



American International University-Bangladesh
Faculty Science & Technology
Mid Term Assignment

COURSE INSTRUCTOR: DR. M M MANJURUL ISLAM

NAME	TAMIMUL ALAM
STUDENT ID	20-42215-1
SUBJECT	ARTIFICIAL INTELLIGENCE AND EXPERT SYSTEM
SECTION	A
DEPARTMENT	BSc CSE

SUBMISSION DATE:10/28/2021

Summarized report

Breadth first search: BFS is a graph traversal algorithm that requires you to start at a specific node (source or starting node) and traverse the graph layer by layer, thus examining the neighbor nodes (nodes which are directly connected to source node). Then proceed to the neighbor nodes on the following level.

Path Cost report:

Possible Paths:

['Arad', 'Sibiu', 'Fagaras', 'Bucharest'] 140

['Arad', 'Sibiu', 'Rimnicu Vilcea', 'Pitesti', 'Bucharest'] 140

['Arad', 'Sibiu', 'Rimnicu Vilcea', 'Pitesti', 'Bucharest'] 239

['Arad', 'Sibiu', 'Rimnicu Vilcea', 'Pitesti', 'Bucharest'] 450

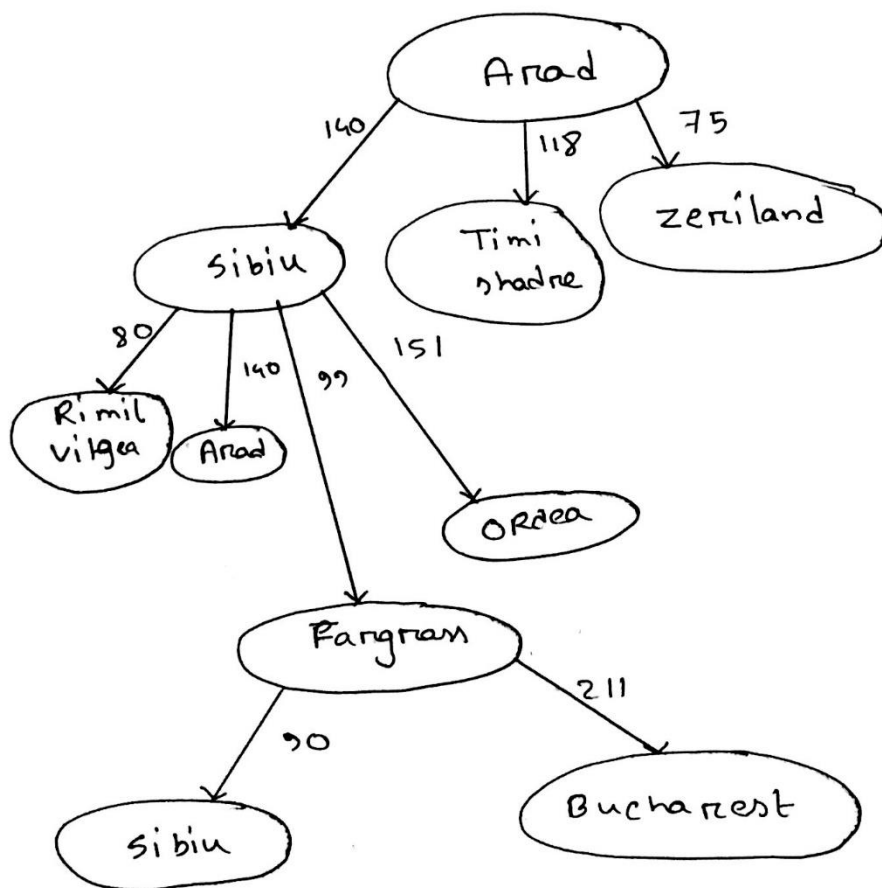
Shortest Path:

['Arad', 'Sibiu', 'Fagaras', 'Bucharest'] 140

Name: Tamimul Alam

ID: 20-42215-1

BFS:



shortest path : Arad \rightarrow Sibiu \rightarrow Fagaras \rightarrow Bucharest
(path cost 140)

Depth first search: The depth-first search (DFS) algorithm is used to traverse or explore data structures such as trees and graphs. The algorithm starts at the root node (in the case of a graph, any random node can be used as the root node) and examines each branch as far as feasible before retracing.

