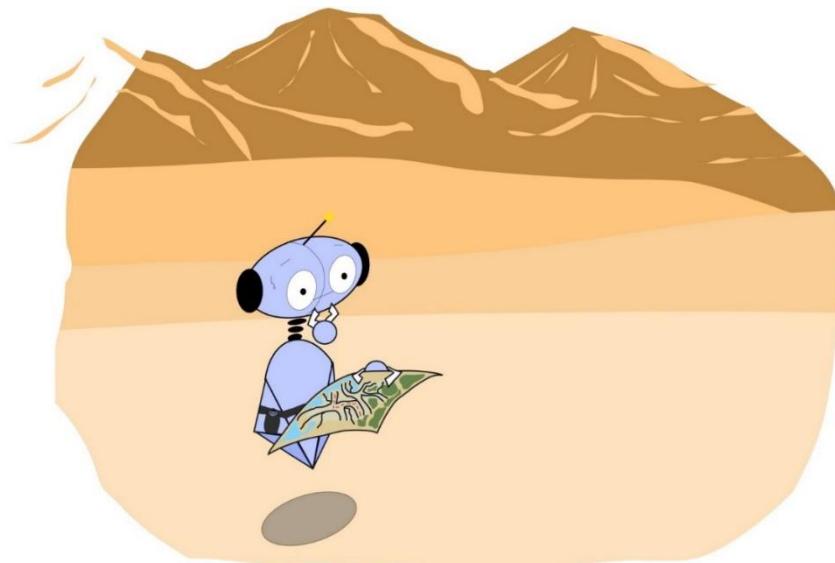


مبانی و کاربردهای هوش مصنوعی

جستجو ناآگاهانه (فصل 3.1 الی 3.4)

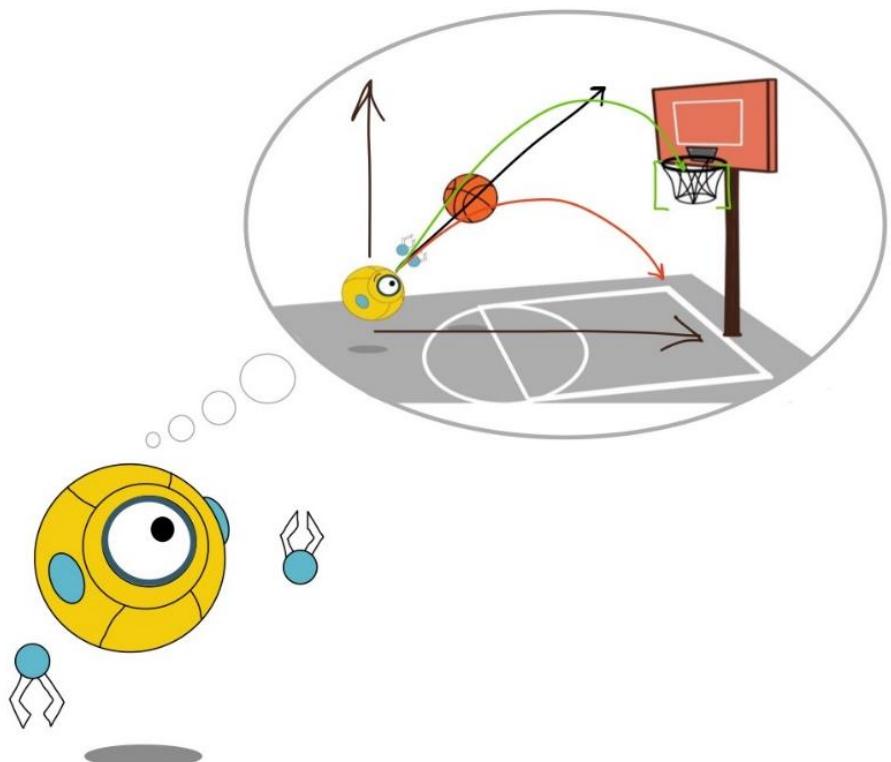


دکتر مهدی جوانمردی

دانشکده مهندسی کامپیوتر دانشگاه صنعتی امیرکبیر

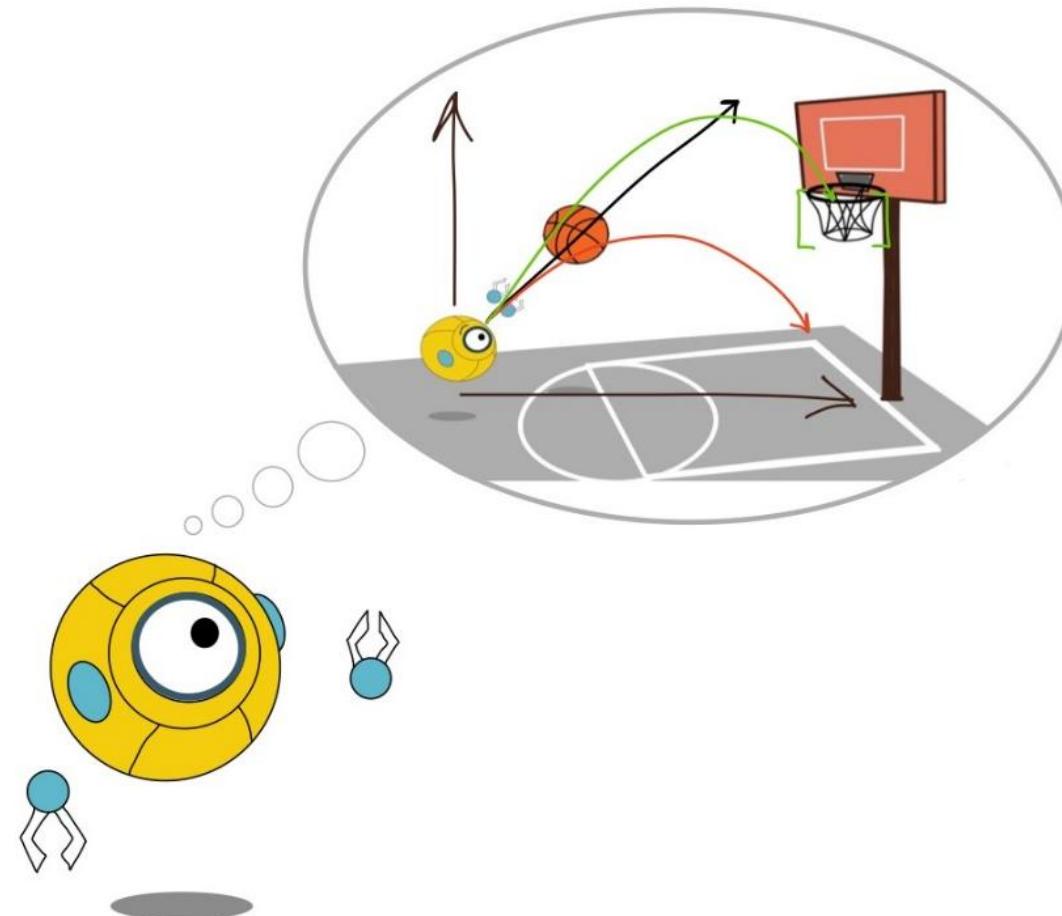


رئوس مطالب



- عامل‌هایی که برای آینده برنامه‌ریزی می‌کند
- مسائل جستجو
- روش‌های جستجوی ناآگاهانه
 - جستجوی اول عمق (Depth-first search)
- جستجوی اول سطح (Breadth-first search)
- جستجوی هزینه یکنواخت (Uniform-cost search)

عامل‌هایی که برنامه‌ریزی می‌کنند



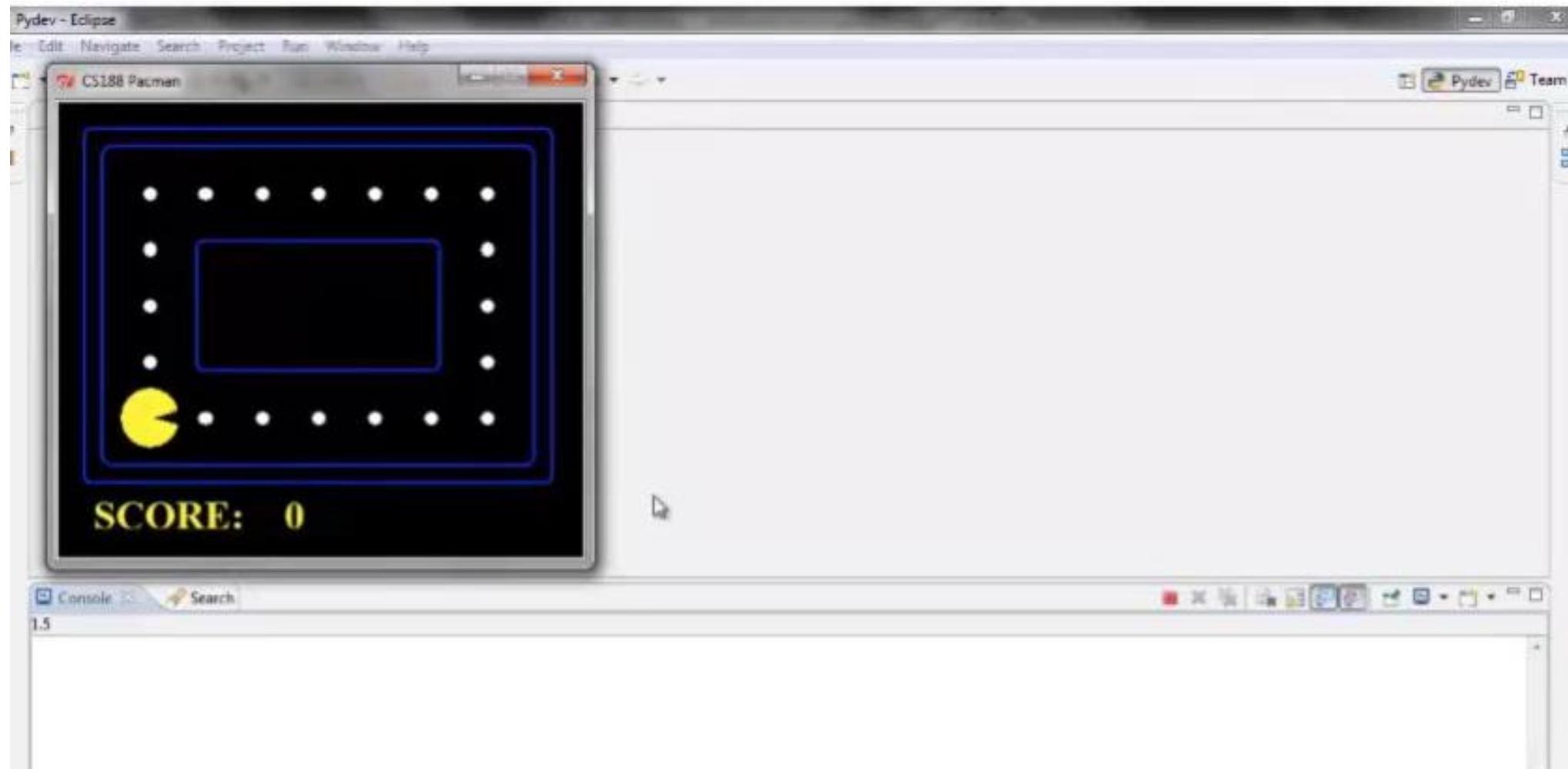
عامل‌های واکنشی

- عامل‌های واکنشی

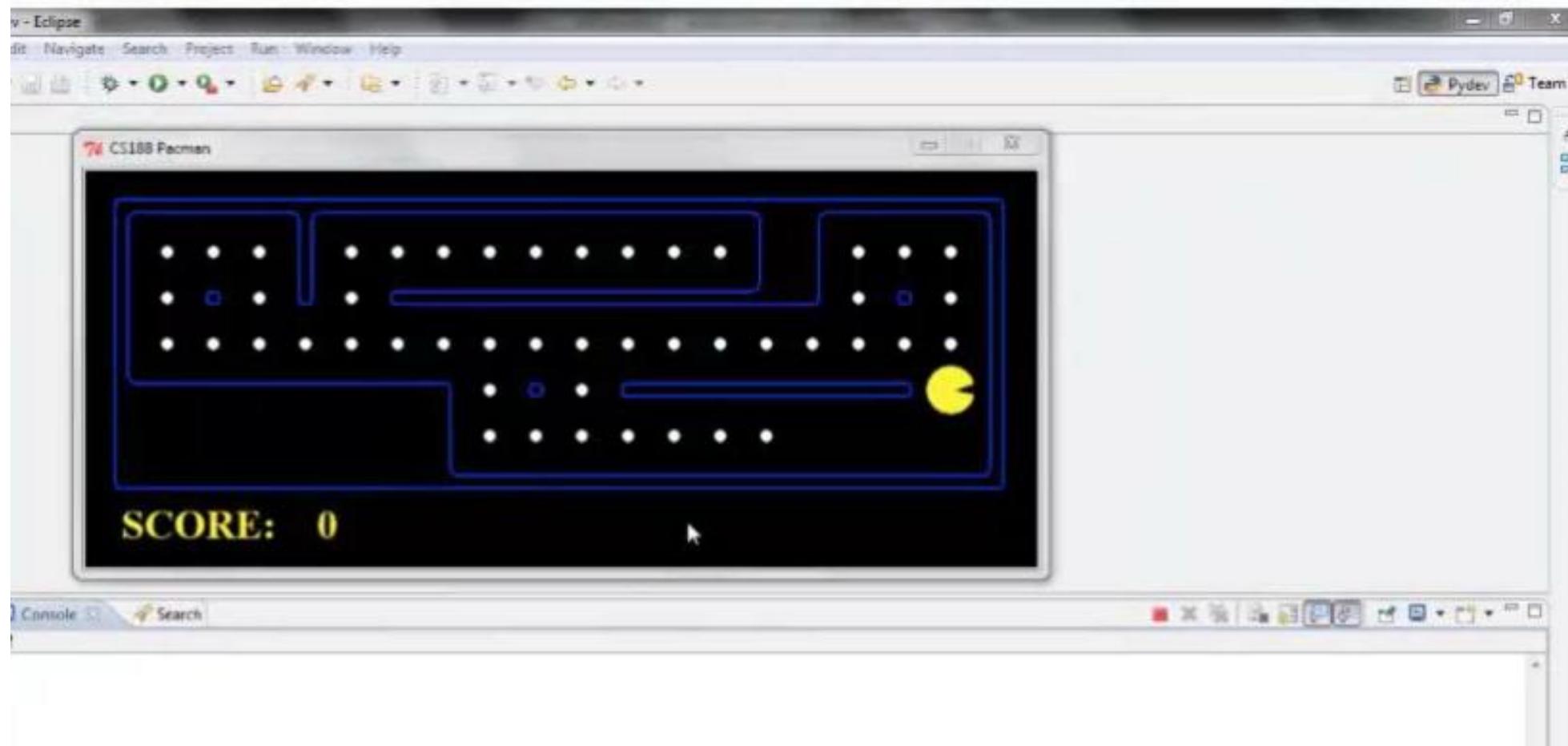
- انتخاب عمل بر اساس ادراک فعلی (و شاید حافظه)
- ممکن است حافظه داشته و یا مدلی از وضعیت فعلی جهان را نگهداری کند
- عواقب آتی اقداماتش را در نظر نمی‌گیرد
- در نظر می‌گیرد که جهان چگونه «**هست**»
- آیا یک عامل واکنشی می‌تواند عقلانی (منطقی) باشد؟



