TASK:3

Implementation of A * Algorithm to find the optimal path

Implementation of A * Algorithm to find the optimal path using Python by following constraints.

•The goal of the A* algorithm is to find the shortest path from the starting point to the goal point as fast as possible.

•The full path cost (f) for each node is calculated as the distance to the starting node (g) plus the distance to the goal node (h).

•Distances is calculated as the manhattan distance (taxicab geometry) between nodes.

Tools- Python, Online Simulator - https://graphonline.ru/en/

PROBLEM STATEMENT: CO2 S3

A software developer working on a project to create a GPS navigation system for autonomous vehicles. The system needs to find the optimal path between two locations on a road network to ensure efficient and safe navigation. To achieve this, you decide to implement the A* algorithm, a popular heuristic search algorithm, in Python.

The road network is represented as a graph, where each node represents an intersection, and an edge between two nodes represents a road segment connecting the intersections. Each road segment has a weight or cost, which corresponds to the distance between the intersections.

The task is to implement the A* algorithm to find the optimal path between two specified locations on the road network. The A* algorithm uses a heuristic function that estimates the cost from each node to the goal, guiding the search towards the most promising path while considering the actual cost of reaching each node.

A * ALGORITHM

AIM

To implement the A* algorithm for GPS navigation in Python to find the shortest (optimal) path from a start location to a goal location

ALGORITHM

1. Initialize the open list as a priority queue (min-heap).

- Add the start node with:

f(start) = g(start) + h(start)

g(start) = 0, h(start) from heuristic.

1. Initialize an empty closed set to keep track of visited nodes.
2. Loop until the open list is empty:

a. Remove the node with the lowest f-value from the open list. Let this node be current.

b. If current is the goal node, Reconstruct and return the path and total cost.

c. If current is already in the closed set, Skip and continue to the next node.

d. Add current to the closed set.

e. For each neighbor of current:

i. If neighbor is in the closed set, skip.

ii. Compute g(neighbor) = g(current) + cost(current, neighbor)

iii. Compute f(neighbor) = g(neighbor) + h(neighbor)

iv. Add the neighbor to the open list with its f-value, g-value, and updated path.

1. If open list becomes empty and goal was not reached, No path exists; return failure.

PROGRAM

A* Algorithm for GPS Navigation

```
import heapq

# A* Algorithm Function
def a_star_algorithm(graph, start, goal, heuristic):
    # Priority queue: (f = g + h, g = cost so far, current_node, path)
    open_list = []
    heapq.heappush(open_list, (heuristic[start], 0, start, [start]))
    visited = set()

    while open_list:
        f, g, current, path = heapq.heappop(open_list)

        if current == goal:
            return path, g  # Path and total cost

        if current in visited:
            continue
        visited.add(current)

        for neighbor, cost in graph.get(current, []):
            if neighbor not in visited:
                g_new = g + cost
                f_new = g_new + heuristic[neighbor]
                heapq.heappush(open_list, (f_new, g_new, neighbor, path + [neighbor]))

    return None, float('inf')  # No path found

# ------------------------------
# Main function
```

```
if __name__ == "__main__":
    # Road Network Graph (nodes = intersections, edges = roads with distances)
    graph = {
        'A': [('B', 2), ('C', 4)],
        'B': [('A', 2), ('D', 5), ('E', 10)],
        'C': [('A', 4), ('F', 3)],
        'D': [('B', 5), ('G', 2)],
        'E': [('B', 10), ('G', 6)],
        'F': [('C', 3), ('G', 4)],
        'G': [('D', 2), ('E', 6), ('F', 4), ('H', 1)],
        'H': [('G', 1)]
    }

    # Heuristic values (estimated distance to goal 'H')
    heuristic = {
        'A': 10,
        'B': 8,
        'C': 7,
        'D': 5,
        'E': 6,
        'F': 4,
        'G': 2,
        'H': 0
    }

    # Start and Goal
    start_node = 'A'
    goal_node = 'H'

    # Run A* Algorithm
    optimal_path, total_cost = a_star_algorithm(graph, start_node, goal_node, heuristic)

    # Print Output
```

```python
    if optimal_path:
        print("Optimal Path:", " → ".join(optimal_path))
        print("Total Distance (Cost):", total_cost)
    else:
        print("No path found from", start_node, "to", goal_node)
```

OUTPUT

```
============================================================ RESTART: C:/Users/gvir4/OneDrive/Documents/vtu26455 task3 output.py ============
Optimal Path: A → B → D → G → H
Total Distance (Cost): 10
```

RESULT

Thus the Implementation of A* Algorithm for GPS Navigation using Python was successfully executed and output was verified.