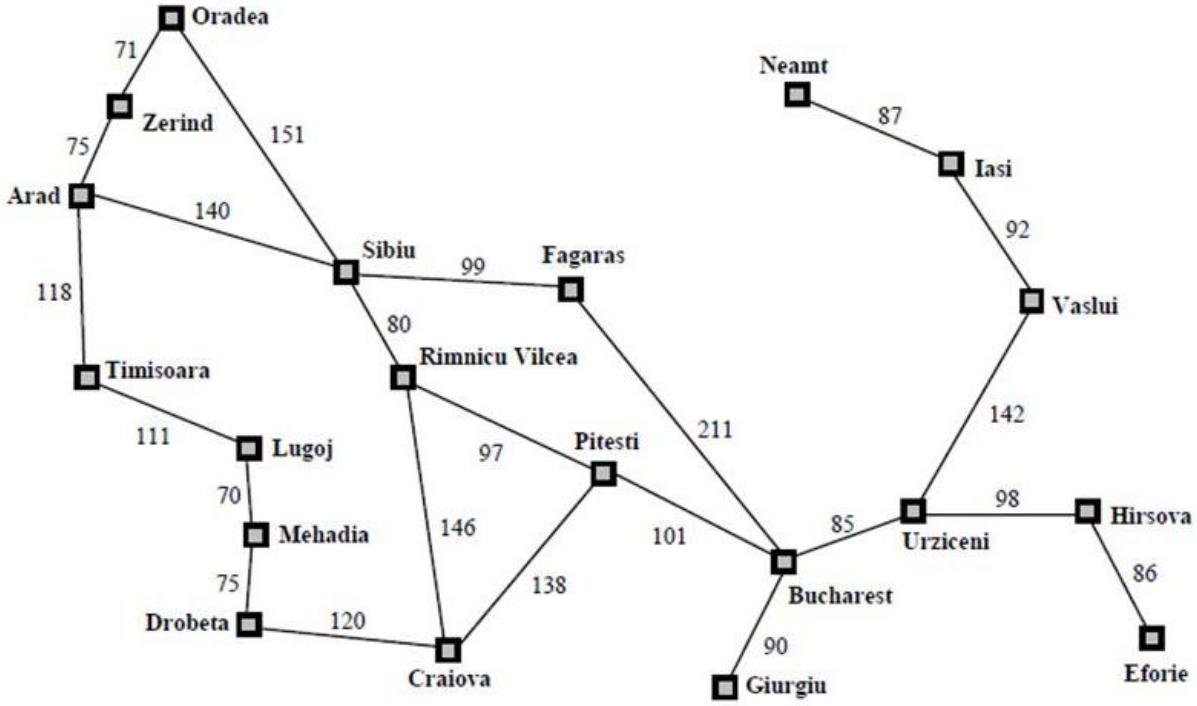
Thực hành môn Trí tuệ nhân tạo_Tuần 3

1. Cài đặt thuật toán Greedy – Best – First search để tìm đường đi từ Arad tới

h(Arad) = 366	h(Hirsova) = 0	h(Rimnicu Vilcea) = 193
h(Bucharest) = 20	h(Iasi) = 226	h(Sibiu) = 253
h(Craiova)= 160	h(Lugoj) = 244	h(Timisoara)=329
h(Drobeta) = 242	h(Mehadia)=241	h(Urziceni) = 10
h(Eforie) = 161	h(Neamt) = 234	h(Vaslui) = 199
h(Fagaras) = 176	h(Oradea) = 380	h(Zerind) = 374
h(Giurgiu) = 77	h(Pitesti) = 100	

   Hirsova như hình với h(n) được xác định như sau:

2. Cài đặt thuật toán A* để tìm đường đi ngắn nhất từ Arad tới Hirsova như hình với hàm h(n) được xác định như trong bài tập 1 và g(n) là khoảng cách giữa 2 thành phố.


Oradea
71
Neamt
87
Zerind
151
Iasi
75
Arad
140
92
Sibiu
99
Fagaras
118
Vaslui
80
Rimnicu Vilcea
Timisoara
142
111
Lugoj
97
Pitesti
211
70
98
Hirsova
Mehadia
146
101
85
Urziceni
75
86
Bucharest
138
Drobeta
120
90
Craiova
Giurgiu
Eforie

HƯỚNG DẪN

1. Greedy – Best – First Search

Ý tưởng: GBFS mở rộng nút gần đích nhất với hi vọng cách làm này sẽ dẫn đến lời giải một cách nhanh nhất. Đánh giá chi phí của các nút chỉ dựa trên hàm heuristic:

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

2. Thuật toán A^*

Thuật toán đánh giá 1 nút dựa trên chi phí đi từ nút gốc đến nút đó (g(n)) cộng với chi phí từ nút đó đến nút đích (h(n)).

$$f(n) = g(n) + h(n)$$

Hàm h(n) được gọi là chấp nhận được nếu với mọi trạng thái n, $h(n) \leq$ độ dài đường đi ngắn nhất thực tế từ u tới trạng thái đích.

