به نام خدا

هوش مصنوعی

فصل چهارم

جستوجو آگاهانه و اکتشاف

مقدمه

❖ روش های جستوجو آگاهانه نیز سعی می کنند از وضعیت اولیه به وضعیت هدف برسند اما برای این کار از دانشی که در مورد مسئله به آنها داده می شود استفاده می کنند .

❖ این دانش در قالب تابعی به نام تابع هیوریستیک به الگوریتم های جستوجو داده می شود.

❖ روش های جستوجو آگاهانه
- 1- روش های بهترین جستوجو
- 2- روش های جستوجو محلی

چند تعریف :

❖**تابع ارزیابی :** تابع ارزیابی f(n)،شایستگی یک وضعیت هدف را تخمین می زند.

❖**تابع هیوریستیک(تابع اکتشافی) :** تابع هیوریستیک h(n) ، تخمینی از هزینه رسیدن به هدف از حالتی خاص است. هزینه تخمینی ارزان ترین مسیر از گره n به گره هدف

❖**تابع بهترین مسیر h*(n) :** هزینه واقعی ارزان ترین مسیر از گره n تا گره هدف .

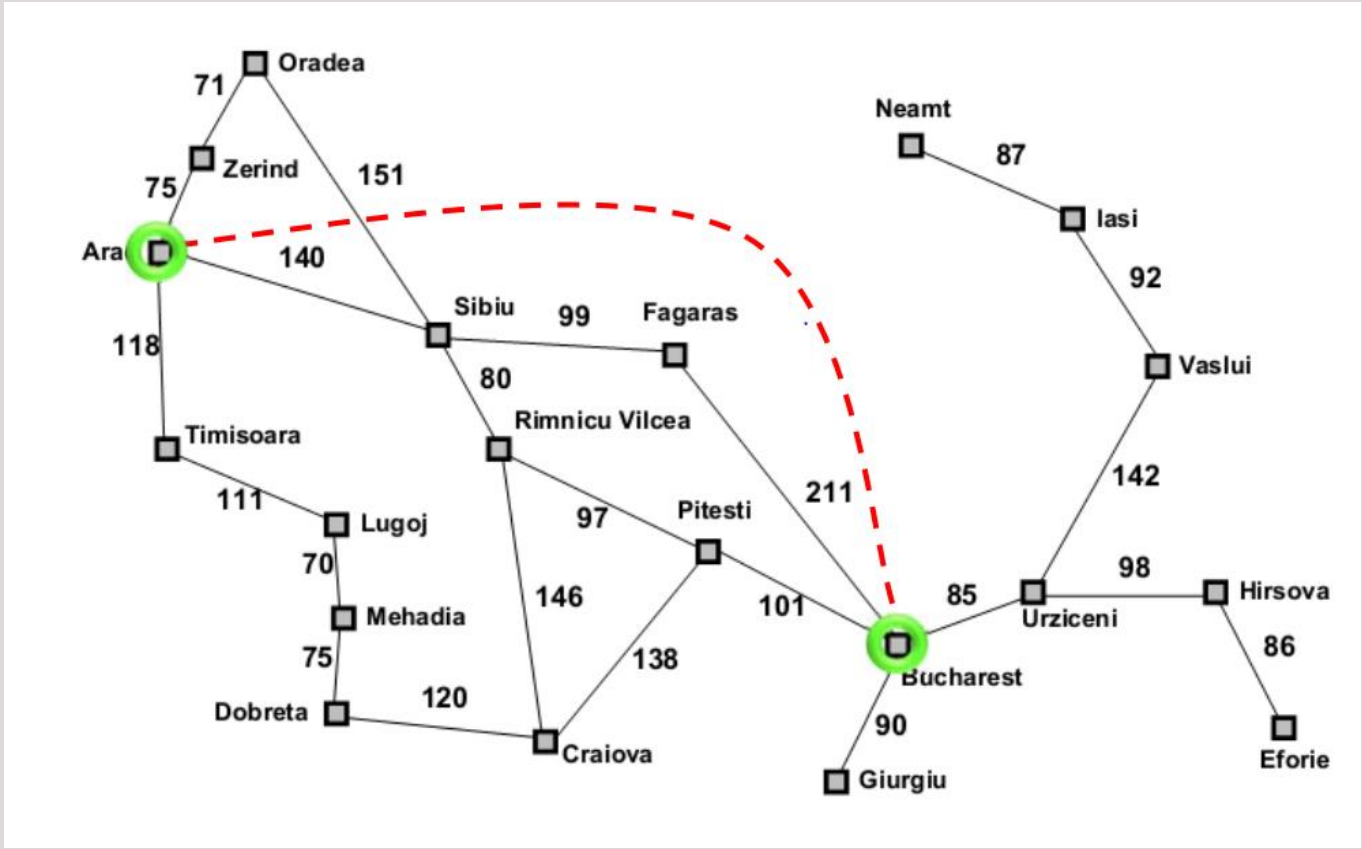
❖**تابع هزینه مسیر g(n) :** هزینه مسیر از گره اولیه تا گره n

جستجوی اول – بهترین حریصانه (greedy best first search)

- در این روش گره ای را که به نظر می رسد به هدف نزدیک تر است در ابتدا توسعه داده می شود.
- در این روش h(n) = f(n)
- جستوجو حریصانه هیچ توجهی به هزینه مسیر تا رسیدن به گره فعلی ندارد .

- **کامل بودن:** خیر، ممکن است در حلقه گیر کند.
- **بهینگی:** خیر، یک مسیر را به سوی هدف دنبال می کند، در واقع شبیه جستجوی اول .–عمق
- **پیچیدگی زمانی:** در بدترین حالت برابر $O(b^m)$، که m حداکثر عمق فضای جستجو است.
- **پیچیدگی فضایی:** این روش تمام گرهها را در حافظه نگه میدارد، بنابراین پیچیدگی مکانی آن برابر $O(b^m)$ است .

