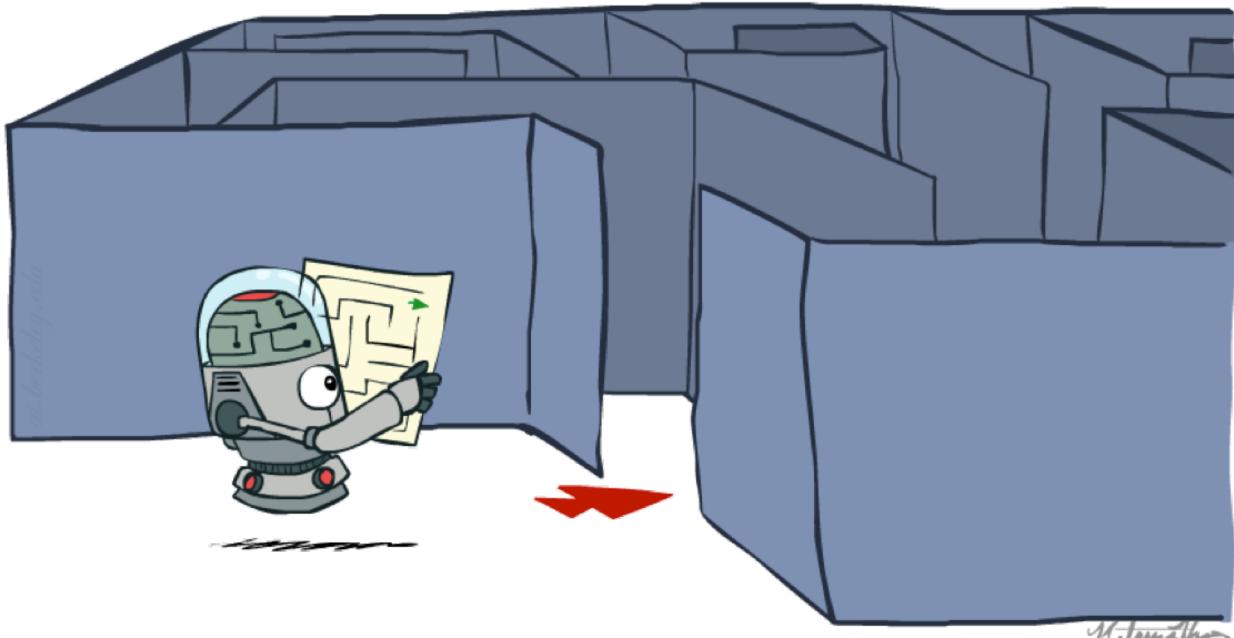




# هوش مصنوعی

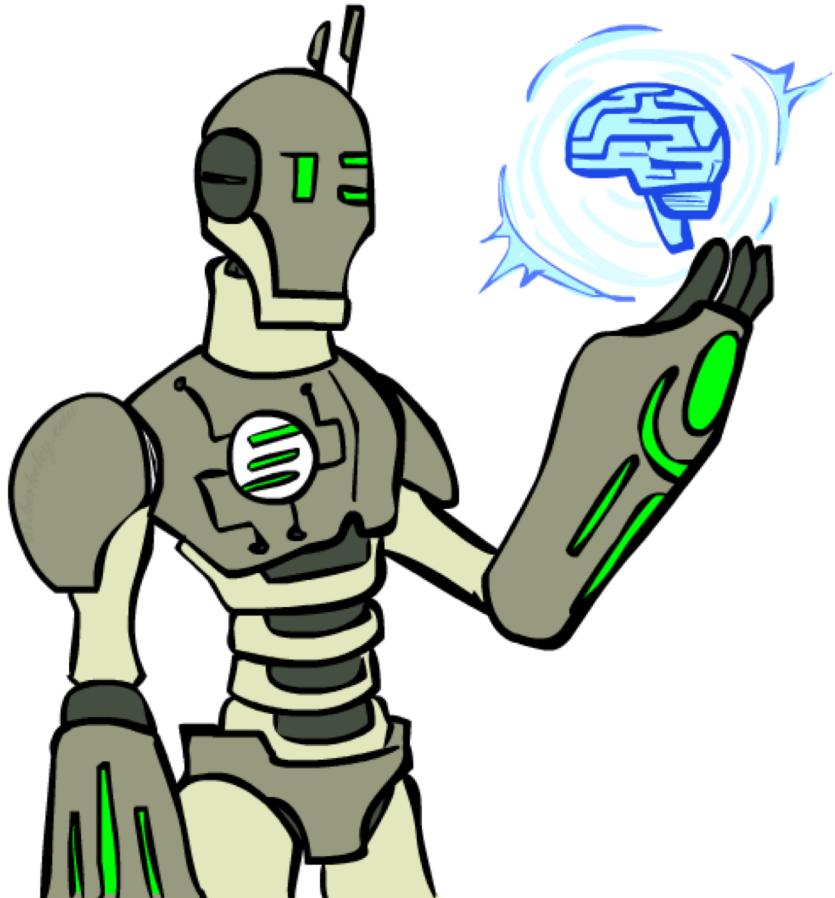
## جستجو



محمد طاهر پیلهور

[These slides were borrowed from CS188 Intro to AI at UC Berkeley.]

# جلسه‌ی قبل



هوش مصنوعی چیست؟

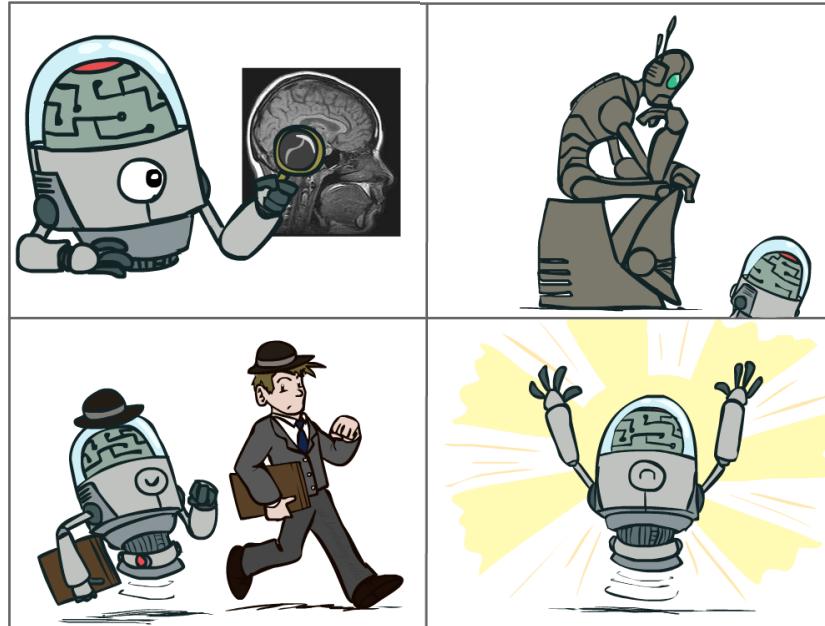
چه کارهایی می تواند بکند؟

وضعیت موجود، پیشرفت‌های اخیر

عامل هوشمند؟!

# جلسه‌ی قبل

مثل آدم‌ها فکر  
می‌کنند



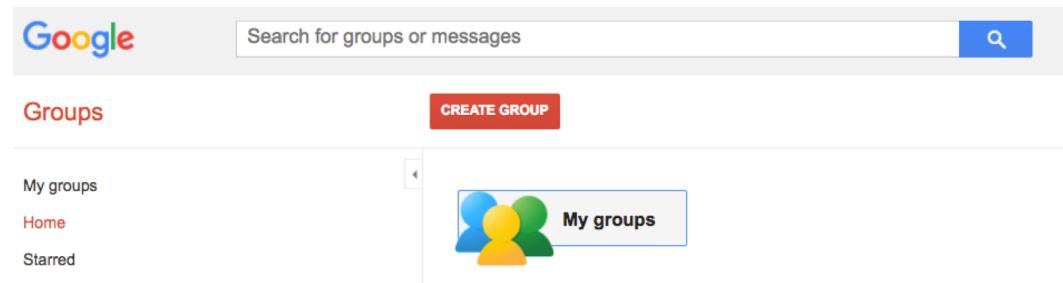
عقلانی فکر می‌کنند

عقلانی عمل می‌کنند

# 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



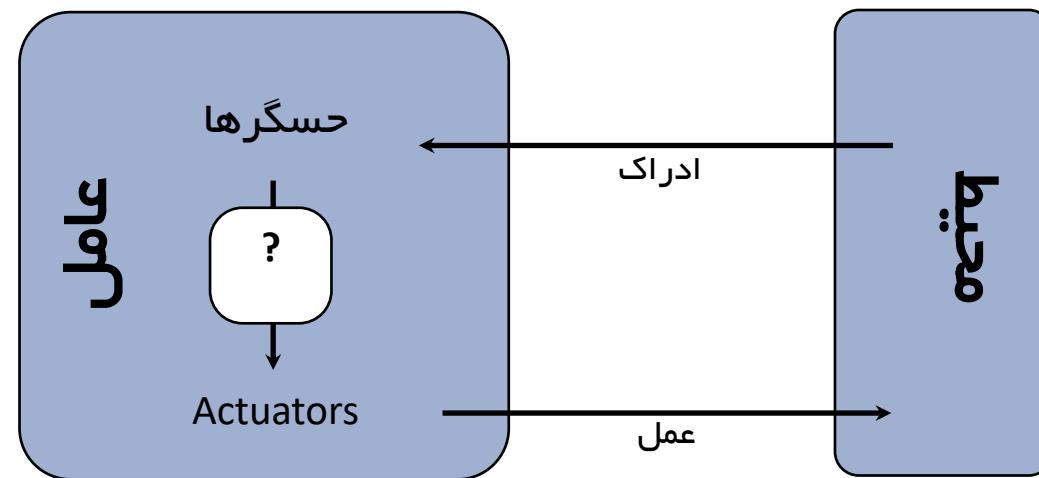
Mostafa  
MohammadAli  
Ebrahim



Kiarash  
Aghakasiri

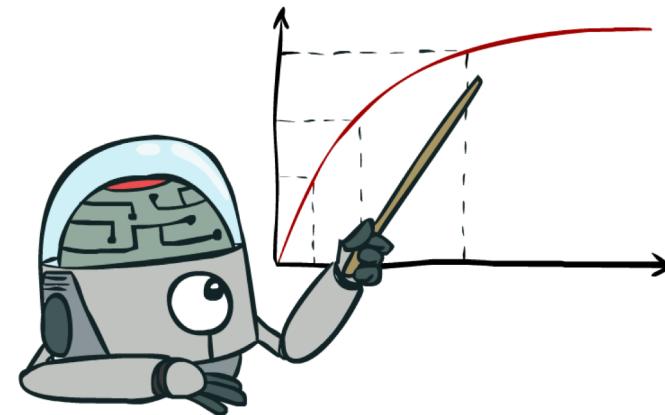
# جلسه‌ی قبل

## عامل و محیط

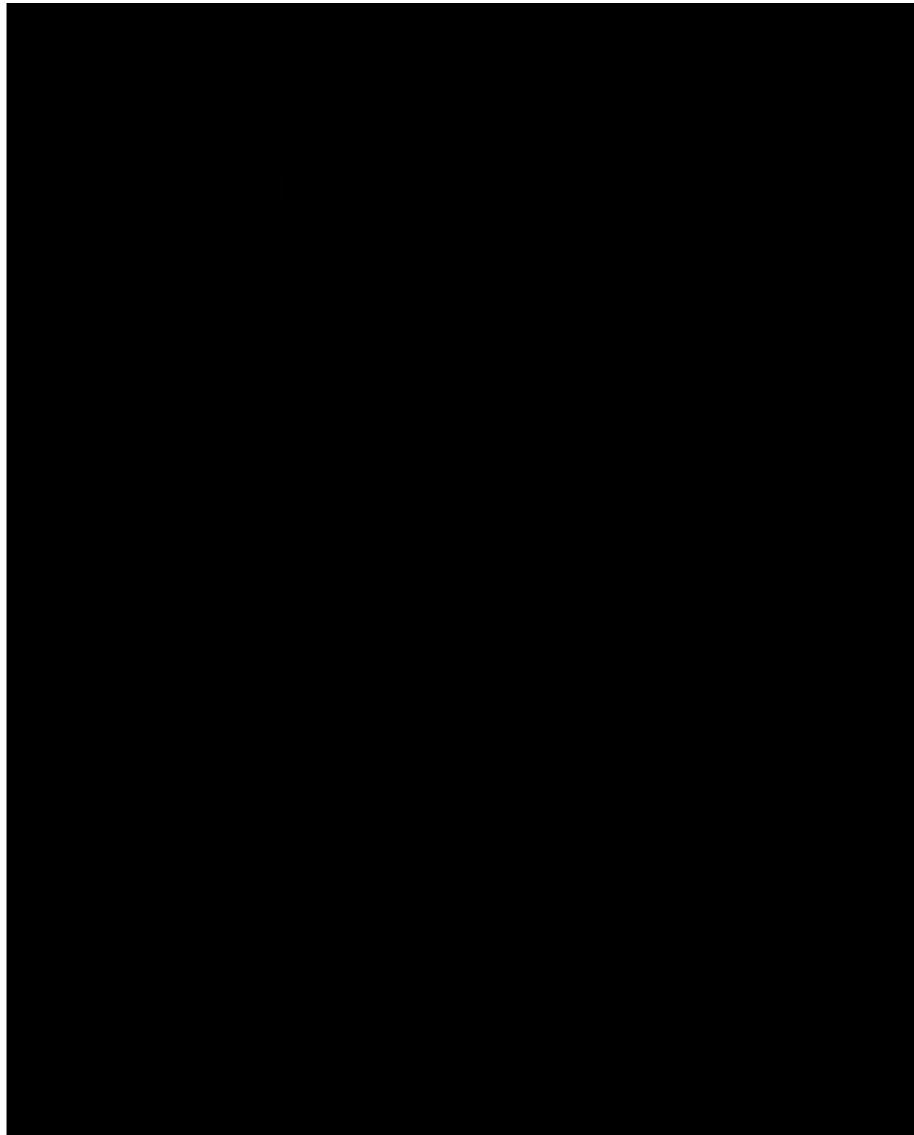


# جلسه‌ی قبل

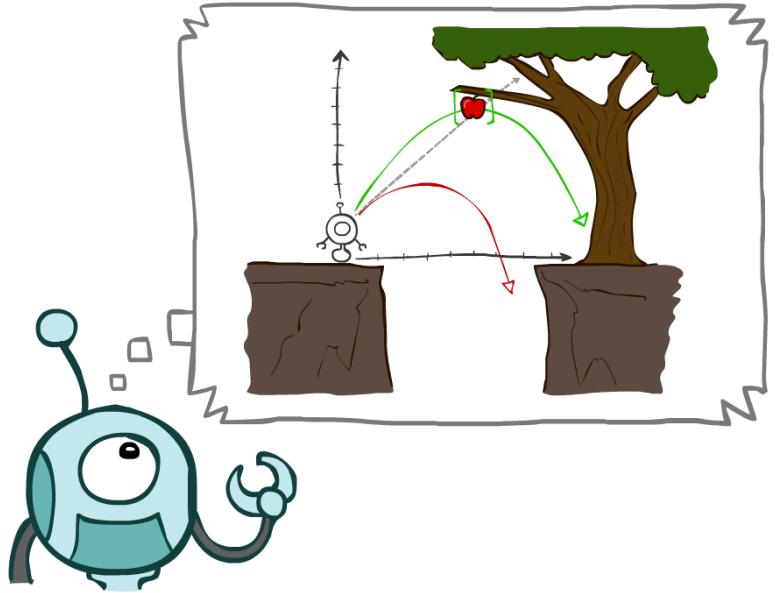
## بیشینه کردن سودمندی مورد انتظار



# دېپمايند - آناري



# امروز



عامل‌هایی که نقشه (طرح) می‌کشند

مسائل جستجو

چطور هوشمندی را در قالب یک مساله جستجو پیاده کنیم؟

روش‌های جستجوی ناگاهانه – Uninformed

جستجوی اول-عمق

جستجوی اول-سطح

جستجوی با هزینه یکسان

# بازی پکمن

Google pacman

All Images Videos Maps Shopping More Settings Tools

About 44,000,000 results (0.49 seconds)

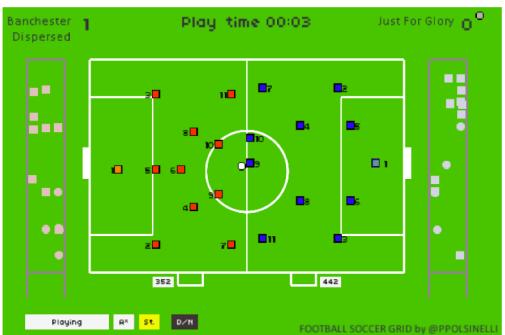
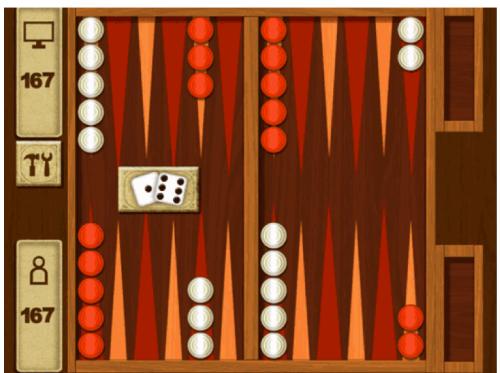
Play PAC-MAN Doodle  
Google home page, 21 May 2010

PAC-MAN™ & ©1980 BANDAI NAMCO Entertainment Inc.