Thuật giải A^* sử dụng 2 tập hợp sau đây:

- OPEN: tập chứa các trạng thái đã được sinh ra nhưng chưa được xét đến $\Rightarrow$ OPEN là 1 hàng đợi ưu tiên (priority queue) mà trong đó, phần tử có độ ưu tiên cao nhất là phần tử tốt nhất.
- CLOSE: tập chứa các trạng thái đã được xét đến. Ta cần lưu trữ những trạng thái này trong bộ nhớ để đề phòng khi một trạng thái mới được tạo ra lại trùng với 1 trạng thái mà ta đã xét đến trước đó. Trong trường hợp không gian tìm kiếm có dạng cây thì không cần dùng tập này.

Khi xét đến một trạng thái T_i bên cạnh việc lưu trữ 3 giá trị cơ bản $g(T_i)$, $h(T_i)$, $f(T_i)$ để phản ánh độ tốt của trạng thái đó, A^* còn lưu trữ thêm 2 thông số sau:

- ✓ *Trạng thái cha của trạng thái T_i (Father(T_i)).* Trong trường hợp có nhiều trạng thái dẫn đến trạng thái T_i thì chọn Father(T_i) sao cho chi phí đi từ trạng thái khởi đầu đến T_i là thấp nhất, nghĩa là $g(T_i) = g(T_{father}) + cost(T_{father}, T_i)$ là thấp nhất
- ✓ *Danh sách các trạng thái kế tiếp của T_i*: danh sách này lưu trữ các trạng thái kế tiếp T_k của T_i sao cho chi phí đến T_k thông qua T_i từ trạng thái ban đầu là thấp nhất$\Rightarrow$ danh sách này được tính từ thuộc tính Cha của các trạng thái được lưu trữ.

Procedure Astar-Search

Begin

1. Đặt OPEN chỉ chứa T_0. Đặt $g(T_0) = 0$, $h(T_0) = 0$ và $f(T_0) = 0$. Đặt CLOSE là tập rỗng.
2. Lặp lại các bước cho đến khi gặp điều kiện dừng
   2.a. Nếu OPEN rỗng: bài toán vô nghiệm, thoát.
   2.b. Ngược lại, chọn T_{max} trong OPEN sao cho $f(T_{max})$ là nhỏ nhất
      2.b.1. Lấy T_{max} ra khỏi OPEN và đưa T_{max} vào CLOSE.
      2.b.2. Nếu T_{max} là T_G (trạng thái đích) thì thoát và thông báo lời giải là T_{max}
      2.b.3. Nếu T_{max} không phải là T_G. Tạo ra danh sách tất cả các trạng thái kế tiếp của T_{max}. Gọi một trạng thái này T_k. Với mỗi T_k, làm các bước sau:
         2.b.3.1. Tính $g(T_k) = g(T_{max}) + cost(T_{max}, T_k)$
         2.b.3.2. Nếu tồn tại $T_{k'}$ trong OPEN trùng với T_k.
            Nếu $g(T_k) < g(T_{k'})$ thì
               Đặt $g(T_{k'}) = g(T_k)$
               Tính lại $f(T_{k'})$
               Đặt $Cha(T_{k'}) = T_{max}$
         2.b.3.3. Nếu tồn tại $T_{k'}$ trong CLOSE trùng với T_k
            Nếu $g(T_k) < g(T_{k'})$ thì
               Đặt $g(T_{k'}) = g(T_k)$
               Tính lại $f(T_{k'})$
               Đặt $Cha(T_{k'}) = T_{max}$
               Lan truyền sự thay đổi giá trị g, f cho tất cả các trạng thái tiếp theo của T_i (ở tất cả các cấp) đã được lưu trữ trong CLOSE và OPEN.
         2.b.3.4. Nếu T_k chưa xuất hiện trong cả OPEN lẫn CLOSE thì
            Thêm T_k vào OPEN
            Tính: $f(T_k) = g(T_k) + h(T_k)$

Xây dựng lại đường đi từ T_0 tới T_G: ta lần ngược theo thuộc tính Cha của các trạng thái đã được lưu trữ trong CLOSE cho đến khi đạt đến T_0.

Ta có h là đường chim bay cho trong bảng và cost(T_i, T_{i+1}) là chiều dài đường đi từ T_i tới T_{i+1}. Khi đó

Ban đầu

OPEN = {(Arad, g = 0, h = 0, f = 0)}

CLOSE = {}

Do OPEN chỉ chứa có 1 thành phố nên thành phố này sẽ là thành phố tốt nhất. Nghĩa là ta chọn T_{max} = Arad. Lấy Arad ra khỏi OPEN và đưa vào CLOSE.

OPEN = {}

CLOSE = {(Arad, g = 0, h = 0, f = 0)}

Từ Arad có thể đi được đến 3 thành phố Sibiu, Timisoara và Zerind. Ta lần lượt tính f, g và h của 3 thành phố này. Do cả 3 nút mới tạo ra này chưa có nút cha nên ban đầu nút cha của chúng đều là Arad.

