Name: Om Vinayak Gaikwad WPI ID: 390281097

RBE 550: A* and Dijkstra Algorithm Implementation

Dijkstra Algorithm:

This is another graph search algorithm just like BFS and DFS but has differences in it. The Dijkstra algorithm helps in finding the "shortest distance" from the source node to all other nodes in a weighted graph. The algorithm uses a Priority Queue data structure to store the current distance and position of the point in the graph. A priority queue keeps all the unvisited nodes and pops the node with the shortest distance.

Step-by-Step implementation of the Dijkstra algorithm:

1. First, we initialize the parent matrix to all zeros, set the visited matrix to FALSE, and create a dist matrix to initialize the distance values to infinity as they are unknown.
2. Now, we create a Priority Queue: I used the heap data structure and the following functions: heappop and heappush.
3. Now pushed the initial distance and start point into the heap and started the while loop until the heap is empty.
4. I considered the weight as 1 to travel to each node, this is a uniformly weighted graph.
5. Now, explore the neighbours in the given order "right, down, left, up" and calculating the distance by adding in the weight (1).
6. Finally, comparing the distance with initialized dist matrix (infinity) and pushing the value with the new x and y position, and continue the loop.

Dijkstra Pseudocode:

- Initialise visited, parent, and dist matrix as Boolean FALSE, 0s, and inf respectively.
- while heap:
    push the initial d,x,y value in heap
    if not visited:
        mark visited
        increment steps

    Explore neigbhours:

        for dx,dy in [(0,1),(1,0),(0,-1),(-1,0)]:
            new_post (x',y') = x+dx, y+dy
            check for grid length, and wall condition:
                update parent
                new = d+1
                if new<dist[x'][y']:
                    dist[x'][y']=new
                    push in heap (new,x',y')

A* Algorithm:

This is another graph search algorithm just like Dijkstra which is used to find the shortest path between the nodes with the addition of **"Heuristics"** to enhance the search process. In order to determine which nodes to visit next, A* uses this heuristic function to assess the cost of achieving the goal.

Step-by-Step implementation of the A* algorithm:

1. First, we initialize the parent matrix to all zeros, set the visited matrix to FALSE, and create a dist matrix to initialize the distance values to infinity as they are unknown.
2. Now, we create a Priority Queue: I used the heap data structure and the following functions: heappop and heappush.
3. Now pushed the initial distance and start point into the heap and started the while loop until the heap is empty.
4. I considered the weight as 1 to travel to each node, this is a uniformly weighted graph.
5. Now, explore the neighbours in the given order "right, down, left, up" and calculating the distance by adding in the weight (1).
6. Added Heuristic Function as the Manhattan distance from the current position to the goal.
7. Finally, comparing the distance with initialized dist matrix (infinity) and pushing the value with the new x and y position, and finally, add the heuristic to the get the resultant f score.

A* Pseudocode:

- Initialise visited, parent, and dist matrix as Boolean FALSE, 0s, and inf respectively.
- while heap:
    push the initial d,x,y value in heap
    if not visited:
        mark visited
        increment steps

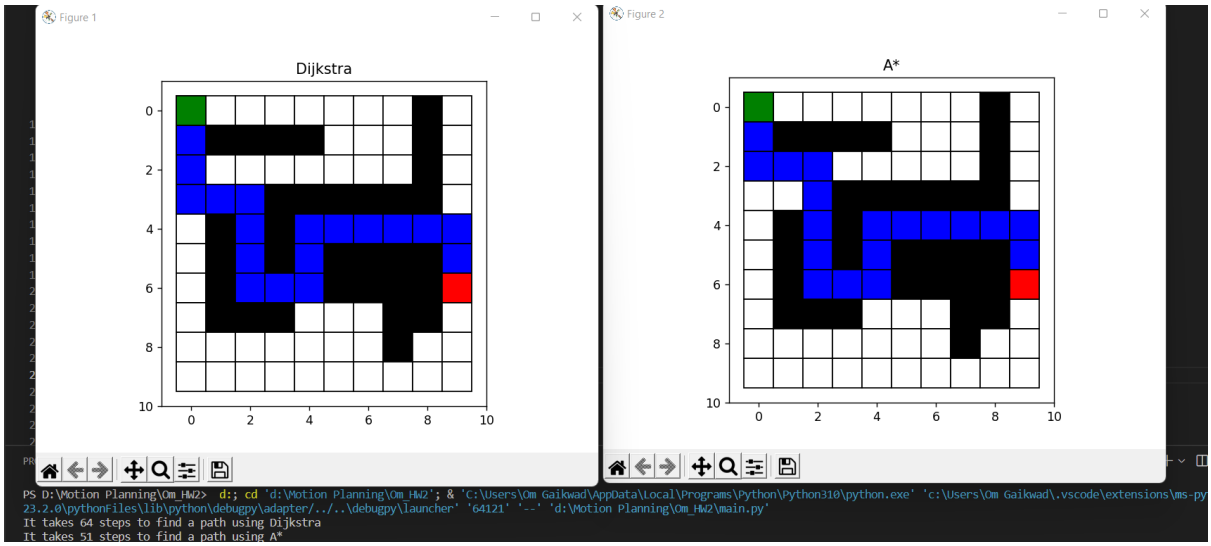
    Explore neigbhours:

        for dx,dy in [(0,1),(1,0),(0,-1),(-1,0)]:
            new_post (x',y') = x+dx, y+dy
            heuristic = Manhattan distance of the x',y' from the goal
            check for grid length, and wall condition:
                update parent
                new = d+1
                if new<dist[x'][y']:
                    dist[x'][y']=new
                    f = new+heuristic
                    push in heap (new,x',y')

