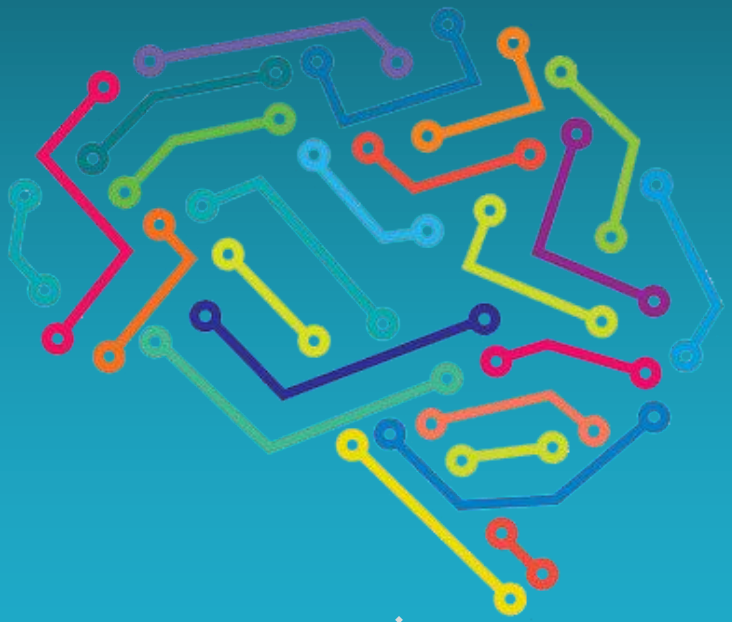
دانشگاه صنعتی شاهرود

حل مساله با جستجو



درس هوش مصنوعی

نیم‌سال اول تحصیلی ۹۸-۱۳۹۷

عامل های حل مساله

- تعریف مساله

۱) حالت اولیه عامل

۲) توصیف فعالیتهای ممکن عامل: استفاده از تابع مابعد

امکان تعریف فضای حالت مساله با توجه به حالت اولیه و تابع مابعد:

فضای حالت، مجموعه ای از حالت هاست که از حالت اولیه می توان به آنها رسید. به صورت گراف نمایش داده می شود

