**TRƯỜNG ĐẠI HỌC THỦ DẦU MỘT**

**VIỆN KỸ THUẬT - CÔNG NGHỆ**

****

**BÁO CÁO GIỮA KỲ**

**MÔN HỌC:**

**NHẬP MÔN TRÍ TUỆ NHÂN TẠO**

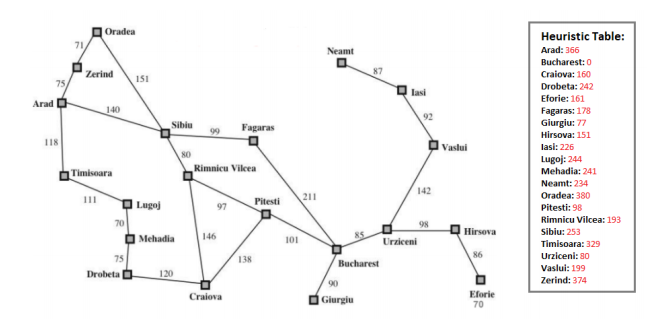
**GVHD : NGUYỄN LÊ HIỀN DUYÊN**

**NHÓM THỰC HIỆN**

|  |  |
| --- | --- |
| ***Lê Thanh Hiệp*** | ***1824801030217*** |
| ***Nguyễn Quang Tú*** | ***1824801030214*** |

*Bình Dương, tháng 3 năm 2021*

**Đề 5: Thành phố bắt đầu: Arad Phương pháp: tìm kiếm tham lam tốt nhất trước (Greedy Best-first search)**

****

1. Mô tả bài toán

Cho bản đồ như hình trên. Có các đường đi thực tế là số km được ghi trên hình từ thành phố này đến thành phố kia. Bảng bên phải là các giá trị đường chim bay từ thành phố bất kì đến Bucharest. Tìm đường đi từ Arad đến Burcharest theo thuật toán Greedy Best-first search

Input:

