SUPERIOR UNIVERSITY

Artificial Intelligence (Lab)

Assignment - 4

Name:

Ali Maqsood.

Roll no:

SU92-BSAIM-F23-050.

Department:

Software Engineering Department.

Program:

Artificial Intelligence.

Section:

BSAI-3A

Question # 1:

A* Algorithm..

Explanation:

This code finds the shortest path from a starting point to a goal in a map using the *A algorithm**. It begins at the start node and keeps track of places to check next in an open_list and places already checked in a closed_list. At each step, it selects the best option based on the total cost, which includes the actual distance traveled (g) and an estimated remaining distance (h). It then updates paths and explores neighbors until it reaches the goal or determines that no path exists. If a path is found, the code traces back the steps and prints the shortest route.

Code:

```
def task1():
    class AStar():
        def __init__(self,graph):
            self.graph=graph


        def neighbors(self,v):
            return self.graph[v]


        def heuristic(self,n):
            heuristic_values={
                "A":1,
                "B":1,
                "C":1,
                "D":1
            }
            return heuristic_values[n]


        def a_star(self,start,goal):
```

```
open_list=[start]
closed_list=[]
g={}
g[start]=0
parents={}
parents[start]=start
while len(open_list)>0:
    n=open_list[0]
    for i in open_list:
        if g[i]+self.heuristic(i)<g[n]+self.heuristic(n):
            n=i
    if n==None:
        print("Path does not exist!!")
        return None
    if n==goal:
        path=[]
        while parents[n]!=n:
            path.append(n)
            n=parents[n]
        path.reverse()
        path.append(start)
        print(f"Path found: {path}.")
        return path
    open_list.remove(n)
    closed_list.append(n)
    for (m,weight) in self.neighbors(n):
        if m not in open_list and m not in closed_list:
```

```
                open_list.append(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.append(m)
            open_list.remove(n)
            closed_list.append(n)
        print("Path does not exist!!")
        return None
    graph={
        "A":[("B",1),("C",3),("D",7)],
        "B":[("D",5)],
        "C":[("D",12)]
    }
    obj1=AStar(graph)
    obj1.a_star("A","D")
task1()
```

Output:

```
Path found: ['A', 'C', 'D']
```