هوش مصنوعی

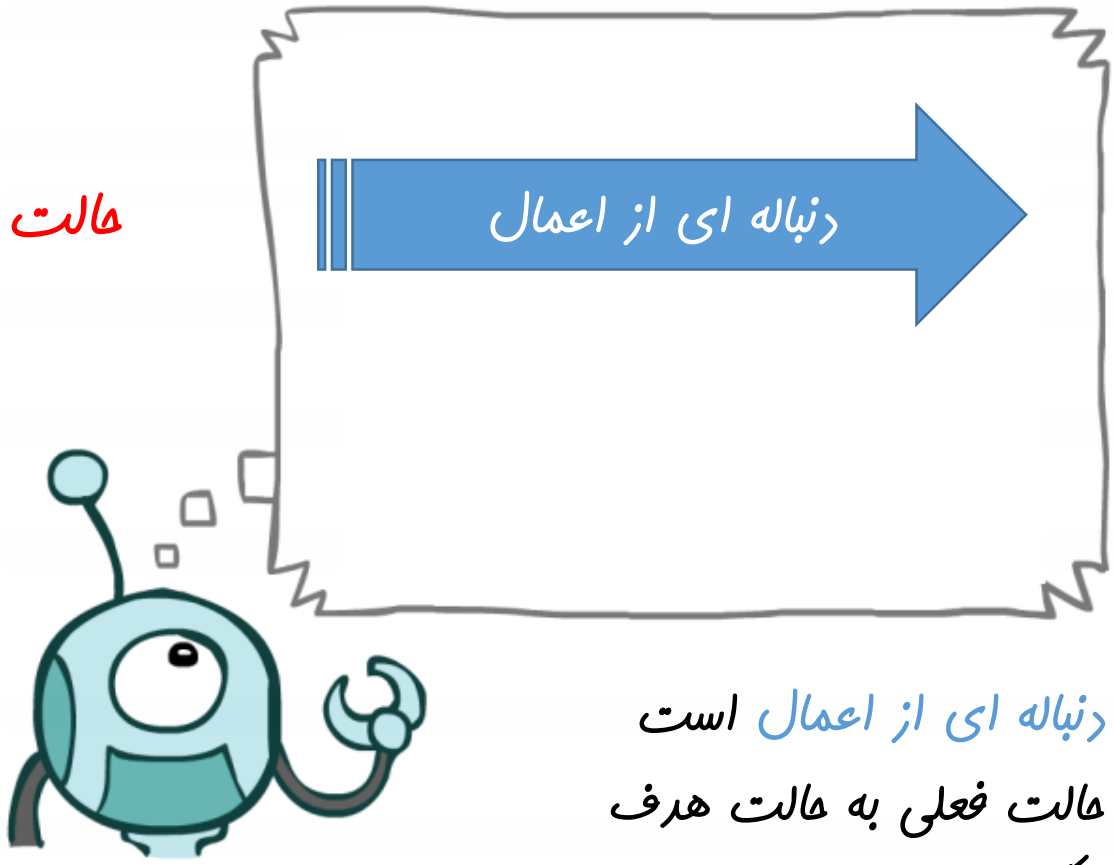
(جستجو-بخش دوم)

صادق اسکندری - دانشکده علوم ریاضی، گروه علوم کامپیوتر

eskandari@guilan.ac.ir

یادآوری ...

در مسائل برنامه ریزی، هدف تغییر وضعیت جهان از حالت فعلی به یک حالت هدف است.



وظیفه یک عامل برنامه ریز، یافتن دنباله ای از اعمال است که با انجام آنها، وضعیت جهان از حالت فعلی به حالت هدف تبدیل می شود. به این عمل جستجو گفته می شود.

یادآوری ...

یک مسئله جستجو شامل موارد زیر است:

۱- یک فضای حالت (State Space)

۲- مجموعه اعمال (Actions)

۳- یک تابع بعدی (Successor Function)

٤- یک تابع هزینه مسیر

٥- یک حالت اولیه (Initial State)

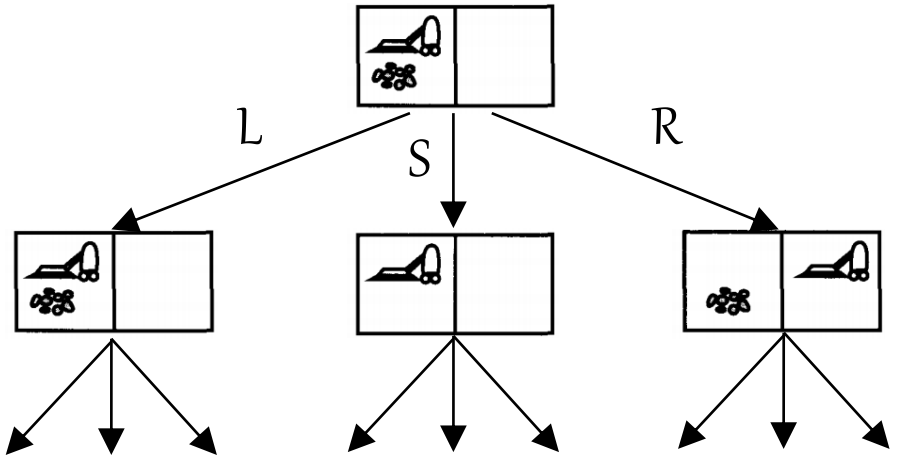
٦- یک تابع تست هرف (Goal Test Function)

یادآوری ...

یک درخت جستجو یک درخت اگر-آنگاه برای برنامه ها و نتایج آنها است.

حالت اولیه ریشه درخت است

فرزندان معادل با حالت های بعدی هستند



حالت فعلی

آینده های ممکن

یادآوری ...

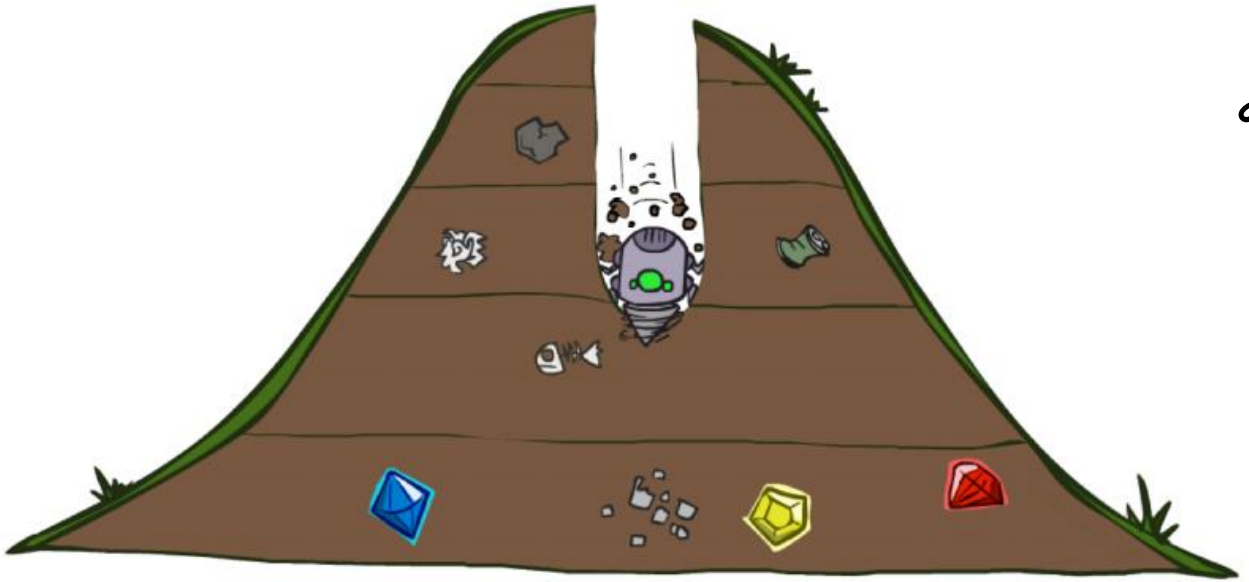
```
function GENERAL-SEARCH( problem, QUEUING-FN) returns a solution, or failure

    nodes ← MAKE-QUEUE(MAKE-NODE(INITIAL-STATE[problem]))
    loop do
        if nodes is empty then return failure
        node ← REMOVE-FRONT(nodes)
        if GOAL-TEST[problem] applied to STATE(node) succeeds then return node
        nodes ← QUEUING-FN(nodes, EXPAND(node, OPERATORS[problem]))
    end
```

در زمان ایجاد درخت جستجو، تعدادی گره جدید تولید شده و منتظر بسط داده شدن هستند. اینکه کدامیک باید زودتر بررسی و بسط داده شوند، استراتژی جستجو نامیده می شود.

یادآوری ...

استراتژی عمق اول



در این استراتژی، همواره عمیق ترین گره زودتر از بقیه بسط داده می شود.

در این استراتژی، فرزندان تولید شده از بسط گرهها، به ابتدای صف اضافه می شوند. به عبارت دیگر، صف همانند یک پشته عمل می کند.

یادآوری ...

