Snake Maze Puzzle

Team Members	Roll Numbers
Moazzam Umer	21L-6210
Abdullah Umar	21L-5604
Muhammad Arham	21L-5699

Strategies Implemented:

- A* Search
- Greedy Best First Search
- Depth First Search
- Breadth First Search

Testing Guide:

Navigate to main.py, on line 49. Change the search function to test different algorithms. Available functions are:

1. self.AgentSnake.SearchSolutionBFS(self.State)
2. self.AgentSnake.SearchSolutionDFS(self.State)
3. self.AgentSnake.SearchSolutionGBFS(self.State)
4. self.AgentSnake.SearchSolutionAstar(self.State)

Algorithms Guide:

1- A* Search:

The SearchSolutionAstar function initializes the algorithm by setting up the initial state and a priority queue to store nodes to be explored, where each node contains the total cost (sum of step cost and heuristic), the heuristic value alone, current position, and path taken. The astar function iteratively explores nodes until the priority queue is empty, popping the node with the minimum total cost. If

the current position is the food, the function returns the path. Otherwise, it explores neighboring positions, calculates their costs, and adds valid positions to the heap for further exploration. The tie-breaker ensures proper ordering of nodes with the same total cost. This implementation efficiently searches for the shortest path by considering both total cost and heuristic estimation.

2- Greedy Best First Search

The SearchSolutionGBFS function initializes the algorithm by setting up the initial state and a priority queue to store nodes to be explored, where each node contains the heuristic value, current position, and path taken. The gbfs function iteratively explores nodes until the priority queue is empty, popping the node with the minimum heuristic value. If the current position is the food, the function returns the path. Otherwise, it explores neighboring positions, calculates their heuristic values, and adds valid positions to the heap for further exploration. The tie-breaker ensures proper ordering of nodes with the same heuristic value. This implementation efficiently searches for the shortest path by prioritizing nodes based on heuristic estimation.

3- Depth First Search

The SearchSolutionDFS function initializes the DFS by setting up the initial state and invoking the helper DFS function with the snake's head position, food position, and an empty set to store visited positions. The dfs function recursively explores neighboring positions starting from the current position until it finds the food or exhausts all possible paths. It checks if the current position is valid and not visited, adds it to the visited set, and recursively explores its neighboring positions. If the food is found, the function returns the path taken. Otherwise, it continues exploring other directions. This DFS approach systematically explores the maze until it finds a path to the food, prioritizing depth over breadth.

4- Breadth First Search

The SearchSolutionBFS function initializes the BFS by setting up the initial state and invoking the helper BFS function with the snake's head position, food position, an empty queue to store nodes to be explored, and an empty set to

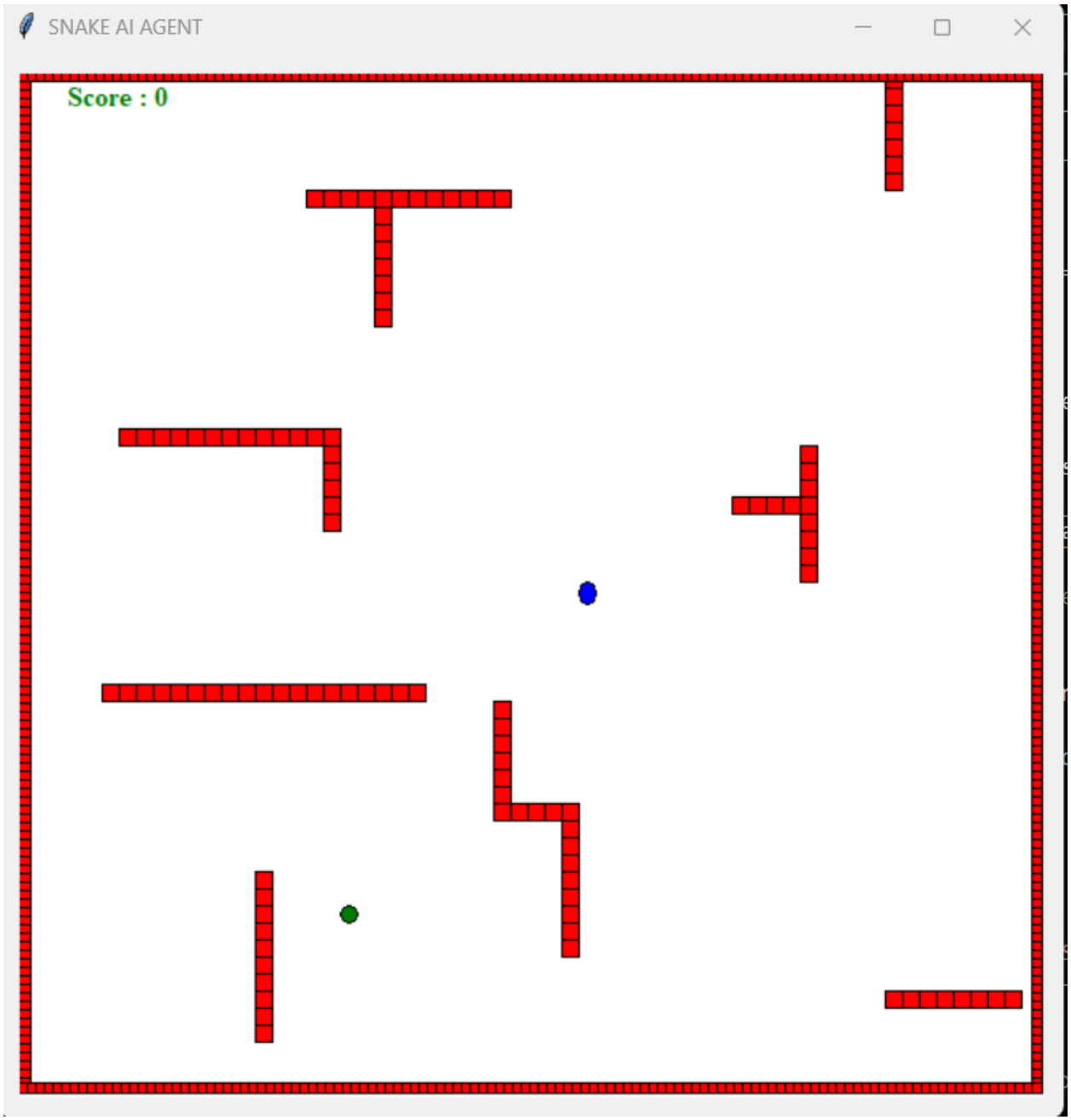
store visited positions. The bfs function explores neighboring positions level by level, starting from the snake's head position and moving outward. It dequeues nodes from the front of the queue, checks if the current position is the food, and explores its neighboring positions. If a valid neighboring position is found, it adds it to the queue and marks it as visited to avoid revisiting. This BFS approach systematically explores the maze level by level until it finds a path to the food, ensuring the shortest path is found.

Results:

All the four implemented algorithms are run for 1 min to test their scores and performance after this time. The maze text file that is used is **Maze1.txt.** It contains some extra walls for better testing. Text file is submitted along with the code files. Below are the results after 1 min of running snake on Maze1.txt . Wall conditions are handled so no hit the wall error occurred.

Algorithm	Score
A* Search	220
GBFS	290
DFS	10
BFS	200

Screenshot of Maze1.txt used for testing


SNAKE AI AGENT
Score : 0