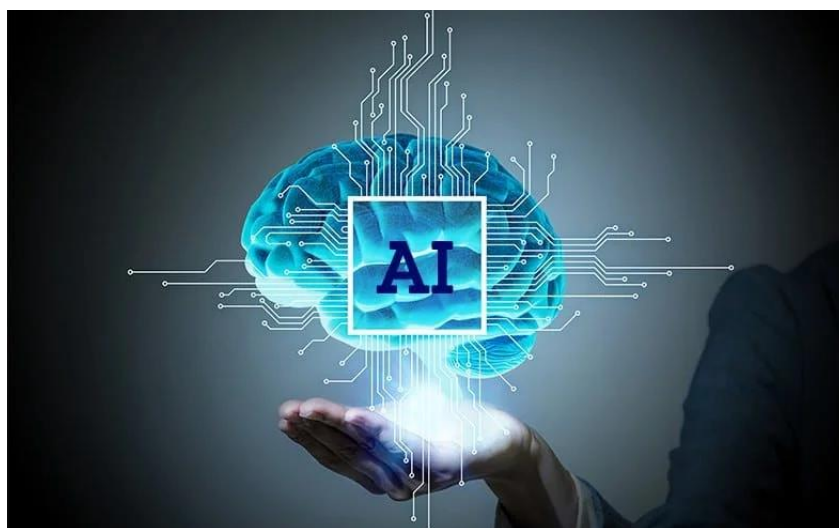


**BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
KHOA TOÁN - TIN HỌC**

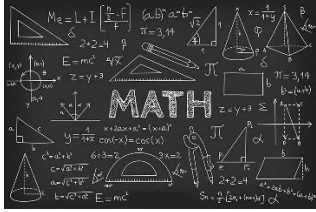


BÀI TẬP THỰC HÀNH TUẦN 3



MÔN HỌC: Nhập môn AI
Sinh Viên: Trần Công Hiếu - 21110294
Lớp: 21TTH_KDL

TP.HCM, ngày 13 tháng 11 năm 2023



TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN TP.HCM
KHOA TOÁN – TIN HỌC



BÀI BÁO CÁO BÀI TẬP THỰC HÀNH TUẦN 3

HK1 - NĂM HỌC: 2023-2024

MÔN: NHẬP MÔN TRÍ TUỆ NHÂN TẠO

SINH VIÊN: TRẦN CÔNG HIẾU

MSSV: 21110294

LỚP: 21TTH_KDL

This image shows a full page of white paper with horizontal dotted lines. The lines are evenly spaced and run across the width of the page, providing a guide for handwriting practice. There are no margins, text, or other markings on the page.

Giảng viên Bộ môn

MUC LUC

I. CÀI ĐẶT VÀ PHÁT HIỆN LỖI.....	5
1. Lỗi trên phần bài lý thuyết.....	5
2. Lỗi trên phần cài đặt.....	5
II. TRÌNH BÀY CHI TIẾT.....	8
1. Trình bày file txt.....	8
2. Trình bày file code.	11
III. CHẠY ĐOẠN MÃ.....	25
1. Chạy tay thuật toán.....	25
2. Chạy đoạn mã.	57
III. NHẬN XÉT.....	58

I. CÀI ĐẶT VÀ PHÁT HIỆN LỖI.

1. Lỗi trên phần bài lý thuyết.

a) Lỗi logic về tập OPEN và 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.

$$OPEN = CLOSE = (Arad, g = 0, h = 0, f = 0)$$

⇒ Theo thuật toán, chọn ra T_{max} từ tập OPEN, lấy nó ra khỏi và đưa qua CLOSE, do đó lúc này:

$$OPEN = \{ \}$$

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

b) Lỗi giá trị g(R.Vilcea).

$$\checkmark h(Craiova) = 160$$

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

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

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 $g(Craiova)$

⇒ Dòng đầu tiên của biện luận, giá trị $g(R.Vilcea)$ mới được tạo ra là 414 chứ không phải 417.

2. Lỗi trên phần cài đặt.

a) Lỗi tên thành phố trong file txt.

```
cities.txt
Arad 29 192
Bucharest 268 55
Craiova 163 22
Dobreta 91 32
Eforie 420 28
Fagaras 208 157
Giurgiu 264 8
Hirsova 396 74
Iasi 347 204
Lugoj 91 98
Mehadia 93 65
Neamt 290 229
Oradea 62 258
Pitesti 220 88
Rimnicu_Vilcea 147 124
Sibiu 126 164
Timisoara 32 124
Urziceni 333 74
Vaslui 376 153
Zerind 44 225
```

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

⇒ Sửa Dobreta thành Drobeta.

b) Lỗi tên file trong hàm.

```
import queue
import matplotlib.pyplot as plt

# getting heuristics from file
def getHeuristics():
    heuristics = {}
    f = open("heuristics1.txt")
    for i in f.readlines():
        node_heuristic_val = i.split()
        heuristics[node_heuristic_val[0]] = int(node_heu

    return heuristics
```

⇒ Sửa tên file sau lệnh open() từ “heuristics1.txt” thành “heuristics.txt”

```
def getCity():
    city = {}
    citiesCode = {}
    f = open("cities1.txt")
    i = 1
```

⇒ Sửa tên file sau lệnh open() từ “cities1.txt” thành “cities.txt”

c) Lỗi thiếu ký tự.

```
# Vẽ hình
def drawMap(city, gbfs, astar, graph):
    for i, j in city.items():
        plt.plot(j[0], [1], "ro")
        plt.annotate(i, (j[0] + 5, j[1]))
```

⇒ Trong đoạn mã, ta đang vẽ các điểm thông qua lệnh plt.plot() với tham số đưa vào là j[0], j[1] và “ro” chứ không phải là j[0], [1] và “ro”.

d) Lỗi chính tả.

```
def drawMap(city, gbfs, astar, graph):
    for i, j in city.items():
        plt.plot(j[0], j[1], "ro")
        plt.annotate(i, (j[0] + 5, j[1]))

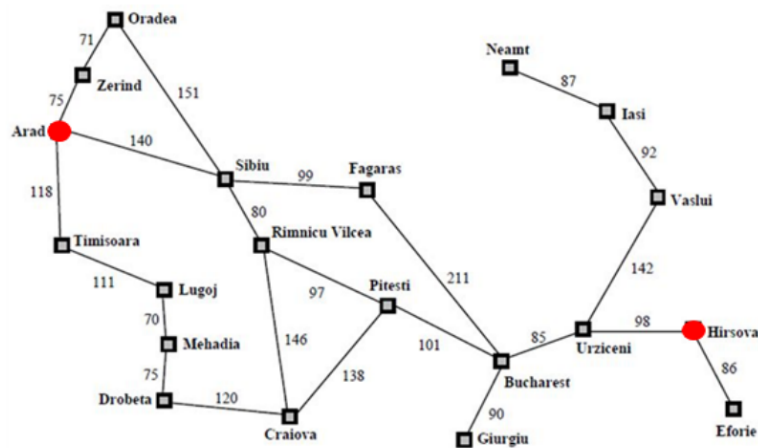
    for k in graph[i]:
        n = city[k[0]]
        plt.plot([j[0], n[0]], [j[1], n[1]], "gray")

    for i in range(len(gbfs)):
        try:
            first = city[gbfs[i]]
            secend = city[gbfs[i + 1]]
```

Trong câu lệnh sau try, ta cần 2 biến để gán giá trị vào. Đầu tiên là first, tiếp theo sau sẽ là second chứ không phải secend. Đây không phải là lỗi quá nghiêm trọng vì trong chương trình thì biến này chỉ mang thông tin được gán và ít được sử dụng, tuy vậy cũng cần lưu ý khi đặt tên cho biến để tốt cho việc trao đổi thông tin, ngữ nghĩa của các biến.

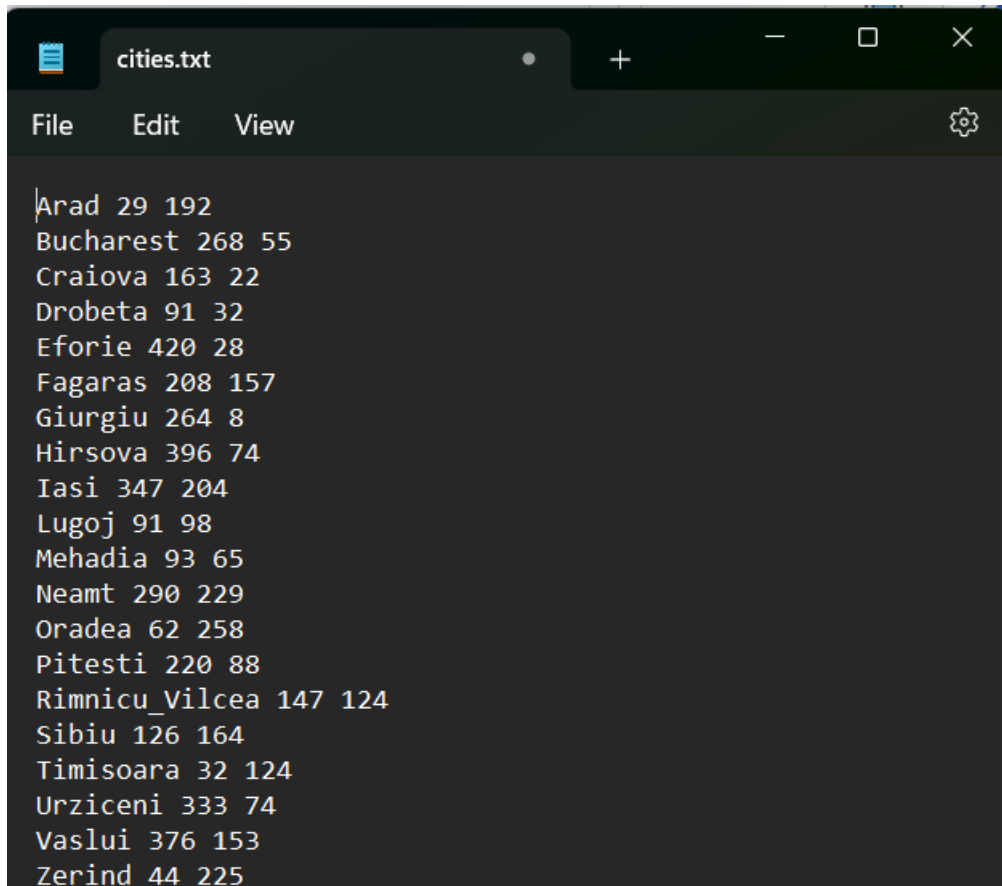
II. TRÌNH BÀY CHI TIẾT.

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



1. Trình bày file txt.

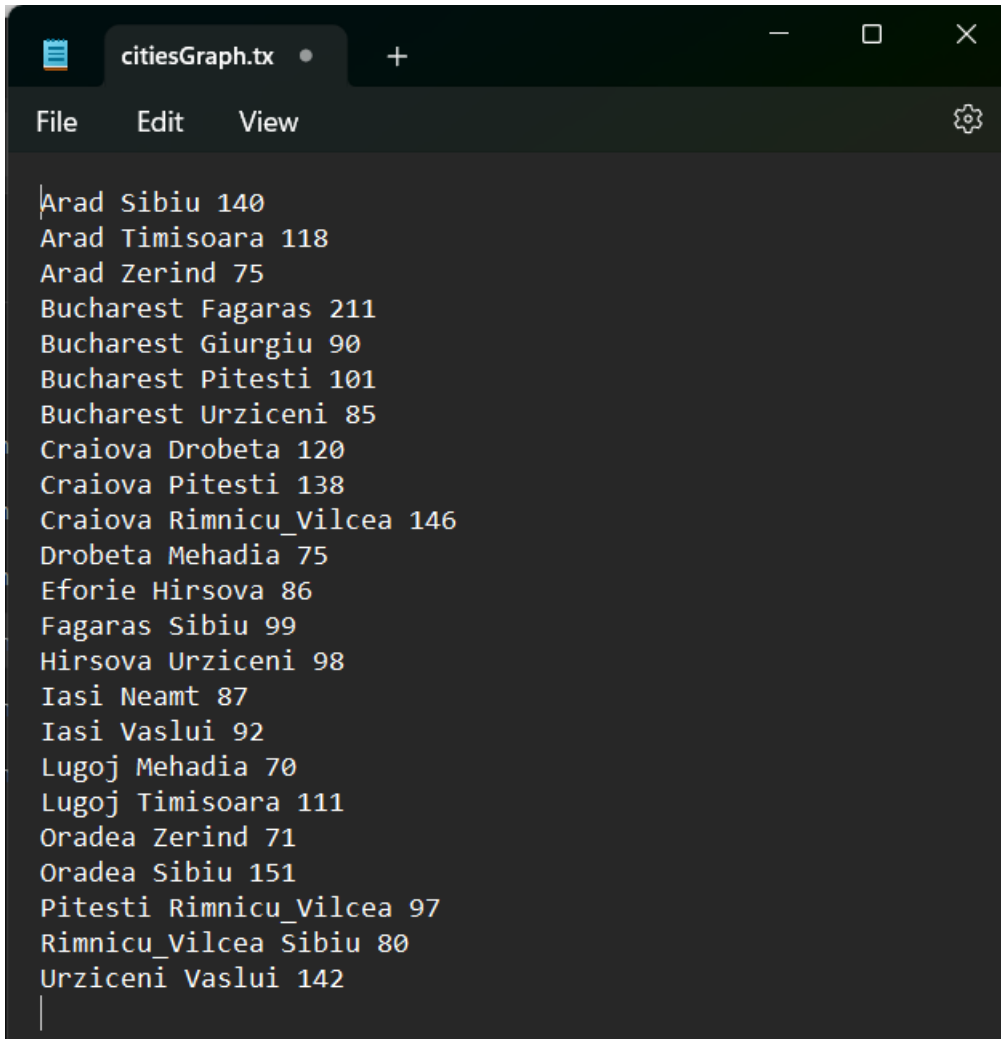
a) File cities.txt.



```
Arad 29 192
Bucharest 268 55
Craiova 163 22
Drobeta 91 32
Eforie 420 28
Fagaras 208 157
Giurgiu 264 8
Hirsova 396 74
Iasi 347 204
Lugoj 91 98
Mehadia 93 65
Neamt 290 229
Oradea 62 258
Pitesti 220 88
Rimnicu_Vilcea 147 124
Sibiu 126 164
Timisoara 32 124
Urziceni 333 74
Vaslui 376 153
Zerind 44 225
```

File chứa các dòng, mỗi dòng có 3 thông tin được ngăn cách nhau bằng dấu cách. Ví dụ cho dòng đầu “Arad 29 192”, mang thông tin là tên thành phố là Arad, có tọa độ x, y sẽ dùng để vẽ biểu đồ là 29, 129.

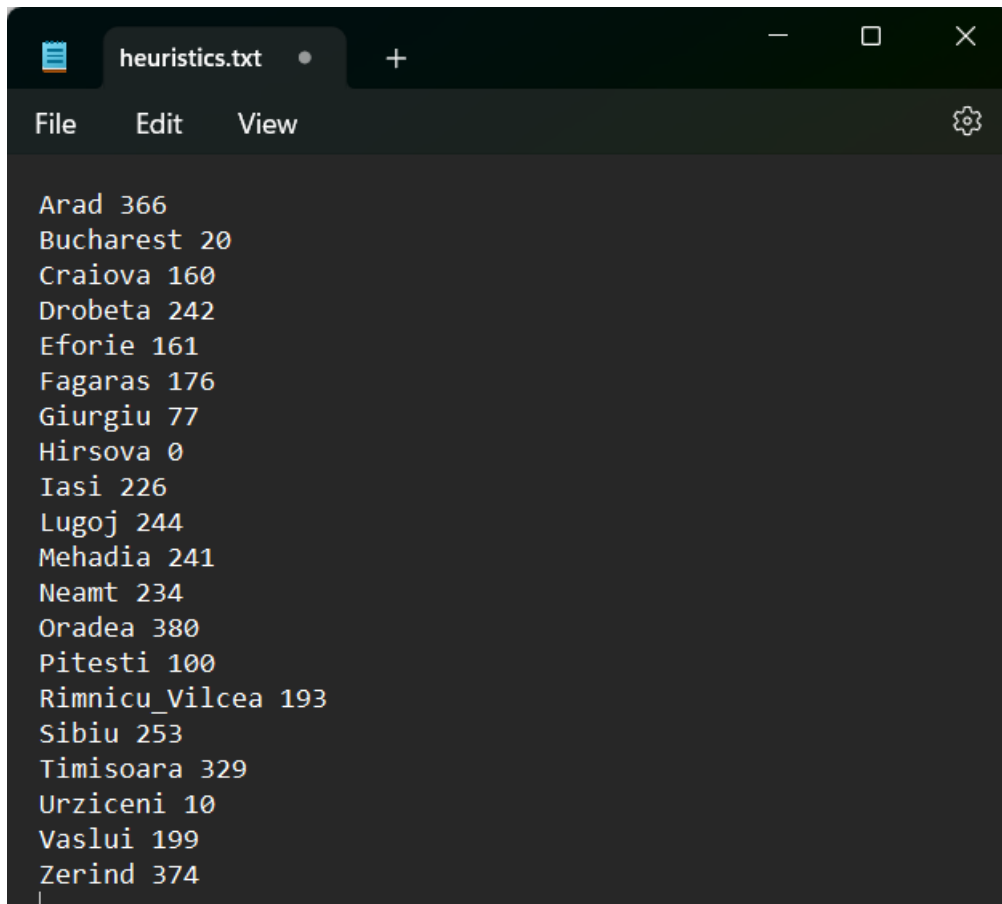
b) File citiesGraph.txt.



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

File chứa các dòng, mỗi dòng có 3 thông tin được ngăn cách nhau bằng dấu cách. Ví dụ cho dòng đầu “Arad Sibiu 140”, mang thông tin là tên thành phố là Arad, có nối đến thành phố Sibiu với khoảng cách là 140.

c) File heuristics.txt.