حالتها: گره های گراف

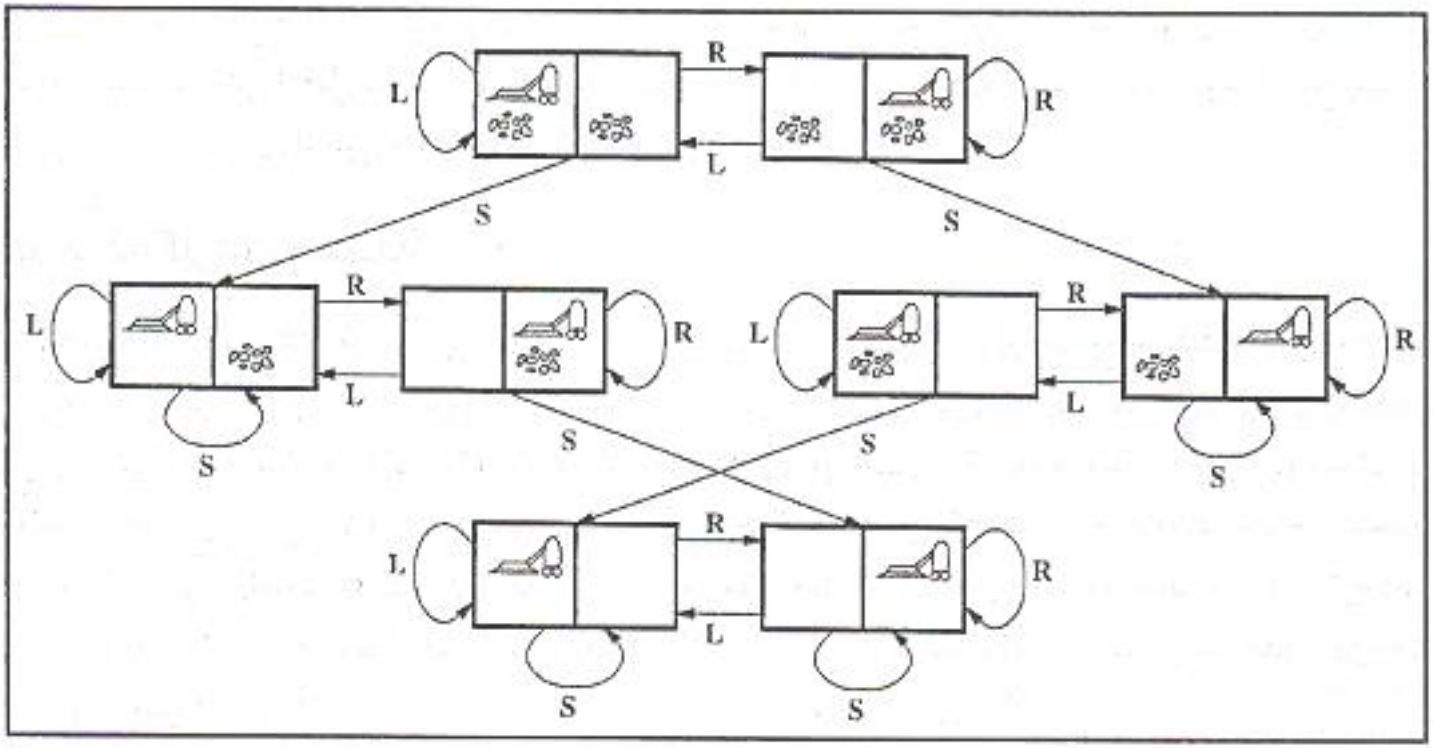
فعالیتها: یال هال گراف

مسیر فضای حالت: دنباله ای از حالتها که توسط دنباله ای از فعالیتها به هم متصل می شوند.

مساله های نمونه

مساله های اسباب بازی

برای تشریح و تمرین کردن بر روی روشهای مختلف حل مساله جهت مقایسه کارایی الگوریتمها



شکل ۳-۳ فضای حالت جهان جارو. یال‌ها فعالیت‌ها را نشان می‌دهند. L = Left، R = Right و S = Suck.

مساله های اسباب بازی

جهان جارو

- **حالتها**

عامل در یکی از دو مکان است که هرکدام ممکن است کثیف باشند یا نباشند. ۸ حالت

- **حالت اولیه**

هر حالتی می تواند به عنوان حالت اولیه طراحی شود

- **تابع مابعد**

حالتهای معتبری را تولید می کند که از ۳ عملیات (Left, Right, Suck) ناشی می شود.

- **آزمون هدف**

تمیز بودن تمام مربع ها

- **هزینه مسیر**

هزینه هر مرحله ۱ است. هزینه مسیر برابر تعداد مراحل در مسیر است

مساله های اسباب بازی

معمای ۸

- **حالتها** مکان هر ۸ خانه شماره دار

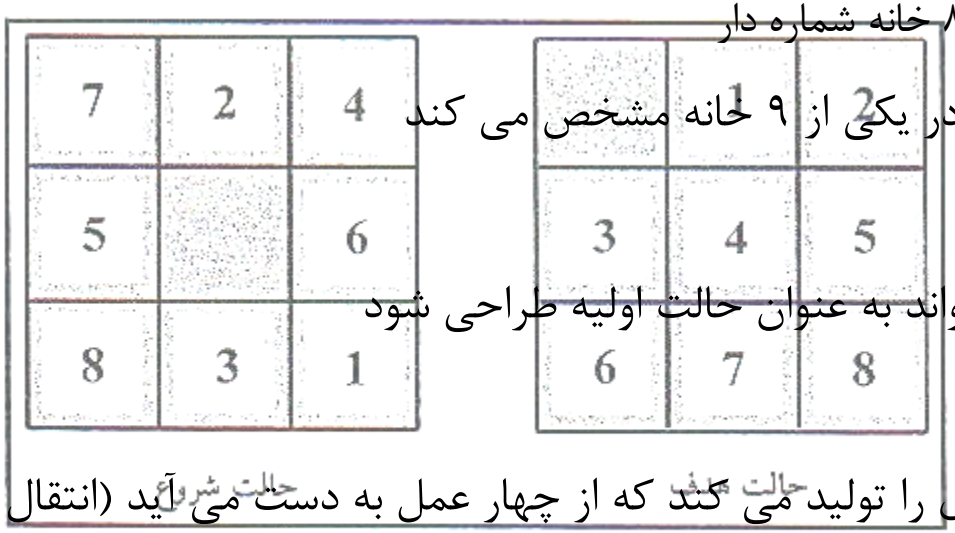
و خانه خالی را در یکی از ۹ خانه مشخص می کند

- **حالت اولیه**

هر حالتی می تواند به عنوان حالت اولیه طراحی شود

- **تابع جانشین**

حالتهای معتبری را تولید می کند که از چهار عمل به دست می آید (انتقال خانه خالی).



شکل ۴-۳ نمونه‌ای از معمای ۸.

- **آزمون هدف**

رسیدن به حالت هدف

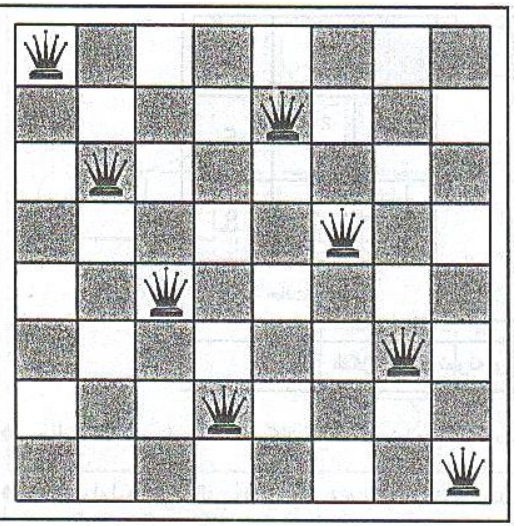
- **هزینه مسیر**

هزینه هر مرحله ۱ است. هزینه مسیر برابر تعداد مراحل در مسیر است

مساله های اسباب بازی

مساله ۸ وزیر

۸ وزیر در صفحه شطرنج طوری قرار گیرند که هر وزیری وزیر دیگر را گارد ندهد



شکل ۵-۳ راه‌حل تقریبی مسئله n وزیر.