جستوجو حریصانه برای حل مسئله نقشه رومانی :

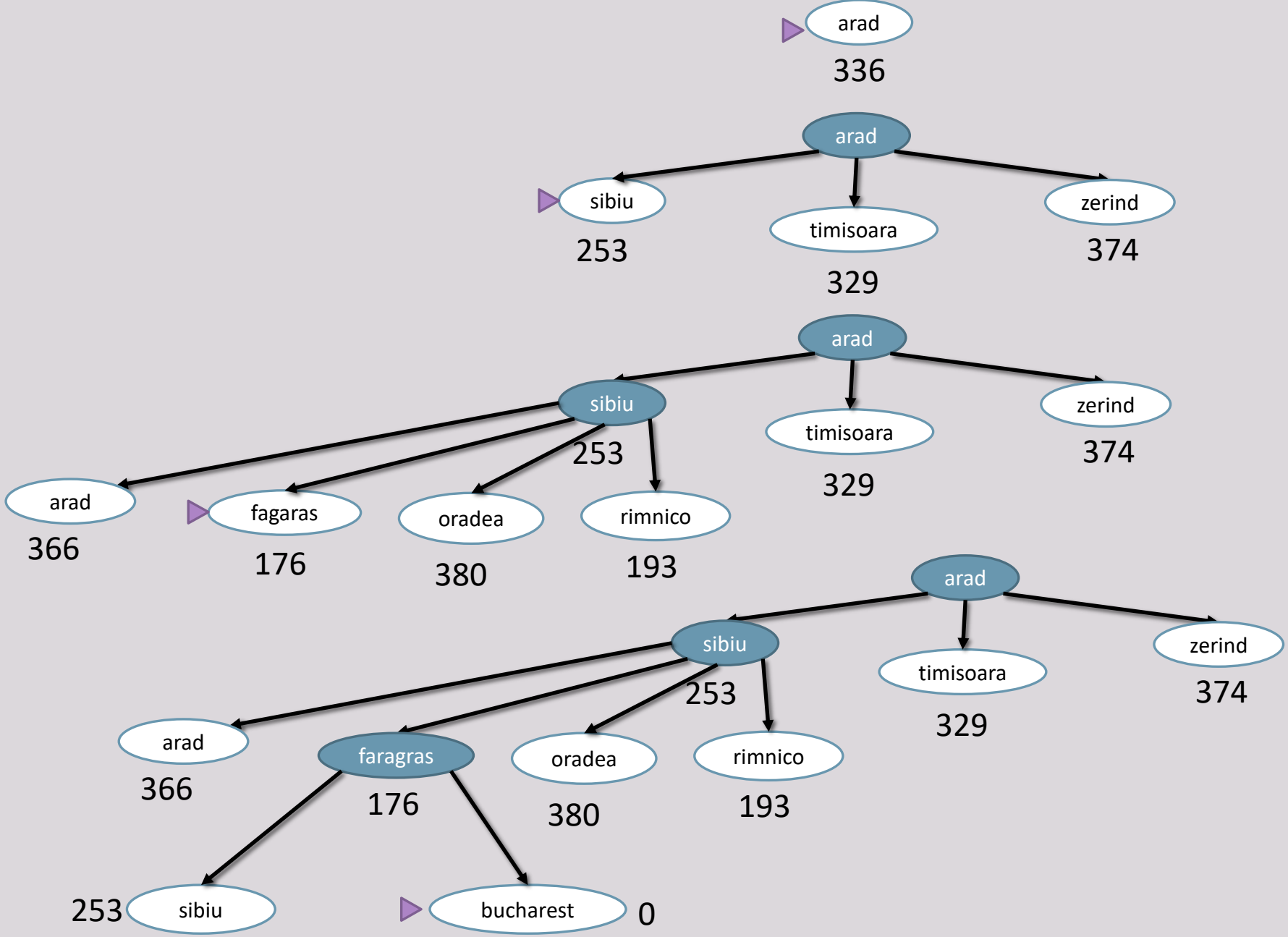


جستوجو حریصانه برای حل مسئله نقشه رومانی :

- برای این منظور از تابع فاصله مستقیم به عنوان تابع اکتشافی استفاده می کنیم.
- اعمال الگوریتم را در اسلاید بعدی مشاهده می کنیم.

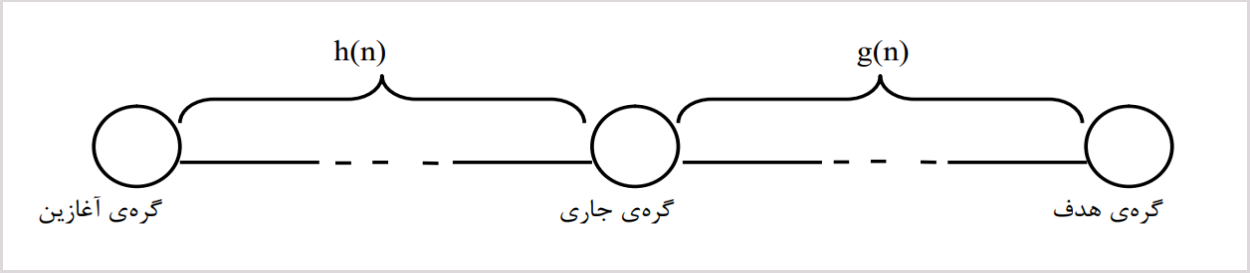
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Dobreta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vasfui	199
Lugoj	244	Zerind	374

مقادیر h مربوط به شهر های نقشه رومانی بر اساس طول خط مستقیم از هر شهر تا بخارست

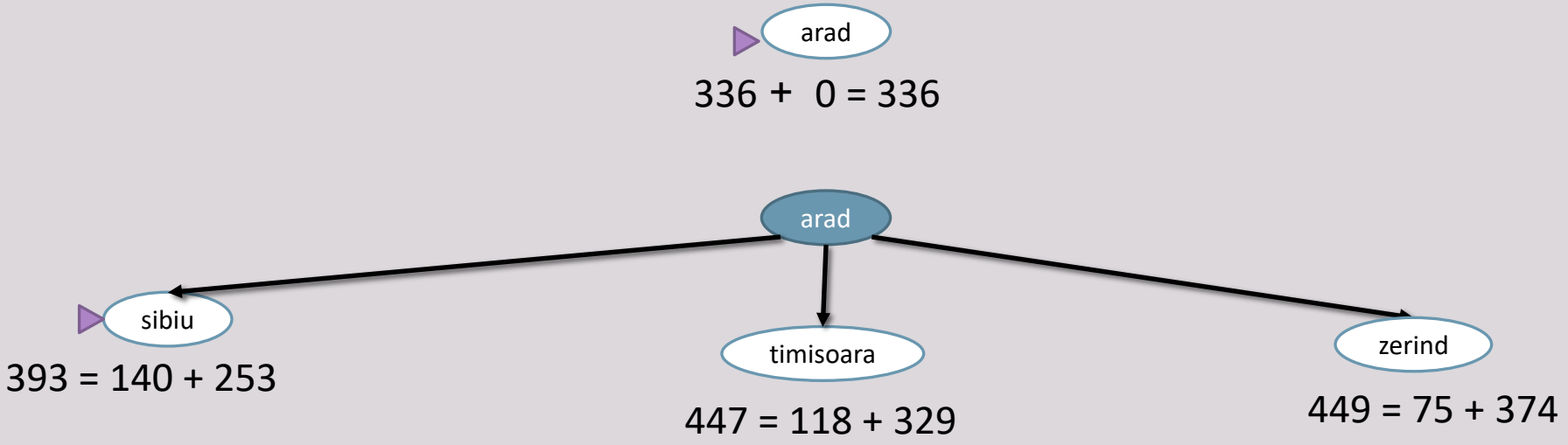

arad
336
arad
sibiu
253
timisoara
329
zerind
374
arad
sibiu
253
timisoara
329
zerind
374
arad
366
fagaras
176
oradea
380
rimnico
193
arad
sibiu
253
timisoara
329
zerind
374
arad
366
faragras
176
oradea
380
rimnico
193
253
sibiu
bucharest
0