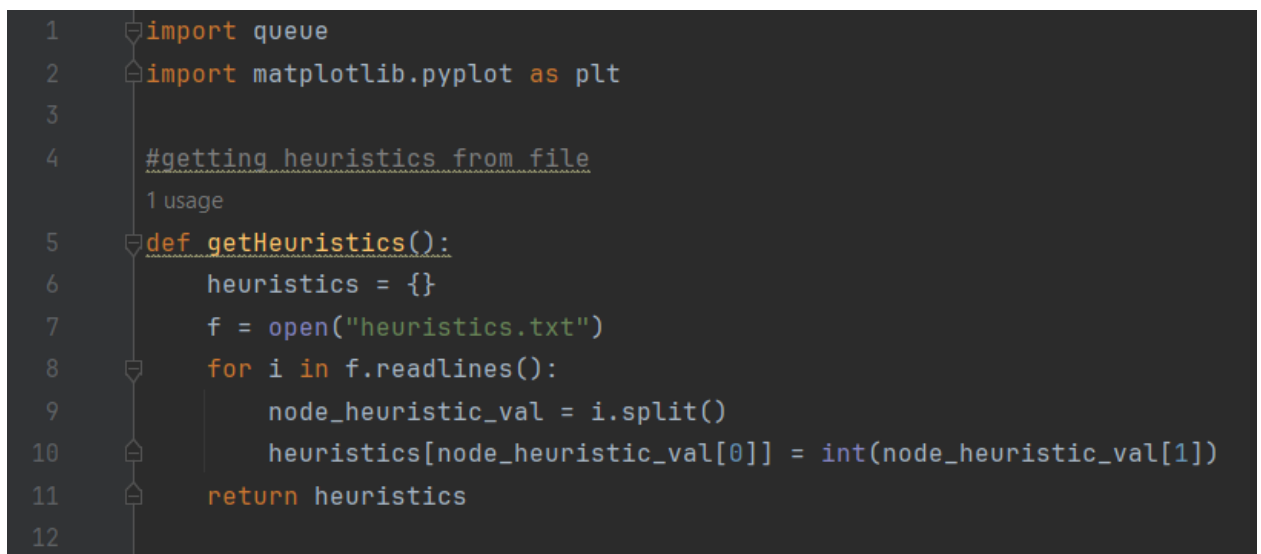


```
File Edit View

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

File chứa các dòng, mỗi dòng có 2 thông tin được ngăn cách nhau bằng dấu cách. Ví dụ cho dòng đầu “Arad 366”, mang thông tin là tên thành phố là Arad, có giá trị heuristic của Arad là 366.

2. Trình bày file code.



```
1  import queue
2  import matplotlib.pyplot as plt
3
4  #getting heuristics from file
5  def getHeuristics():
6      heuristics = {}
7      f = open("heuristics.txt")
8      for i in f.readlines():
9          node_heuristic_val = i.split()
10         heuristics[node_heuristic_val[0]] = int(node_heuristic_val[1])
11     return heuristics
12
```

“import queue”: Import thư viện queue để sử dụng cấu trúc dữ liệu hàng đợi.

“import matplotlib.pyplot as plt”: Import thư viện matplotlib để vẽ đồ thị.

“def getHeuristics()”: Định nghĩa một hàm có tên là getHeuristics dùng để lấy giá trị heuristic cho từng node.

“heuristics = {}”: Khởi tạo một từ điển trống để lưu trữ thông tin về heuristics.

“f = open("heuristics.txt")”: Mở tệp tin "heuristics.txt" để đọc dữ liệu.

“for i in f.readlines()”: Duyệt qua từng dòng trong tệp tin.

“node_heuristic_val = i.split()”: Tách các giá trị trên mỗi dòng bằng cách sử dụng phương thức split(). Kết quả là một danh sách (list) chứa các chuỗi con. Ví dụ cho dòng văn bản đầu tiên trong file "heuristic.txt" là "Arad 366", sau khi áp dụng .split(), node_heuristic_val sẽ trở thành danh sách ['Arad', '366'].

“heuristics[node_heuristic_val[0]] = int(node_heuristic_val[1])”: Thêm một cặp key-value vào từ điển heuristics, trong đó key là phần tử đầu tiên của danh sách chứa tên các vị trí và value là phần tử thứ hai chuyển đổi sang kiểu số nguyên.

“return heuristics”: Hàm trả về từ điển heuristics sau khi đã lấy được thông tin từ tệp tin.

```
13 def getCity():
14     city = {}
15     citiesCode = {}
16     f = open("cities.txt")
17     j = 1
18     for i in f.readlines():
19         node_city_val = i.split()
20         city[node_city_val[0]] = [int(node_city_val[1]), int(node_city_val[2])]
21
22         citiesCode[j] = node_city_val[0]
23         j += 1
24
25     return city, citiesCode
26
```

“city = {}”: Một từ điển để lưu trữ thông tin về thành phố, trong đó key là tên thành phố (node_city_val[i]) và value là một list chứa hai phần tử là cặp giá trị int đại diện cho tọa độ của thành phố.

“citiesCode = {}”: Là một từ điển khác cũng để lưu trữ thông tin về tên thành phố, trong đó key là một số nguyên j tăng dần và value là tên thành phố.

“f = open("cities.txt")”: Mở tệp tin "cities.txt" để đọc dữ liệu.

Sử dụng vòng lặp “for i in f.readlines()” để duyệt qua từng dòng trong tệp tin.

“node_city_val = i.split()”: Tách dòng thành các từ (dùng khoảng trắng làm dấu phân cách) và lưu vào node_city_val. Ví dụ cho dòng văn bản đầu tiên trong file "cities.txt" là "Arad 29 192", sau khi áp dụng .split(), node_city_val sẽ trở thành danh sách ['Arad', '29', '192'].

“city[node_city_val[0]] = [int(node_city_val[1]), int(node_city_val[2])]”: Chuyển đổi chuỗi node_city_val[1] và node_city_val[2] thành kiểu số int. Sau đó gán cho dictionary city với index là tên thành phố tương ứng.

“citiesCode[j] = node_city_val[0]”: Lưu thông tin về tên thành phố vào từ điển citiesCode. Key là một số nguyên tăng dần (j), và value là tên thành phố.

“j += 1”: Tăng giá trị của j lên 1 đơn vị để sử dụng làm key cho citiesCode trong lần lặp tiếp theo.

“return city, citiesCode”: Hàm trả về hai từ điển city và citiesCode.

```

27  def createGraph():
28      graph = {}
29      file = open("citiesGraph.txt")
30      for i in file.readlines():
31          node_val = i.split()
32
33          if node_val[0] in graph and node_val[1] in graph:
34              c = graph.get(node_val[0])
35              c.append([node_val[1], node_val[2]])
36              graph.update({node_val[0]: c})
37
38              c = graph.get(node_val[1])
39              c.append([node_val[0], node_val[2]])
40              graph.update({node_val[1]: c})
41

```

“graph = {}”: Tạo một từ điển trống để biểu diễn đồ thị. Trong từ điển này, các khóa sẽ là tên các thành phố, và giá trị tương ứng là danh sách các thành phố kề và trọng số của các cạnh.

“file = open("citiesGraph.txt")”: Mở tệp tin "citiesGraph.txt" để đọc dữ liệu. Đối tượng file này sẽ được sử dụng để lặp qua từng dòng của tệp tin.

“for i in file.readlines()”: Lặp qua mỗi dòng trong tệp tin, readlines() trả về một danh sách các dòng trong tệp tin.

“node_val = i.split()”: Tách dòng hiện tại thành một danh sách các giá trị, sử dụng khoảng trắng làm dấu phân tách. Kết quả là một danh sách node_val chứa các giá trị từ dòng hiện tại. Ví dụ cho dòng văn bản đầu tiên trong file "citiesGraph.txt" là " Arad Sibiu 140", sau khi áp dụng .split(), node_val sẽ trở thành danh sách ['Arad', ' Sibiu ', '140']. Vậy khi muốn gọi thành phố đầu tiên của dòng trong file thì là node_val[0], tương tự thì node_val[1] là thành phố nối với thành phố trước và node_val[2] là khoảng cách giữa 2 thành phố.

“if node_val[0] in graph and node_val[1] in graph:”: Kiểm tra xem cả hai thành phố từ dòng hiện tại đã được thêm vào đồ thị (graph) hay chưa.

“`c = graph.get(node_val[0])`”: Lấy danh sách các thành phố kề với `node_val[0]` đang xét và trọng số của cạnh kề. Hiểu là `graph` là một dictionary truy xuất value chứa 2 giá trị thông qua key là node hiện tại.

“`c.append([node_val[1], node_val[2]])`”: Thêm một phần tử mới vào danh sách này, đó là một danh sách con chứa tên thành phố kề (`node_val[1]`) và trọng số của cạnh (`node_val[2]`).

“`graph.update({node_val[0]: c})`”: Cập nhật đồ thị với danh sách đã được cập nhật của `node_val[0]`.

Tương tự, với `node_val[1]` chính là node kề với `node_val[0]`, ta thực hiện cùng một quá trình cho thành phố thứ hai (`node_val[1]`).

```
42     elif node_val[0] in graph:
43         c = graph.get(node_val[0])
44         c.append([node_val[1], node_val[2]])
45         graph.update({node_val[0]: c})
46
47         graph[node_val[1]] = [[node_val[0], node_val[2]]]
48
49     elif node_val[1] in graph:
50         c = graph.get(node_val[1])
51         c.append([node_val[0], node_val[2]])
52         graph.update({node_val[1]: c})
53
54         graph[node_val[0]] = [[node_val[1], node_val[2]]]
55
56     else:
57         graph[node_val[0]] = [[node_val[1], node_val[2]]]
58         graph[node_val[1]] = [[node_val[0], node_val[2]]]
59
60     return graph
```

“`elif node_val[0] in graph:`”: Nếu chỉ thành phố đầu tiên `node_val[0]` đã tồn tại trong `graph`, nhưng thành phố thứ hai `node_val[1]` chưa tồn tại, thì chương trình sẽ thực hiện bước lấy danh sách các thành phố kề và trọng số của cạnh kề của `node_val[0]`.

Thêm một phần tử mới vào danh sách này (“`c.append([node_val[1], node_val[2]])`”) chứa tên thành phố thứ hai và trọng số của cạnh kề.

“graph.update({node_val[0]}: c)”: Cập nhật graph với danh sách đã được cập nhật cho node_val[0].

Tiếp theo, chương trình sẽ tạo một cặp key-value mới trong graph, với key là node_val[1] và value là một danh sách mới chứa thông tin về thành phố node_val[0] và trọng số của cạnh kề. Điều này nhằm mô tả mối quan hệ vô hướng giữa 2 thành phố.

“elif node_val[1] in graph:”: Ngược lại, nếu chỉ thành phố thứ hai node_val[1] đã tồn tại trong graph, nhưng thành phố đầu tiên node_val[0] chưa tồn tại, thì chương trình sẽ thực hiện những bước tương tự như trên.

Cụ thể là lấy danh sách các thành phố kề và trọng số của cạnh kề của node_val[1].

Thêm một phần tử mới vào danh sách này (“c.append([node_val[0], node_val[2]])”) chứa tên thành phố đầu tiên và trọng số của cạnh kề.

“graph.update({node_val[1]}: c)”: Cập nhật graph với danh sách đã được cập nhật cho node_val[1].

Tiếp theo, chương trình sẽ tạo một cặp key-value mới trong graph, với key là node_val[0] và value là một danh sách mới chứa thông tin về thành phố node_val[1] và trọng số của cạnh kề.

“else:”: Trong trường hợp cả hai thành phố node_val[0] và node_val[1] đều chưa tồn tại trong graph, chương trình sẽ thực hiện những bước sau:

Tạo một cặp key-value mới trong graph cho node_val[0], với value là một danh sách mới chứa thông tin về thành phố node_val[1] và trọng số của cạnh kề.

Tạo một cặp key-value mới trong graph cho node_val[1], với value là một danh sách mới chứa thông tin về thành phố node_val[0] và trọng số của cạnh kề.

Cuối cùng, hàm trả về đồ thị (graph) sau khi đã xử lý tất cả các dòng trong tệp tin.


```

63 def GBFS(startNode, heuristics, graph, goalNode):
64     priorityQueue = queue.PriorityQueue()
65     priorityQueue.put((heuristics[startNode], startNode))
66
67     path = []
68
69     while priorityQueue.empty() == False:
70         current = priorityQueue.get()[1]
71         path.append(current)
72
73         if current == goalNode:
74             break
75
76         priorityQueue = queue.PriorityQueue()
77
78         for i in graph[current]:
79             if i[0] not in path:
80                 priorityQueue.put((heuristics[i[0]], i[0]))
81
82     return path

```

Định nghĩa hàm GBFS với 4 tham số đầu vào là startNode, heuristics, graph và goalNode.

“priorityQueue = queue.PriorityQueue()” khởi tạo hàng đợi ưu tiên, sử dụng lớp PriorityQueue từ module queue.

“priorityQueue.put((heuristics[startNode], startNode))”: Nút ban đầu (startNode) được thêm vào hàng đợi ưu tiên với giá trị heuristic là ưu tiên.

“path = []”: Khởi tạo một danh sách rỗng path để theo dõi các node đã thăm qua trong quá trình tìm kiếm.

“while priorityQueue.empty() == False”: Bắt đầu một while vòng lặp tiếp tục miễn là hàng đợi ưu tiên không trống. Vòng lặp sẽ thực hiện thuật toán tìm kiếm cho đến khi đạt được nút mục tiêu hoặc tất cả các nút đã được khám phá.

“current = priorityQueue.get()[1]”: Bên trong vòng lặp, nút hiện tại được chọn bằng cách trích xuất phần tử thứ hai (chỉ mục 1) của bộ dữ liệu được trả về bởi priorityQueue.get(). Bộ dữ liệu này chứa giá trị heuristic và nút.

“path.append(current)”: Nút hiện tại được thêm vào path danh sách, cho biết rằng nó đã được truy cập.

“if current == goalNode:” Nếu nút hiện tại là nút mục tiêu, vòng lặp sẽ kết thúc bằng cách sử dụng break. Thuật toán đã tìm thấy đường dẫn từ nút bắt đầu đến nút mục tiêu.

“priorityQueue = queue.PriorityQueue()”: làm mới lại hàng đợi, bởi sau vòng lặp while thứ 2 nếu không làm mới lại sẽ dẫn đến các node kề của node trước đó còn nằm trong hàng đợi, vô tình khi heuristic của các node đó lại bé hơn heuristic của các node kề của current thì sẽ dẫn đến sai sót.

“for i in graph[current]:”: Vòng lặp này duyệt qua tất cả các node kề của node hiện tại (current). Biến i sẽ lần lượt đại diện cho mỗi node kề.

“if i[0] not in path:”: Kiểm tra xem node kề i có thuộc đường đi đã được thăm (path) hay không. Nếu node kề chưa được thăm, thì tiếp tục xử lý. Điều này giúp tránh việc lặp lại các node đã được thăm.

“priorityQueue.put((heuristics[i[0]], i[0]))”: Thêm node kề i vào hàng đợi ưu tiên (priorityQueue). Đối số của put là một tuple gồm hai phần tử: heuristic của node kề và chính node kề đó i[0]. Hàng đợi ưu tiên sẽ sắp xếp các đỉnh theo giá trị heuristic, đưa đỉnh với heuristic thấp nhất lên đầu để ưu tiên thăm trước.

“return path”: Cuối cùng, hàm trả về danh sách các node đã truy cập (path). Lưu ý rằng nếu không đạt được nút mục tiêu, danh sách này thể hiện đường dẫn được khám phá cho đến khi thuật toán kết thúc.

```
85 def Astar(startNode, heuristics, graph, goalNode):
86     priorityQueue = queue.PriorityQueue()
87     distance = 0
88     path = []
89
90     priorityQueue.put((heuristics[startNode] + distance, [startNode, 0]))
91
92     while priorityQueue.empty() == False:
93         current = priorityQueue.get()[1]
94         path.append(current[0])
95         distance += int(current[1])
96
97         if current[0] == goalNode:
98             break
99
```

Định nghĩa hàm Astar với 4 tham số đầu vào là startNode, heuristics, graph và goalNode.

“priorityQueue = queue.PriorityQueue()” khởi tạo hàng đợi ưu tiên, sử dụng lớp PriorityQueue từ module queue sắp xếp các nút dựa trên giá trị ước lượng của hàm chi phí (f-cost).

“distance = 0”: Khởi tạo biến lưu tổng chi phí đi từ đỉnh xuất phát đến đỉnh hiện tại.

“path = []”: Danh sách rỗng lưu trữ đường đi từ đỉnh xuất phát đến đỉnh hiện tại.

“priorityQueue.put((heuristics[startNode] + distance, [startNode, 0]))”: Đưa đỉnh xuất phát vào hàng đợi ưu tiên với giá trị ước lượng f-cost là tổng của heuristics và distance, và đường đi hiện tại là [startNode, 0] với 0 là chi phí hiện tại.

“while priorityQueue.empty() == False”: Lặp cho đến khi hàng đợi ưu tiên trống rỗng.

“current = priorityQueue.get()[1]”: Lấy nút có giá trị ước lượng f-cost thấp nhất từ hàng đợi ưu tiên. Tham số [1] cho biết current sẽ lấy [startNode, 0] trong hàng đợi. Tức current[0] sẽ là startNode.

“path.append(current[0])”: Thêm đỉnh hiện tại vào đường đi.

“distance += int(current[1])”: Cập nhật tổng chi phí bằng cách cộng thêm số nguyên được ép kiểu từ current[1] trả về f-cost.

“if current[0] == goalNode”: Nếu đỉnh hiện tại current[0] là node đích thì break để thoát khỏi vòng lặp.

```
100         priorityQueue = queue.PriorityQueue()
101
102         for i in graph[current[0]]:
103             if i[0] not in path:
104                 priorityQueue.put((heuristics[i[0]] + int(i[1]) + distance, i))
105
106         return path
```

“priorityQueue = queue.PriorityQueue()”: Khởi tạo lại hàng đợi ưu tiên để xử lý các đỉnh kề của node hiện tại.

“for i in graph[current[0]]”: Duyệt qua các node kề của node hiện tại.

“if i[0] not in path:”: Nếu node kề chưa được thăm, đưa nó vào hàng đợi ưu tiên với giá trị ước lượng f-cost được tính dựa trên heuristics, chi phí từ node hiện tại đến node kề và tổng chi phí đã đi được.

Sau cùng là “return path” để trả về đường đi từ đỉnh xuất phát đến đỉnh đích.