- **فرمول بندی: ۱- افزایشی ۲- حالت کامل**
- **حالتها** هر ترتیبی از ۰ تا ۸ وزیر در صفحه (افزایشی)
- **حالت اولیه:** هیچ وزیری در صفحه نیست
- **تابع جانشین:** وزیری را به خانه خالی اضافه می کند
- **آزمون هدف**

۸ وزیر در صفحه وجود دارند و هیچکدام به دیگری گارد نمی دهند

- **$۳*۱۰^{۱۴}$ دنباله ممکن باید بررسی شود.**
- **اصلاح فرمول بندی**

← ۲۰۵۷ دنباله ممکن باید بررسی شود.

مساله های نمونه

مساله های جهان واقعی

مساله مسیریابی:

- مسیریابی در شبکه های رایانه‌ای
- برنامه ریزی عملیات نظامی
- سیستمهای برنامه ریزی مسافرت هوایی

مساله های توریستی: مساله فروشنده دوره گرد

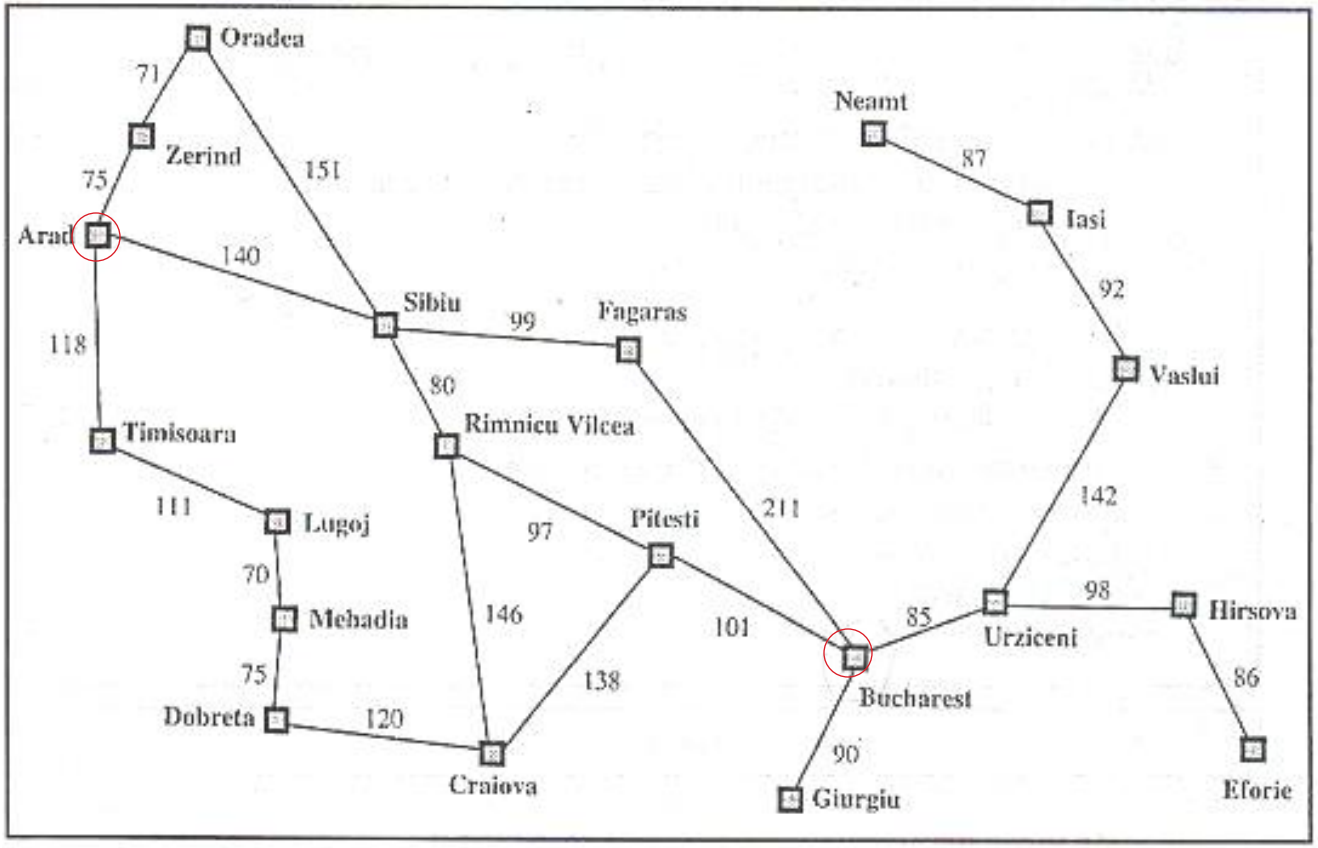
مساله طرح VLSI

هدایت روبات

خط تولید خودکار

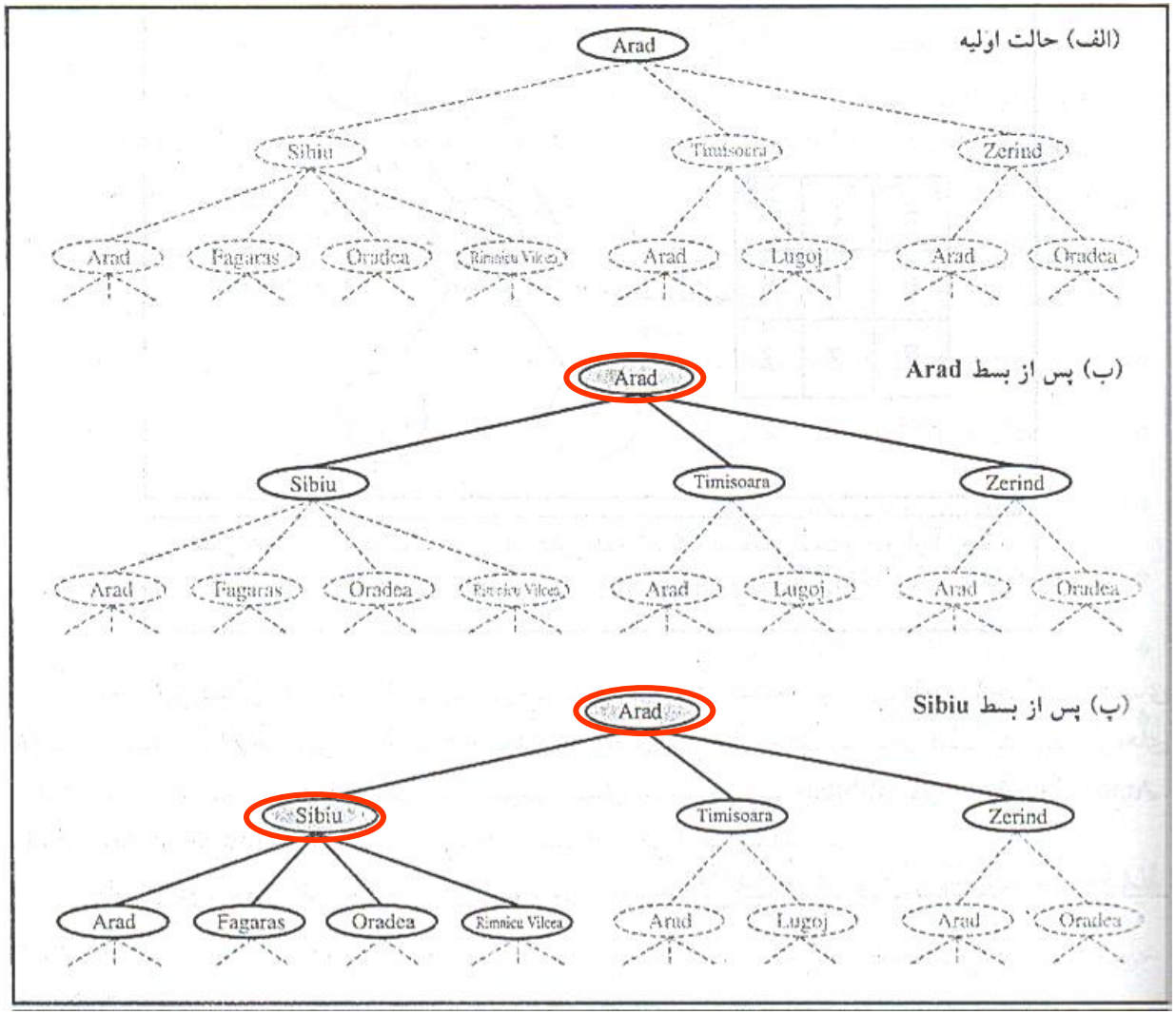
طراحی پروتئین

مساله پیدا کردن مسیر



شکل ۲-۳ نقشه ساده شده‌ای از جاده‌های رومانی.