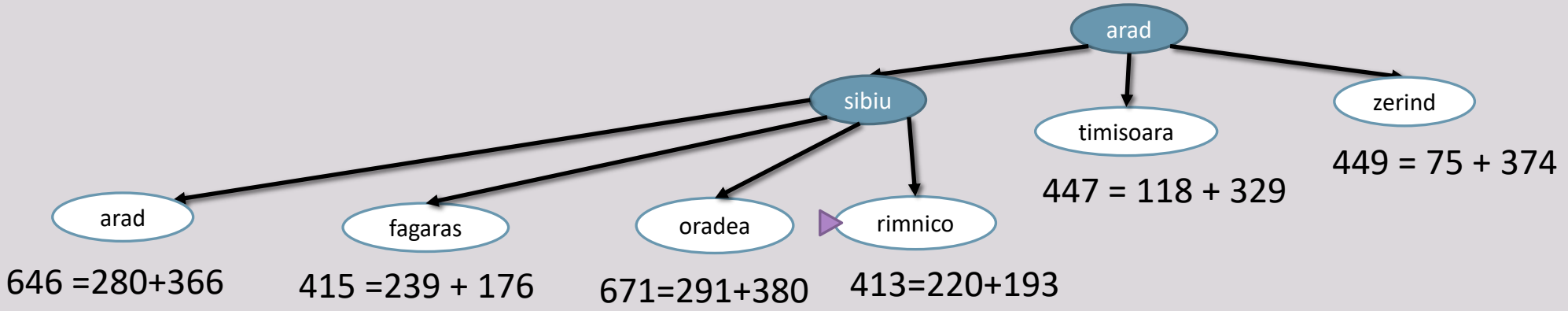
جستوجوی A*:

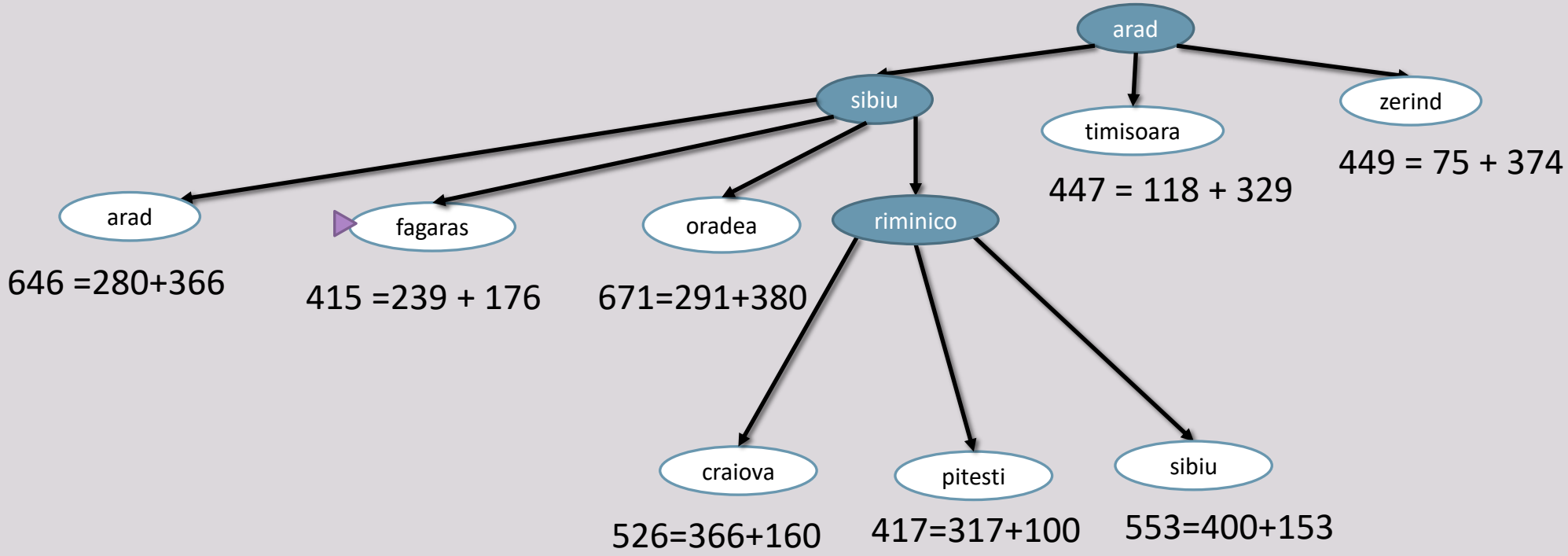
- معروف ترین شکل جستوجوی best first search ، جستوجوی A* است .
- تابع ارزیابی در این جستوجو به صورت : $f(n) = g(n) + h(n)$ تعریف میشود .
- این الگوریتم مشکل بهینه نبودن الگوریتم حریصانه را برطرف میکند : در روش حریصانه، هزینه ی رسیدن به گره ی جاری را مورد توجه قرار نمی گیرد. ولی روش جستجوی A* از توسعه دادن مسیرهایی که پر هزینه هستند، اجتناب می کند.
- در تابع ارزیابی علاوه بر تابع اکتشافی به هزینه ی رسیدن به هر گره هم توجه می شود.