```
110 def drawMap(city, gbfs, astar, graph):
111     for i, j in city.items():
112         plt.plot(*args: j[0], j[1], "ro")
113         plt.annotate(i, xy: (j[0] + 5, j[1]))
114
115     for k in graph[i]:
116         n = city[k[0]]
117         plt.plot(*args: [j[0], n[0]], [j[1], n[1]], "gray")
118
119     for i in range(len(gbfs)):
120         try:
121             first = city[gbfs[i]]
122             second = city[gbfs[i + 1]]
123
124             plt.plot(*args: [first[0], second[0]], [first[1], second[1]], "green")
125         except:
126             continue
```

Định nghĩa hàm drawMap với 4 tham số đầu vào là city, gbfs, astar và graph.

1. Vòng lặp for đầu tiên.

Vòng lặp “for i, j in city.items()”: với city.items() trả về các cặp key-value từ dictionary city, trong đó key là tên của thành phố và value là một tuple chứa tọa độ (hoành độ, tung độ) của thành phố đó trên bản đồ.

Biến i chứa tên của thành phố và j chứa tuple tọa độ tương ứng. Vẽ điểm đỏ “plt.plot(j[0], j[1], "ro")”: plt.plot được sử dụng để vẽ điểm trên đồ thị. j[0] và j[1] lần lượt là hoành độ và tung độ của thành phố. "ro" chỉ định loại đường vẽ ("r" là màu đỏ và "o" là hình tròn), do đó, điểm được vẽ sẽ có màu đỏ và có hình dạng là hình tròn.

Chú thích tên thành phố “plt.annotate(i, (j[0] + 5, j[1]))”: plt.annotate được sử dụng để thêm chú thích vào đồ thị. Trong trường hợp này, chú thích là tên của thành phố. i là tên thành phố cần chú thích. (j[0] + 5, j[1]) là tọa độ nơi chú thích được đặt. Ở đây, chú thích được đặt tại vị trí nằm bên phải của điểm đại diện cho thành phố, cách xa 5 đơn vị hoành độ.

Vòng lặp “for k in graph[i]:”: Trong đó, graph[i] trả về danh sách các thành phố kết nối trực tiếp với thành phố hiện tại i trong đồ thị. k là một cặp giá trị trong danh sách này, trong đó k[0] là tên của thành phố kết nối với thành phố hiện tại.

Lấy tọa độ của thành phố kết nối “n = city[k[0]]”. Với k[0] là tên của thành phố kết nối với thành phố hiện tại. Thì khi đó city[k[0]] trả về tuple chứa tọa độ của thành phố kết nối.

Vẽ đoạn đường màu xám “plt.plot([j[0], n[0]], [j[1], n[1]], "gray")”: plt.plot được sử dụng để vẽ đoạn đường nối giữa hai thành phố trên đồ thị. [j[0], n[0]] là danh sách chứa hoành độ của thành phố hiện tại và thành phố kết nối. [j[1], n[1]] là danh sách chứa tung độ tương ứng. "gray" chỉ định màu của đoạn đường là màu xám.

2. Vòng lặp for thứ hai.

Vòng lặp “for i in range(len(gbfs)):”: Duyệt qua từng thành phố trong danh sách gbfs, đây là thứ tự các thành phố được duyệt qua bởi thuật toán GBFS.

Cố gắng thực hiện các dòng code trong khối try. Nếu có bất kỳ lỗi nào xảy ra, chương trình sẽ nhảy đến khối except mà không dừng lại vì lệnh continue.

Gán tọa độ của các thành phố “first = city[gbfs[i]]” và “second = city[gbfs[i + 1]]”: Trong đó, gbfs[i] là tên của thành phố hiện tại trong danh sách GBFS và city[gbfs[i]] trả về tọa độ của thành phố đó trong city. Và gbfs[i + 1] là tên của thành phố tiếp theo trong danh sách GBFS và city[gbfs[i + 1]] trả về tọa độ của thành phố tiếp theo.

Vẽ đoạn đường màu xanh “plt.plot([first[0], second[0]], [first[1], second[1]], "green")”: plt.plot được sử dụng để vẽ đoạn đường nối giữa hai thành phố liên tiếp trên đồ thị. [first[0], second[0]] là danh sách chứa hoành độ của thành phố hiện tại và thành phố tiếp theo. [first[1], second[1]] là danh sách chứa tung độ tương ứng. "green" chỉ định màu của đoạn đường là màu xanh.

Nếu có bất kỳ lỗi nào xảy ra trong khối try, chương trình sẽ tiếp tục thực hiện vòng lặp và không bị gián đoạn.

```

128     for i in range(len(aster)):
129         try:
130             first = city[aster[i]]
131             second = city[aster[i + 1]]
132
133             plt.plot(*args: [first[0], second[0]], [first[1], second[1]], "blue")
134         except:
135             continue
136
137     plt.errorbar(x=1, y=1, label="GBFS", color="green")
138     plt.errorbar(x=1, y=1, label="ASTAR", color="blue")
139     plt.legend(loc="lower left")
140
141     plt.show()
142

```

3. Vòng lặp for thứ ba.

Vòng lặp “for i in range(len(aster))”: Duyệt qua từng thành phố trong danh sách aster, đây là thứ tự các thành phố được duyệt qua bởi thuật toán A*.

Cố gắng thực hiện các dòng code trong khối try. Nếu có bất kỳ lỗi nào xảy ra, chương trình sẽ nhảy đến khối except mà không dừng lại.

Gán tọa độ của các thành phố “first = city[aster[i]] và second = city[aster[i + 1]]”. Trong đó:

- + aster[i] là tên của thành phố hiện tại trong danh sách A*.
- + city[aster[i]] trả về tọa độ của thành phố đó trong city.
- + aster[i + 1] là tên của thành phố tiếp theo trong danh sách A*.
- + city[aster[i + 1]] trả về tọa độ của thành phố tiếp theo.

Vẽ đoạn đường màu xanh “plt.plot([first[0], second[0]], [first[1], second[1]], "blue")”: plt.plot được sử dụng để vẽ đoạn đường nối giữa hai thành phố liên tiếp trên đồ thị. Với [first[0], second[0]] là danh sách chứa hoành độ của thành phố hiện tại và thành phố tiếp theo. [first[1], second[1]] là danh sách chứa tung độ tương ứng. "blue" chỉ định màu của đoạn đường là màu xanh.

Nếu có bất kỳ lỗi nào xảy ra trong khối try, chương trình sẽ tiếp tục thực hiện vòng lặp và không bị gián đoạn.

```

137 plt.errorbar(x=1, y=1, label="GBFS", color="green")
138 plt.errorbar(x=1, y=1, label="ASTAR", color="blue")
139 plt.legend(loc="lower left")
140
141 plt.show()
142

```

“plt.errorbar(1, 1, label="GBFS", color="green")”: Được sử dụng để tạo một điểm (1, 1) với màu xanh lá ("green"). Với label="GBFS" được sử dụng để đặt nhãn cho điểm này là "GBFS".

plt.errorbar(1, 1, label="ASTAR", color="blue"): Tương tự như trên, đang được sử dụng để tạo một điểm (1, 1) với màu xanh ("blue"). Với label="ASTAR" được sử dụng để đặt nhãn cho điểm này là "ASTAR".

“plt.legend(loc="lower left")”: plt.legend được sử dụng để thêm hình chú thích vào đồ thị, hiển thị các nhãn đã được đặt trước đó. loc="lower left" chỉ định vị trí của hình chú thích ở góc dưới bên trái của đồ thị.

“plt.show()”: Hiển thị đồ thị.

```

143 #run_code
144 if __name__ == "__main__":
145     heuristic = getHeuristics()
146     graph = createGraph()
147     city, citiesCode = getCity()
148
149     for i, j in citiesCode.items():
150         print(i, j)
151
152     while True:
153         inputCode1 = int(input("Nhập đỉnh bắt đầu: "))
154         inputCode2 = int(input("Nhập đỉnh kết thúc: "))
155
156         if inputCode1 == 0 or inputCode2 == 0:
157             break
158

```

“__name__” là một biến đặc biệt trong Python. Khi một chương trình Python được thực thi, biến __name__ của nó sẽ được đặt là "__main__". Do đó, if __name__ == "__main__": kiểm tra xem đoạn mã đang chạy có phải là chương trình chính hay không. Nếu nó là chương trình

chính (được thực thi trực tiếp), các dòng code bên trong khối này sẽ được thực thi. Nếu nó được import làm module trong một chương trình khác, khối này sẽ được bỏ qua.

“`heuristic = getHeuristics()`”: Gọi hàm `getHeuristics()`, để lấy thông tin về các heuristic (ước lượng chi phí) cho các thành phố trong bản đồ. Kết quả của hàm được gán vào biến `heuristic`.

“`graph = createGraph()`”: Gọi hàm `createGraph()`, để tạo và trả về đồ thị của các thành phố, mô tả các kết nối giữa chúng. Kết quả của hàm được gán vào biến `graph`.

“`city, citiesCode = getCity()`”: Gọi hàm `getCity()`, một hàm để lấy thông tin về các thành phố (`citiesCode`) và tọa độ của chúng `city`. Kết quả của hàm được gán vào hai biến: `city` (là một từ điển chứa vị trí của các thành phố) và `citiesCode` (là một danh sách tên của các thành phố).

Vòng lặp `for` duyệt các cặp trong dictionary `citiesCode`. Sau đó in ra để người dùng dễ dàng chọn lựa thành phố đầu và điểm đến cuối cùng.

Vòng lặp vô hạn “`while True:`”: Tạo một vòng lặp vô hạn để liên tục nhận input từ người dùng và thực hiện công việc liên quan.

Nhập đỉnh bắt đầu `inputCode1` và đỉnh kết thúc `inputCode2`. Sử dụng hàm `input` để nhận input từ người dùng, và sau đó chuyển đổi input thành kiểu số nguyên bằng `int(input(...))`.

“`if inputCode1 == 0 or inputCode2 == 0:`”: Kiểm tra nếu người dùng nhập vào một giá trị 0 cho đỉnh bắt đầu hoặc kết thúc thì thoát khỏi vòng lặp bằng lệnh `break`.