جستجو برای راه حل ها



الگوریتم جستجو

function TREE-SEARCH(*problem*) **returns** a solution, or failure
  initialize the frontier using the initial state of *problem*
  loop do
    if the frontier is empty **then return** failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state **then return** the corresponding solution
    expand the chosen node, adding the resulting nodes to the frontier

---

function GRAPH-SEARCH(*problem*) **returns** a solution, or failure
  initialize the frontier using the initial state of *problem*
  initialize the explored set to be empty
  loop do
    if the frontier is empty **then return** failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state **then return** the corresponding solution
    add the node to the explored set
    expand the chosen node, adding the resulting nodes to the frontier
      only if not in the frontier or explored set

اندازه گیری کارایی حل مساله

- روشهای اندازه گیری کارایی الگوریتم حل مساله
  - ✓ کامل بودن: تضمین الگوریتم به پیدا کردن راه حل
  - ✓ هینگی: ارائه راه حل بهینه
  - ✓ پیچیدگی زمانی: زمان لازم برای پیدا کردن راه حل
  - ✓ پیچیدگی فضا: حافظه مورد نیاز برای جستجو

اندازه گیری کارایی حل مساله

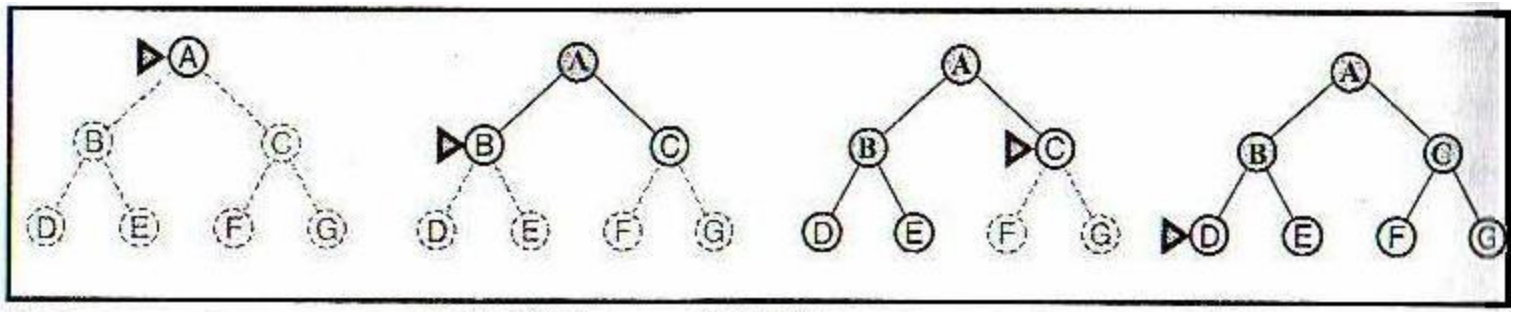
- پیچیدگی مسئله بر حسب سه کمیت بیان می شود
  - ✓ فاکتور انشعاب: b
  - ✓ عمق نزدیکترین گره هدف: d
  - ✓ حداکثر طول مسیرها در فضای حالت: m
- سنجش زمان بر حسب تعدا گره های تولید شده در حین جستجو
- سنجش فضا بر حسب حداکثر گره های ذخیره شده در حافظه

راهبردهای جستجوی نا آگاهانه

- **جستجوی نا آگاهانه (جستجوی کور)**
  - ✓ هیچ اطلاعاتی غیر از تعریف مساله در اختیار الگوریتم نیست.
  - ✓ تولید نود بعدی
  - ✓ تشخیص حالت هدف و غیر هدف
- **جستجوی آگاهانه (جستجوی اکتشافی)**
  - ✓ توانایی مقایسه میزان امیدبخش بودن حالت غیر هدف نسبت به حالت غیر هدف دیگر

جستجوی اول‌سطح (Breadth First)

- ابتدا ریشه بسط داده می شود
- سپس تمام جانشین های ریشه بسط داده می شوند و به همین ترتیب
- یعنی بسط تمام گره های موجود در یک عمق درخت و سپس حرکت در عمق بعدی



شکل ۱۰-۳ جست‌وجوی عرضی در یک درخت دودویی. در هر مرحله، گره‌ای که بعداً باید بسط داده شود، علامت‌دار شده است.

پیچیدگی BFS

- فرض کنید هر گره دارای b گره مابعد است:
  - ✓ b
  - ✓ b^2
  - ✓ b^3
  - ✓ ... تا سطح d، b^d:

$$b + b^2 + b^3 + \ldots + b^d = b\sum_{i=0}^{d-1} b^i = b\frac{b^d - 1}{b - 1} = O(b^d)$$

- پیچیدگی زمانی این الگوریتم $O(b^d)$ می باشد

پیچیدگی BFS

- **پیچیدگی فضایی این الگوریتم نیز $O(b^d)$ می باشد.**
- با فرض b=10، و اینکه ذخیره هر نود نیازمند ۱۰۰۰ بایت است، برای سیستمی با توانایی تولید یک میلیون نود درثانیه:

Depth	Nodes	Time	Memory
2	110	.11 milliseconds	107 kilobytes
4	11,110	11 milliseconds	10.6 megabytes
6	10^6	1.1 seconds	1 gigabyte
8	10^8	2 minutes	103 gigabytes
10	10^{10}	3 hours	10 terabytes
12	10^{12}	13 days	1 petabyte
14	10^{14}	3.5 years	99 petabytes
16	10^{16}	350 years	10 exabytes

Figure 3.13 Time and memory requirements for breadth-first search. The numbers shown assume branching factor $b = 10$; 1 million nodes/second; 1000 bytes/node.