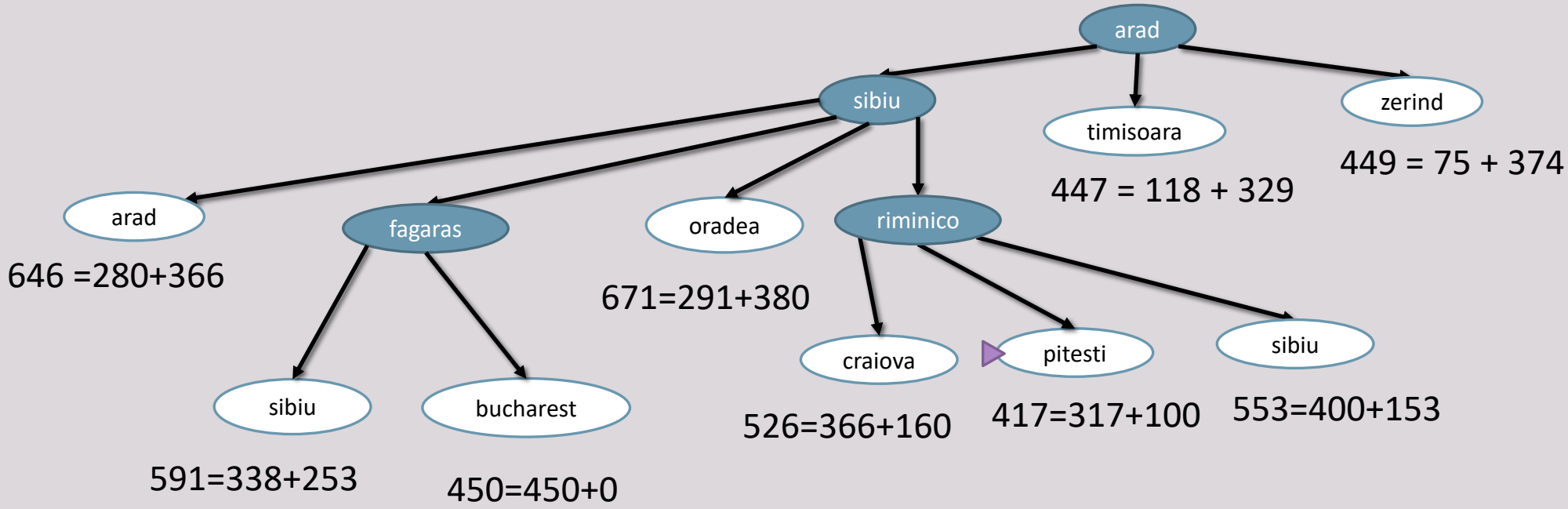


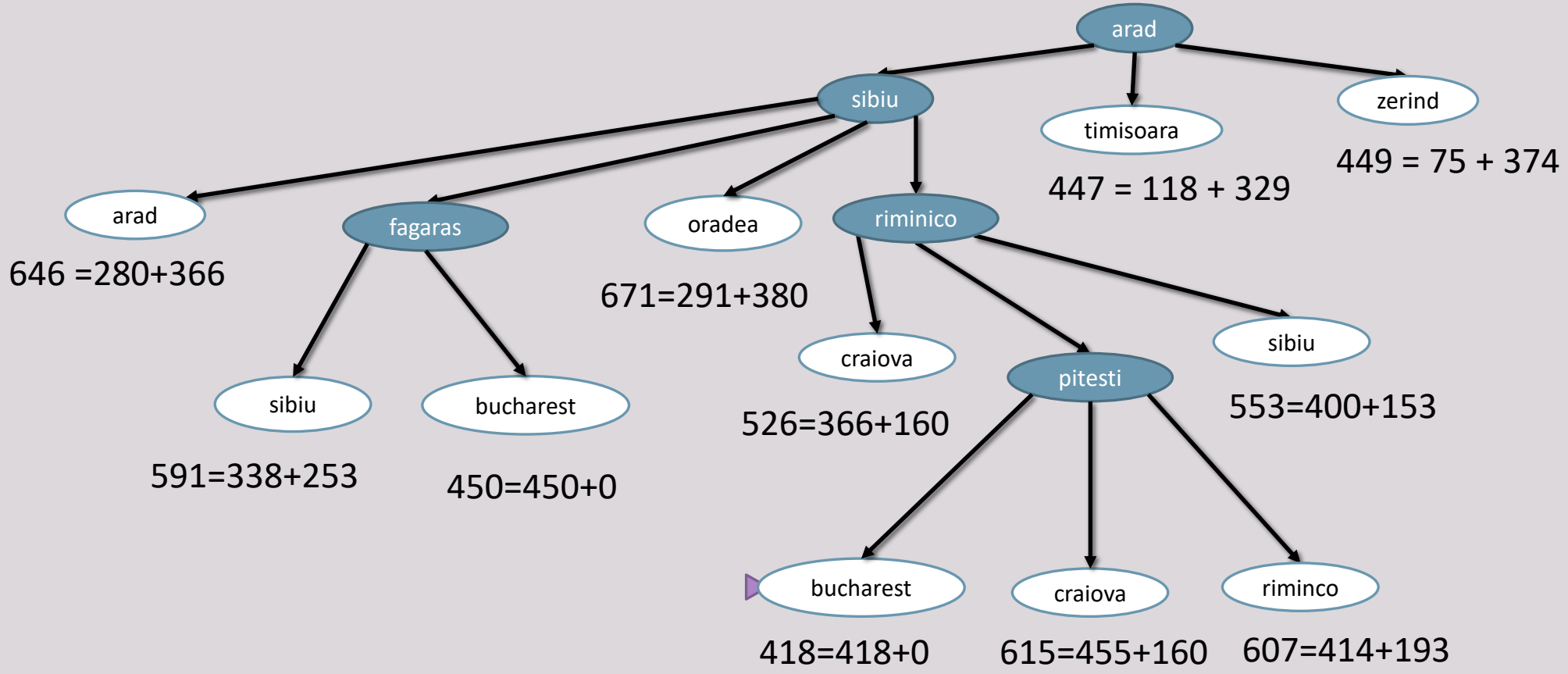
مراحل اجرای جستجوی ی A* بر نقشه رومانی :




arad
sibiu
timisoara
zerind
arad
fagaras
oradea
rimnico
447 = 118 + 329
449 = 75 + 374
646 =280+366
415 =239 + 176
671=291+380
413=220+193


arad
sibiu
timisoara
zerind
arad
fagaras
oradea
riminico
447 = 118 + 329
449 = 75 + 374
646 =280+366
415 =239 + 176
671=291+380
craiova
pitesti
sibiu
526=366+160
417=317+100
553=400+153


arad
sibiu
timisoara
zerind
449 = 75 + 374
447 = 118 + 329
arad
fagaras
oradea
riminico
646 =280+366
671=291+380
sibiu
bucharest
craiova
pitesti
sibiu
591=338+253
450=450+0
526=366+160
417=317+100
553=400+153


arad
sibiu
timisoara
zerind
449 = 75 + 374
447 = 118 + 329
arad
fagaras
oradea
riminico
646 =280+366
671=291+380
sibiu
bucharest
craiova
pitesti
sibiu
591=338+253
450=450+0
526=366+160
553=400+153
bucharest
craiova
riminco
418=418+0
615=455+160
607=414+193

ارزیابی جستوجو A* :

- **کامل بودن :** کامل است به شرطی که هزینه هرعمل بیشتر از ۱ باشد و فاکتور انشعاب متناهی باشد .
- **بهینه بودن :** اگر تابع هیوریستیک "قابل قبول" باشد جستوجوی درختی A* بهینه است . اگر تابع هیوریستیک "یکنواخت" باشد جستوجوی گراف A* بهینه است.
- **پیچیدگی زمانی :** $O(b^m)$ اما اگر h=h* آنگاه O(bd)
- **پیچیدگی فضایی :** $O(b^m)$ اما اگر h=h* آنگاه O(bd)

هیوریستیک قابل قبول و هیوریستیک یکنواخت:

❖گوییم تابع اکتشافی h(n) **قابل قبول** است اگر هزینه مسیر هر گره تا گره هدف را بیشتر از واقعیت تخمین نزند.

