

BÀI 3: CÁC THUẬT TOÁN TÌM KIẾM GREEDY-BEST-FIRST SEARCH VÀ A^*

I. MỤC TIÊU:

Sau khi thực hành xong, sinh viên nắm được:

- Thuật toán tìm kiếm Greedy-Best-First Search và A^* .
- Cài đặt được các thuật toán này trên máy tính.

II. TÓM TẮT LÝ THUYẾT:

1. Thuật toán Greedy-Best-First Search:

là một chiến lược tìm kiếm với tri thức bổ sung từ việc sử dụng các tri thức cụ thể của bài toán. Thuật toán sẽ sử dụng 1 hàm đánh giá là hàm heuristic $h(n)$ (hàm heuristic $h(n)$ đánh giá chi phí để đi từ nút hiện tại n đến nút đích (mục tiêu)).

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

$$g(T_i) = g(T_{father}) + cost(T_{father}, T_i) \text{ là thấp nhất}$$

- ✓ *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 $g(T_k) = g(T_{\max}) + \text{cost}(T_{\max}, T_k)$
 - 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 $\text{Cha}(T_{k'}) = T_{\max}$
 - 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 $\text{Cha}(T_{k'}) = T_{\max}$
 - 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 $\text{cost}(T_i, T_{i+1})$ 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.

$$✓ h(Sibiu) = 253$$

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

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

$$Cha(Sibiu) = Arad$$

$$✓ h(Timisoara) = 329$$

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

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

$$Cha(Timisoara) = Arad$$

$$✓ h(Zerind) = 374 \quad g(Zerind) = g(Arad) + cost(Arad, Zerind) = 0 + 75 = 75$$

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

$$Cha(Zerind) = Arad$$

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 $T_{max} = Sibiu$.

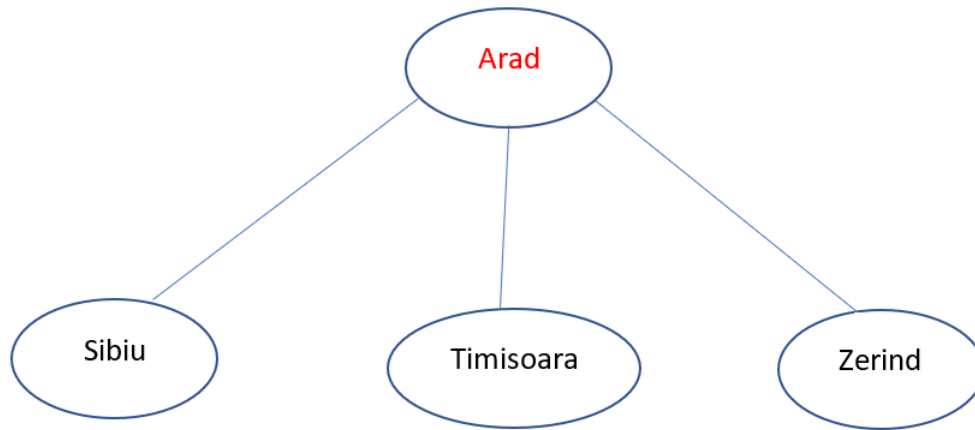
Ta lấy Sibiu ra khỏi OPEN và đưa vào CLOSE

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

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

$$CLOSE = \{(Arad, g = 0, h = 0, f = 0), (Sibiu, g = 140, h = 253, f = 393, Cha = Arad)\}$$

Từ Sibiu có thể đi được đến Arad, Fagaras, Oradea, R. Vilcea. Ta lần lượt tính h, g



và f của các nút này.