- ✓ h(Sibiu) = 253

  g(Sibiu) = g(Arad) + cost(Arad, Sibiu) = 0 + 140 = 140

  f(Sibiu) = g(Sibiu) + h(Sibiu) = 140 + 253 = 393

  Cha(Sibiu) = Arad
- ✓ h(Timisoara) = 329

  g(Timisoara) = g(Arad) + cost(Arad, Timisoara) = 0 + 118 = 118

  f(Timisoara) = g(Timisoara) + h(Timisoara) = 118 + 329 = 447

  Cha(Timisoara) = Arad
- ✓ h(Zerind) = 374

  g(Zerind) = g(Arad) + cost(Arad, Zerind) = 0 + 75 = 75

  f(Zerind) = g(Zerind) + h(Zerind) = 75 + 374 = 449

  Cha(Zerind) = Arad

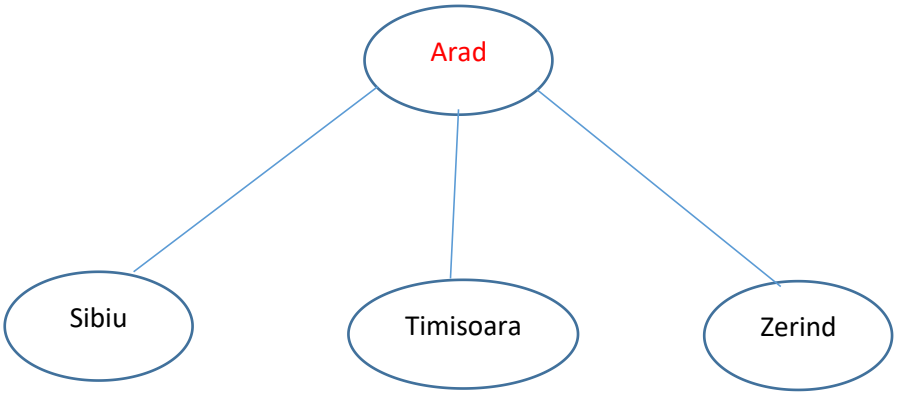
Do Sibiu, Timisoara và Zerind đều không có trong cả OPEN và CLOSE nên ta thêm 3 nút này vào OPEN.

OPEN = {(Sibiu, g = 140, h = 253, f = 393, Cha = Arad),

(Timisoara, g = 118, h = 329, f = 447, Cha = Arad),

(Zerind, g = 75, h = 374, f = 449, Cha = Arad)}

CLOSE = {(Arad, g = 0, h = 0, f = 0)}



(Lưu ý: Tên thành phố có màu đỏ là nút trong tập CLOSE, ngược lại là nút trong tập OPEN)

Trong tập OPEN, Sibiu là nút có giá trị f nhỏ nhất nên ta sẽ chọn T_{max} = Sibiu. Ta lấy Sibiu ra khỏi OPEN và đưa vào CLOSE

OPEN = {(Timisoara, g = 118, h = 329, f = 447, Cha = Arad),

(Zerind, g = 75, h = 374, f = 449, Cha = Arad)}

CLOSE = {(Arad, g = 0, h = 0, f = 0),

(Sibiu, g = 140, h = 253, f = 393, Cha = Arad)}

Từ Sibiu có thể đi được đến Arad, Fagaras, Oradea, R. Vilcea. Ta lần lượt tính h, g và f của các nút này.

- ✓ h(Arad) = 366

g(Arad) = g(Sibiu) + cost(Sibiu, Arad) = 140 + 140 = 280

f(Arad) = g(Arad) + h(Arad) = 280 + 366 = 646

✓ h(Fagaras) = 176

g(Fagaras) = g(Sibiu) + cost(Sibiu, Fagaras) = 140 + 99 = 239

f(Fagaras) = g(Fagaras) + h(Fagaras) = 239 + 176 = 415

✓ h(Oradea) = 380

g(Oradea) = g(Sibiu) + cost(Sibiu, Oradea) = 140 + 151 = 291

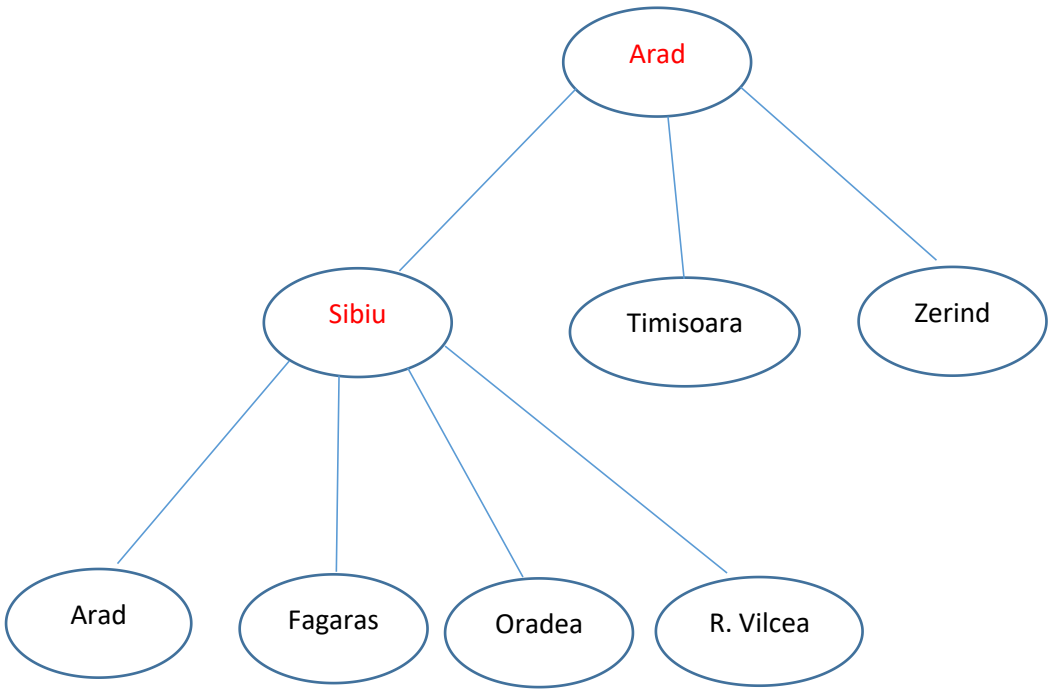
f(Oradea) = g(Oradea) + h(Oradea) = 291 + 380 = 671