❖تابع h(n) **سازگار(یکنواخت–یکنوا)** است اگر برای هر گره n و هر جانشین آن مثل n’ که با انجام عمل a به آن برسیم داشته باشیم :

$$h(n) =< c(n , a , n') + h(n')$$

که در آن c(n,a,n’) = هزینه مرحله ای رفتن از n به n’ با انجام عمل a .

چند نکته درمورد الگوریتم A*:

- اگر C*، هزینه مسیر راه حل بهینه باشد:

1. جستجوی A* تمام گره ها با f(n) < C* را توسعه می دهد.
2. جستوی A* بعضی از گره هایی که f(n)=C* است را توسعه می دهد .
3. جستوجوی A* گره هایی که f(n) > C* است را توسعه نمی دهد.

جستجوی هیوریستیک با حافظه محدود IDA*:

- ساده ترین کار برای کاهش نیاز به حافظه جستجوی A*، به کار گیری ایده الگوریتم عمیق شونده تکراری در جستجوی هیوریستیک می باشد که به آن الگوریتم A*عمیق شونده تکراری (IDA*) گویند
- IDA* به جای برش براساس عمق از $f(n) = g(n) + h(n)$ استفاده می کند .
- ایرادی که این روش دارد برخورد با هزینه های واقعی است. که همانند الگوریتم هزینه یکنواخت می باشد.
- IDA* برای اغلب مسئله های با هزینه های مرحله ای، مناسب است و از سربار ناشی از نگهداری صف مرتبی از گره ها اجتناب میکند.

جستجوی هیوریستیک با حافظه محدود IDA*:

```
node                current node
g                   the cost to reach current node
f                   estimated cost of the cheapest path (root..node..goal)
h(node)             estimated cost of the cheapest path (node..goal)
cost(node, succ)    path cost function
is_goal(node)       goal test
successors(node)    node expanding function

procedure ida_star(root)
  bound := h(root)
  loop
    t := search(root, 0, bound)
    if t = FOUND then return FOUND
    if t = ∞ then return NOT_FOUND
    bound := t
  end loop
end procedure

function search(node, g, bound)
  f := g + h(node)
  if f > bound then return f
  if is_goal(node) then return FOUND
  min := ∞
  for succ in successors(node) do
    t := search(succ, g + cost(node, succ), bound)
    if t = FOUND then return FOUND
    if t < min then min := t
  end for
  return min
end function
```

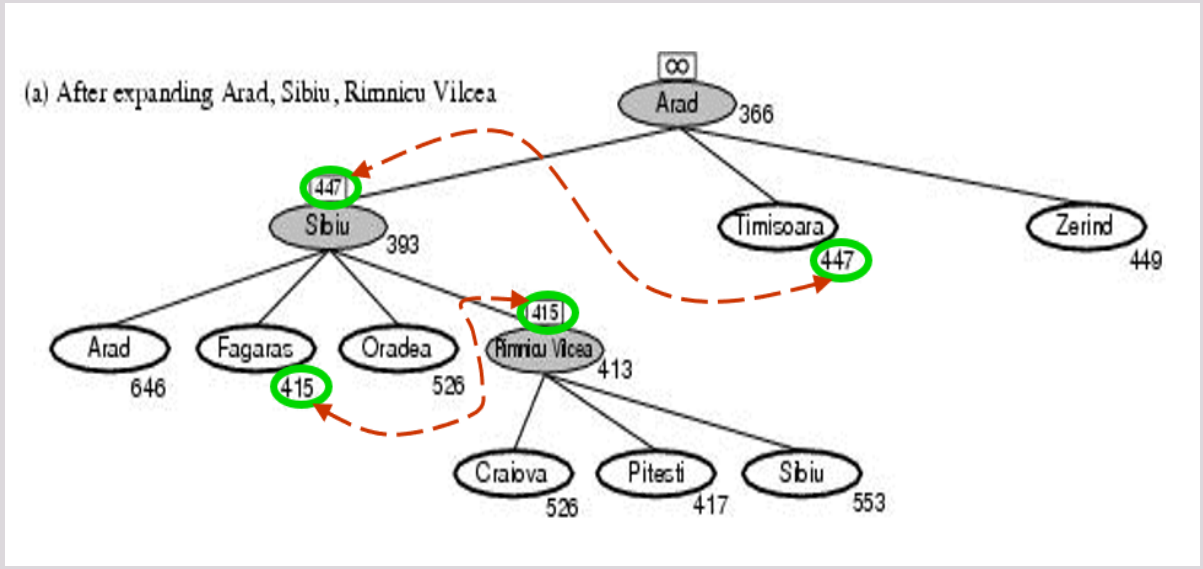
بهترین جستوجوی بازگشتی RBFS:

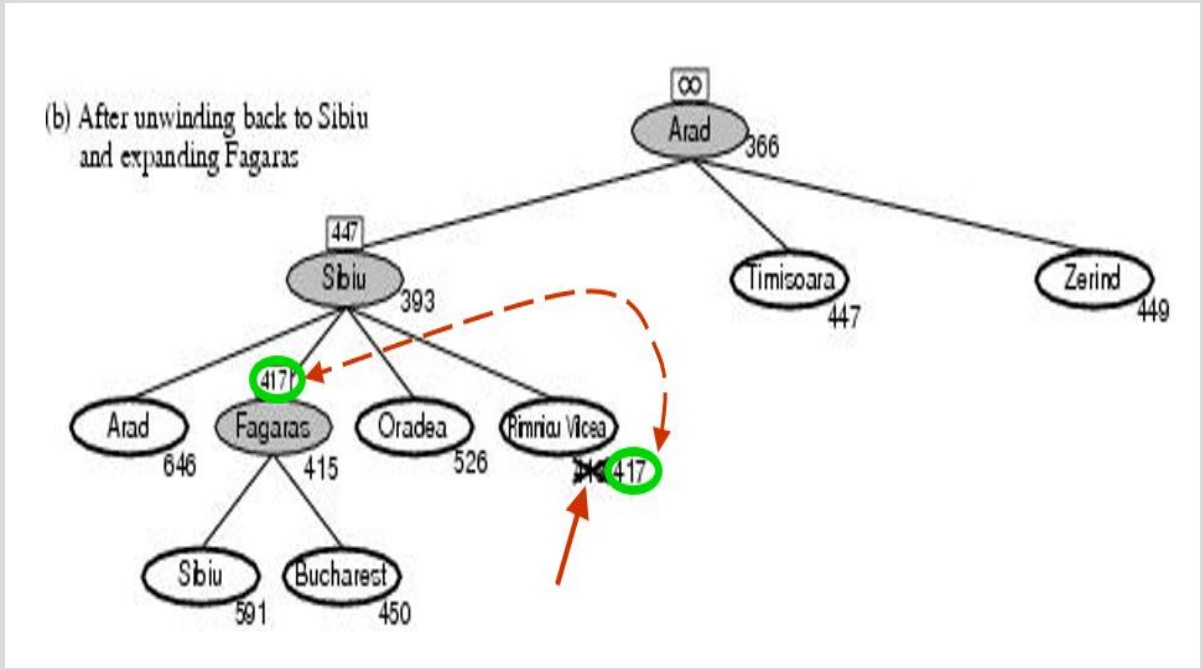
- ساختار آن شبیه جست و جوی عمقی بازگشتی است، اما به جای اینکه دائما به طرف پایین مسیر حرکت کند، مقدار f مربوط به بهترین مسیر از هر جد گره فعلی را نگهداری میکند، اگر گره فعلی از این حد تجاوز کند، بازگشتی به عقب برمیگردد تا مسیر دیگری را انتخاب کند .
- این جستجو اگر تابع اکتشافی قابل قبولی داشته باشد، بهینه است.
- پیچیدگی فضایی آن O(bd)است.
- تعیین پیچیدگی زمانی آن به دقت تابع اکتشافی و میزان تغییر بهترین مسیر در اثر بسط گره ها بستگی دار