✓ $h(\text{Arad}) = 366$

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

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

✓ $h(\text{Fagaras}) = 176$

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

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

✓ $h(\text{Oradea}) = 380$

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

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

✓ $h(\text{R.Vilcea}) = 193$

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

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

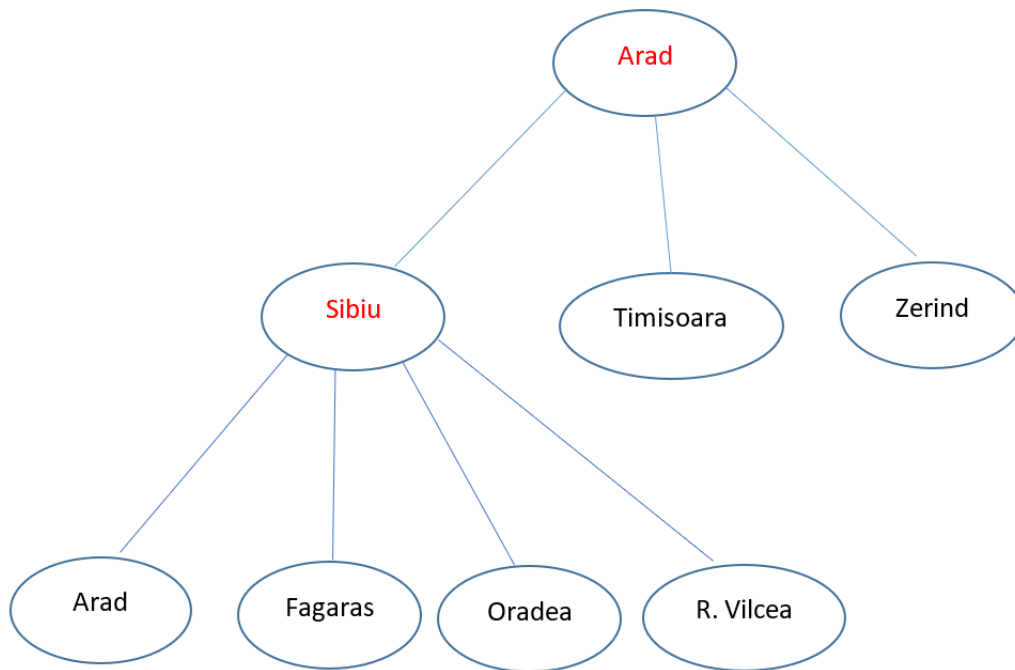
Nút Arad đã có trong CLOSE và $g(\text{Arad})$ mới được tạo ra có giá trị là 280 lớn hơn $g(\text{Arad})$ lưu trong CLOSE có giá trị là 0 nên ta sẽ không cập nhật giá trị g và f của Arad lưu trong CLOSE. 3 nút Fagaras, Oradea và R. Vilcea đều không có trong OPEN và CLOSE nên ta sẽ thêm 3 nút này vào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

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

$$(\text{Zerind}, g = 75, h = 374, f = 449, \text{Cha} = \text{Arad}),$$

$$(\text{Fagaras}, g = 239, h = 176, f = 415, \text{Cha} = \text{Sibiu}),$$

$$(\text{Oradea}, g = 291, h = 380, f = 617, \text{Cha} = \text{Sibiu}),$$



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

$CLOSE = \{(Arad, g = 0, h = 0, f = 0), (Sibiu, g = 140, h = 253, f = 393, Cha = Arad)\}$

Trong tập OPEN, R. Vilcea là nút có giá trị f nhỏ nhất nên ta chọn $Tmax = R. Vilcea$. 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.