ویدیوی دموی واکنشی بهینه



ویدیوی دموی واکنشی ناموفق

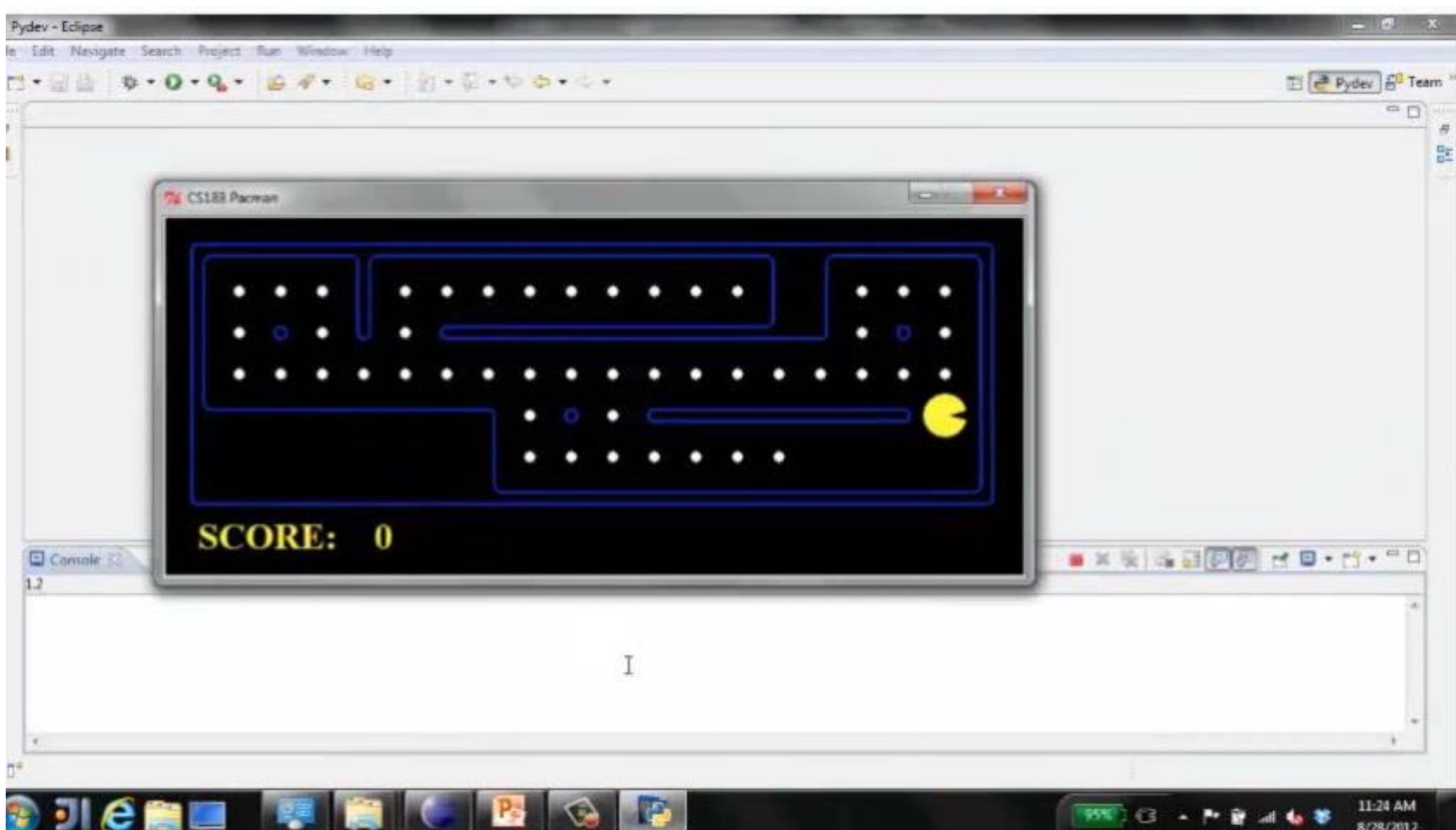


عامل‌های برنامه‌ریز

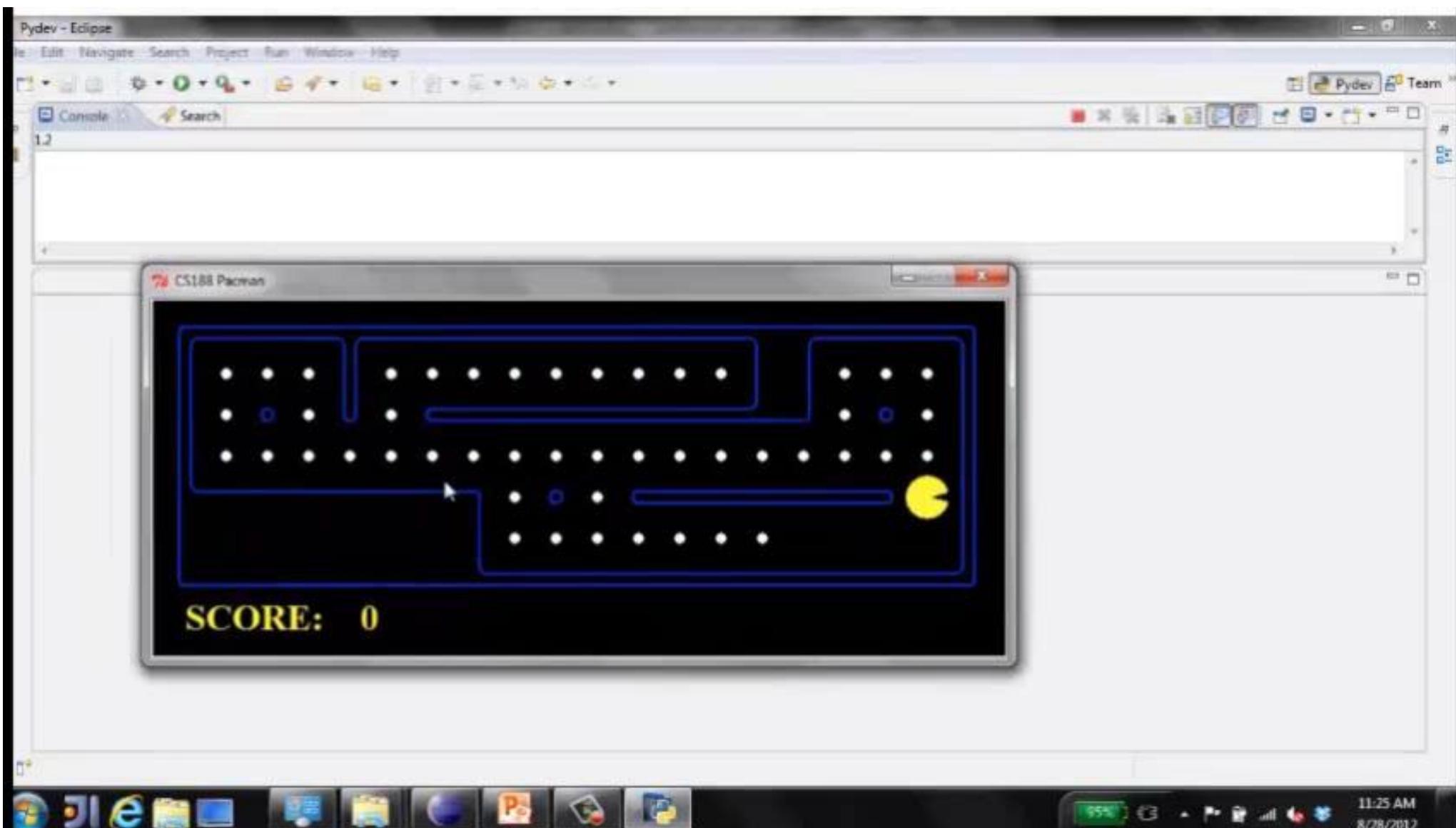


- عامل‌های برنامه‌ریز
- می‌پرسد "چه خواهد شد اگر (What if)
- تصمیم گیری بر اساس پیامدهای (فرضی) اقدامات
- باید مدلی از چگونگی تکامل جهان در واکنش به اعمال داشته باشد
- باید «هدف» را بصورت صریح فرموله کند (آزمون)
- در نظر می‌گیرد که جهان «چگونه خواهد بود» اگر...
- برنامه‌ریزی بهینه (optimal) در مقابل برنامه ریزی کامل (complete)
- برنامه‌ریزی (planning) در مقابل برنامه ریزی مجدد (replanning)

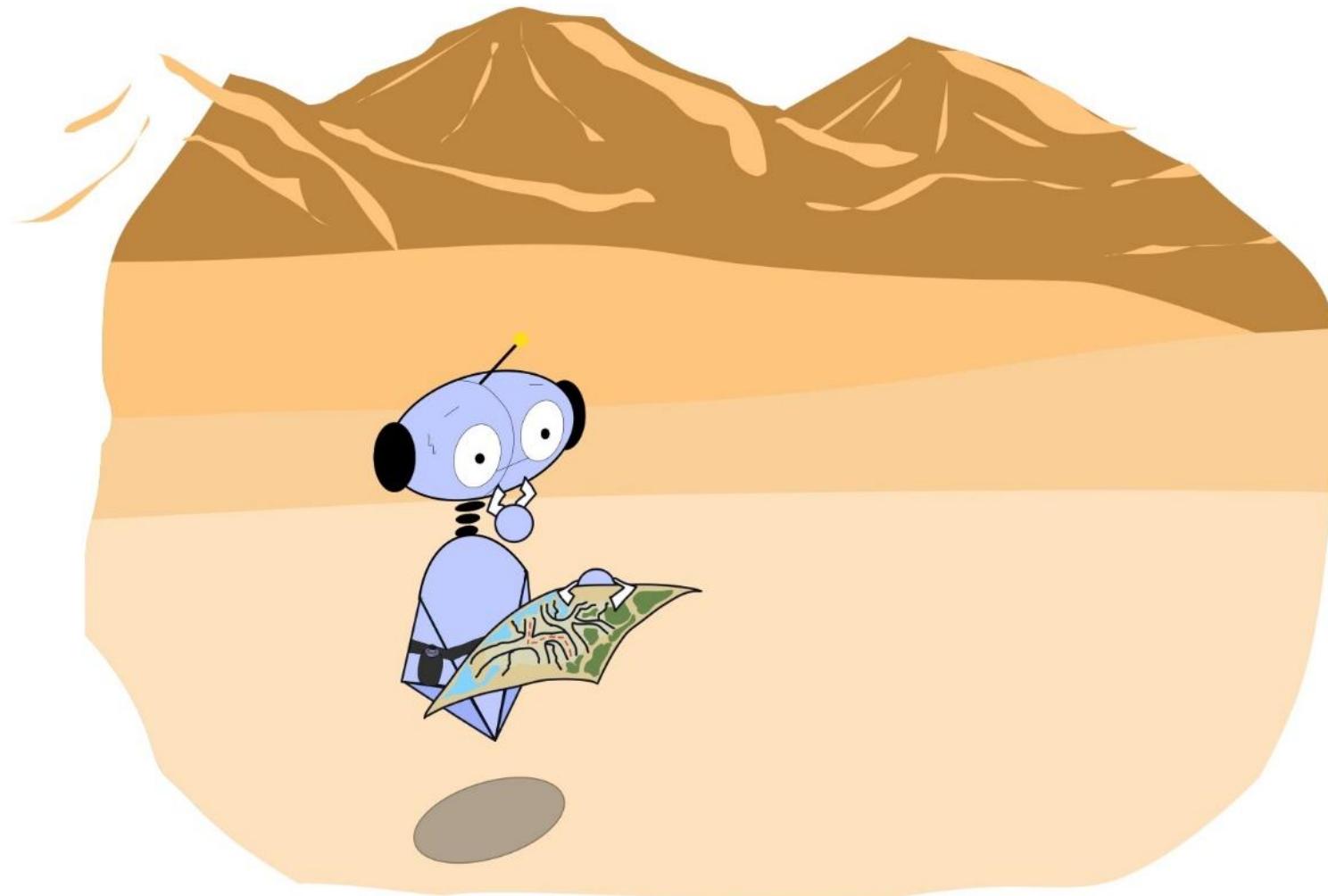
ویدیوی دموی پکمن برنامه‌ریزی مجدد



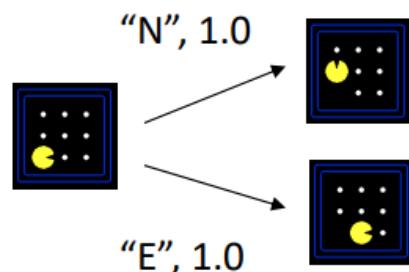
ویدیوی دموی پکمن مغز متفسکر



مسئله جستجو



مسائل جستجو (Search problems)

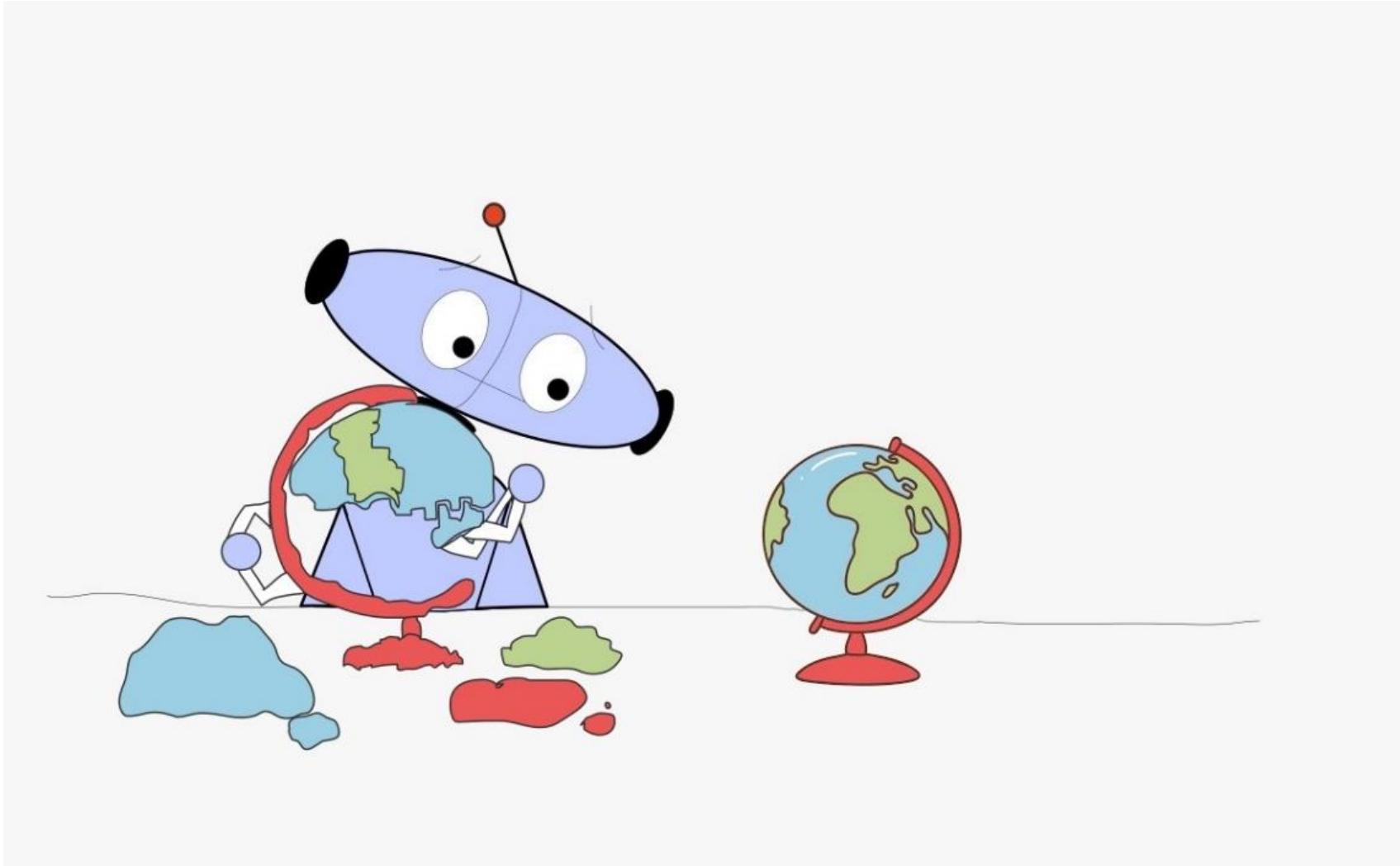


- یک راه حل (solution) مجموعه‌ای از اقدامات (برنامه‌ریزی) است که حالت شروع را به حالت هدف تبدیل می‌کند

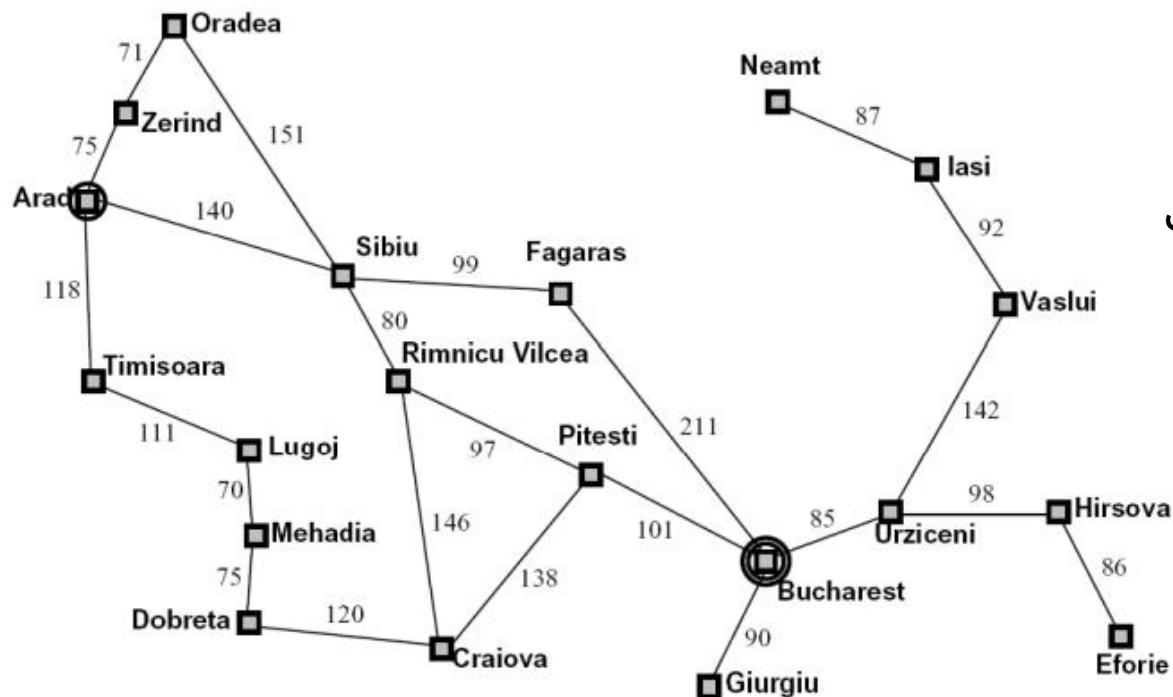
- یک مساله‌ی جستجو شامل موارد زیر است:
 - فضای حالت (state space)