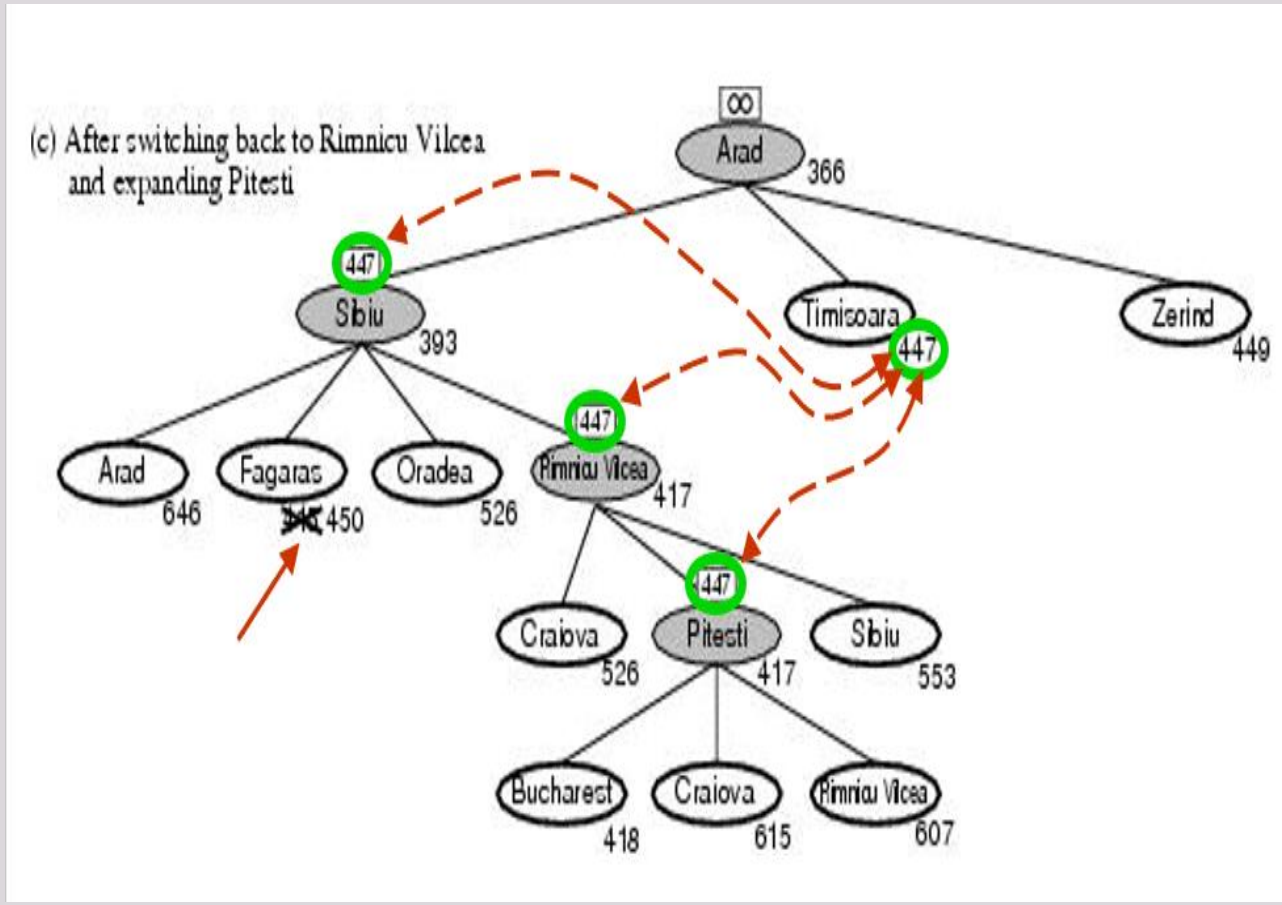
بهترین جستوجوی بازگشتی RBFS:

- IDA* و RBFS از فضای اندکی استفاده میکنند که به آنها آسیب میرساند.
- IDA* بین هر تکرار فقط یک عدد را نگهداری می کند که هزینه فعلی f است . RBFS اطلاعات بیشتری در حافظه نگهداری می کند .
- RBFSتا حدی از IDA* کارآمدتر است، اما گره های زیادی تولید میکند.

مسیریابی در نقشه رومانی RBFS:







جستوجوی حافظه محدود ساده SMA*:

SIMPLIFIED MEMORY BOUNDED A*

- SMA* قادر است تا از تمام حافظه موجود برای اجرای جستجو استفاده کند. به کارگیری حافظه بیشتر کارایی جستجو را بیشتر میکند. این الگوریتم از تمام حافظه قابل دسترس استفاده کرده و از حالات تکراری تا جایی که حافظه اجازه میدهد، جلوگیری . میکند
- SMA* بهترین برگ را بسط می دهد تا حافظه پر شود .
- این الگوریتم کامل است به شرط آنکه حافظه برای ذخیره کم عمقترین مسیر راه حل کافی باشد.
- همچنین بهینه است، به شرطی ک حافظه کافی برای ذخیره کم عمقترین گره هدف کافی باشد.
- بعلاوه بهترین راه حلی را برمیگرداند که بتواند با حافظه موجود مطابقت داشته باشد.
- زمانی که حافظه موجود برای درخت جستجوی کامل کافی باشد، جستجوی بسیار مناسبی است.