✓ $h(Sibiu) = 253$

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

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

✓ $h(Craiova) = 160$

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

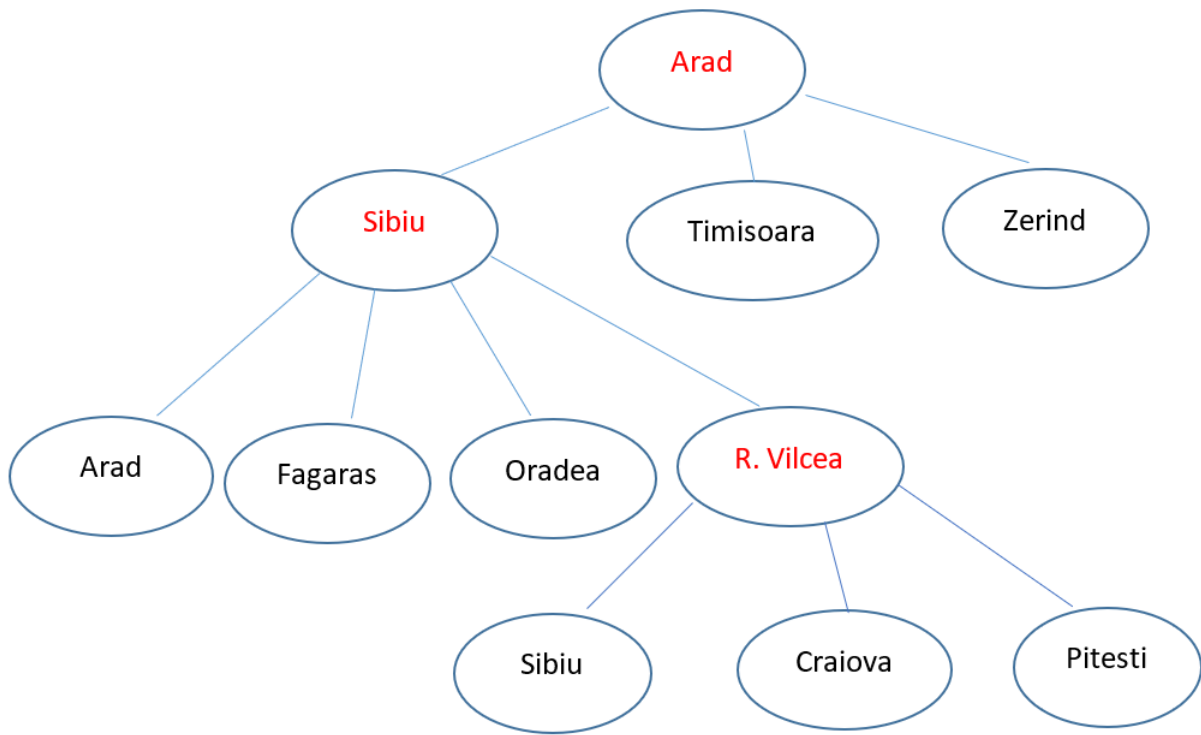
$$f(Craiova) = g(Craiova) + h(Craiova) = 366 + 160 = 526$$

✓ $h(Pitesti) = 100$

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

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

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.

✓ $h(Sibiu) = 253$

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

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

$$\checkmark h(Bucharest) = 20$$

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

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

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.

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

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

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

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

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

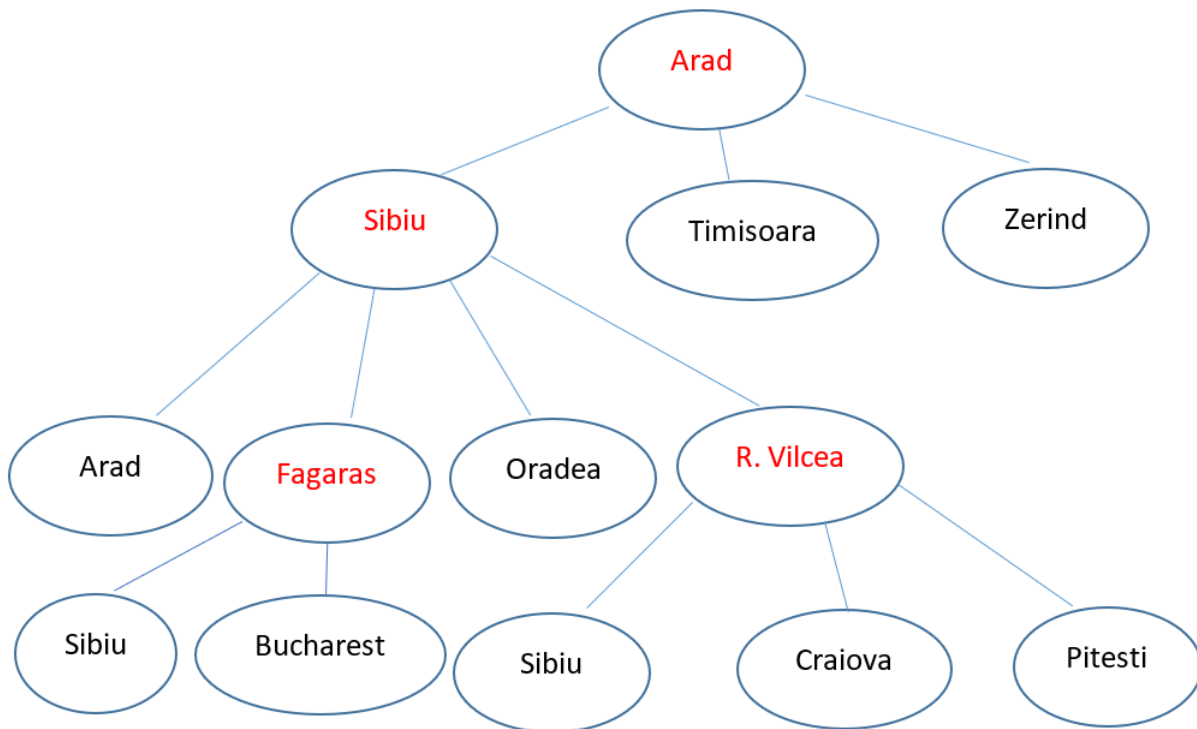
$$(Bucharest, g = 450, h = 20, f = 470, Cha = Fagaras)\}$$