```
159     startCity = citiesCode[inputCode1]
160     endCity = citiesCode[inputCode2]
161
162     gbfs = GBFS(startCity, heuristic, graph, endCity)
163     astar = Astar(startCity, heuristic, graph, endCity)
164     print("GBFS => ", gbfs)
165     print("ASTAR => ", astar)
166     drawMap(city, gbfs, astar, graph)
167
```

Lấy thông tin về thành phố bắt đầu và kết thúc từ điển `citiesCode`, với index lần lượt là 2 đỉnh đầu vào bằng câu lệnh “`startCity = citiesCode[inputCode1]`” và “`endCity = citiesCode[inputCode2]`”.

Sử dụng hàm GBFS và Astar đã định nghĩa ở trên để tìm đường đi giữa hai thành phố bắt đầu và kết thúc, giá trị trả về gán lần lượt gbfs và astar.

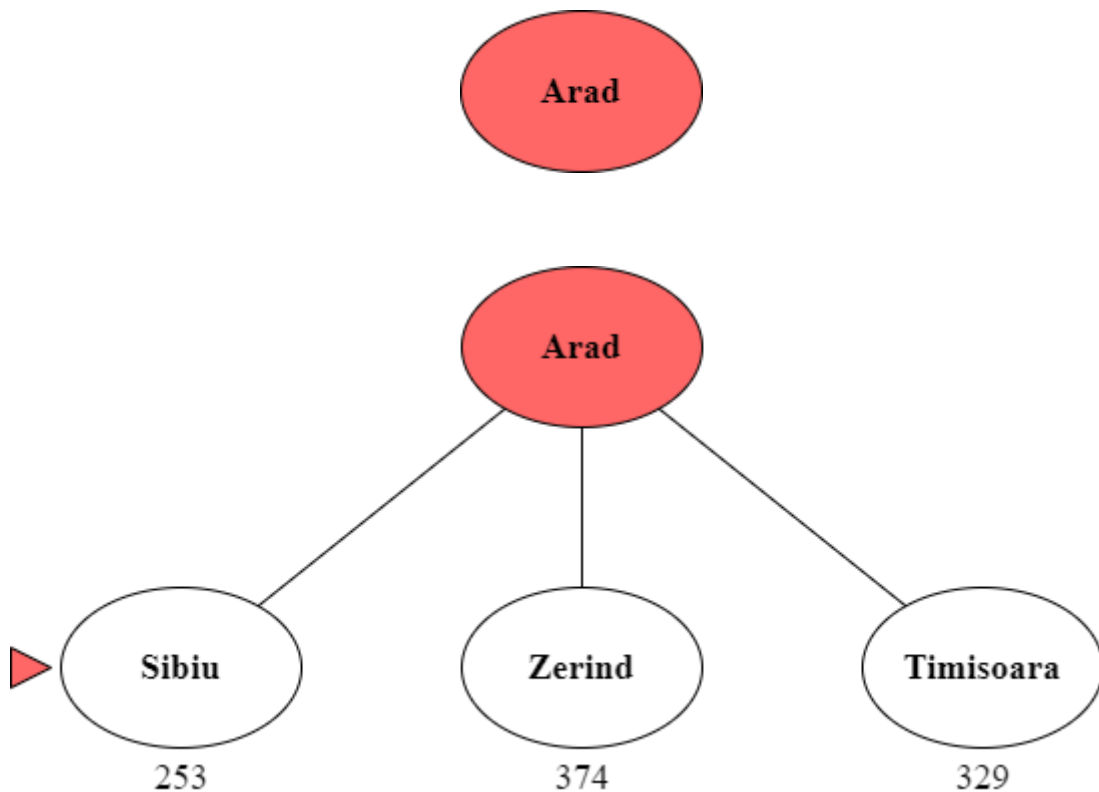
Lần lượt in các path dẫn được trả về từ hàm GBFS() và Astar() thông qua câu lệnh print().

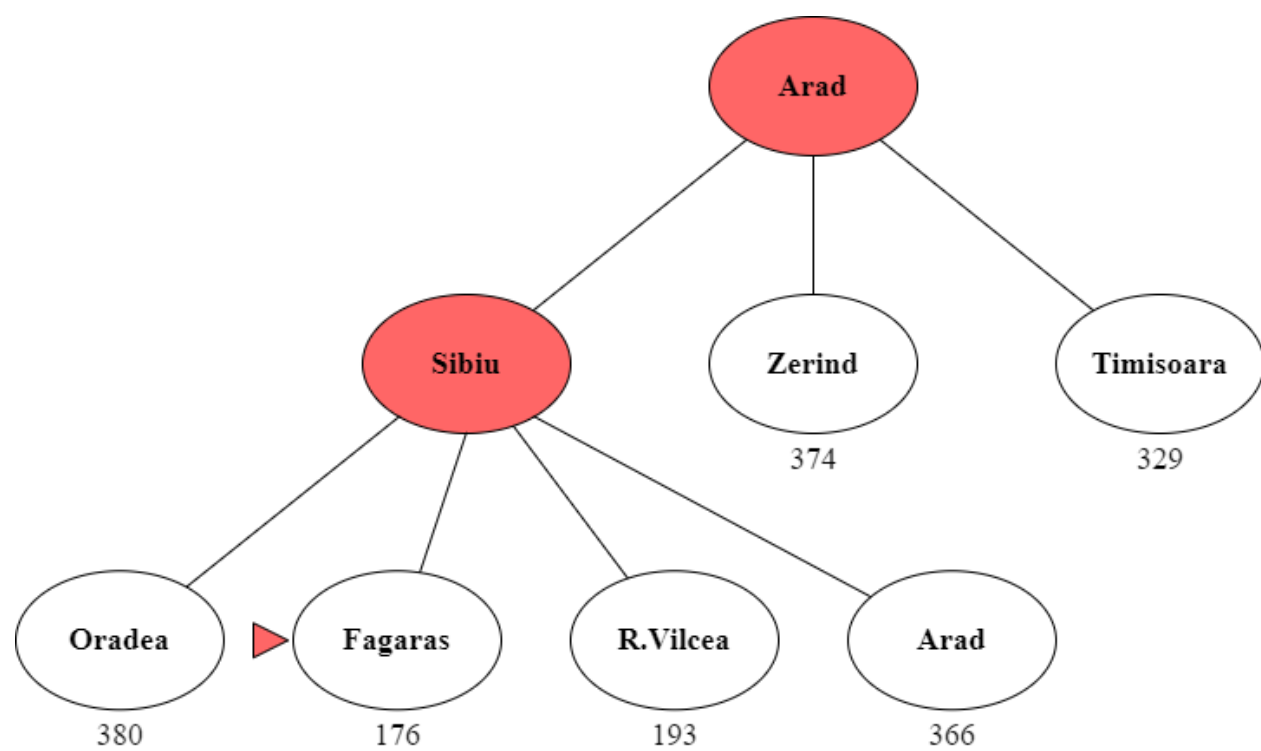
Các tham số truyền vào là thông tin về thành phố bắt đầu, heuristic, đồ thị và thành phố kết thúc.

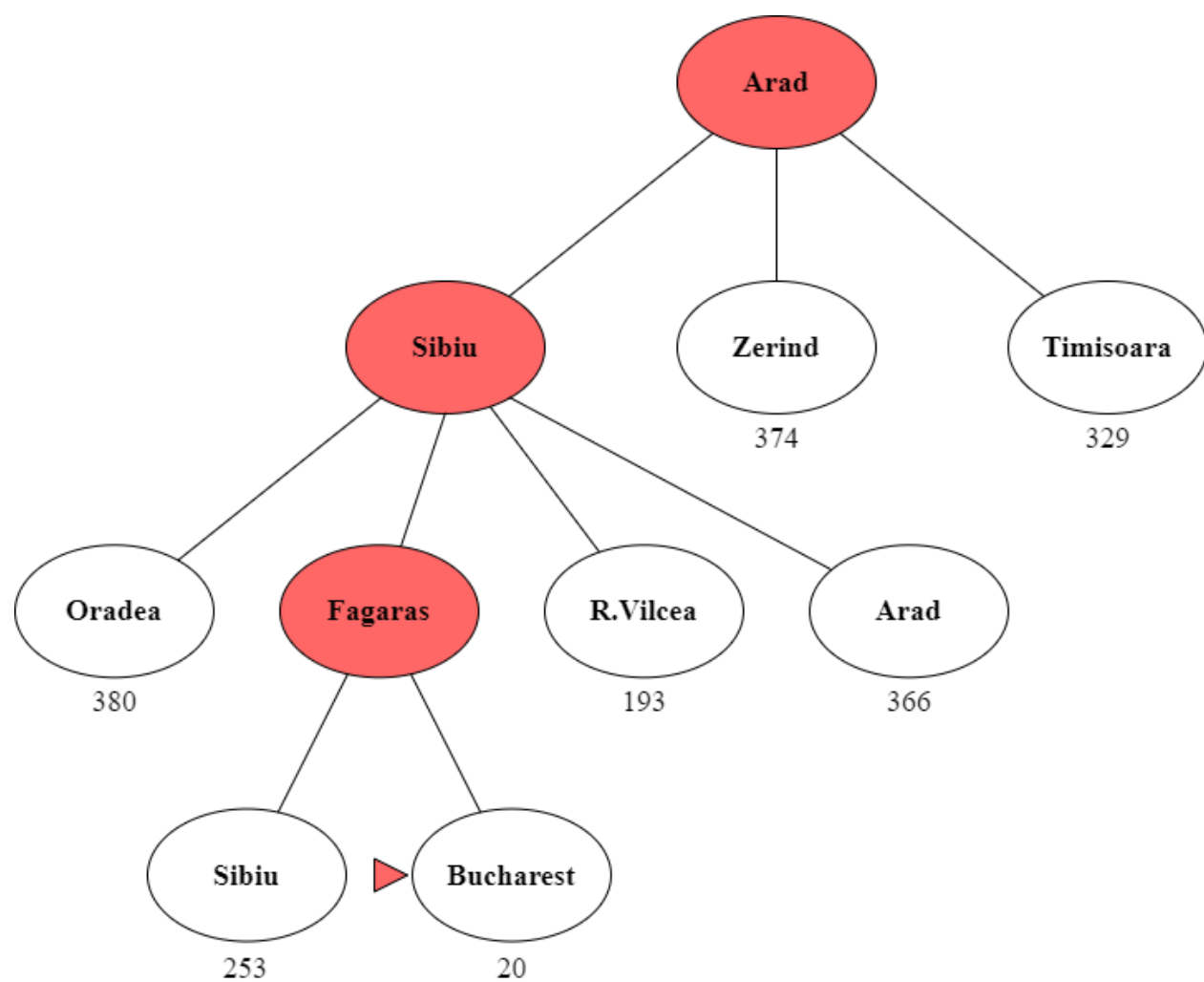
III. CHẠY ĐOẠN MÃ.

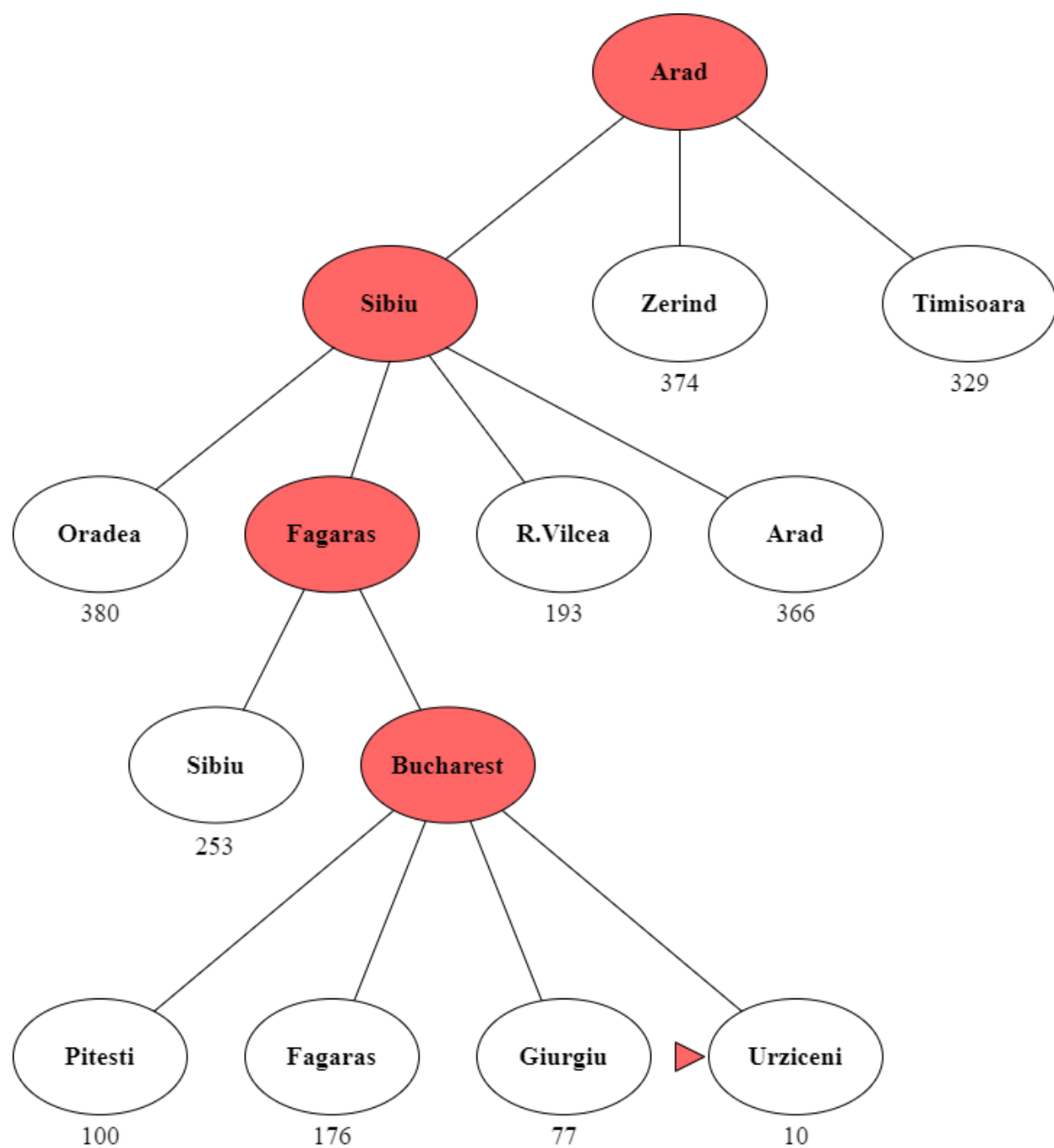
1. Chạy tay thuật toán.

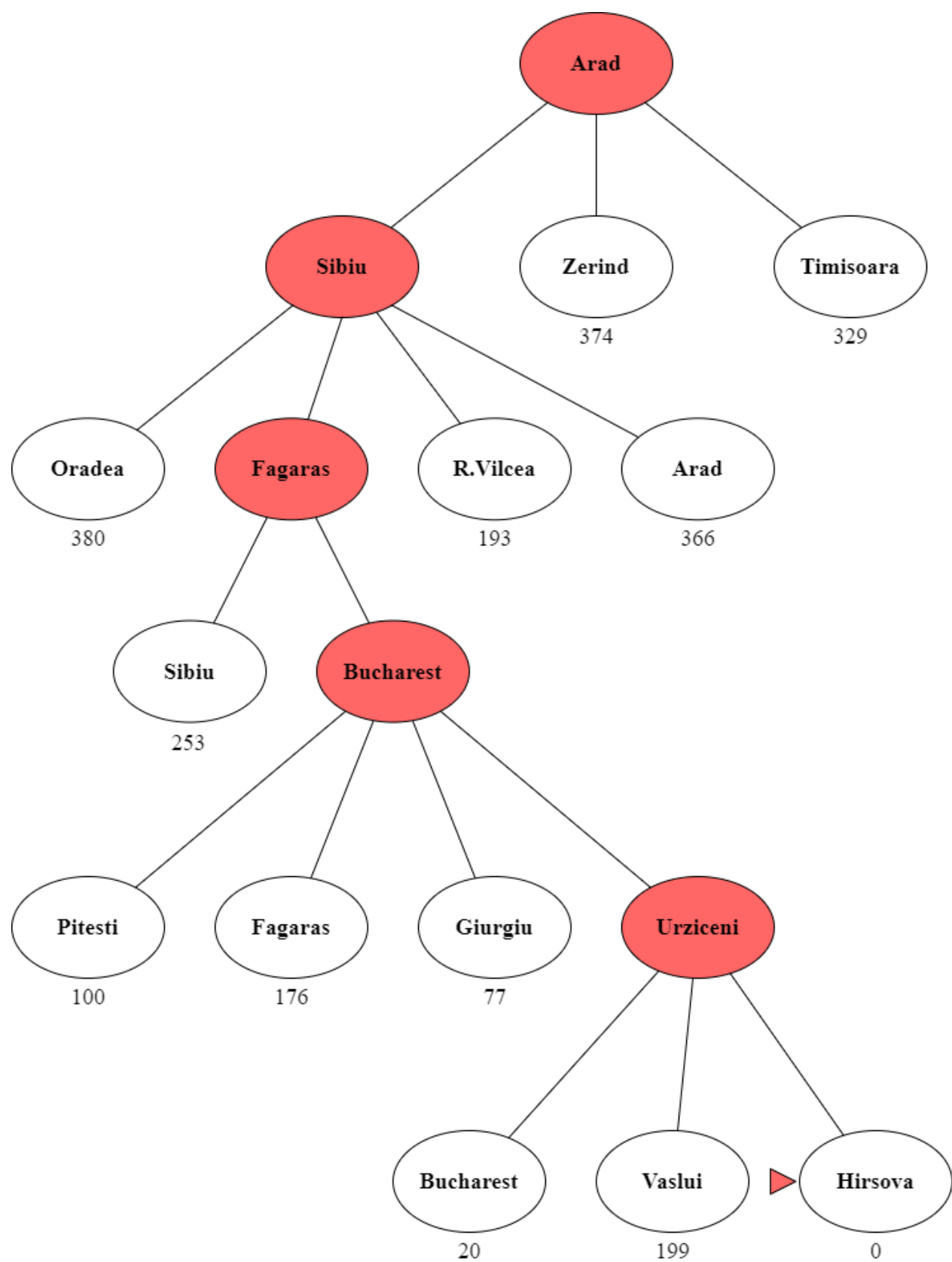
a) Chạy tay thuật toán GBFS.

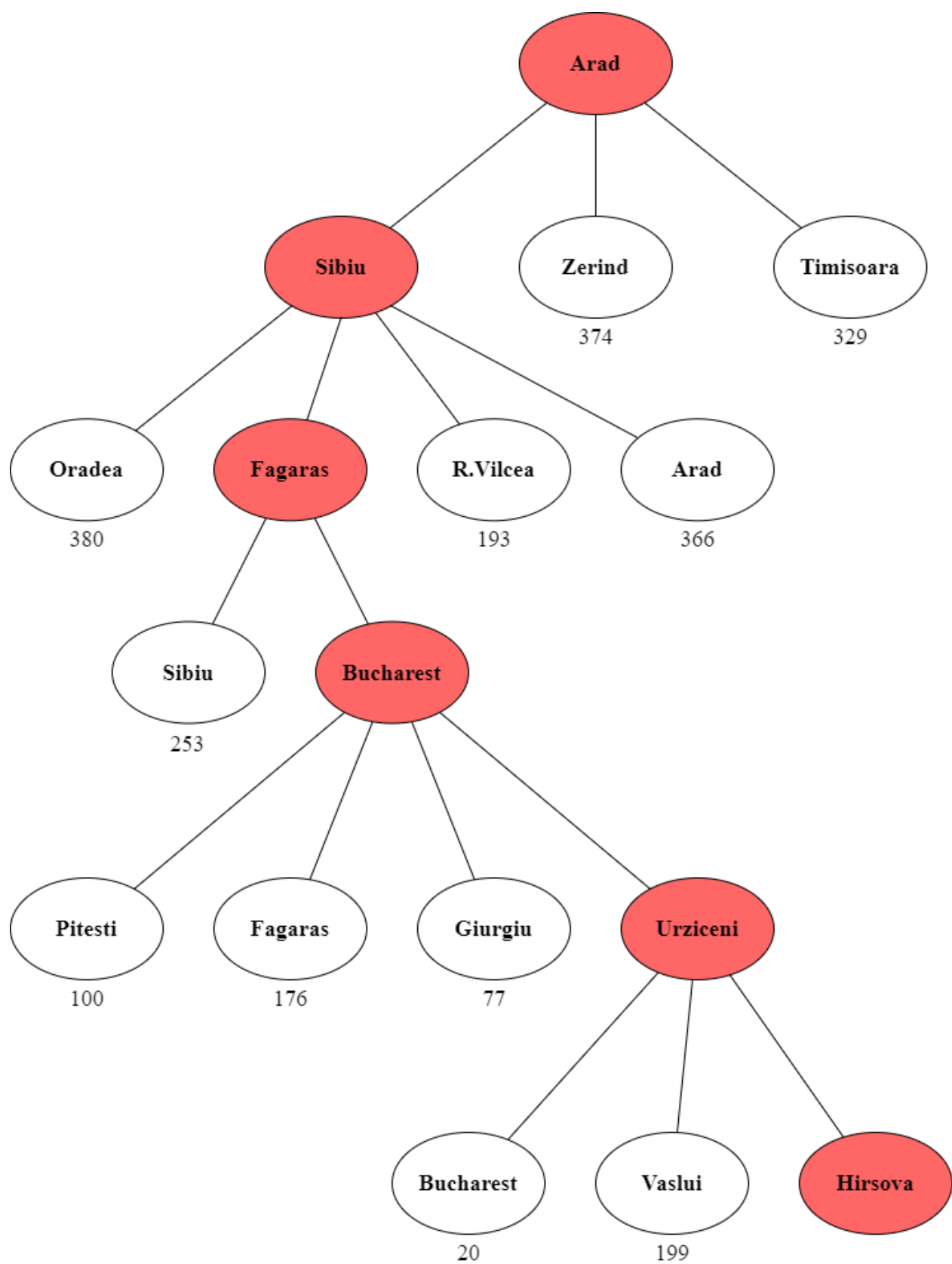












Kết quả:

GBFS => ['Arad', 'Sibiu', 'Fagaras', 'Bucharest', 'Urziceni', 'Hirsova']

b) Chạy tay thuật toán A*.

Ban đầu

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

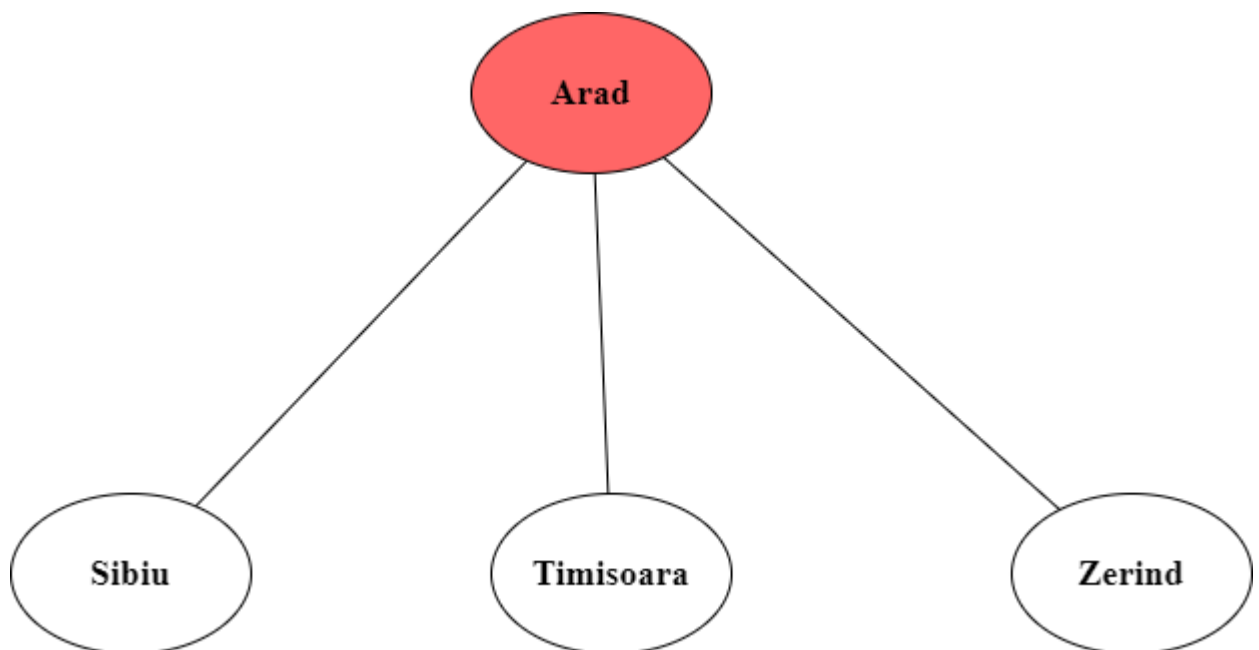
CLOSE = { }

Do OPEN chỉ chứa có 1 thành phố nên thành phố này sẽ là thành phố tốt nhất. Nghĩa là ta chọn $T_{\max} = \text{Arad}$. 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 này chưa có nút cha ban đầu nên nút cha của chúng đều là Arad.



✓ $h(\text{Sibiu}) = 253$

$g(\text{Sibiu}) = g(\text{Arad}) + \text{cost}(\text{Arad}, \text{Sibiu}) = 0 + 140 = 140$

$f(\text{Sibiu}) = g(\text{Sibiu}) + h(\text{Sibiu}) = 140 + 253 = 393$

Cha(Sibiu) = Arad

✓ $h(\text{Timisoara}) = 329$

$$g(\text{Timisoara}) = g(\text{Arad}) + \text{cost}(\text{Arad}, \text{Timisoara}) = 0 + 118 = 118$$

$$f(\text{Timisoara}) = g(\text{Timisoara}) + h(\text{Timisoara}) = 118 + 329 = 447$$

$$\text{Cha}(\text{Timisoara}) = \text{Arad}$$

$$\checkmark h(\text{Zerind}) = 374$$

$$g(\text{Zerind}) = g(\text{Arad}) + \text{cost}(\text{Arad}, \text{Zerind}) = 0 + 75 = 75$$

$$f(\text{Zerind}) = g(\text{Zerind}) + h(\text{Zerind}) = 75 + 374 = 449$$

$$\text{Cha}(\text{Zerind}) = \text{Arad}$$

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

$$\begin{aligned} \text{OPEN} = \{ & (\text{Sibiu}, g = 140, h = 253, f = 393, \text{Cha} = \text{Arad}), \\ & (\text{Timisoara}, g = 118, h = 329, f = 447, \text{Cha} = \text{Arad}), \\ & (\text{Zerind}, g = 75, h = 374, f = 449, \text{Cha} = \text{Arad}) \} \end{aligned}$$

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

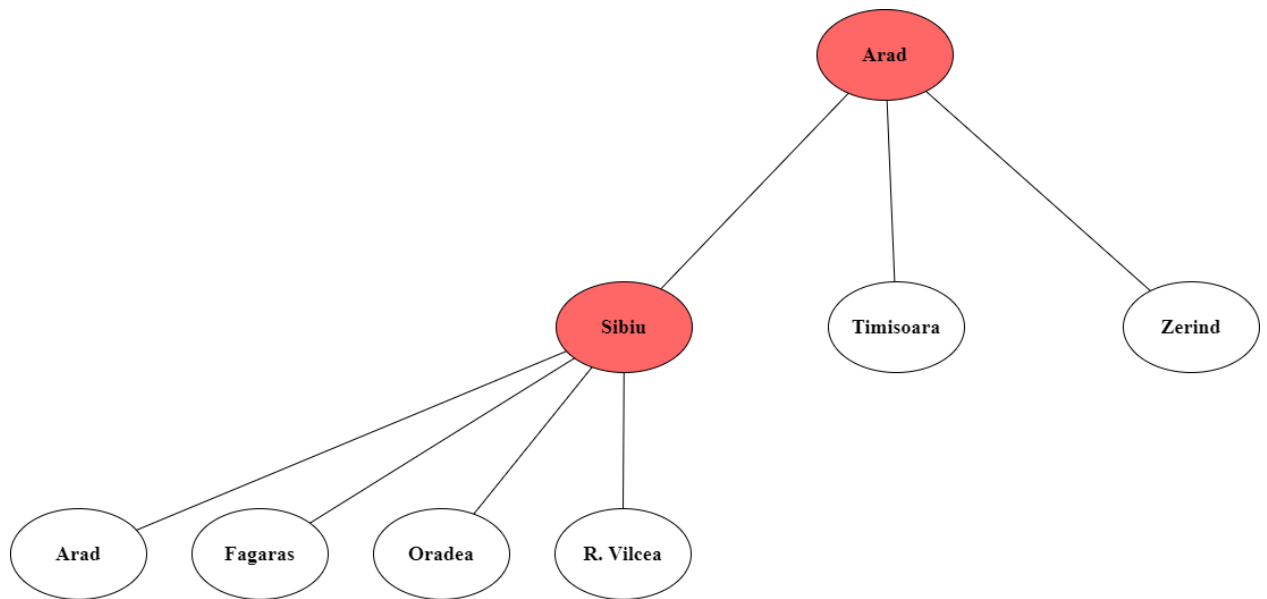
Trong tập OPEN, Sibiu là nút có giá trị f nhỏ nhất nên ta sẽ chọn $T_{\max} = \text{Sibiu}$.

Ta lấy Sibiu ra khỏi OPEN và đưa vào CLOSE.

$$\begin{aligned} \text{OPEN} = \{ & (\text{Timisoara}, g = 118, h = 329, f = 447, \text{Cha} = \text{Arad}), \\ & (\text{Zerind}, g = 75, h = 374, f = 449, \text{Cha} = \text{Arad}) \} \end{aligned}$$

$$\begin{aligned} \text{CLOSE} = \{ & (\text{Arad}, g = 0, h = 0, f = 0), \\ & (\text{Sibiu}, g = 140, h = 253, f = 393, \text{Cha} = \text{Arad}) \} \end{aligned}$$

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



✓ $h(\text{Arad}) = 366$

$$g(\text{Arad}) = g(\text{Sibiu}) + \text{cost}(\text{Sibiu}, \text{Arad}) = 140 + 140 = 280$$

$$f(\text{Arad}) = g(\text{Arad}) + h(\text{Arad}) = 280 + 366 = 646$$

✓ $h(\text{Fagaras}) = 176$

$$g(\text{Fagaras}) = g(\text{Sibiu}) + \text{cost}(\text{Sibiu}, \text{Fagaras}) = 140 + 99 = 239$$

$$f(\text{Fagaras}) = g(\text{Fagaras}) + h(\text{Fagaras}) = 239 + 176 = 415$$

✓ $h(\text{Oradea}) = 380$

$$g(\text{Oradea}) = g(\text{Sibiu}) + \text{cost}(\text{Sibiu}, \text{Oradea}) = 140 + 151 = 291$$

$$f(\text{Oradea}) = g(\text{Oradea}) + h(\text{Oradea}) = 291 + 380 = 671$$

✓ $h(\text{R.Vilcea}) = 193$

$$g(\text{R.Vilcea}) = g(\text{Sibiu}) + \text{cost}(\text{Sibiu}, \text{R.Vilcea}) = 140 + 80 = 220$$

$$f(\text{R.Vilcea}) = g(\text{R.Vilcea}) + h(\text{R.Vilcea}) = 220 + 193 = 413$$

Nút Arad đã có trong CLOSE và $g(\text{Arad})$ mới được tạo ra có giá trị là 280 lớn hơn $g(\text{Arad})$ 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.

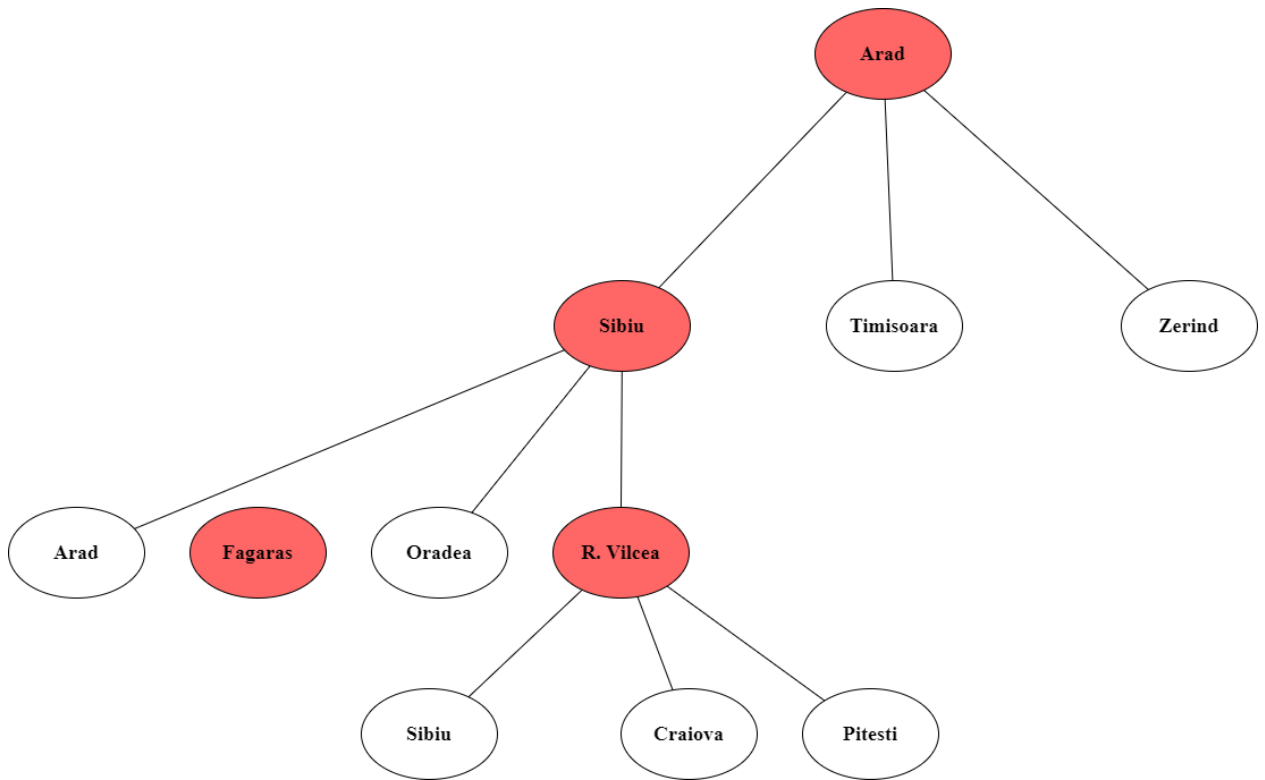
OPEN = { (Timisoara, $g = 118$, $h = 329$, $f = 447$, Cha = Arad),

(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.



✓ $h(\text{Sibiu}) = 253$

$g(\text{Sibiu}) = g(\text{R.Vilcea}) + \text{cost}(\text{R.Vilcea}, \text{Sibiu}) = 220 + 80 = 300$

$f(\text{Sibiu}) = g(\text{Sibiu}) + h(\text{Sibiu}) = 300 + 253 = 553$

✓ $h(\text{Craiova}) = 160$

$g(\text{Craiova}) = g(\text{R.Vilcea}) + \text{cost}(\text{R.Vilcea}, \text{Craiova}) = 220 + 146 = 366$

$f(\text{Craiova}) = g(\text{Craiova}) + h(\text{Craiova}) = 366 + 160 = 526$

✓ $h(\text{Pitesti}) = 100$

$$g(\text{Pitesti}) = g(\text{R.Vilcea}) + \text{cost}(\text{R.Vilcea}, \text{Pitesti}) = 220 + 97 = 317$$

$$f(\text{Pitesti}) = g(\text{Pitesti}) + h(\text{Pitesti}) = 317 + 100 = 417$$

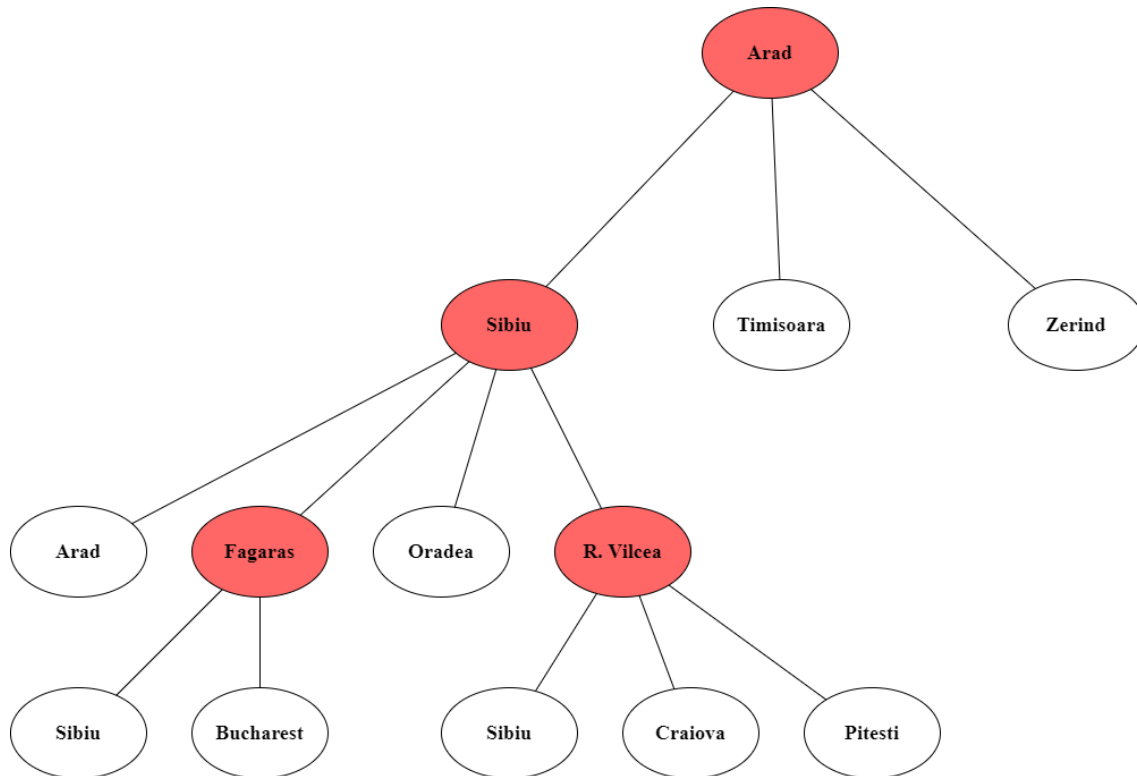
Do Sibiu đã có trong CLOSE và $g(\text{Sibiu})$ mới có giá trị là 553 lớn hơn $g(\text{Sibiu})$ trong CLOSE có giá trị là 393 nên ta không cập nhật lại các giá trị Sibiu được lưu trong CLOSE.

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 $T_{\max} = \text{Fagaras}$. 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.



✓ $h(\text{Sibiu}) = 253$

$$g(\text{Sibiu}) = g(\text{Fagaras}) + \text{cost}(\text{Fagaras}, \text{Sibiu}) = 239 + 99 = 338$$

$$f(\text{Sibiu}) = g(\text{Sibiu}) + h(\text{Sibiu}) = 338 + 253 = 591$$

✓ $h(\text{Bucharest}) = 20$

$$g(\text{Bucharest}) = g(\text{Fagaras}) + \text{cost}(\text{Fagaras}, \text{Bucharest}) = 239 + 211 = 450$$

