بسمه تعالی

هوش مصنوعی

حل مسئله – ۶

نیمسال اول ۱۴۰۲-۱۴۰۱

دکتر مازیار پالهنگ

آزمایشگاه هوش مصنوعی

دانشکدهٔ مهندسی برق و کامپیوتر

دانشگاه صنعتی اصفهان

یادآوری

- جستجوی آگاهانه
- جستجوی بهترین نخست حریصانه
- حستجوی A^*

عمیق ساز تکراری A* (IDA*)

- A* حافظهٔ زیادی می تواند استفاده نماید.
- عمیق ساز تکراری A* همانند جستجوی عمیق ساز تکراری می باشد،
- با این تفاوت که مقدار f=g+h برای حد عمق استفاده می شود.
- در هر تکرار مقدار حد، کمترین مقدار f رأسی است که از حد عمق در مرحلهٔ قبل عبور کرده بود.
- در این حالت نیاز به استفاده از یک صف اولویت دار نمی باشد.

عمیق ساز تکراری A* (IDA*)

- الگوریتم همانند عمیق ساز تکراری می تواند بصورت بازگشتی پیاده سازی شود،
- فقط حد عمق با استفاده از مقدار f تعیین می شود، و
- حد بعدی هم از روی مقدار f تکرار قبل بدست می آید.
- بنابر این میزان استفادهٔ حافظهٔ این الگوریتم شبیه به روش عمق نخست می باشد.
- ولی در شرایطی می تواند میزان رئوسی که بازدید می شوند از A* بیشتر باشد.

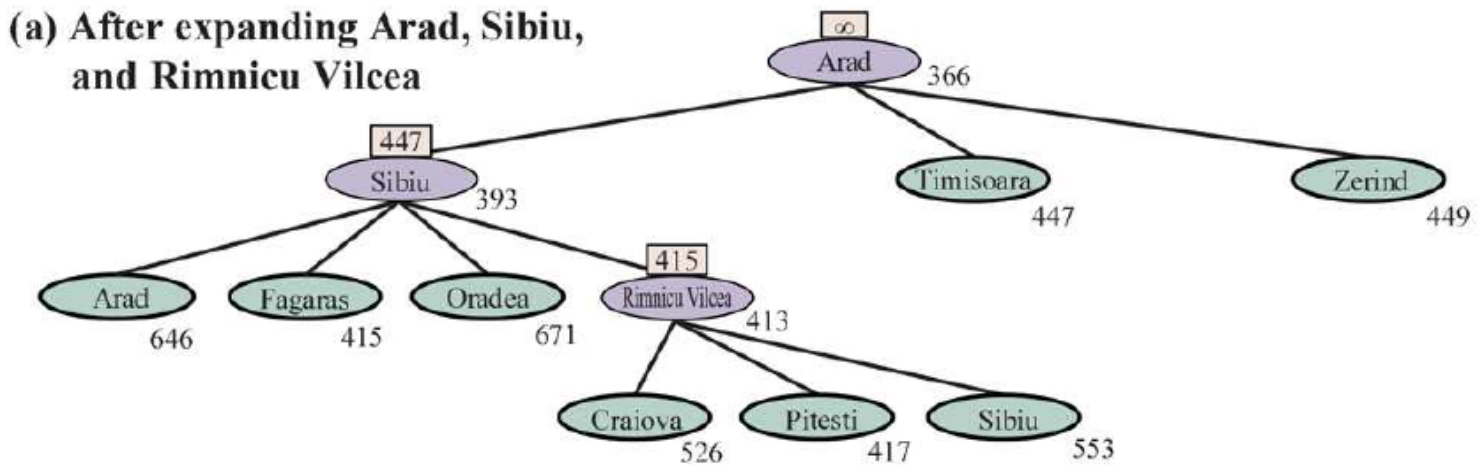
```
function IDA*(problem) returns a solution, or failure
        root = Make-Node(problem.Initial-State)
        limit = f(root)
        while  limit < max_limit
                result, new_limit = DLS(root, limit)
                if result = solution, return solution
                limit = new_limit
        return failure

function DLS(n, limit) returns a solution, or new_limit
        if f(n) > limit return f(n)
        if problem.GoalTest(n.State) then return solution
        else for all s in Successors(n)
                result, new_limit = DLS(s, limit)
                if result = solution return result
                else if  new_limit < min
                        min = new_limit
        return min
```