✓ h(R. Vilcea) = 193

g(R. Vilcea) = g(Sibiu) + cost(Sibiu, R. Vilcea) = 140 + 80 = 220

f(R. Vilcea) = g(R. Vilcea) + h(R. Vilcea) = 220 + 193 = 413

Nút Arad đã có trong CLOSE và g(Arad) mới được tạo ra có giá trị là 280 lớn hơn g(Arad) lưu trong CLOSE có giá trị là 0 nên ta sẽ không cập nhật giá trị g và f của Arad lưu trong CLOSE. 3 nút Fagaras, Oradea và R. Vilcea đều không có trong OPEN và CLOSE nên ta sẽ thêm 3 nút này vào OPEN, đặt cha của chúng là Sibiu.



OPEN = {(Timisoara, g = 118, h = 329, f = 447, Cha = Arad),

(Zerind, g = 75, h = 374, f = 449, Cha = Arad),

(Fagaras, g = 239, h = 176, f = 415, Cha = Sibiu),

(Oradea, g = 291, h = 380, f = 617, Cha = Sibiu),

(R. Vilcea, g = 220, h = 193, f = 413, Cha = Sibiu)}

CLOSE = {(Arad, g = 0, h = 0, f = 0),

(Sibiu, g = 140, h = 253, f = 393, Cha = Arad)}

Trong tập OPEN, R. Vilcea là nút có giá trị f nhỏ nhất nên ta chọn T_{max} = R. Vilcea. Chuyển R. Vilcea từ tập OPEN sang tập CLOSE. Từ R. Vilcea có thể đi được tới 3 thành phố là Craiova, Pitesti và Sibiu. Ta lần lượt tính các giá trị h, g và f của 3 thành phố này.