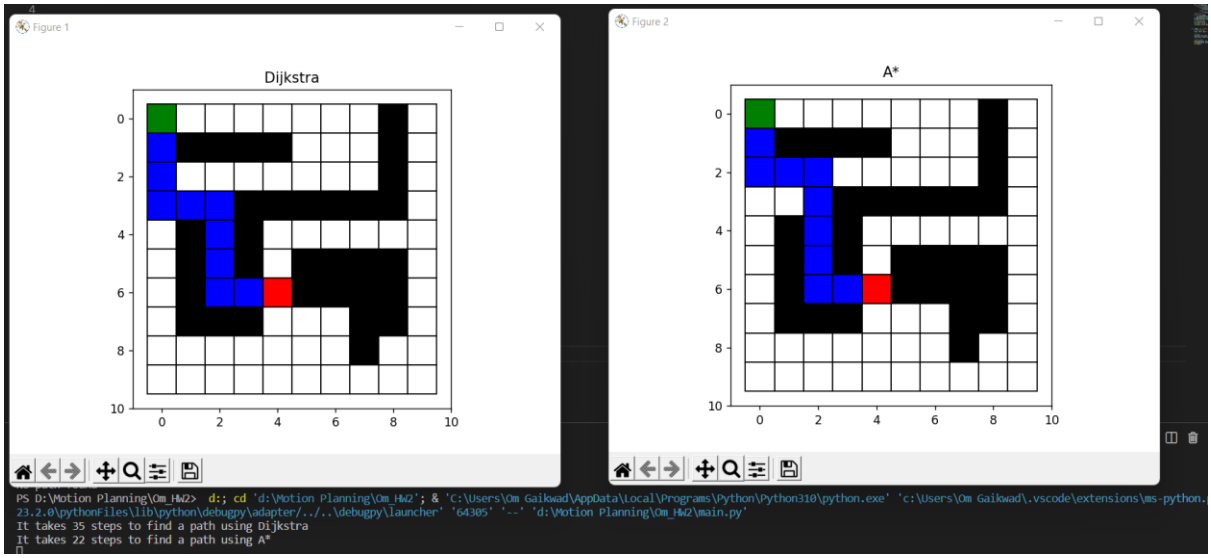
Name: Om Vinayak Gaikwad WPI ID: 390281097

Results:

Goal: 6,9

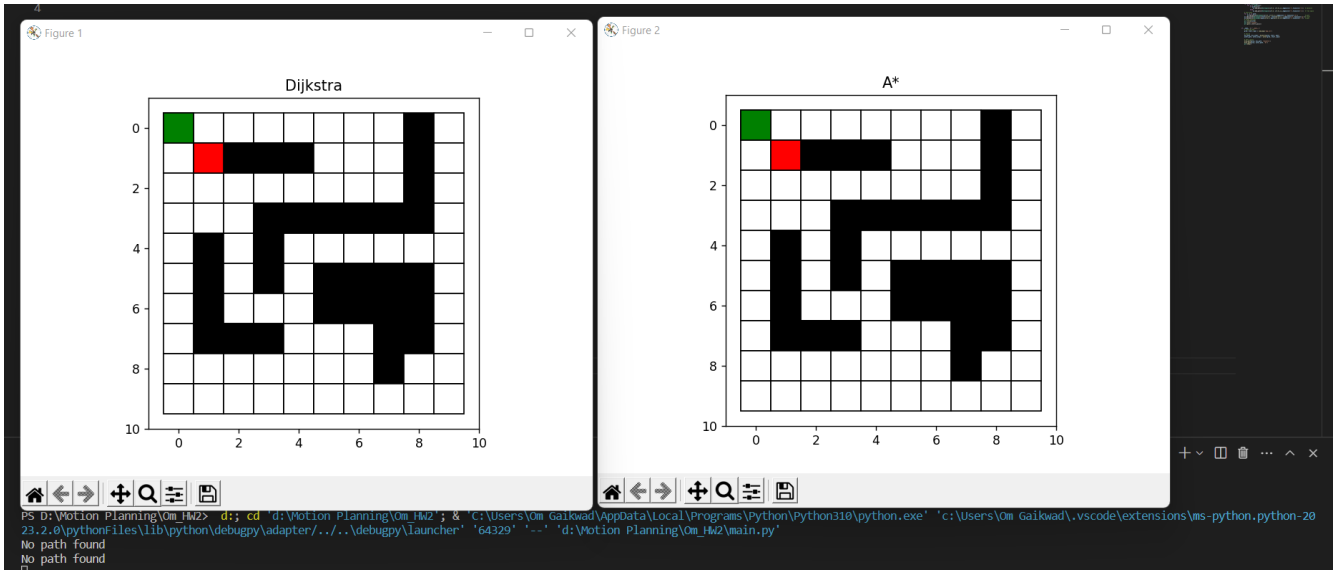


Goal 6,4:



Goal 1,1 (edge case/obstacle):

Name: Om Vinayak Gaikwad WPI ID: 390281097



References:

1. https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/
2. https://www.youtube.com/watch?v=XB4MIexjvY0
3. https://everythingcomputerscience.com/algorithms/Dijkstras_Algorithm.html
4. https://www.geeksforgeeks.org/a-search-algorithm/
5. https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/a-star-algorithm