- تابع پسین (successor function), (با اقدامات و هزینه‌ها)
- حالت شروع و آزمون هدف

مسائل جستجو مبتنی بر مدل هستند



مثال: مسافت در رومانی

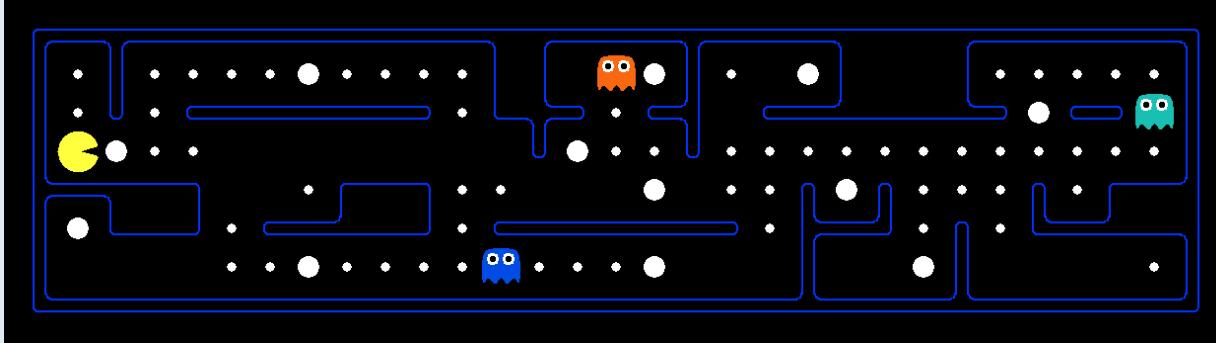


مسافت = هزینه

- فضای حالت
- شهرها
- تابع پسین
- جاده‌ها: به شهر مجاور بروید با هزینه = مسافت
- حالت شروع
- شهر آراد
- تست هدف
- آیا حالت == بخارست است؟
- راه حل؟

فضای حالت چیست؟

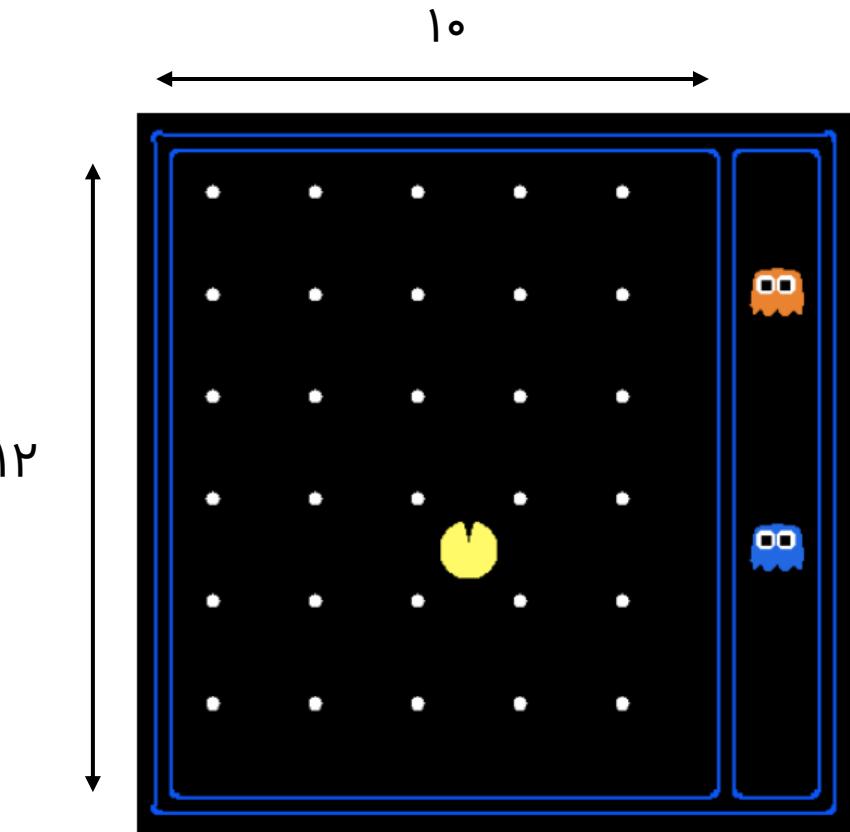
حالت جهان (world state) شامل همه‌ی جزئیات محیط است



حالت جستجو (search state) فقط جزئیات مورد نیاز برای برنامه‌ریزی را نگه می‌دارد (انتزاع)

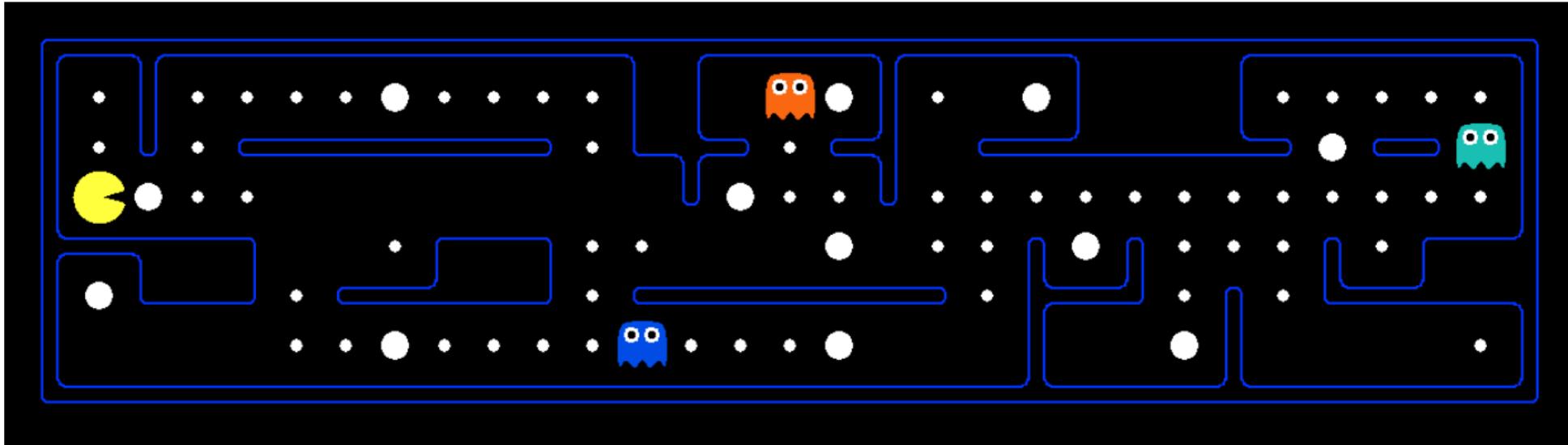
- مساله: خوردن همه نقطه‌ها
- حالات: $\{(Y, X)\}$, حالت بولی نقطه‌ها
- اقدامات: شمال، جنوب، شرق، غرب
- تابع پسین: بروزرسانی مکان و شاید بولی یک نقطه
- آزمون هدف: $\text{آزمون هدف: } \text{پایان} = (x, y)$ باشد
- مساله: مسیریابی
- حالات: مکان (X, Y)
- اقدامات: شمال، جنوب، شرق، غرب
- تابع پسین: فقط مکان به روزرسانی شود
- آزمون هدف: $\text{آزمون هدف: } \text{پایان} = (x, y)$ باشد

اندازه‌ی فضای حالت



- حالت جهان:
- مکان‌های عامل: ۱۲۰
- تعداد غذا: ۳۰
- مکان روح‌ها: ۱۲
- جهت عامل: شمال، جنوب، شرق، غرب
- چه تعداد حالت؟
$$30 \times 2$$
$$120 \times 2 \times 4$$
- حالت جهان؟
- حالت‌های مسیریابی؟ ۱۲۰
- حالت‌های خوردن همه نقطه‌ها؟
$$30 \times 2$$
$$120 \times 2$$

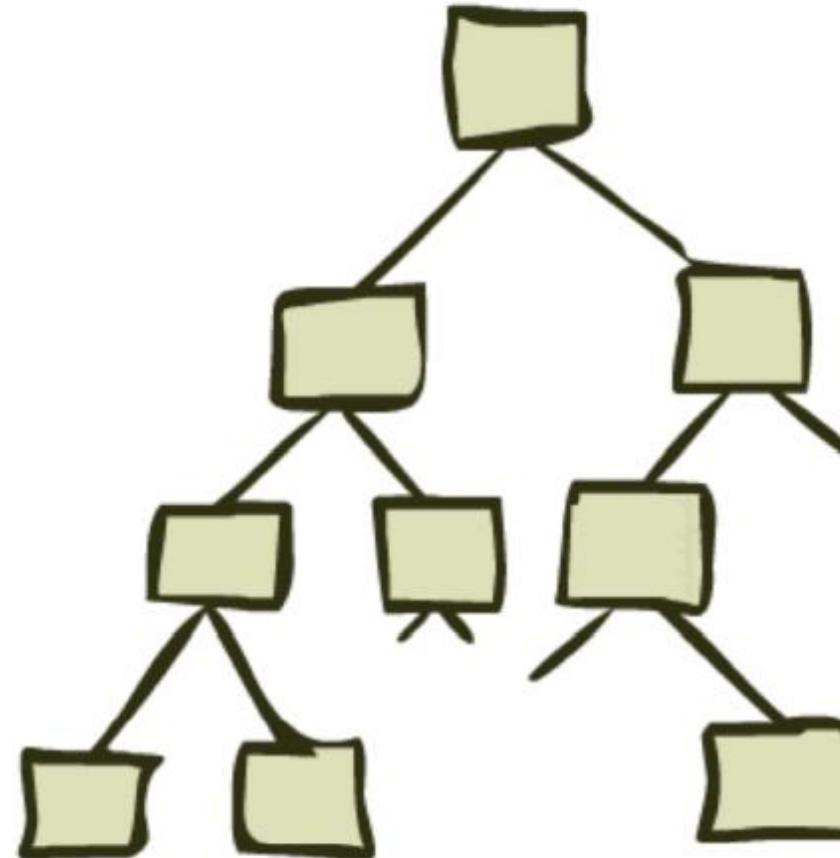
آزمونک: گذرگاه امن



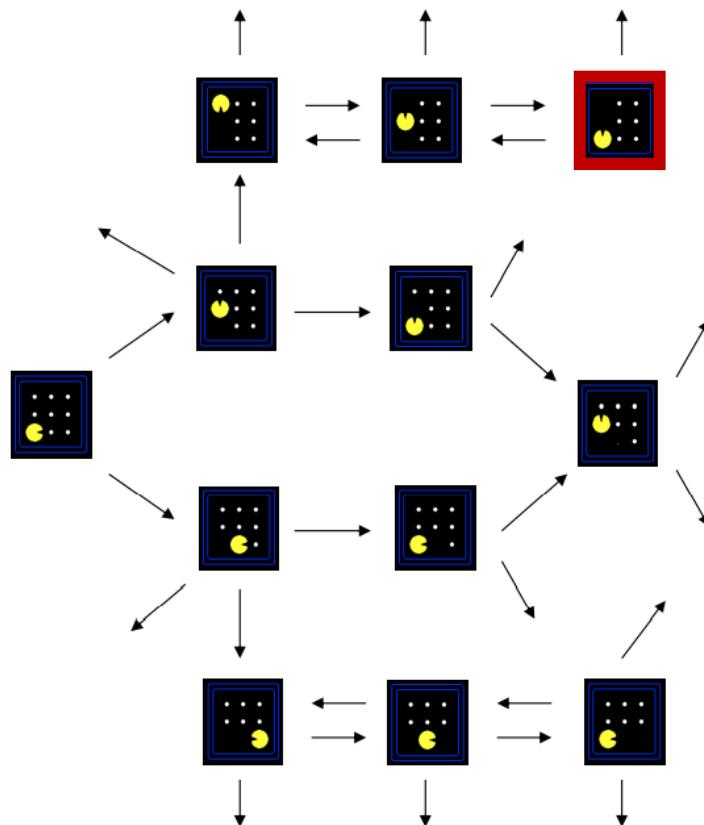
- مساله: تمام نقاط را بخورد در حالی که ارواح را همیشه ترساندهاید.
- فضای حالت باید چه چیزی را مشخص کند؟

(موقعیت عامل، بولی‌های نقطه‌ای، بولی‌های قرص قدرت، زمان ترس باقیمانده)

گراف فضای حالات و درخت جستجو

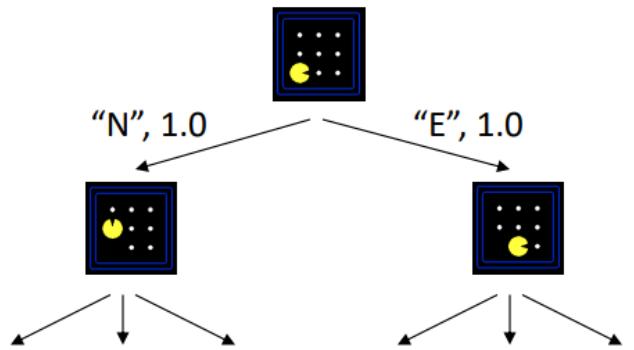


گراف فضای حالت



- گراف فضای حالت: نمایش ریاضیاتی یک مسئله جستجو
- گره‌ها پیکربندی جهان را نشان میدهند (خلاصه شده)
- یال‌ها نشان دهنده پسین‌ها هستند (نتایج اعمال)
- آزمون هدف مجموعه‌ای از گره‌های هدف است (شاید فقط یک گره باشد)
- در گراف فضای حالت، هر حالت فقط یک بار رخ می‌دهد!
- ما به ندرت می‌توانیم این نمودار را کامل در حافظه بسازیم (خیلی بزرگ است)، اما این یک ایده قابل استفاده است.
- در گراف جستجو با انتخاب یک گره نمی‌توان مسیر دستیابی یکتا از گره ریشه به آن گره را تعیین کرد

درخت‌های جستجو



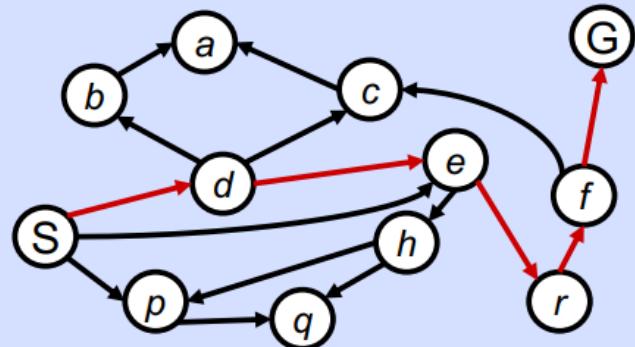
این حالت کنونی / شروع است.

آینده‌های محتمل

- درخت جستجو:
- درخت «چه می‌شد اگر(what if)» از برنامه‌ها و نتایج آنها
- حالت شروع گره ریشه است
- فرزندان نشانده‌نده پسین‌ها
- گره‌ها حالت‌ها را نشان می‌دهند، اما مطابق با **برنامه**‌هایی هستند که به آن حالت‌ها دست می‌یابند
- برای اکثر مسائل، ما هرگز نمی‌توانیم کل درخت را بسازیم (حتی از گراف جستجو هم بزرگ‌تر است!)

گراف فضای حالت در مقابل درخت‌های جستجو

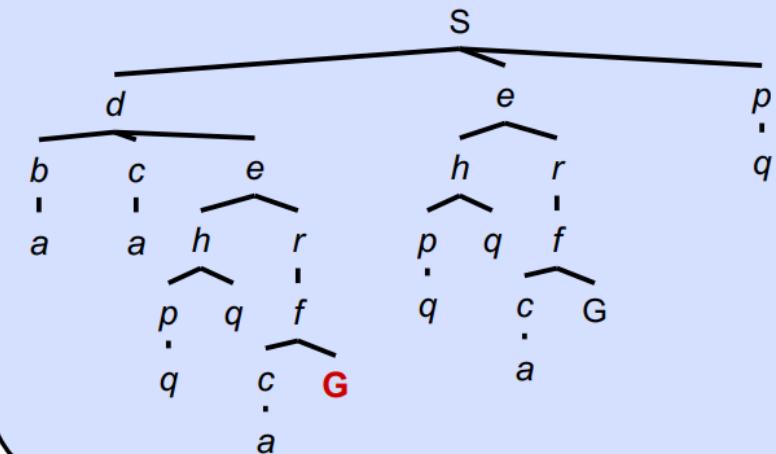
State Space Graph



هر گره در درخت جستجو
یک مسیر کامل در نمودار
فضای حالت است.