- ✓ h(Sibiu) = 253

  g(Sibiu) = g(R. Vilcea) + cost(R. Vilcea, Sibiu) = 220 + 80 = 300

  f(Sibiu) = g(Sibiu) + h(Sibiu) = 300 + 253 = 553

- ✓ h(Craiova) = 160

  g(Craiova) = g(R. Vilcea) + cost(R. Vilcea, Craiova) = 220 + 146 = 366

  f(Craiova) = g(Craiova) + h(Craiova) = 366 + 160 = 526

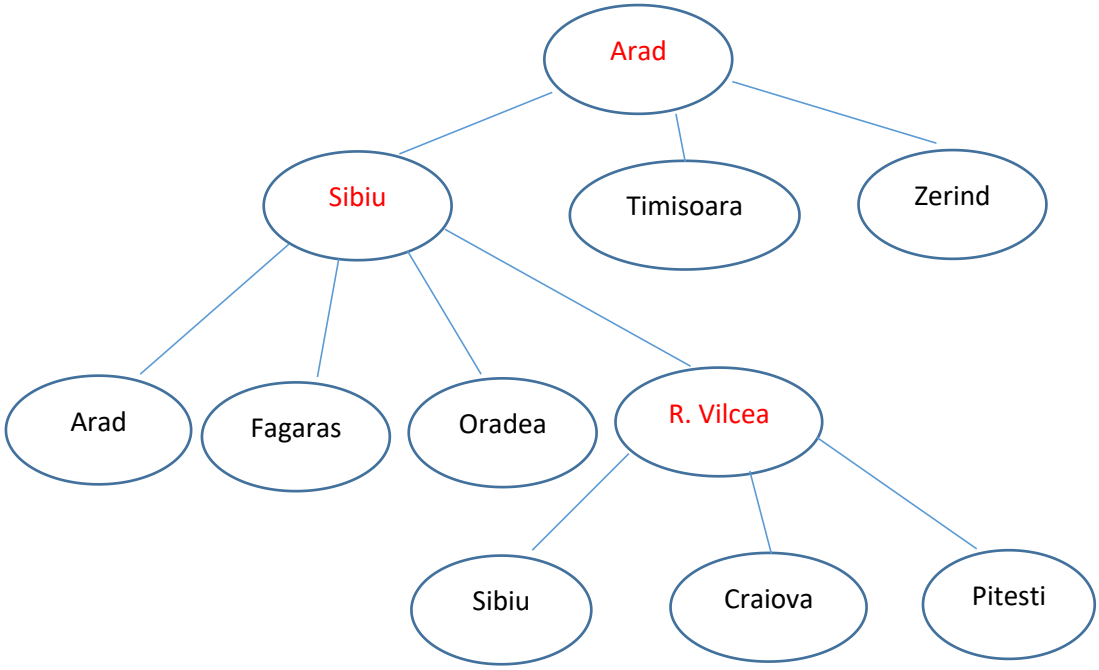
- ✓ h(Pitesti) = 100

  g(Pitesti) = g(R. Vilcea) + cost(R. Vilcea, Pitesti) = 220 + 97 = 317

  f(Pitesti) = g(Pitesti) + h(Pitesti) = 317 + 100 = 417

Do Sibiu đã có trong CLOSE và g(Sibiu) mới có giá trị là 553 lớn hơn g(Sibiu) trong CLOSE có giá trị là 393 nên ta không cập nhật lại các giá trị Sibiu được lưu

trong CLOSE. Craiova và Pitesti đều không có trong OPEN lẫn CLOSE nên ta sẽ đưa chúng vào OPEN và đặt cha của chúng là R. Vilcea.



OPEN = {(Timisoara, g = 118, h = 329, f = 447, Cha = Arad),

(Zerind, g = 75, h = 374, f = 449, Cha = Arad),

(Fagaras, g = 239, h = 176, f = 415, Cha = Sibiu),

(Oradea, g = 291, h = 380, f = 617, Cha = Sibiu),

(Craiova, g = 366, h = 160, f = 526, Cha = R. Vilcea),

(Pitesti, g = 317, h = 100, f = 417, Cha = R. Vilcea)}

CLOSE = {(Arad, g = 0, h = 0, f = 0),

(Sibiu, g = 140, h = 253, f = 393, Cha = Arad),

(R. Vilcea, g = 220, h = 193, f = 413, Cha = Sibiu)}

Từ tập OPEN, nút Fagaras có giá trị f nhỏ nhất nên T_{max} = Fagaras. Từ Fagaras ta có thể đi được tới Sibiu và Bucharest. Lấy Fagaras ra khỏi tập OPEN và đưa vào CLOSE. Ta cũng tính các giá trị h, g và f của Sibiu và Bucharest.

- ✓ h(Sibiu) = 253

  g(Sibiu) = g(Fagaras) + cost(Fagaras, Sibiu) = 239 + 99 = 338

  f(Sibiu) = g(Sibiu) + h(Sibiu) = 338 + 253 = 591

- ✓ h(Bucharest) = 20

  g(Bucharest) = g(Fagaras) + cost(Fagaras, Bucharest) = 239 + 211 = 450

  f(Bucharest) = g(Bucharest) + h(Bucharest) = 450 + 20 = 470

Sibiu đã có trong tập OPEN nhưng do g(Sibiu) mới tạo có giá trị là 338 lớn hơn g(Sibiu) trong CLOSE có giá trị là 140 nên ta sẽ không cập nhật lại giá trị g và h của Sibiu. Bucharest không có trong tập OPEN lẫn CLOSE nên ta sẽ thêm nút này vào OPEN.

OPEN = {(Timisoara, g = 118, h = 329, f = 447, Cha = Arad),

(Zerind, g = 75, h = 374, f = 449, Cha = Arad),

(Oradea, g = 291, h = 380, f = 617, Cha = Sibiu),

(Craiova, g = 366, h = 160, f = 526, Cha = R. Vilcea),

(Pitesti, g = 317, h = 100, f = 417, Cha = R. Vilcea),

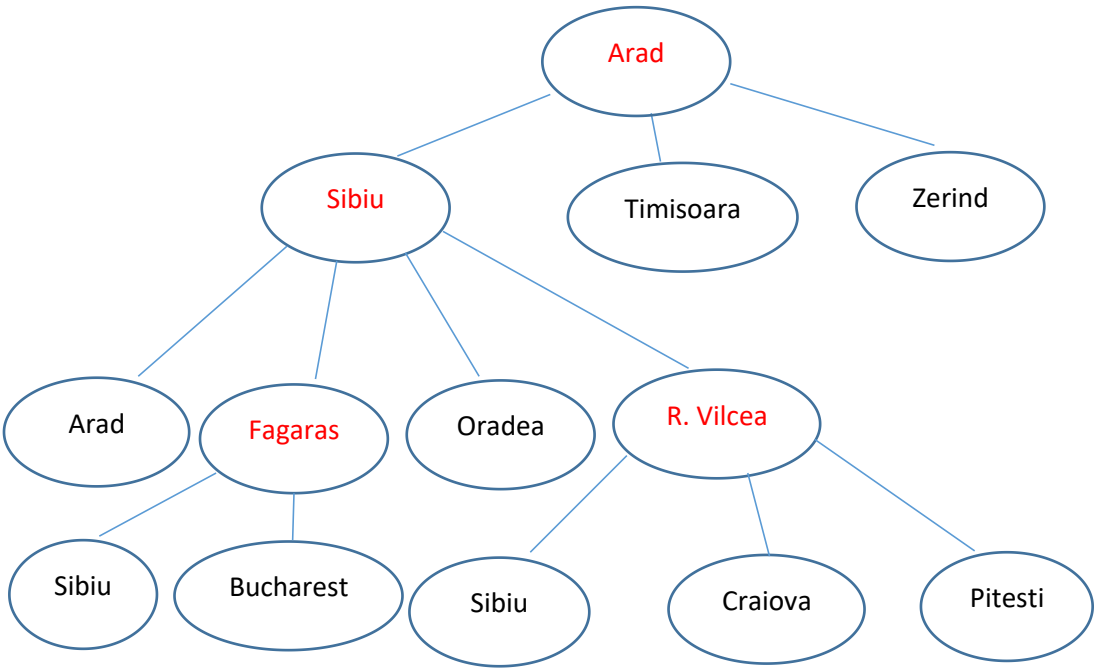
(Bucharest, g = 450, h = 20, f = 470, Cha = Fagaras)}