Feedback

# انواع محیط

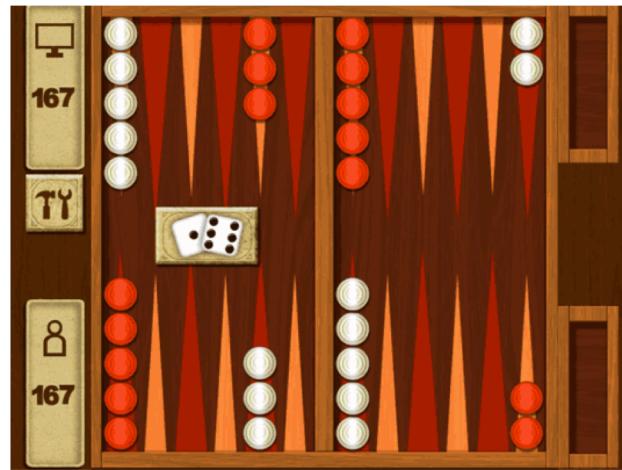
قابل مشاهده کامل یا قابل مشاهده جزئی  
آیا سنسورهای عامل درک کاملی از کل محیط می دهند یا خیر؟



# انواع محیط

## تصادفی یا قطعی

آیا حالت بعدی به طور کامل از حالت فعلی و تصمیم عامل قابل پیش‌بینی است؟



5	3		7			
6			1	9	5	
	9	8				6
8			6			3
4			8	3		1
7			2			6
	6			2	8	
		4	1	9		5
			8		7	9

# انواع محیط

ایستا یا دینامیک

5	3		7		
6		1	9	5	
	9	8			6
8			6		3
4		8	3		1
7			2		6
6			2	8	
	4	1	9		5
		8		7	9



آیا محیط فراتر از کنترل عامل تغییر می کند؟

# انواع محیط

پیوسته یا گستته



5	3		7			
6			1	9	5	
	9	8				6
8			6			3
4		8	3			1
7			2			6
	6			2	8	
		4	1	9		5
			8		7	9



# انواع محیط

تک عاملی یا چند عاملی



5	3		7			
6			1	9	5	
	9	8				6
8			6			3
4		8	3			1
7			2			6
	6			2	8	
		4	1	9		5
			8		7	9

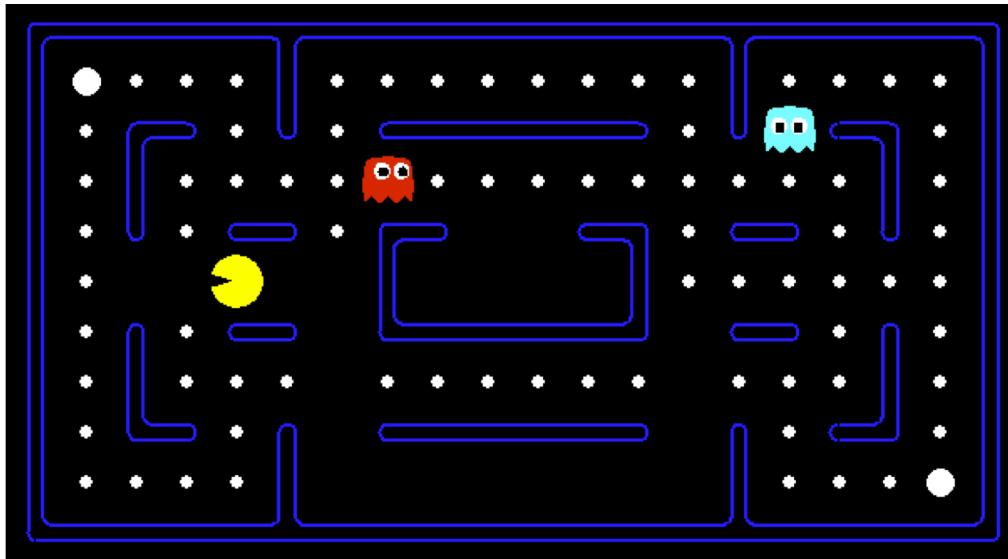


# عامل‌هایی که نقشه می‌کشند

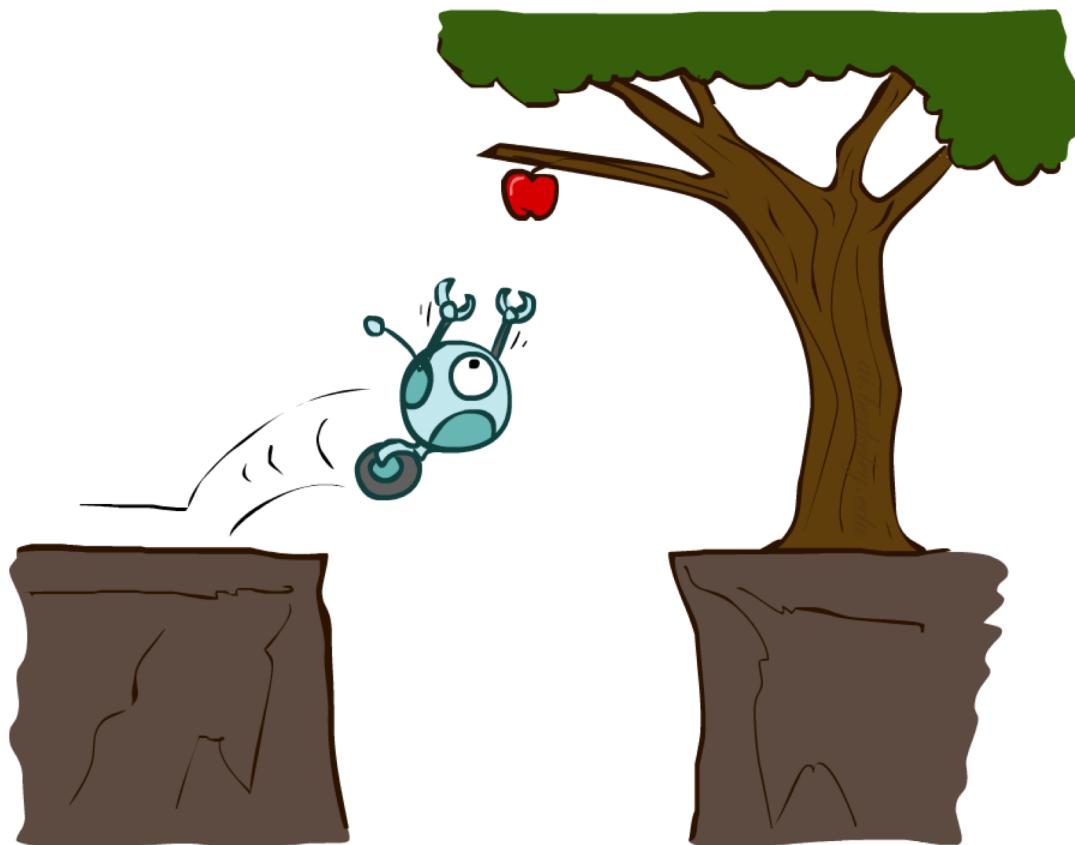
عامل هوشمند باید بتواند **درک** کند! (محیط را ببیند)

- تعداد حالات ممکن محیط نامتناهی است

- با تعریف قوانین if-then کار به جای نمیریم!



# عامل‌های واکنشی – Reflex



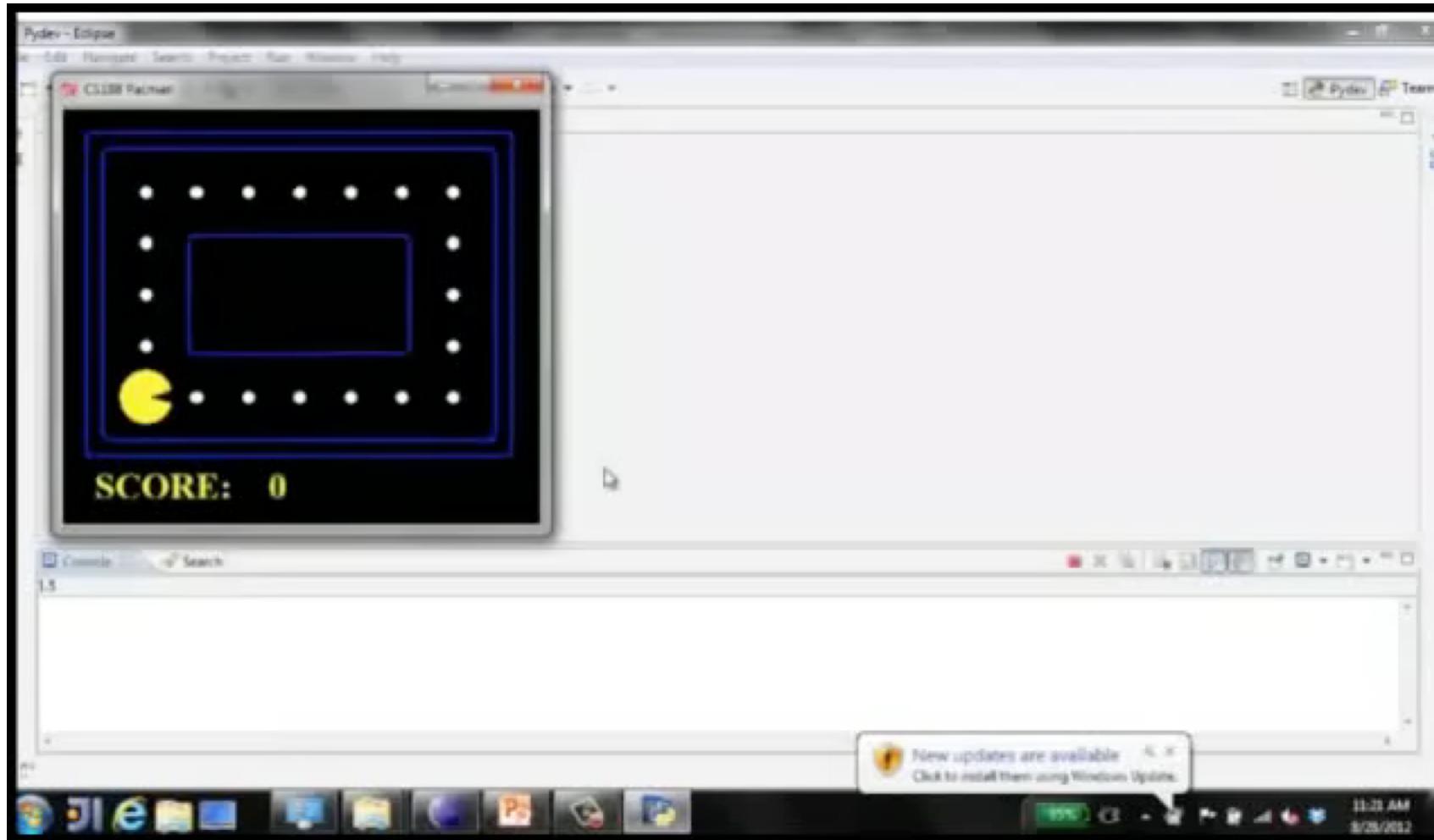
با توجه به درک فعلی (از محیط) تصمیم‌گیری می‌کنند

پیامدهای تصمیمی که می‌گیرند را در نظر نمی‌گیرند

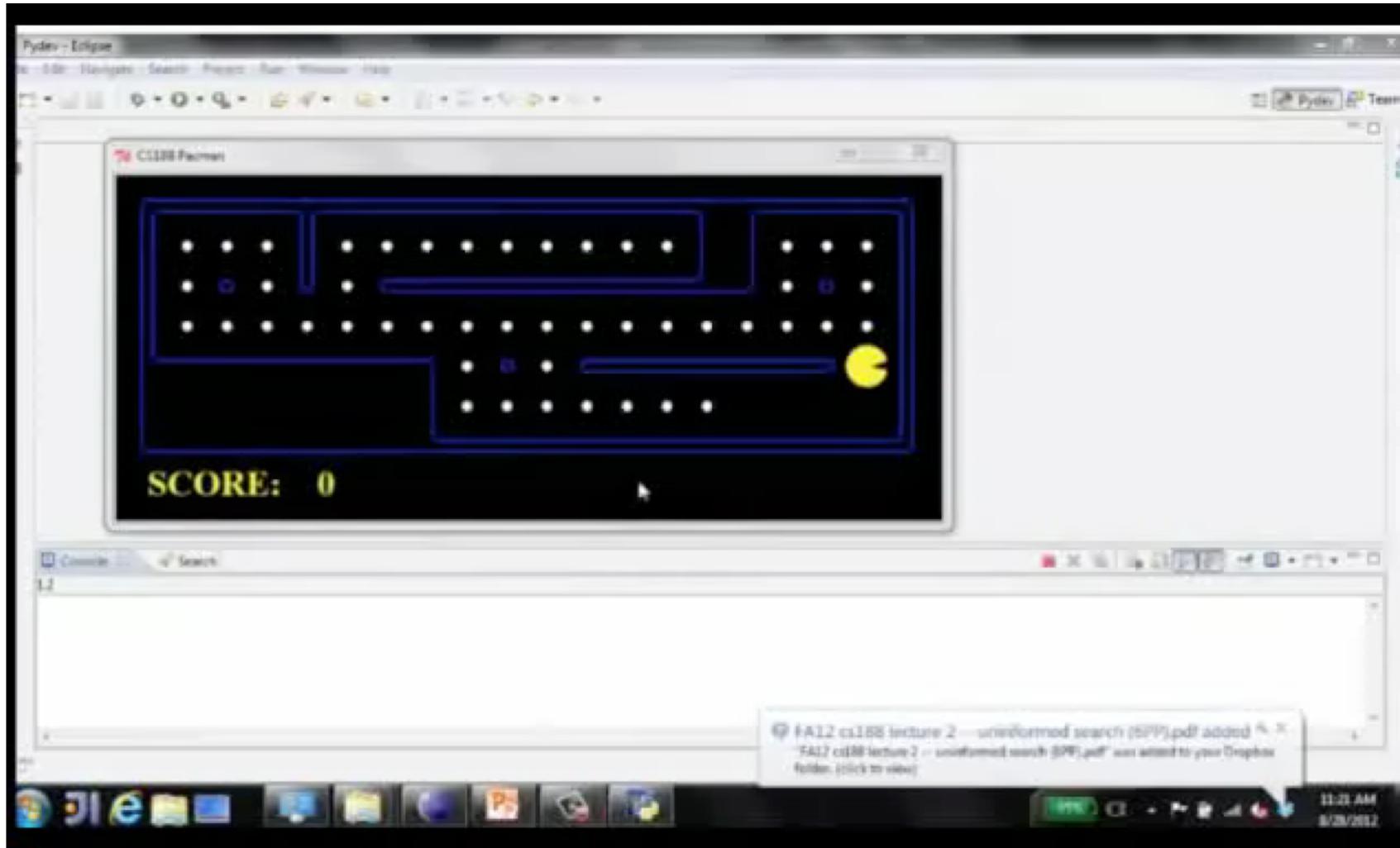
چیزی که برایشان اهمیت دارد:  
محیط در حال حاضر چگونه است؟