ما هر دو را بنا به تقاضا و تا
حد امکان کوچکتر می‌سازیم.

Search Tree

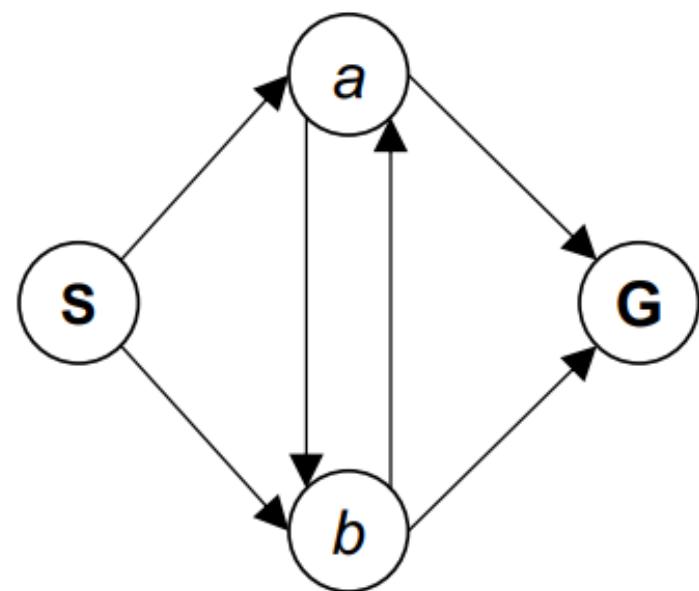


گراف فضای حالت در مقابل درخت‌های جستجو

اندازه درخت جستجو چقدر است؟
(با فرض شروع از حالت S)

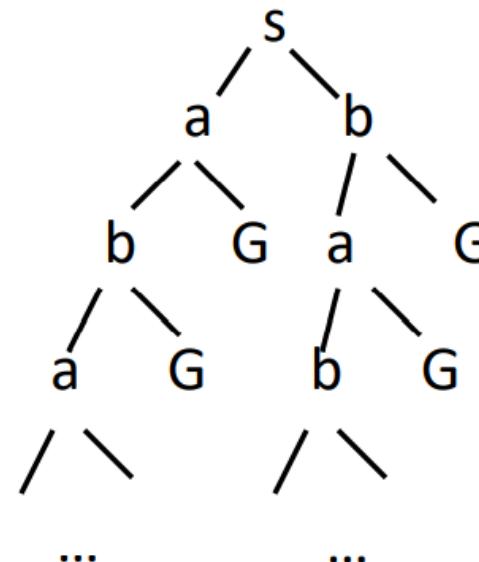


این گراف 4 حالت را در نظر بگیرید.

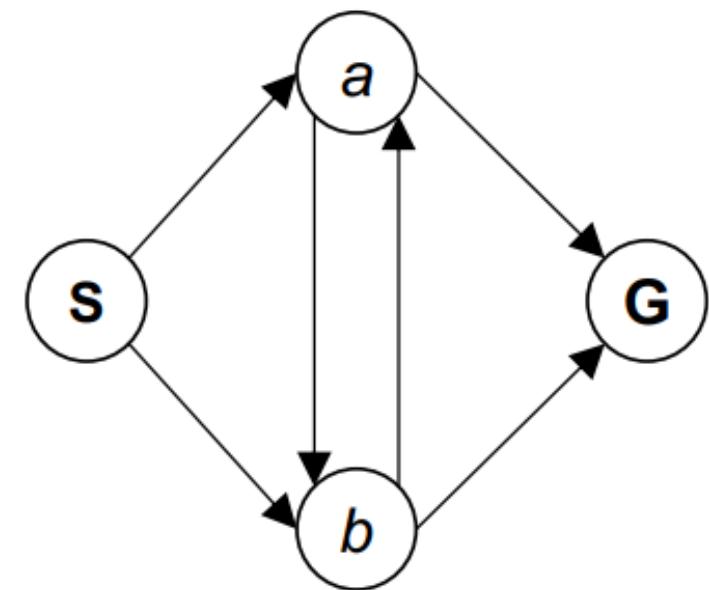


گراف فضای حالت در مقابل درخت‌های جستجو

اندازه درخت جستجو چقدر است؟
(با فرض شروع از حالت S)

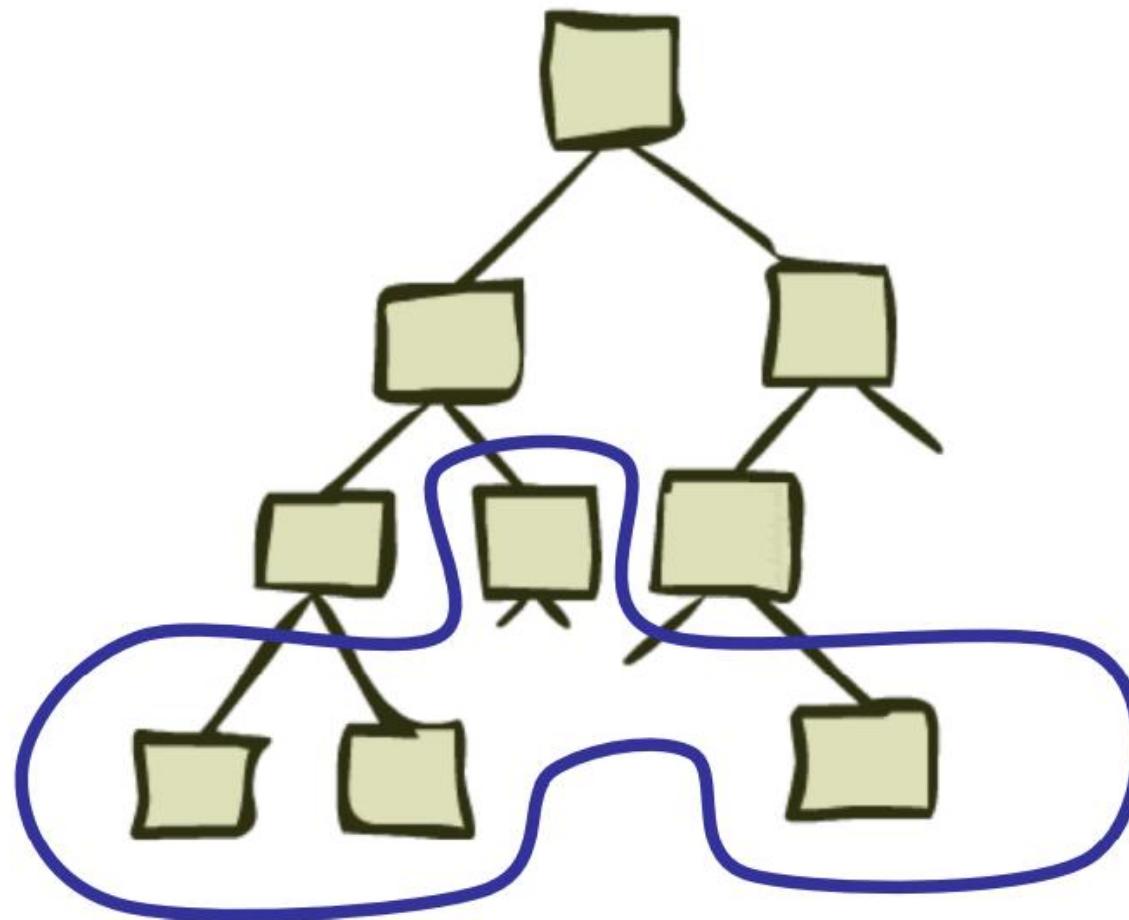


این گراف 4 حالت را در نظر بگیرید.

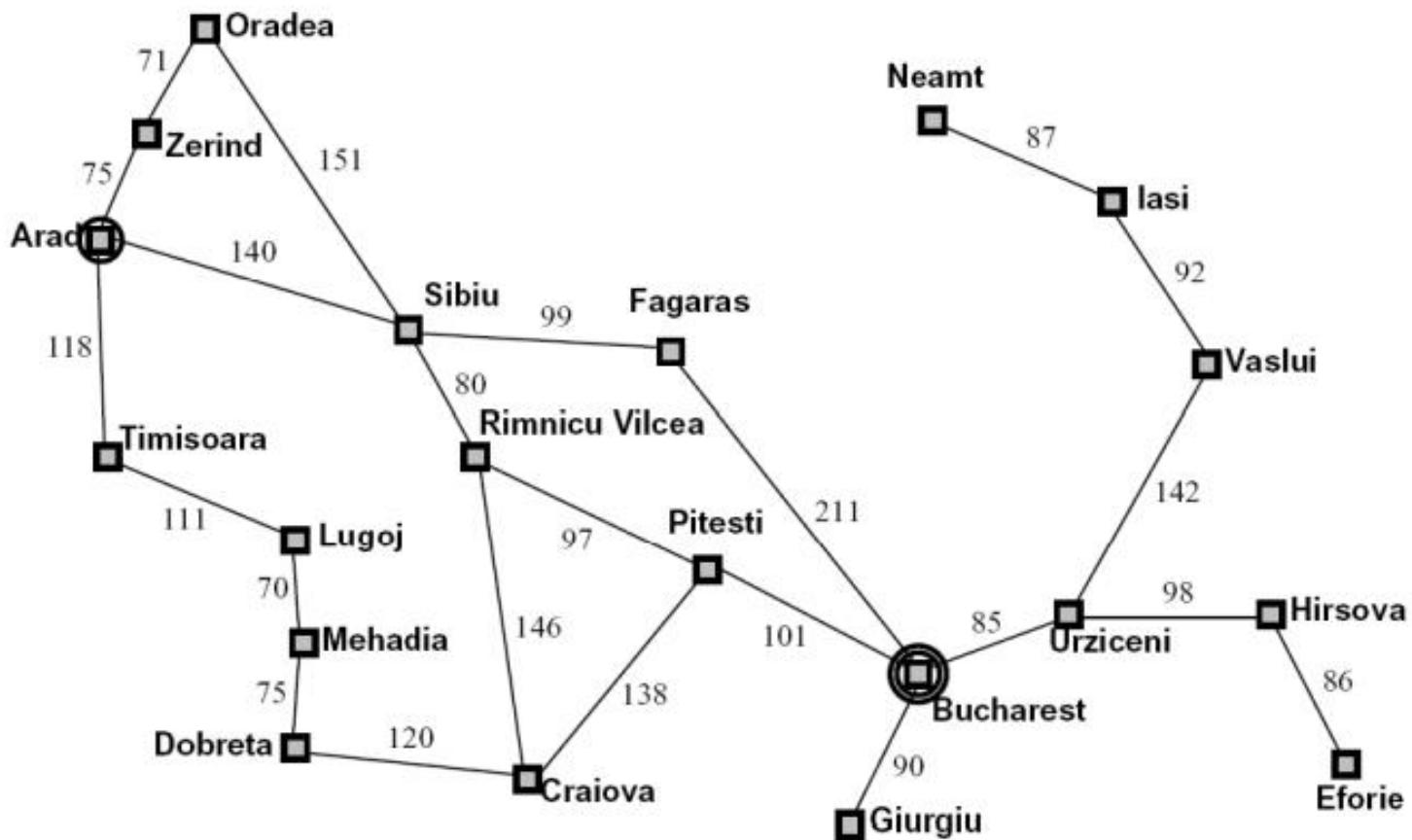


نکته مهم: تعداد زیادی ساختار تکراری در درخت جستجو!

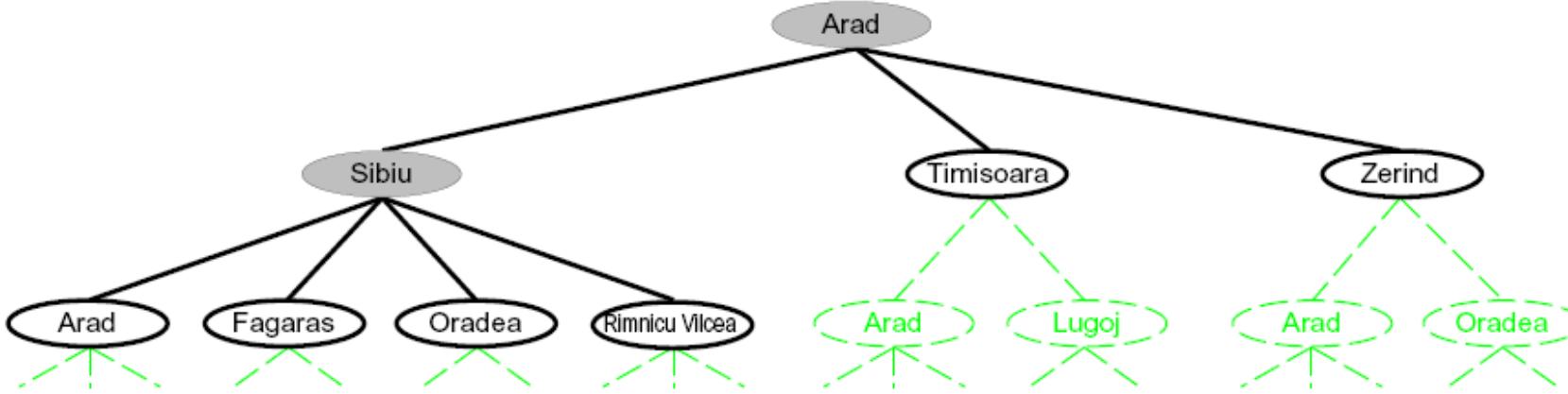
جستجوی درختی



مثال جستجو: رومانی



جستجو با درخت جستجو



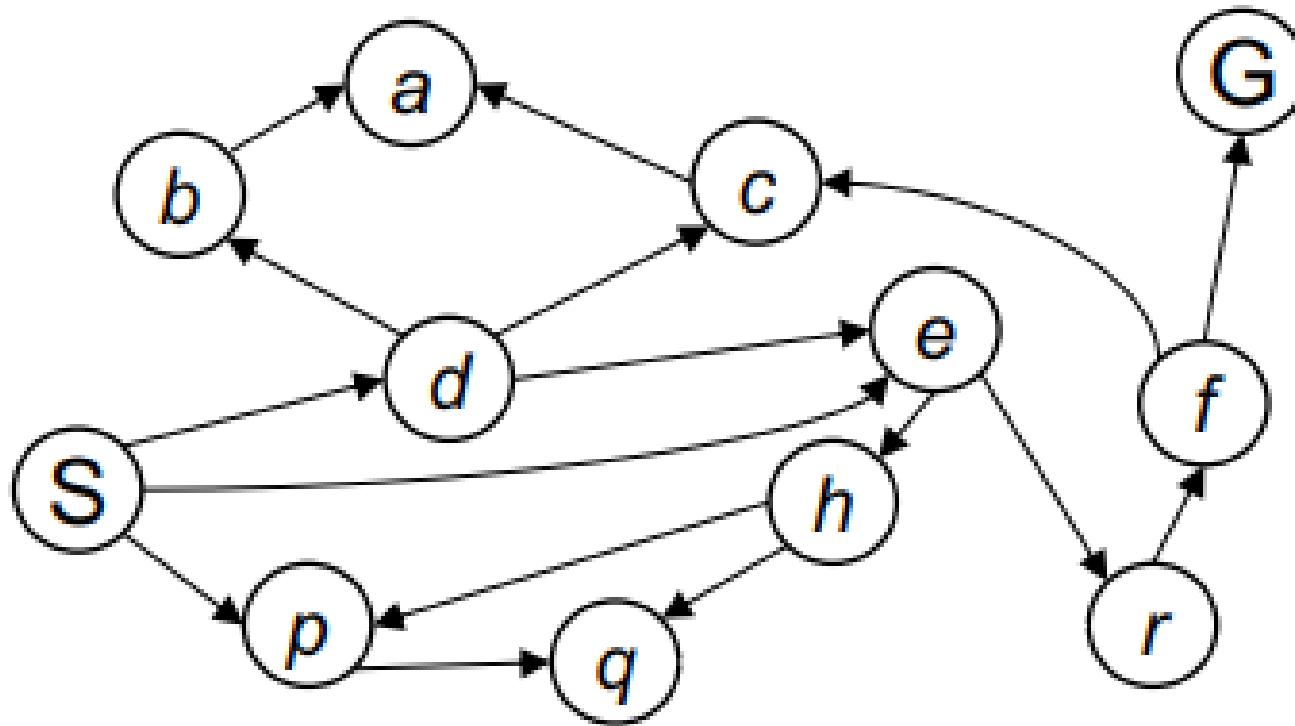
- جستجو:
- بسط برنامه ریزی های بالقوه (گره های درخت)
- **لیست حاشیه (fringe)** از برنامه ریزی های جزئی در دست بررسی را در حافظه نگهدارید
- سعی کنید تا حد امکان تعداد کمتری از گره های درختی را بسط دهید