DFS چه گرههایی را بسط می دهد؟

این الگوریتم همواره یک پیشوند چپ از کل درخت جستجو را بسط می دهد

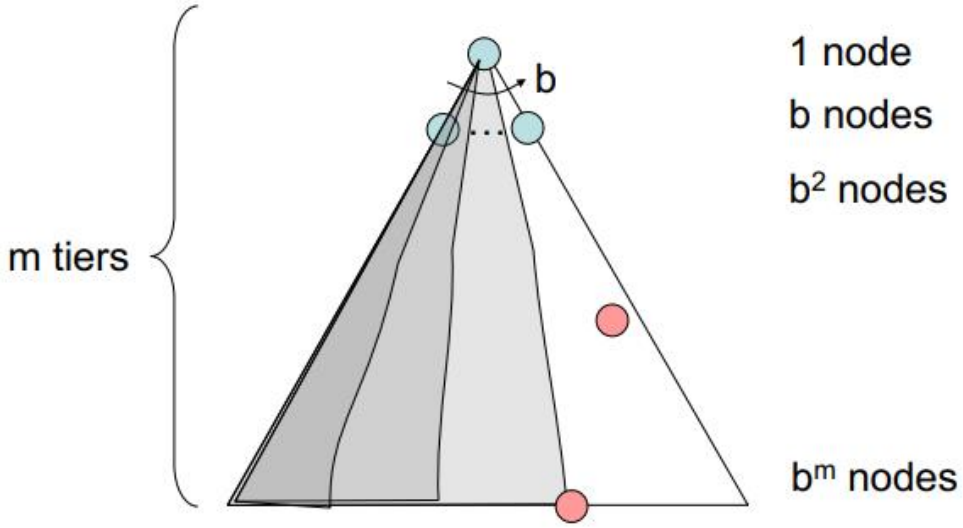
در صورتی که حداکثر عمق محدود باشد، می تواند کل درخت جستجو را پردازش کند.

پیچیدگی زمانی

در صورتی که حداکثر عمق محدود باشد، دارای پیچیدگی زمانی $O(b^m)$ است.

پیچیدگی فضایی

تنها همموالدهای گرههای تا ریشه را نگهداری می کند. بنابراین پیچیدگی فضایی $O(bm)$ است.



آیا DFS کامل است؟

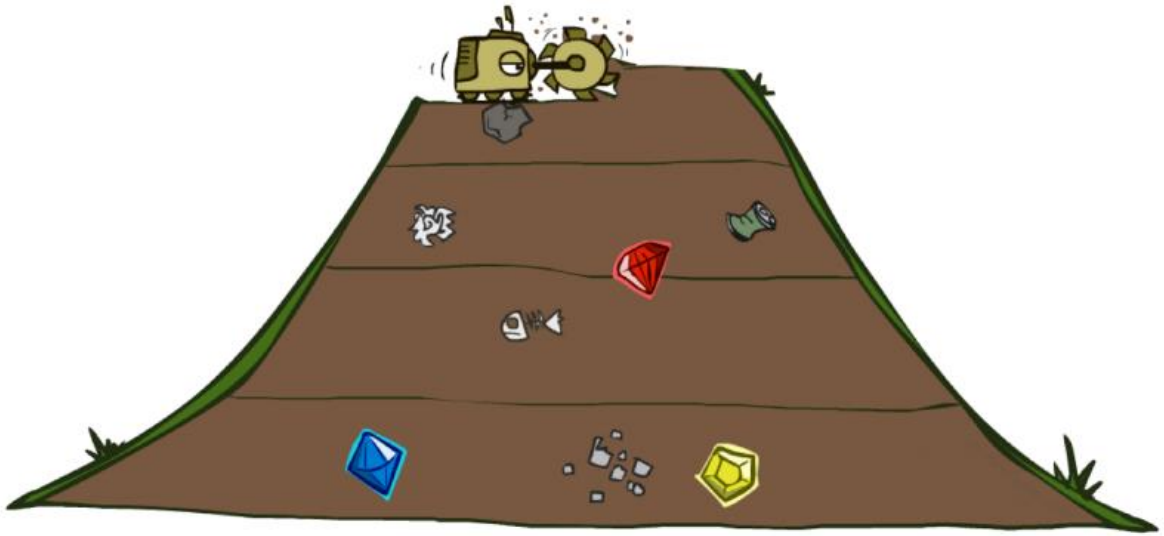
خیر، m می تواند بینهایت باشد. تنها در صورتی که m محدود باشد یا از گرههای تکراری جلوگیری کنیم، این استراتژی کامل خواهد بود.

آیا DFS بهینه است؟

خیر، این الگوریتم همواره سمت چپ ترین گره را بدون در نظر گرفتن عمق یا هزینه می یابد.

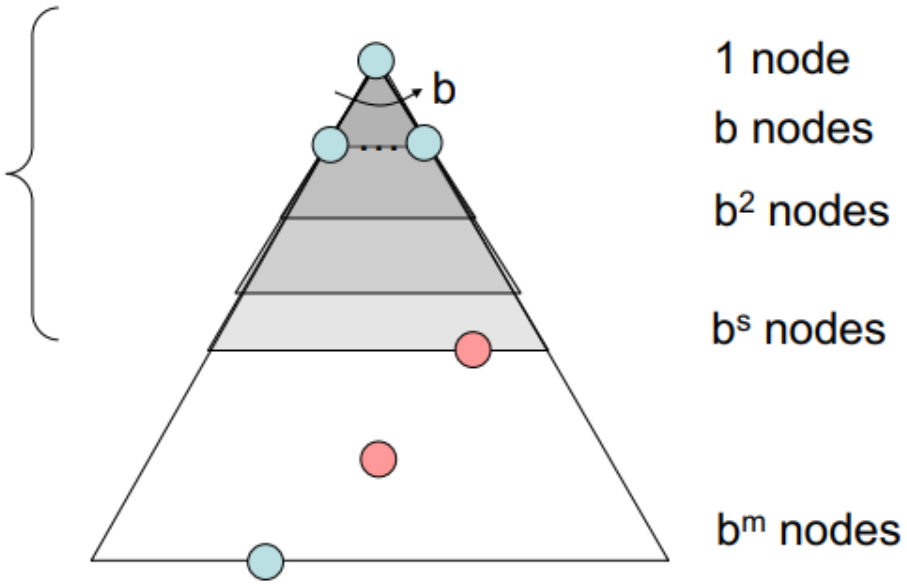
استراتژی های جستجو: سطح اول

در این استراتژی، همواره سطحی ترین گره بسط داده می شود.



در این استراتژی، فرزندان تولید شده از بسط گرهها، به انتهای صف اضافه می شوند.

ارزیابی استراتژی جستجوی سطح اول



BFS چه گرههایی را بسط می دهد؟

این الگوریتم همواره تمامی گرههای بالاتر از سطحی ترین پاسخ را پردازش می کند.

پیچیدگی زمانی

در صورتی که سطحی ترین پاسخ در عمق s باشد، پیچیدگی زمانی آن $O(b^s)$ خواهد بود

پیچیدگی فضایی

همواره گره های آخرین سطح در صف منتظر خواهند بود. بنابراین، پیچیدگی فضایی آن $O(b^s)$ است.

آیا BFS کامل است؟

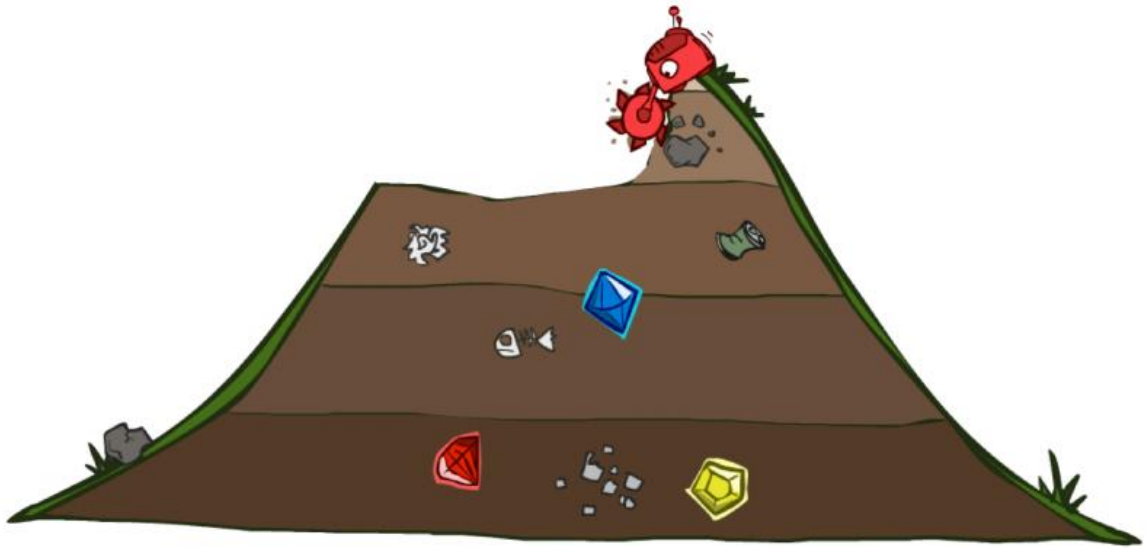
بله

آیا BFS بهینه است؟

در صورتی که هزینه مسیر یک تابع صعودی یکنواخت از عمق باشد، این استراتژی بهینه است و در غیر این صورت خیر

استراتژی های جستجو: جستجو با هزینه یکنواخت

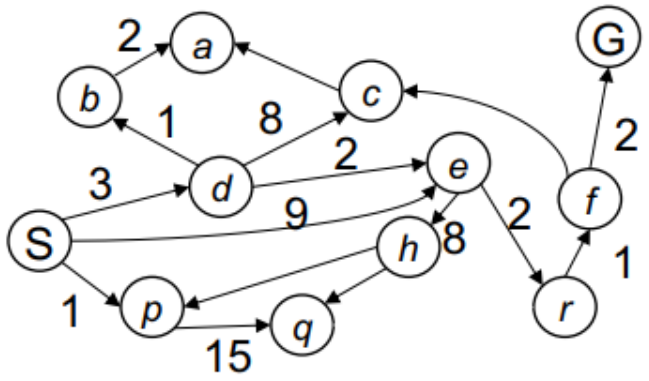
در این استراتژی، همواره ارزان ترین گره بسط داده می شود.