CLOSE = {(Arad, g = 0, h = 0, f = 0),

(Sibiu, g = 140, h = 253, f = 393, Cha = Arad),

(R. Vilcea, g = 220, h = 193, f = 413, Cha = Sibiu),

(Fagaras, g = 239, h = 176, f = 415, Cha = Sibiu)}



Từ tập OPEN, nút tốt nhất là Pitesti nên T_{max} = Pitesti. Từ Pitesti ta có thể đi được đến R. Vilcea, Bucharest và Craiova. Lấy Pitesti ra khỏi tập OPEN và đưa vào tập CLOSE. Tương tự ta cũng tính các giá trị h, g và f của các thành phố này.

- ✓ h(R. Vilcea) = 193

  g(R. Vilcea) = g(Pitesti) + cost(Pitesti, R. Vilcea) = 317 + 97 = 414

  f(R. Vilcea) = g(R. Vilcea) + h(R. Vilcea) = 414 + 193 = 607

- ✓ h(Bucharest) = 20

  g(Bucharest) = g(Pitesti) + cost(Pitesti, Bucharest) = 317 + 101 = 418

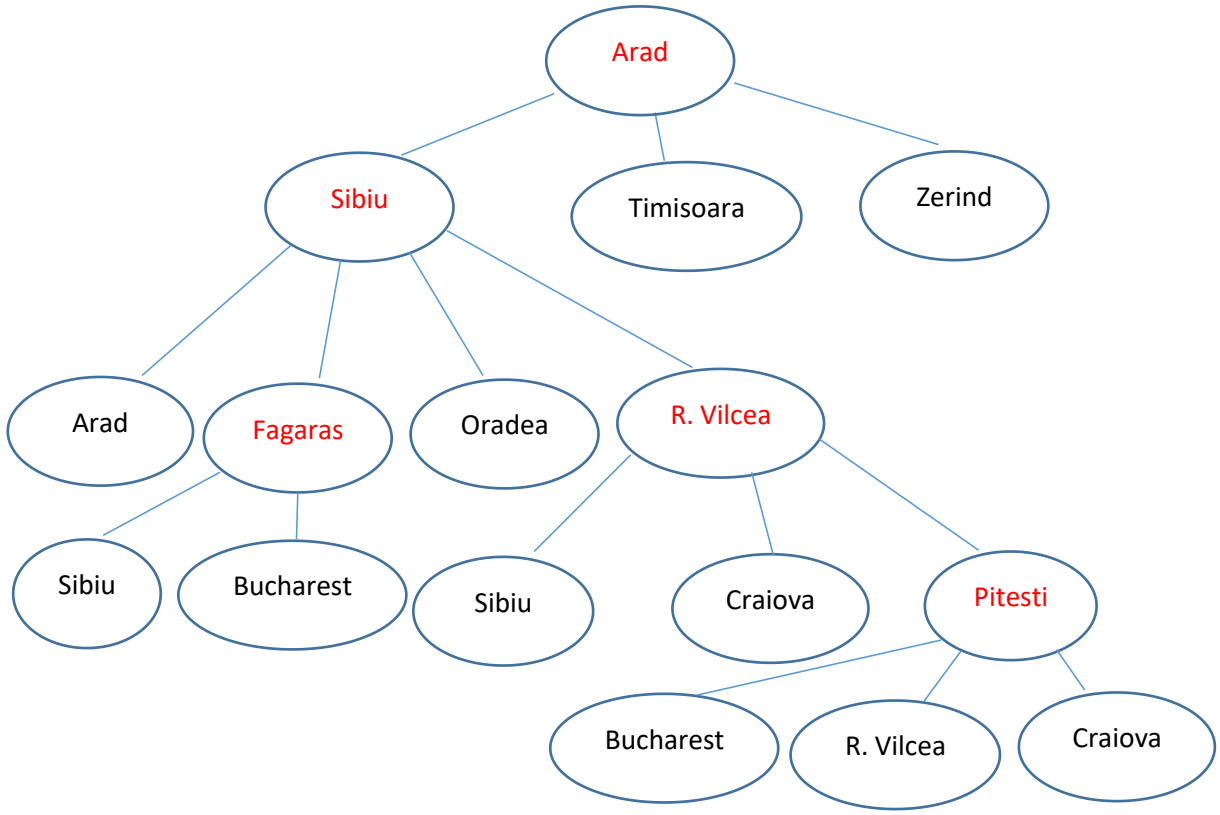
  f(Bucharest) = g(Bucharest) + h(Bucharest) = 418 + 20 = 438