الگوریتم BFS

```
function BREADTH-FIRST-SEARCH(problem) returns a solution, or failure
    node ← a node with STATE = problem.INITIAL-STATE, PATH-COST = 0
    if problem.GOAL-TEST(node.STATE) then return SOLUTION(node)
    frontier ← a FIFO queue with node as the only element
    explored ← an empty set
    loop do
        if EMPTY?(frontier) then return failure
        node ← POP(frontier)  /* chooses the shallowest node in frontier */
        add node.STATE to explored
        for each action in problem.ACTIONS(node.STATE) do
            child ← CHILD-NODE(problem, node, action)
            if child.STATE is not in explored or frontier then
                if problem.GOAL-TEST(child.STATE) then return SOLUTION(child)
                frontier ← INSERT(child, frontier)
```

Figure 3.11 Breadth-first search on a graph.

مثال

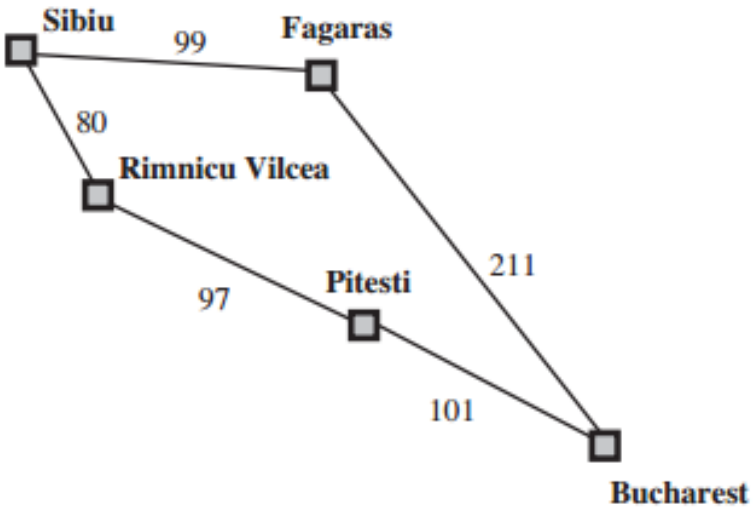


Figure 3.15 Part of the Romania state space, selected to illustrate uniform-cost search.

(S) → (F, R) → (P, F) → return SFB

تحلیل BFS

- پیچیدگی زمانی
- پیچیدگی فضایی:
- کامل است: یعنی اگر جوابی باشد، بالاخره به آن می‌رسد.
- بهینه است اگر:
  - ✓ اگر هزینه مسیر یک تابع غیرنزولی از عمق تود باشد
  - ✓ مثال رایج: همه اعمال (یالها) یکسان باشد
- با چند اصلاحیه می‌تواند همیشه بهینه باشد:

✓ Uniform-cost BFS

جستجوی یکنواخت

- همان جستجوی عرضی است که به جای بسط سطحی ترین گره، گرهی با کمترین هزینه مسیر بسط می دهد.
- اهمیت در کل هزینه مراحل است نه تعداد مراحل
- پیچیدگی زمان و فضا اگر:
  - ✓ C^* هزینه راه حل بهینه باشد،
  - ✓ هزینه هر عملی حداقل ε باشد،

در بدترین حالت:

$$b^{1+\lfloor C^*/\varepsilon \rfloor}$$

مثال

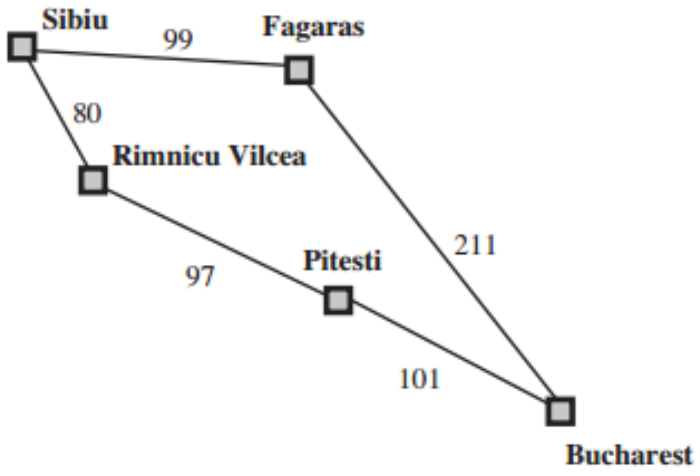


Figure 3.15 Part of the Romania state space, selected to illustrate uniform-cost search.