در این استراتژی، فرزندان تولید شده از بسط گرهها، به گونه ای به صف اضافه می شوند که صف از نظر هزینه مسیر مرتب باشد.

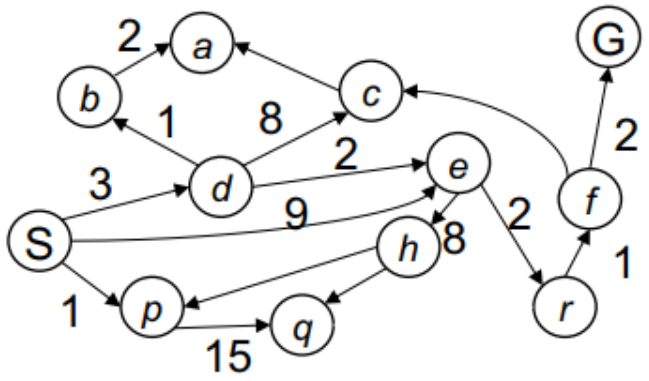
استراتژی های جستجو: جستجو با هزینه یکنواخت

مثال

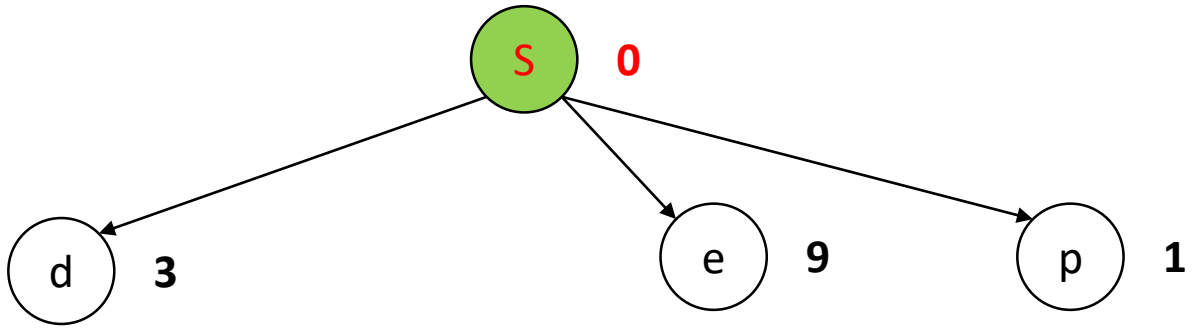


S 0

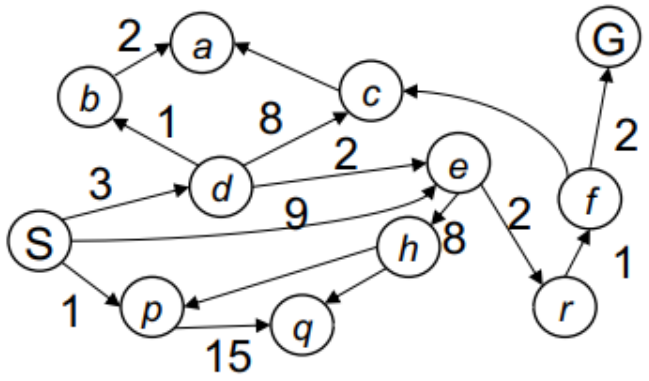
استراتژی های جستجو: جستجو با هزینه یکنواخت