آیا این عامل‌ها می‌توانند عقلانی رفتار کنند؟

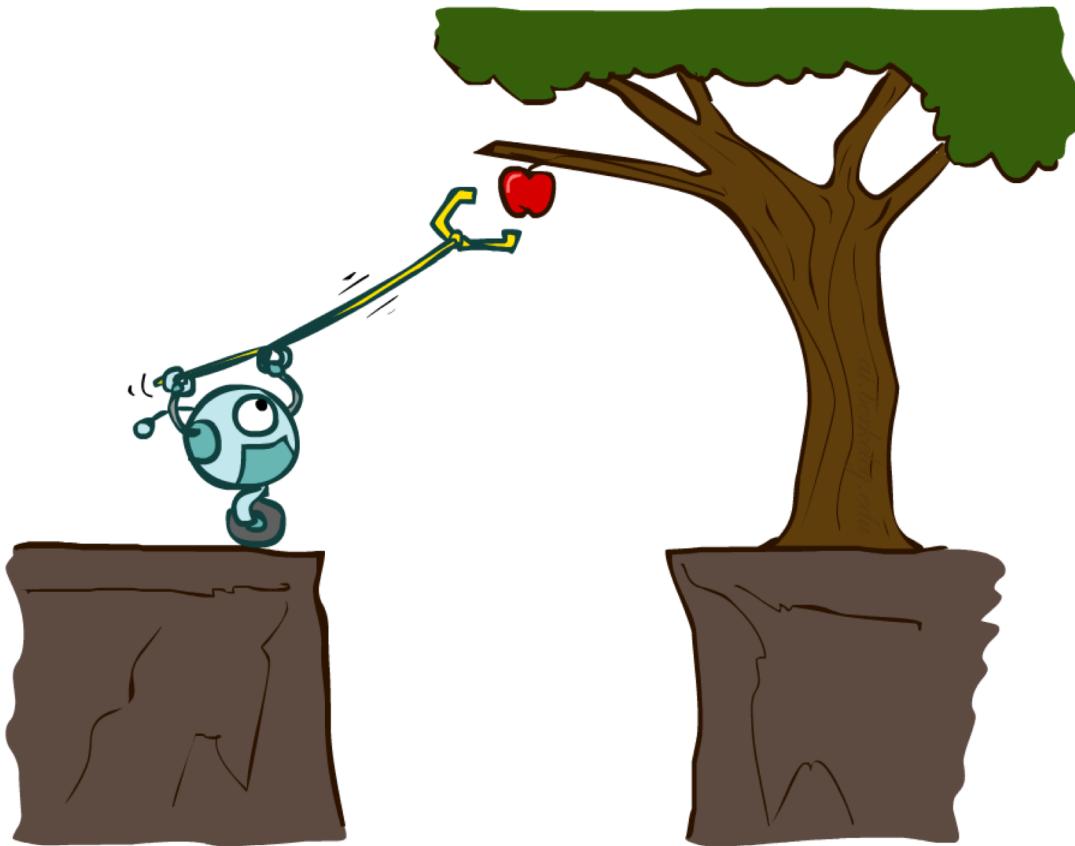
# عامل‌های واکنشی – Reflex



# عامل‌های واکنشی – Reflex



# عامل‌های نقشه‌کش



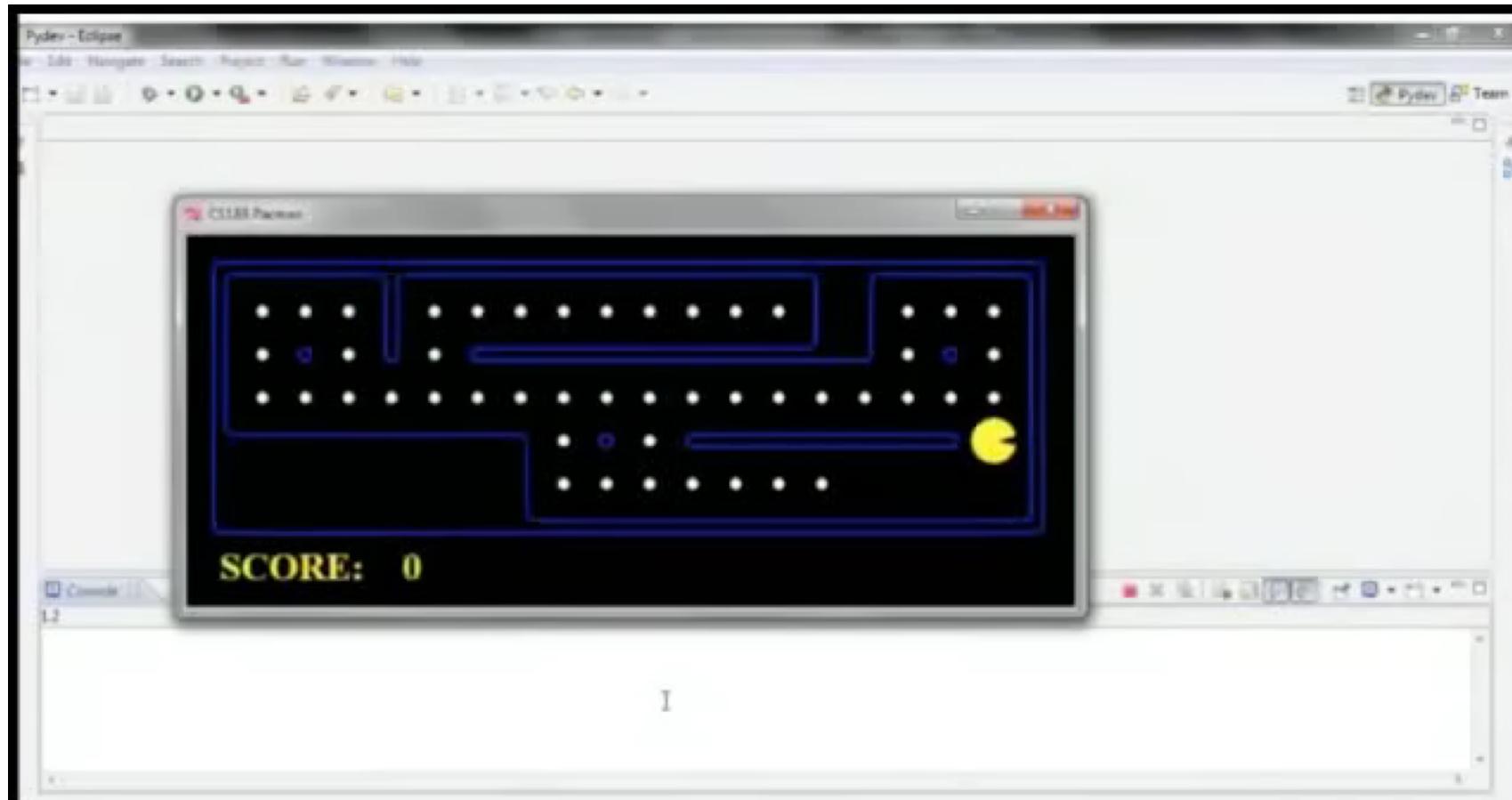
بر اساس خروجی کارها تصمیم‌گیری می‌کنند

به مدلی نیاز دارند که چگونی تحولات محیط را نشان دهد

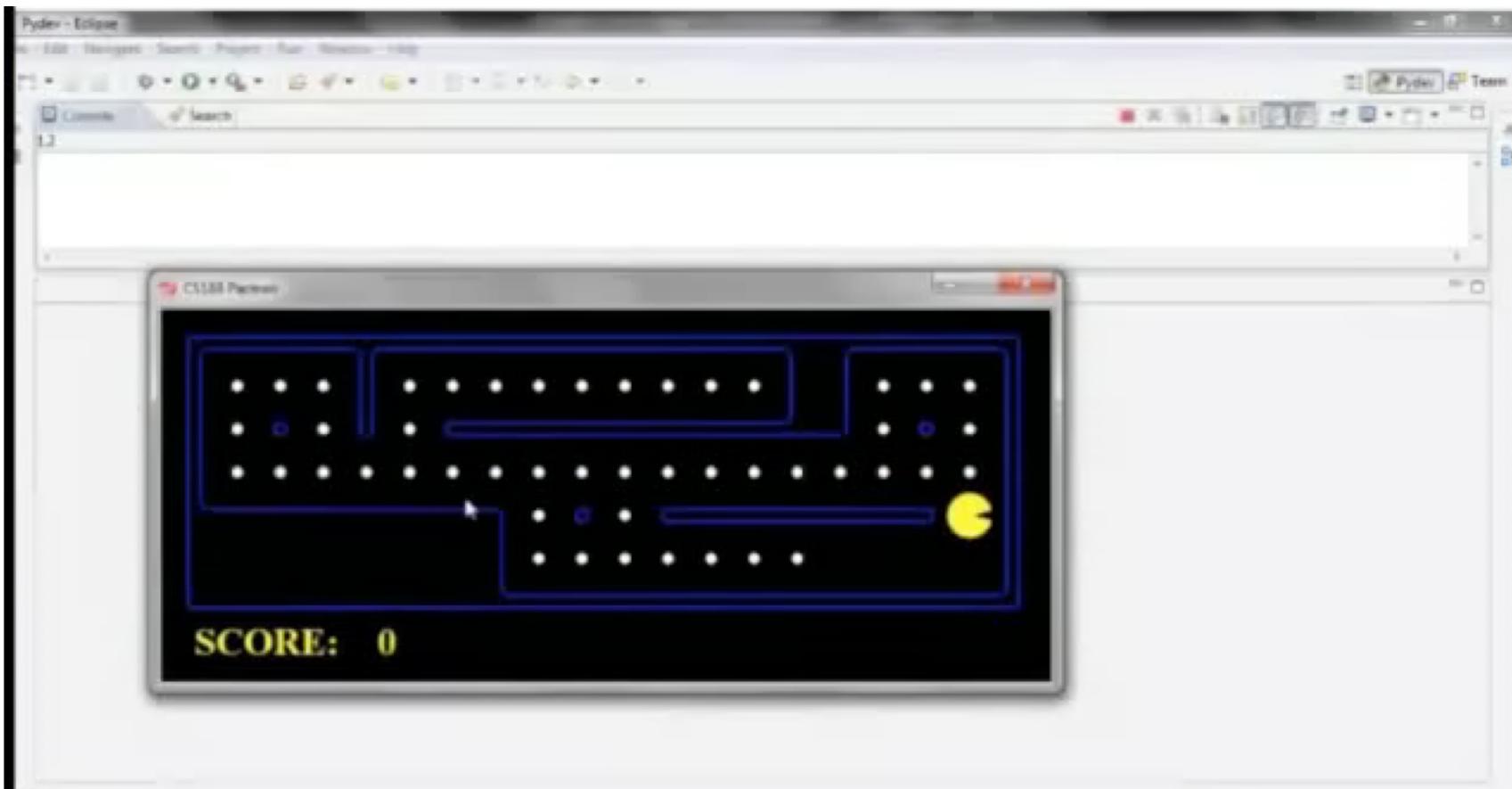
چیزی که برایشان اهمیت دارد:  
محیط در آینده چگونه خواهد بود؟

نقشه‌ی «بهینه» و «کامل»  
Optimal & complete

# نقشه‌کشی مجدد – Re-planning

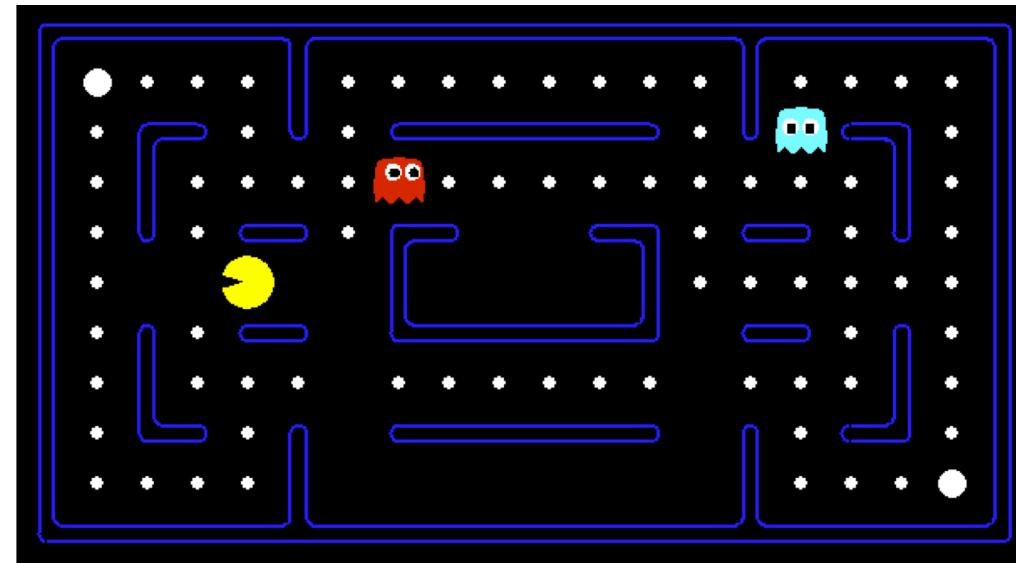


# از پیش نقشه‌کشی – Planning

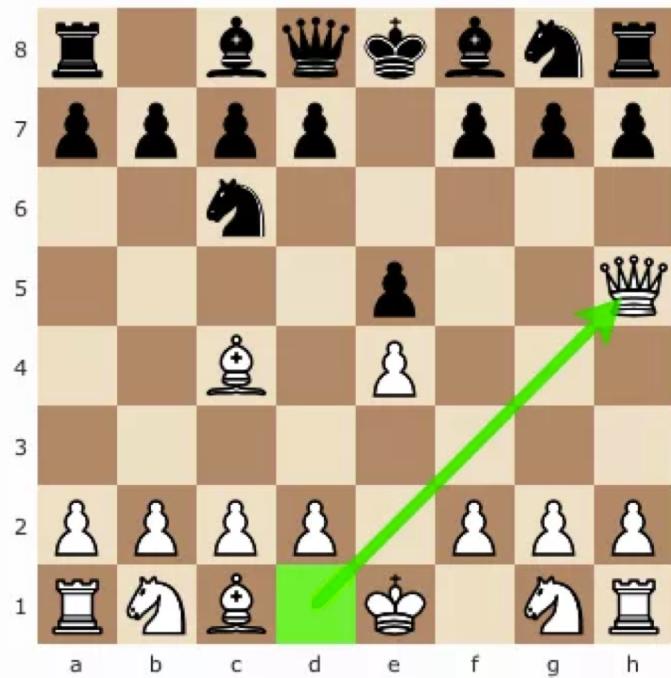


# مسائل جستجو

---

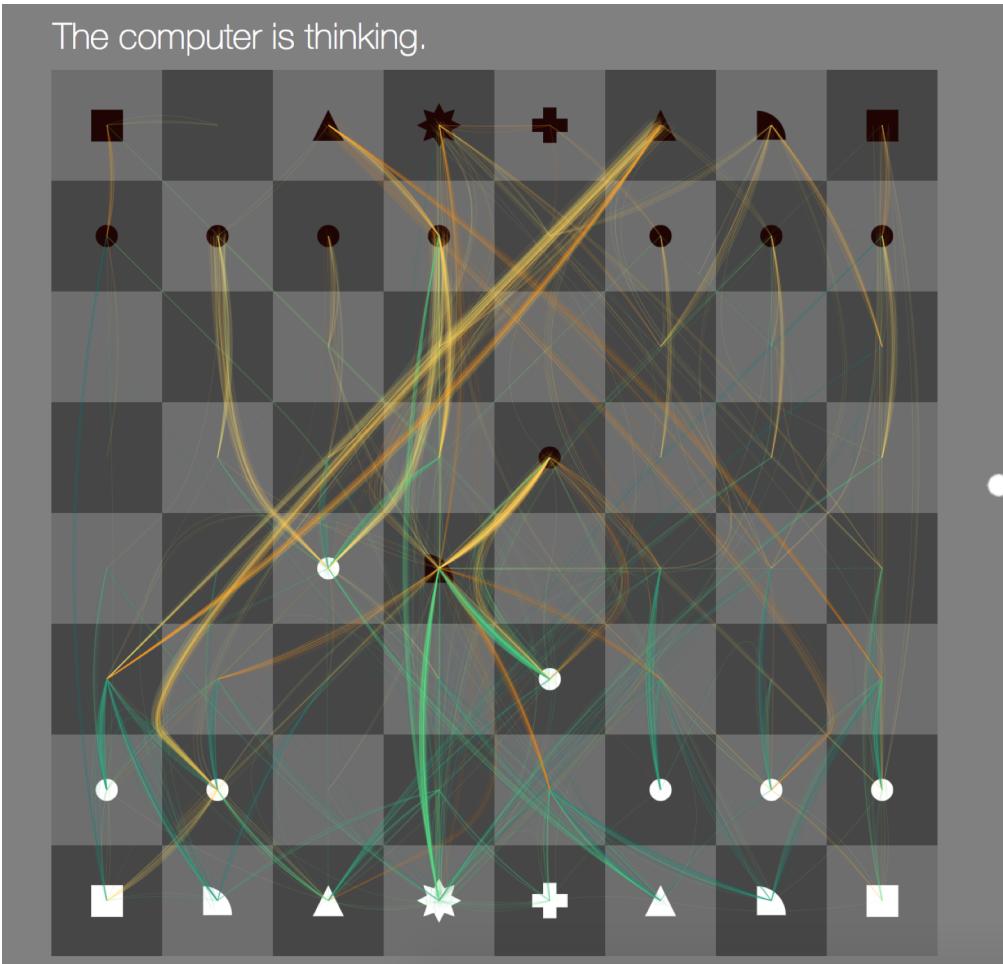


# مسائل جستجو



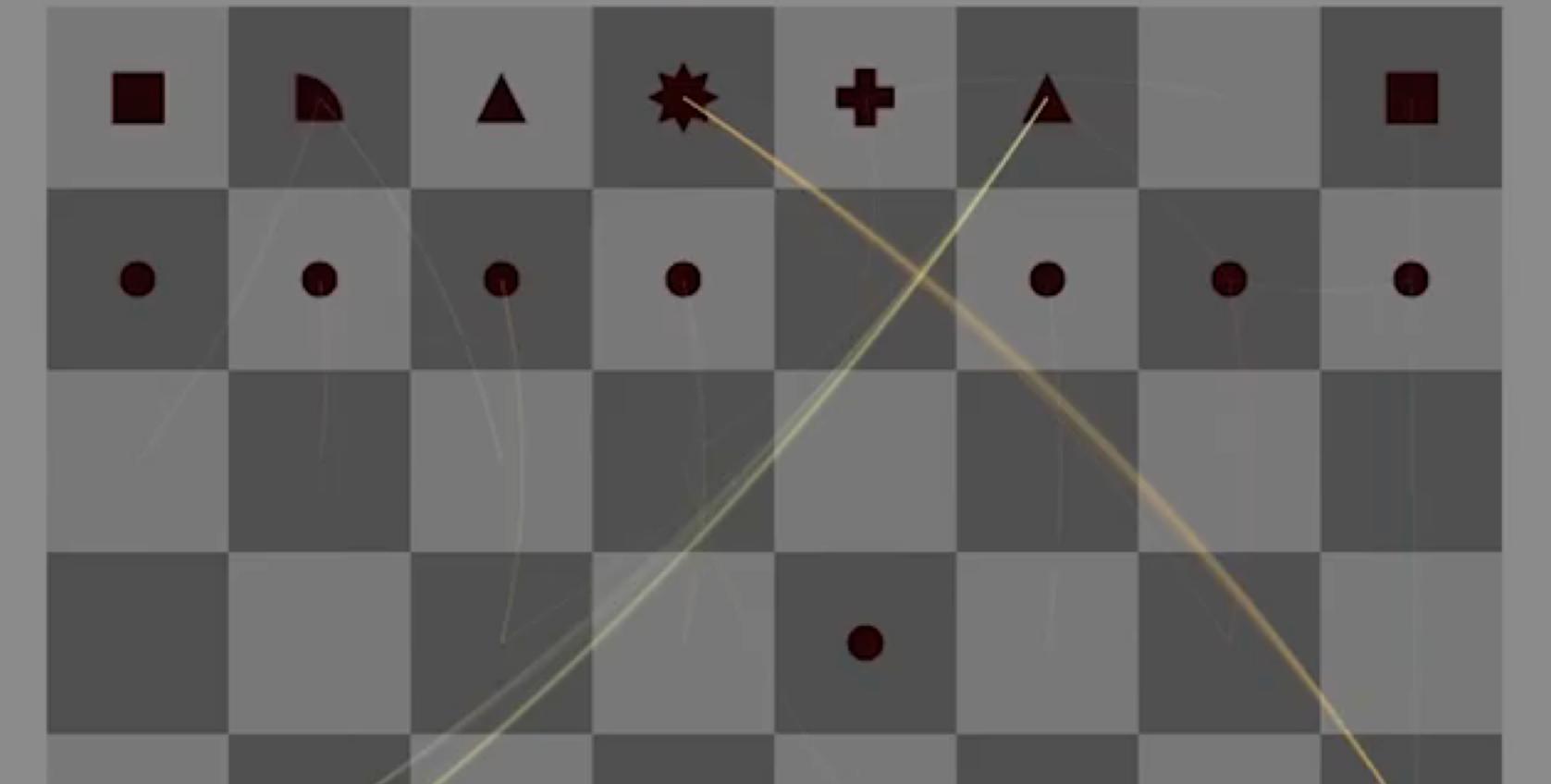
# مسائل جستجو

<http://www.bewitched.com/chess/>



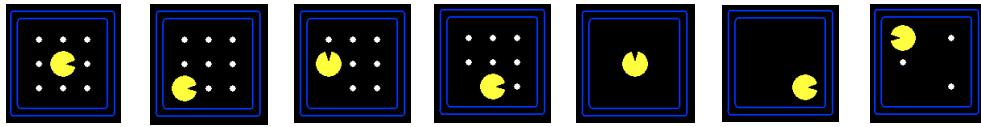
# مسائل جستجو

The computer is thinking.

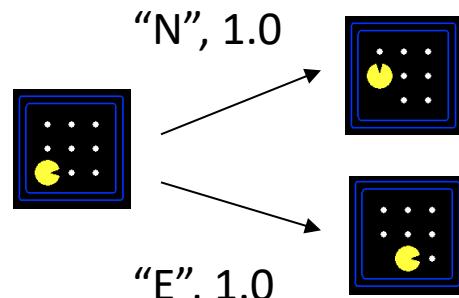


# مسائل جستجو

اجزای یک مسئله جستجو:



یک فضای حالت - State space  
در هر لحظه محیط چگونه است؟



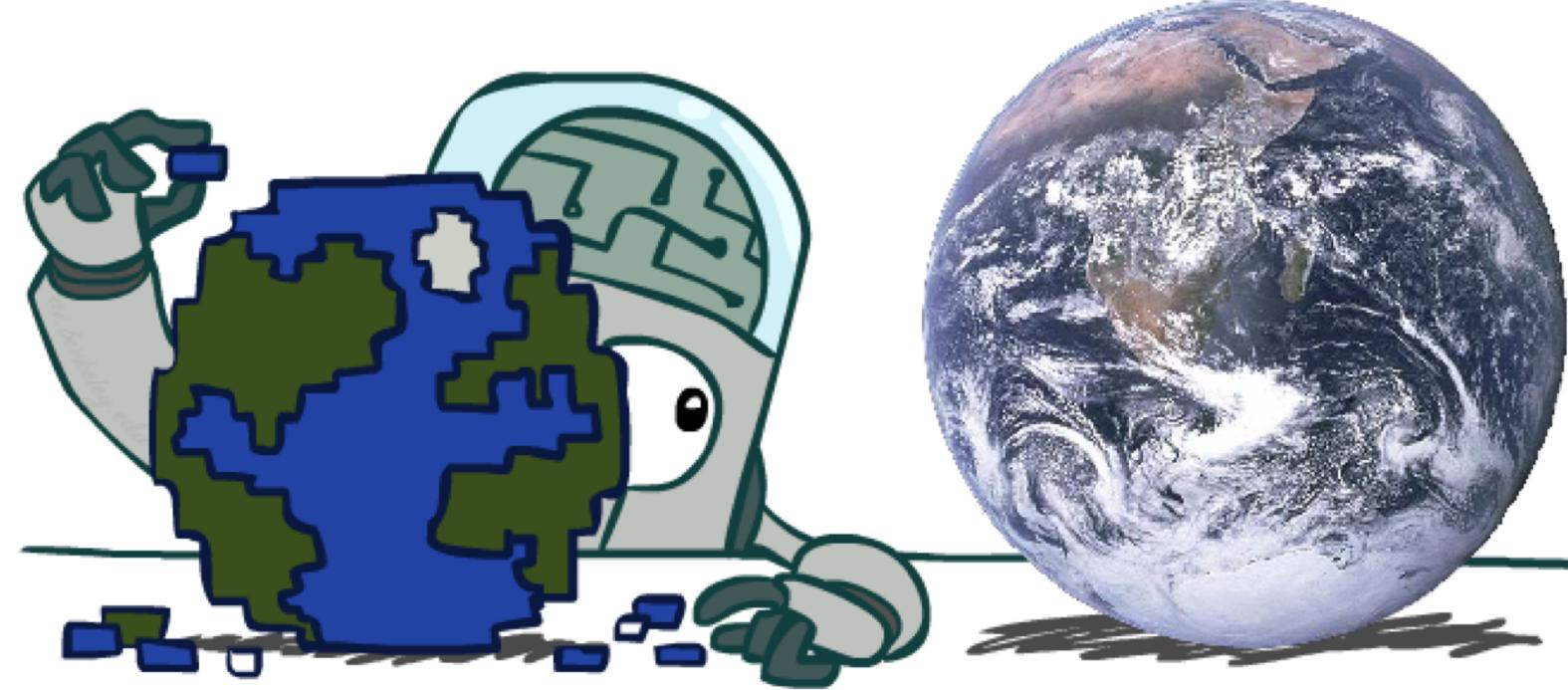
یک تابع «پیشبر» A successor function  
محیط چگونه پاسخ می‌دهد؟ حالت فعلی و تصمیم مدنظر را گرفته و حرکت‌ها (کنش‌های) ممکن را با هزینه‌شان نشان می‌دهد