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

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

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

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



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.

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

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

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

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

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

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

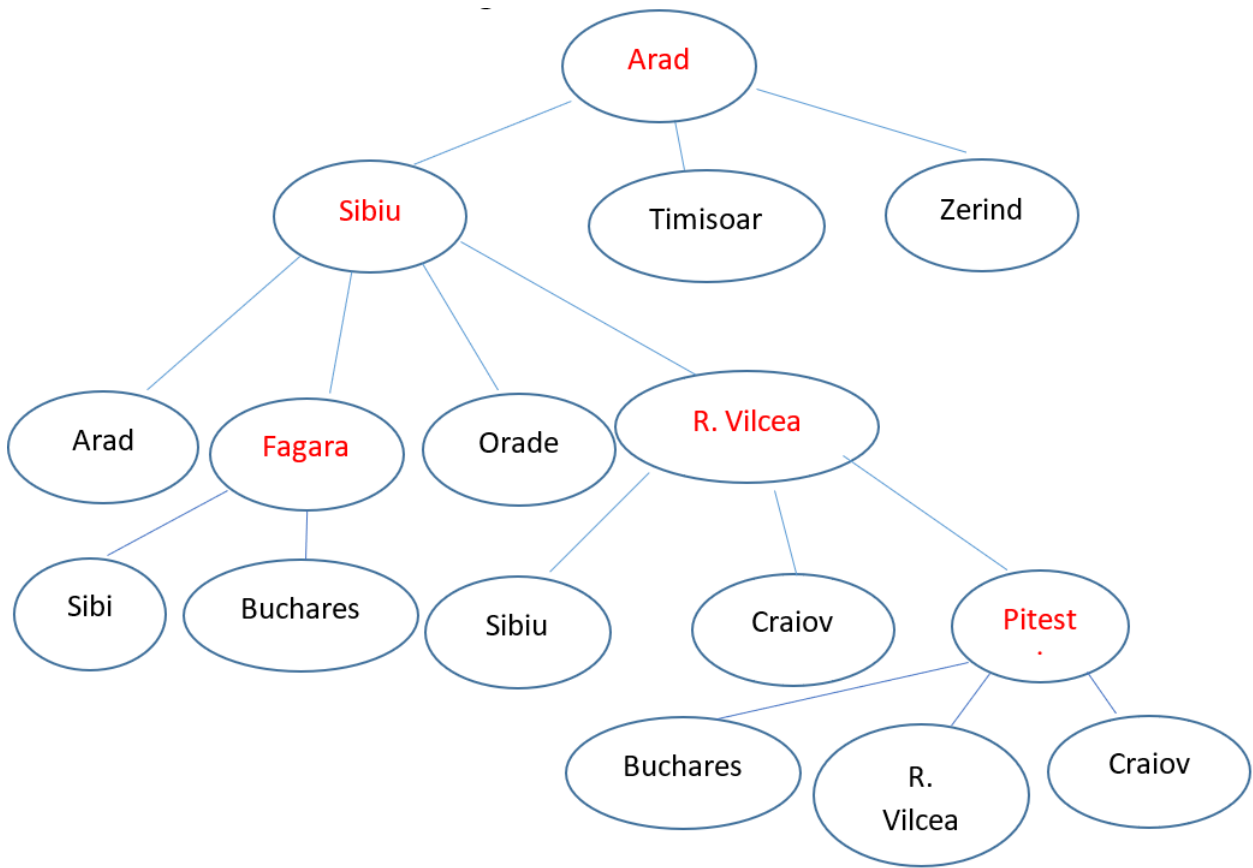
$$(Sibiu, g = 140, h = 253, f = 393, Cha = Arad),$$

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

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

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

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.

