IIAI30017

Artificial Intelligence

HW1: Greedy Best-First Search

Instructor: YuanFu Yang

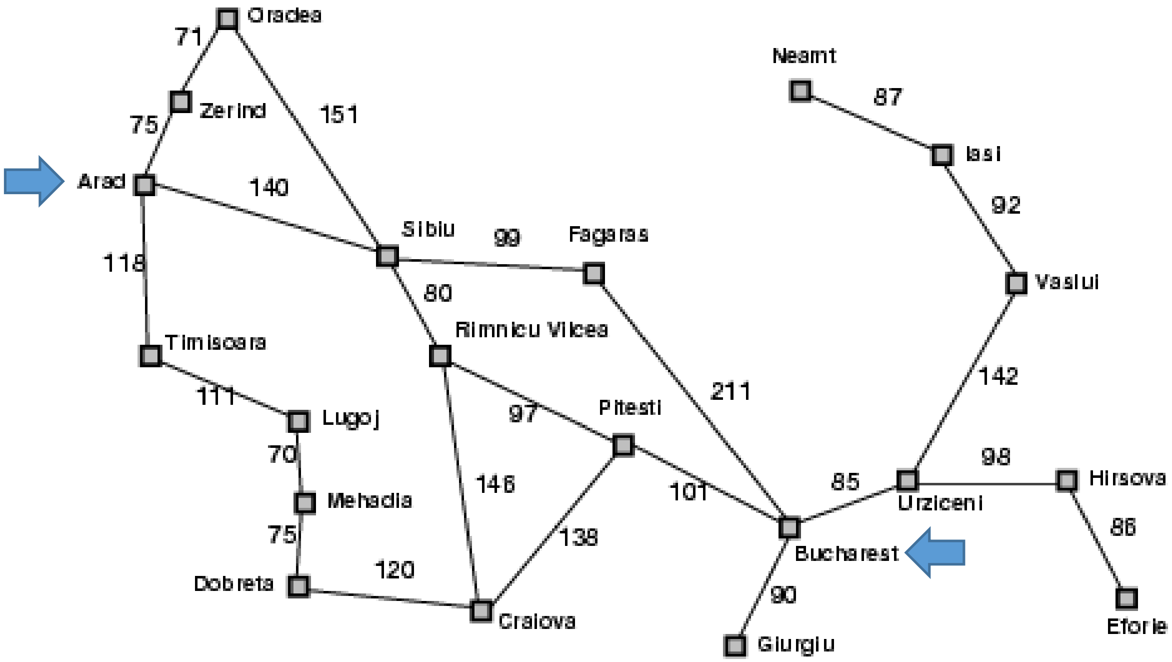
yfyangd@nycu.edu.tw

Greedy Best-First Search

- Homework due: 10/1
- Late submissions will incur a penalty of one point for each day overdue.
- The assignment allows a maximum extension of 3 days (it will not be accepted if submitted later than 3 days).
- Submit files: code and report (4 questions), and submit them in both **.ipynb** and **PDF** file formats respectively.
- This assignment can be carried out using Colab or completed on your PC.

Greedy Best-First Search

- Design a algorithm of Greedy Best-First Search for the shortest path search problem on the European railway network. Below is a European railway map where each node represents a station, and each edge represents the straight-line distance between stations.
- The table on the right shows the straight-line distance from each station to the destination station, Bucharest.
- Our goal is to find the optimal path from the starting station to the destination station, Bucharest.



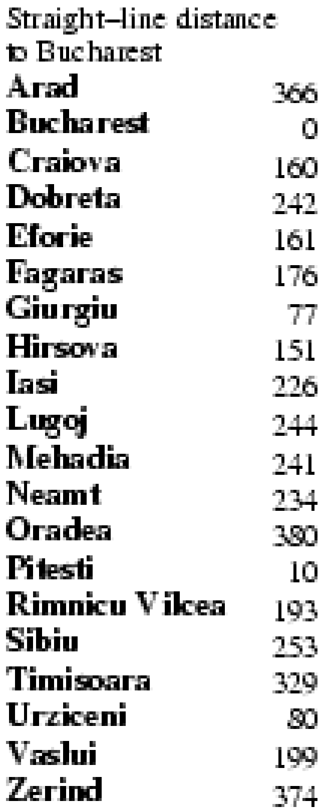
Straight−line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