نقطه (حالت) شروع و پایان

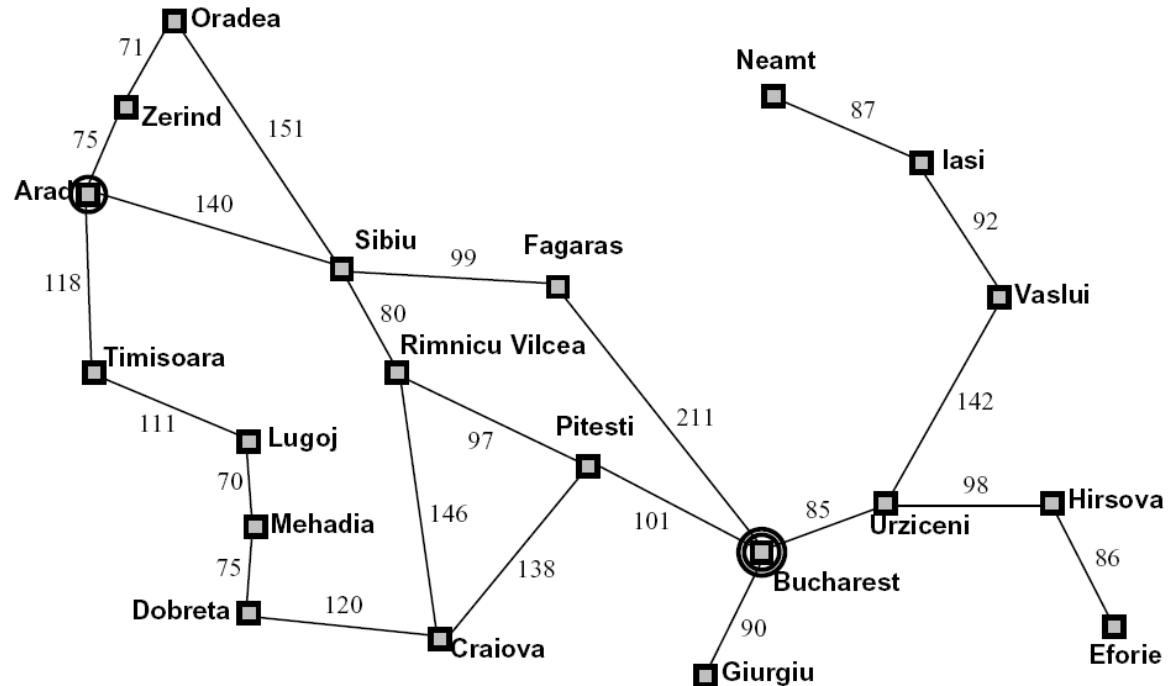
جواب مسئله چیزی نیست جز تعدادی کنش (نقشه) که حالت شروع را به حالت پایان می‌رساند

# مسائل جستجو - اهمیت مدل

---



# مثال: سفر در رومانی!



فضای حالت

شهرها

تابع پیشبر

جاده‌های بین شهری

هزینه: مسافت

نقطه‌ی شروع:

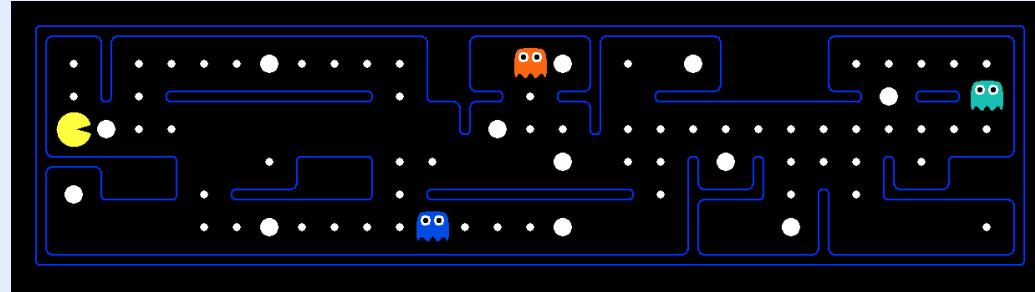
هدف (نقطه‌ی پایان)

Is state == Bucharest?

را حل؟

# فضای حالت

«حالت جهان» شامل تمامی جزئیات محیط می‌شود



«حالت مسئله» تنها جزئیات مورد نیاز را در می‌گیرد

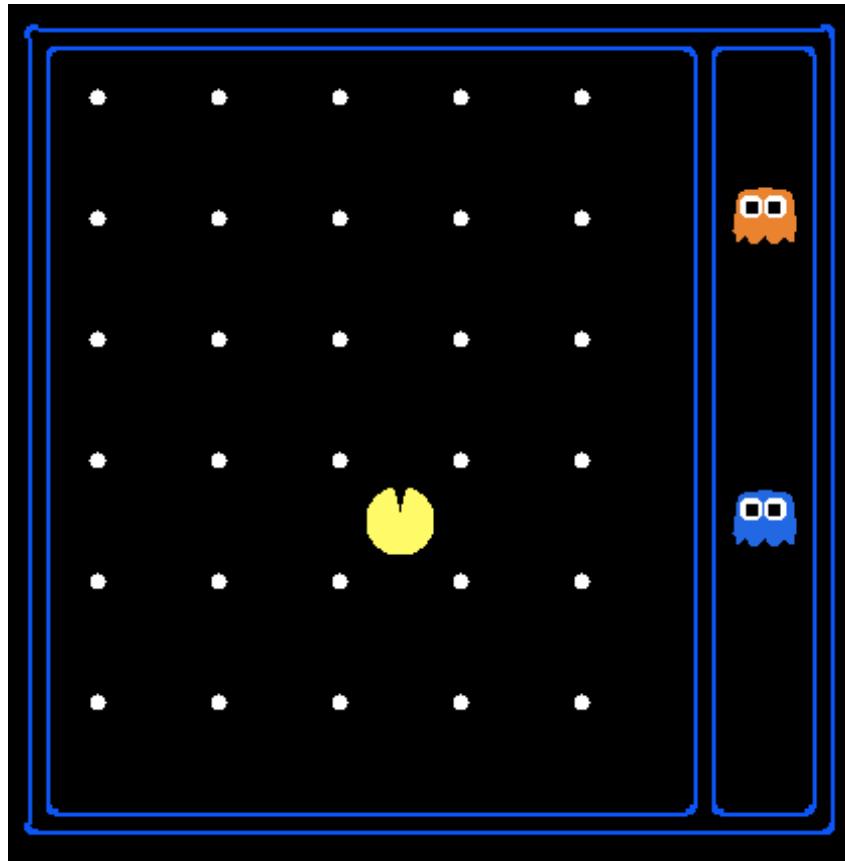
## مسیریابی

- States:  $(x,y)$  location
- Actions: NSEW
- Successor: update location only
- Goal test: is  $(x,y)=\text{END}$

## خوردن غذایها

- States:  $\{(x,y), \text{dot booleans}\}$
- Actions: NSEW
- Successor: update location and possibly a dot boolean
- Goal test: dots all false

# اندازه‌ی فضای حالت؟



تعداد موقعیت‌های عامل: ۱۲۰

تعداد «غذا»: ۳۰

موقعیت‌های «روح»‌ها: ۱۲

تعداد جهات عامل: ۴

چه تعداد؟

حالت در جهان مسئله داریم؟

$$120 \times (2^{30}) \times (12^2) \times 4$$

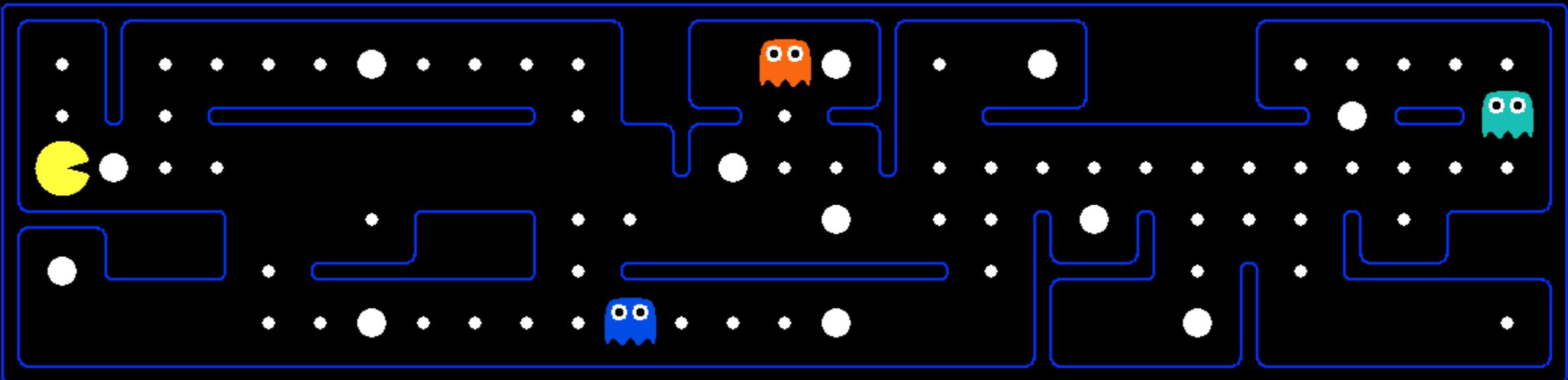
تعداد حالات برای مسیریابی؟

۱۲۰

تعداد حالات برای خوردن غذاهای؟

$$120 \times (2^{30})$$

# کوییز!



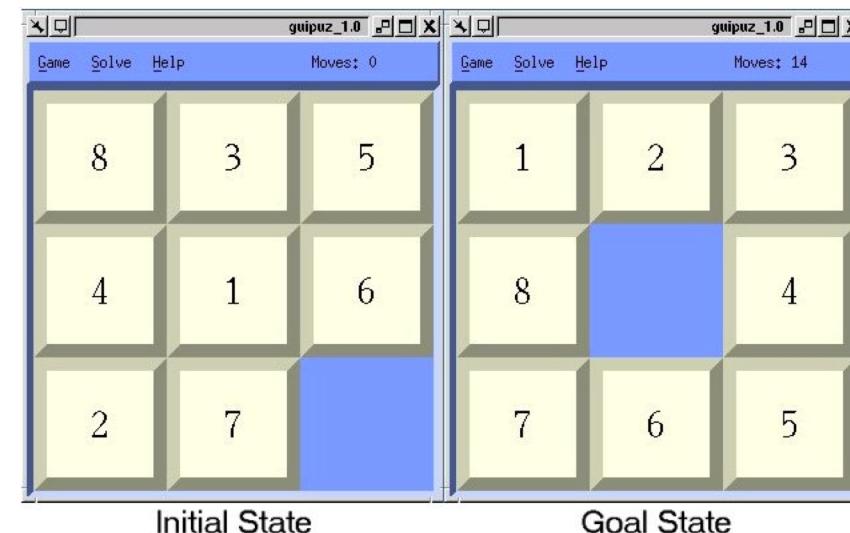
مساله: تمام غذاها رو بخور در حالی که روح‌ها همیشه شوکه باشند  
فضای حالت چه جزئیاتی را داشته باشد؟

موقعیت عامل، غذاها (بولین)، دایرها (بولین)، زمان باقی مانده از شوک

# کوییز!

فضای حالت چه جزئیاتی را داشته باشد؟

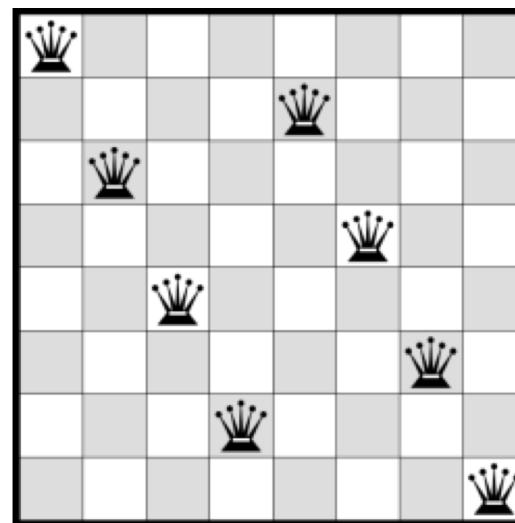
## 8-puzzle game



# کوییز!

فضای حالت چه جزئیاتی را داشته باشد؟

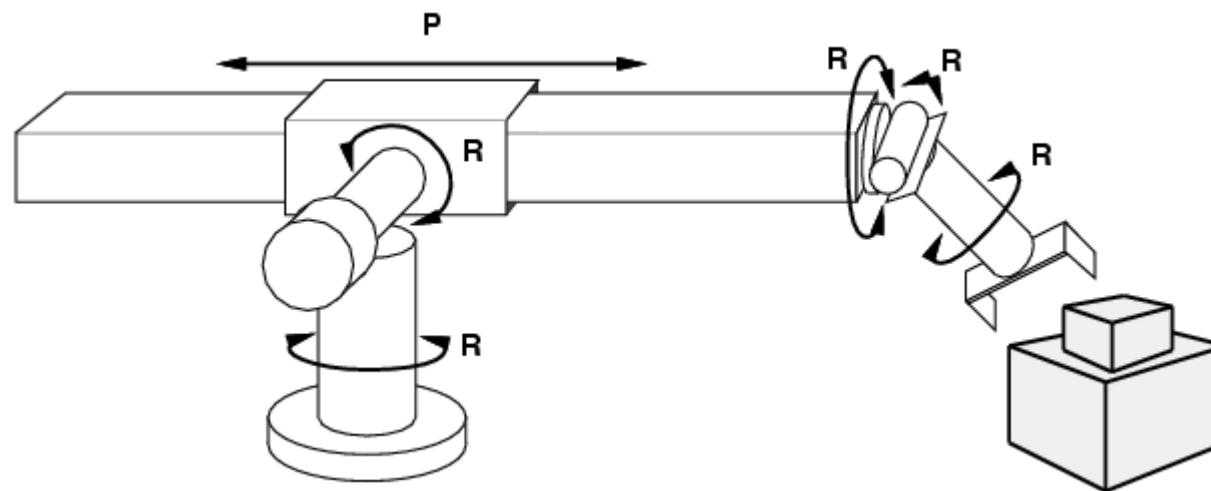
8-queen game



# کوییز!

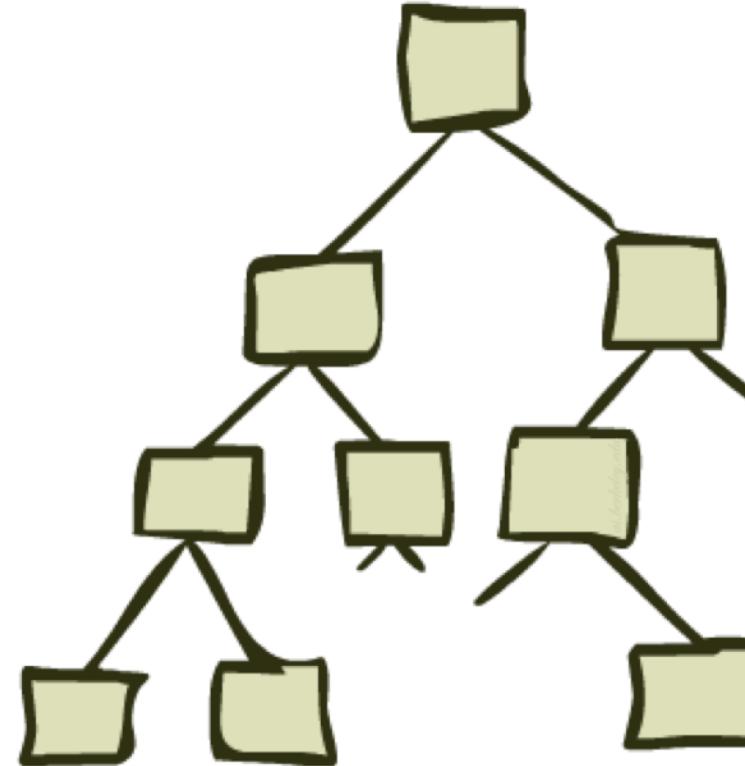
فضای حالت چه جزئیاتی را داشته باشد؟

Robotic assembly

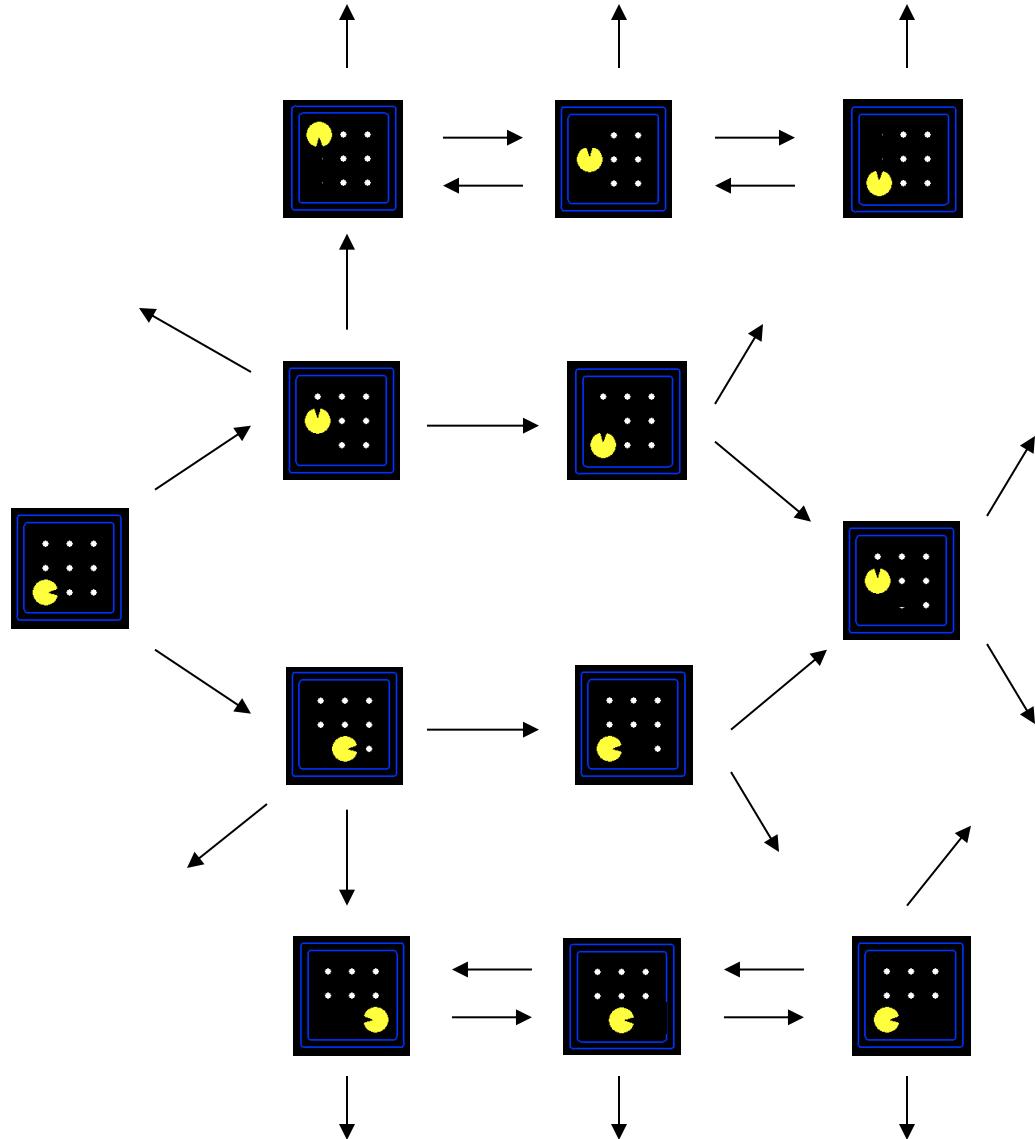


# گراف‌های فضای حالت – درخت جستجو

---



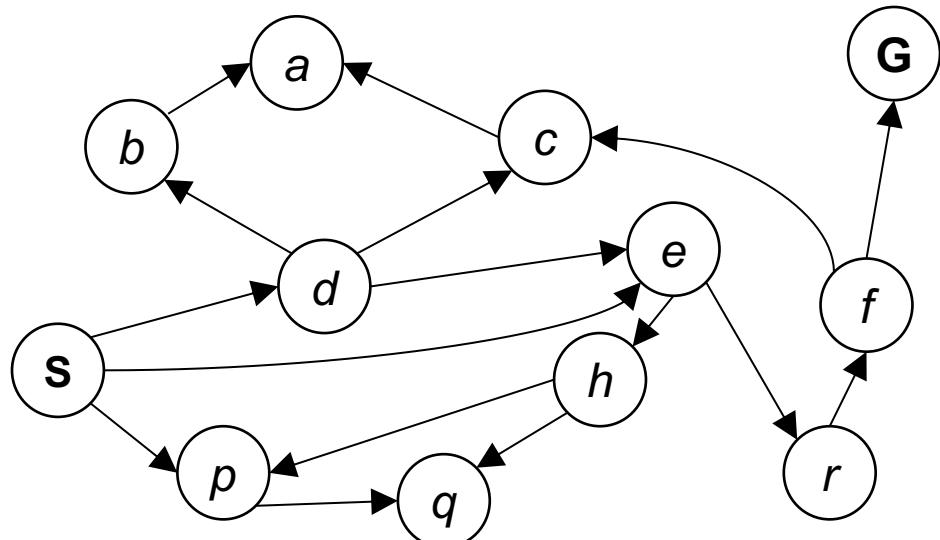
# گراف فضای حالت



یک مدل ریاضی از مسئله  
گره‌ها وضعیت‌های ممکن هستند  
لبه‌ها، تابع‌های پیش‌بر  
هدف، یکی (یا تعدادی) از گره‌های است