$(S_0) \rightarrow (F_{99,}\ R_{80}) \rightarrow (P_{177,}\ F_{99}) \rightarrow (B_{310}\ ,P_{177})$
$\rightarrow (B278) \rightarrow$ return SRPB

جستجوی عمقی

بسط عمیقترین گره در حاشیه فعلی درخت جستجو، تاجاییکه گره ها فاقد جانشین باشند

حذف این گره ها از حاشیه درخت

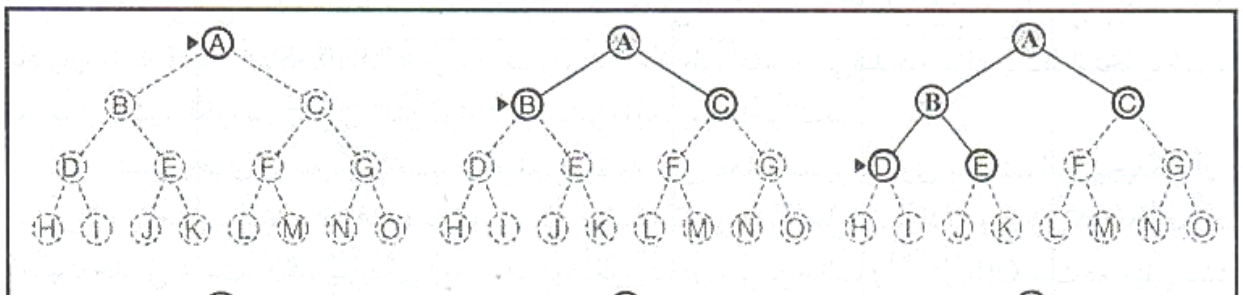
- استفاده از صف LIFO (پشته)
- کامل نیست
- بهینه نیست

پیچیدگی زمان و فضا $O(mb)$

جستجوی عقبگرد (Backtracking)

پیچیدگی زمان و فضا $O(m)$

جستجوی عمقی



جستجوی عمقی محدود

حل مساله درختها نامحدود توسط جستجوی عمقی با عمق محدود l

- اگر $l<d$ باشد کامل نیست
- اگر $l>d$ باشد بهینه نیست

پیچیدگی زمانی $O(b^l)$

پیچیدگی فضا $O(bl)$

جستجوی عمیق کننده تکراری

بهترین عمق محدود را می یابد

همانند جستجوی عرضی وقتی کامل است که فاکتور انشعاب محدود باشد

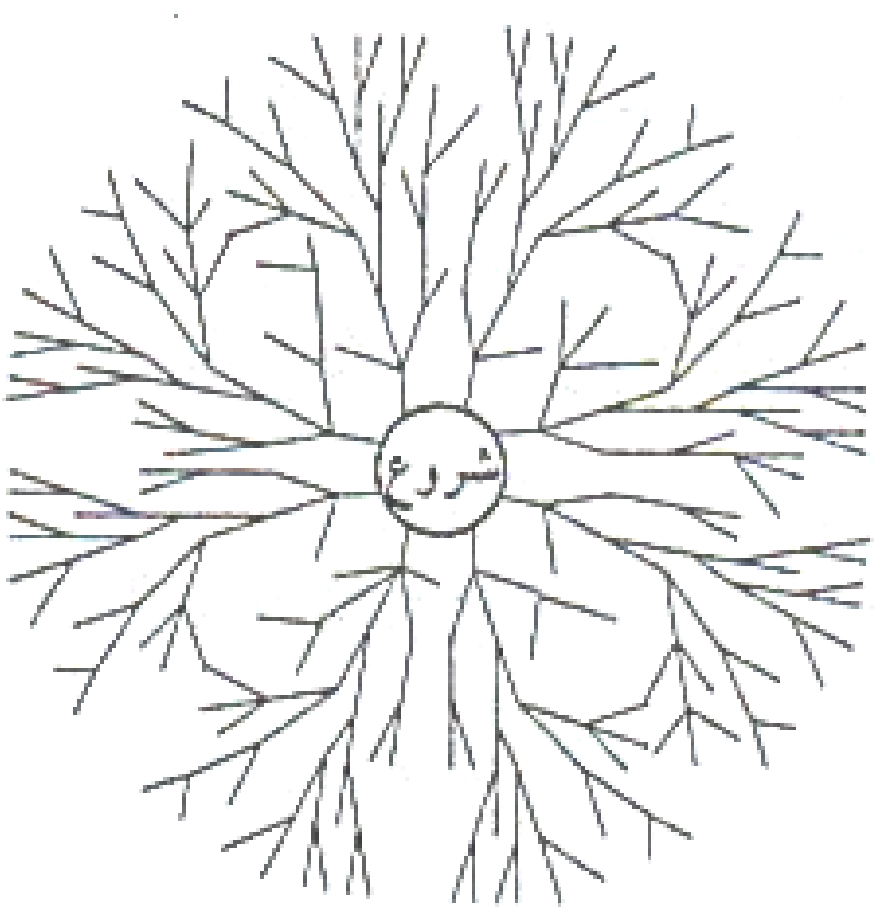
وقتی بهینه است که هزینه مسیر تابعی غیر نزولی از عمق گره باشد