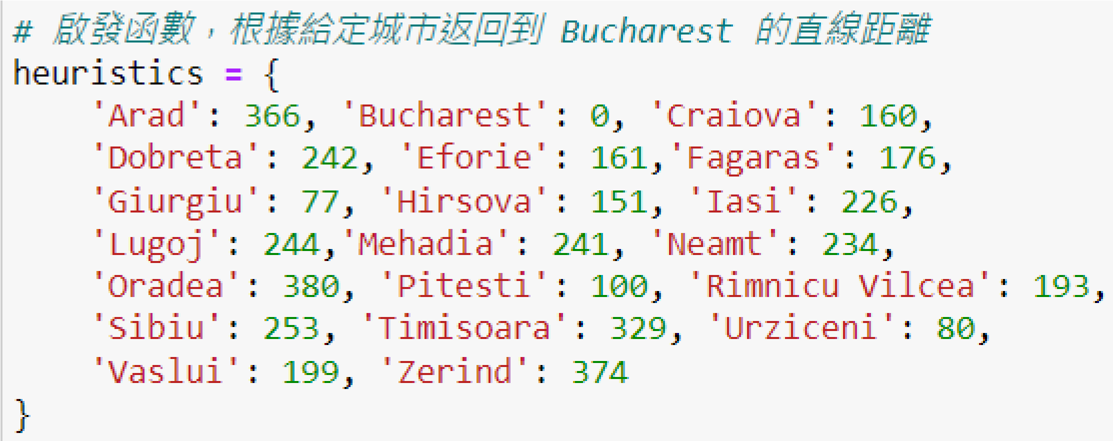
通大學 TUNG UNIVERSITY

Greedy Best-First Search

- The heuristic function (the straight-line distance from each station to the destination station, Bucharest) is defined as follows. Please make good use of it to perform Greedy Best-First Search.



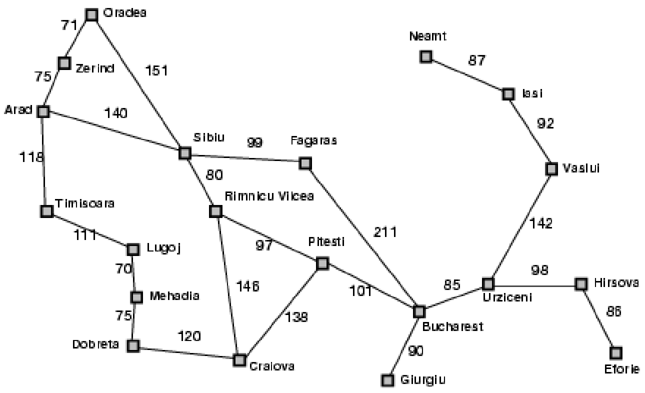
Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

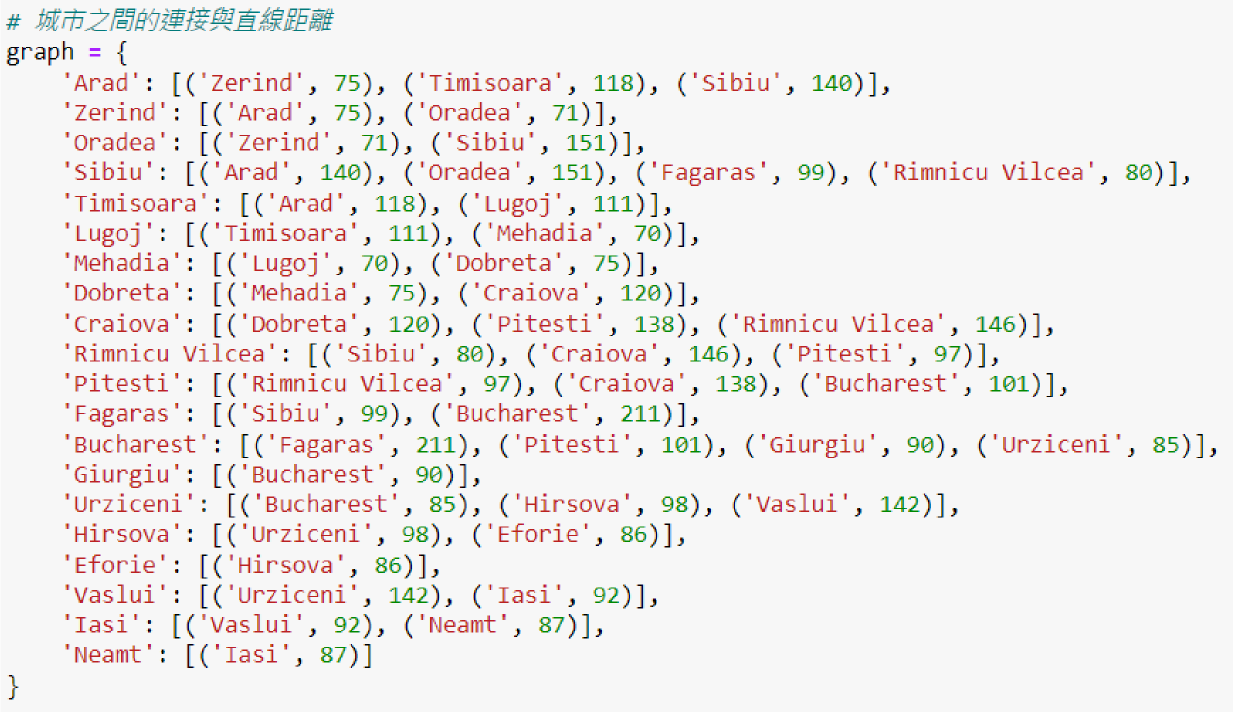


```python
# 啟發函數，根據給定城市返回到 Bucharest 的直線距離
heuristics = {
    'Arad': 366, 'Bucharest': 0, 'Craiova': 160,
    'Dobreta': 242, 'Eforie': 161,'Fagaras': 176,
    'Giurgiu': 77, 'Hirsova': 151, 'Iasi': 226,
    'Lugoj': 244,'Mehadia': 241, 'Neamt': 234,
    'Oradea': 380, 'Pitesti': 100, 'Rimnicu Vilcea': 193,
    'Sibiu': 253, 'Timisoara': 329, 'Urziceni': 80,
    'Vaslui': 199, 'Zerind': 374
}
```

