Name: Prasanna R

Reg no:231701039

**A\* SEARCH**

**Aim:**

To implement the A\* Search.

**Program:**

import heapq

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

open\_list = [(0 + heuristic[start], 0, [start])]

closed\_set = set()

while open\_list:

f\_score, g\_score, path = heapq.heappop(open\_list)

current = path[-1]

if current == goal:

return path, g\_score

if current in closed\_set:

continue

closed\_set.add(current)

for neighbor, weight in graph.get(current, {}).items():

if neighbor not in closed\_set:

new\_g\_score = g\_score + weight

new\_f\_score = new\_g\_score + heuristic[neighbor]

new\_path = list(path)

new\_path.append(neighbor)

heapq.heappush(open\_list, (new\_f\_score, new\_g\_score, new\_path))

return None

graph = {

'A': {'B': 2, 'C': 4},

'B': {'C': 1, 'D': 7},

'C': {'D': 3},

'D': {},

'E': {'D': 8},

}

heuristic = {

'A': 5,

'B': 4,

'C': 2,

'D': 0,

'E': 9,

}

start = 'A'

goal = 'D'

result = a\_star(graph, start, goal, heuristic)

if result:

path, cost = result

print(f"Shortest path: {path}")

print(f"Cost: {cost}")

else:

print("No path found.")

graph2 = {

'A': {'B': 4, 'C': 2},

'B': {'D': 5, 'E': 12},

'C': {'E': 10},

'D': {'F': 16},

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

'F': {}

}

heuristic2 = {

'A': 17,

'B': 13,

'C': 14,

'D': 10,

'E': 8,

'F': 0

}

start2 = 'A'

goal2 = 'F'

result2 = a\_star(graph2, start2, goal2, heuristic2)

if result2:

path2, cost2 = result2

print(f"Shortest path: {path2}")

print(f"Cost: {cost2}")

else:

print("No path found.")

**Output:**

Shortest path: ['A', 'B', 'C', 'D']

Cost: 6

Shortest path: ['A', 'C', 'E', 'F']

Cost: 21

**Result:**

Thus the code is executed successfully.