

**INSTITUTE OF SPACE TECHNOLOGY**

**ASSIGNMENT # 1**

***ARTIFICIAL INTELLIGENCE***

**NAME: HAMAAD ALI**

**REG NO: 200901044**

**SEC: B**

**SUBMITTED TO: MAM REEDA**

**SUBMISSION DATE: 23 FEB 2023**

**BSCS\_01\_B**

**EXPLANATION:**

We must change the default DFS algorithm such that it takes edge weights into account in order to construct DFS with weights. To keep track of the nodes that need to be visited in DFS with weights, a priority queue must be used rather than a stack. The source node is our initial stop, and we first investigate its closest neighbours based on their weights. After we get at the desired node, we repeat the same procedure recursively.

We must adapt the basic BFS algorithm so that it takes edge weights into account in order to implement BFS with weights. We need to utilise a priority queue rather than a queue in BFS with weights to keep track of the nodes that need to be visited. The source node is our initial stop, and we first investigate its closest neighbours based on their weights. The same procedure is then repeated till we arrive at the destination node.

**CODE:**

import heapq

def dfs(graph, start, goal):

visited = set()

stack = [(0, start, [])] # tuple of (cost, node, path)

while stack:

(cost, current, path) = heapq.heappop(stack)

if current not in visited:

visited.add(current)

path = path + [current]

if current == goal:

return path

for neighbor, weight in graph[current].items():

if neighbor not in visited:

heapq.heappush(stack, (cost + weight, neighbor, path))

return None

def bfs(graph, start, goal):

visited = set()

queue = [(0, start, [])] # tuple of (cost, node, path)

while queue:

(cost, current, path) = heapq.heappop(queue)

if current not in visited:

visited.add(current)

path = path + [current]

if current == goal:

return path

for neighbor, weight in graph[current].items():

if neighbor not in visited:

heapq.heappush(queue, (cost + weight, neighbor, path))

return None

graph = {

"Arad": {"Zerind": 75, "Timisoara": 118, "Sibiu": 140},

"Zerind": {"Oradea": 71, "Arad": 75},

"Oradea": {"Sibiu": 151, "Zerind": 71},

"Timisoara": {"Lugoj": 111, "Arad": 118},

"Lugoj": {"Mehadia": 70, "Timisoara": 111},

"Mehadia": {"Drobeta": 75, "Lugoj": 70},

"Drobeta": {"Craiova": 120, "Mehadia": 75},

"Sibiu": {"Rimnicu Vilcea": 80, "Fagaras": 99, "Arad": 140, "Oradea": 151},

"Rimnicu Vilcea": {"Craiova": 146, "Sibiu": 80, "Pitesti": 97},

"Craiova": {"Drobeta": 120, "Rimnicu Vilcea": 146, "Pitesti": 138

**GITHUB LINK:**

[**https://github.com/HamaadSahi**](https://github.com/HamaadSahi)