✓ $h(Pitesti) = 100$

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

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

✓ $h(Fagaras) = 176$

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

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

✓ $h(Giurgiu) = 77$

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

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

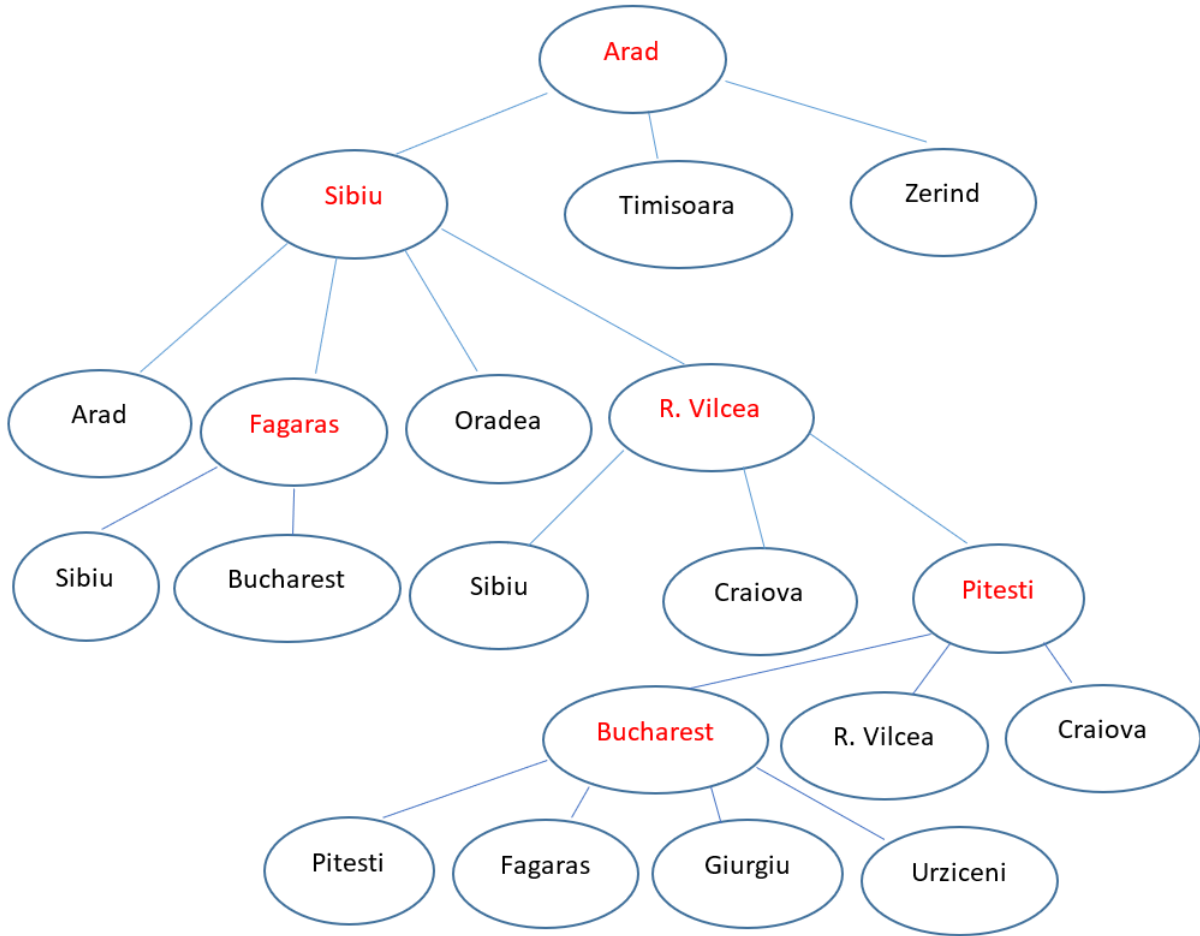
✓ $h(Urziceni) = 10$

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

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

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.