جستجو با درخت جستجو

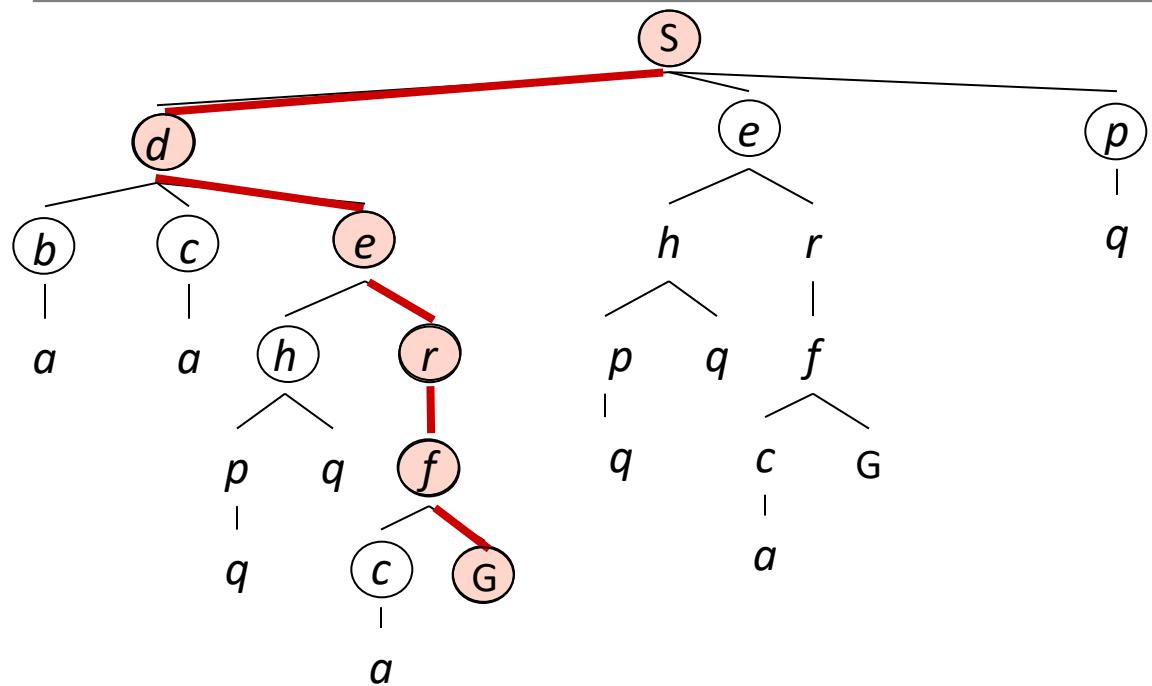
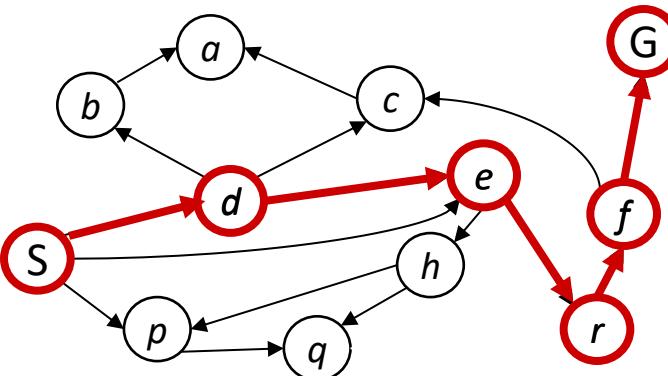
```
function TREE-SEARCH(problem, strategy) returns a solution, or failure
    initialize the search tree using the initial state of problem
    loop do
        if there are no candidates for expansion then return failure
        choose a leaf node for expansion according to strategy
        if the node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

- ایده‌های مهم:
- لیست حاشیه
- بسط دهنده (expansion)
- استراتژی بسط دهنده (strategy)
- سوال اصلی: کدام گره‌های لیست حاشیه را بسط (expand) دهیم؟

مثال: جستجو درختی

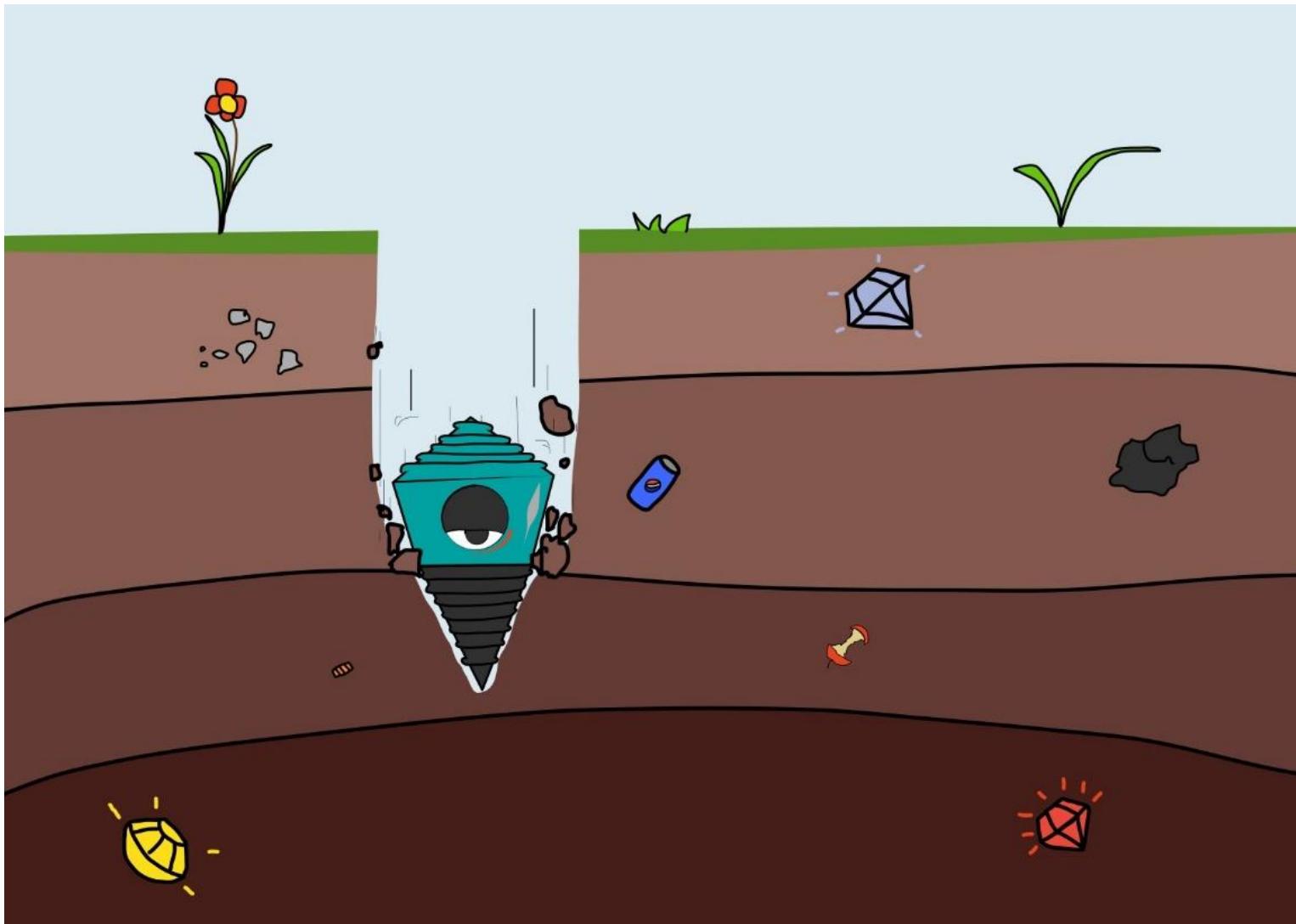


مثال: جستجو درختی

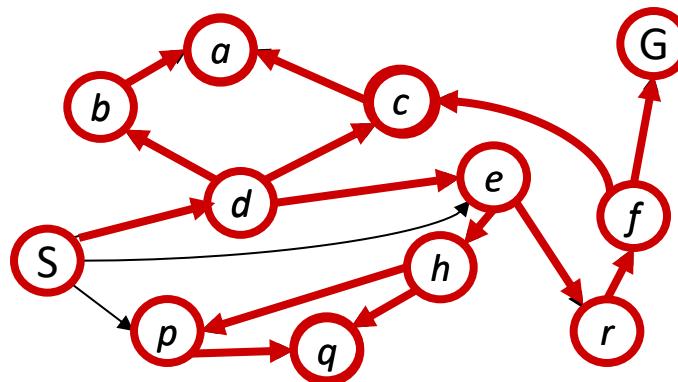


s
s → d
s → e
s → p
s → d → b
s → d → c
s → d → e
s → d → e → h
s → d → e → r
s → d → e → r → f
s → d → e → r → f → c
s → d → e → r → f → G

جستجوی اول عمق (DFS)

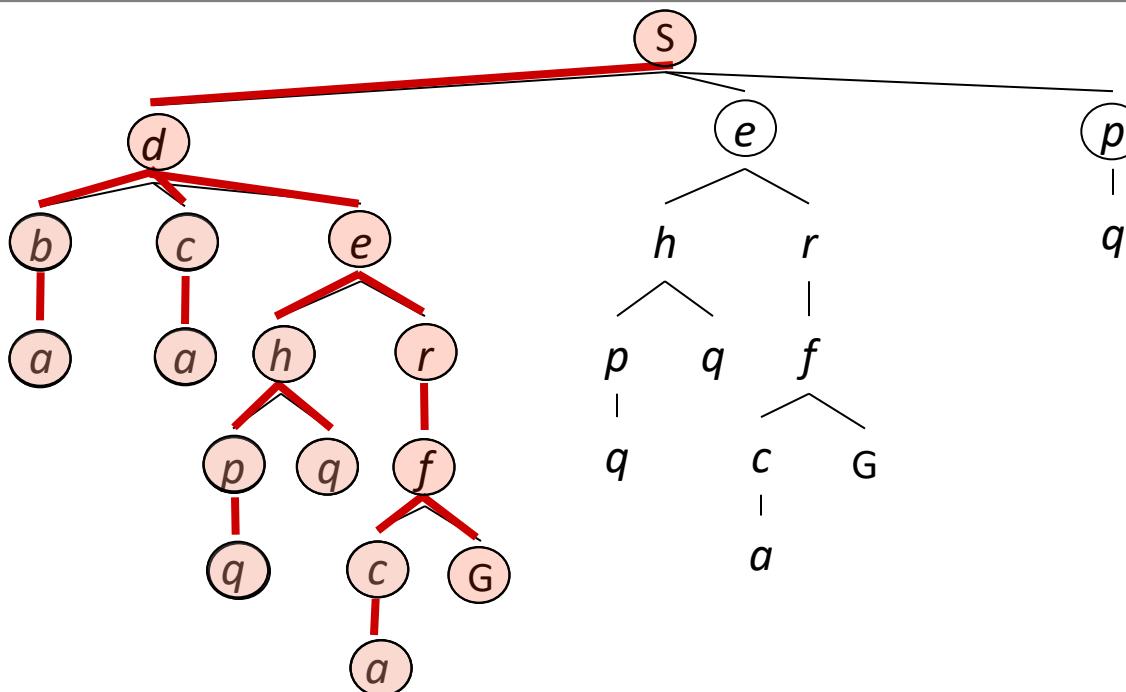


جستجوی اول عمق (DFS)

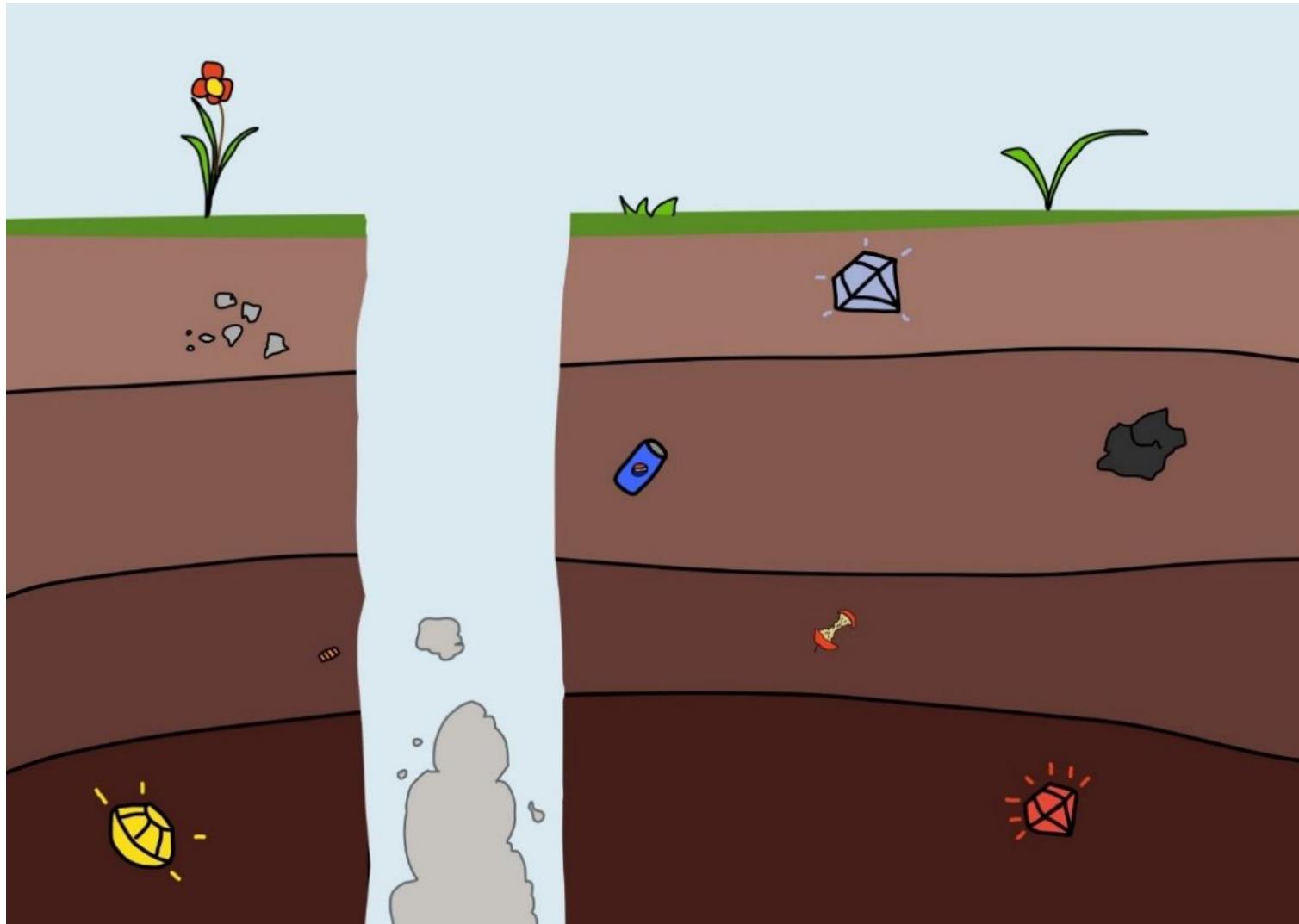


استراتژی:
ابتدا عمیق‌ترین گره را بسط دهید.

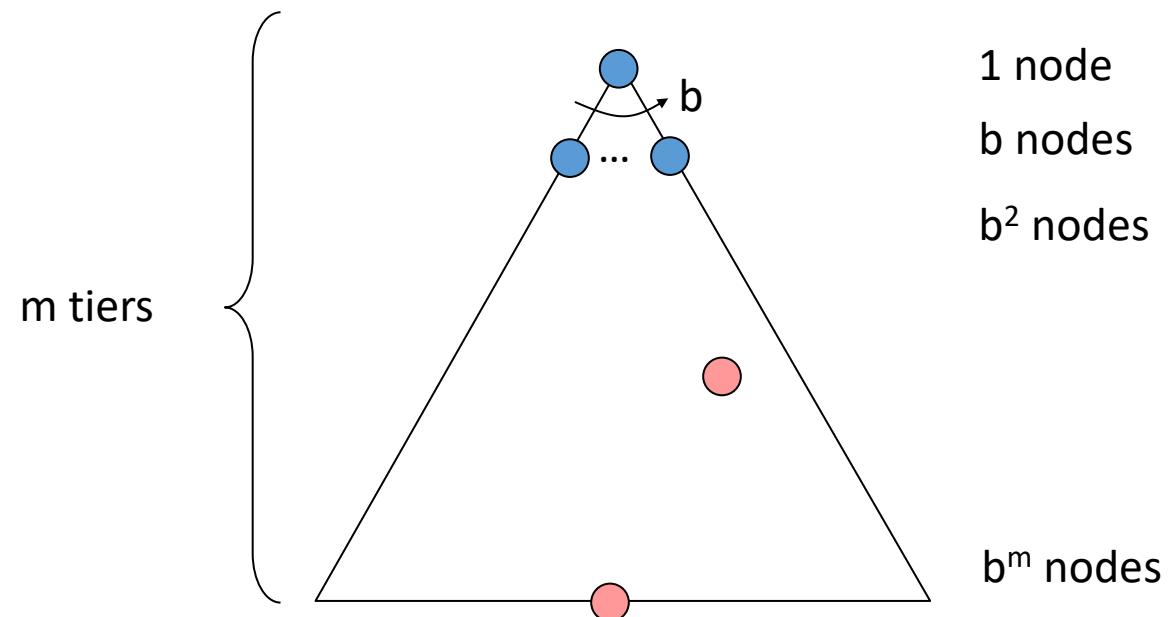
پیاده‌سازی:
یک پشته LIFO Fringe



ویژگی‌های الگوریتم جستجو



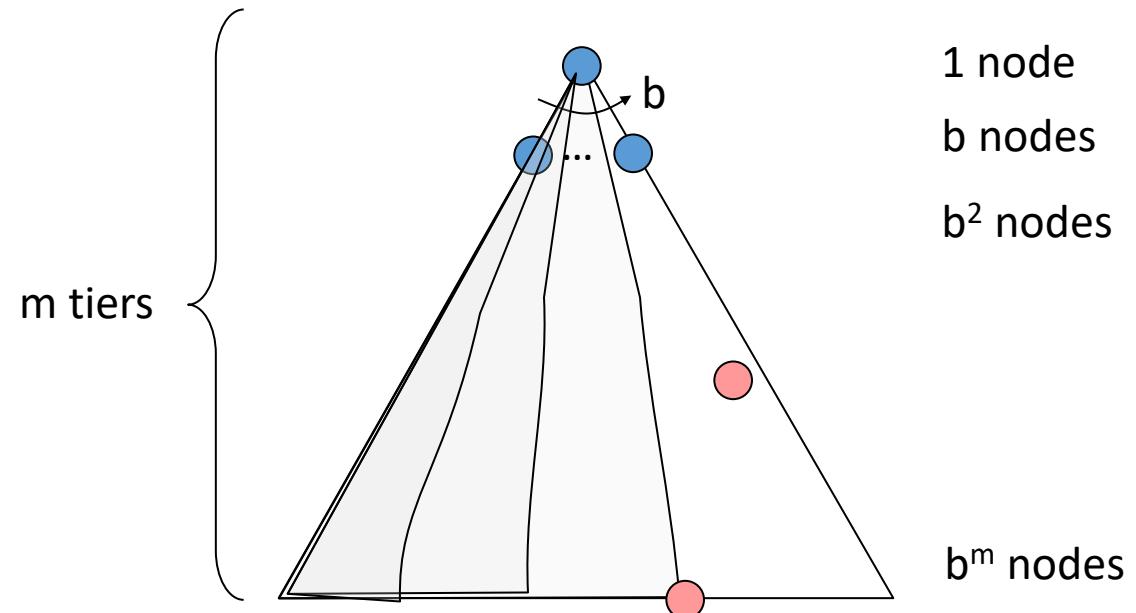
ویژگی‌های الگوریتم جستجو



- کامل بودن (complete): تضمین می‌کند که اگر راه حلی وجود داشته باشد آنرا پیدا کند.
- بهینه (optimal): تضمین می‌کند که مسیر با کمترین هزینه را پیدا کند.

- پیچیدگی زمانی؟
- پیچیدگی فضای؟
- کاریکاتور درخت جستجو:
- b ضریب انشعاب
- m حداقل عمق
- راه حلها در اعماق مختلف
- تعداد گرهها در کل درخت:
$$1 + b + b^2 + \dots + b^m = O(b^m)$$

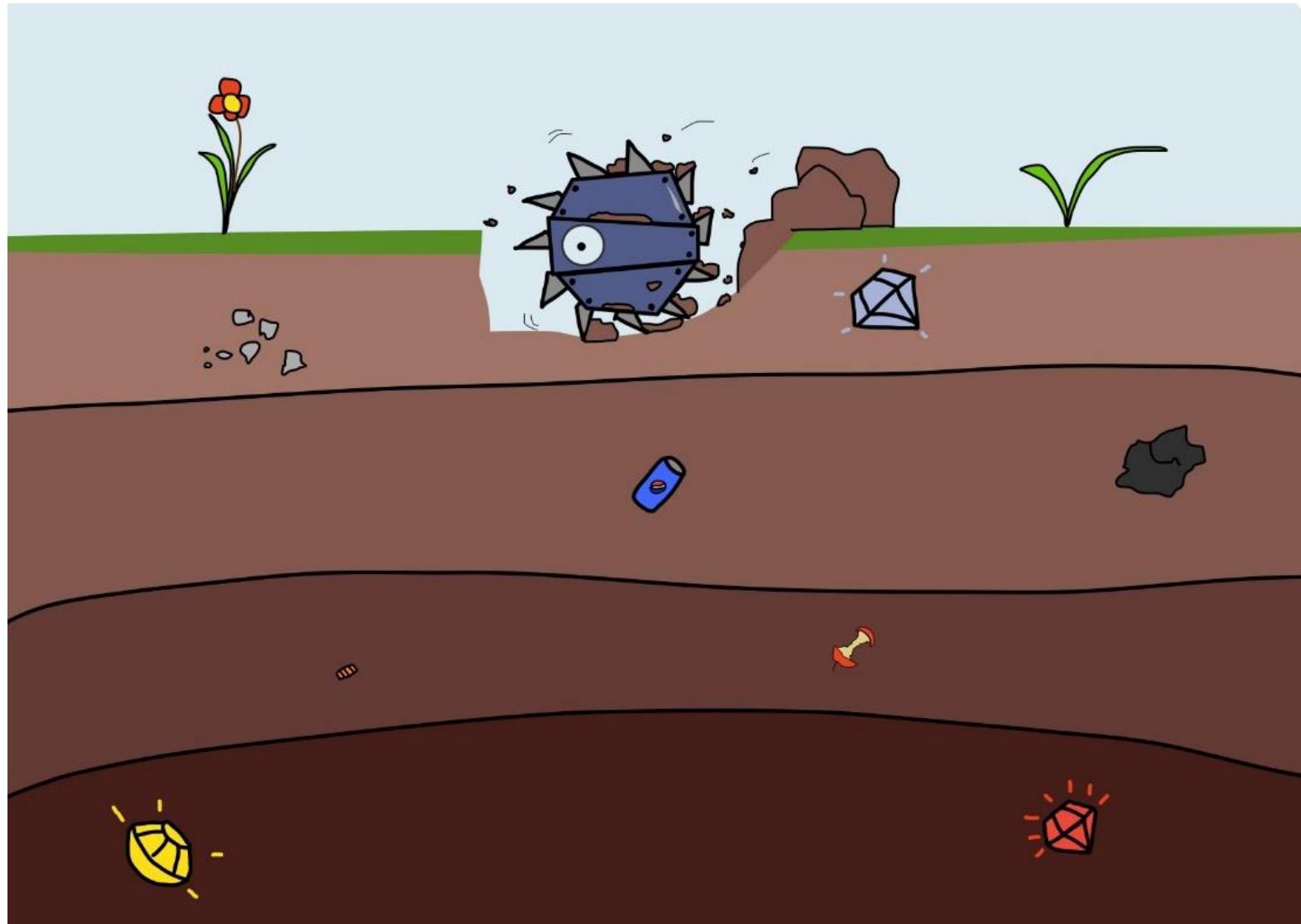