Path Cost report:

Possible paths:

['Arad', 'Timisoara', 'Lugoj', 'Mehadia', 'Drobeta', 'Craiova', 'Pitesti', 'Bucharest']
118

['Arad', 'Timisoara', 'Lugoj', 'Mehadia', 'Drobeta', 'Craiova', 'Pitesti', 'Rimnicu
Vilcea', 'Sibiu', 'Fagaras', 'Bucharest'] 118

['Arad', 'Timisoara', 'Lugoj', 'Mehadia', 'Drobeta', 'Craiova', 'Pitesti', 'Rimnicu
Vilcea', 'Sibiu', 'Fagaras', 'Bucharest'] 229

['Arad', 'Timisoara', 'Lugoj', 'Mehadia', 'Drobeta', 'Craiova', 'Pitesti', 'Rimnicu
Vilcea', 'Sibiu', 'Fagaras', 'Bucharest'] 299

['Arad', 'Timisoara', 'Lugoj', 'Mehadia', 'Drobeta', 'Craiova', 'Pitesti', 'Rimnicu
Vilcea', 'Sibiu', 'Fagaras', 'Bucharest'] 374

['Arad', 'Timisoara', 'Lugoj', 'Mehadia', 'Drobeta', 'Craiova', 'Pitesti', 'Rimnicu
Vilcea', 'Sibiu', 'Fagaras', 'Bucharest'] 494

['Arad', 'Timisoara', 'Lugoj', 'Mehadia', 'Drobeta', 'Craiova', 'Pitesti', 'Rimnicu
Vilcea', 'Sibiu', 'Fagaras', 'Bucharest'] 632

['Arad', 'Timisoara', 'Lugoj', 'Mehadia', 'Drobeta', 'Craiova', 'Pitesti', 'Rimnicu
Vilcea', 'Sibiu', 'Fagaras', 'Bucharest'] 733

Shortest path:

['Arad', 'Timisoara', 'Lugoj', 'Mehadia', 'Drobeta', 'Craiova', 'Pitesti', 'Bucharest']
118

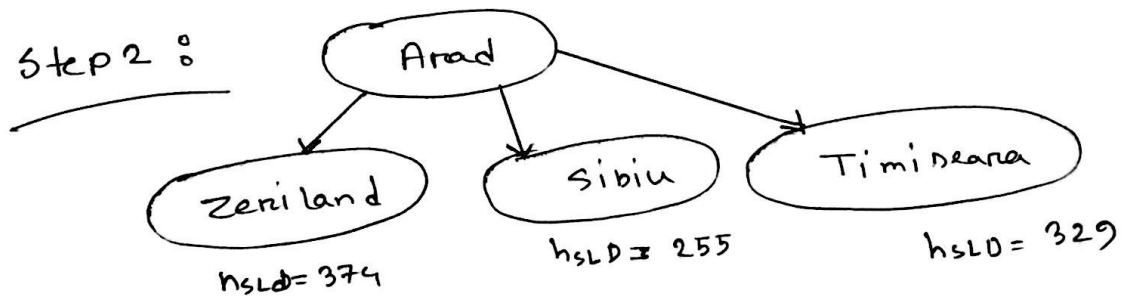
Greedy best first search: Expand the parent's first successor using a greedy method. Following the creation of a successor

- i) If the successor's heuristic outperforms the parent's, the successor is moved to the front of the queue (with the parent reinserted exactly behind it) and the loop is restarted.
- ii) If not, the successor is added to the queue (in a location determined by its heuristic value). The method will assess the parent's remaining successors if any.

