Batch: A-1 (Honours) **Experiment Number:** 3

Roll Number: 16010421059 **Name:** Chinmay Mhatre

Aim of the Experiment: Implementation of Informed search algorithm- A*

Program/ Steps:

```
from collections import deque
class Graph:

    def__init_(self, adjacency_list):
        self.adjacency_list = adjacency_list

    def get_neighbors(self, v):
        return self.adjacency_list[v]

    def h(self, n):
        H = {
            'A': 1,
            'B': 1,
            'C': 1,
            'D': 1
        }

        return H[n]

    def a_star_algorithm(self, start_node, stop_node):
        open_list = set([start_node])
        closed_list = set([])

        print("Open - ", open_list)

        g = {}
        g[start_node] = 0

        parents = {}
        parents[start_node] = start_node

        while len(open_list) > 0:
```

```
n = None

for v in open_list:
    if n == None or g[v] + self.h(v) < g[n] + self.h(n):
        n = v

if n == None:
    print('Path does not exist!')
    return None

if n == stop_node:
    reconst_path = []

    while parents[n] != n:
        reconst_path.append(n)
        n = parents[n]

    reconst_path.append(start_node)
    reconst_path.reverse()

    print('Path found: {}'.format(reconst_path))
    return reconst_path

for (m, weight) in self.get_neighbors(n):
    if m not in open_list and m not in closed_list:
        open_list.add(m)
        parents[m] = n
        g[m] = g[n] + weight

    else:
        if g[m] > g[n] + weight:
            g[m] = g[n] + weight
            parents[m] = n

            if m in closed_list:
                closed_list.remove(m)
                open_list.add(m)

open_list.remove(n)
closed_list.add(n)
print("\n\n")
print("Open - ", open_list)
print("\nclosed - ", closed_list)
```

```
        print('Path does not exist!')
        return None

adjacency_list = {
    'A': [('B', 1), ('C', 3), ('D', 7)],
    'B': [('A', 4), ('D', 5)],
    'C': [('D', 12)],
    'D': [('A', 7)]
}
graph1 = Graph(adjacency_list)
graph1.a_star_algorithm('A', 'D')
```

Output/Result:

```
Shell
Open -  {'A'}
Open -  {'B', 'C', 'D'}
closed - {'A'}
Open -  {'C', 'D'}
closed - {'B', 'A'}
Open -  {'D'}
closed - {'B', 'C', 'A'}
Path found: ['A', 'B', 'D']
>
```

Outcomes:

CO2: Analyze and formalize the problem (as a state space, graph, etc.) and select the appropriate search method and write the algorithm.

Conclusion (based on the Results and outcomes achieved):

We successfully implemented Informed search algorithm- A* for graph traversal.

References:

1. Stuart Russell and Peter Norvig, Artificial Intelligence: A Modern Approach, Second Edition, Pearson Publication

2. Luger, George F. Artificial Intelligence : Structures and strategies for complex problem solving , 2009 ,6th Edition, Pearson Education

(A Constituent College of Somaiya Vidyavihar University)