UNIVERSITY OF WESTMINSTER

INFORMATICS INSTITUTE OF TECHNOLOGY

Informatics Institute of Technology

Department of Computing

BEng (Hons) Software Engineering

Algorithms: Theory, Design, and Implementation

5SENG003C

Coursework

Name - Shashank Pratheepkumar
UoW ID - w1953148

IIT ID - 20220191

a) A short explanation of your choice of data structure and algorithm.

Choice of Algorithm:

The A* algorithm is used for its effectiveness on finding the shortest path in the weighted graph and maps. It prioritizes nodes based on the heuristic function that predicts the cost to reach the goal from the start node. This method of approach tells guides the search towards the goal directly than Dijkstra's algorithm. It ensures optimal performance with the time complexity but also finds the shortest path possible.

The data structures that have been used:

1. HashMap

g_cost: Tracks the cost from the start node to any given node.

h_cost: Stores the estimated cost from any node to the goal.

2. PriorityQueue

It maintains the node to be explored and prioritized by their potential to guide the destination quickly and also with the lowest cost f_cost, it always get explored first and optimizes the search.

3. HashSet

It tracks the visited nodes to prevent the processing again.

b) A run of the algorithm on a small benchmark example: puzzle_10.txt

The algorithm works well and it finds the shortest path for the benchmark puzzle_10 and also other benchmarks as well. The output of the execution of the puzzle_10

1. Start at (2, 8)

2. Move up to (2, 6)

3. Move right to (3, 6)

4. Move down to (3, 10)

5. Done!

c) A performance analysis of the algorithmic design and implementation

1. Algorithmic Complexity

Time Complexity: If every node is visited, the complexity is calculated by the number of nodes N in the grid and the operations per node. So the worst-case time complexity for this algorithm is O(N Log N), where N represents all nodes in the grid. It keeps each node potentially being added to and removed from the priority queue once.

2. Empirical Analysis

In conducting empirical analysis, the doubling hypothesis involves calculating the executing time for larger grids and recording the time growth.

Benchmark Grid Size	Time (ms)
10 x 10	10
20 x 20	12
40 x 40	20
80 x 80	31
160 x 160	35
320 x 320	40
640 x 640	75
1280 x 1280	101
2560 x 2560	378

The above time complexity table matches with the O (N Log N) complexity, which shows the logarithmical scale with the number of nodes because of the use of a priority queue.

Conclusion

The A* algorithm is highly effective for pathfinding in typical use cases. It uses heuristics to cut down the search space. The empirical analyses provide a framework for understanding and predicting the algorithm's behavior in many scenarios.