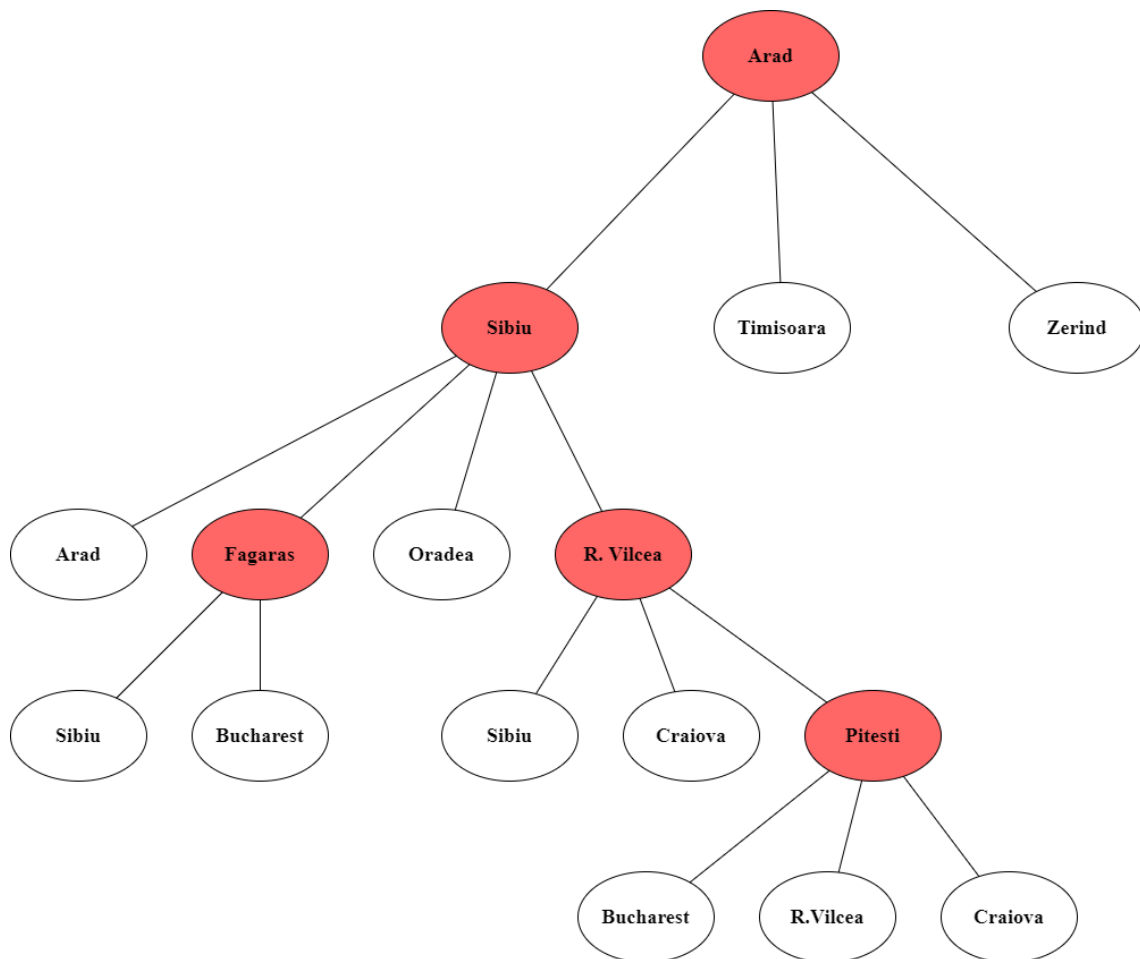
$$f(\text{Bucharest}) = g(\text{Bucharest}) + h(\text{Bucharest}) = 450 + 20 = 470$$

Do Sibiu đã có trong CLOSE và $g(\text{Sibiu})$ mới có giá trị là 338 lớn hơn $g(\text{Sibiu})$ 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 và đặt cha của nó là Fagaras.

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 $T_{\max} = \text{Pitesti}$. 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.



✓ $h(\text{R.Vilcea}) = 193$

$g(\text{R.Vilcea}) = g(\text{Pitesti}) + \text{cost}(\text{Pitesti}, \text{R.Vilcea}) = 317 + 97 = 414$

$f(\text{R.Vilcea}) = g(\text{R.Vilcea}) + h(\text{R.Vilcea}) = 414 + 193 = 607$

✓ $h(\text{Bucharest}) = 20$

$g(\text{Bucharest}) = g(\text{Pitesti}) + \text{cost}(\text{Pitesti}, \text{Bucharest}) = 317 + 101 = 418$

$$f(\text{Bucharest}) = g(\text{Bucharest}) + h(\text{Bucharest}) = 418 + 20 = 438$$

$$\checkmark h(\text{Craiova}) = 160$$

$$g(\text{Craiova}) = g(\text{Pitesti}) + \text{cost}(\text{Pitesti}, \text{Craiova}) = 317 + 138 = 455$$

$$f(\text{Craiova}) = g(\text{Craiova}) + h(\text{Craiova}) = 455 + 160 = 615$$

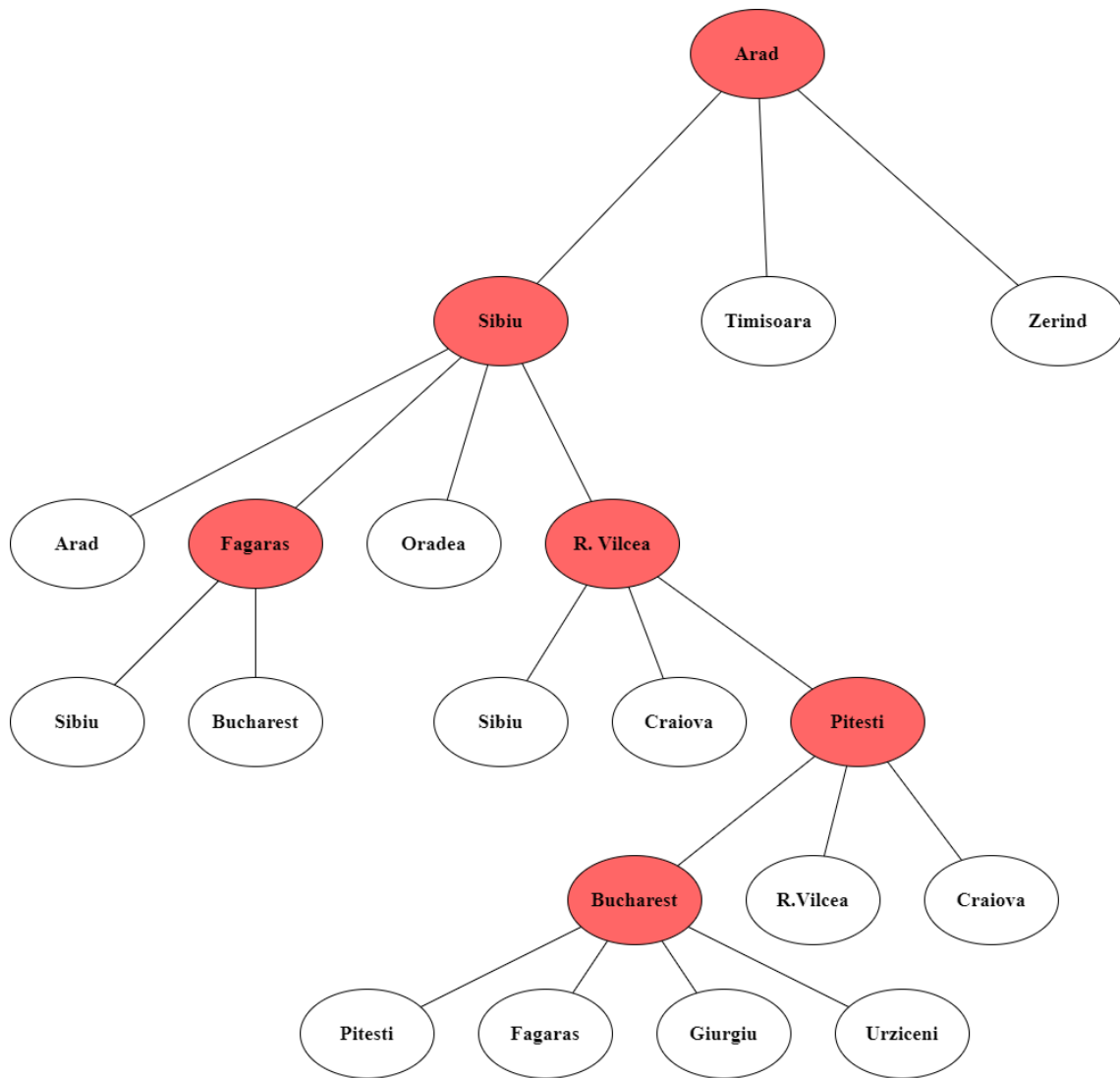
Do R.Vilcea đã có trong CLOSE và $g(\text{R.Vilcea})$ mới được tạo ra có giá trị là 414 lớn hơn $g(\text{R.Vilcea})$ lưu trong CLOSE có giá trị là 220 nên ta sẽ không cập nhật giá trị g và f của R.Vilcea lưu trong CLOSE. Craiova đã có trong OPEN và $g(\text{Craiova})$ mới được tạo có giá trị là 455 lớn hơn $g(\text{Craiova})$ 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 $g(\text{Bucharest})$ mới tạo có giá trị là 418 nhỏ hơn $g(\text{Bucharest})$ 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 $T_{\max} = \text{Bucharest}$. 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.



✓ $h(\text{Pitesti}) = 100$

$$g(\text{Pitesti}) = g(\text{Bucharest}) + \text{cost}(\text{Bucharest}, \text{Pitesti}) = 418 + 101 = 519$$

$$f(\text{Pitesti}) = g(\text{Pitesti}) + h(\text{Pitesti}) = 519 + 100 = 619$$

✓ $h(\text{Fagaras}) = 176$

$$g(\text{Fagaras}) = g(\text{Bucharest}) + \text{cost}(\text{Bucharest}, \text{Fagaras}) = 418 + 211 = 629$$

$$f(\text{Fagaras}) = g(\text{Fagaras}) + h(\text{Fagaras}) = 629 + 176 = 805$$

✓ $h(\text{Giurgiu}) = 77$

$$g(\text{Giurgiu}) = g(\text{Bucharest}) + \text{cost}(\text{Bucharest}, \text{Giurgiu}) = 418 + 90 = 508$$

$$f(\text{Giurgiu}) = g(\text{Giurgiu}) + h(\text{Giurgiu}) = 508 + 77 = 585$$

✓ $h(\text{Urziceni}) = 10$

$$g(\text{Urziceni}) = g(\text{Bucharest}) + \text{cost}(\text{Bucharest}, \text{Urziceni}) = 418 + 85 = 503$$

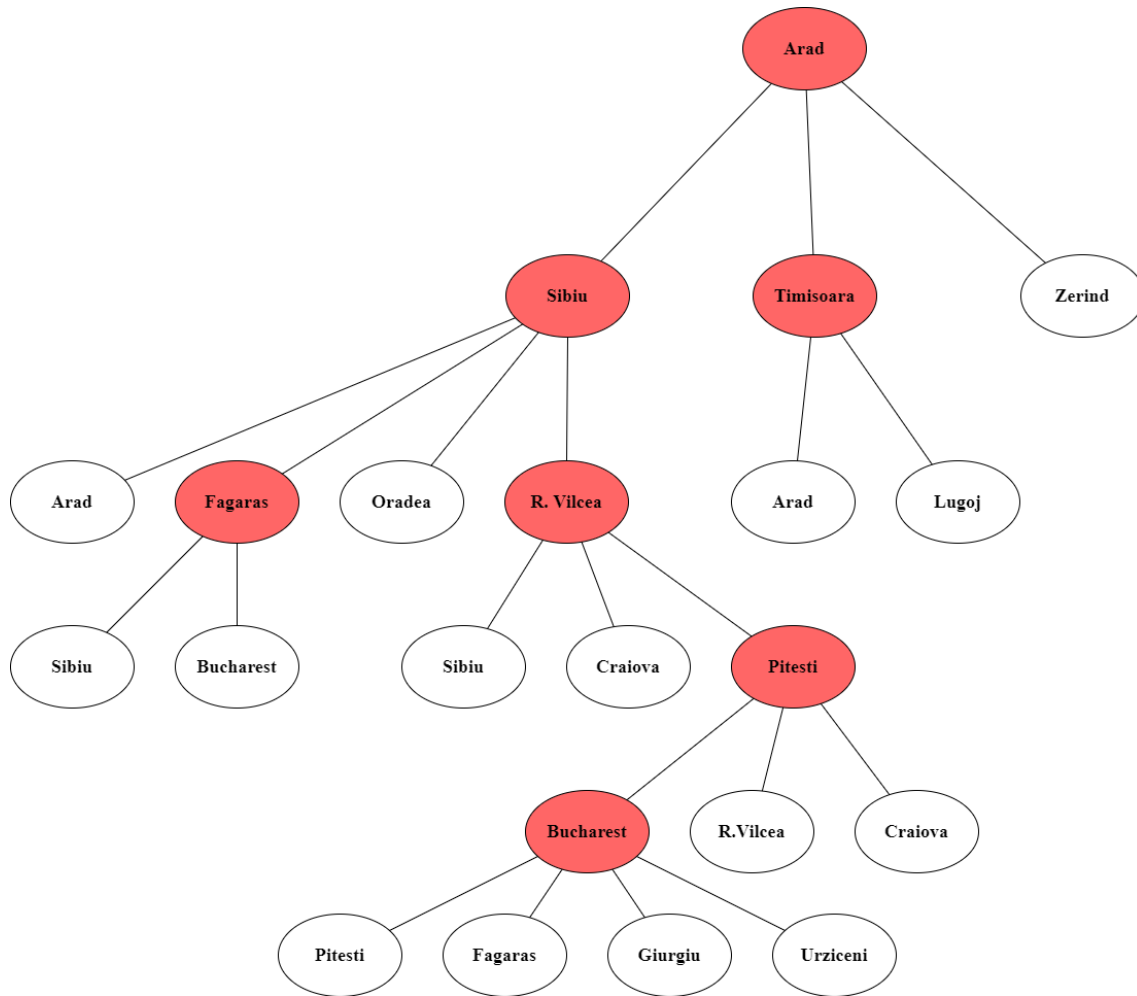
$$f(\text{Urziceni}) = g(\text{Urziceni}) + h(\text{Urziceni}) = 503 + 10 = 513$$

Pitesti và Fagaras đã có trong tập CLOSE và $g(\text{Pitesti})$, $g(\text{Fagaras})$ mới được tạo ra có giá trị lớn hơn $g(\text{Pitesti})$, $g(\text{Fagaras})$ 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) }

Trong tập OPEN, Timisoara là nút có giá trị f nhỏ nhất nên ta chọn $T_{\max} = \text{Timisoara}$. Chuyển Timisoara từ tập OPEN sang tập CLOSE. Từ Timisoara có thể đi được tới 2 thành phố là Arad và Lugoj. Ta lần lượt tính các giá trị h , g và f của 2 thành phố này.



✓ $h(\text{Arad}) = 366$

$$g(\text{Arad}) = g(\text{Timisoara}) + \text{cost}(\text{Timisoara}, \text{Arad}) = 118 + 118 = 236$$

$$f(\text{Arad}) = g(\text{Arad}) + h(\text{Arad}) = 236 + 366 = 602$$

✓ $h(\text{Lugoj}) = 244$

$$g(\text{Lugoj}) = g(\text{Timisoara}) + \text{cost}(\text{Timisoara}, \text{Lugoj}) = 118 + 111 = 229$$

$$f(\text{Lugoj}) = g(\text{Lugoj}) + h(\text{Lugoj}) = 229 + 244 = 473$$

Do Arad đã có trong CLOSE và $g(\text{Arad})$ mới được tạo ra có giá trị là 236 lớn hơn $g(\text{Arad})$ lưu trong CLOSE có giá trị là 0 nên ta sẽ không cập nhật giá trị g và f của Arad lưu trong CLOSE. Lugoj không có trong tập OPEN lẫn CLOSE nên ta sẽ thêm nút này vào tập OPEN và đặt cha của nó là Timisoara.

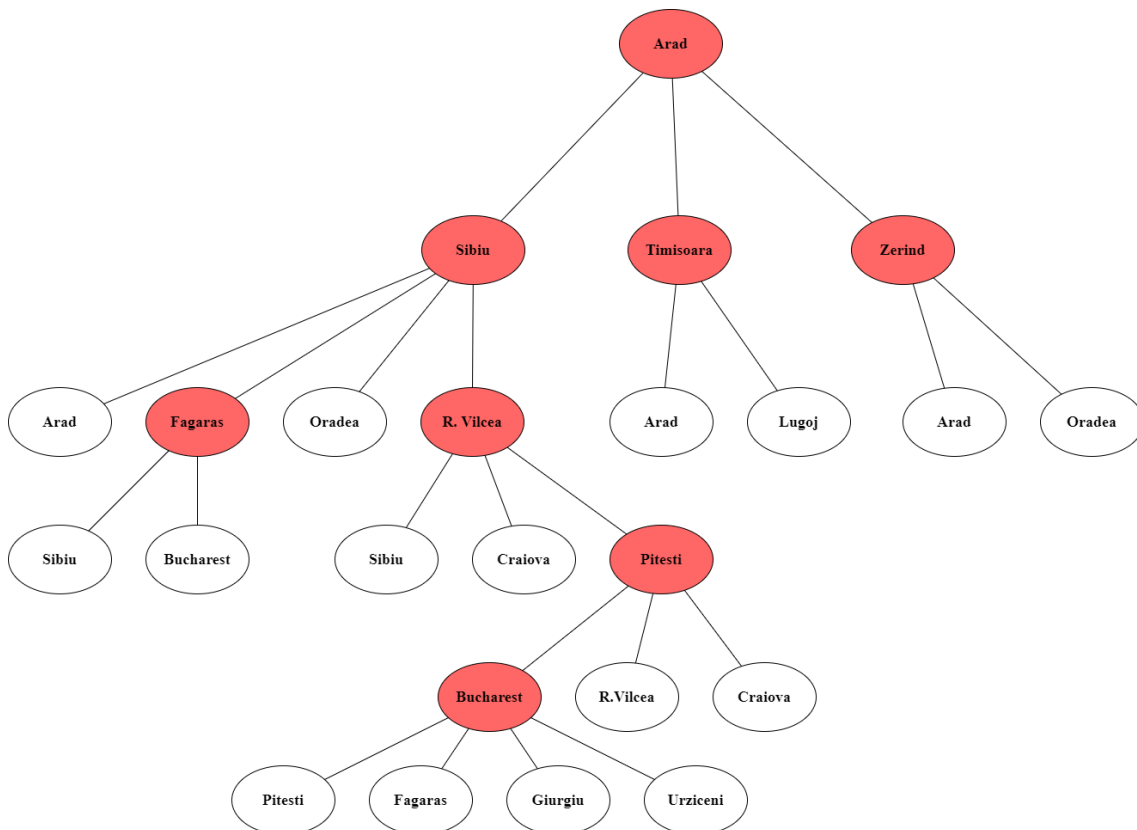
OPEN = { (Zerind, $g = 75$, $h = 374$, $f = 449$, Cha = Arad),

(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),
 (Lugoj, $g = 229$, $h = 244$, $f = 473$, Cha = Timisoara) }

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

Trong tập OPEN, Zerind là nút có giá trị f nhỏ nhất nên ta chọn $T_{\max} =$ Zerind. Chuyển Zerind từ tập OPEN sang tập CLOSE. Từ Zerind có thể đi được tới 2 thành phố là Arad và Oradea.



Ta lần lượt tính các giá trị h , g và f của 2 thành phố này.

$$\checkmark h(\text{Arad}) = 366$$

$$g(\text{Arad}) = g(\text{Zerind}) + \text{cost}(\text{Zerind}, \text{Arad}) = 75 + 75 = 150$$

$$f(\text{Arad}) = g(\text{Arad}) + h(\text{Arad}) = 150 + 366 = 516$$

$$\checkmark h(\text{Oradea}) = 380$$

$$g(\text{Oradea}) = g(\text{Zerind}) + \text{cost}(\text{Zerind}, \text{Oradea}) = 75 + 71 = 146$$

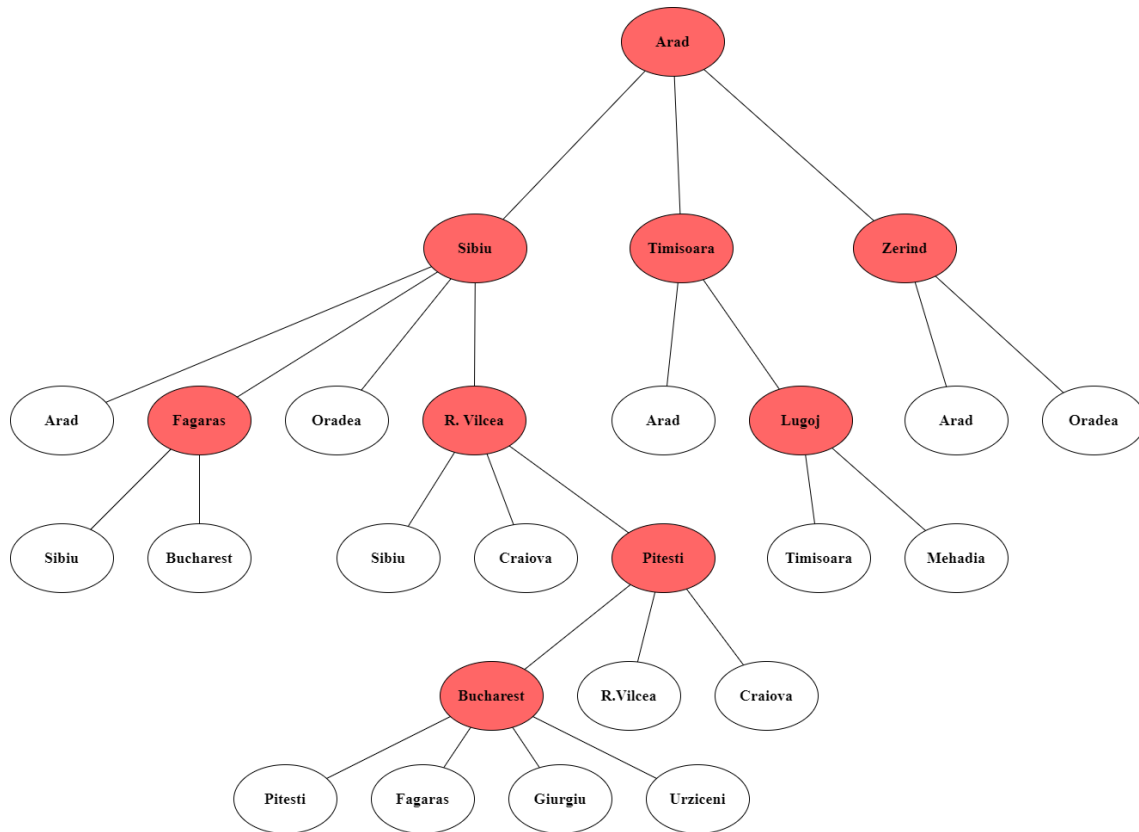
$$f(\text{Oradea}) = g(\text{Oradea}) + h(\text{Oradea}) = 146 + 380 = 526$$

Do Arad đã có trong CLOSE và $g(\text{Arad})$ mới được tạo ra có giá trị là 150 lớn hơn $g(\text{Arad})$ lưu trong CLOSE có giá trị là 0 nên ta sẽ không cập nhật giá trị g và f của Arad lưu trong CLOSE. Oradea đã có trong OPEN và $g(\text{Oradea})$ mới được tạo ra có giá trị là 146 nhỏ hơn $g(\text{Oradea})$ lưu trong OPEN có giá trị là 291 nên ta sẽ cập nhật giá trị g , h , f và cha của Oradea lưu trong OPEN.

OPEN = { (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),
 (Lugoj, $g = 229$, $h = 244$, $f = 473$, Cha = Timisoara),
 (Oradea, $g = 146$, $h = 380$, $f = 526$, Cha = Zerind) }

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),
 (Timisoara, $g = 118$, $h = 329$, $f = 447$, Cha = Arad),
 (Zerind, $g = 75$, $h = 374$, $f = 449$, Cha = Arad) }

Trong tập OPEN, Lugoj là nút có giá trị f nhỏ nhất nên ta chọn $T_{\max} = \text{Lugoj}$. Chuyển Lugoj từ tập OPEN sang tập CLOSE. Từ Lugoj có thể đi được tới 2 thành phố là Timisoara và Mehadia.



