Mini Assignment 4
Search

Anshul Mahajan

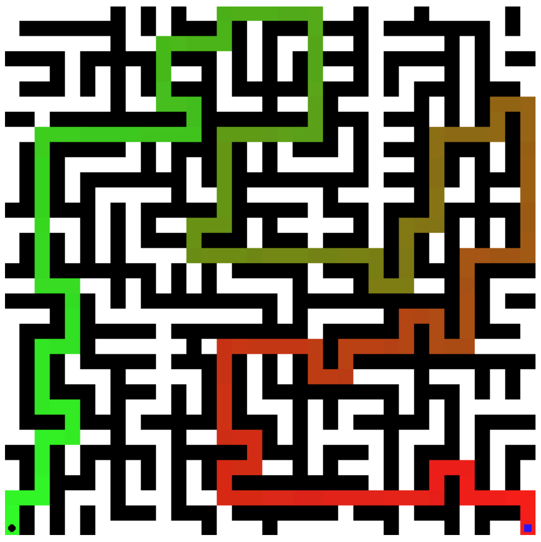
Objective

This assignment aims to implement search algorithms and solve path-finding problems.

Part 1: Finding a single dot

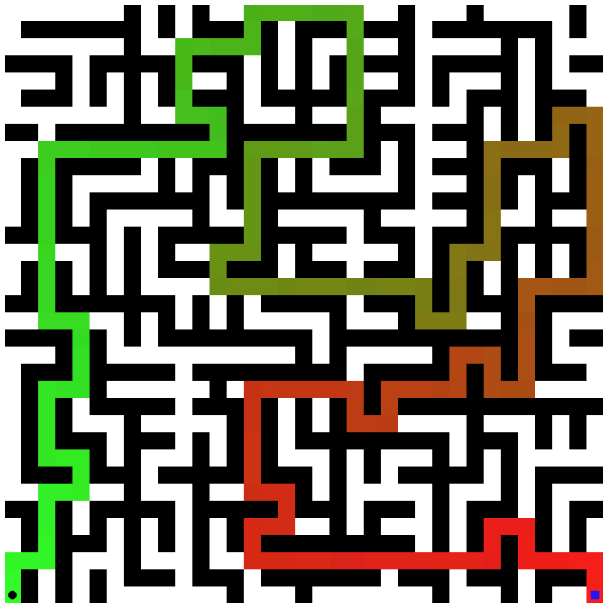
In this part, the task is to find the shortest path from the starting point to a single objective in the maze using different search algorithms.

- Breadth-First Search
  A queue data structure (FIFO) is used to explore the nodes.
  The algorithm starts at the initial position and explores all neighbouring nodes level by level.
  The first time the objective is found, the path is returned.



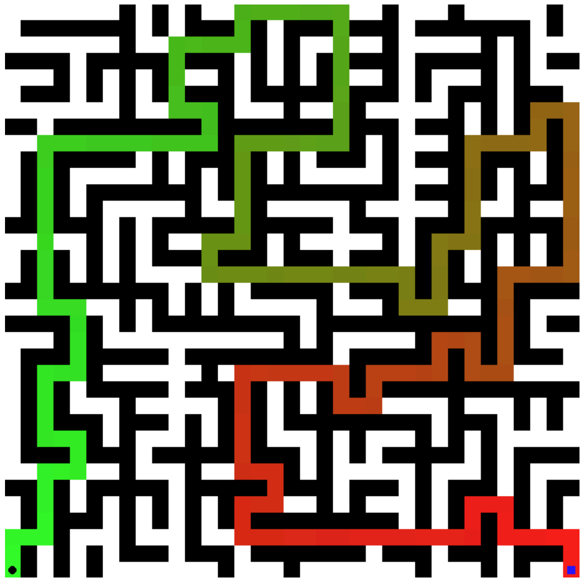
Path Length: 211
States Explored: 619
Total time 0.0008320808410644531 seconds

- Depth-First Search
  A stack data structure (LIFO) is used to explore nodes.
  DFS dives deeper into the maze by exploring a single branch to its fullest before backtracking.
  The first path to the objective is returned, but it does not guarantee the shortest path.