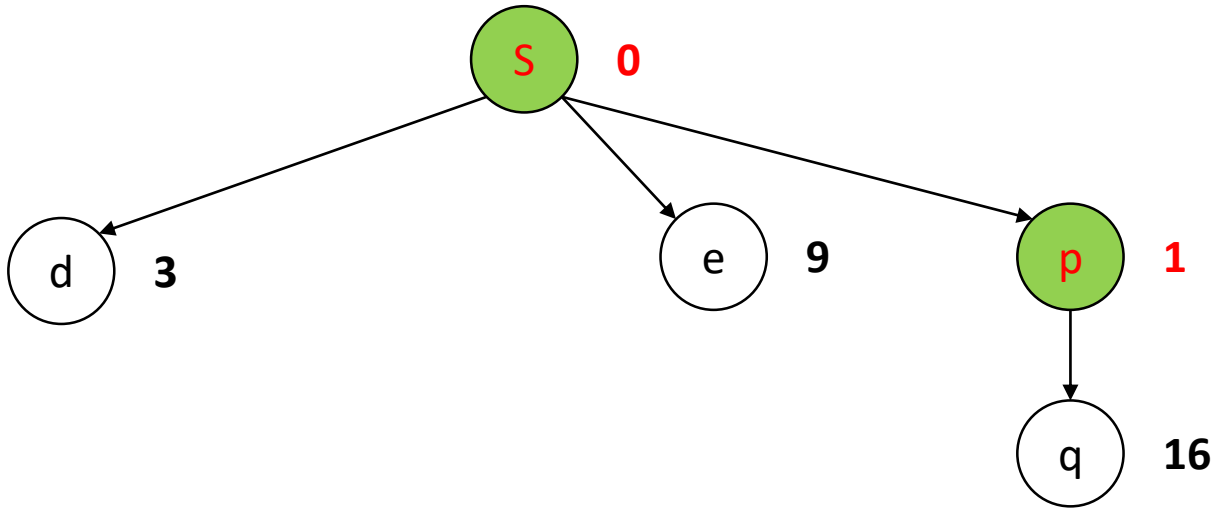
مثال



استراتژی های جستجو: جستجو با هزینه یکنواخت

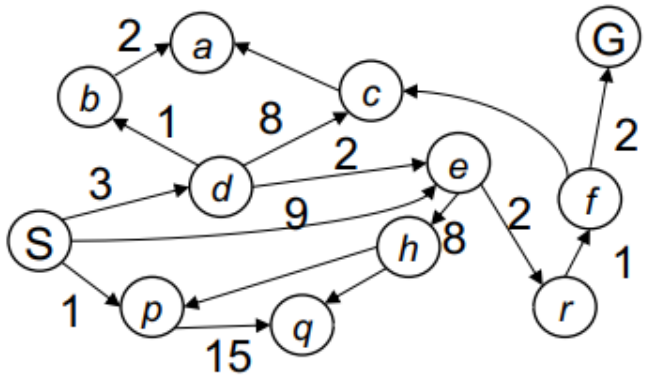


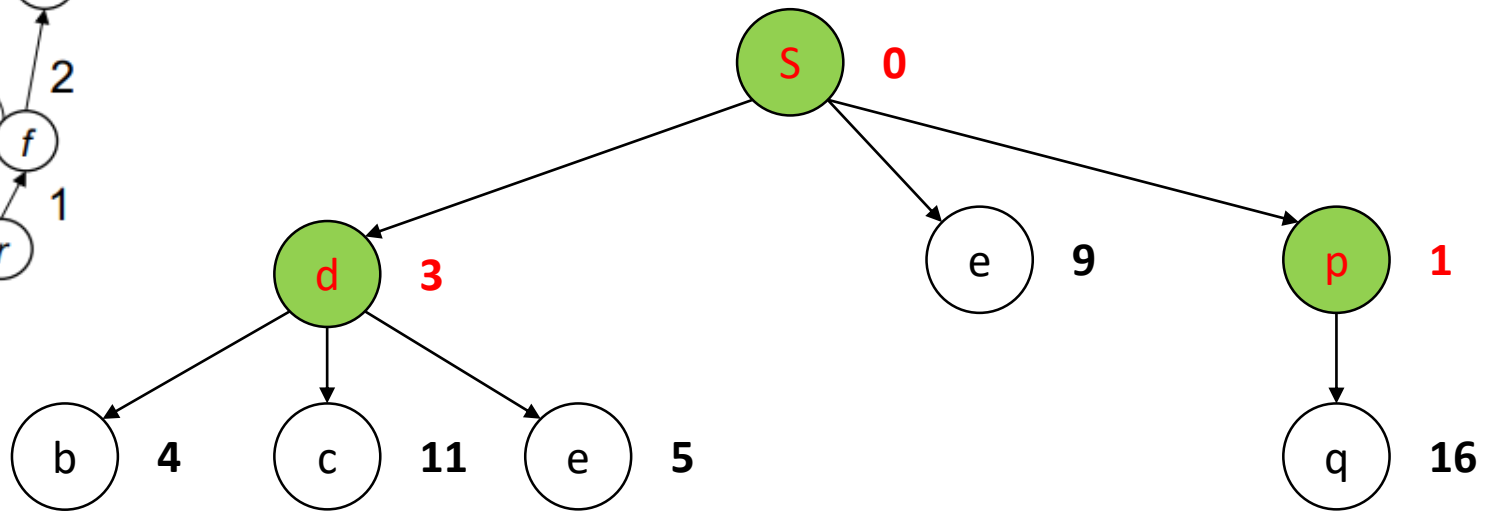
مثال



استراتژی های جستجو: جستجو با هزینه یکنواخت

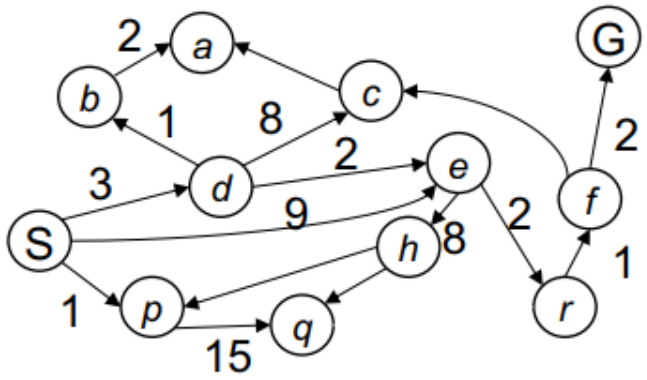
مثال

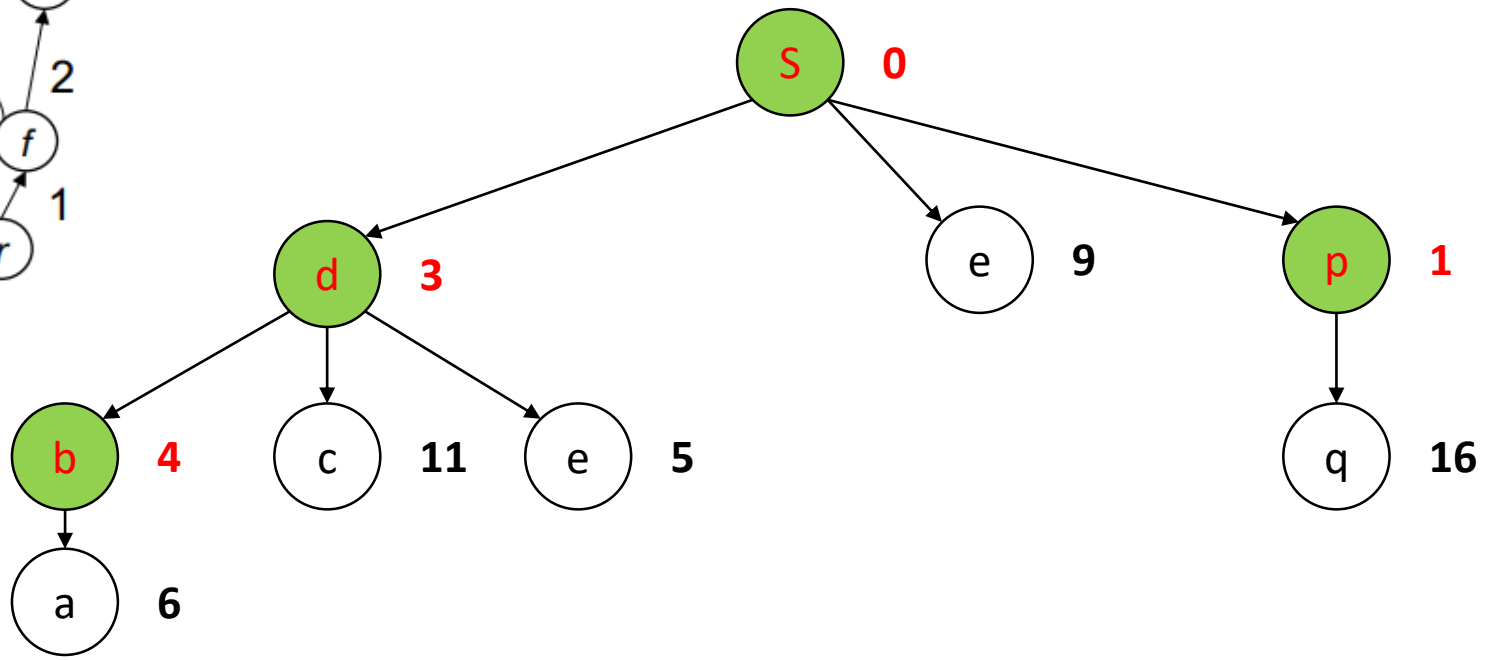




استراتژی های جستجو: جستجو با هزینه یکنواخت

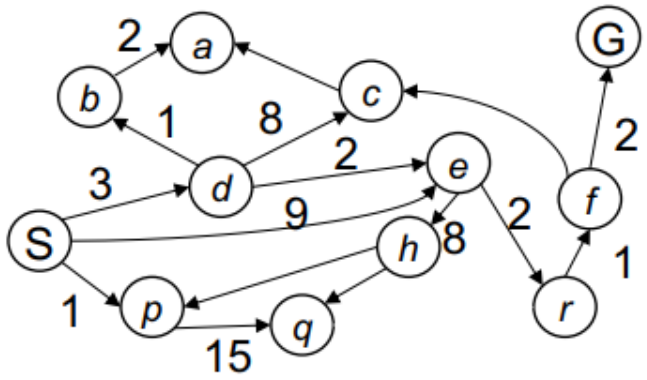
مثال

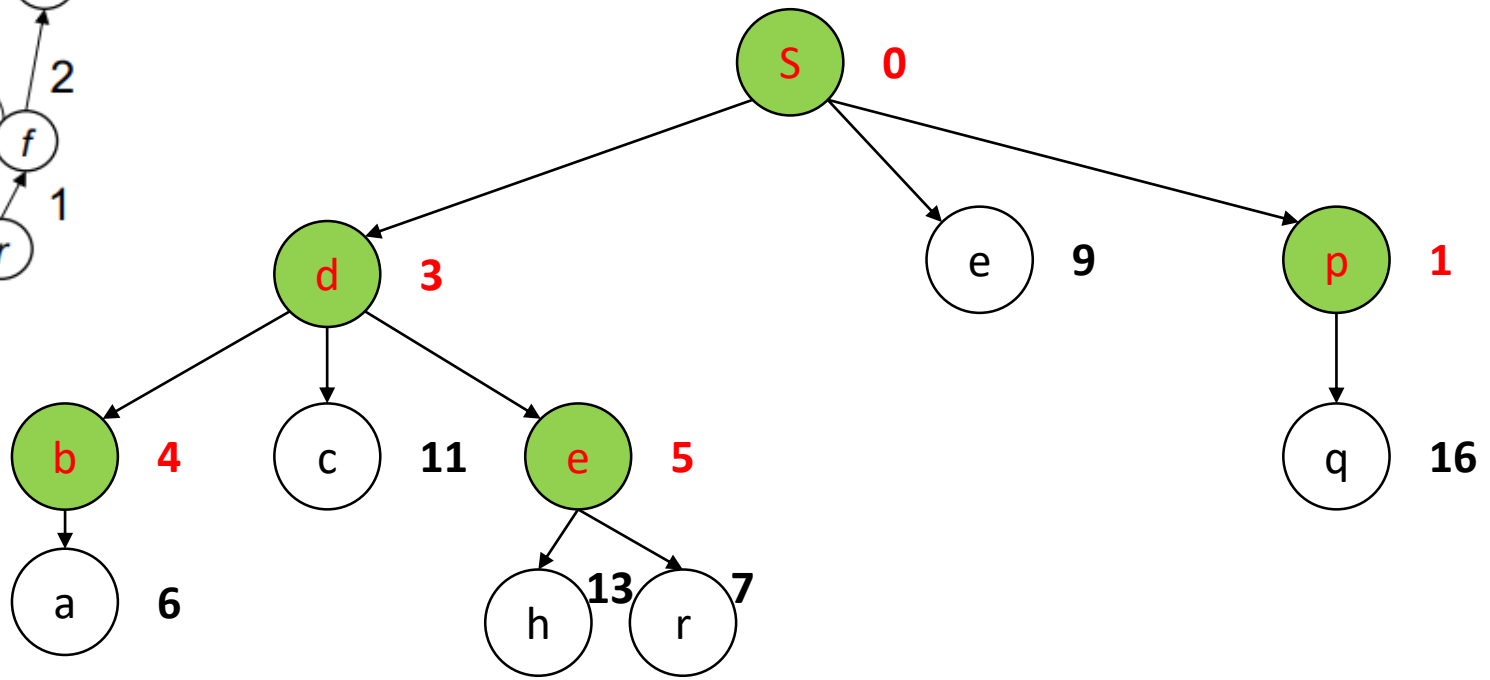




استراتژی های جستجو: جستجو با هزینه یکنواخت

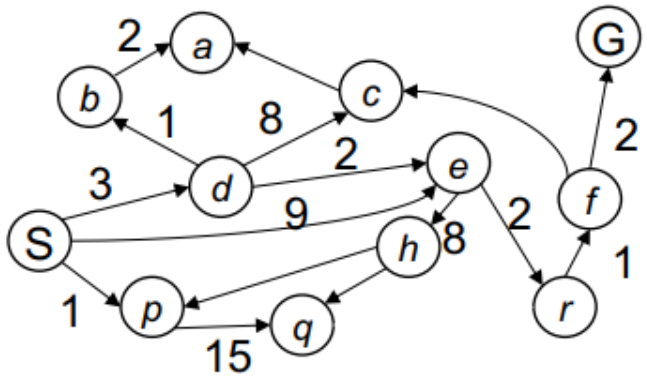
مثال

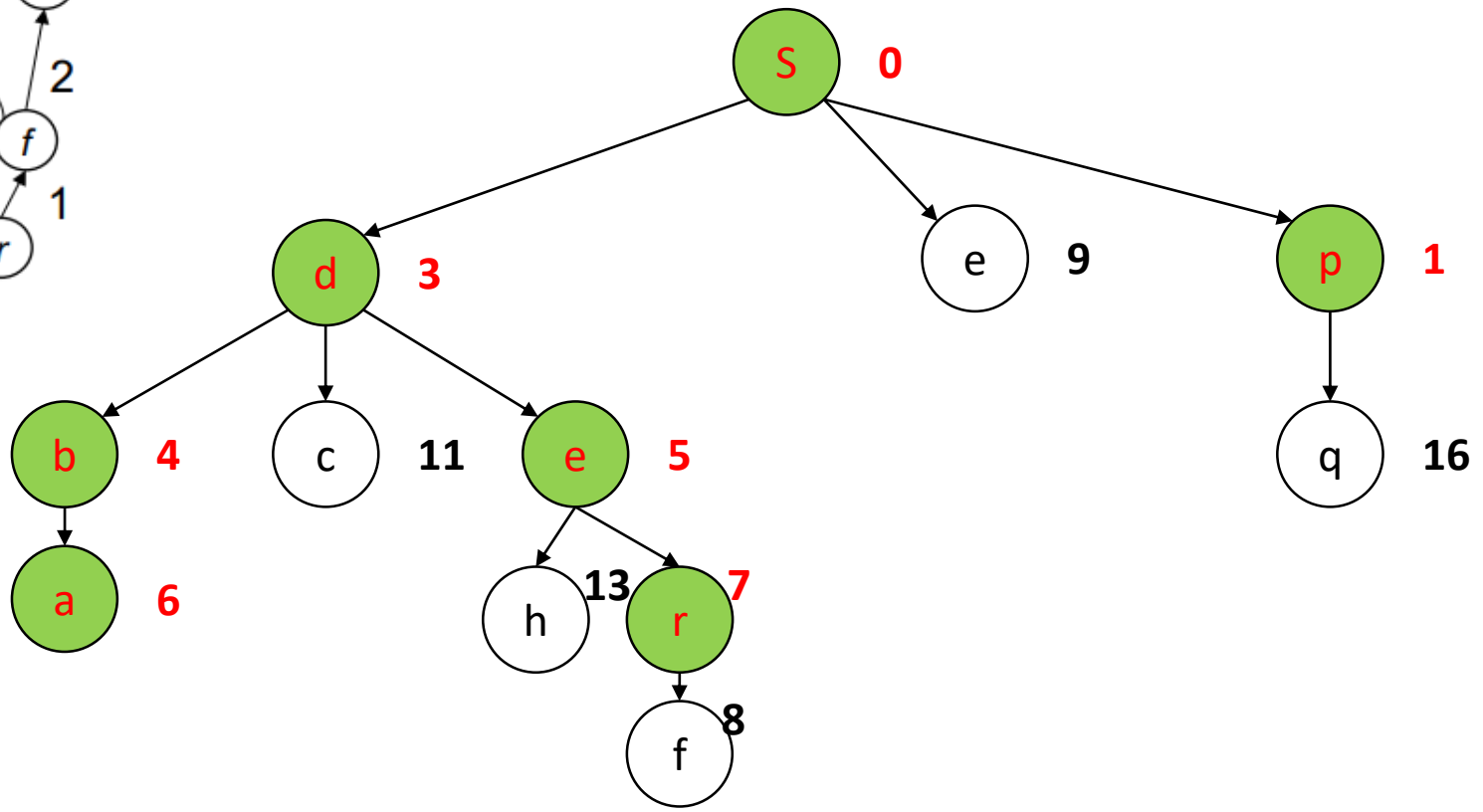




استراتژی های جستجو: جستجو با هزینه یکنواخت

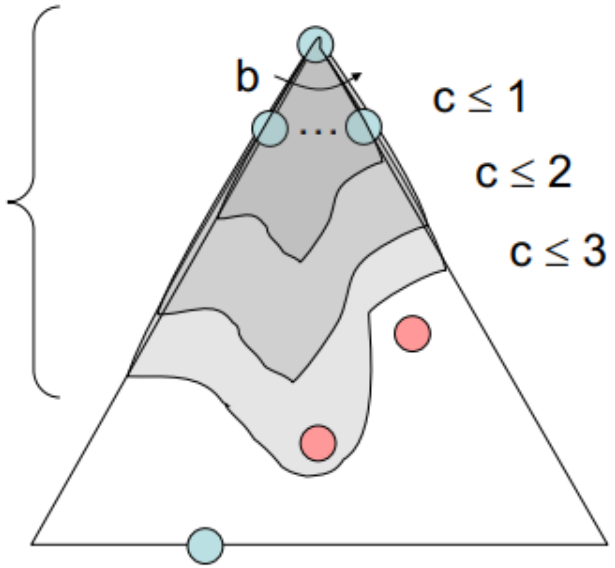
مثال





ادامه به عنوان تمرین