Ta lần lượt tính các giá trị h, g và f của 2 thành phố này.

✓ $h(\text{Timisoara}) = 329$

$$g(\text{Timisoara}) = g(\text{Lugoj}) + \text{cost}(\text{Lugoj}, \text{Timisoara}) = 229 + 111 = 340$$

$$f(\text{Timisoara}) = g(\text{Timisoara}) + h(\text{Timisoara}) = 340 + 329 = 669$$

✓ $h(\text{Mehadia}) = 241$

$$g(\text{Mehadia}) = g(\text{Lugoj}) + \text{cost}(\text{Lugoj}, \text{Mehadia}) = 229 + 70 = 299$$

$$f(\text{Mehadia}) = g(\text{Mehadia}) + h(\text{Mehadia}) = 299 + 241 = 540$$

Do Timisoara đã có trong CLOSE và $g(\text{Timisoara})$ mới được tạo ra có giá trị là 229 lớn hơn $g(\text{Timisoara})$ lưu trong CLOSE có giá trị là 118 nên ta sẽ không cập nhật giá trị g và f của Timisoara lưu trong CLOSE. Mehadia không có trong tập OPEN lẫn CLOSE nên ta sẽ thêm nút này vào tập OPEN và đặt cha của nó là Lugoj.

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

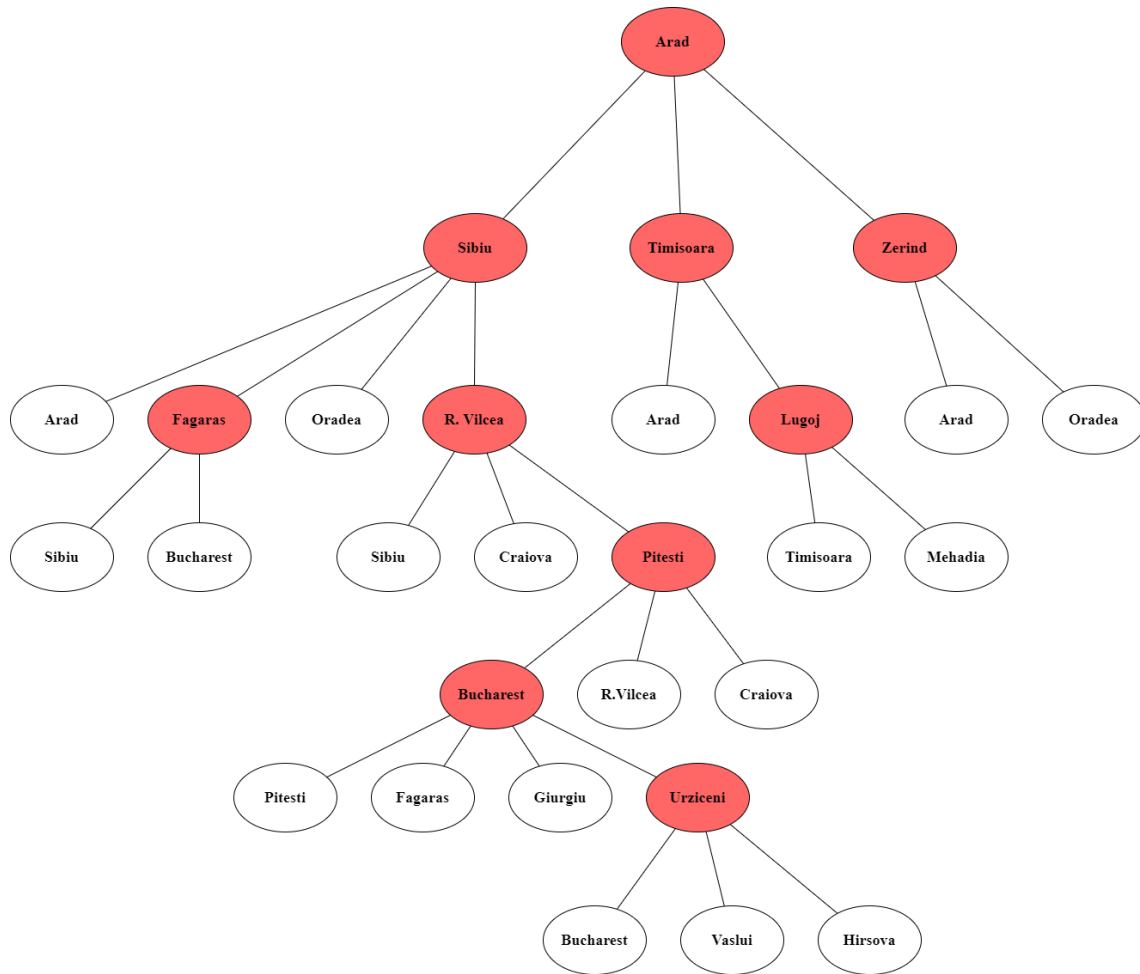
(Giurgiu, g = 508, h = 77, f = 585, Cha = Bucharest),

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

(Oradea, $g = 146$, $h = 380$, $f = 526$, Cha = Zerind),
(Mehadia, $g = 299$, $h = 241$, $f = 540$, Cha = Lugoj) }

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),
(Timisoara, $g = 118$, $h = 329$, $f = 447$, Cha = Arad),
(Zerind, $g = 75$, $h = 374$, $f = 449$, Cha = Arad),
(Lugoj, $g = 229$, $h = 244$, $f = 473$, Cha = Timisoara) }

Trong tập OPEN, Urziceni là nút có giá trị f nhỏ nhất nên ta chọn $T_{\max} =$ Urziceni. Chuyển Urziceni từ tập OPEN sang tập CLOSE. Từ Urziceni có thể đi được tới 3 thành phố là Bucharest, Vaslui và Hirsova.



Ta lần lượt tính các giá trị h , g và f của 3 thành phố này.

✓ $h(\text{Bucharest}) = 20$

$$g(\text{Bucharest}) = g() + \text{cost}(\text{Urziceni}, \text{Bucharest}) = 503 + 85 = 588$$

$$f(\text{Bucharest}) = g(\text{Bucharest}) + h(\text{Bucharest}) = 588 + 20 = 608$$

✓ $h(\text{Vaslui}) = 199$

$$g(\text{Vaslui}) = g(\text{Urziceni}) + \text{cost}(\text{Urziceni}, \text{Vaslui}) = 503 + 142 = 645$$

$$f(\text{Vaslui}) = g(\text{Vaslui}) + h(\text{Vaslui}) = 645 + 199 = 844$$

✓ $h(\text{Hirsova}) = 0$

$$g(\text{Hirsova}) = g(\text{Urziceni}) + \text{cost}(\text{Urziceni}, \text{Hirsova}) = 503 + 98 = 601$$

$$f(\text{Hirsova}) = g(\text{Hirsova}) + h(\text{Hirsova}) = 601 + 0 = 601$$

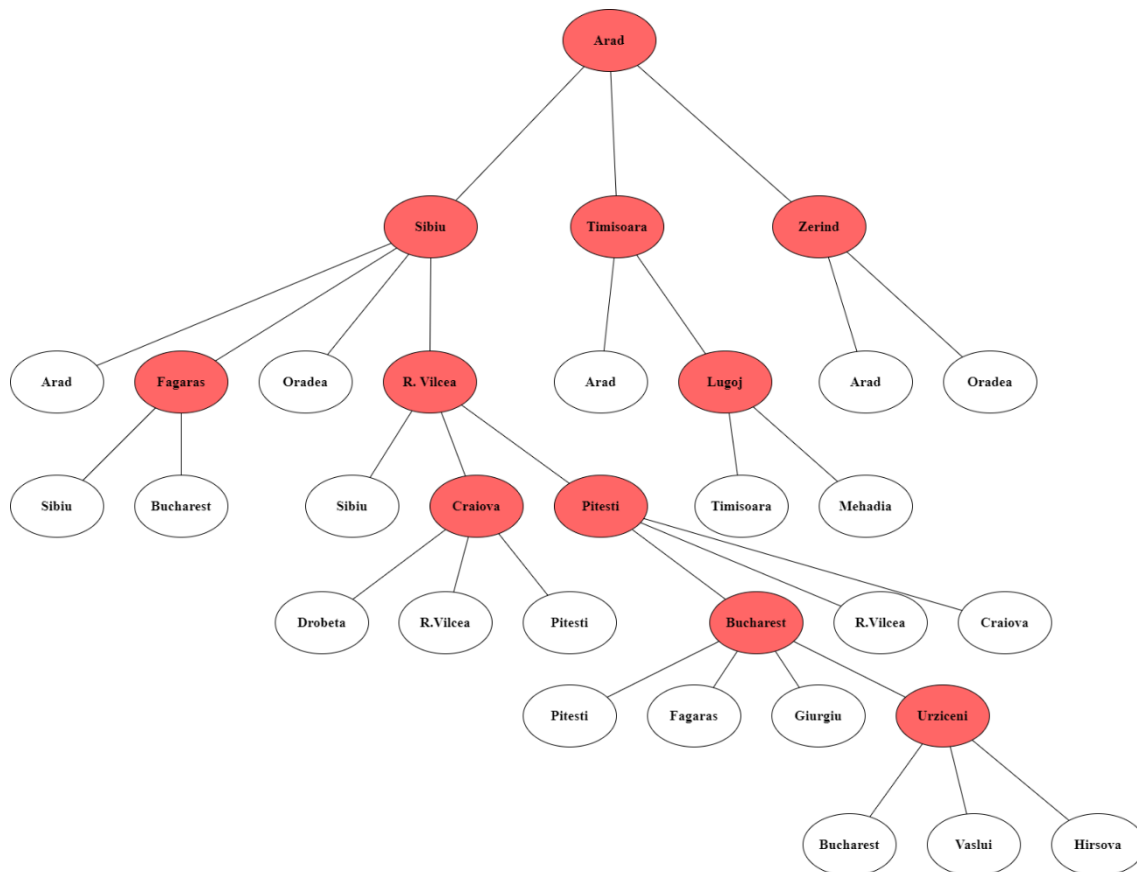
Do Bucharest đã có trong CLOSE và $g(\text{Bucharest})$ mới được tạo ra có giá trị là 588 lớn hơn $g(\text{Bucharest})$ lưu trong CLOSE có giá trị là 418 nên ta sẽ

không cập nhật giá trị g và f của Bucharest lưu trong CLOSE. Vaslui và Hirsova không có trong tập OPEN lẫn CLOSE nên ta sẽ thêm nút này vào tập OPEN và đặt cha của nó là Urziceni.

OPEN = { (Craiova, g = 366, h = 160, f = 526, Cha = R.Vilcea),
(Giurgiu, g = 508, h = 77, f = 585, Cha = Bucharest),
(Oradea, g = 146, h = 380, f = 526, Cha = Zerind),
(Mehadia, g = 299, h = 241, f = 540, Cha = Lugoj),
(Vaslui, g = 645, h = 199, f = 844, Cha = Urziceni),
(Hirsova, g = 601, h = 0, f = 601, Cha = Urziceni) }

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),
(Timisoara, g = 118, h = 329, f = 447, Cha = Arad),
(Zerind, g = 75, h = 374, f = 449, Cha = Arad),
(Lugoj, g = 229, h = 244, f = 473, Cha = Timisoara),
(Urziceni, g = 503, h = 10, f = 513, Cha = Bucharest) }

Trong tập OPEN, Craiova là nút có giá trị f nhỏ nhất nên ta chọn $T_{\max} =$ Craiova. Chuyển Craiova từ tập OPEN sang tập CLOSE. Từ Craiova có thể đi được tới 3 thành phố là Drobeta, R.Vilcea và Pitesti.



Ta lần lượt tính các giá trị h , g và f của 3 thành phố này.

✓ $h(\text{Drobeta}) = 242$

$$g(\text{Drobeta}) = g(\text{Craiova}) + \text{cost}(\text{Craiova}, \text{Drobeta}) = 366 + 120 = 486$$

$$f(\text{Drobeta}) = g(\text{Drobeta}) + h(\text{Drobeta}) = 486 + 242 = 728$$

✓ $h(\text{R.Vilcea}) = 193$

$$g(\text{R.Vilcea}) = g(\text{Craiova}) + \text{cost}(\text{Craiova}, \text{R.Vilcea}) = 366 + 146 = 512$$

$$f(\text{R.Vilcea}) = g(\text{R.Vilcea}) + h(\text{R.Vilcea}) = 512 + 193 = 705$$

✓ $h(\text{Pitesti}) = 100$

$$g(\text{Pitesti}) = g(\text{Craiova}) + \text{cost}(\text{Craiova}, \text{Pitesti}) = 366 + 138 = 504$$

$$f(\text{Pitesti}) = g(\text{Pitesti}) + h(\text{Pitesti}) = 504 + 100 = 604$$

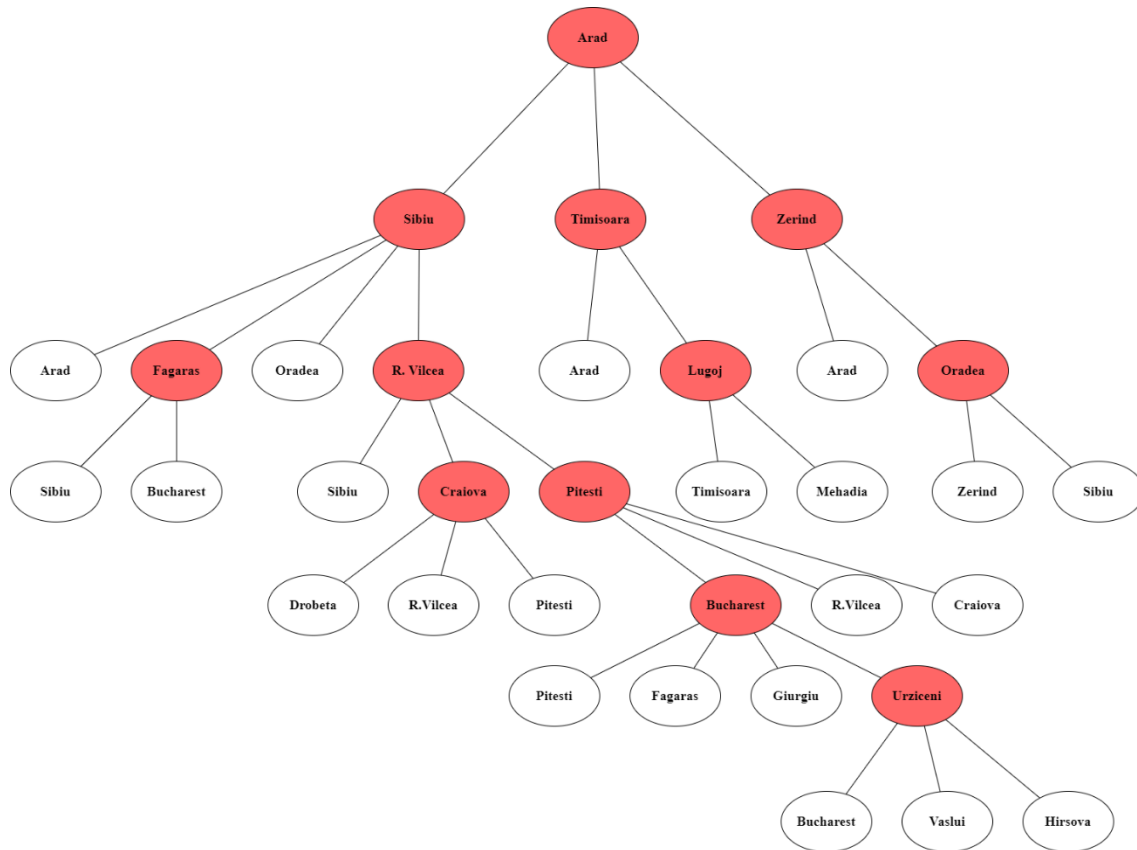
Do Drobeta không có trong tập OPEN lẫn CLOSE nên ta sẽ thêm nút này vào tập OPEN và đặt cha của nó là Craiova. Do R.Vilcea đã có trong CLOSE và $g(\text{R.Vilcea})$ mới được tạo ra có giá trị là 512 lớn hơn $g(\text{R.Vilcea})$ lưu trong CLOSE có giá trị là 220 nên ta sẽ không cập nhật giá trị g và f của R.Vilcea lưu trong CLOSE. Pitesti đã có trong CLOSE và

$g(\text{Pitesti})$ mới được tạo ra có giá trị là 504 lớn hơn $g(\text{Pitesti})$ lưu trong CLOSE có giá trị là 317 nên ta sẽ không cập nhật giá trị g và f của Pitesti lưu trong CLOSE.

OPEN = { (Giurgiu, $g = 508$, $h = 77$, $f = 585$, Cha = Bucharest),
(Oradea, $g = 146$, $h = 380$, $f = 526$, Cha = Zerind),
(Mehadia, $g = 299$, $h = 241$, $f = 540$, Cha = Lugoj),
(Vaslui, $g = 645$, $h = 199$, $f = 844$, Cha = Urziceni),
(Hirsova, $g = 601$, $h = 0$, $f = 601$, Cha = Urziceni),
(Drobeta, $g = 486$, $h = 242$, $f = 728$, Cha = Craiova) }

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),
(Timisoara, $g = 118$, $h = 329$, $f = 447$, Cha = Arad),
(Zerind, $g = 75$, $h = 374$, $f = 449$, Cha = Arad),
(Lugoj, $g = 229$, $h = 244$, $f = 473$, Cha = Timisoara),
(Urziceni, $g = 503$, $h = 10$, $f = 513$, Cha = Bucharest),
(Craiova, $g = 366$, $h = 160$, $f = 526$, Cha = R.Vilcea) }

Trong tập OPEN, Oradea là nút có giá trị f nhỏ nhất nên ta chọn $T_{\max} = \text{Oradea}$. Chuyển Oradea từ tập OPEN sang tập CLOSE. Từ Oradea có thể đi được tới 2 thành phố là Zerind và Sibiu.



Ta lần lượt tính các giá trị h , g và f của 2 thành phố này.

✓ $h(\text{Zerind}) = 374$

$$g(\text{Zerind}) = g(\text{Oradea}) + \text{cost}(\text{Oradea}, \text{Zerind}) = 146 + 71 = 217$$