جستوجوی حافظه محدود ساده SMA*:

SIMPLIFIED MEMORY BOUNDED A*

- زمانی که این الگوریتم نیاز به تولید فرزند داشته ولی حافظه ای در دسترس نباشد، نیاز به ساختن فضا بر روی صف دارد. برای انجام این امر، یک گره را حذف میکند
- گره هایی که به این طریق از صف حذف میشوند، گره های فراموش شده نامیده می شوند .
- به منظور اجتناب از جستجوی مجدد، زیردرخت هایی که از حافظه حذف شده اند، در گره های اجدادی، اطلاعاتی را در مورد کیفیت بهترین مسیر در زیر درخت فراموش شده، نگهداری می کنند.

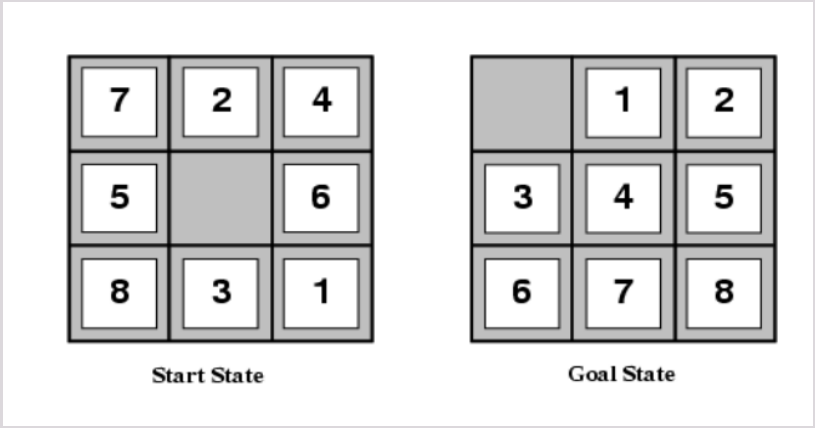
توابع هیوریستیک :

❖پیدا کردن یک تابع هیوریستیک خوب، برای جستجو مهم است.

به عنوان مثال، دو روش هیوریستیک متداول برای پازل هشت تایی به صورت زیر است

H1(n) = تعداد کاشی ها در مکان های نادرست

H2(n) = مجموعه فواصل کاشی ها از موقعیت های هدف آنها



H1 = 8
H2 = 18

فاکتور انشعاب موثر b*:

- اگر تعداد گره هایی که برای یک مسئله خاص توسط A* تولید می شود برابر با N و عمق راه حل برابر با d باشد آنگاه b* فاکتور انشعابی است که درخت یکنواختی به عمق d باید داشته باشد تا حاوی N+1 گره باشد.
- فاکتور انشعاب مؤثر معمولاً برای مسئله های سخت ثابت است
- اندازه گیری تجربی b* بر روی مجموعه کوچکی از مسئله ها می تواند راهنمای خوبی برای مفید بودن اکتشاف باشد .
- مقدار b* در اکتشافی با طراحی خوب برابر ۱ است .
- اگر برای هر گره n داشته باشیم h2(n) >= h1(n) آنگاه h2 بر h1 غالب است.

الگوریتم های جستجوی محلی و بهینه سازی:

- الگوریتم های قبلی، فضای جست و جو را به طور سیستماتیک بررسی میکنند.
  - ✓ تا رسیدن به هدف یک یا چند مسیر نگهداری میشوند
  - ✓ مسیر رسیدن به هدف، راه حل مسئله را تشکیل میدهد

- در الگوریتم های محلی مسیر رسیدن به هدف مهم نیست .

- الگوریتم های قبلی تمام مسیر از وضعیت اولیه تا نهایی را نگهداری می کردند که این کار باعث مصرف حافظه زیاد در مورد مساله های بزرگ می شود. برای برخی از مسائل اطلاعات مسیر، ضروری می باشد، مثل مسیریابی و پازل ۸ تایی. در برخی مسائل نیز مسیر به هدف ارتباطی ندارد، به عنوان مثال در مساله ۸ وزیر مهم حالت نهایی است که ۸ وزیر یکدیگر را تهدید نکنند.

- الگوریتم جستجوئی که فقط از اطلاعات وضعیت جاری استفاده میکند و مسیر هدف برای آن اهمیت نداشته باشد، الگوریتم جستجوی محلی نامیده میشود.

مزایای جستجوی محلی:

- نیاز به حافظه ثابت
- ارائه راه حلهای منطقی در فضاهای بزرگ و نامتناهی
- علاوه بر یافتن اهداف، الگوریتم های جستجوی محلی در حل مسائل خالص بهینه سازی که یافتن بهترین حالت بر اساس تابع هدف است، سودمند می باشند.

- در روی شکل حالت فعلی فرد نشان داده شده استو هدف یافتن بیشینه عمومی است که در آن تابع هدف بیشترین مقدار خود را دارد.

جستوجوی تپه نوردی: hill climbing

- ساده ترین شکل جستجوی محلی، جستجوی تپه نوردی است .
- در این جستجو، در هر نقطه بـه همسـایه ها نگاه می شود و بهترین جهت برای حرکت انتخاب می شود.
- این روش به حافظه کمی نیاز دارد. ساختمان داده گره فعلـی، فقـط حالت فعلی و مقدار تابع هدف را نگه میدارد جست و جوی محلی حریصانه نیـز نـام دارد.

```
Function HILL-CLIMBING(problem)returns a state that is a local maximum
    Input:problem,a problem
    Local variables: current, a node
                     neighbor,a node
    Current← Make-Node(INITIAL-STATE[problem])
    Loop DO
       neighbor ← a highest- value successor of current
       if value [neighbor] ≤ VALUE[current] then return State [currnt]
       Current ← neighbor
```

جستوجوی تپه نوردی: hill climbing

❖تپه نوردی به دلایل زیر میتواند متوقف شود :

1. **بیشینه محلی :** یک بیشینه محلی، برخلاف بیشینه عمومی، قله ای است که پائین تر از بلندترین قله درفضای حالت است. زمانی که روی بیشینه محلی هستیم، الگوریتم متوقف می شود.
2. **برآمدگیها:** نوک کوه، دارای لبه های سراشیب است. بنابراین جستجو به بالای نوک کوه به آسانی میرسد، اما بعد با ملایمت به سمت قله میرود. مگر اینکه عملگرهایی موجود باشند که مستقیماً به سمت بالای نوک کوه حرکت کنند. جستجو ممکن است از لبه ای به لبه دیگر نوسان داشته باشد و پیشرفت کمی را حاصل شود
3. **فلات :** یک فلات محوطه ای از فضای حالت است که تابع ارزیابی یکنواخت باشد. جستجو یک قدم تصادفی را برخواهد داشت.