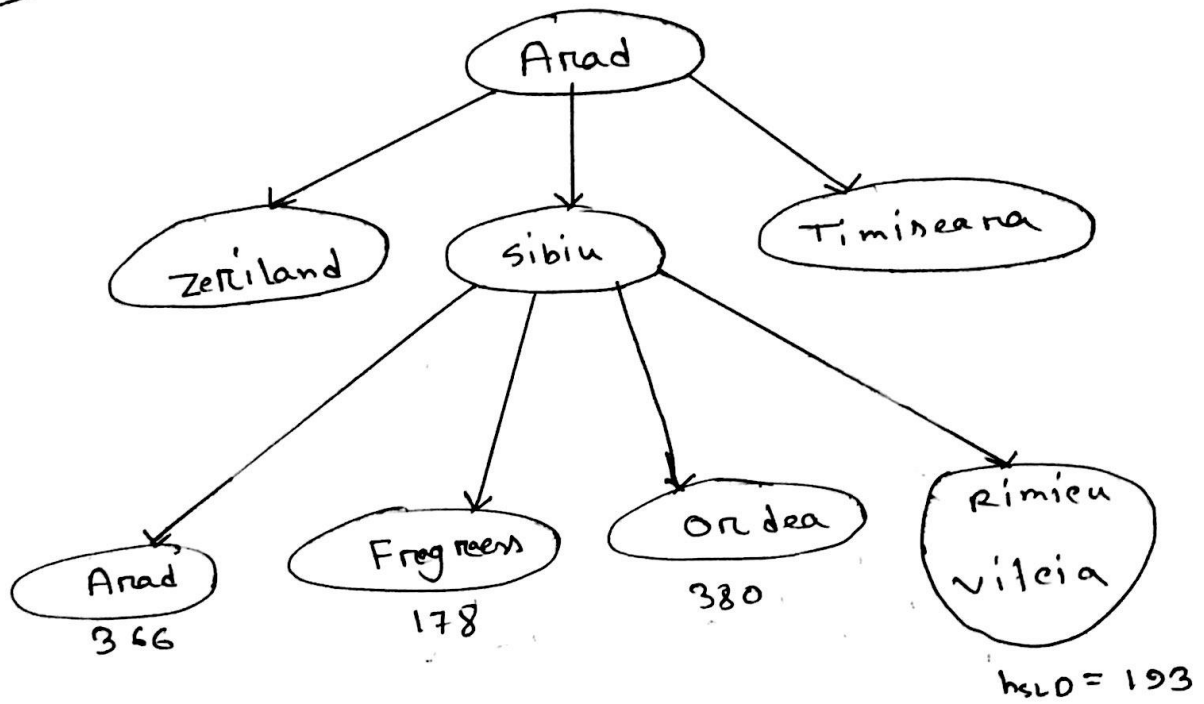
Greedy best first search

Goal is Bucharest \rightarrow

Start node is Arad

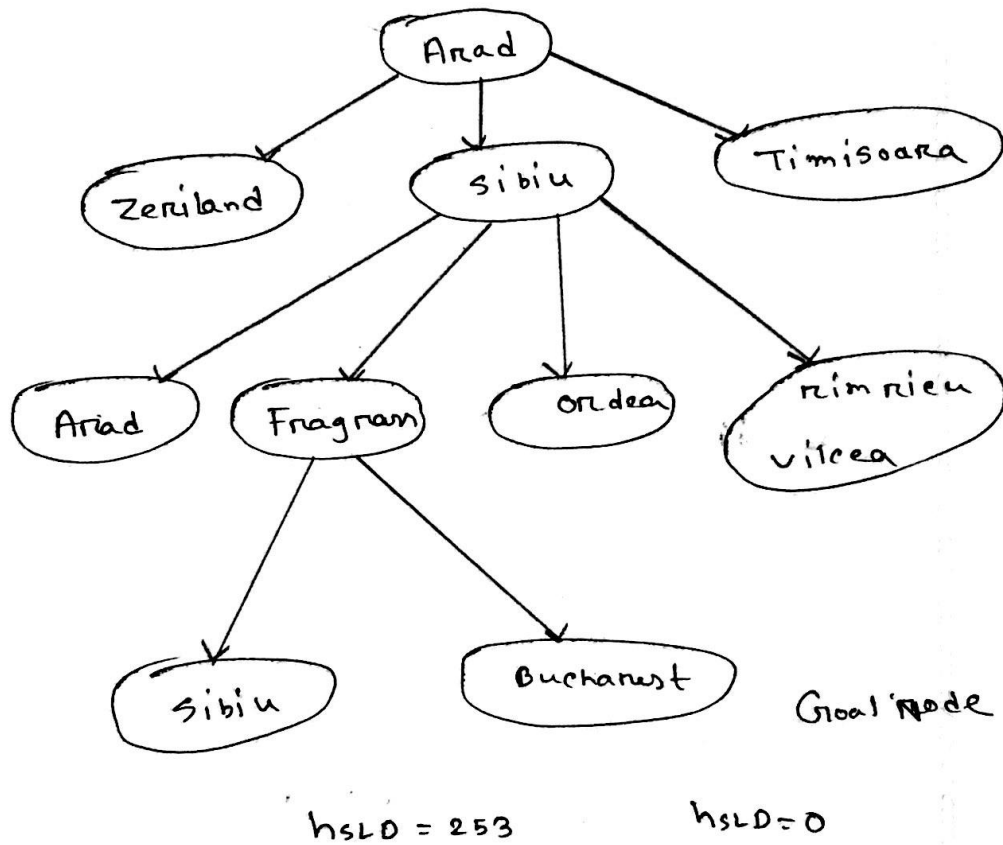


Step 3:



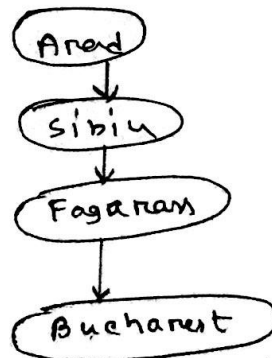
Expand with minimum hSLD

Step 4 :