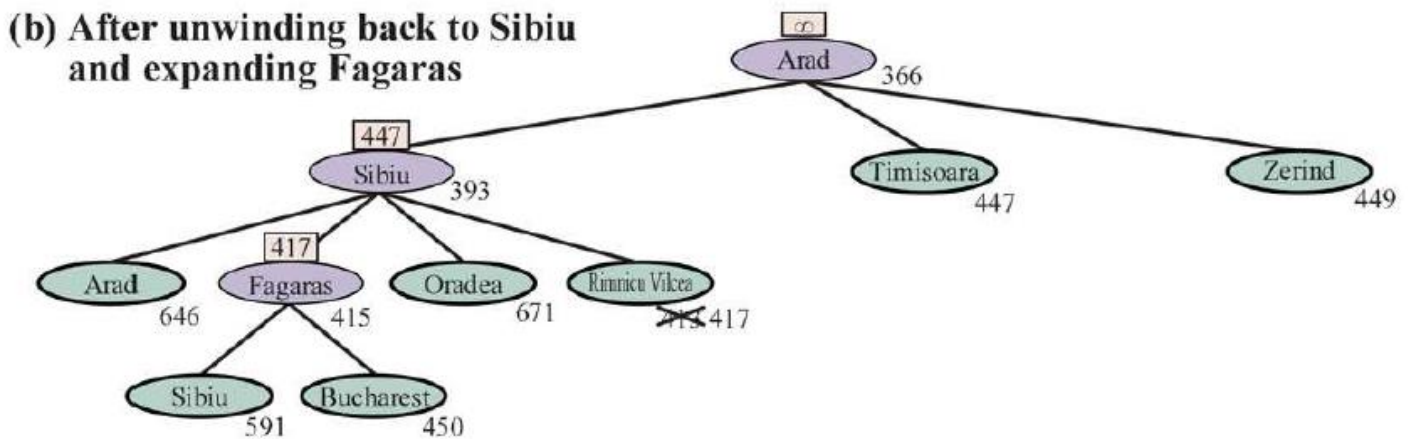
internet

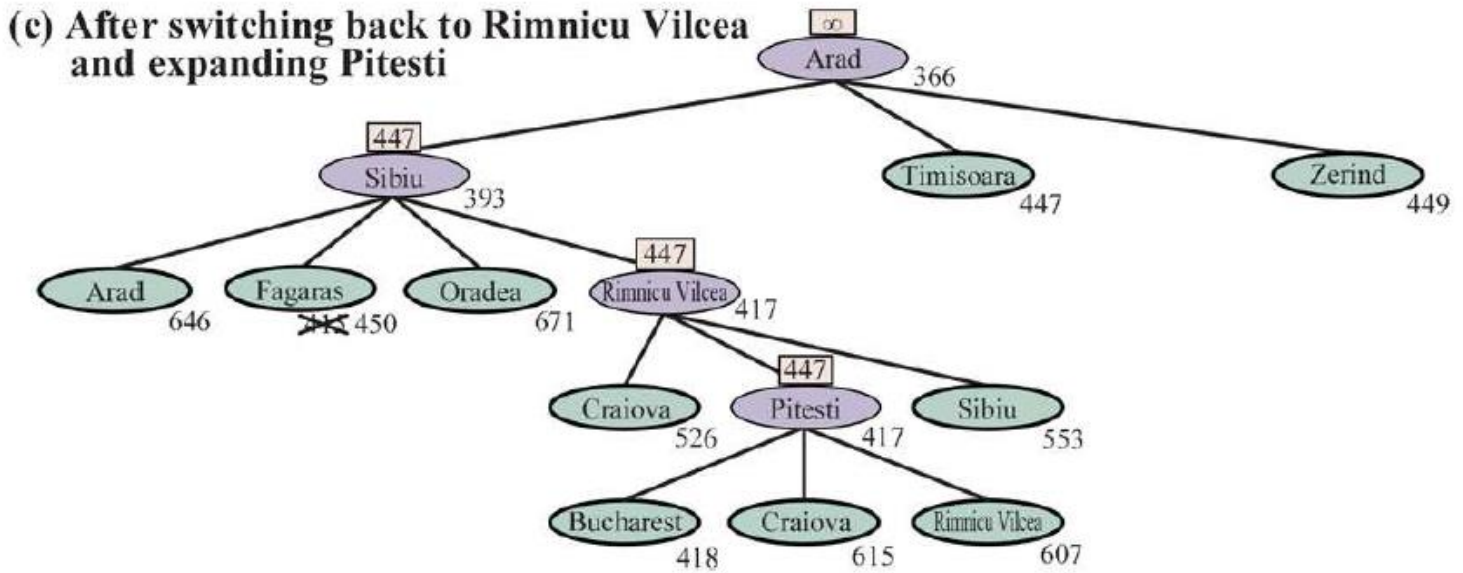
جستجوی بهترین نخست بازگشتی

- شبیه به عمق نخست بازگشتی عمق محدود شده
- الگوریتم مقدار f بهترین مسیر جایگزین از هر جد رأس فعلی را دنبال می کند.
- اگر f رأس فعلی از این حد عبور کند، بازگشت به مسیر جایگزین انجام می گیرد.

.


(a) After expanding Arad, Sibiu, and Rimnicu Vilcea
∞
Arad
366
447
Sibiu
393
Timisoara
447
Zerind
449
Arad
646
Fagaras
415
Oradea
671
415
Rimnicu Vilcea
413
Craiova
526
Pitesti
417
Sibiu
553


(b) After unwinding back to Sibiu and expanding Fagaras
∞
Arad
366
447
Sibiu
393
Timisoara
447
Zerind
449
417
Arad
646
Fagaras
415
Oradea
671
Rimnicu Vilcea
417
Sibiu
591
Bucharest
450


(c) After switching back to Rimnicu Vilcea and expanding Pitesti
∞
Arad
366
447
Sibiu
393
Timisoara
447
Zerind
449
Arad
646
Fagaras
450
Oradea
671
447
Rimnicu Vilcea
417
Craiova
526
447
Pitesti
417
Sibiu
553
Bucharest
418
Craiova
615
Rimnicu Vilcea
607

- بهینه؟
  - بله اگر h قابل پذیرش باشد
- کامل؟
  - بله
- پیچیدگی فضا $O(bd)$
- پیچیدگی زمان – مشخص نیست.

الگوریتم SMA*

- RBFS حافظهٔ زیادی استفاده نمی کند.
- SMA* همانند A* جستجو می کند تا حافظه پر شود.
- در این هنگام برگ با بزرگترین f را انداخته و مقدار f آن را در والدش ذخیره می کند.
- ممکن است رأسی در مسیر پاسخ بهینه قرار داشته باشد ولی به علت کمبود حافظه دیگر قابل بسط دادن نباشد.

