

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:
 $f(\text{start}) = g(\text{start}) + h(\text{start})$
 $g(\text{start}) = 0$, $h(\text{start})$ from heuristic.
2. Initialize an empty closed set to keep track of visited nodes.
3. 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 $g(\text{neighbor}) = g(\text{current}) + \text{cost}(\text{current}, \text{neighbor})$
 - iii. Compute $f(\text{neighbor}) = g(\text{neighbor}) + h(\text{neighbor})$
 - iv. Add the neighbor to the open list with its f-value, g-value, and updated path.
4. If open list becomes empty and goal was not reached, No path exists; return failure.

PROGRAM

A* Algorithm for GPS Navigation

```
import heapq

def a_star(graph, start, goal, heuristic):
    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

    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.get(neighbor, 0)
            heapq.heappush(open_list, (f_new, g_new, neighbor, path + [neighbor]))

    return None, float('inf')

if __name__ == "__main__":
    graph = {
        'A': [('B', 2), ('C', 4)],
        'B': [('A', 2), ('D', 5)],
        'C': [('A', 4), ('D', 1)],
        'D': [('B', 5), ('C', 1)]
    }

    heuristic = {'A': 6, 'B': 4, 'C': 2, 'D': 0}
    start_node = 'A'
    goal_node = 'D'
    optimal_path, total_cost = a_star(graph, start_node, goal_node, heuristic)

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

```

OUTPUT

```
PS C:\Users\karna\AppData\Local\Programs\Microsoft VS Code> & C:\Users\karna\AppData\Local\Programs\Python\Python313\python.exe c:/Downloads/vtu26619.py
Optimal Path: A → C → D
Total Cost: 5
PS C:\Users\karna\AppData\Local\Programs\Microsoft VS Code> & C:\Users\karna\AppData\Local\Programs\Python\Python313\python.exe c:/Downloads/vtu26619.py
Optimal Path: A → C → D
Total Cost: 5
PS C:\Users\karna\AppData\Local\Programs\Microsoft VS Code> █
```

RESULT

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