- در گراف فضای حالت، هر حالت فقط یک بار اتفاق می‌افتد
- هرچند معمولاً برای مسائل نمی‌توان کل گراف را کشید (یا در مموری جا کرد)

# گراف فضای حالت

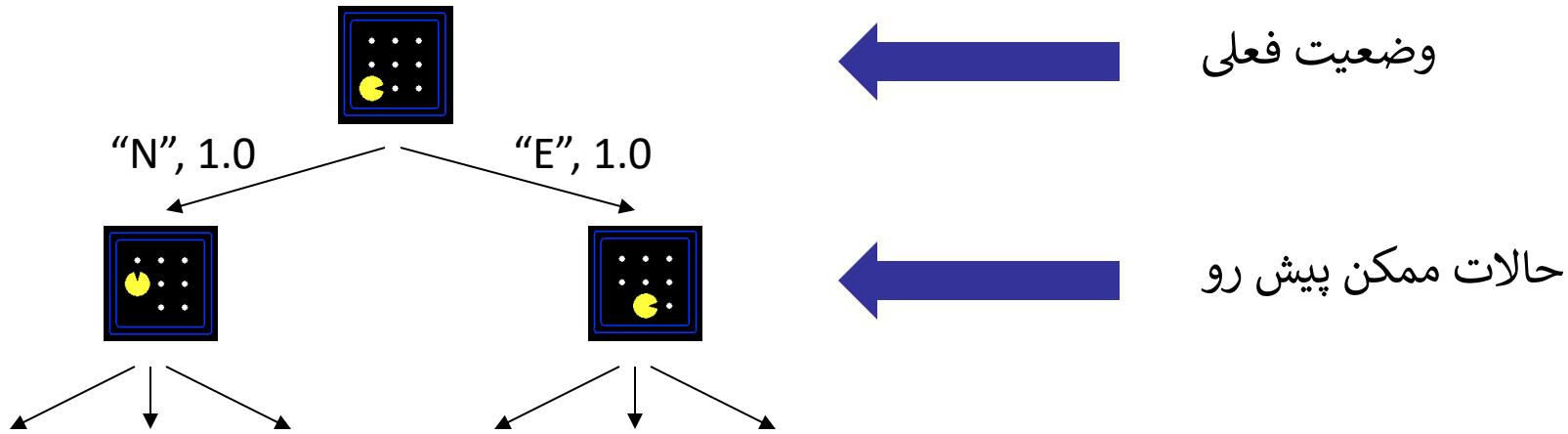


یک مدل ریاضی از مسئله  
گره‌ها وضعیت‌های ممکن هستند  
لبه‌ها، تابع‌های پیش‌بر  
هدف، یکی (یا تعدادی) از گره‌های است

در گراف فضای حالت، هر حالت فقط یک بار اتفاق  
می‌افتد

هرچند معمولاً برای مسائل نمی‌توان کل گراف را کشید  
(یا در مموری جا کرد)

# درخت‌های جستجو



## درخت جستجو

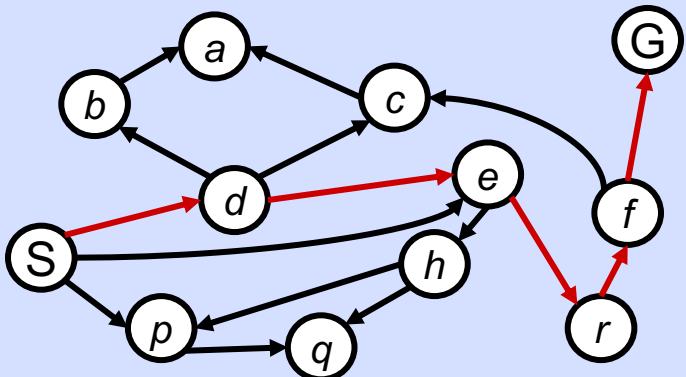
درخت «اینگونه چطور؟» تصمیمات و نتایج آنها

گره‌ها نشان‌دهندهٔ حالات هستند، ولی در عین حال نشان می‌دهند چه تصمیماتی گرفته شده تا به این حالت رسیده شود

برای اکثر مسائل درخت خیلی بزرگ و کشیدن آن ناممکن است

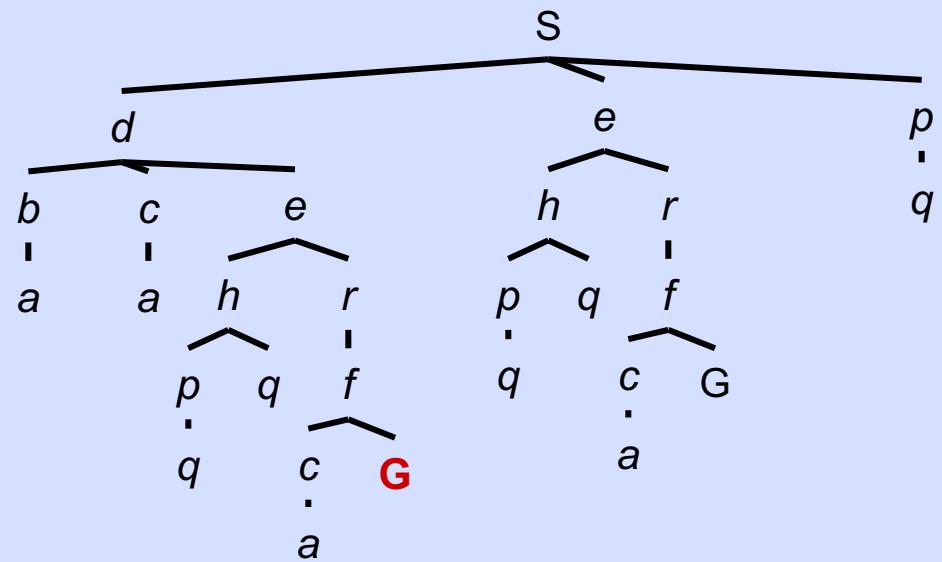
# گراف فضای حالت - درخت جستجو

## گراف فضای حالت



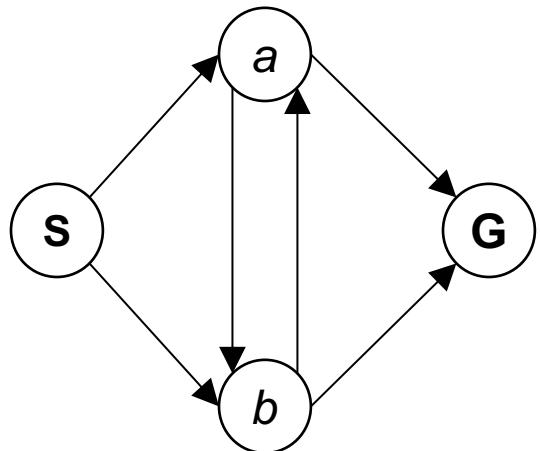
گره‌ها در درخت  
جستجو نشان‌دهنده کل  
مسیری هستند که تا به  
اینجا پیموده شد

## درخت جستجو



# کوییز!

گراف چهار حالتی



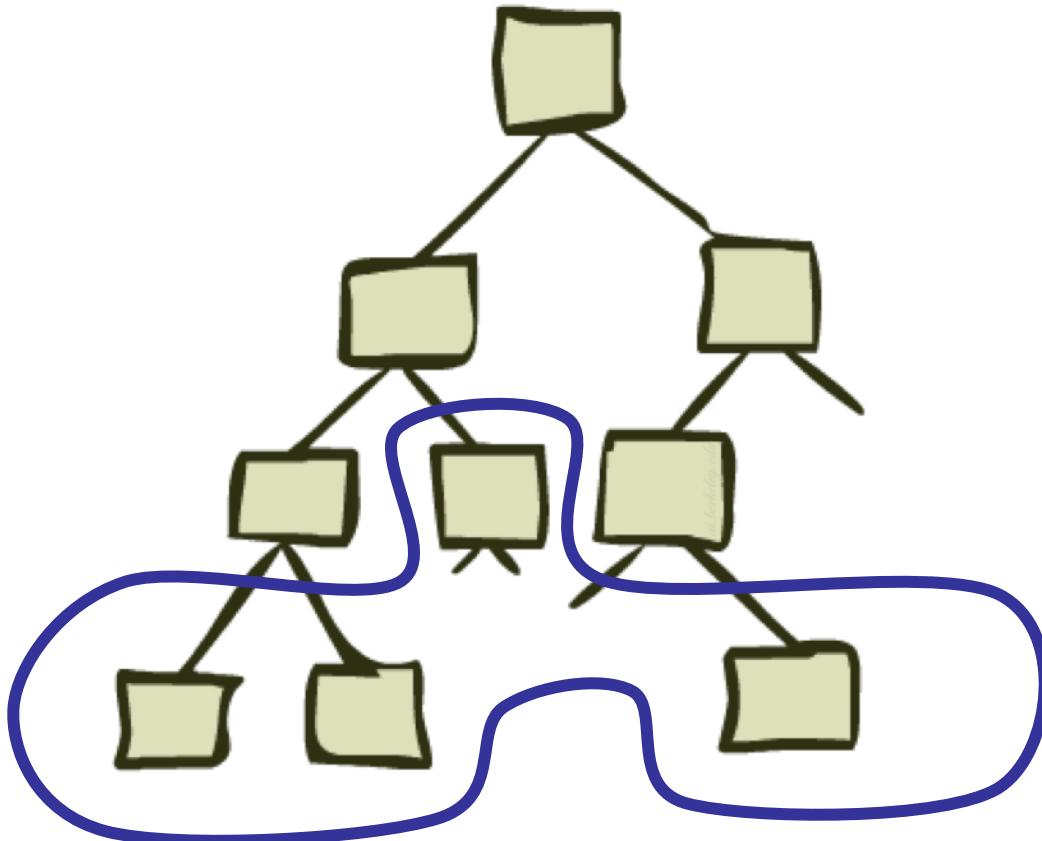
درخت جستجو؟



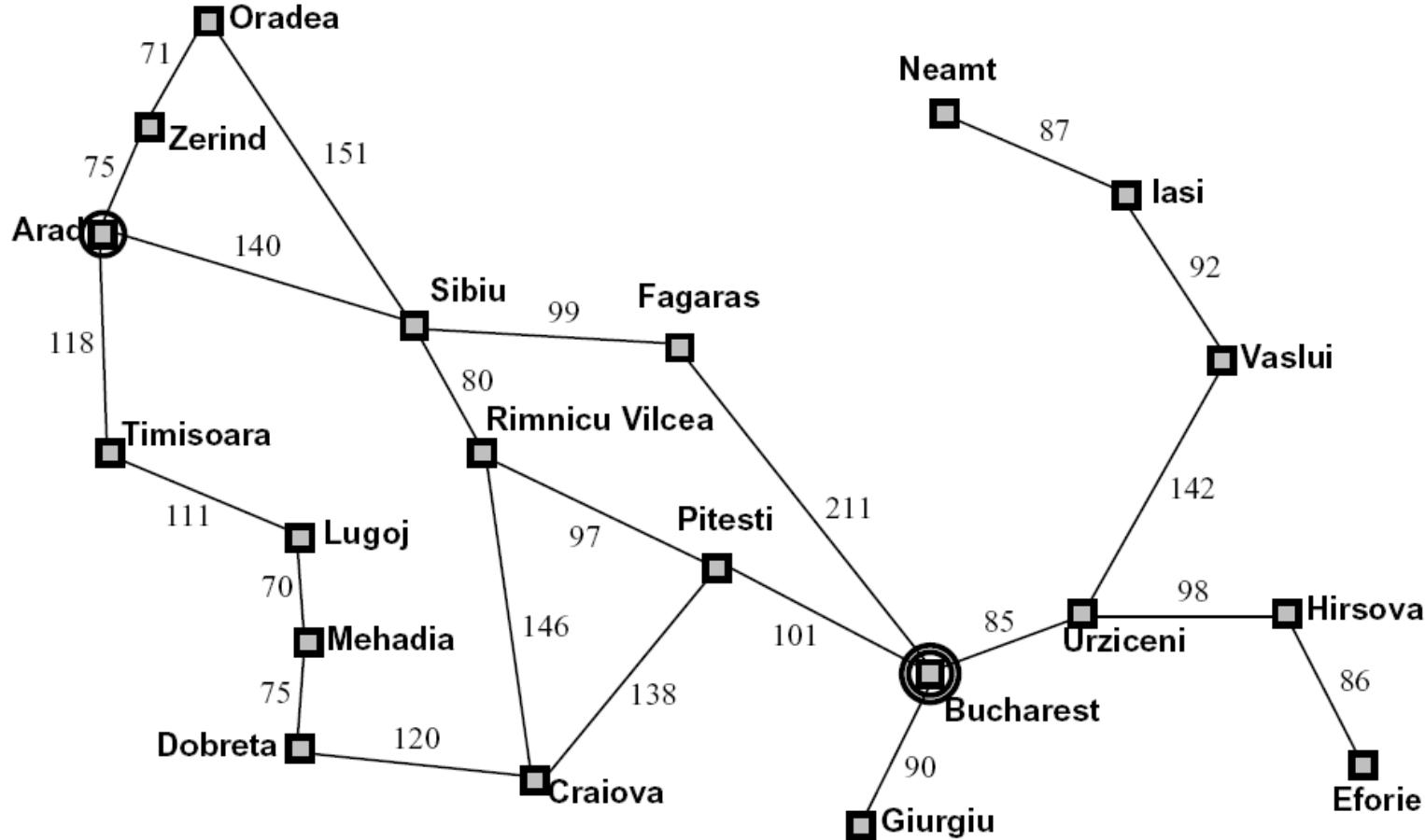
درخت‌های جستجو قسمت‌های تکراری زیادی دارند

# جستجوی درخت

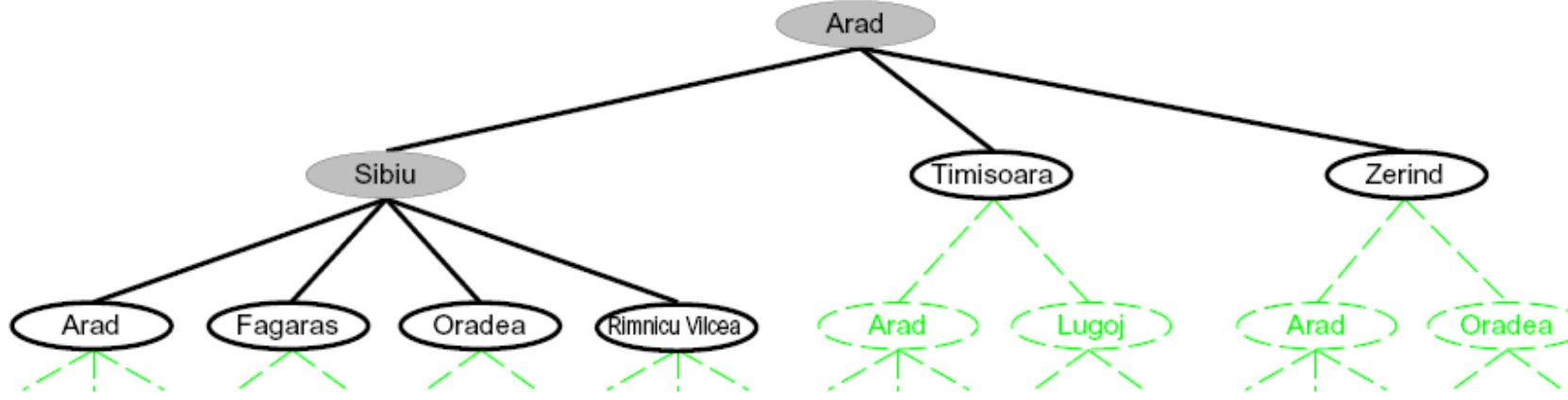
---



# مثال: رومانی



# جستجو توسط درخت جستجو



- ۱- در هر گره، گره های بعدی را بسط بد  
ه
- ۲- گره های را که هنوز بسط نداده ای در «فرینج» نگه دار  
م
- ۳- تا حد امکان، گره های کمتری بسط بد  
ه (راه حل سریع تر)!