$$f(\text{Zerind}) = g(\text{Zerind}) + h(\text{Zerind}) = 217 + 374 = 591$$

✓ $h(\text{Sibiu}) = 253$

$$g(\text{Sibiu}) = g(\text{Oradea}) + \text{cost}(\text{Oradea}, \text{Sibiu}) = 146 + 151 = 297$$

$$f(\text{Sibiu}) = g(\text{Sibiu}) + h(\text{Sibiu}) = 297 + 253 = 550$$

Do Zerind đã có trong CLOSE và $g(\text{Zerind})$ mới được tạo ra có giá trị là 217 lớn hơn $g(\text{Zerind})$ lưu trong CLOSE có giá trị là 75 nên ta sẽ không cập nhật giá trị g và f của Zerind lưu trong CLOSE. Sibiu đã có trong CLOSE và $g(\text{Sibiu})$ mới được tạo ra có giá trị là 297 lớn hơn $g(\text{Sibiu})$ lưu trong CLOSE có giá trị là 140 nên ta sẽ không cập nhật giá trị g và f của Sibiu lưu trong CLOSE.

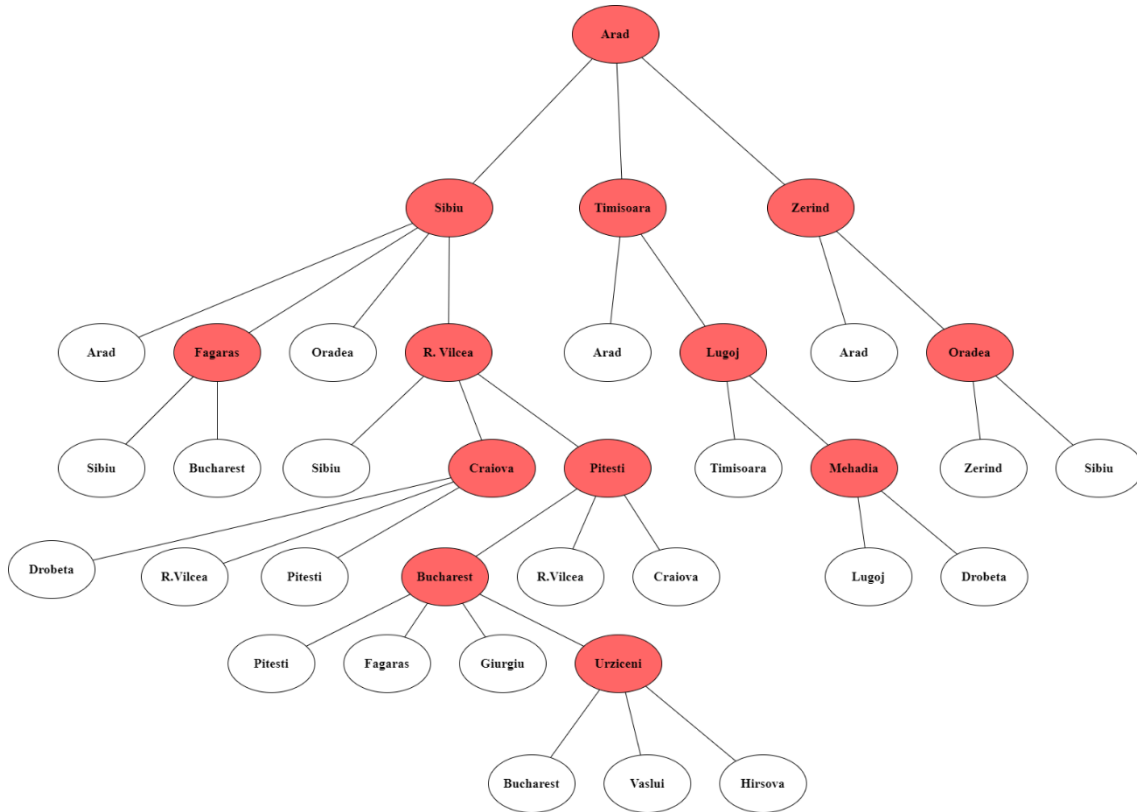
OPEN = { (Giurgiu, $g = 508$, $h = 77$, $f = 585$, Cha = Bucharest),

(Mehadia, $g = 299$, $h = 241$, $f = 540$, Cha = Lugoj),

(Vaslui, $g = 645$, $h = 199$, $f = 844$, Cha = Urziceni),
 (Hirsova, $g = 601$, $h = 0$, $f = 601$, Cha = Urziceni),
 (Drobeta, $g = 486$, $h = 242$, $f = 728$, Cha = Craiova) }

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),
 (Timisoara, $g = 118$, $h = 329$, $f = 447$, Cha = Arad),
 (Zerind, $g = 75$, $h = 374$, $f = 449$, Cha = Arad),
 (Lugoj, $g = 229$, $h = 244$, $f = 473$, Cha = Timisoara),
 (Urziceni, $g = 503$, $h = 10$, $f = 513$, Cha = Bucharest),
 (Craiova, $g = 366$, $h = 160$, $f = 526$, Cha = R.Vilcea),
 (Oradea, $g = 146$, $h = 380$, $f = 526$, Cha = Zerind) }

Trong tập OPEN, Mehadia là nút có giá trị f nhỏ nhất nên ta chọn $T_{\max} =$ Mehadia. Chuyển Mehadia từ tập OPEN sang tập CLOSE. Từ Mehadia có thể đi được tới 2 thành phố là Lugoj và Drobeta.



Ta lần lượt tính các giá trị h , g và f của 2 thành phố này.

✓ $h(\text{Lugoj}) = 244$

$$g(\text{Lugoj}) = g(\text{Mehadia}) + \text{cost}(\text{Mehadia}, \text{Lugoj}) = 299 + 70 = 369$$

$$f(\text{Lugoj}) = g(\text{Lugoj}) + h(\text{Lugoj}) = 369 + 244 = 613$$

✓ $h(\text{Drobeta}) = 242$

$$g(\text{Drobeta}) = g(\text{Mehadia}) + \text{cost}(\text{Mehadia}, \text{Drobeta}) = 299 + 75 = 374$$

$$f(\text{Drobeta}) = g(\text{Drobeta}) + h(\text{Drobeta}) = 374 + 242 = 616$$

Do Lugoj đã có trong CLOSE và $g(\text{Lugoj})$ mới được tạo ra có giá trị là 369 lớn hơn $g(\text{Lugoj})$ lưu trong CLOSE có giá trị là 229 nên ta sẽ không cập nhật giá trị g và f của Lugoj lưu trong CLOSE. Drobeta đã có trong OPEN và $g(\text{Drobeta})$ mới được tạo ra có giá trị là 374 nhỏ hơn $g(\text{Drobeta})$ lưu trong OPEN có giá trị là 486 nên ta sẽ cập nhật giá trị g , h , f và cha của Drobeta lưu trong OPEN.

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

(Vaslui, $g = 645$, $h = 199$, $f = 844$, Cha = Urziceni),

(Hirsova, $g = 601$, $h = 0$, $f = 601$, Cha = Urziceni),

(Drobeta, $g = 374$, $h = 242$, $f = 616$, Cha = Mehadia) }

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),

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

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

(Lugoj, $g = 229$, $h = 244$, $f = 473$, Cha = Timisoara),

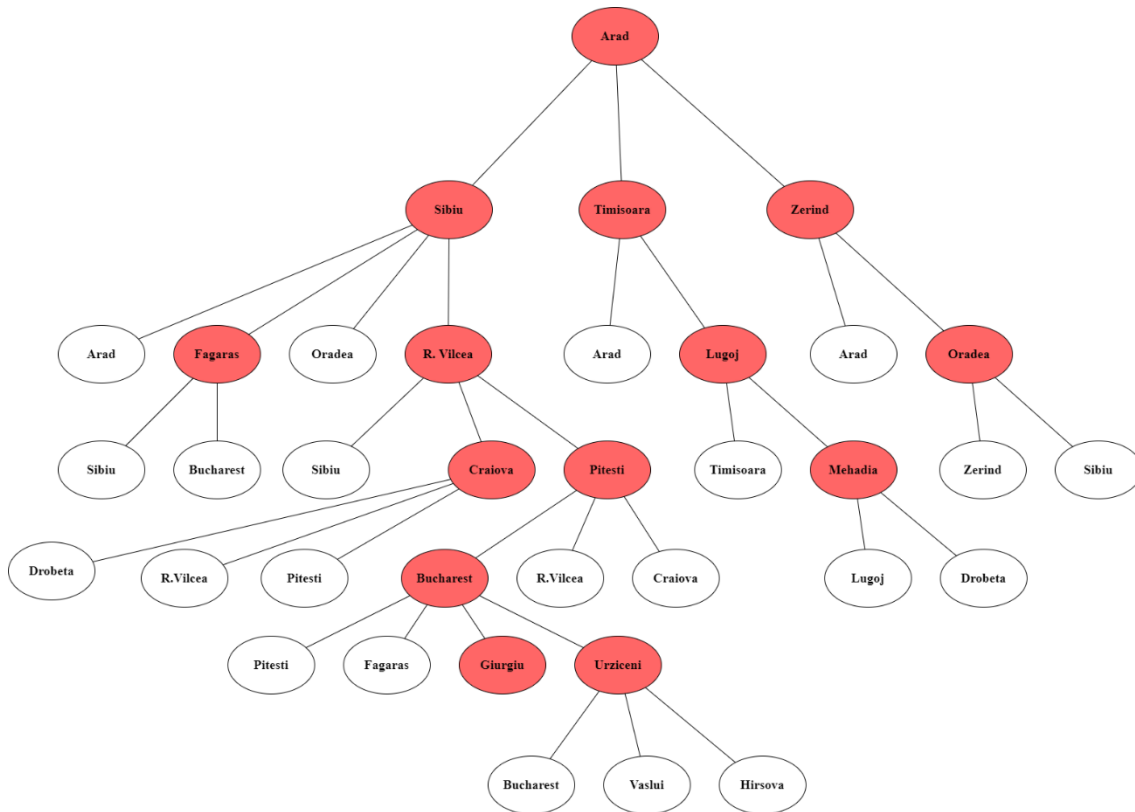
(Urziceni, $g = 503$, $h = 10$, $f = 513$, Cha = Bucharest),

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

(Oradea, $g = 146$, $h = 380$, $f = 526$, Cha = Zerind),

(Mehadia, $g = 299$, $h = 241$, $f = 540$, Cha = Lugoj) }

Trong tập OPEN, Giurgiu là nút có giá trị f nhỏ nhất nên ta chọn $T_{\max} =$ Giurgiu. Chuyển Giurgiu từ tập OPEN sang tập CLOSE. Từ Giurgiu có thể đi được tới duy nhất thành phố là Bucharest.



Ta lần lượt tính các giá trị h , g và f của thành phố này.

✓ $h(\text{Bucharest}) = 20$

$$g(\text{Bucharest}) = g(\text{Giurgiu}) + \text{cost}(\text{Giurgiu}, \text{Bucharest}) = 508 + 90 = 598$$

$$f(\text{Bucharest}) = g(\text{Bucharest}) + h(\text{Bucharest}) = 598 + 20 = 618$$

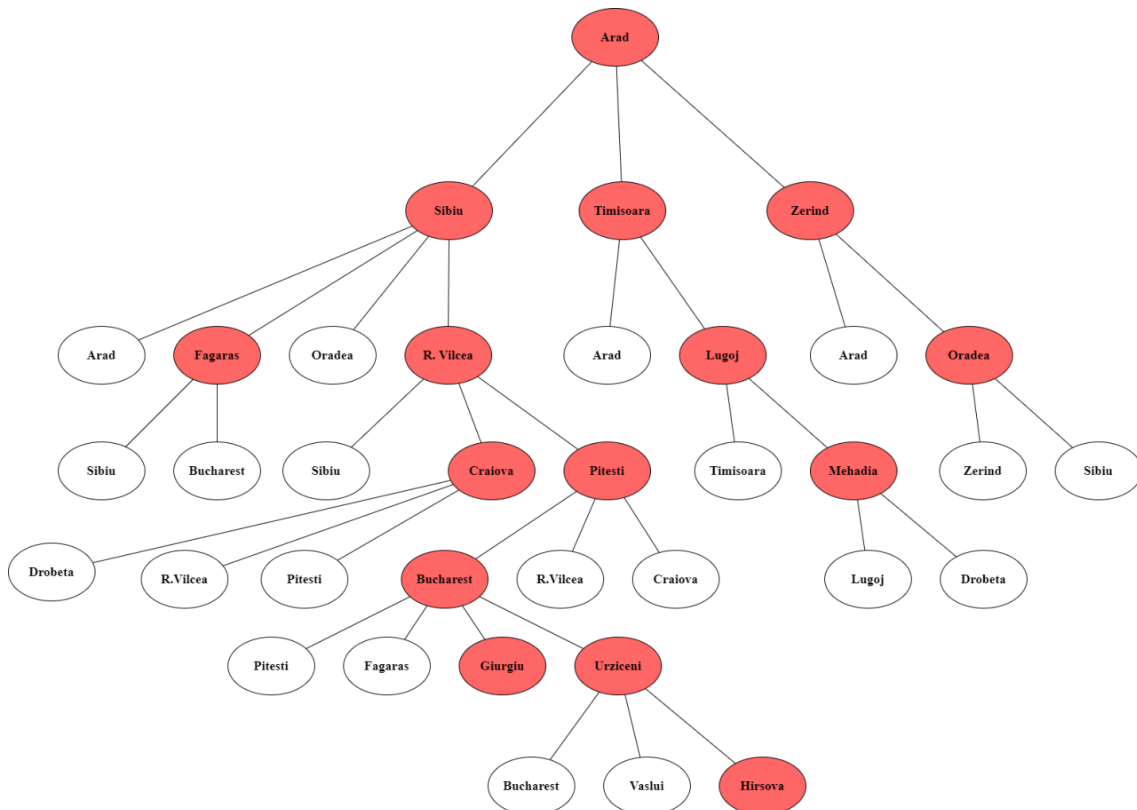
Do Bucharest đã có trong CLOSE và $g(\text{Bucharest})$ mới được tạo ra có giá trị là 598 lớn hơn $g(\text{Bucharest})$ lưu trong CLOSE có giá trị là 418 nên ta sẽ không cập nhật giá trị g và f của Bucharest lưu trong CLOSE.

OPEN = { (Vaslui, $g = 645$, $h = 199$, $f = 844$, Cha = Urziceni),
 (Hirsova, $g = 601$, $h = 0$, $f = 601$, Cha = Urziceni),
 (Drobeta, $g = 374$, $h = 242$, $f = 616$, Cha = Mehadia) }

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),
 (Timisoara, $g = 118$, $h = 329$, $f = 447$, Cha = Arad),
 (Zerind, $g = 75$, $h = 374$, $f = 449$, Cha = Arad),
 (Lugoj, $g = 229$, $h = 244$, $f = 473$, Cha = Timisoara),
 (Urziceni, $g = 503$, $h = 10$, $f = 513$, Cha = Bucharest),
 (Craiova, $g = 366$, $h = 160$, $f = 526$, Cha = R.Vilcea),
 (Oradea, $g = 146$, $h = 380$, $f = 526$, Cha = Zerind),
 (Mehadia, $g = 299$, $h = 241$, $f = 540$, Cha = Lugoj),
 (Giurgiu, $g = 508$, $h = 77$, $f = 585$, Cha = Bucharest) }

Trong tập OPEN, Hirsova là nút có giá trị f nhỏ nhất nên ta chọn $T_{\max} =$ Hirsova. Chuyển Hirsova từ tập OPEN sang tập CLOSE. Hirsova chính là trạng thái đích của bài toán, lúc này thoát vòng lặp và thông báo lời giải là T_{\max} .



OPEN = { (Vaslui, $g = 645$, $h = 199$, $f = 844$, Cha = Urziceni),
 (Drobeta, $g = 374$, $h = 242$, $f = 616$, Cha = Mehadia) }

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),
 (Timisoara, g = 118, h = 329, f = 447, Cha = Arad),
 (Zerind, g = 75, h = 374, f = 449, Cha = Arad),
 (Lugoj, g = 229, h = 244, f = 473, Cha = Timisoara),
 (Urziceni, g = 503, h = 10, f = 513, Cha = Bucharest),
 (Craiova, g = 366, h = 160, f = 526, Cha = R.Vilcea),
 (Oradea, g = 146, h = 380, f = 526, Cha = Zerind),
 (Mehadia, g = 299, h = 241, f = 540, Cha = Lugoj),
 (Giurgiu, g = 508, h = 77, f = 585, Cha = Bucharest),
 (Hirsova, g = 601, h = 0, f = 601, Cha = Urziceni) }

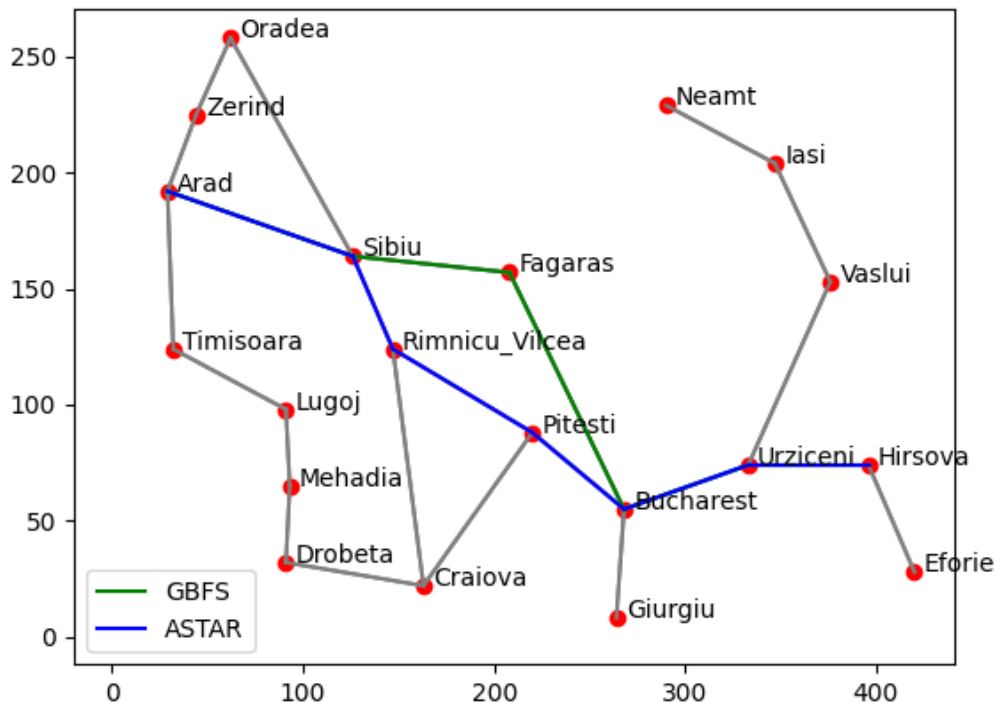
Kết quả:

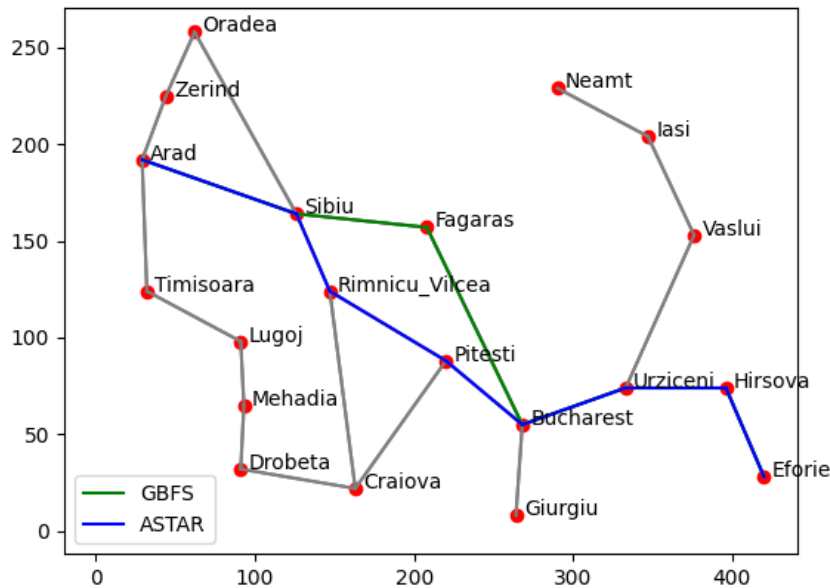
ASTAR => ['Arad', 'Sibiu', 'Rimnicu_Vilcea', 'Pitesti', 'Bucharest',
 'Urziceni', 'Hirsova']

2. Chạy đoạn mã.