ارزیابی استراتژی جستجوی هزینه یکنواخت



UCS چه گرههایی را بسط می دهد؟

این الگوریتم همواره تمامی گرهها با هزینه کمتر از ارزان ترین جواب را پردازش می کند.

اگر هزینه ارزانترین جواب C^* و حداقل هزینه یالها ϵ باشد، آنگاه عمق موثر تقریبا برابر C^*/ϵ خواهد بود.

پیچیدگی زمانی

$O(b^{\frac{C^*}{\epsilon}})$

پیچیدگی فضایی

$O(b^{\frac{C^*}{\epsilon}})$

آیا UCS کامل است؟

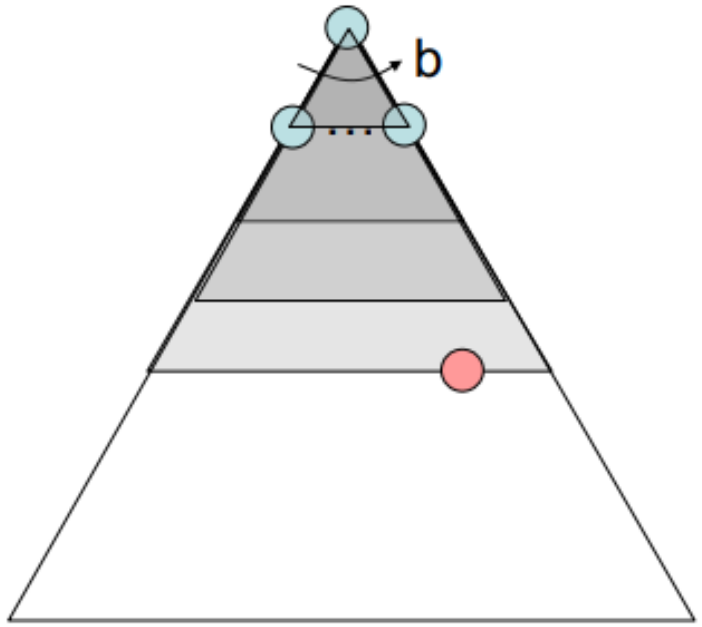
بله

آیا UCS بهینه است؟

بله

استراتژی های جستجو: عمیق سازی تکراری

این استراتژی سعی در ترکیب ویژگی های خوب استراتژی های BFS و DFS دارد.



از استراتژی DFS پیچیدگی فضایی

از استراتژی BFS کامل بودن و سطحی ترین پاسخ

الگوریتم کلی:

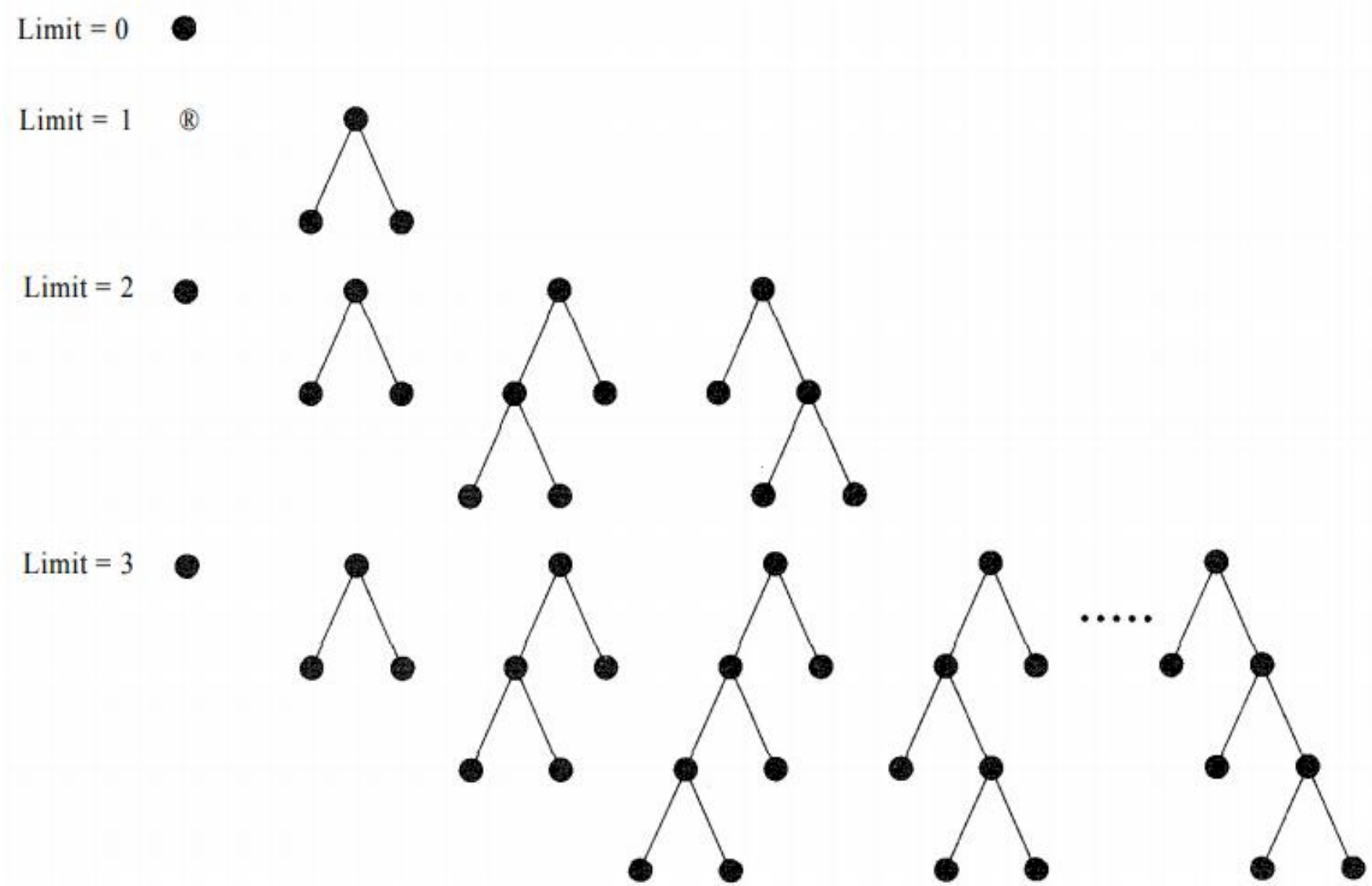
حداکثر عمق را صفر در نظر گرفته و جستجوی DFS را اجرا کن