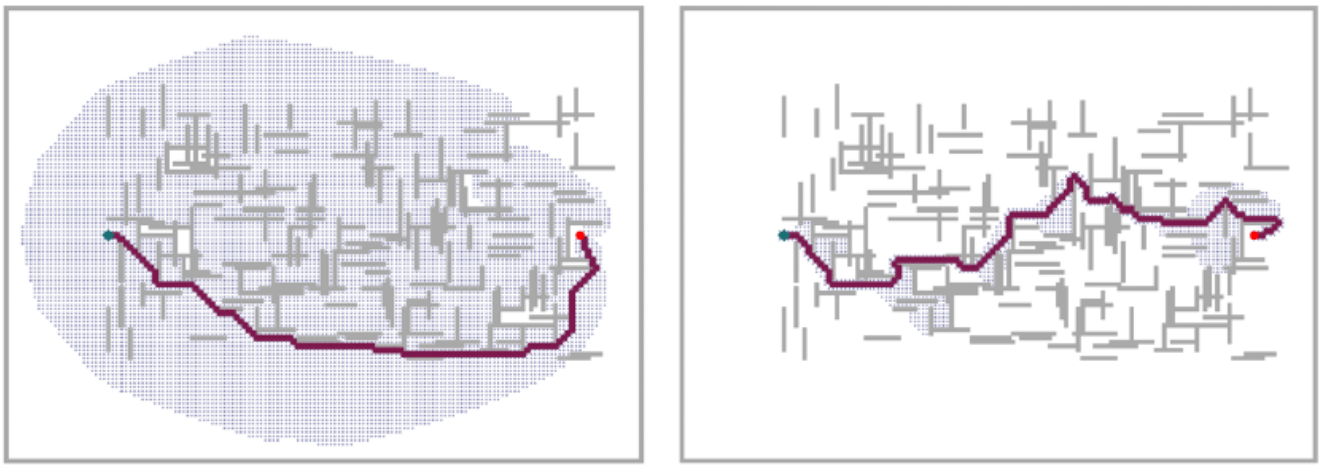
A* وزندار

- در صورتی که به پاسخ زیربهینه ولی نسبتاً خوب راضی باشیم می توان از A× وزندار استفاده کرد.
- در این صورت مکاشفه غیرقابل پذیرش است.
- در این حالت:

$$f(n) = g(n) + W \times h(n)$$

- W>1

Figure 3.21

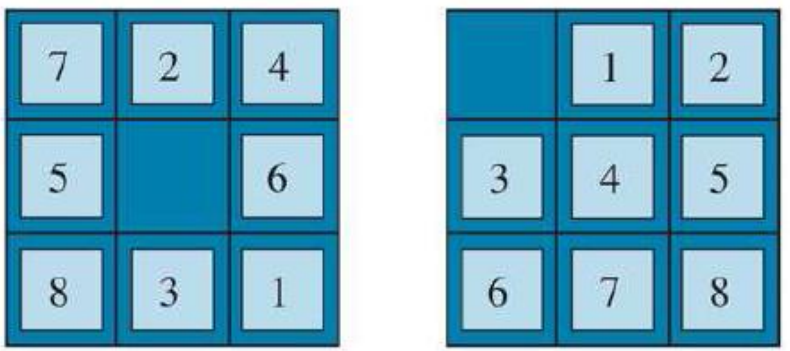


(a) (b)

Two searches on the same grid: (a) an A* search and (b) a weighted A* search with weight $W = 2$. The gray bars are obstacles, the purple line is the path from the green start to red goal, and the small dots are states that were reached by each search. On this particular problem, weighted A* explores 7 times fewer states and finds a path that is 5% more costly.

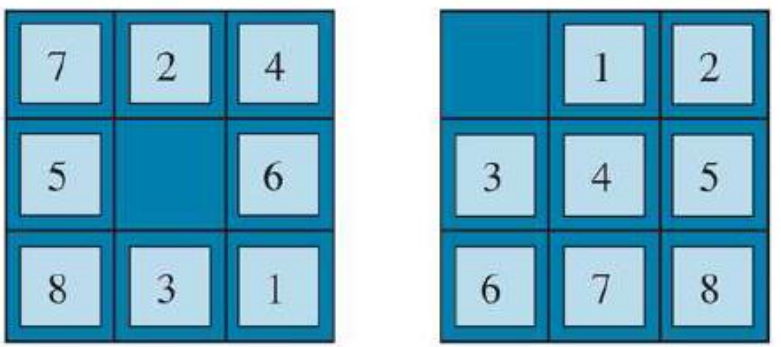
توابع مکاشفه ای

- جورچین ۸
- حل نوعی حدود ۲۲ مرحله
- ضریب انشعاب حدود ۳



- جستجوی کامل درختی حدود $3^{22} \approx 3.1 \times 10^{10}$ حالت بازدید می شود.

- h_1 تعداد خانه هائی که در مکانهای غلط قرار دارند.
- h_2 مجموع مسافتهای خانه ها از مکان هدفشان (فاصلهٔ مانهاتان یا بلوک شهر)
- مثال
- $h_1=8$
- $h_2=3+1+2+2+2+3+3+2=18$