ویژگی‌های جستجوی اول عمق (DFS)



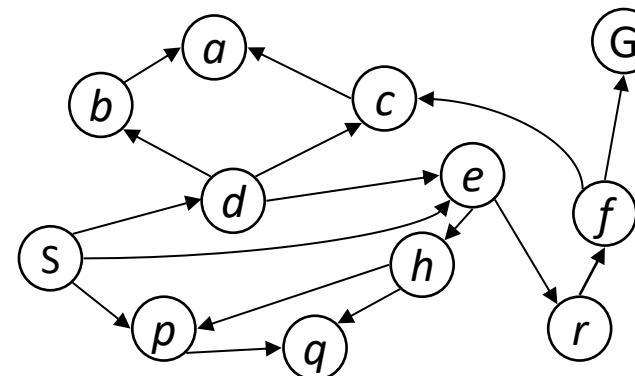
خیر، بدون در نظر گرفتن عمق یا هزینه، "سمت چپ ترین" راه حل را پیدا می‌کند

- چه گرهایی را بسط می‌دهد؟
 - برخی از پیشوندهای سمت چپ درخت می‌تواند کل درخت را پیمایش کند (به شرط عمق محدود)
 - اگر m متناهی باشد، جستجو $O(b^m)$ زمان می‌برد
- لیست حاشیه چه اندازه فضا اشغال می‌کند؟
 - فقط گرهای خواهربرادر (siblings) در مسیر ریشه‌ی درخت را شامل می‌شود، بنابراین $O(bm)$
- آیا کامل است؟
 - می‌تواند بینهایت باشد، بنابراین فقط در صورتی که از دورها جلوگیری کنیم (در ادامه بیشتر توضیح داده می‌شود)
- آیا بهینه است؟

جستجوی اول سطح (BFS)

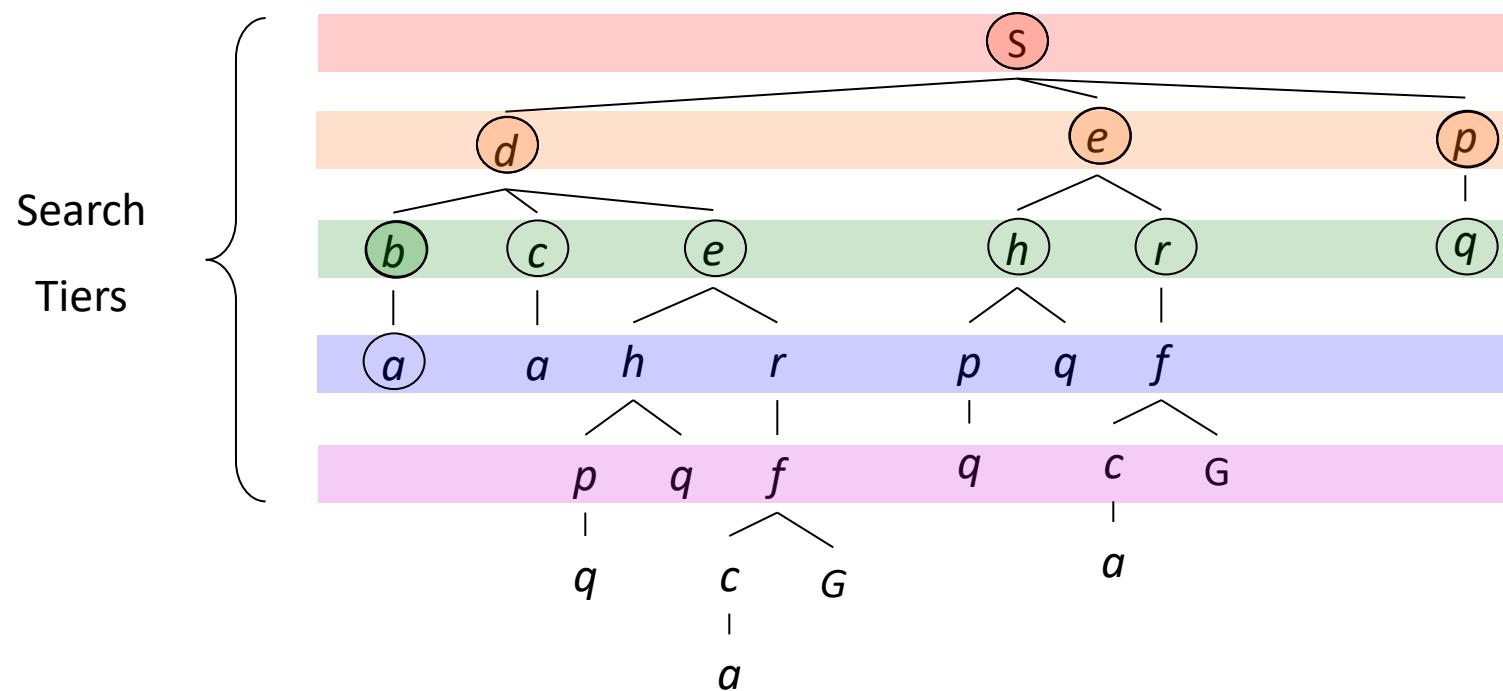


جستجوی اول سطح (BFS)

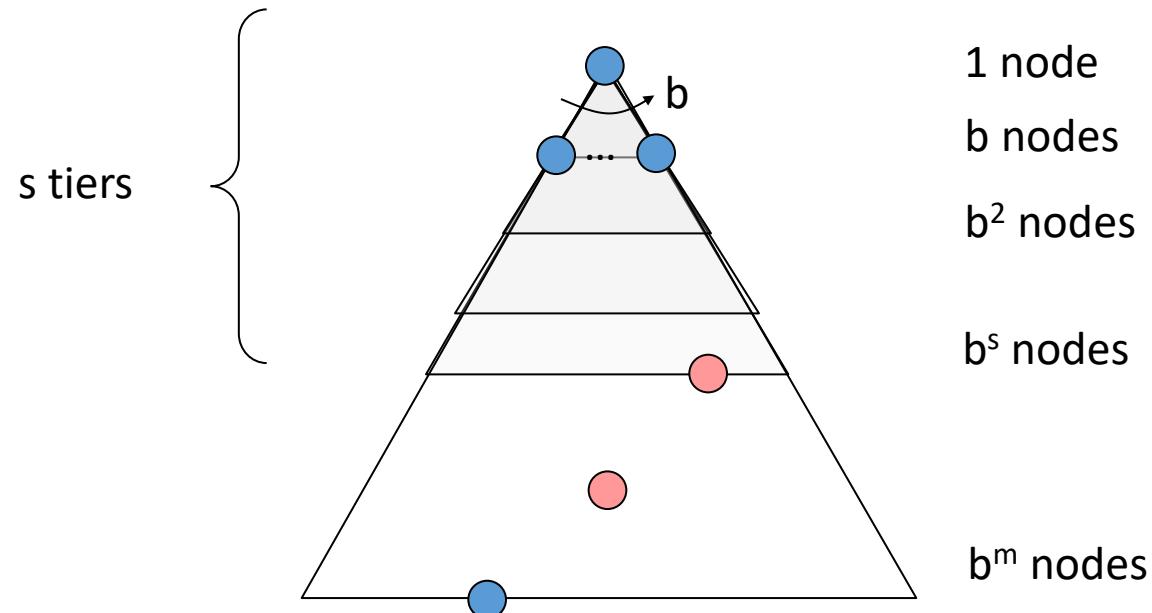


استراتژی:
ابتدا کم عمق‌ترین گره را بسط دهید.

پیاده‌سازی:
یک صف FIFO است.



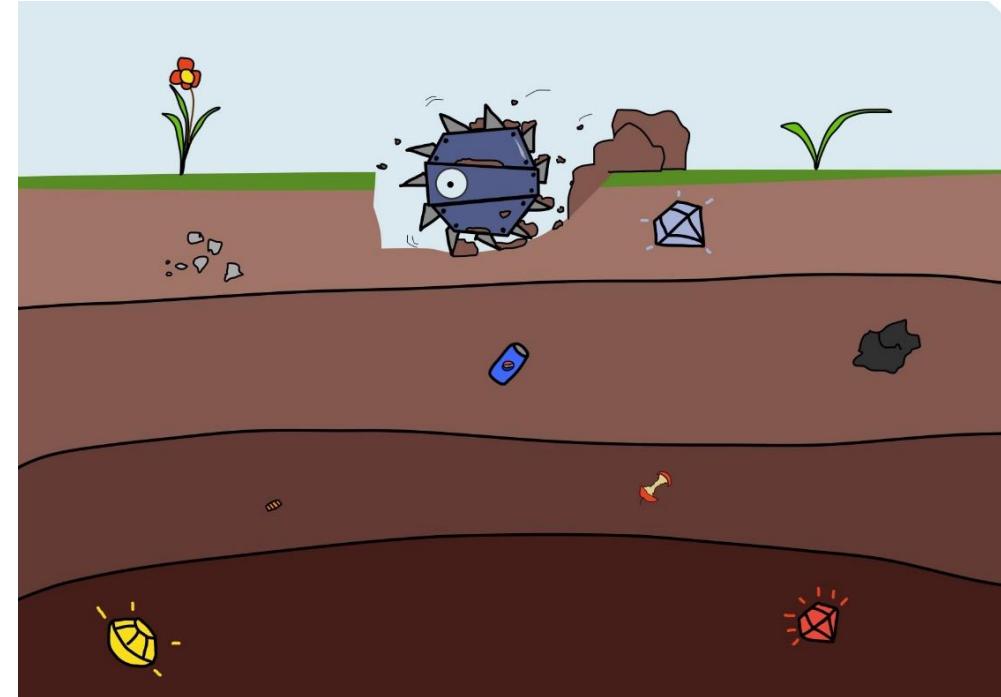
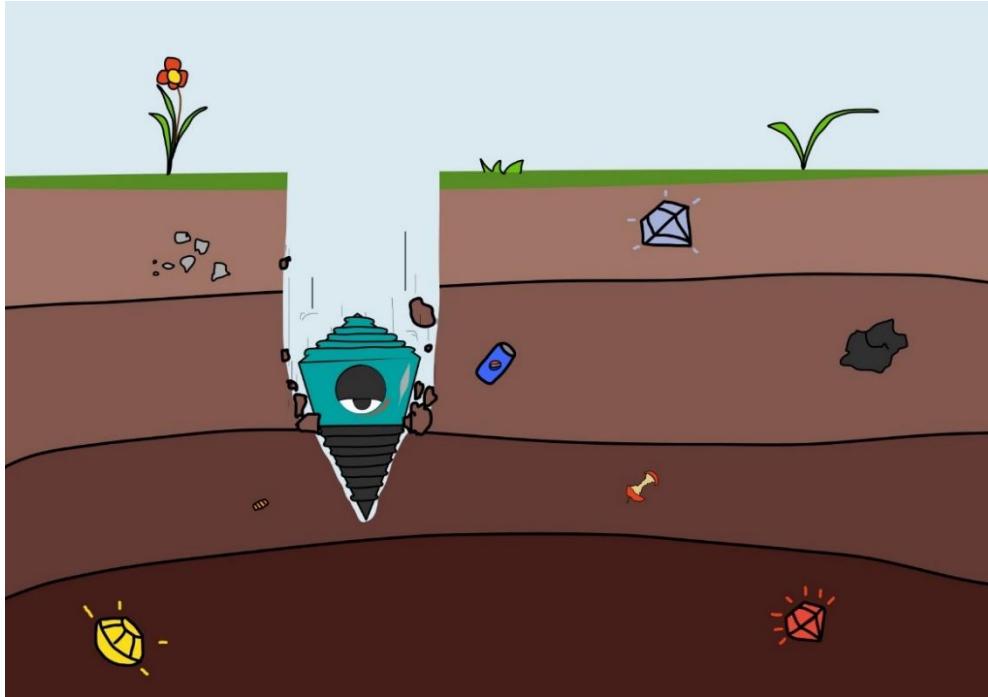
ویژگی‌های جستجوی اول سطح (BFS)



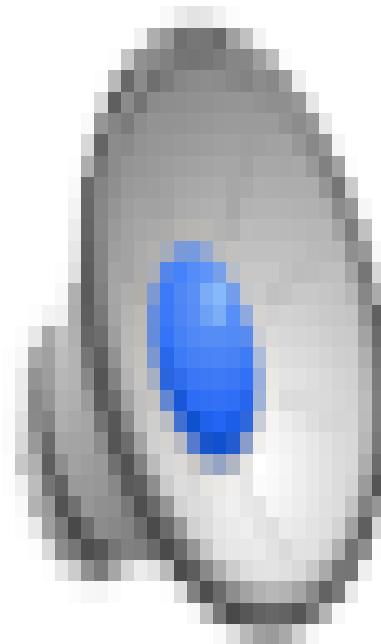
- فقط در صورتی که هزینه‌ها همه 1 باشد (در آینده بیشتر در مورد هزینه‌ها صحبت می‌شود)

- چه گرهایی را بسط می‌دهد?
 - تمام گرهای بالاتر از سطحی‌ترین راه حل را پردازش می‌کند
 - فرض کنید کم عمق‌ترین راه حل در عمق S باشد
 - جستجو $O(b^S)$ زمان می‌برد
- لیست حاشیه چه اندازه فضای اشغال می‌کند?
 - تقریبا آخرین ردیف در بر می‌گیرد، بنابراین $O(b^S)$
 - آیا کامل است؟
 - اگر راه حلی وجود داشته باشد s باید محدود باشد، بنابراین بله!
- آیا بهینه است?

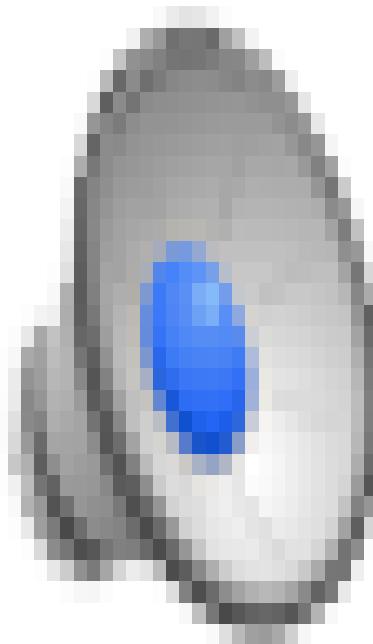
تقابل BFS و DFS



ویدیوی Demo Maze Water DFS/BFS (قسمت اول)

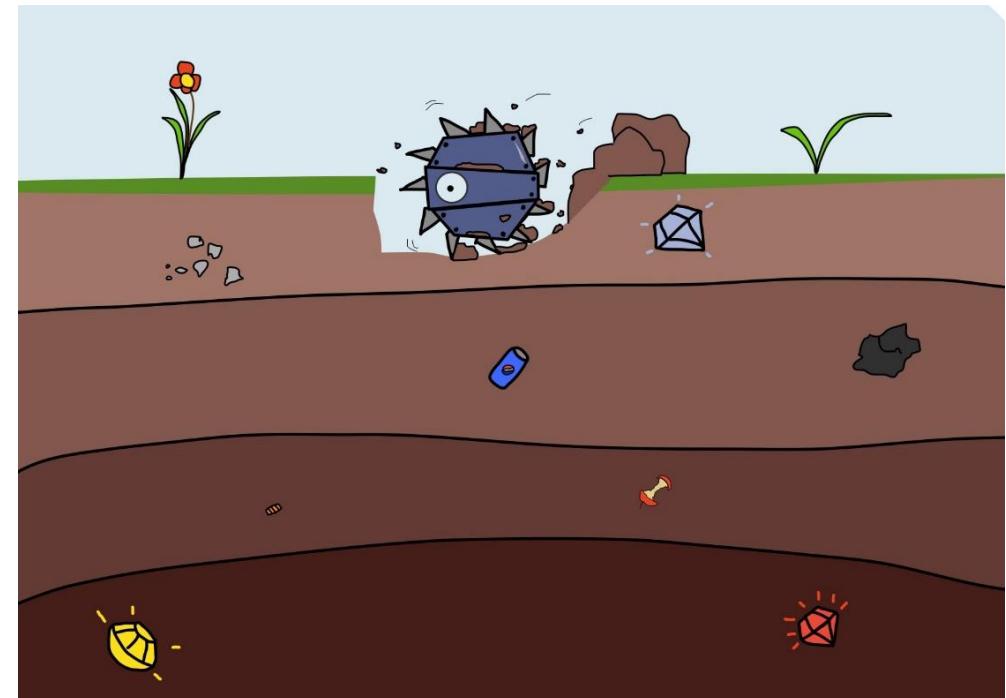
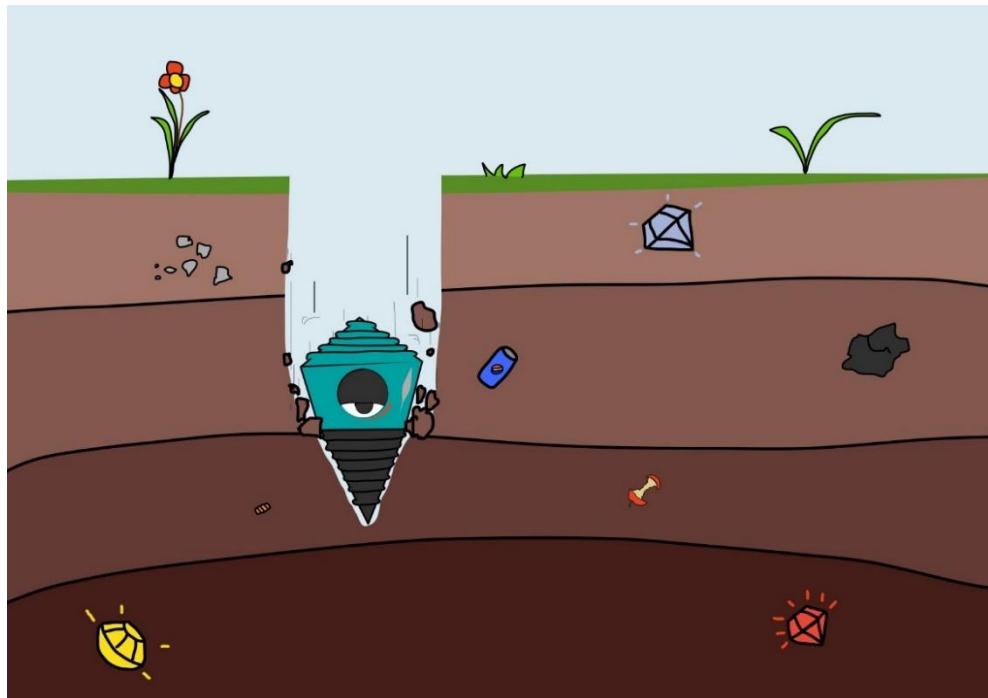