Greedy Best-First Search

- The European railway network graph and the distances between stations are also defined as follows. Please follow this graph to perform Greedy Best-First Search.

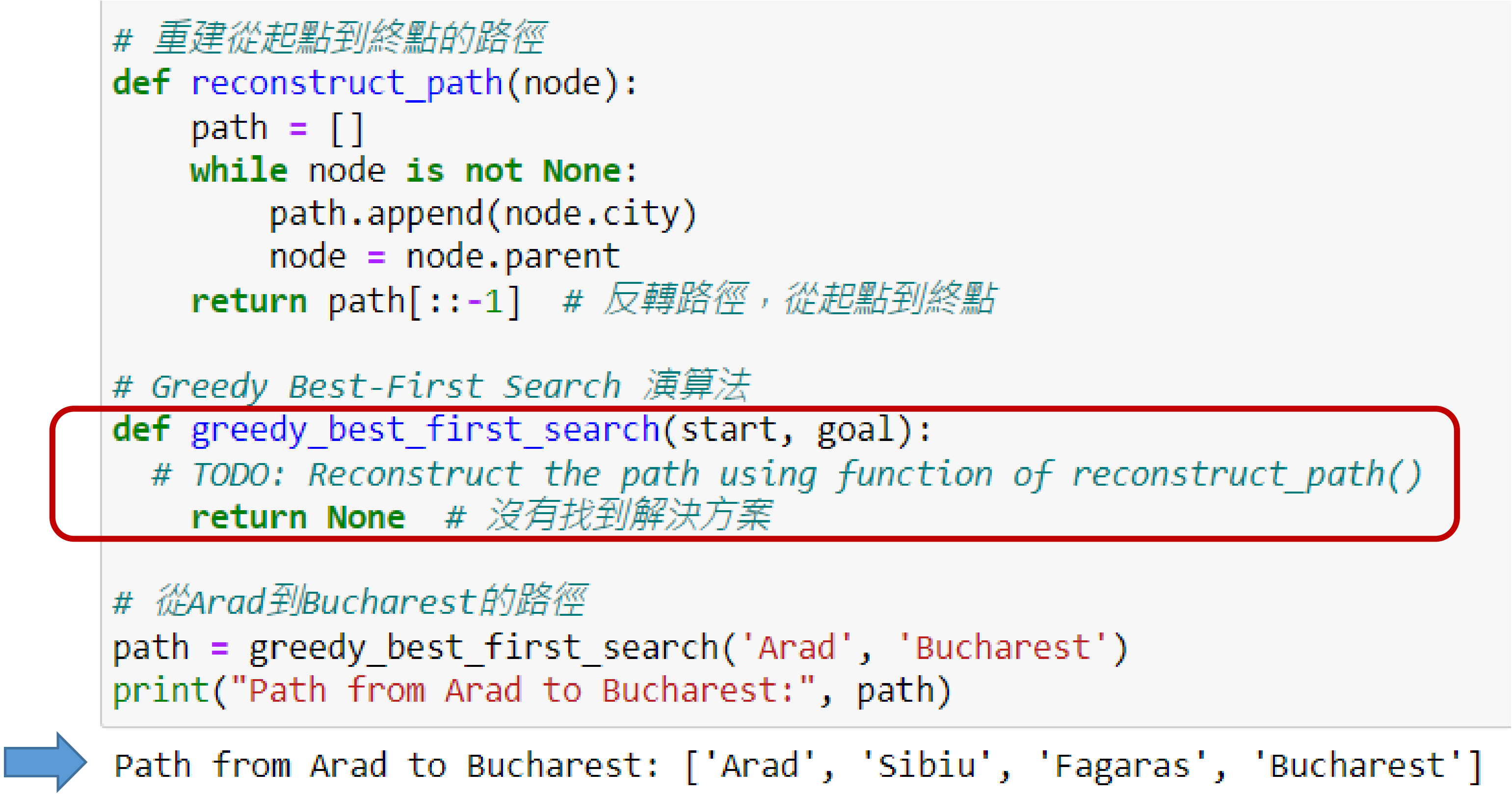




```
# 城市之間的連接與直線距離
graph = {
    'Arad': [('Zerind', 75), ('Timisoara', 118), ('Sibiu', 140)],
    'Zerind': [('Arad', 75), ('Oradea', 71)],
    'Oradea': [('Zerind', 71), ('Sibiu', 151)],
    'Sibiu': [('Arad', 140), ('Oradea', 151), ('Fagaras', 99), ('Rimnicu Vilcea', 80)],
    'Timisoara': [('Arad', 118), ('Lugoj', 111)],
    'Lugoj': [('Timisoara', 111), ('Mehadia', 70)],
    'Mehadia': [('Lugoj', 70), ('Dobreta', 75)],
    'Dobreta': [('Mehadia', 75), ('Craiova', 120)],
    'Craiova': [('Dobreta', 120), ('Pitesti', 138), ('Rimnicu Vilcea', 146)],
    'Rimnicu Vilcea': [('Sibiu', 80), ('Craiova', 146), ('Pitesti', 97)],
    'Pitesti': [('Rimnicu Vilcea', 97), ('Craiova', 138), ('Bucharest', 101)],
    'Fagaras': [('Sibiu', 99), ('Bucharest', 211)],
    'Bucharest': [('Fagaras', 211), ('Pitesti', 101), ('Giurgiu', 90), ('Urziceni', 85)],
    'Giurgiu': [('Bucharest', 90)],
    'Urziceni': [('Bucharest', 85), ('Hirsova', 98), ('Vaslui', 142)],
    'Hirsova': [('Urziceni', 98), ('Eforie', 86)],
    'Eforie': [('Hirsova', 86)],
    'Vaslui': [('Urziceni', 142), ('Iasi', 92)],
    'Iasi': [('Vaslui', 92), ('Neamt', 87)],
    'Neamt': [('Iasi', 87)]
}
```

Greedy Best-First Search

- Plan-to-Do
  - Complete and refine a function for greedy_best_first_search(start, goal), and execute the route planning to ultimately provide the route from Arad to Bucharest (For Question-1).