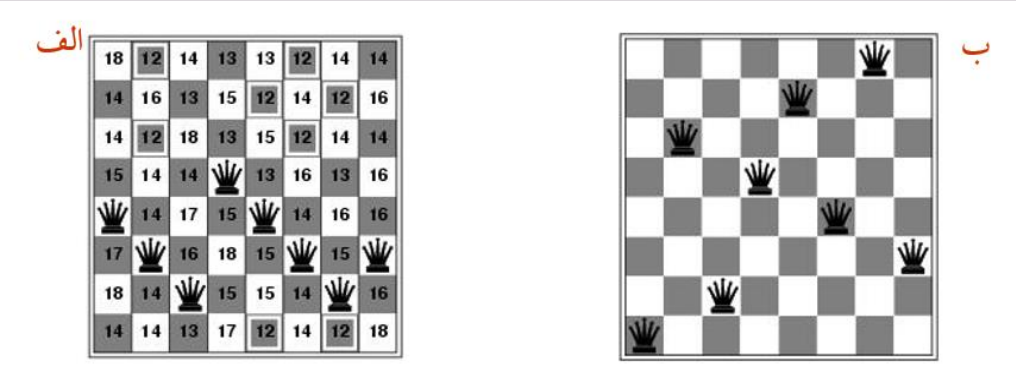
جستوجوی تپه نوردی: hill climbing

❖مثال ۸ وزیر :

در هر حالت ۸ وزیر در صفحه قرار دارند .

✓تابع جانشین : انتقال یک وزیر به مربع دیگر در همان ستون

✓تابع اکتشاف: جفت وزیرهایی که مستقیم یا غیر مستقیم به یکدیگر گارد می دهند .



الف) حالت با هزینه h=17 که مقدارh را برای هر جانشین نشان می دهد .

ب) کمینه محلی در فضای حالت ۸ وزیر ؛ h=1

جستوجو پرتو محلی (local beam search)

- در این روش به جای یک حالت ، kحالت را نگه می دارد و بالاترین مقدار را در بین همه پسین ها انتخاب می کند.
- این روش اغلب کامل می باشد ولی تمام حالت های kدر نهایت، در بالای تپه محلی قرار می گیرند. در این حالت kپسین را به طور تصادفی انتخاب می کنیم.
- یک روش بر مبنای مسیر است که به چند مسیر اطراف مسیر جاری نگاه می کند و kوضعیت را به جای یک وضعیت در حافظه نگاه می دارد.

- ✓ حالت اولیه : k حالت تصادفی
- ✓ گام بعد : جانشین همه k حالت تولید می شود
- ✓ اگر یکی از جانشین ها هدف بود تمام می شود .
- ✓ وگرنه بهترین جانشین را انتخاب کرده ، تکرار می کند .

الگوریتم های ژنتیک (GA) :

- این الگوریتم ها مبتنی بر روند تکاملی می باشند که برگرفته از نظام طبیعت است.
- مزیت الگـوریتم های ژنتیکی در این است که برای مسائلی که جواب قطعی وجود ندارند و یا نمی توان از راه حل های معمول به جواب رسـید طـی یک روند تکاملی مجموعه ای از نزدیک ترین جواب ها به بهترین جواب را نشان خواهد داد.
- یک روش عددی و تصادفی و مبتنی بر تکرار است.
- در هر تکرار همزمان چند نقطه از فضای حالت بررسی می شود پس احتمال اینکه در ماکزیمم محلی گیر کند کم است.
- الگوریتم ژنتیک یک الگوریتم برای مسائل بهینه سازی است. در الگوریتم های ژنتیک به هر جواب مسئله یک کرومـوزوم و بـه هـر متغیر از آن یک ژن می گوئیم.

الگوریتم های ژنتیک (GA) :

❖**کدینگ مسأله** : نگاشتی از تعریف مساله به فضای ژنتیک می باشد. الگوریتم ژنتیک به جای اینکه بر روی پارامترها یا متغیرهای مساله کار کند، با شکل کد شده آنها به طور مناسب سروکار دارد. کدینگ استاندارد در ژنتیک کدینگ باینری است. کدینگ درست مسئله در الگوریتم ژنتیک سخت ترین قسمت کار است.

❖**تولید جمعیت اولیه :**

✓الگوریتم ژنتیک کار خود را با تولید جمعیت اولیه ای از کروموزوم ها آغاز می کند

✓ سپس در یک حلقه به طور مکرر تعدادی از کروموزوم های برتر نسل فعلی را انتخاب کرده و سپس نسل جدیدی را از این کروموزومها تولید می کند.

✓هر کروموزوم نشان دهنده یک حالت از فضای حالت مسئله می باشد و بنابراین منظور از تولید جمعیت اولیه، تولید تعدادی جواب برای مسئله خواهد بود.

الگوریتم های ژنتیک (GA) :

❖ ارزش گذاری : در واقع میزان شایستگی افراد برای ادامه نسل است. آن را با Fitnessنشان می دهند. هر چه کیفیت و شایستگی یک فرد بیشتر باشد احتمال اینکه در تولید نسل بعد مشارکت کند بیشتر است.

❖ شکلی از جستجوی پرتو غیر قطعی است که در آن حالت های جانشین از طریق ترکیب دو حالتِ والد تولید می شود.

❖ GA همانند جستجوی پرتو با مجموعه ای از K حالت تصادفی شروع می کند که جمعیت نام دارد.

❖ حالت یا فرد بصورت رشته ای بر روی الفبای متناهی نشان داده می شود(معمولا مجموعه ای از صفرها و یک ها)

هر حالت توسط تابع ارزیابی یا تابع برازش (fitness funtion) نرخ بندی شده است.

این تابع باید مقادیربزرگتر را برای حالت های بهتر برگرداند.(برای مسئله ی ۸ وزیر ، تعداد جفت های بدون گارد)

```
Function GENETIC-ALGORITHM(population,FITNESS-FN)returns an  in  devidual
   inputs: population,a sef of individiduals
           FITNESS-FN,a function that measures the fitness of an
Repeat
    new-population ← empty  set
   Loop  for  I  from  1  to  SIZE(population) do
        x ← RANDOM-SELECTION(population,FITNESS-FN)
        y ← RANDOM-SELECTION(population,FITNESS-FN)
        child ← REPERODUCE(x,y)
        if (small  random  probability) then  child ←MUTATE(child)
        add  child  to  new- population
       population ← new-population
  Until sum individual is fit enough or enough time has elapsed
  Return  the  best  individual  in  population,according  to FITNESS-FN
---------------------------------------------------------------------
Function  REPERODUCE(x,y)returns  an  indidvidual
  input: x,y,parrent-individuals
  n ← LENGHT(x)
  c ← random  number  from  1 to  n
  Return  APPEND (SUBSTRING(x,1,c),SUBSTRING(y,c+1,n))
```

پایان فصل چهارم