# جستجوی درخت - حالت کلی

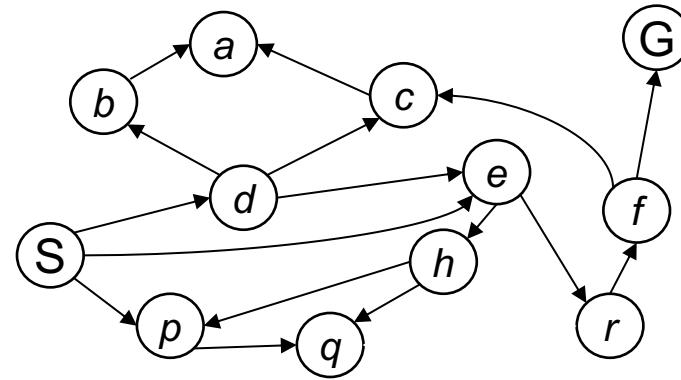
```
function TREE-SEARCH(problem, strategy) returns a solution, or failure
    initialize the search tree using the initial state of problem
    loop do
        if there are no candidates for expansion then return failure
        choose a leaf node for expansion according to strategy
        if the node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

موارد مهم:

فرینج  
بسط دادن  
استراتژی جستجو

سوال اصلی: کدامیک از گره‌ها در فرینج را اول بسط دهیم؟

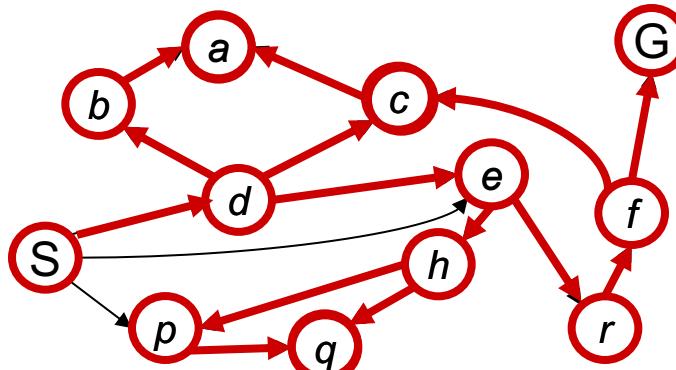
# مثال: جستجوی درختی



# جستجوی اول-عمق – Depth-first search

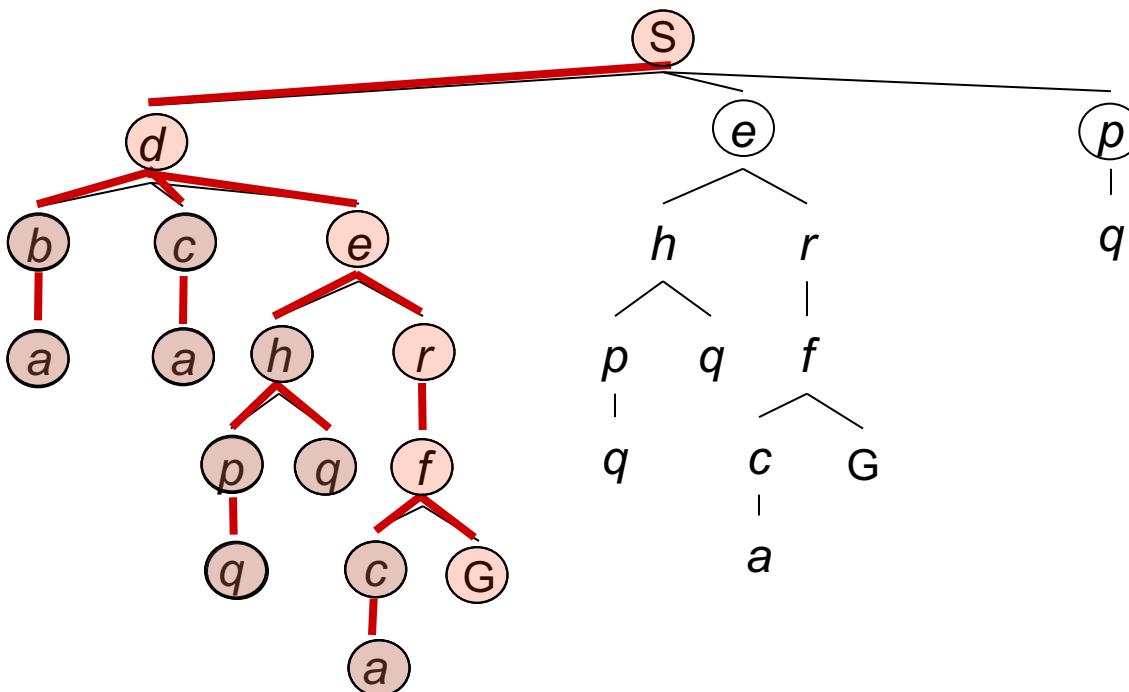


# جستجوی اول-عمق

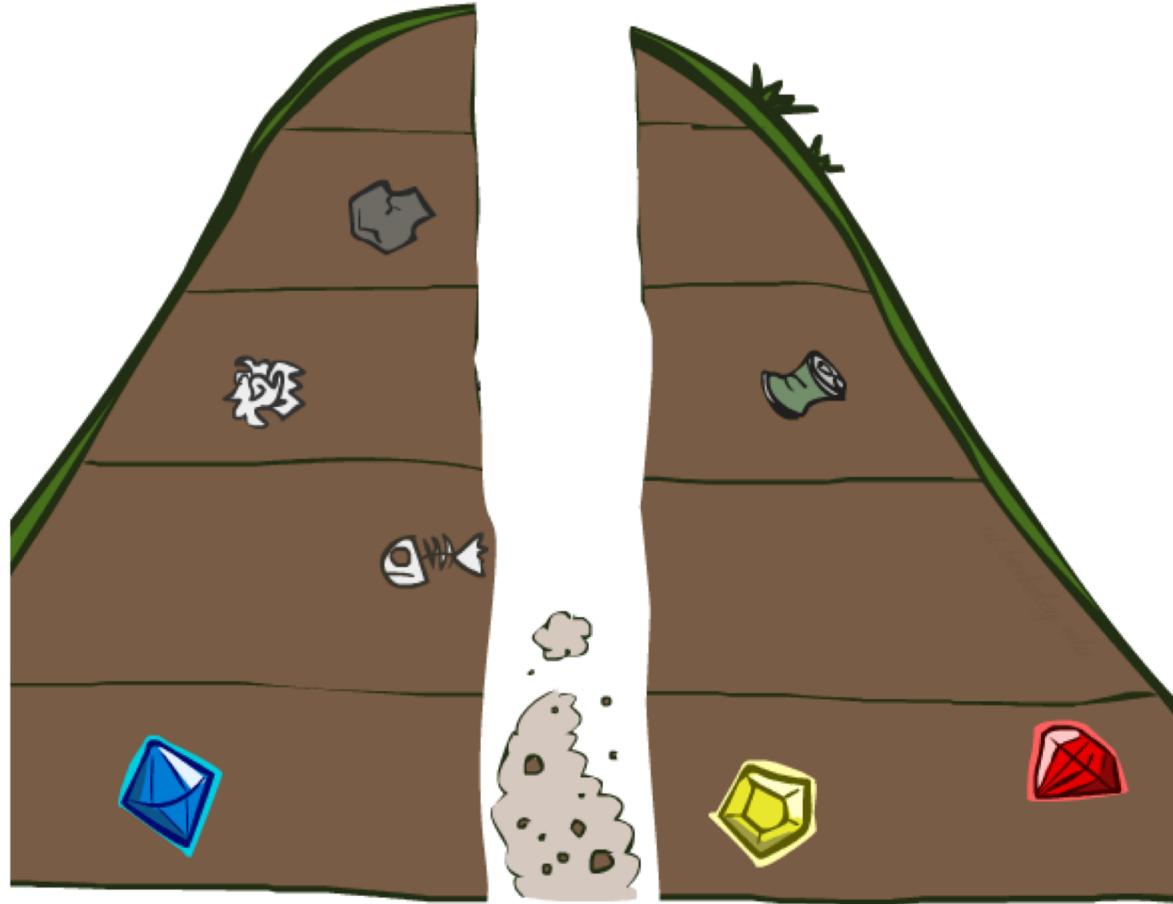


استراتژی: در عمق فرو برو!

فرینج یک دنباله LIFO



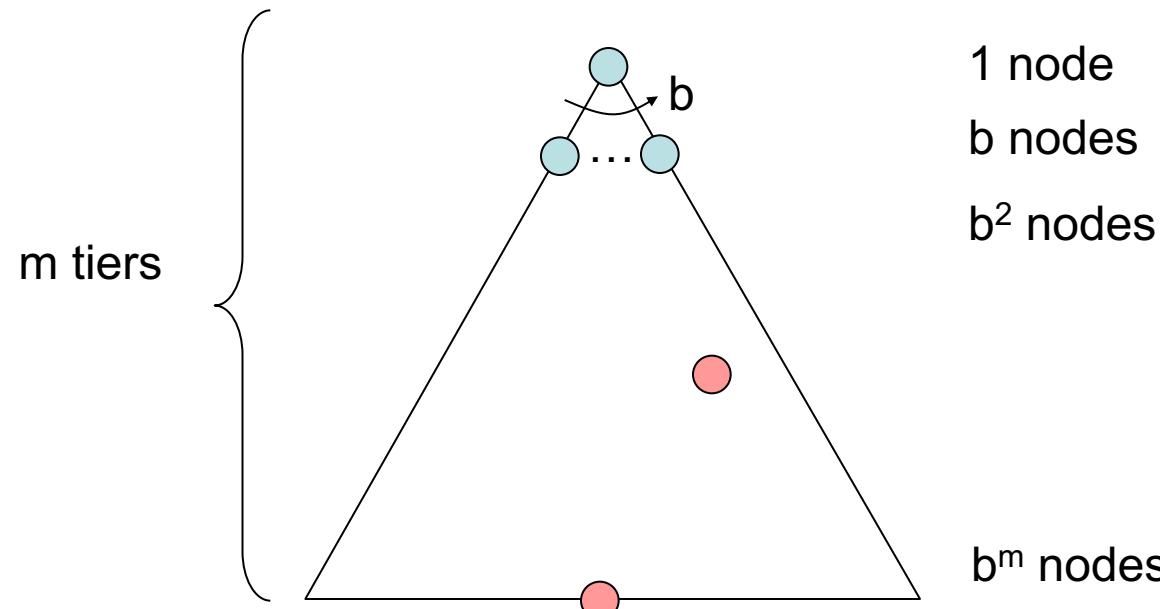
# وېزگىھای الگوريتم جستجو



# ویژگی‌های الگوریتم جستجو

کامل: اگر راه حل وجود دارد، الگوریتم حتما آن را پیدا خواهد کرد

بهینه: الگوریتم بهترین راه حل را پیدا خواهد کرد



زمان اجرا؟

حافظه مورد نیاز؟

درخت جستجو

ضریب انشعاب -

حداکثر عمق:  $m$

راه حل در اعماق مختلف

تعداد گره‌های درخت

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

# ویژگی‌های الگوریتم جستجوی اول-عمق

چه گره‌ای را اول بسط دهیم؟

از چپ در عمق  
وقتی به آخر (برگ) رسیدی، به شاخه‌ی بعدی برو  
اگر عمق متناهی باشد، زمان  $O(b^m)$

حافظه؟

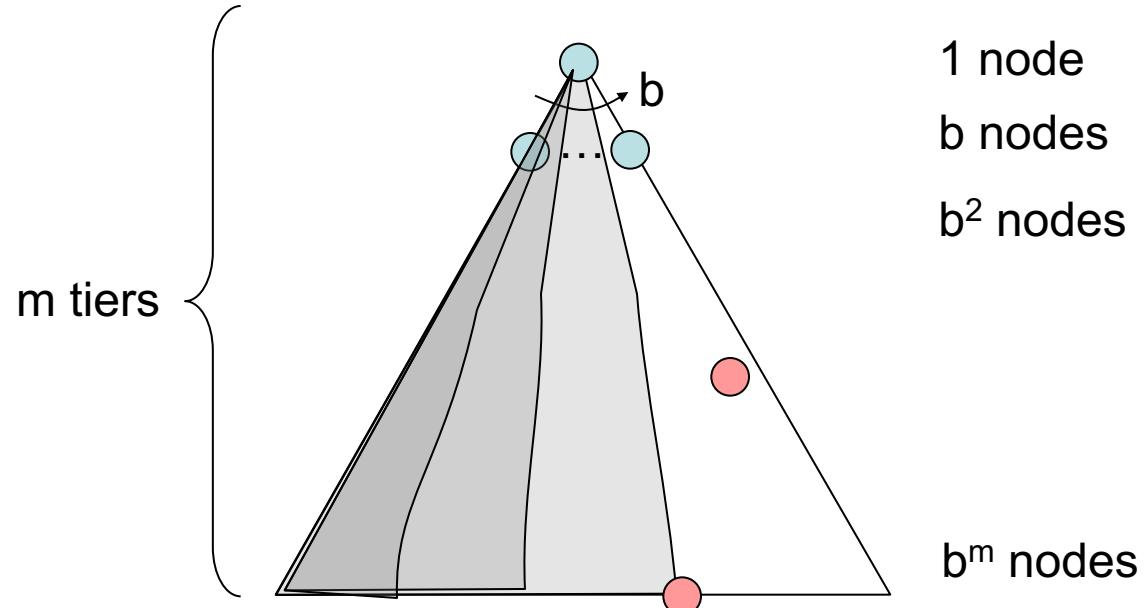
خیلی خوب ( $km$ )، فقط گره‌های شاخه‌ی فعال -  $O(bm)$

کامل؟

اگر عمق متناهی باشد (سیکل نداشته باشیم)

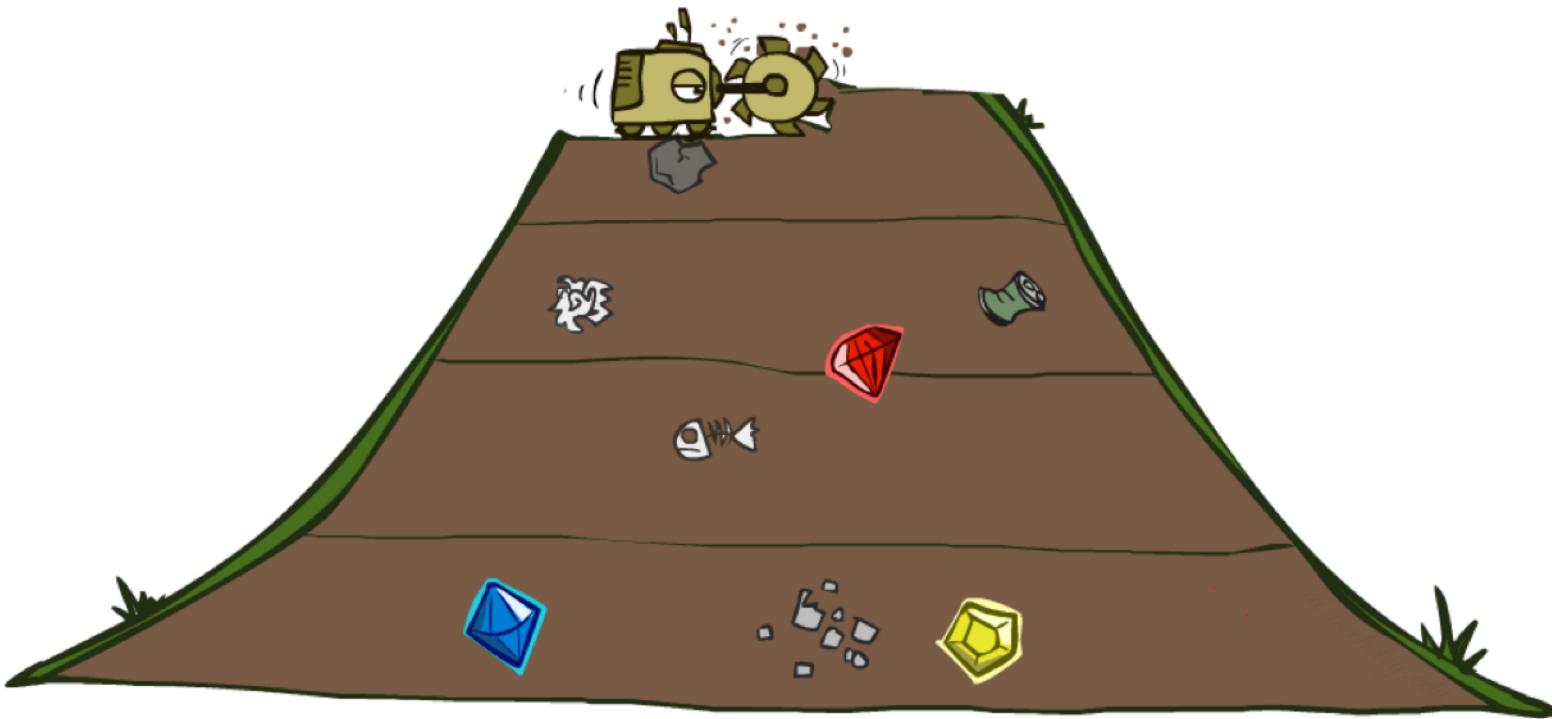
بهینه؟

خیر! چپ‌ترین راه حل



# جستجوی اول-سطح

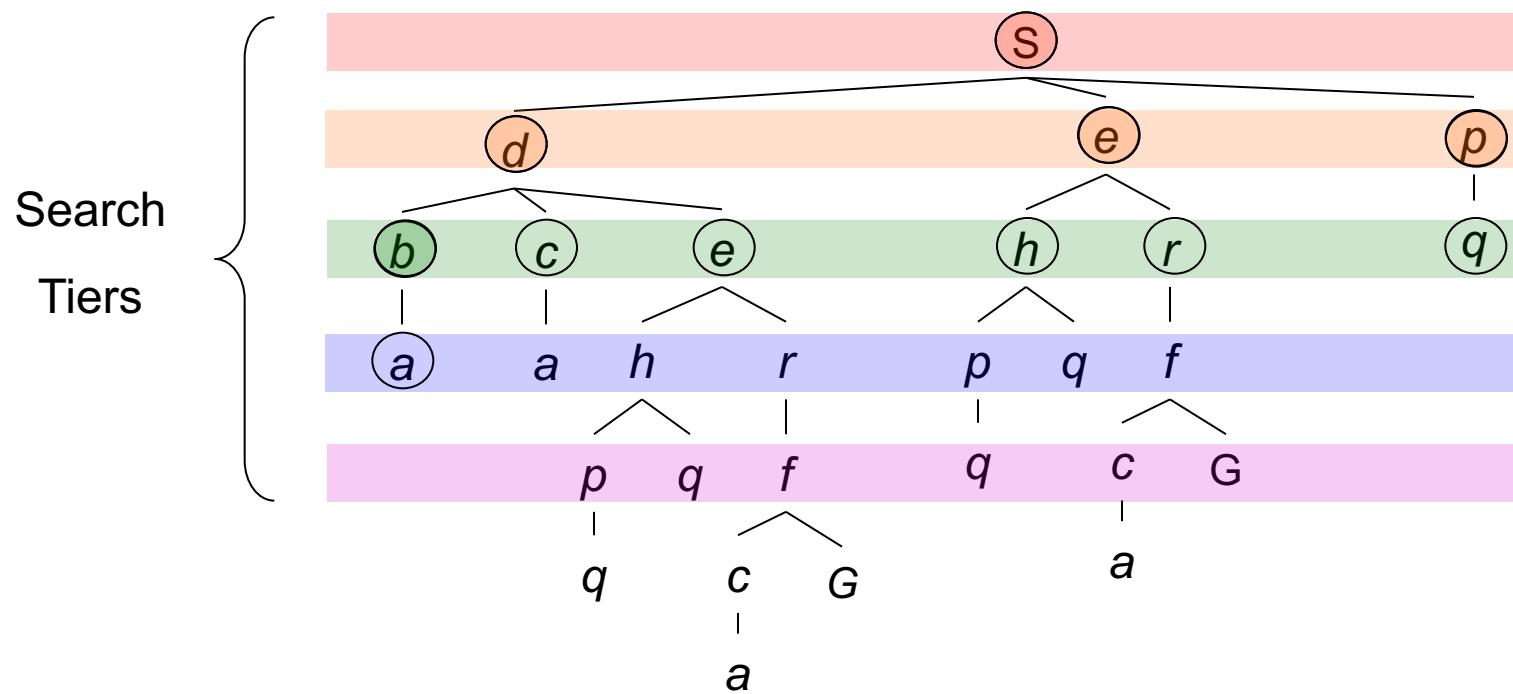
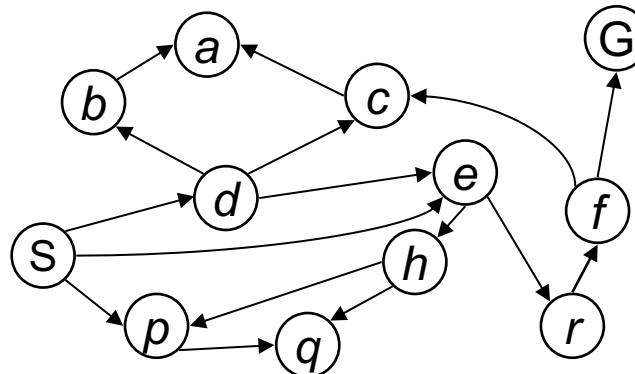
---



# جستجوی اول-سطح

استراتژی: قبل از اینکه در عمق بروی، سطح فعلی را بسط بده!

FIFO



# ویژگی‌های الگوریتم اول-سطح

چه گره‌ای را اول بسط دهیم؟

تمامی گره‌های بالاتر از کم عمق‌ترین راه حل وقتی به راست ترین گره رسیدی، یک سطح در عمق فرو برو  