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

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

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

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

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

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

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

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

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

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

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

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

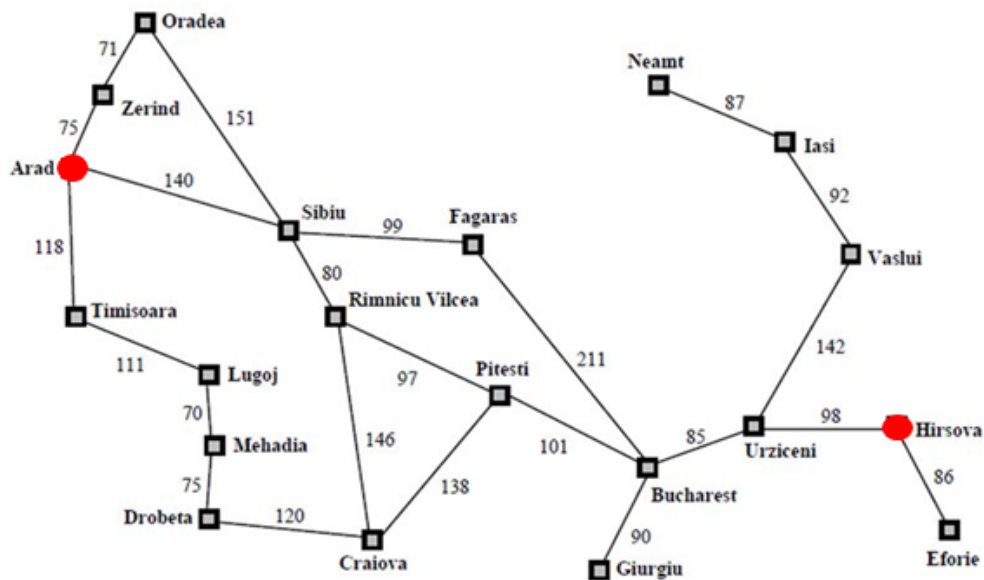
Tương tự,

III. NỘI DUNG THỰC HÀNH

1. Bài toán:

- a. Cài đặt thuật toán Greedy – Best – First search để tìm đường đi từ Arad tới Hirsova như hình với $h(n)$ được xác định như sau:

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

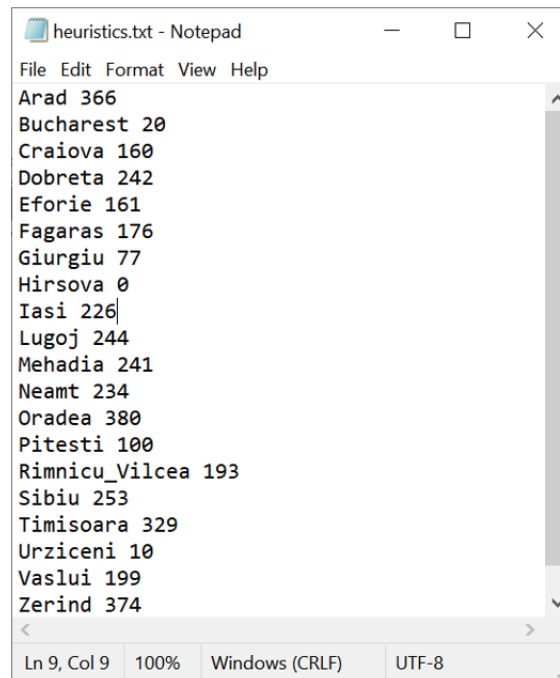


- b. Cài đặt thuật toán A^* để tìm đường đi ngắn nhất từ Arad tới Hirsova như hình với hàm $h(n)$ được xác định như trong bài tập 1 và $g(n)$ là khoảng cách giữa 2 thành phố.

2. Cài đặt:

- a. Đọc dữ liệu từ file txt:

- Lấy hàm heuristic từ file:



