Lab A Design

Vassily Lombard - Charles Bowen-Rayer

February 2024

1 General Design

We represent the environment using two-dimensional arrays. It is a convenient way to keep track of the agent's location in the maze as well as giving it instructions to move. The transition model takes advantage of this environment representation by simply modifying the agent's row/column location depending on the action performed. It returns all the different possible moves, the ones that avoid walls. Therefore, in this design the actions are performed by the transition model, going North, South, East or West. We chose to return a list of possible moves as it will allow these moves to be added straight into the respective data structure for the search algorithm.

2 Part 2 Design

In part 2, we will implement a Depth First Search algorithm to find the shortest path to the prize location. Using a stack data structure (First In, Last Out), we will keep track of the nodes visited. The general approach of this algorithm is a traversal approach in which the traverse begins at the root node and proceeds through the nodes as far as possible until it reaches a dead end. In practice, the visited nodes are added to the stack and removed when there are no more nodes to visit.

3 Part 3 Design

In part 3, we will implement three algorithms, the Breadth First search, the Greedy best-first search, and A* search, which all of them will search for mazes with one prize. Concerning the Breadth First search, the data structure used is a queue (First In, First Out). This queue will allow the agent to explore all the nodes on the same level before exploring the next level. In other words, the agent explores the possible moves around it, deleting the nodes explored several times from the queue. The Greedy best-first search and A* search are two algorithms selecting the next node to explore based on heuristic cost. In the context of finding the shortest path between two points in a maze, the heuristic cost is simply the Manhattan distance from the starting point to the prize. In the case of the Greedy best-first search, each possible path is evaluated, and the agent explores the node with the lowest heuristic cost (the one closest to the goal, the prize). This process is repeated until the prize is reached. The A* search algorithms pick the node according to two parameters, the actual cost of reaching the current node from the start node and the heuristic estimate of the cost from the current node and the prize node.

4 Part 4 Design

In part 4, we will implement the A* search algorithm in the context of finding multiple prizes in a maze. Similarly to part 3, the algorithm will use Manhattan distance as a heuristic function, but its general functioning will slightly differ. It will have to consider all the prizes as a whole goal (it avoids selecting the nearest prize, does A* search for it, and then select the next nearest prize). We will take into account the list of all the prize locations in a different order and then determine the path generating the best performance. Swapping prizes in order in the list might lead to a performance change.