در صورتی که پاسخ پیدا نشد، حداکثر عمق را یک در نظر گرفته و جستجوی DFS را اجرا کن

در صورتی که پاسخ پیدا نشد، حداکثر عمق را دو در نظر گرفته و جستجوی DFS را اجرا کن

...

استراتژی های جستجو: عمیق سازی تکراری



استراتژی های جستجو: عمیق سازی تکراری

سوال: با توجه به اینکه در استراتژی عمیق سازی تکراری یک گره ممکن است چندین بار بررسی و بسط داده شود، آیا افزایش پیچیدگی زمانی آن قابل تحمل است؟

فرض کنید پاسخ در عمق d قرار داشته باشد. آنگاه با استفاده از روش سطح اول، تعداد کل گرههای پردازش شده برابر است با:

$$1 + b + b^2 + \cdots + b^{d-1} + b^d$$

با فرض اینکه ی $d = 5$ و $b = 10$، تعداد گرهها برابر است با: $1 + 10 + 100 + \cdots + 10000 + 100000 = 111111$

حال اگر از روش IDS استفاده کنیم، گرههای آخرین عمق یک بار، گرههای قبل از آخر، دو بار و ... و گره ریشه $d + 1$ بار پردازش خواهند شد. بنابراین، تعداد کل گرههای پردازش شده در این روش برابر است با:

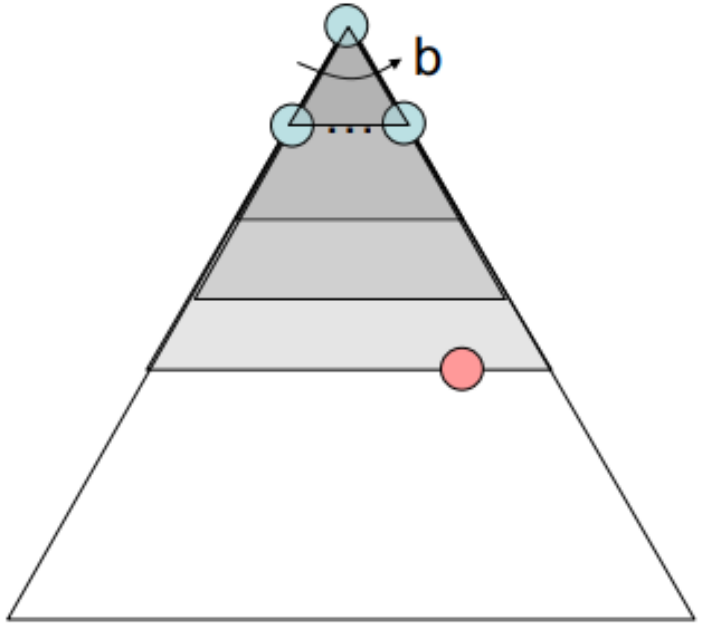
$$(d+1)1 + (d)b + (d-1)b^2 + \cdots + (2)b^{d-1} + (1)b^d$$

با فرض اینکه ی $d = 5$ و $b = 10$، تعداد گرهها برابر است با: $6 + 50 + 400 + \cdots + 20000 + 100000 = 123456$

با استفاده از روش IDS، تعداد گرههای پردازش شده فقط یازده درصد افزایش یافت!

دلیل این رخداد این است که بیشتر گرههای درخت جستجو در آخرین سطح قرار دارند و این گرهها فقط یکبار بسط داده می شوند.

ارزیابی استراتژی جستجوی عمیق سازی تکراری



IDS چه گرههایی را بسط می دهد؟

مشابه الگوریتم BFS

پیچیدگی زمانی

$O(b^d)$

پیچیدگی فضایی

$O(bd)$ (مشابه DFS)

آیا IDS کامل است؟

بله

آیا IDS بهینه است؟

در صورتی که هزینه مسیر یک تابع صعودی یکنواخت از عمق باشد، این استراتژی بهینه است و در غیر این صورت خیر (مشابه BFS)

استراتژی های جستجو

استراتژی های جستجوی کورکورانه یا ناآگاهانه (Uninformed)

DFS

BFS

UCS

IDS

از هیچ اطلاعات اضافی در جستجوی خود استفاده نمی کنند

استراتژی های جستجوی غیر کورکورانه یا آگاهانه (Informed)

Greedy Search

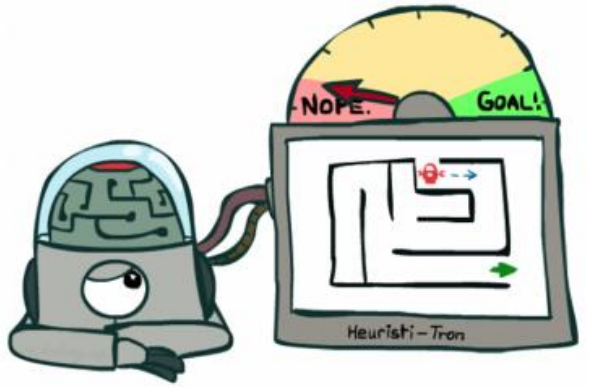
A* Search

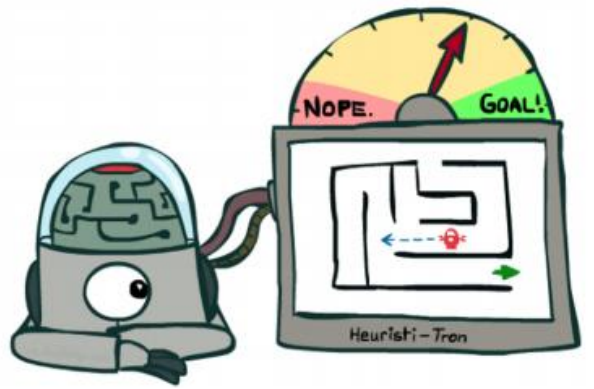
Graph Search

از اطلاعات اضافی در جستجوی خود استفاده می کنند.
این اطلاعات در قالب یک تابع ابتکاری نشان داده می شود.

استراتژی های جستجوی آگاهانه: تابع ابتکاری

یک *تابع ابتکاری* (Heuristic Function) میزان نزدیکی یک حالت به حالت هدف را **تخمین** می زند.

