سازگاری دلالت بر قابل قبول بودن دارد
- به طور کلی، بیشتر هیوریستیک‌های قابل قبول طبیعی تمایل به

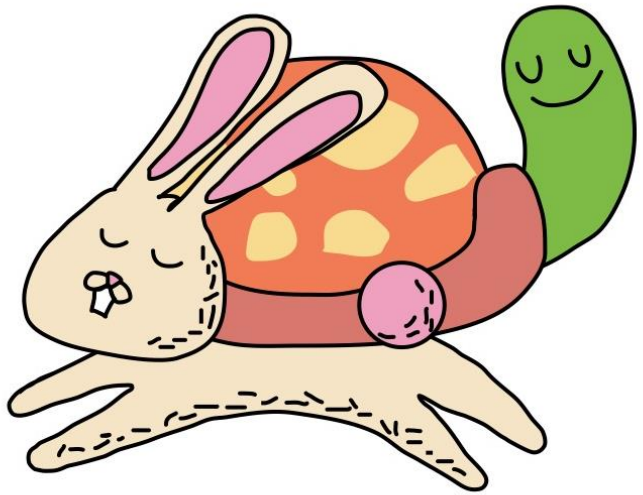
سازگاری دارند، به خصوص اگر ناشی از مسائل ساده‌سازی شده باشند

A*: خلاصه



A*: خلاصه

- A* هم از هزینه‌های پیشین و هم (برآورد) هزینه‌های آتی استفاده می‌کند
- A* با هیوریستیک قابل قبول/سازگار بهینه است
- کلید کار طراحی هیوریستیک است: می‌توانید از مسائل ساده‌سازی شده استفاده کنید



شبه‌کد جستجوی درختی

```
function TREE-SEARCH(problem, fringe) return a solution, or failure
    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
        for child-node in EXPAND(STATE[node], problem) do
            fringe ← INSERT(child-node, fringe)
        end
    end
```

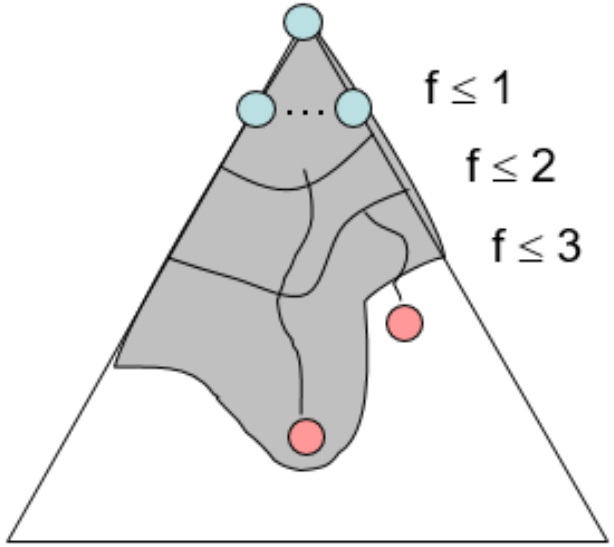
شبه‌کد جستجوی گرافی

```
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
```

بهینه بودن جستجوی گرافی A*

- آنچه را که A* انجام می‌دهد در نظر بگیرید:
  - گره‌ها را با افزایش مقدار f کل گسترش می‌دهد
  - یادآوری: $f(n) = g(n) + h(n) = \text{cost to } n + \text{heuristic}$
  - ایده اثبات: هدف(های) بهینه کمترین مقدار f را دارند، بنابراین ابتدا باید بسط داده شود

این استدلال اشکالی دارد.
آیا آن چیزی که فرض کرده ایم صحیح است؟



بهینه بودن جستجوی گرافی A*

اثبات:

- مشکل احتمالی جدید: یک n ای در مسیر G* زمانی که به آن نیاز داریم در صف قرار ندارد، زیرا یک n’ بدتر برای همان حالت پیشتر از صف خارج شده و گسترش یافته است. (فاجعه!)
- بالاترین چنین n ای را در درخت در نظر بگیرید
- فرض کنید p از اجداد n باشد که هنگامی که n’ از صف خارج شد، در صف قرار داشته
- f(p) <f(n) به دلیل سازگاری
- f(n) < f(n’) زیرا n’ غیربهینه است
- p باید قبل از n’ گسترش یافته باشد
- تناقض!