Path Length: 211
States Explored: 426
Total time 0.0006968975067138672 seconds

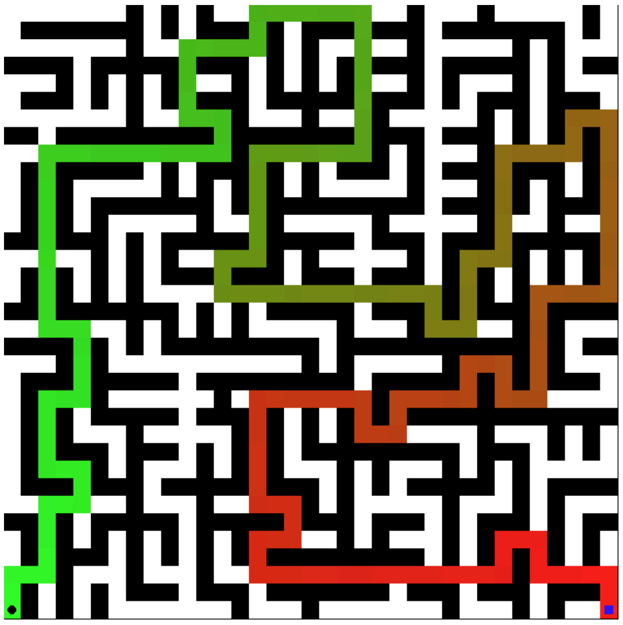
- Uniform-Cost Search
  UCS uses a priority queue where nodes are expanded based on their path cost.
  The algorithm guarantees finding the optimal path by always expanding the least-cost node first.
  In this maze, since all movements have the same cost, **UCS behaves similarly to BFS**.



Path Length: 211
States Explored: 619
Total time 0.0008127689361572266 seconds

- A* Search

A* search combines both UCS and a heuristic to guide the search toward the objective.
The heuristic used is the **Manhattan distance**, which provides an estimate of how far a node is from the objective.
A* prioritizes nodes that minimize the total cost (path cost + heuristic).



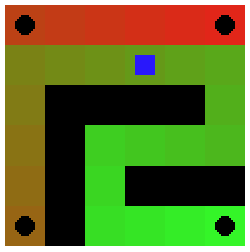
Path Length: 211
States Explored: 549
Total time 0.0008351802825927734 seconds

All algorithms return consistent paths in this specific map. However, more testing suggests that this is a coincidence.

Part 2: Finding all corners

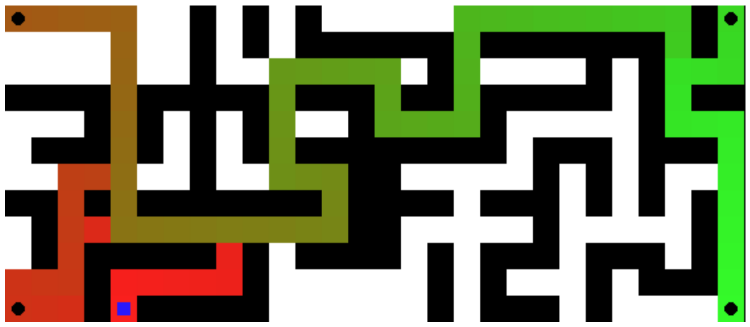
In this part, the objective is to find the shortest path through all four corners of the maze. The challenge is to visit all the corners efficiently.
A modified A* search is implemented to handle multiple objectives.
The search iteratively finds the shortest path to the closest corner using the Manhattan distance as a heuristic.
After reaching a corner, the algorithm continues from that point to find the next closest corner.

tinyCorners.txt



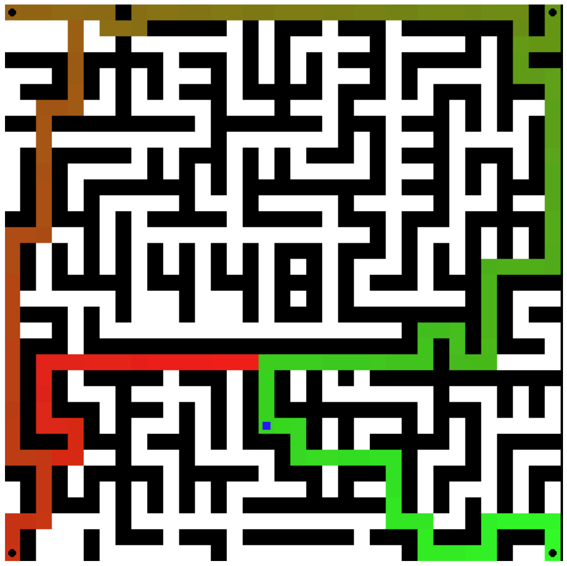
Path Length: 33
States Explored: 46
Total time 0.00010418891906738281 seconds