Start State Goal State

- روشی برای ارزیابی مکاشفه ها ضریب انشعاب مؤثر b^* می باشد.
- فرض کنید کل رئوس بسط داده شده توسط A^* برای مسئله ای برابر N باشد و حل در عمق d باشد.
- b^* برابر ضریب انشعاب درخت یکنواختی با عمق d است که همان تعداد رأس داشته باشیم:

$$N + 1 = 1 + b^* + (b^*)^2 + \cdots + (b^*)^d .$$

	Search Cost (nodes generated)			Effective Branching Factor		
d	BFS	$A^*(h_1)$	$A^*(h_2)$	BFS	$A^*(h_1)$	$A^*(h_2)$
6	128	24	19	2.01	1.42	1.34
8	368	48	31	1.91	1.40	1.30
10	1033	116	48	1.85	1.43	1.27
12	2672	279	84	1.80	1.45	1.28
14	6783	678	174	1.77	1.47	1.31
16	17270	1683	364	1.74	1.48	1.32
18	41558	4102	751	1.72	1.49	1.34
20	91493	9905	1318	1.69	1.50	1.34
22	175921	22955	2548	1.66	1.50	1.34
24	290082	53039	5733	1.62	1.50	1.36
26	395355	110372	10080	1.58	1.50	1.35
28	463234	202565	22055	1.53	1.49	1.36

Data are averaged over 100 puzzles for each solution length

- اگر $h_2(n) \geq h_1(n)$ برای همهٔ n ها (هر دو قابل پذیرش)
- آنگاه h_2 بر h_1 چیرگی دارد.
- می دانیم اگر مکاشفه سازگار باشد هر رأس با $f(n) < C^*$ حتماً بسط داده می شود.
- پس هر رأس با $h(n) < C^* - g(n)$ حتماً بسط داده می شود.
- ممکن است رأسی باشد که $h_2(n)$ بزرگتر از سمت راست بوده ولی $h_1(n)$ کمتر باشد،
- بنابر این جستجو با h_1 رئوس بیشتری را بسط می دهد.

مسئلهٔ آسوده شده

- مسئله ای که قیود کمتری روی اعمال دارد مسئلهٔ آسوده شده نام دارد.
- هزینهٔ حل بهینه برای یک مسئلهٔ آسوده شده یک مکاشفهٔ قابل پذیرش برای مسئلهٔ اصلی است.

مسئله آسوده شده

- یک خانه می تواند از مربع A به مربع B حرکت کند اگر A مجاور B بوده و B خالی باشد.

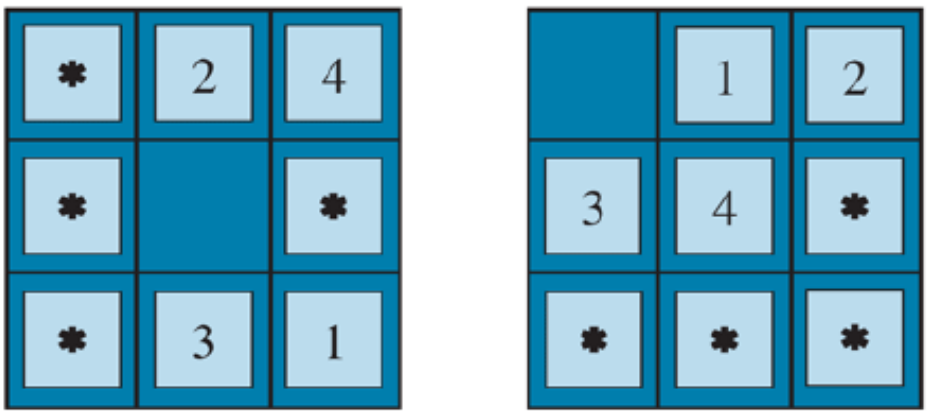
1. یک خانه می تواند از مربع A به مربع B حرکت کند اگر A مجاور B باشد.
2. یک خانه می تواند از مربع A به مربع B حرکت کند اگر B خالی باشد.
3. یک خانه می تواند از مربع A به مربع B حرکت کند.