"Arad": {

    "Zerind": { "cost": 75 },

    "Timisoara": { "cost": 118 },

    "Sibiu": { "cost": 140 }

  },

  "Zerind": {

    "Oradea": { "cost": 71 },

    "Arad": { "cost": 75 }

  },

  "Timisoara": {

    "Lugoj": { "cost": 111 },

    "Arad": { "cost": 118 }

  },

  "Sibiu": {

    "Fagaras": { "cost": 99 },

    "Rimnicu Vilcea": { "cost": 80 },

    "Oradea": { "cost": 151 },

    "Arad": { "cost": 140 }

  },

  "Oradea": {

    "Sibiu": { "cost": 151 },

    "Zerind": { "cost": 71 }

  },

  "Lugoj": {

    "Mehadia": { "cost": 70 },

    "Timisoara": { "cost": 111 }

  },

  "Fagaras": {

    "Bucharest": { "cost": 211 },

    "Sibiu": { "cost": 99 }

  },

  "Rimnicu Vilcea": {

    "Pitesti": { "cost": 97 },

    "Craiova": { "cost": 146 },

    "Sibiu": { "cost": 99 }

  },

  "Mehadia": {

    "Drobeta": { "cost": 75 },

    "Lugoj": { "cost": 70 }

  },

  "Pitesti": {

    "Bucharest": { "cost": 101 },

    "Rimnicu Vilcea": { "cost": 97 },

    "Craiova": { "cost": 138 }

  },

  "Craiova": {

    "Pitesti": { "cost": 138 },

    "Rimnicu Vilcea": { "cost": 146 },

    "Drobeta": { "cost": 120 }

  },

  "Drobeta": {

    "Craiova": { "cost": 120 },

    "Mehadia": { "cost": 75 }

  },

  "Bucharest": {

    "Fagaras": { "cost": 211 },

    "Pitesti": { "cost": 101 },

    "Giurgiu": { "cost": 90 },

    "Urziceni": { "cost": 85 }

  },

  "Giurgiu": {

    "Bucharest": { "cost": 90 }

  },

  "Urziceni": {

    "Bucharest": { "cost": 85 },

    "Hirsova": { "cost": 98 },

    "Vaslui": { "cost": 142 }

  },

  "Hirsova": {

    "Eforie": { "cost": 70 },

    "Urziceni": { "cost": 98 }

  },

  "Vaslui": {

    "Iasi": { "cost": 92 },

    "Urziceni": { "cost": 142 }

  },

  "Iasi": {

    "Neamt": { "cost": 87 },

    "Vaslui": { "cost": 92 }

  },

  "Neamt": {

    "Iasi": { "cost": 87 }

  }

}

heuristic = {

  "Arad": 366,

  "Zerind": 374,

  "Timisoara": 329,

  "Sibiu": 253,

  "Oradea": 380,

  "Lugoj": 244,

  "Fagaras": 178,

  "Rimnicu Vilcea": 193,

  "Mehadia": 241,

  "Pitesti": 98,

  "Craiova": 160,

  "Drobeta": 242,

  "Bucharest": 0,

  "Giurgiu": 77,

  "Urziceni": 80,

  "Hirsova": 151,

  "Eforie": 161,

  "Vaslui": 199,

  "Iasi": 226,

  "Neamt": 234

}

Output: ('Bucharest', 0) -> ('Fagaras', 178) -> ('Sibiu', 253) -> ('Arad', 366)

1. Phương pháp sử dụng:

Sử dụng thuật toán Greedy Best-first search để tìm kiếm

Bắt đầu từ arad

Chọn sibiu vì có giá trị nhỏ nhất

1. Mô tả thuật toán

B1: Gán giá trị cho start và goal là arad và Bucharest

B2:Chạy hàm Gbfs

B3:Khởi tạo hàng đợi q và thêm giá trị start vào hàng đợi

B4:Khởi tạo previous kiểu dictionary lưu thông tin thành phố gán from= none cho thấy chưa các thành phố chưa được duyệt qua

B5:chạy vòng lặp while dequeue q giá trị vào biến curCity

B6 nếu curCity= goal in ra success ! và trả ra thành phố kết thúc

B7: ngược lại duyệt qua thành phố có thể đi qua từ curCity nếu chưa duyệt qua thì thêm thành phố vào q đánh dấu thành phố đã đi qua

B8: Nếu q rỗng in ra fail ngược lại sắp xếp các phần tử trong q theo giá trị tăng dần

B9: quay lại bước 5

1. Cài đặt các hàm

* Modules
* def showResultWithAttr(previous, start, goal, attr='total\_cost'):
* curCity = goal # Truy vết từ điểm đích về điểm bắt đầu
* print('Result:', (curCity, previous[curCity][attr]), end=' ')
* while curCity != start:
* curCity = previous[curCity]['from']
* print('->', (curCity, previous[curCity][attr]), end=' ')

Show kết quả

* def aweSomeSort(array, previous, sortBy='total\_cost'): # QuickSort (python version)
* less = []
* equal = []
* greater = []
* if len(array) > 1:
* pivot = previous[array[0]][sortBy]
* for city in array:
* cost = previous[city][sortBy]
* if cost < pivot:
* less.append(city)
* if cost == pivot:
* equal.append(city)
* if cost > pivot:
* greater.append(city)
* return aweSomeSort(less, previous, sortBy) + equal + aweSomeSort(greater, previous, sortBy) # toán tử nối mảng
* else:
* return array

Sắp xếp tăng dần theo giá trị heuristic

* Gbfs
* def GBFS(start, goal):
* q = queue.deque()
* q.append(start)
* previous = [] # khởi tạo previous kiểu dictionary
* for city in myMap.keys():
* previous.append((city, {'from': None, 'heuristic': heuristic[city]}))
* previous = dict(previous)
* while 1:
* curCity = q.popleft() # dequeue
* if curCity == goal:
* print('Success!')
* showResultWithAttr(previous, start, goal, 'heuristic')
* return True # Tìm thấy goal
* for city in myMap[curCity].keys(): # Các thành phố có thể đi đến được từ thành phố hiện tại
* if previous[city]['from'] == None: # Nếu chưa đi qua thành phố này
* q.append(city) # Thêm vào hàng đợi
* previous[city]['from'] = curCity # Lưu lại dấu vết
* if len(q) == 0:
* print('Fail!')
* return False # Không thể đi đến goal từ start
* q = queue.deque(aweSomeSort(q, previous, 'heuristic'))

1. Kết quả ('Bucharest', 0) -> ('Fagaras', 178) -> ('Sibiu', 253) -> ('Arad', 366)