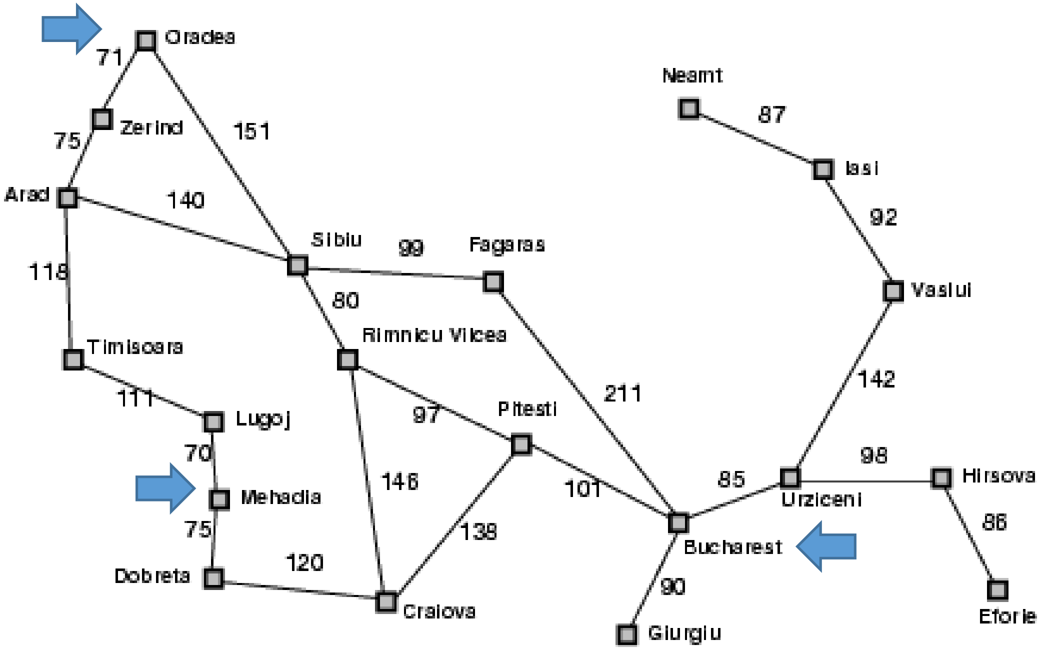
```
# 重建從起點到終點的路徑
def reconstruct_path(node):
    path = []
    while node is not None:
        path.append(node.city)
        node = node.parent
    return path[::-1]  # 反轉路徑，從起點到終點

# Greedy Best-First Search 演算法
def greedy_best_first_search(start, goal):
  # TODO: Reconstruct the path using function of reconstruct_path()
    return None  # 沒有找到解決方案

# 從Arad到Bucharest的路徑
path = greedy_best_first_search('Arad', 'Bucharest')
print("Path from Arad to Bucharest:", path)
```

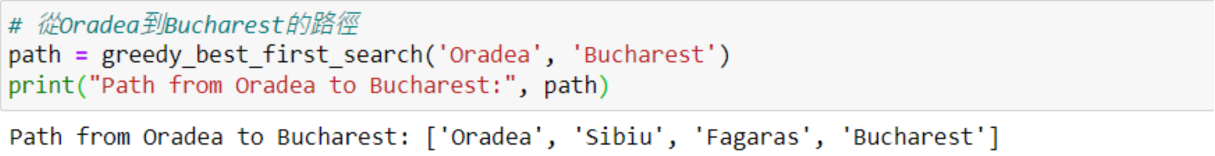
Output

```
Path from Arad to Bucharest: ['Arad', 'Sibiu', 'Fagaras', 'Bucharest']
```

Greedy Best-First Search

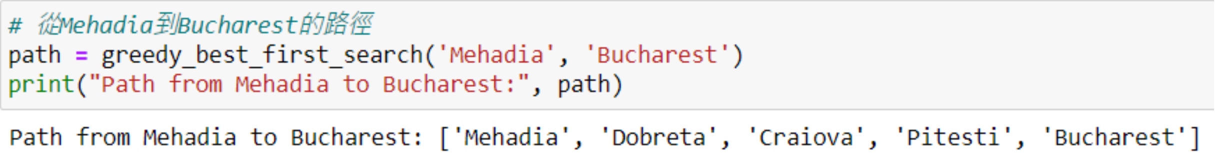
- Provide the route planning from Oradea and Mehadia to the destination station Bucharest, respectively (Q2).





```
# 從Oradea到Bucharest的路徑
path = greedy_best_first_search('Oradea', 'Bucharest')
print("Path from Oradea to Bucharest:", path)
```

```
Path from Oradea to Bucharest: ['Oradea', 'Sibiu', 'Fagaras', 'Bucharest']
```



```
# 從Mehadia到Bucharest的路徑
path = greedy_best_first_search('Mehadia', 'Bucharest')
print("Path from Mehadia to Bucharest:", path)
```

```
Path from Mehadia to Bucharest: ['Mehadia', 'Dobreta', 'Craiova', 'Pitesti', 'Bucharest']
```

Greedy Best-First Search

- Homework Question:

Q1: Provide the route planning from Arad to the destination station Bucharest.

Q1: Provide the route planning from Oradea and Mehadia to the destination station Bucharest, respectively.

Q3: Discuss the Pros and Cons of Greedy Best-First Search.

Q4: Provide suggestions for improving the disadvantages of Greedy Best-First Search.

IIAI30017

Artificial Intelligence

HW1: Greedy Best-First Search

Q & A

Instructor: YuanFu Yang

yfyangd@nycu.edu.tw