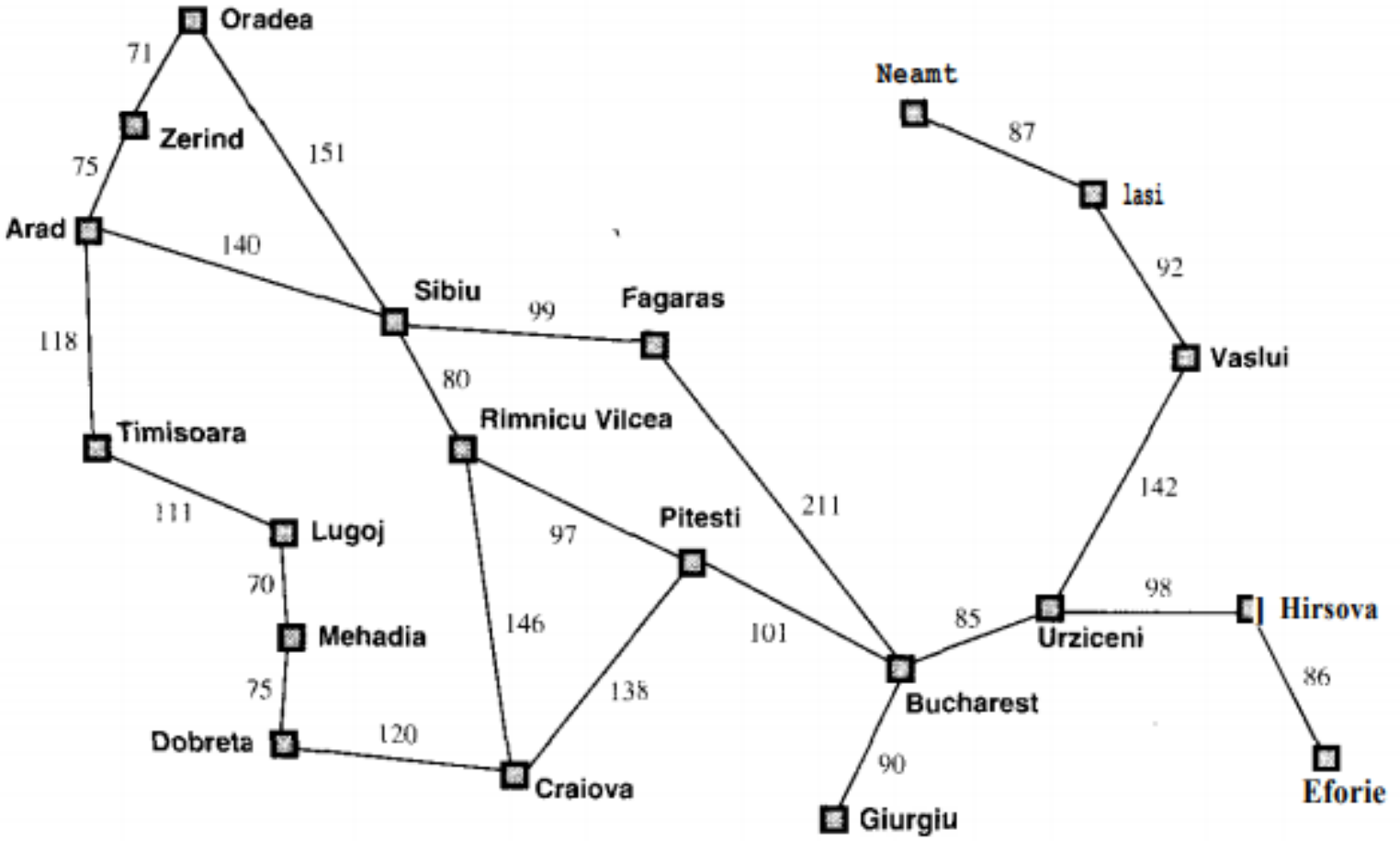




یک تابع ابتکاری کلی وجود ندارد و برای هر مسئله باید به شکل جدا طراحی شود.

استراتژی های جستجوی آگاهانه: تابع ابتکاری

مثال: تابع ابتکاری برای مسئله مسافرت در رومانی



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)

استراتژی های جستجوی آگاهانه: جستجوی حریصانه

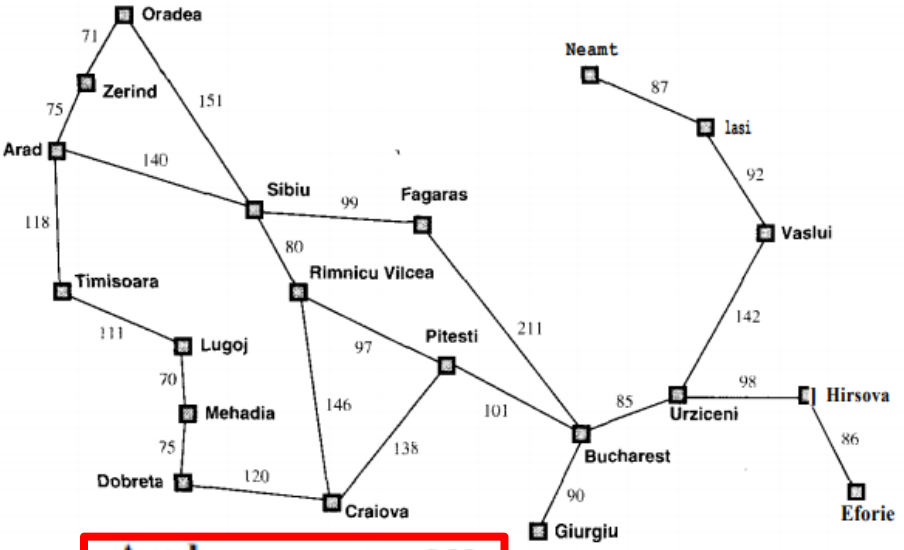
در این استراتژی، همواره گرهی بسط داده می شود که کمترین فاصله تخمینی را با هدف داشته باشد.

در این استراتژی، فرزندان تولید شده از بسط گرهها، به گونه ای به صف اضافه می شوند که صف از نظر تابع ارزیاب زیر مرتب باشد:

$$f(n) = h(n)$$

استراتژی های جستجوی آگاهانه: جستجوی حریصانه

مثال: تابع ابتکاری برای مسئله مسافرت در رومانی

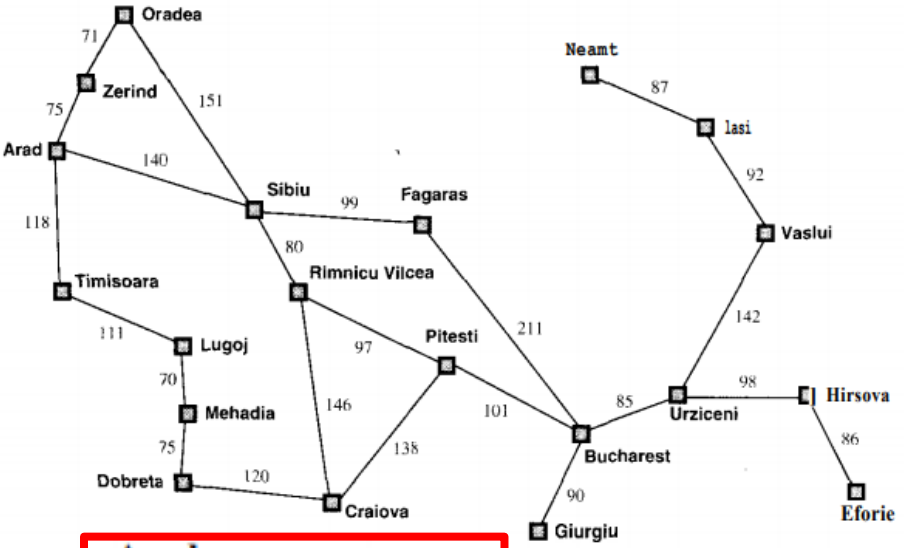


Arad	**366**
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	**226**
Lugoj	244
Mehadia	241
Neamt	**234**
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