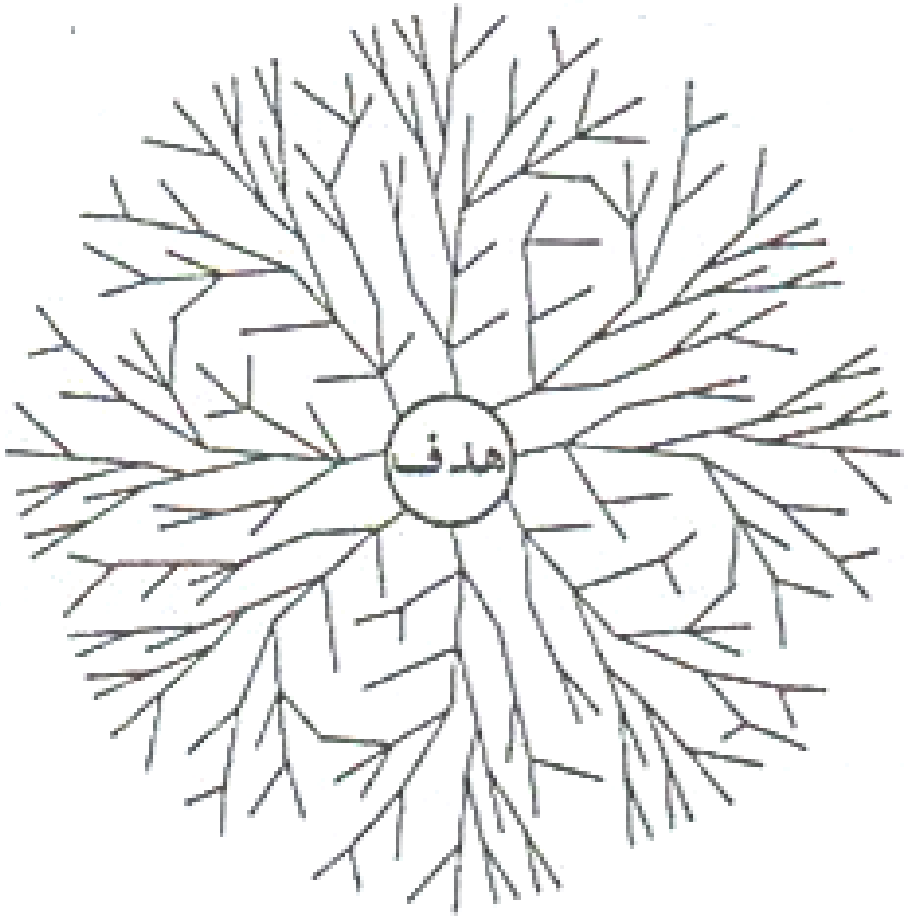
پیچیدگی زمانی $O(b^d)$

همانند جستجوی عمقی پیچیدگی فضا $O(bd)$

جستجوی عمیق کننده تکراری

جستجوی دوطرفه





جستجوی دوطرفه

گره را قبل از بسط در حاشیه درخت دیگر جستجو می کند

پیچیدگی زمانی $O\ (b^{d/2})$

همانند جستجوی عمقی پیچیدگی فضا $O\ (b^{d/2})$

مقایسه راهبردهای جست و جوی نا آگاهانه

ملاک	عرضی	هزینه یکنواخت	عمقی	عمق محدود	عمیق‌کننده تکراری	دوطرفه (درصورت‌امکان)
کامل؟	Yes[a]	Yes [a, b]	No	No	Yes [a]	Yes [a, d]
زمان	$O(b^{d+1})$	$O(b^{\lceil C^*/\varepsilon \rceil})$	$O(b^m)$	$O(b^{\ell})$	$O(b^{d})$	$O(b^{d/2})$
فضا	$O(b^{d+1})$	$O(b^{\lceil C^*/\varepsilon \rceil})$	$O(bm)$	$O(b\ell)$	$O(bd)$	$O(b^{d/2})$
بهینگی؟	Yes[c]	Yes	No	No	Yes [c]	Yes [c, d]

شکل ۱۷-۳ ارزیابی راهبردهای جست‌وجو. b فاکتور انشعاب، d عمق سطحی‌ترین راه‌حل، m حداکثر عمق درخت جست‌وجو، ℓ عمق محدود است. مفهوم اندیس‌های بالا: [a] کامل است اگر b کامل باشد، [b] کامل است اگر برای ε مثبت، $\varepsilon <$ هزینه مرحله. [c] بهینه است اگر هزینه تمام مراحل یکسان باشند. [d] اگر هر دو طرف از جست‌وجوی عرضی استفاده کنند.