ویدیوی Demo Maze Water DFS/BFS (قسمت دوم)



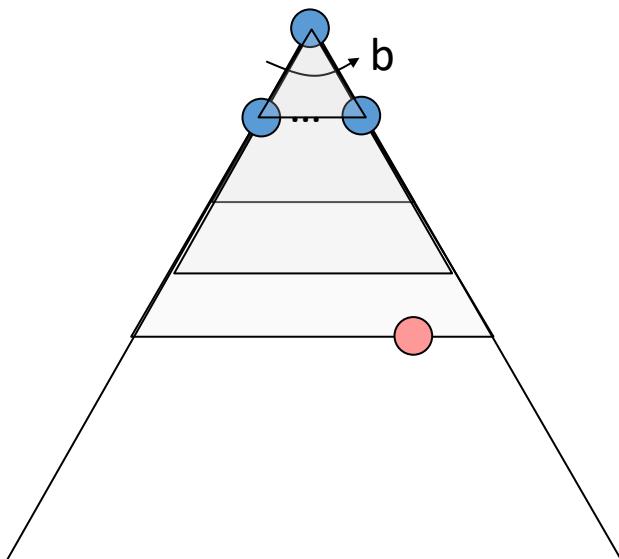
آزمونک: تقابل BFS و DFS

- چه زمانی BFS بهتر از DFS عمل می‌کند؟
- چه زمانی DFS بهتر از BFS عمل می‌کند؟



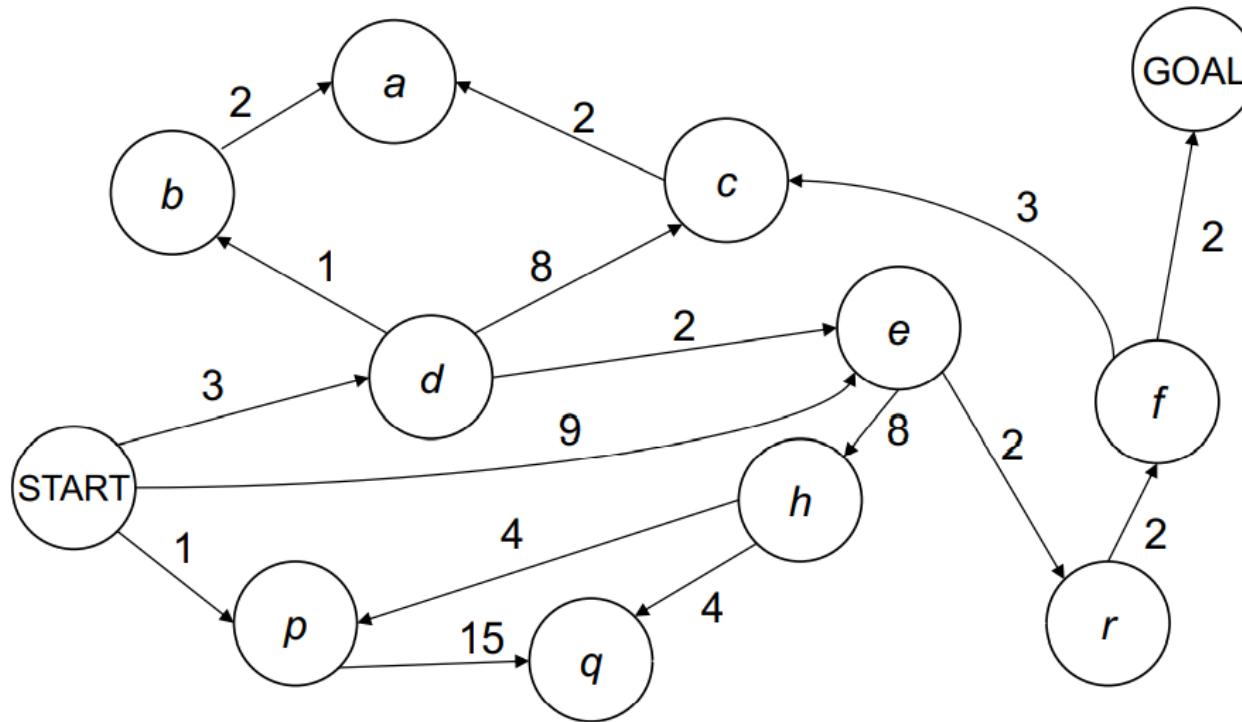
عمیق‌سازی تکرارشونده (iterative deepening)

- ایده: مزیت فضای DFS را با مزایای زمان/ راه حل کم عمق BFS همزمان داشته باشیم



- یک DFS با محدودیت عمق 1 اجرا کنید. اگر راه حلی وجود نداشت...
- یک DFS با محدودیت عمق 2 اجرا کنید. اگر راه حلی وجود نداشت...
- یک DFS با محدودیت عمق 3 و ...
- آیا این کار بیهوده تکراری نیست؟
- به طور کلی بیشترین کار در جستجوی پایین‌ترین سطح اتفاق می‌افتد، بنابراین خیلی بد نیست!

جستجوی حساس به هزینه

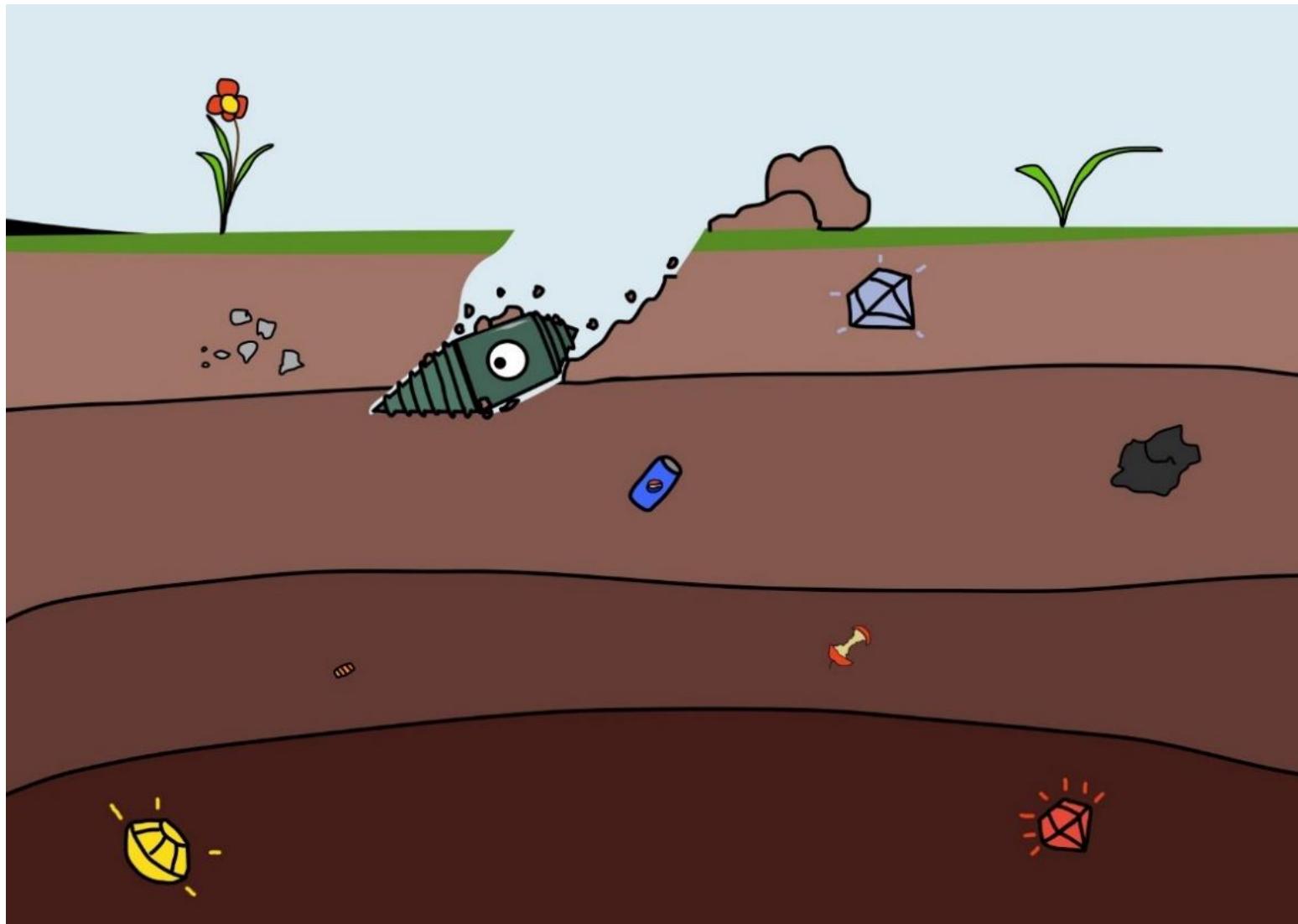


BFS کوتاه ترین مسیر را از نظر تعداد اعمال پیدا می‌کند.

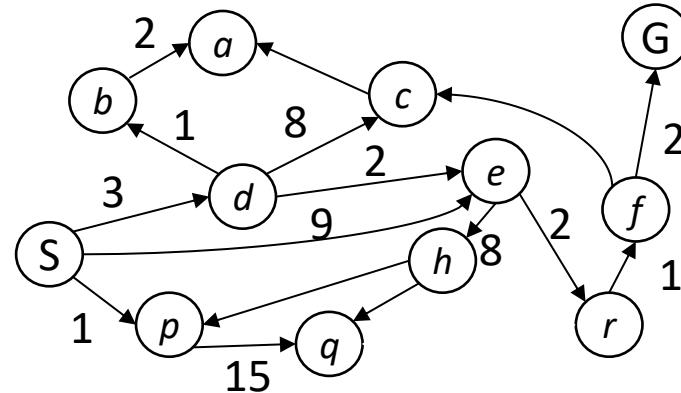
کمترین هزینه را پیدا نمی‌کند. اکنون الگوریتم مشابهی را پوشش می‌دهیم که مسیر کم هزینه را پیدا می‌کند.

چگونه؟

جستجوی هزینه یکنواخت (UCS)

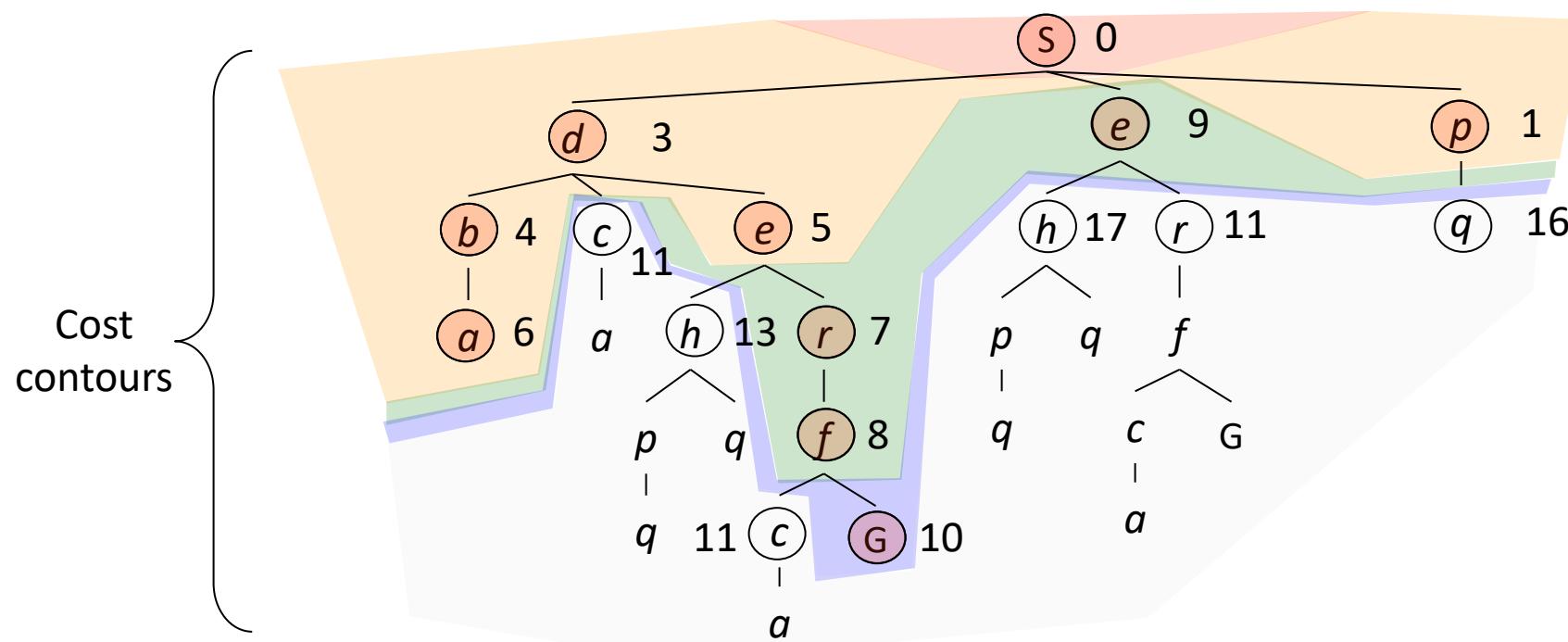


جستجوی هزینه یکنواخت

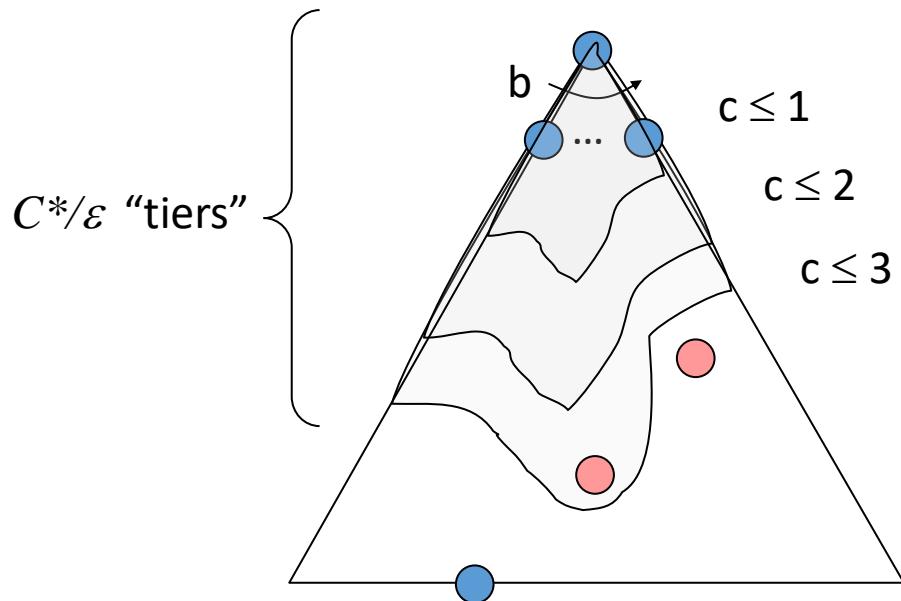


استراتژی:
ابتدا ارزان‌ترین گره را بسط دهید

پیاده‌سازی:
یک صف اولویت است (اولویت: هزینه تجمعی)



ویژگی‌های جستجوی هزینه یکنواخت (UCS)



• اچ‌اچ‌ای را بسط می‌دهد؟

• تمام گره‌هایی که هزینه‌ای کمتر از ارزان‌ترین راه حل دارند را پردازش می‌کند!

• اگر آن هزینه راه حل بهینه C^* و حداقل هزینه کمان ϵ باشد، «عمق مؤثر» تقریباً C^*/ϵ است.

• جستجو $O(b^{C^*/\epsilon})$ زمان می‌برد (تعداد گره‌های بسط یافته)

(به صورت نمایی در عمق مؤثر)

• لیست حاشیه چه اندازه فضا اشغال می‌کند؟

• تقریباً آخرین ردیف را در لیست حاشیه دارد، بنابراین $O(b^{C^*/\epsilon})$

