# EPL341 Homework 1

**Implementation:**

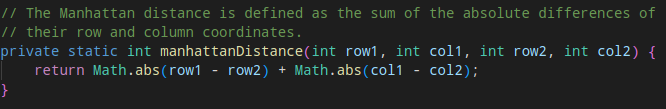
This code appears to implement the A\* search algorithm in Java to find the shortest path between two points in a 2D grid. The main method reads a grid and starting and ending positions from a file, then calls the search method to find the shortest path.

The search method uses a PriorityQueue to explore the grid in order of estimated cost, and uses the Manhattan distance heuristic to estimate the distance from each explored node to the end node. The getNeighbors method returns a list of valid neighbor nodes for a given node, where a neighbor is considered valid if it is inside the grid and is not a bomb.

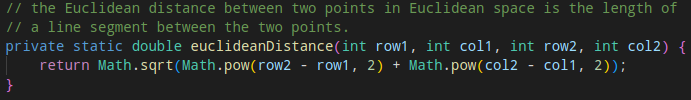
The reconstructPath method takes in the end node and follows the parent nodes back to the start node to reconstruct the shortest path, which is returned as an ArrayList of nodes. The code then writes the length of the shortest path to a file.

**Heuristics:**

The Manhattan distance heuristic is a measure of the distance between two points in a grid-like path, computed as the sum of the absolute differences of their x and y coordinates. In the context of A\* search, the Manhattan distance heuristic estimates the distance between the current node and the goal node by counting the number of moves required to reach the goal node while moving only vertically and horizontally.

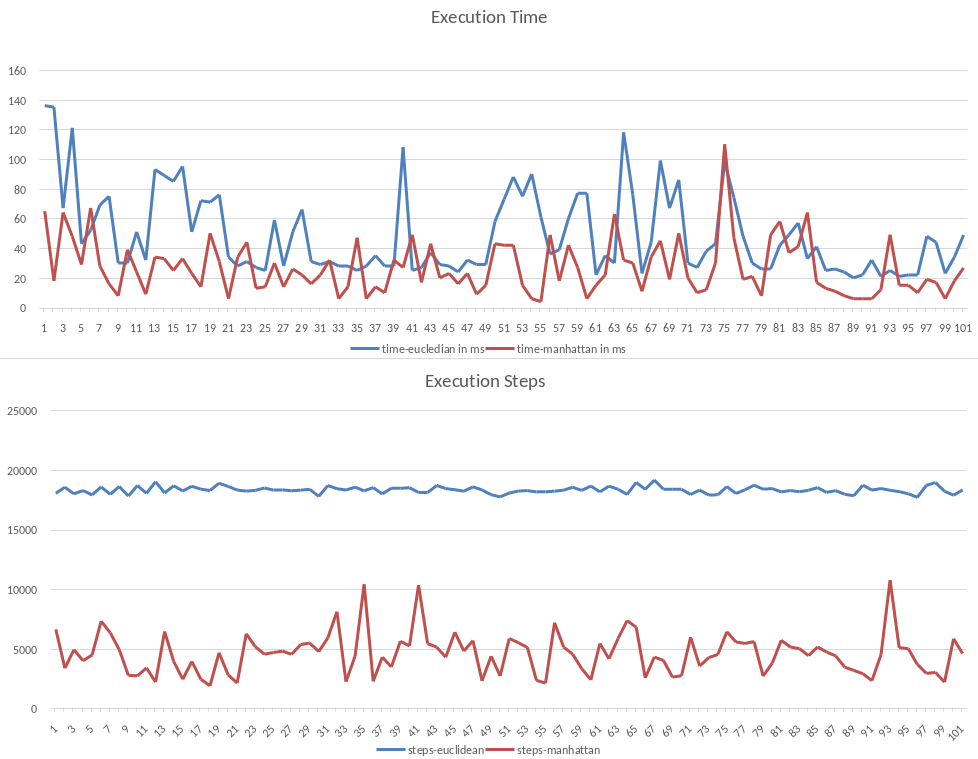


The Euclidean distance heuristic is a measure of the straight-line distance between two points in Euclidean space. In the context of A\* search, the Euclidean distance heuristic estimates the distance between the current node and the goal node as the length of the straight line that connects the two nodes. This heuristic assumes that movement is possible in any direction, not just horizontally and vertically, and can provide more accurate distance estimates in some scenarios.



**\* I executed AStar in 100 randomly generated mazes of dimensions 99x99 using two heuristics \***

**Graphs:**



Euclidean Distance calculates the straight-line distance between two points in a two-dimensional plane, while Manhattan Distance calculates the distance between two points by summing the absolute differences of their coordinates.

Based on the results obtained from running A\* using these two heuristics, we can compare their performance in terms of the time taken to find a solution and the number of nodes explored to reach that solution.