- ✓ h(Craiova) = 160

  g(Craiova) = g(Pitesti) + cost(Pitesti, Craiova) = 317 + 138 = 455

  f(Craiova) = g(Craiova) + h(Craiova) = 455 + 160 = 615

Do R. Vilcea đã có trong CLOSE và g(R. Vilcea) mới được tạo ra có giá trị là 417 lớn hơn g(R. Vilcea) lưu trong CLOSE có giá trị là 220 nên ta sẽ không cập nhật giá trị g và f của R. Vilcea lưu trong CLOSE. Craiova đã có trong OPEN và g(Craiova) mới được tạo ra có giá trị là 455 lớn hơn g(Craiova) trong OPEN có giá trị là 366 nên ta cũng không cập nhật trị g và f của Craiova. Bucharest đã có trong OPEN và g(Bucharest) mới tạo có giá trị là 418 nhỏ hơn g(Bucharest) trong OPEN



có giá trị là 450 nên ta sẽ cập nhật giá trị g và f của Bucharest.

OPEN = {(Timisoara, g = 118, h = 329, f = 447, Cha = Arad),

(Zerind, g = 75, h = 374, f = 449, Cha = Arad),

(Oradea, g = 291, h = 380, f = 617, Cha = Sibiu),

(Craiova, g = 366, h = 160, f = 526, Cha = R. Vilcea),

(Bucharest, g = 418, h = 20, f = 438, Cha = Pitesti)}

CLOSE = {(Arad, g = 0, h = 0, f = 0),

(Sibiu, g = 140, h = 253, f = 393, Cha = Arad),

(R. Vilcea, g = 220, h = 193, f = 413, Cha = Sibiu),

(Fagaras, g = 239, h = 176, f = 415, Cha = Sibiu),

(Pitesti, g = 317, h = 100, f = 417, Cha = R. Vilcea)}

Trong tập OPEN, Bucharest có giá trị f nhỏ nhất nên T_{max} = Bucharest. Từ Bucharest ta có thể được tới 4 thành phố Pitesti, Fagaras, Giurgiu, và Urziceni. Lấy Bucharest ra khỏi tập OPEN và đưa vào tập CLOSE. Tương tự, ta cũng tính giá trị h, g và f của các thành phố này.

- ✓ h(Pitesti) = 100

  g(Pitesti) = g(Bucharest) + cost(Bucharest, Pitesti) = 418 + 101 = 519

  f(Pitesti) = g(Pitesti) + h(Pitesti) = 519 + 100 = 619

- ✓ h(Fagaras) = 176

  g(Fagaras) = g(Bucharest) + cost(Bucharest, Fagaras) = 418 + 211 = 629

  f(Fagaras) = g(Fagaras) + h(Fagaras) = 629 + 176 = 805

- ✓ h(Giurgiu) = 77

  g(Giurgiu) = g(Bucharest) + cost(Bucharest, Giurgiu) = 418 + 90 = 508

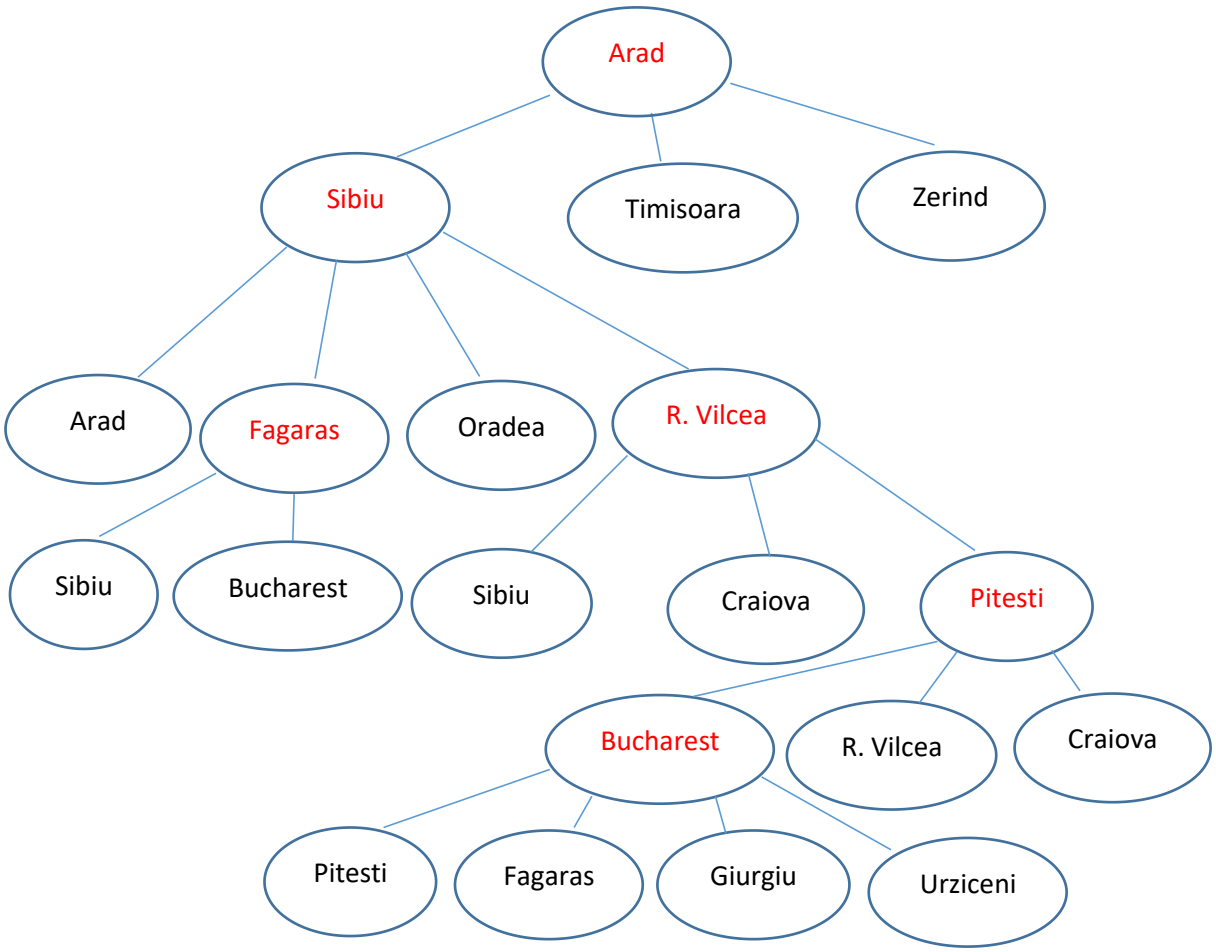
  f(Giurgiu) = g(Giurgiu) + h(Giurgiu) = 508 + 77 = 585