• آیا کامل است؟

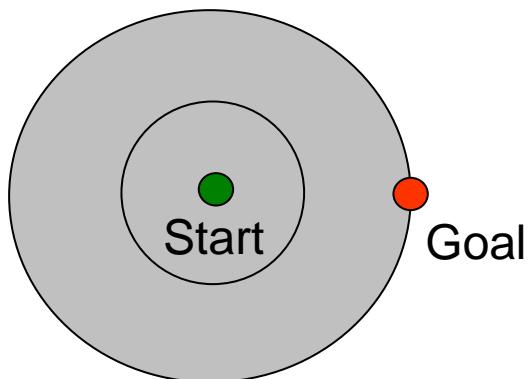
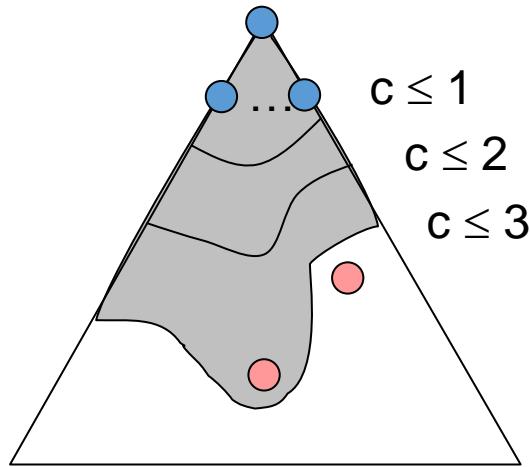
• با فرض اینکه بهترین راه حل دارای هزینه محدود است

و حداقل هزینه قوس مثبت است، بله!

• آیا بهینه است؟

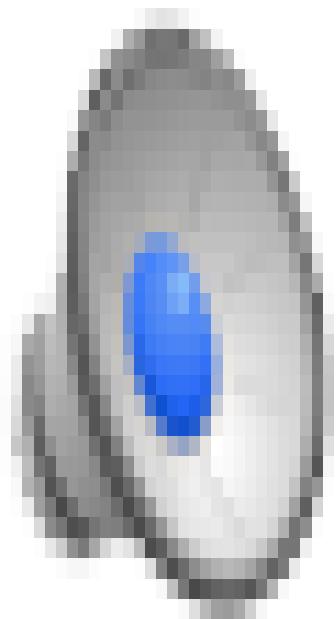
• بله! (اثبات در اسلاید بعدی از طریق A^*)

نکات جستجوی هزینه یکنواخت

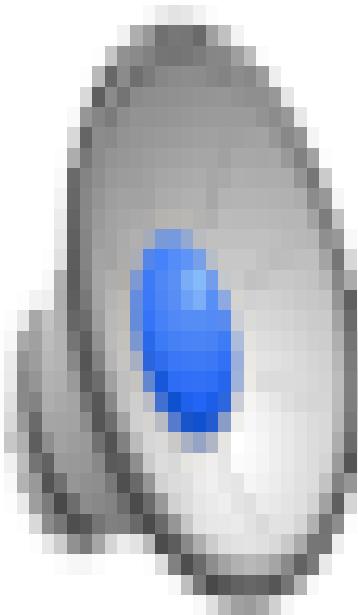


- به یاد داشته باشید: UCS خطوط افزایش هزینه را کاوش می‌کند!
- تمام گره‌هایی که هزینه‌ای کمتر از ارزان‌ترین راه حل دارند را پردازش می‌کند!
- حسن: UCS کامل و بهینه است
- عیب:
 - گزینه‌ها را در هر جهت بررسی می‌کند
 - اطلاعاتی در مورد مکان هدف در نظر نمی‌گیرد
- به زودی این مشکل را برطرف می‌کنیم!

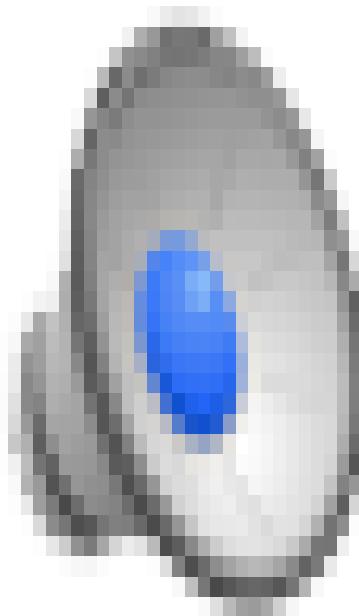
ویدیوی Empty UCS Demo



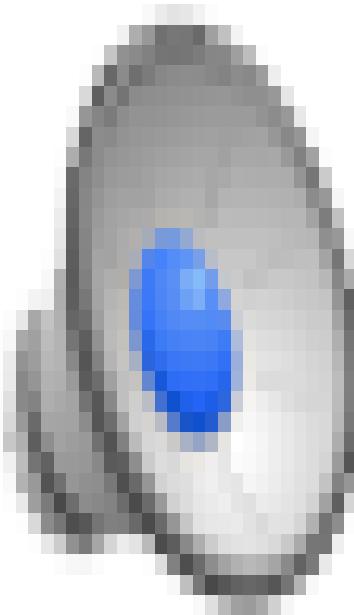
ویدیوی UCS یا Deep/Shallow Water --- DFS, BFS با Demo Maze (قسمت اول)



ویدیوی UCS یا Deep/Shallow Water --- DFS, BFS با Demo Maze (قسمت دوم)



ویدیوی UCS یا Deep/Shallow Water --- DFS, BFS با Demo Maze (قسمت سوم)



یک صف

- همه این الگوریتم‌های جستجو به جز در استراتژی‌های کاوش لیست حاشیه یکسان هستند
- از نظر مفهومی، تمام لیست حاشیه‌ها صفحه‌ای اولویت هستند. (مثلاً مجموعه‌ای از گره‌ها با اولویت‌های پیوست شده)
- عملأً، برای DFS و BFS، می‌توانید با استفاده از پشته‌ها و صفحه‌ها، از سریار $\log(n)$ از یک صفحه اولویت واقعی اجتناب کنید



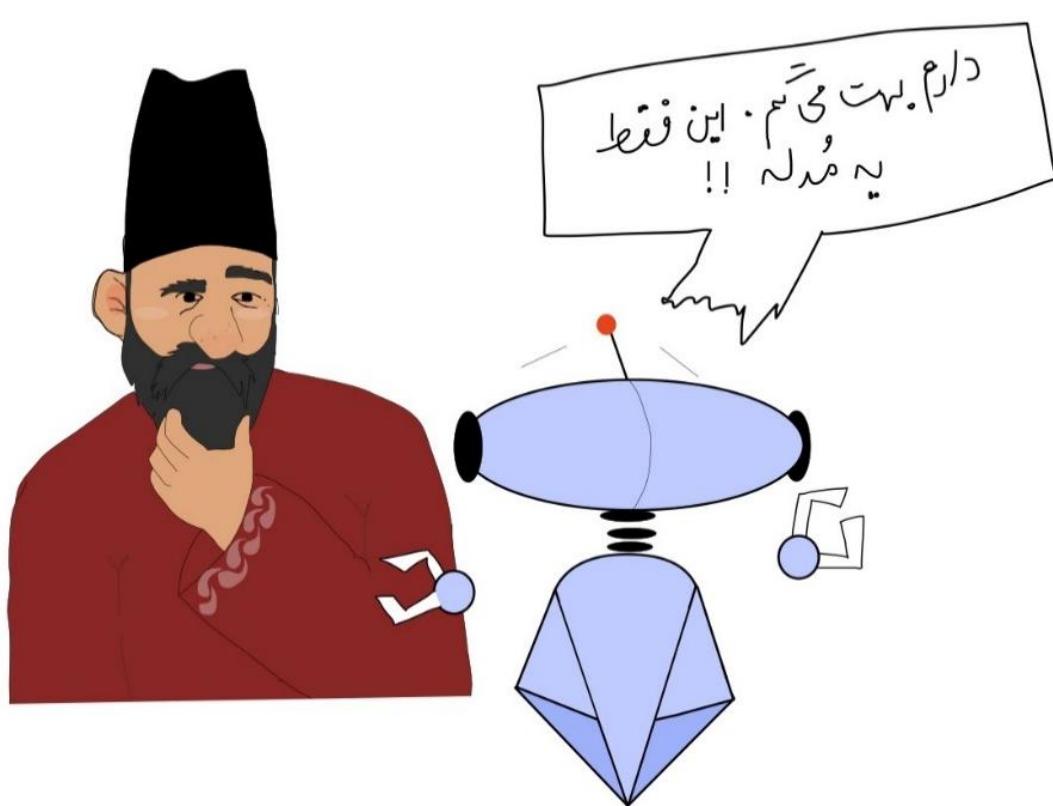
جستجو و مدل

- جستجو بر روی مدل‌های جهان عمل می‌کند و نه خود جهان!!

- عامل در واقع تمام برنامه‌ریزی‌ها را در دنیای واقعی امتحان نمی‌کند!

- برنامه‌ریزی همه در " شبیه‌سازی " است

- جستجوی شما تنها به اندازه مدل شما خوب است...



جستجو به خط رفت؟