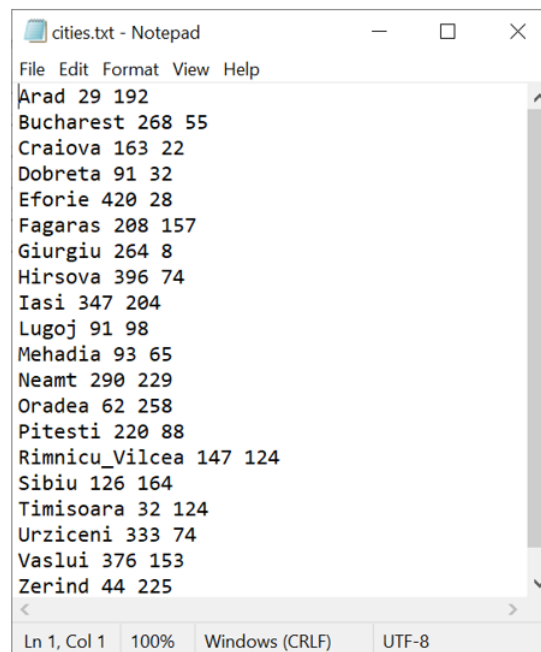
```
File Edit Format View Help
Arad 366
Bucharest 20
Craiova 160
Dobreta 242
Eforie 161
Fagaras 176
Giurgiu 77
Hirsova 0
Iasi 226
Lugoj 244
Mehadia 241
Neamt 234
Oradea 380
Pitesti 100
Rimnicu_Vilcea 193
Sibiu 253
Timisoara 329
Urziceni 10
Vaslui 199
Zerind 374
< >
Ln 9, Col 9 100% Windows (CRLF) UTF-8
```

```
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_heuristic_val[1])

    return heuristics
```

- Lấy vị trí của các thành phố từ file:



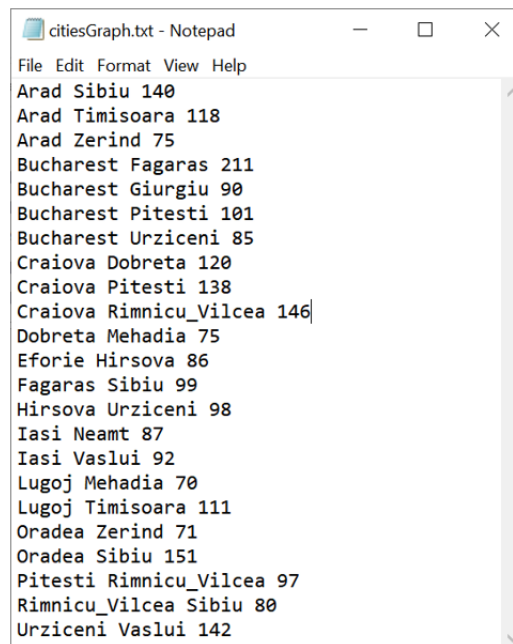
```
File Edit Format View Help
Arad 29 192
Bucharest 268 55
Craiova 163 22
Dobreta 91 32
Eforie 420 28
Fagaras 208 157
Giurgiu 264 8
Hirsova 396 74
Iasi 347 204
Lugoj 91 98
Mehadia 93 65
Neamt 290 229
Oradea 62 258
Pitesti 220 88
Rimnicu_Vilcea 147 124
Sibiu 126 164
Timisoara 32 124
Urziceni 333 74
Vaslui 376 153
Zerind 44 225
< >
Ln 1, Col 1 100% Windows (CRLF) UTF-8
```

```
def getCity():
    city = {}
    citiesCode = {}
    f = open("cities1.txt")
    j = 1
    for i in f.readlines():
        node_city_val = i.split()
        city[node_city_val[0]] = [int(node_city_val[1]), int(node_city_val[2])]

        citiesCode[j] = node_city_val[0]
        j += 1

    return city, citiesCode
```

- Khởi tạo đồ thị các thành phố từ file:



```
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
```

```
def createGraph():
    graph = {}
    file = open("citiesGraph.txt")
    for i in file.readlines():
        node_val = i.split()

        if node_val[0] in graph and node_val[1] in graph:
            c = graph.get(node_val[0])
            c.append([node_val[1], node_val[2]])
            graph.update({node_val[0]: c})

            c = graph.get(node_val[1])
            c.append([node_val[0], node_val[2]])
            graph.update({node_val[1]: c})

        elif node_val[0] in graph:
            c = graph.get(node_val[0])
            c.append([node_val[1], node_val[2]])
            graph.update({node_val[0]: c})

            graph[node_val[1]] = [[node_val[0], node_val[2]]]

        elif node_val[1] in graph:
            c = graph.get(node_val[1])
            c.append([node_val[0], node_val[2]])
            graph.update({node_val[1]: c})

            graph[node_val[0]] = [[node_val[1], node_val[2]]]

        else:
            graph[node_val[0]] = [[node_val[1], node_val[2]]]
            graph[node_val[1]] = [[node_val[0], node_val[2]]]

    return graph
```