Optimum path is →

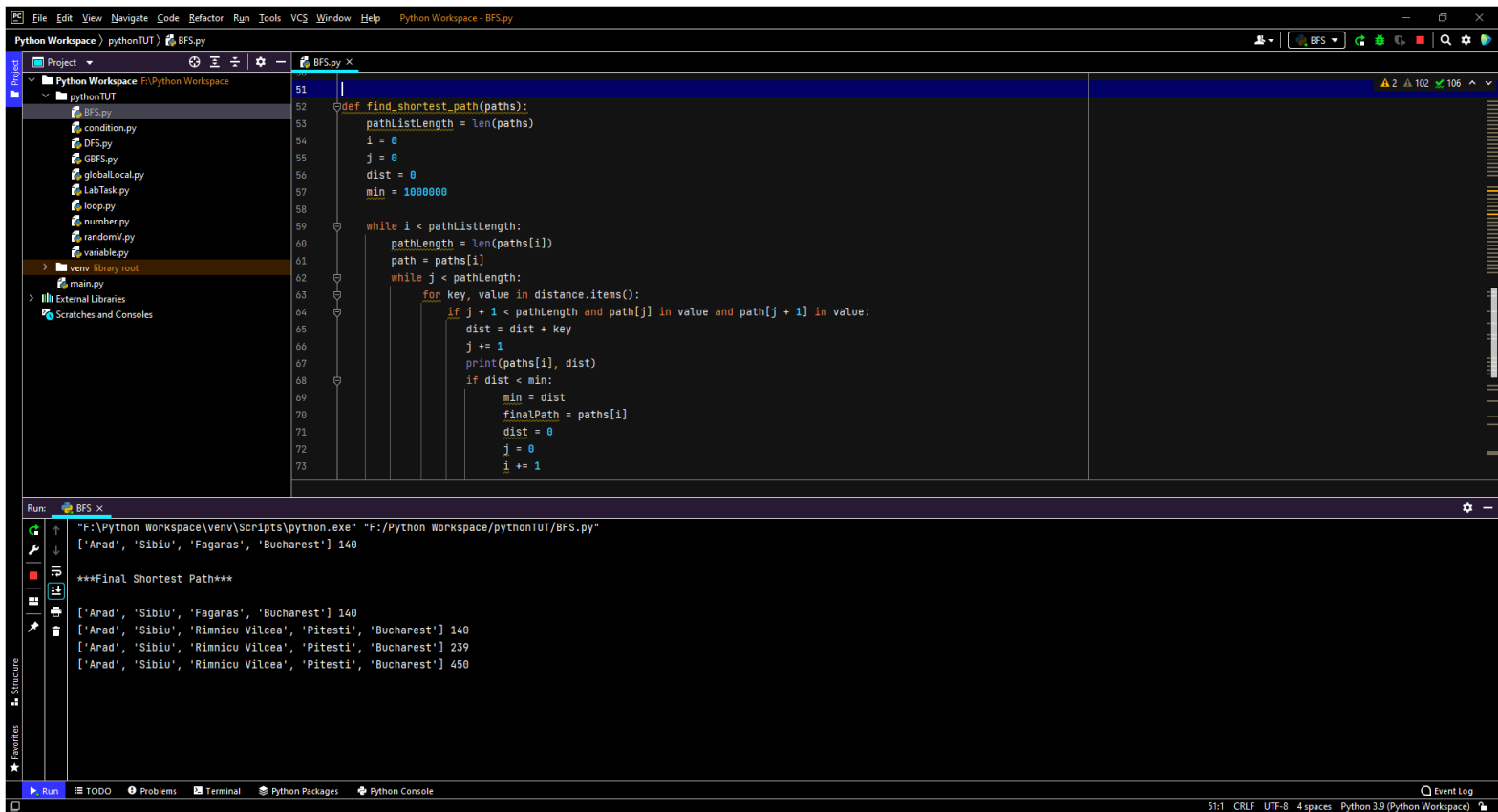
GBFS



Edge cost
⇒ 140 + 99 + 211
= 450

Output Screenshot

BFS



The screenshot displays a Python IDE with a project named 'Python Workspace'. The file explorer on the left shows a directory structure with files like 'BFS.py', 'condition.py', 'DFS.py', 'GBFS.py', 'globalLocal.py', 'LabTask.py', 'loop.py', 'number.py', 'randomV.py', 'variable.py', 'main.py', and 'venv'. The main editor window shows the code for 'BFS.py', which implements a Breadth-First Search algorithm to find the shortest path between nodes. The code includes a function 'find_shortest_path' that takes a list of paths and returns the shortest one. The output window at the bottom shows the execution results, including the shortest path and its length.

```
def find_shortest_path(paths):
    pathListLength = len(paths)
    i = 0
    j = 0
    dist = 0
    min = 1000000

    while i < pathListLength:
        pathLength = len(paths[i])
        path = paths[i]
        while j < pathLength:
            for key, value in distance.items():
                if j + 1 < pathLength and path[j] in value and path[j + 1] in value:
                    dist = dist + key
                    j += 1
                    print(paths[i], dist)
                    if dist < min:
                        min = dist
                        finalPath = paths[i]
                        dist = 0
                        j = 0
                        i += 1
```

Run: BFS x

```
"F:\Python Workspace\venv\Scripts\python.exe" "F:/Python Workspace/pythonTUT/BFS.py"
['Arad', 'Sibiu', 'Fagaras', 'Bucharest'] 140