```
Run: main
C:\Users\PC-LENOVO\AppData\Local\Programs\Python\Python310\python.exe C:\Users\PC-LENOVO\Downloads\Pycharm_code\main.py
1 Arad
2 Bucharest
3 Craiova
4 Drobeta
5 Eforie
6 Fagaras
7 Giurgiu
8 Hirsova
9 Iasi
10 Lugoj
11 Mehadia
12 Neamt
13 Oradea
14 Pitesti
15 Rimnicu_Vilcea
16 Sibiu
17 Timisoara
18 Urziceni
19 Vaslui
20 Zerind
Nhập đỉnh bắt đầu: 1
Nhập đỉnh kết thúc: 2
GBFS => ['Arad', 'Sibiu', 'Fagaras', 'Bucharest', 'Urziceni', 'Hirsova']
ASTAR => ['Arad', 'Sibiu', 'Rimnicu_Vilcea', 'Pitesti', 'Bucharest', 'Urziceni', 'Hirsova']
```

```
Nhập đỉnh bắt đầu: 1
Nhập đỉnh kết thúc: 20
GBFS => ['Arad', 'Sibiu', 'Fagaras', 'Bucharest', 'Urziceni', 'Hirsova', 'Eforie']
ASTAR => ['Arad', 'Sibiu', 'Rimnicu_Vilcea', 'Pitesti', 'Bucharest', 'Urziceni', 'Hirsova', 'Eforie']
Nhập đỉnh bắt đầu: 1
Nhập đỉnh kết thúc: 0
Process finished with exit code 0
```





Kết quả:

GBFS => ['Arad', 'Sibiu', 'Fagaras', 'Bucharest', 'Urziceni', 'Hirsova']

ASTAR => ['Arad', 'Sibiu', 'Rimnicu_Vilcea', 'Pitesti', 'Bucharest', 'Urziceni', 'Hirsova']

III. NHẬN XÉT.

Các kết quả từ chạy tay và code đều cho ra kết quả giống nhau. Ở kết quả chạy code, khi ta chọn điểm đầu và điểm cuối không thích hợp, kết quả vẫn sẽ được trả về nhưng sẽ không như mong muốn, có thể tìm hiểu và phát triển giải quyết vấn đề này.