عمق بالاترین راه حل  $s$   
زمان  $O(b^s)$

حافظه؟

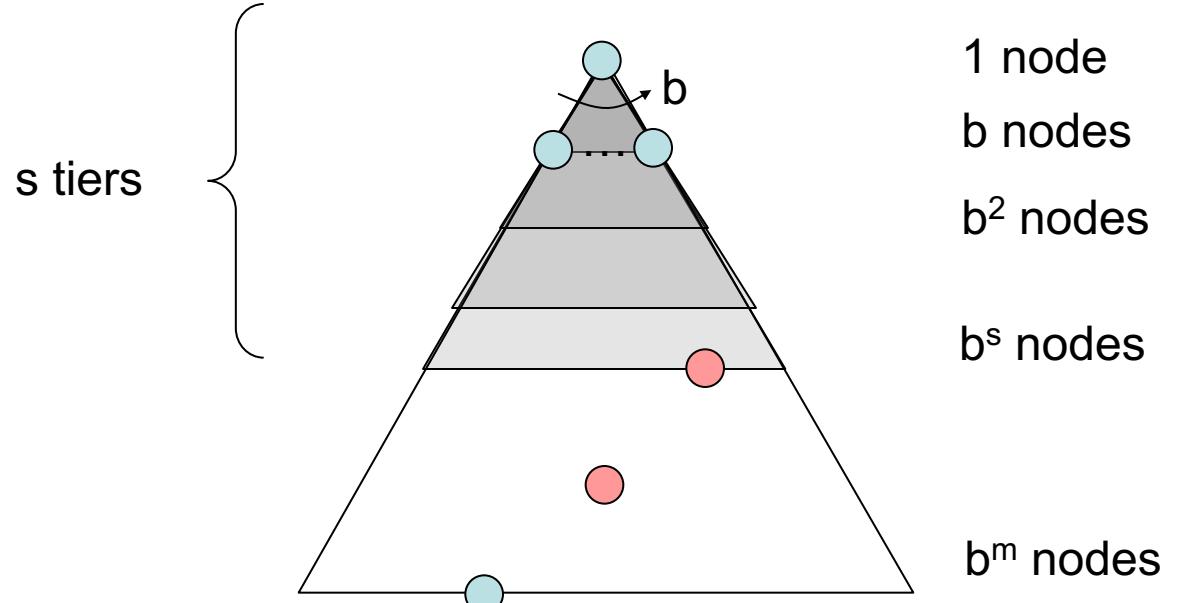
زیاد! تمام گره‌های سطح بالاتر -  $O(b^s)$

کامل؟

بله

بهینه؟

بله، اگر هزینه‌ی تمامی تصمیمات یکسان باشد (در مورد هزینه بیشتر صحبت خواهیم کرد)



# کوییز!

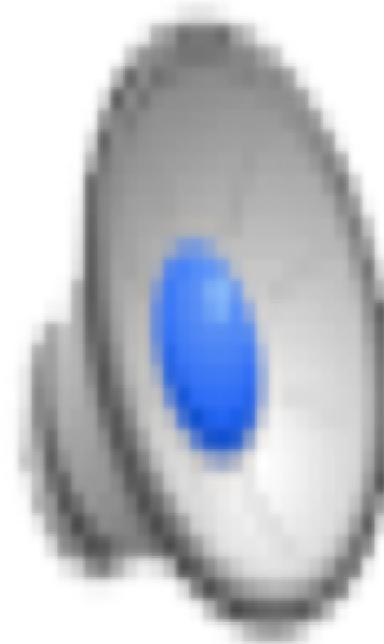
---

در چه شرایطی اول-سطح بهتر عمل خواهد کرد؟

در چه شرایطی اول-عمق بهتر عمل خواهد کرد؟

# اول-سطح یا اول عمق؟

---



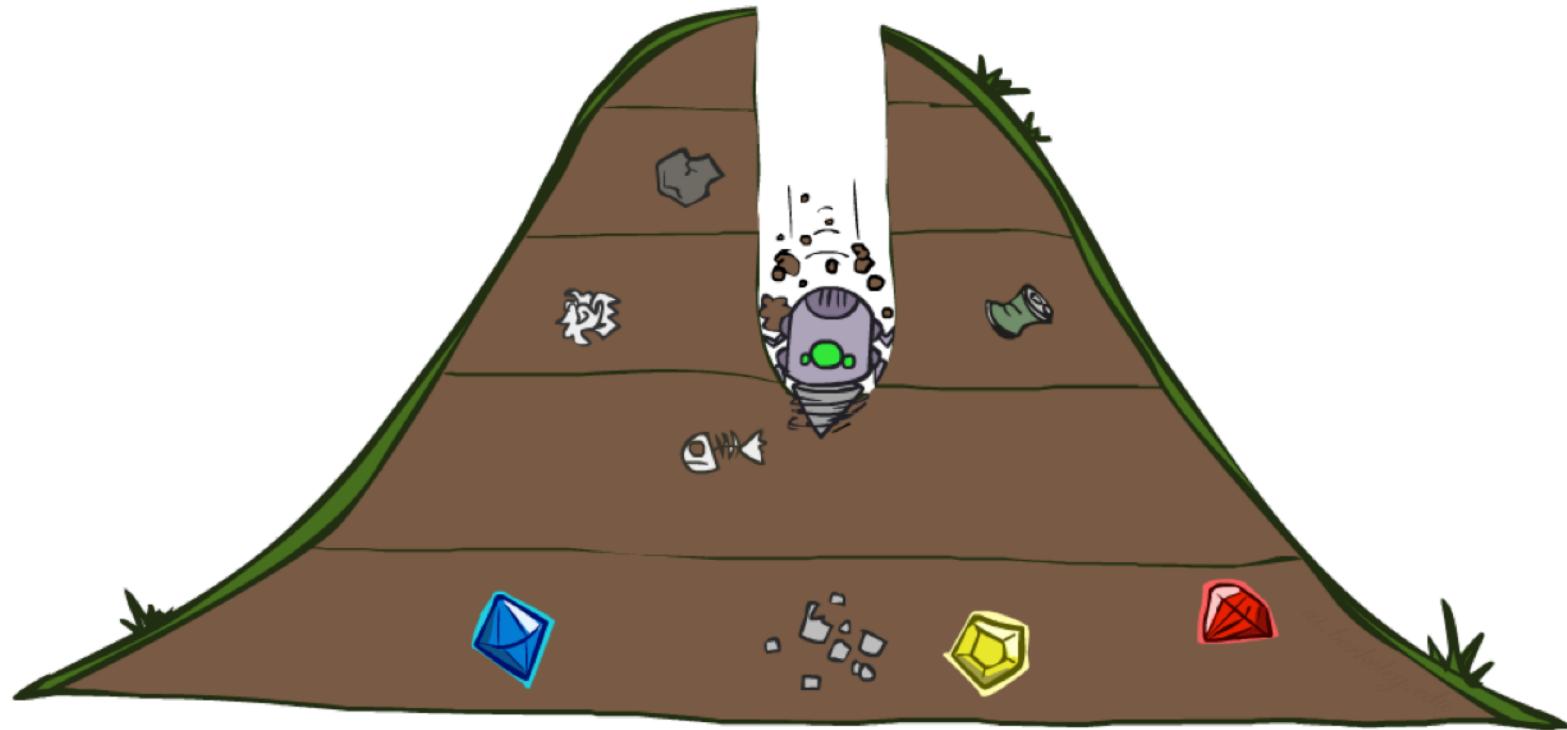
# اول-سطح یا اول عمق؟

---

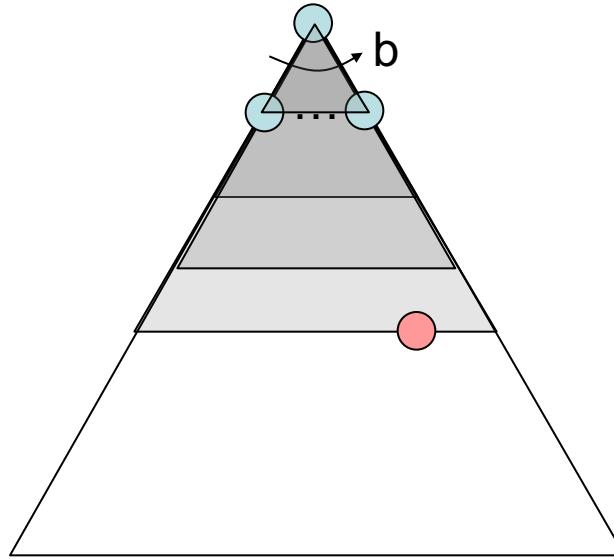


# جستجوی عمق محدود – depth limited

---



# عمیق‌کننده تکراری – Iterative Deepening



مزیت‌های اول-سطح و اول-عمق را ترکیب می‌کند

در عمق ۱، جستجوی اول-عمق انجام بده، اگر راه حلی و پیدا نشد...

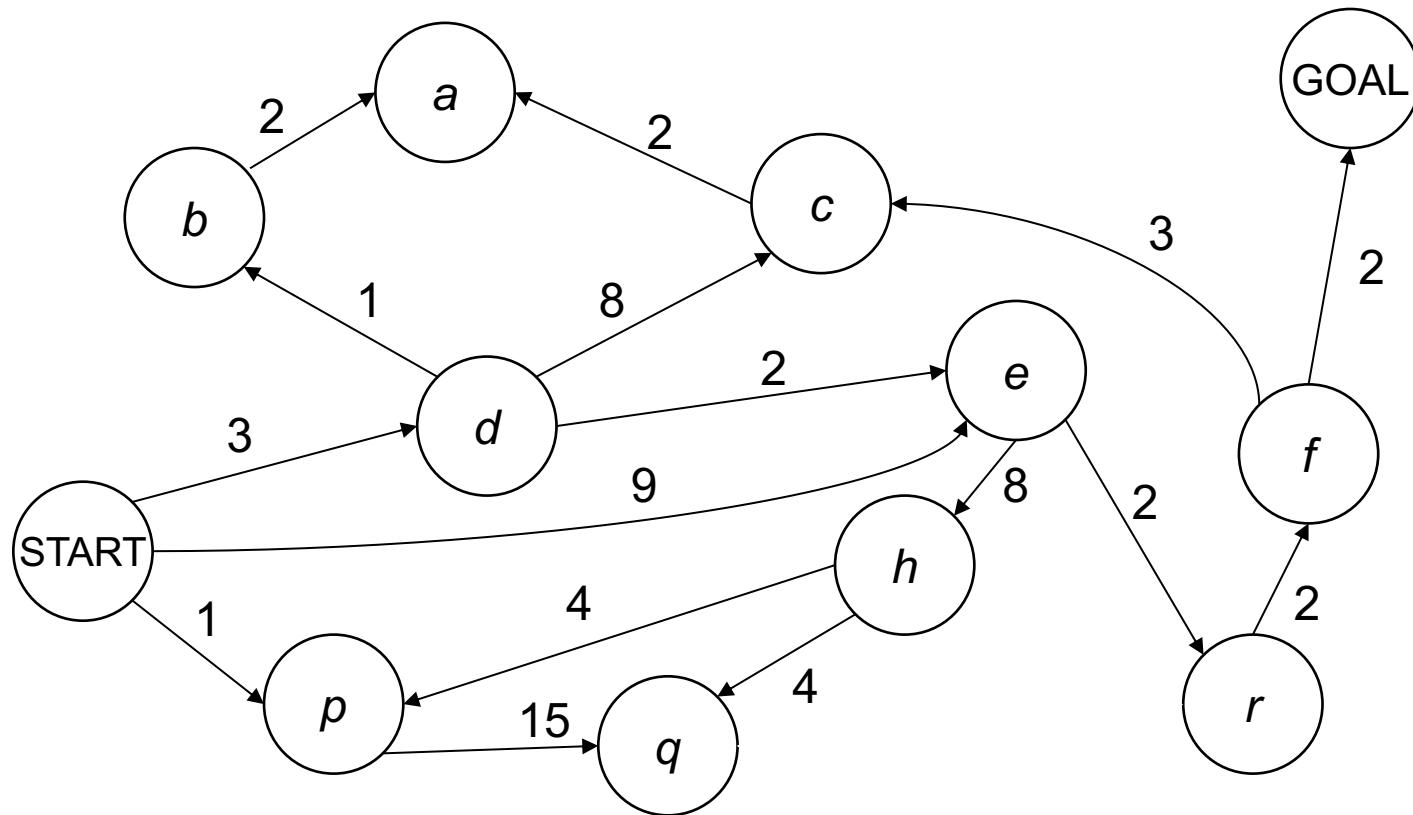
در عمق ۲، جستجوی اول-عمق انجام بده، اگر راه حلی و پیدا نشد...

در عمق ۳...

کار تکراری نمی‌کنیم؟

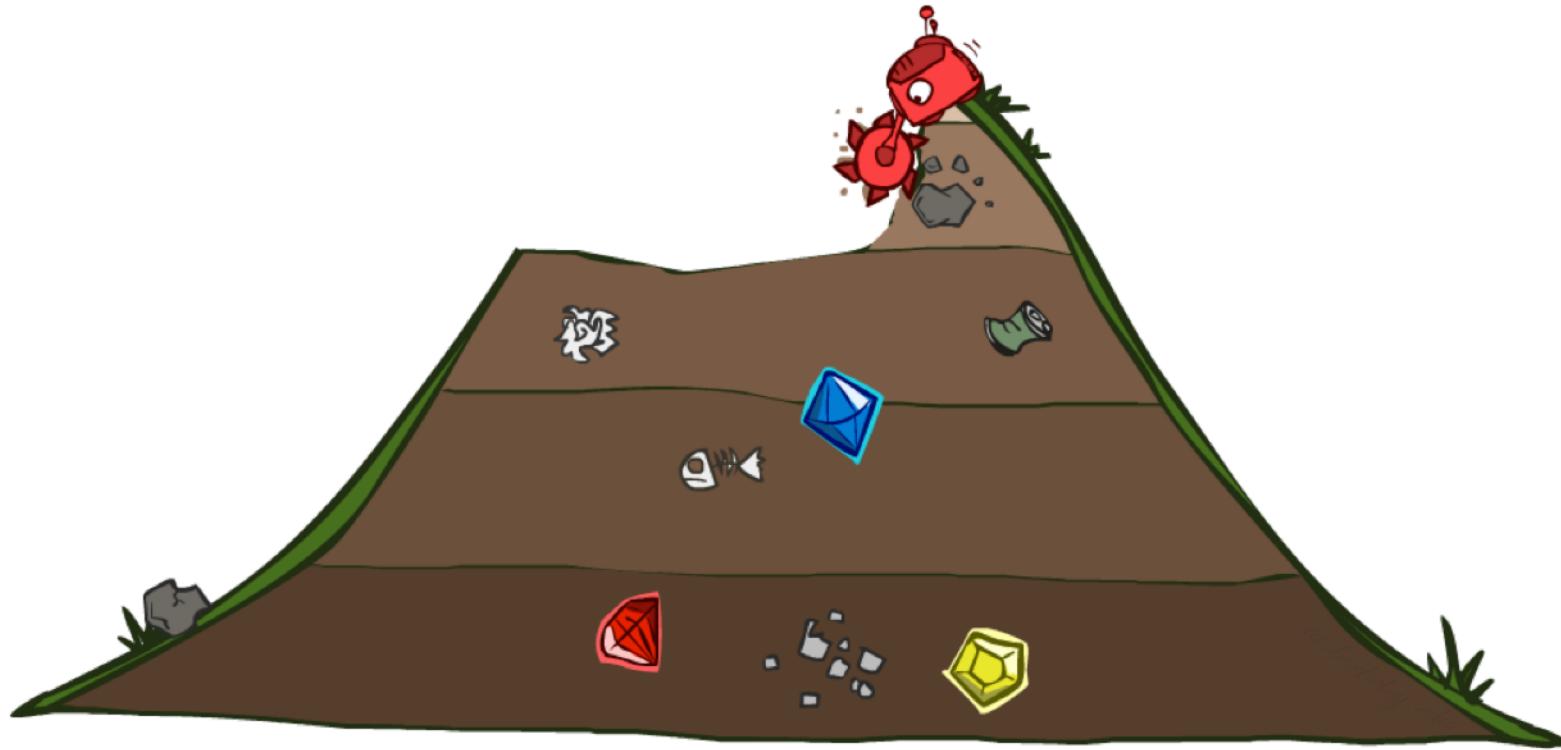
خیر، با توجه به اینکه در یک درخت اکثر گره‌ها برگ هستند

# جستجو با در نظر گرفتن هزینه

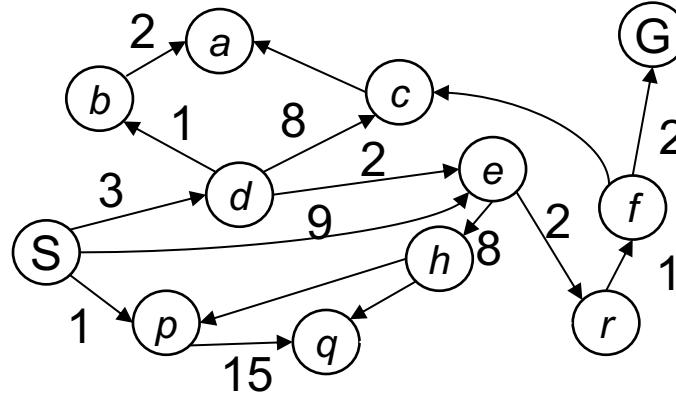


راه حلی که اول سطح پیدا می کند الزاما کم هزینه ترین راه حل نیست!

# جستجو با هزینه یکسان - Uniform Cost Search

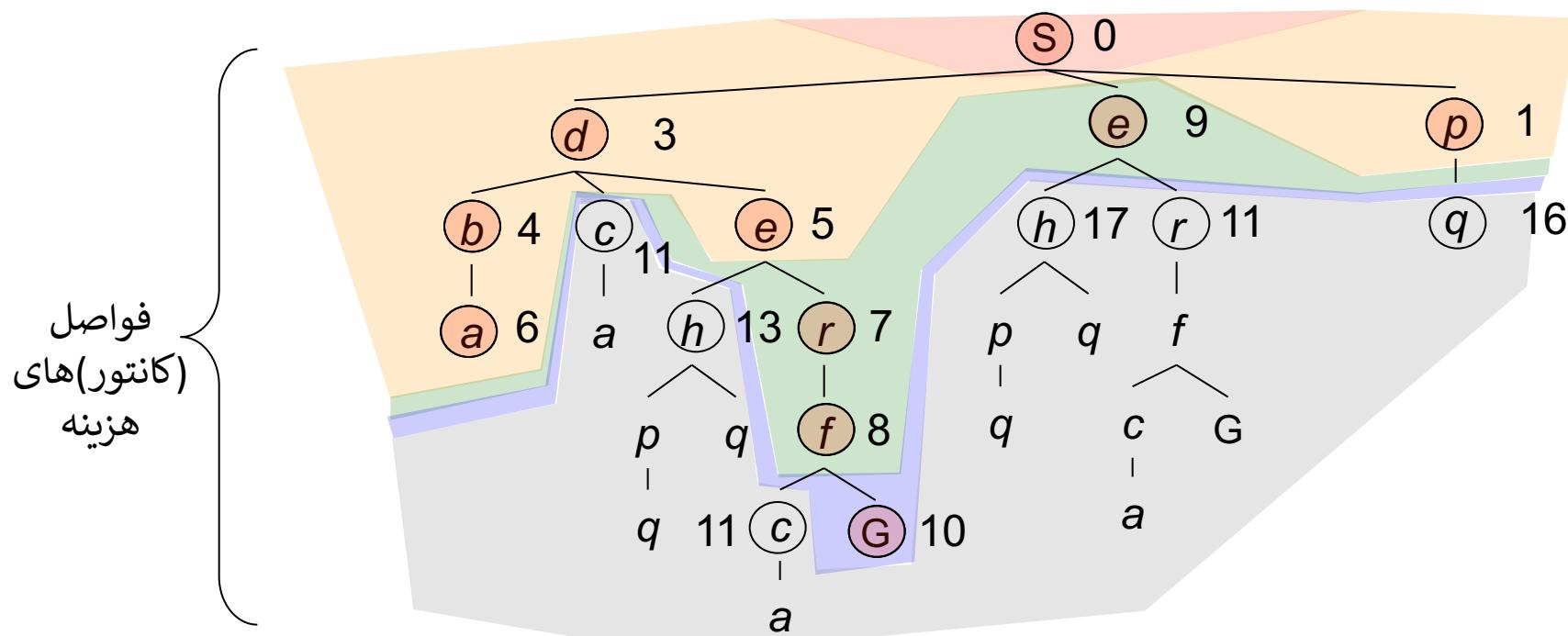


# جستجو با هزینه یکسان



استراتژی: کم‌هزینه‌ترین گره  
را بسط بد!

فرینج یک دنباله‌ی اولویت‌دار!



# ویژگی‌های الگوریتم هزینه-یکسان

## چه گره‌ای را اول بسط دهیم؟

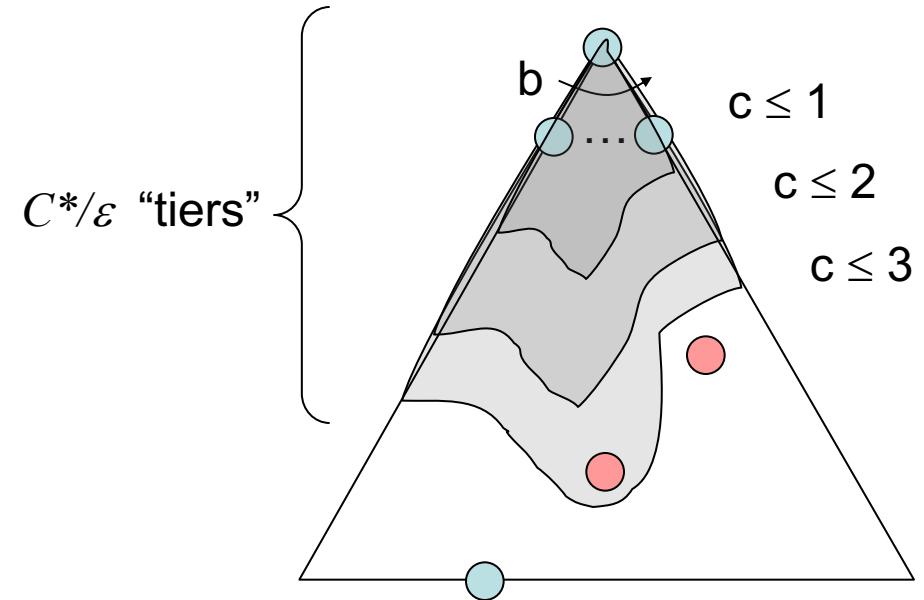
تمام گره‌هایی که هزینه کمتری از کم‌هزینه‌ترین راه حل دارند را بسط می‌دهد

هزینه‌ی این راه حل  $C^*$

حداقل هزینه‌ی هر تصمیم  $\varepsilon$

عمق تصمیم حدوداً  $\varepsilon/C^*$

زمان مورد نیاز:  $O(b^{C^*/\varepsilon})$



## حافظه؟

تقریباً تمام گره‌های سطح بالاتر -  $O(b^{C^*/\varepsilon})$

## کامل؟

بله، اگر هزینه‌ی راه حل متناهی و هزینه‌ی تصمیمات مثبت باشند!

## بهینه؟

بله (اثبات به زودی)

# الگوریتم هزینه یکسان



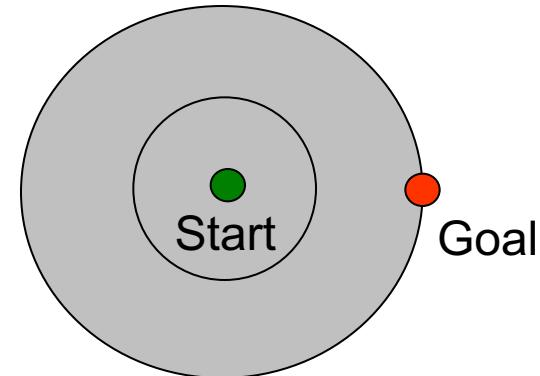