***Final Shortest Path***

['Arad', 'Sibiu', 'Fagaras', 'Bucharest'] 140
['Arad', 'Sibiu', 'Rimnicu Vilcea', 'Pitesti', 'Bucharest'] 140
['Arad', 'Sibiu', 'Rimnicu Vilcea', 'Pitesti', 'Bucharest'] 239
['Arad', 'Sibiu', 'Rimnicu Vilcea', 'Pitesti', 'Bucharest'] 450
```

51:1 CRLF UTF-8 4 spaces Python 3.9 (Python Workspace)

DFS

The screenshot displays a Python IDE with a project named 'Python Workspace'. The file explorer on the left shows a directory structure with files like 'BFS.py', 'condition.py', 'DFS.py', 'GBFS.py', 'globalLocal.py', 'LabTask.py', 'loop.py', 'number.py', 'randomV.py', 'variable.py', 'venv\library root', 'main.py', and 'External Libraries'. The 'DFS.py' file is open in the editor, showing a Depth-First Search (DFS) algorithm. The code starts with a list of paths and iterates through them, updating the shortest path found. The output window at the bottom shows the execution results, including the shortest path and its length.

```
54     j = 0
55     dist = 0
56     min = 1000000
57
58     while i < pathListLength:
59         pathLength = len(paths[i])
60         path = paths[i]
61         while j < pathLength:
62             for key, value in distance.items():
63                 if j + 1 < pathLength and path[j] in value and path[j + 1] in value:
64                     dist = dist + key
65                     j += 1
66                     print(paths[i], dist)
67                     if dist < min:
68                         min = dist
69                         finalPath = paths[i]
70                         dist = 0
71                         j = 0
72                         i += 1
73
```