b. Greedy – Best – First Search:

```
def GBFS(startNode, heuristics, graph, goalNode):
    priorityQueue = queue.PriorityQueue()
    priorityQueue.put((heuristics[startNode], startNode))

    path = []

    while priorityQueue.empty() == False:
        current = priorityQueue.get()[1]
        path.append(current)

        if current == goalNode:
            break

        priorityQueue = queue.PriorityQueue()

        for i in graph[current]:
            if i[0] not in path:
                priorityQueue.put((heuristics[i[0]], i[0]))

    return path
```

c. A*:

```
def Astar(startNode, heuristics, graph, goalNode):
    priorityQueue = queue.PriorityQueue()
    distance = 0
    path = []

    priorityQueue.put((heuristics[startNode] + distance, [startNode, 0]))

    while priorityQueue.empty() == False:
        current = priorityQueue.get()[1]
        path.append(current[0])
        distance += int(current[1])

        if current[0] == goalNode:
            break

        priorityQueue = queue.PriorityQueue()

        for i in graph[current[0]]:
            if i[0] not in path:
                priorityQueue.put((heuristics[i[0]] + int(i[1]) + distance, i))

    return path
```

d. Vẽ hình:


```

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]]
            second = city[gbfs[i + 1]]

            plt.plot([first[0], second[0]], [first[1], second[1]], "green")
        except:
            continue

    for i in range(len(astar)):
        try:
            first = city[astar[i]]
            second = city[astar[i + 1]]

            plt.plot([first[0], second[0]], [first[1], second[1]], "blue")
        except:
            continue

    plt.errorbar(1, 1, label="GBFS", color="green")
    plt.errorbar(1, 1, label="ASTAR", color="blue")
    plt.legend(loc="lower left")

    plt.show()

```

e. Hàm main:

```

if __name__ == "__main__":
    heuristic = getHeuristics()
    graph = createGraph()
    city, citiesCode = getCity()

    for i, j in citiesCode.items():
        print(i, j)

    while True:
        inputCode1 = int(input("Nhập đỉnh bắt đầu: "))
        inputCode2 = int(input("Nhập đỉnh kết thúc: "))

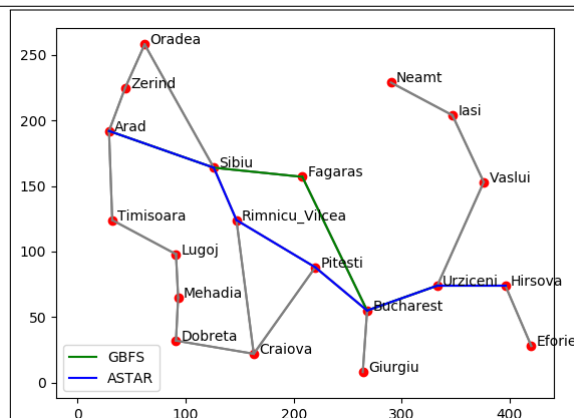
        if inputCode1 == 0 or inputCode2 == 0:
            break

        startCity = citiesCode[inputCode1]
        endCity = citiesCode[inputCode2]

        gbfs = GBFS(startCity, heuristic, graph, endCity)
        astar = Astar(startCity, heuristic, graph, endCity)
        print("GBFS => ", gbfs)
        print("ASTAR => ", astar)

        drawMap(city, gbfs, astar, graph)

```



3. Yêu cầu:

- Cài đặt và thực thi chương trình.
- Viết báo cáo trình bày:
 - ✿ Nếu chương trình bị báo lỗi thì lỗi ở dòng nào và sửa lại như thế nào? (Nếu có).
 - ✿ Chạy tay thuật toán GBFS và tiếp tục phần chạy tay chưa xong của thuật toán A^* .
 - ✿ Các thuật toán đã cho sẵn code như trên chạy ra kết quả đúng không? Nếu chưa đúng thì em sửa lại như thế nào cho phù hợp?
 - ✿ Từ đó, em có nhận xét gì về kết quả chạy tay với kết quả chạy trên máy tính.