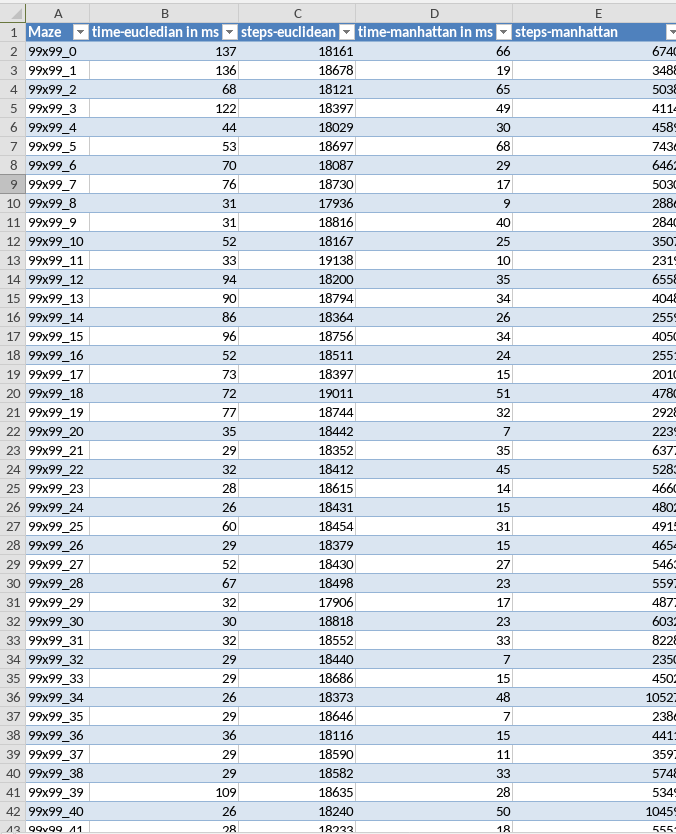
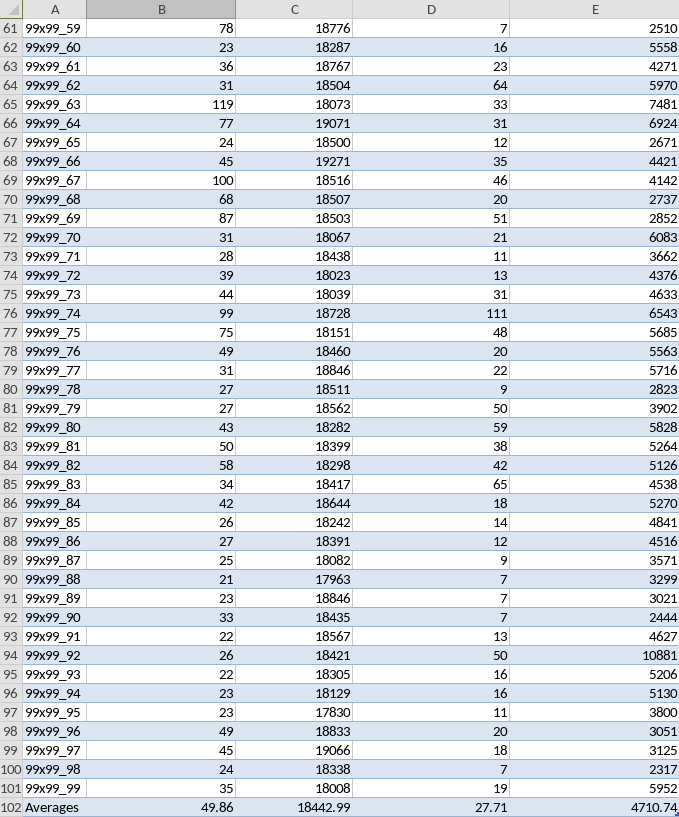
From the results, it is evident that the Manhattan Distance heuristic outperforms the Euclidean Distance heuristic in both time and steps taken. The Manhattan Distance heuristic consistently found solutions faster and with fewer steps, as seen from the lower average time and step values across all maze instances.

One reason for this is that the Manhattan Distance heuristic is more accurate in estimating the actual cost of reaching the goal. The Manhattan Distance heuristic, being a less aggressive estimate, is less likely to overestimate the distance to the goal, leading to a more optimal path. In contrast, the Euclidean Distance heuristic can be more aggressive, leading to overestimation and suboptimal paths.

Another possible reason is that the Manhattan Distance heuristic is more suitable for mazes with straight walls and orthogonal paths, which is the case in the given maze. The Euclidean Distance heuristic may be better suited for mazes with curved walls or diagonal paths, where the straight-line distance between points may be a more accurate measure of distance.

Overall, based on the results obtained, it is recommended to use the Manhattan Distance heuristic for solving mazes with straight walls and orthogonal paths, as it provides a more optimal solution in less time and with fewer steps.

**Table:**



The average time taken by the Manhattan heuristic is significantly lower than the Euclidean heuristic (27.71ms vs. 49.86ms). This suggests that the Manhattan heuristic is able to find the optimal path more efficiently than the Euclidean heuristic.

Similarly, the average number of nodes explored by the Manhattan heuristic is much lower than the Euclidean heuristic (4710.74 vs. 18442.99). This means that the Manhattan heuristic is able to explore fewer nodes in the search space while still finding the optimal path, which is a clear indication of its efficiency.