- اگر مجموعه ای از مکاشفه ها وجود داشته باشند که هیچکدام بر دیگری چیرگی نداشته باشند می توانیم بصورت زیر از آنها استفاده کنیم:

$$h(n) = \max\{h_1(n), h_2(n), \cdots, h_m(n)\}$$

پایگاه دادهٔ الگو

- ذخیرهٔ هزینهٔ حل دقیق به زیرمسئله



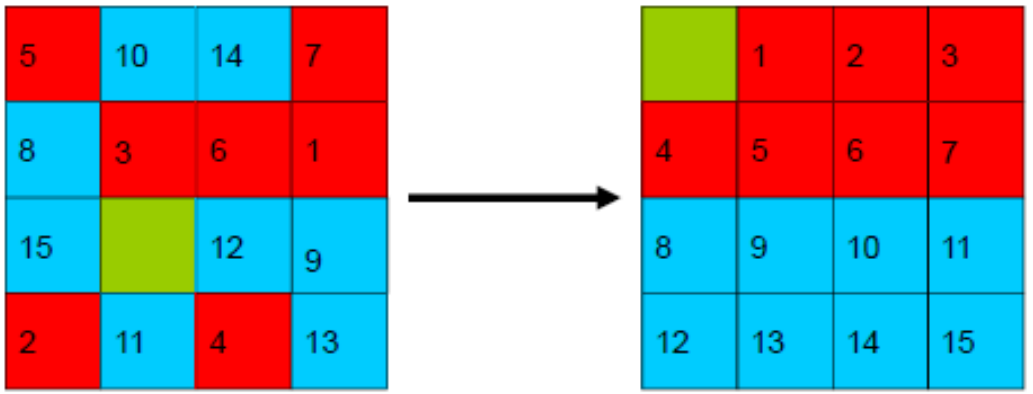
Start State Goal State

A subproblem of the 8-puzzle instance given in Figure 3.25. The task is to get tiles 1, 2, 3, 4, and the blank into their correct positions, without worrying about what happens to the other tiles.

- مسلماً حل زیر مسئله هزینهٔ کمتری از حل مسئله هنگامی که همهٔ کاشیها حضور دارند خواهد داشت.
- ذخیرهٔ هر زیر مسئلهٔ ممکن
- همهٔ پیکربندیهای ۴ کاشی مشخص و یک جای خالی
- ایجاد یک پایگاه داده از آنها
- استفاده از مقادیر آنها به عنوان مکاشفه با نگاه به پایگاه داده و یافتن زیر مسئلهٔ متناظر

- می توان زیر مسائل دیگری با ۴ کاشی دیگر مثلا ۵-۶-۷-۸ را در نظر گرفت.
- می توان آنها را همانگونه که گفته شد از هزینهٔ هر دو نیز استفاده نمود و هزینه حداکثر را استفاده نمود.
- ولی نمی توان آنها را با هم جمع کرد چون حرکت دو دسته با ۴ کاشی مستقل از هم نیست و هنگام حرکت ۴ کاشی از یک دسته کاشیهای دسته دیگر نیز جابجا می شوند.

- اگر هنگام شمارش حرکات مورد نیاز هر دسته فقط حرکات کاشیهای آن دسته شمرده شوند و دو دسته منفصل باشند می توان هزینهٔ آن دو دسته را با هم جمع نمود.
- فاصلهٔ مانهاتان حالت خاصی از این مکاشفه است که هر الگو فقط شامل یک کاشی است.
- به این مکاشفه پایگاه دادهٔ الگوی منفصل (disjoint pattern database) گفته می شود.



20 moves needed to solve red tiles

25 moves needed to solve blue tiles

Overall heuristic is sum, or 20+25=45 moves

Dan Weld Slides

خلاصه

- IDA*
- جستجوی بهترین نخست بازگشتی
- SMA*
- A* وزندار
- توابع مکاشفه ای
- چیرگی
- ساخت توابع مکاشفه ای
- پایگاه دادهٔ الگو


والسلام

- دقت نمائید که پاورپوینت ابزاری جهت کمک به یک ارائهٔ شفاهی می باشد و **به هیچ وجه یک جزوهٔ درسی نیست** و شما را از خواندن مراجع درس بی نیاز نمی کند.
- لذا حتماً مراجع اصلی درس را مطالعه نمائید.

- دقت نمائید که پاورپوینت ابزاری جهت کمک به یک ارائهٔ شفاهی می باشد و **به هیچ وجه یک جزوهٔ درسی نیست** و شما را از خواندن مراجع درس بی نیاز نمی کند.
- لذا حتماً مراجع اصلی درس را مطالعه نمائید.