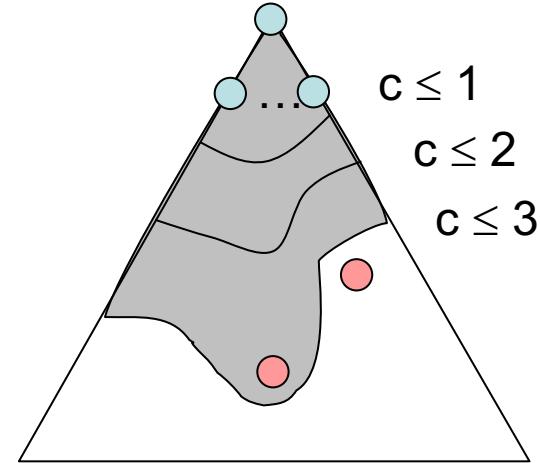
کامل و بهینه



تمامی جهات را بررسی می‌کند

هیچ آگاهی‌ای از اینکه راه حل کجاست ندارد!

به زودی این مشکل را رفع خواهیم کرد!



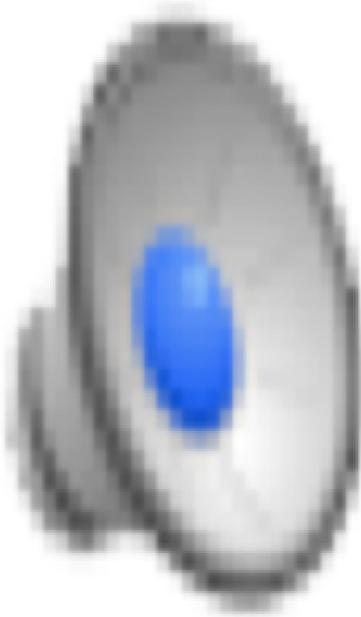
# جستجوی هزینه یکسان

---



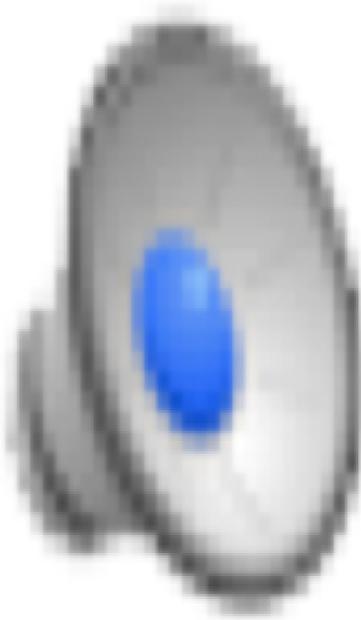
اول-سطح، اول-عمق یا هزینه-یکسان؟

---



اول-سطح، اول-عمق یا هزینه-یکسان؟

---



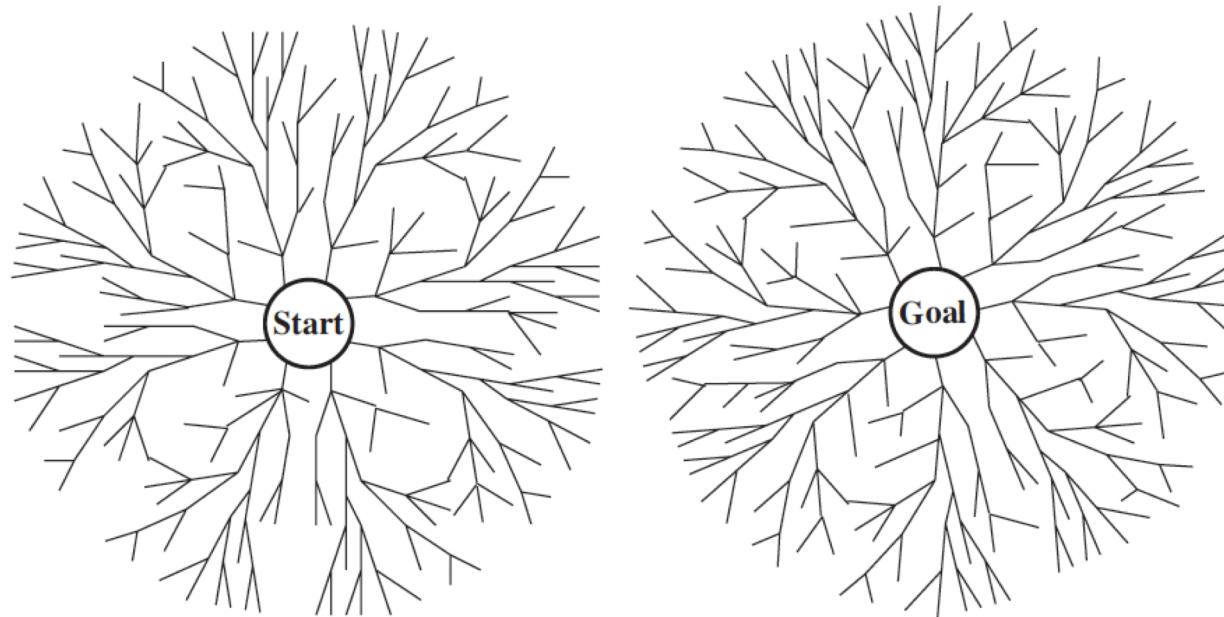
اول-سطح، اول-عمق یا هزینه-یکسان؟

---



# جستجوی دوطرفه – bidirectional

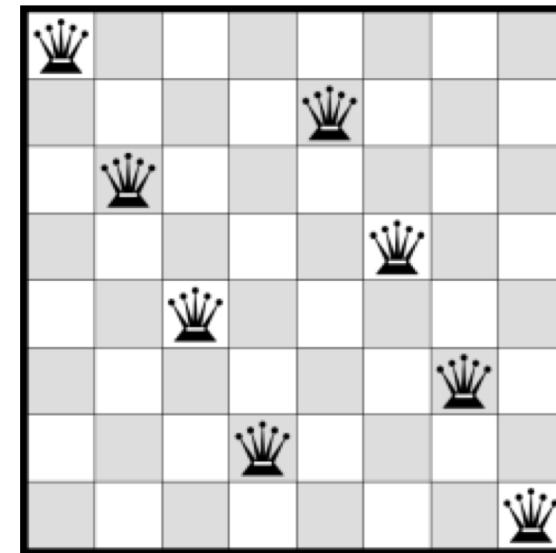
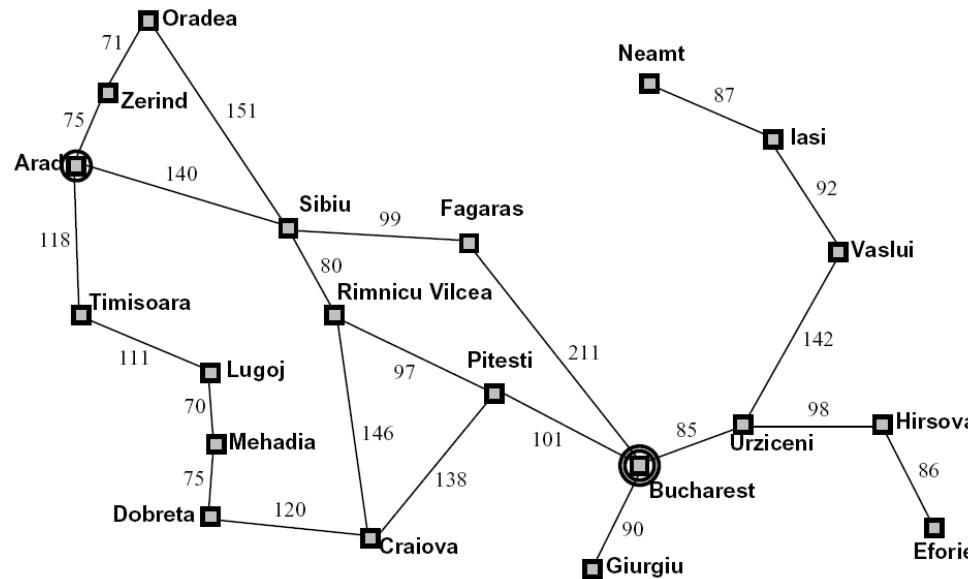
دو جستجوی همزمان، یکی forward، یکی backward  
به جای یک جستجوی  $b^d$  - دو جستجوی  $b^{d/2}$



# جستجوی دوطرفه - bidirectional

به متودی نیاز داریم که مراحل قبلی هر مرحله را بدهد

بسته به نوع مساله می‌تواند دشوار باشد



# مقایسه الگوریتم‌های جستجو

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Complete?	Yes <sup>a</sup>	Yes <sup>a,b</sup>	No	No	Yes <sup>a</sup>	Yes <sup>a,d</sup>
Time	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(b^m)$	$O(b^\ell)$	$O(b^d)$	$O(b^{d/2})$
Space	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(bm)$	$O(b\ell)$	$O(bd)$	$O(b^{d/2})$
Optimal?	Yes <sup>c</sup>	Yes	No	No	Yes <sup>c</sup>	Yes <sup>c,d</sup>

**a** - complete if  $b$  is finite

**b** - complete if step costs  $\geq \epsilon$  for positive  $\epsilon$

**c** - optimal if step costs are all identical

**d** - if both directions use breadth-first search

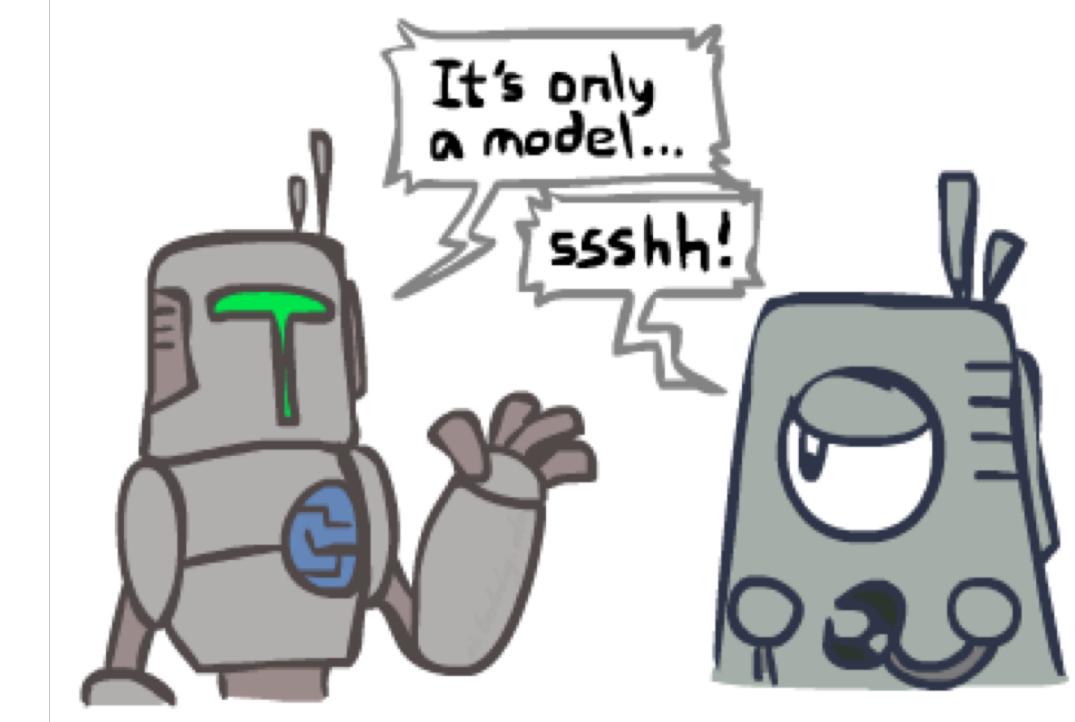
# جستجو - مدلسازی

ما در حقیقت در مدل محیط (جهان)  
جستجو می‌کنیم

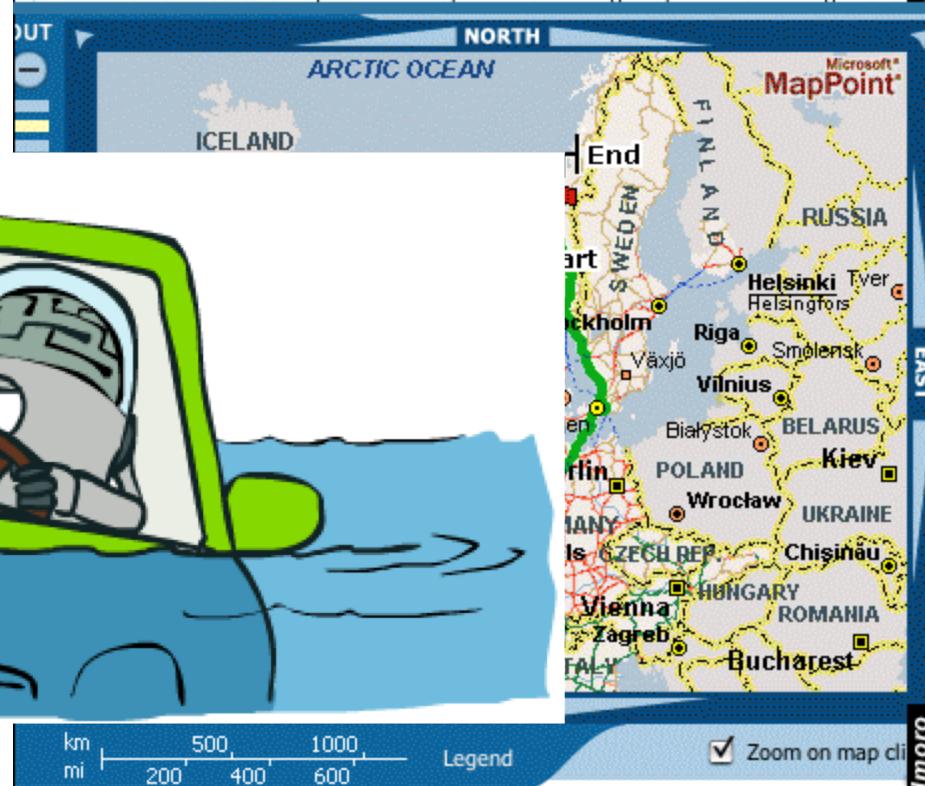
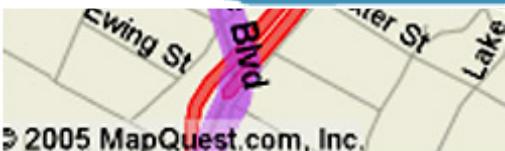
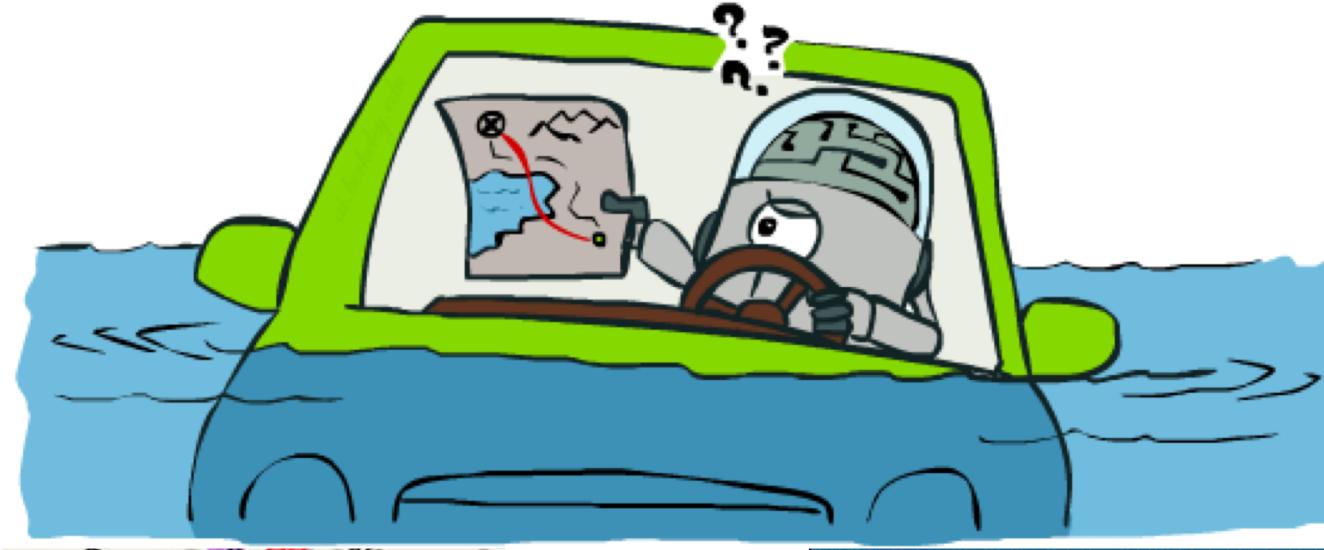
عامل، تصمیمات را در جهان واقعی چک  
نمی‌کند!

تمام تصمیم‌گیری در ذهن عامل و با  
شبیه‌سازی انجام می‌شود.

بنابراین، موثر بودن جستجو وابستگی زیادی  
به مدلسازی دارد



# سوتی‌های الگوریتم‌های جستجو!



# سؤال؟

---