Run: DFS x

```
"F:\Python Workspace\venv\Scripts\python.exe" "F:\Python Workspace\pythonTUT\DFS.py"
['Arad', 'Timisoara', 'Lugoj', 'Mehadia', 'Drobeta', 'Craiova', 'Rimnicu Vilcea', 'Sibiu', 'Fagaras', 'Bucharest'] 118

***Final Shortest Path***

['Arad', 'Timisoara', 'Lugoj', 'Mehadia', 'Drobeta', 'Craiova', 'Rimnicu Vilcea', 'Sibiu', 'Fagaras', 'Bucharest'] 118
['Arad', 'Timisoara', 'Lugoj', 'Mehadia', 'Drobeta', 'Craiova', 'Rimnicu Vilcea', 'Pitesti', 'Bucharest'] 118
['Arad', 'Timisoara', 'Lugoj', 'Mehadia', 'Drobeta', 'Craiova', 'Rimnicu Vilcea', 'Pitesti', 'Bucharest'] 229
['Arad', 'Timisoara', 'Lugoj', 'Mehadia', 'Drobeta', 'Craiova', 'Rimnicu Vilcea', 'Pitesti', 'Bucharest'] 299
['Arad', 'Timisoara', 'Lugoj', 'Mehadia', 'Drobeta', 'Craiova', 'Rimnicu Vilcea', 'Pitesti', 'Bucharest'] 374
['Arad', 'Timisoara', 'Lugoj', 'Mehadia', 'Drobeta', 'Craiova', 'Rimnicu Vilcea', 'Pitesti', 'Bucharest'] 494
['Arad', 'Timisoara', 'Lugoj', 'Mehadia', 'Drobeta', 'Craiova', 'Rimnicu Vilcea', 'Pitesti', 'Bucharest'] 640
['Arad', 'Timisoara', 'Lugoj', 'Mehadia', 'Drobeta', 'Craiova', 'Rimnicu Vilcea', 'Pitesti', 'Bucharest'] 720
['Arad', 'Timisoara', 'Lugoj', 'Mehadia', 'Drobeta', 'Craiova', 'Rimnicu Vilcea', 'Pitesti', 'Bucharest'] 819
['Arad', 'Timisoara', 'Lugoj', 'Mehadia', 'Drobeta', 'Craiova', 'Rimnicu Vilcea', 'Pitesti', 'Bucharest'] 1030
```

49:1 CRLF UTF-8 4 spaces Python 3.9 (Python Workspace)

GBFS

The screenshot shows an IDE window titled "Python Workspace - GBFS.py". The left sidebar displays a project tree with the following structure:

- Python Workspace
- pythonTUT
 - BFS.py
 - condition.py
 - DFS.py
 - GBFS.py
 - globalLocal.py
 - LabTask.py
 - loop.py
 - number.py
 - randomV.py
 - variable.py
- venv library root
- main.py
- External Libraries
- Scratches and Consoles

The main editor displays the code for GBFS.py, with line numbers 37 to 55 visible:

```
37
38     snode = (h[startingNode], startingNode, [startingNode], 0)
39     frontier.put(snode)
40
41     while not frontier.empty():
42         unode = frontier.get()
43         u = unode[1]
44
45         if u == destinationNode:
46             print(unode)
47             print(expanded)
48             return unode[2]
49         expanded.append(u)
50         for v in R_map[u].keys():
51             if v not in expanded:
52                 cost = unode[3] + R_map[u][v]
53                 path = unode[2] + [v]
54                 frontier.put((h[v], v, path, cost))
55     print('Failed to Find')
```

The bottom panel shows the Run output for GBFS.py:

```
Run: GBFS x
"F:\Python Workspace\venv\Scripts\python.exe" "F:/Python Workspace/pythonTUT/GBFS.py"
(0, 'Bucharest', ['Arad', 'Sibiu', 'Fagaras', 'Bucharest'], 450)
['Arad', 'Sibiu', 'Fagaras']
Process finished with exit code 0
```

The status bar at the bottom indicates the file encoding is UTF-8, the line length is 31, and the Python version is 3.9.