- ✓ h(Urziceni) = 10

  g(Urziceni) = g(Bucharest) + cost(Bucharest, Urziceni) = 418 + 85 = 503

  f(Urziceni) = g(Urziceni) + h(Urziceni) = 503 + 10 = 513

Pitesti và Fagaras đã có trong tập CLOSE và g(Pitesti), g(Fagaras) mới được tạo ra có giá trị lớn hơn g(Pitesti), g(Fagaras) trong tập CLOSE nên ta sẽ không cập nhật giá trị g và f của chúng. Giurgiu và Urziceni không có trong tập OPEN lẫn CLOSE



nên ta sẽ thêm 2 nút này vào tập OPEN.

OPEN = {(Timisoara, g = 118, h = 329, f = 447, Cha = Arad),

(Zerind, g = 75, h = 374, f = 449, Cha = Arad),

(Oradea, g = 291, h = 380, f = 617, Cha = Sibiu),

(Craiova, g = 366, h = 160, f = 526, Cha = R. Vilcea),

(Giurgiu, g = 508, h = 77, f = 585, Cha = Bucharest),

(Urziceni, g = 503, h = 10, f = 513, Cha = Bucharest)}

CLOSE = {(Arad, g = 0, h = 0, f = 0),

(Sibiu, g = 140, h = 253, f = 393, Cha = Arad),

(R. Vilcea, g = 220, h = 193, f = 413, Cha = Sibiu),

(Fagaras, g = 239, h = 176, f = 415, Cha = Sibiu),

(Pitesti, g = 317, h = 100, f = 417, Cha = R. Vilcea),

(Bucharest, g = 418, h = 20, f = 438, Cha = Pitesti)}

Tương tự, ……

❖ Cài đặt:

```
def aStar(start, goal, grid):
    #The open and closed sets
    openset = set()
    closedset = set()
    #Current point is the starting point
    current = start
    #Add the starting point to the open set
    openset.add(current)
    #While the open set is not empty
    while openset:
        #Find the item in the open set with the lowest G + H score
        current = min(openset, key=lambda o:o.G + o.H)
        #If it is the item we want, retrace the path and return it
        if current == goal:
            path = []
            while current.parent:
                path.append(current)
                current = current.parent
            path.append(current)
            return path[::-1]
        #Remove the item from the open set
        openset.remove(current)
        #Add it to the closed set
        closedset.add(current)
        #Loop through the node's children/siblings
        for node in children(current,grid):
            #If it is already in the closed set, skip it
            if node in closedset:
                continue
            #Otherwise if it is already in the open set
            if node in openset:
                #Check if we beat the G score
                new_g = current.G + current.move_cost(node)
                if node.G > new_g:
                    #If so, update the node to have a new parent
                    node.G = new_g
                    node.parent = current
            else:
                #If it isn't in the open set, calculate the G and H score for the node
                node.G = current.G + current.move_cost(node)
                node.H = manhattan(node, goal)
                #Set the parent to our current item
                node.parent = current
                #Add it to the set
                openset.add(node)
    #Throw an exception if there is no path
    raise ValueError('No Path Found')
def next_move(pacman,food,grid):
    #Convert all the points to instances of Node
    for x in xrange(len(grid)):
        for y in xrange(len(grid[x])):
            grid[x][y] = Node(grid[x][y],(x,y))
    #Get the path
    path = aStar(grid[pacman[0]][pacman[1]],grid[food[0]][food[1]],grid)
    #Output the path
    print (len(path) - 1)
    for node in path:
        x, y = node.point
        print (x, y)
```

……