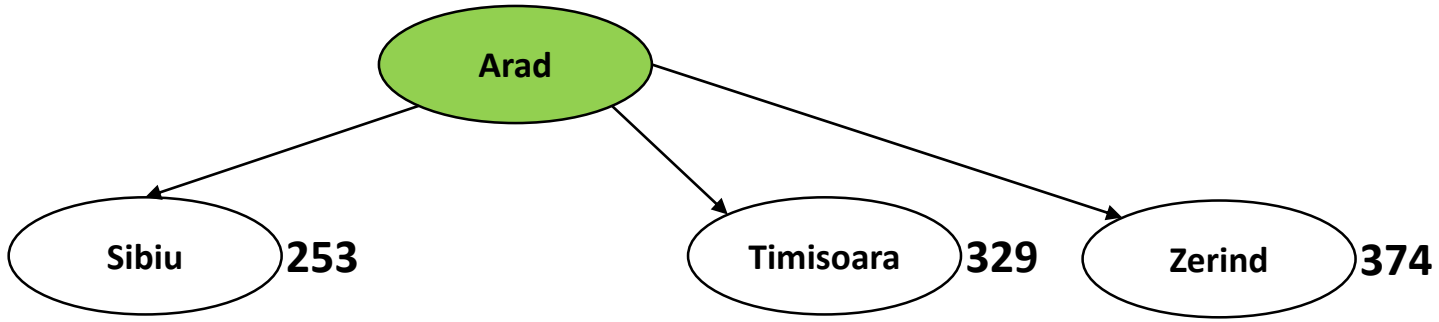
Arad **366**

استراتژی های جستجوی آگاهانه: جستجوی حریصانه

مثال: تابع ابتکاری برای مسئله مسافرت در رومانی

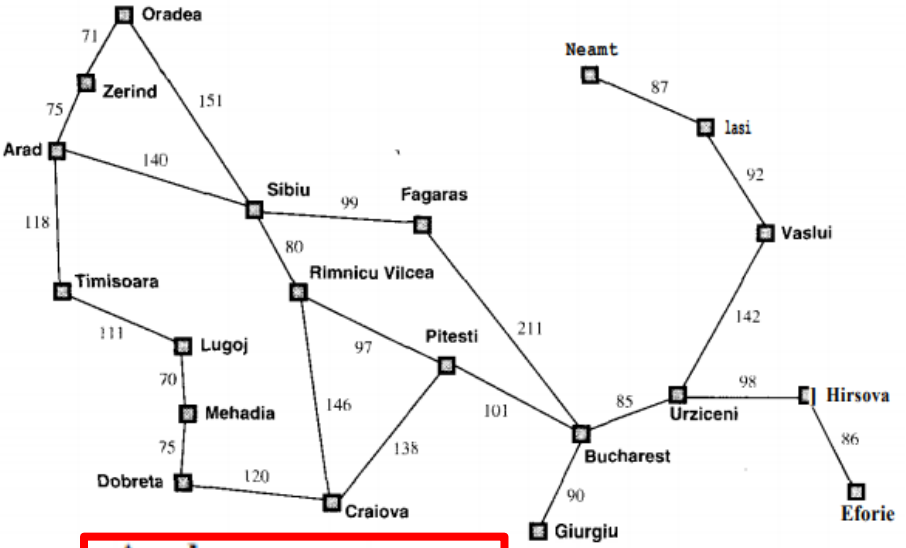


Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

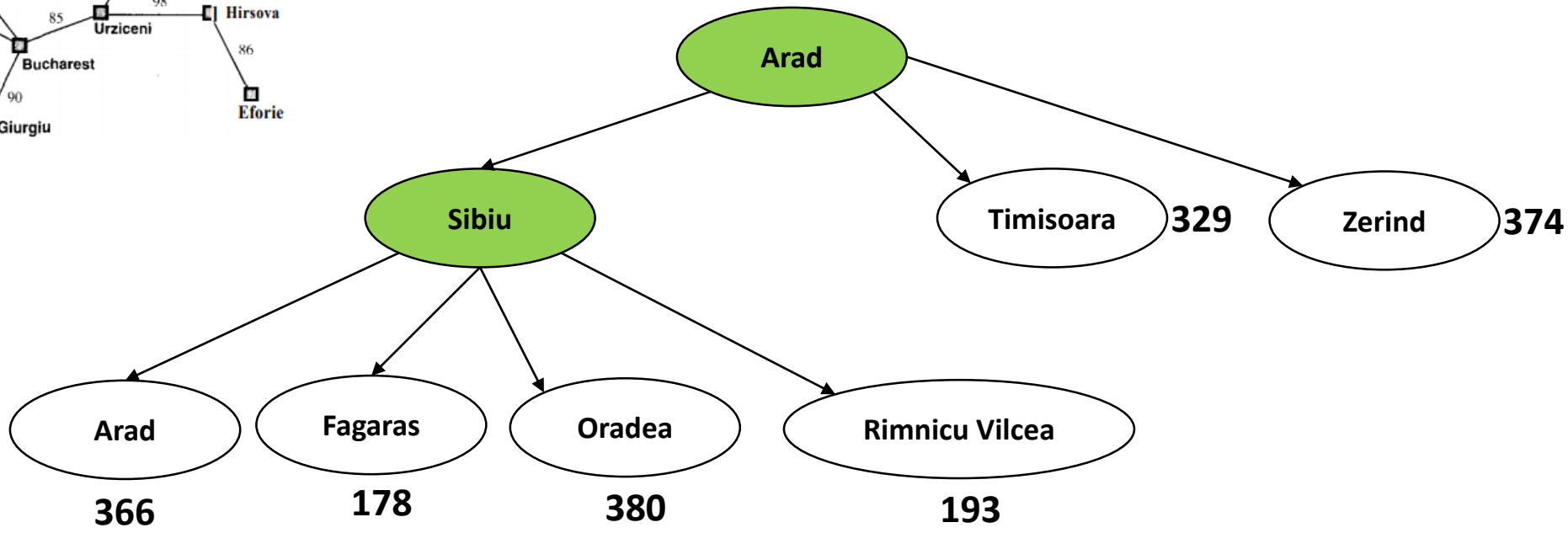


استراتژی های جستجوی آگاهانه: جستجوی حریصانه

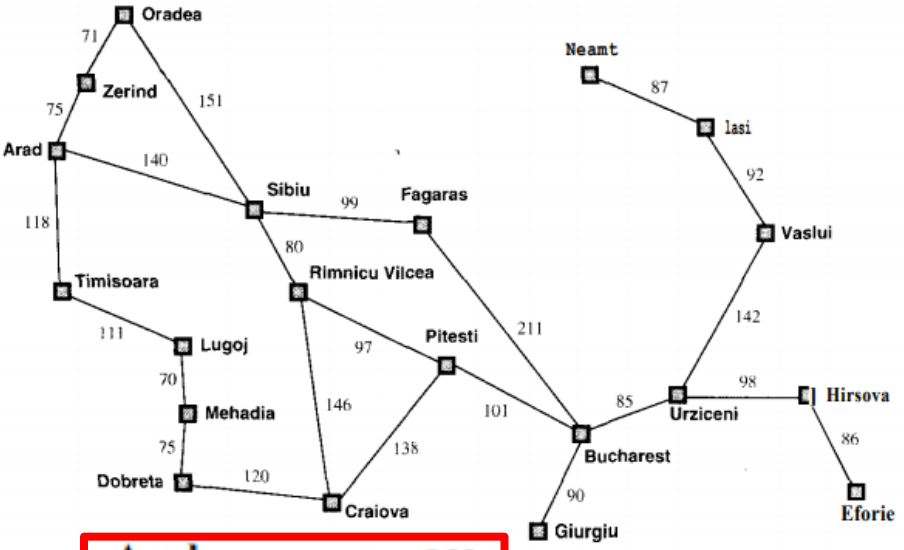
مثال: تابع ابتکاری برای مسئله مسافرت در رومانی



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

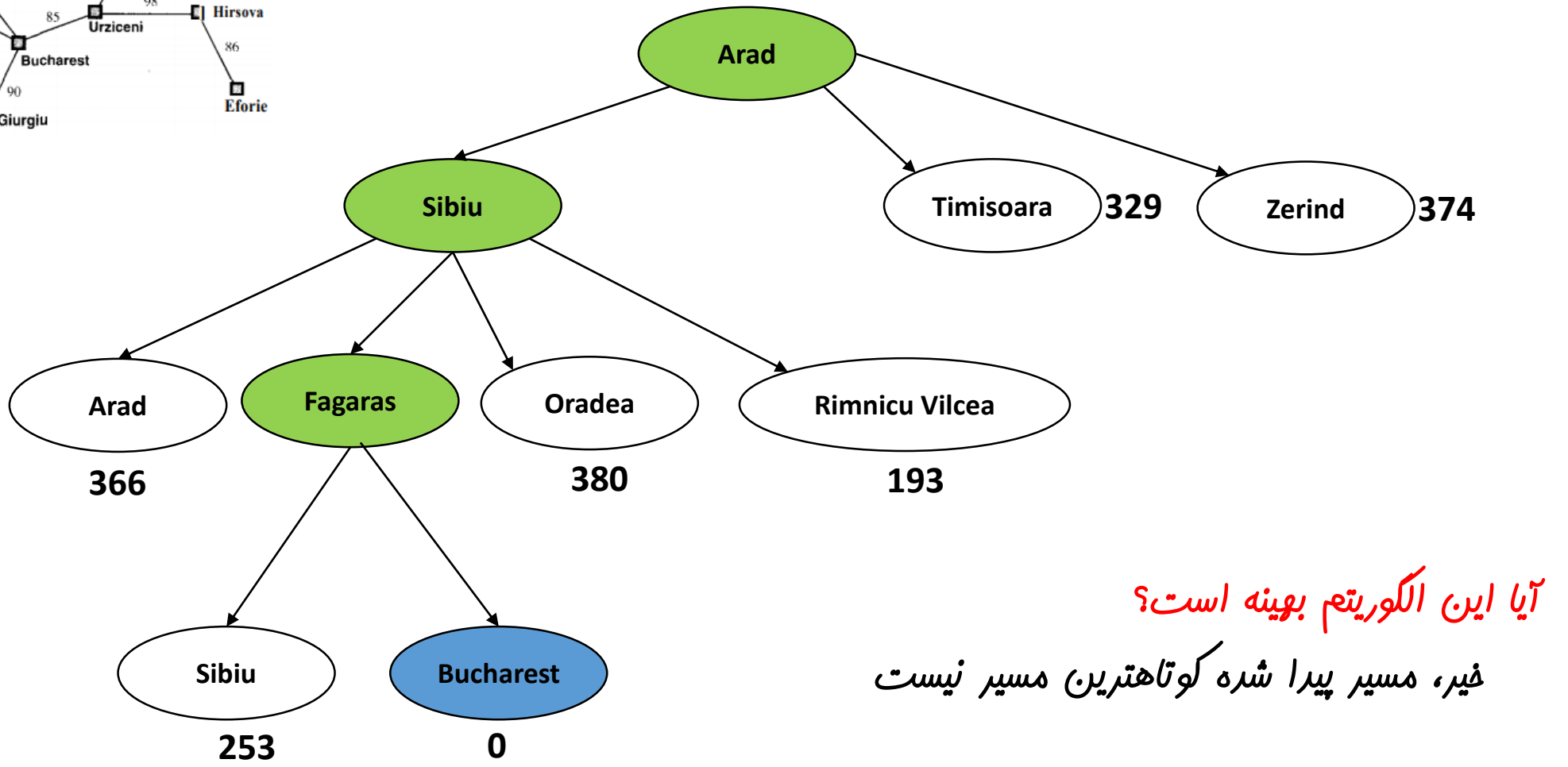


استراتژی های جستجوی آگاهانه: جستجوی حریصانه



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

مثال: تابع ابتکاری برای مسئله مسافرت در رومانی



آیا این الگوریتم بهینه است؟

خیر، مسیر پیدا شده کوتاهترین مسیر نیست

جسله آینده: جستجوی A*