mediumCorners.txt



Path Length: 108
States Explored: 728
Total time 0.00180292129 seconds

bigCorners.txt



Path Length: 205
States Explored: 779
Total time 0.0014152526855546875 seconds

While this approach guarantees visiting all corners, it does not always guarantee finding the shortest overall path through all corners due to the greedy nature of the heuristic i.e. choosing the closest corner at each step.

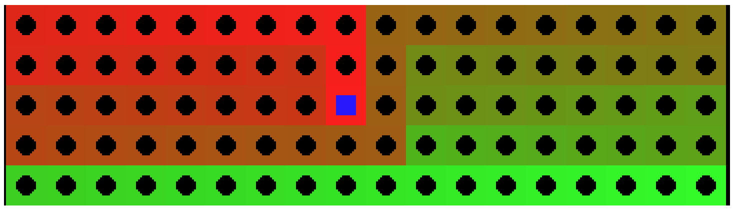
Part 3: Finding multiple dots

In this part, the problem is extended to finding the shortest path through multiple dots scattered across the maze. This is a more challenging problem since finding the exact optimal path can be computationally expensive.

A* search is extended to find paths through multiple objectives.
The algorithm uses the same approach as in part 2, where it repeatedly finds the closest objective using the Manhattan distance.
After reaching an objective, the search continues to the next closest objective until all objectives are visited.

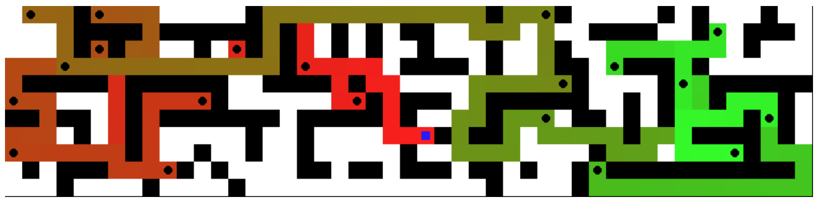
In a search for a different heuristic function, I stumbled upon the Minimum Spanning Tree (MST) algorithm. However, I was unable to implement it myself and hence, it is not submitted.

openSearch.txt



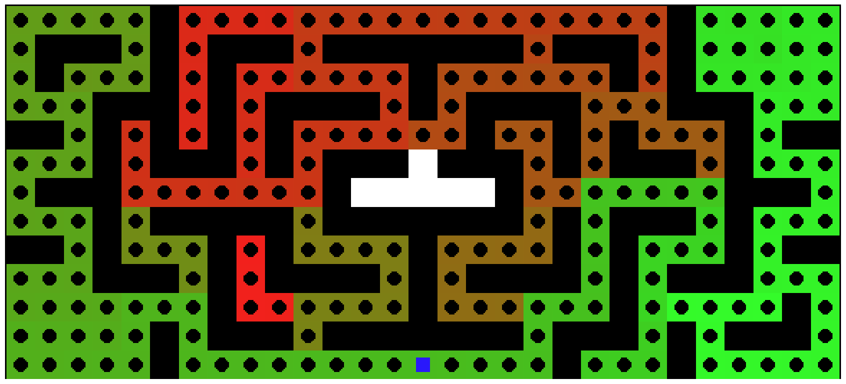
Path Length: 100
States Explored: 99
Total time 0.0006470680236816406 seconds

mediumSearch.txt



Path Length: 282
States Explored: 2618
Total time 0.008559703826904297 seconds

bigSearch.txt



Path Length: 395
States Explored: 732
Total time 0.0034191608428955078 seconds

The Manhattan heuristic may not be optimal for multiple objectives, and a more sophisticated heuristic (such as the Minimum Spanning Tree heuristic) could further improve efficiency. While this approach may not always find the globally optimal path, it provides a reasonable solution in a practical amount of time.