**DEPARTMENT OF COMPUTER & SOFTWARE ENGINEERING**

**COLLEGE OF E&ME, NUST, RAWALPINDI**

AI & Decision Support Systems

Lab Mid

**Student Name: Nawab Aarij Imam**

**Degree/ Syndicate: 43 CE - A**

**Task1:**

**Code:**

def a\_star(start, goal, graph, heuristics):

open\_list = [(start, 0, heuristics[start])]

visited = set()

cost\_so\_far = {start: 0}

cameFrom = {start: None}

while open\_list:

current\_node, current\_g, current\_f = open\_list.pop(0)

if current\_node == goal:

return reconstruct\_path(cameFrom, current\_node)

visited.add(current\_node)

for neighbor, edge\_cost in graph[current\_node].items():

if neighbor in visited:

continue

tentative\_cost = cost\_so\_far[current\_node] + edge\_cost

if neighbor not in cost\_so\_far or tentative\_cost < cost\_so\_far[neighbor]:

cameFrom[neighbor] = current\_node

cost\_so\_far[neighbor] = tentative\_cost

new\_cost = tentative\_cost + heuristics[neighbor]

open\_list.append((neighbor, tentative\_cost, new\_cost))

open\_list = sorted(open\_list, key=lambda x: x[2])

return None

def reconstruct\_path(cameFrom, current):

total\_path = [current]

while current in cameFrom and cameFrom[current] is not None:

current = cameFrom[current]

total\_path.append(current)

total\_path.reverse()

return total\_path

def main():

graph = {

'A': {'B': 5, 'C': 5},

'B': {'A': 5, 'C': 4, 'D': 3},

'C': {'A': 5, 'B': 4, 'D': 7, 'E': 7, 'H': 8},

'D': {'B': 3, 'C': 7, 'H': 11,'K':16, 'L': 13, 'M': 14},

'E': {'C': 7, 'F': 4, 'H': 5},

'F': {'E': 4, 'G': 9},

'G': {'F': 9, 'N': 12},

'H': {'C': 8, 'D': 11, 'E': 5, 'I': 3},

'I': {'H': 3, 'J': 4},

'J': {'I': 4,'N': 3, 'P': 8},

'K': {'N': 7, 'L': 5, 'N': 7, 'D': 16},

'L': {'D': 13, 'K': 5, 'M': 9, 'O': 4},

'M': {'D': 14, 'L': 9, 'O': 5},

'N': {'J': 3, 'K': 7, 'P': 7,'G': 12},

'O': {'L': 4, 'M': 5},

'P': {'K': 4, 'N': 7, 'J': 8}

}

heuristics = {

'A': 16,

'B': 17,

'C': 13,

'D': 16,

'E': 16,

'F': 20,

'G': 17,

'H': 11,

'I': 10,

'J': 8,

'K': 4,

'L': 7,

'M': 10,

'N': 7,

'O': 5,

'P': 0

}

start = 'A'

goal = 'P'

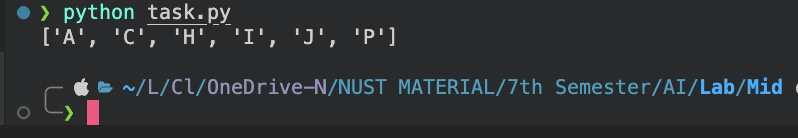
path = a\_star(start, goal, graph, heuristics)

print(path)

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Output:**

****