Stage#1:

1) To apply A* algorithm to the waiter agent problem, we would need to have the following information and make the following assumptions:

1. The layout of the restaurant: We need to know the physical layout of the restaurant in terms of the position of tables, chairs, walls, and other obstacles. This information is necessary to construct the search space and determine the distance between different positions.

2. The positions of the customers and their orders: We need to know the position of each customer and the order they have placed. This information is necessary to determine the path that the waiter should take to serve each customer and deliver their order.

3. The position of the waiter: We need to know the current position of the waiter at each step of the algorithm. This information is necessary to determine the distance between the waiter and the nearest unserved customer.

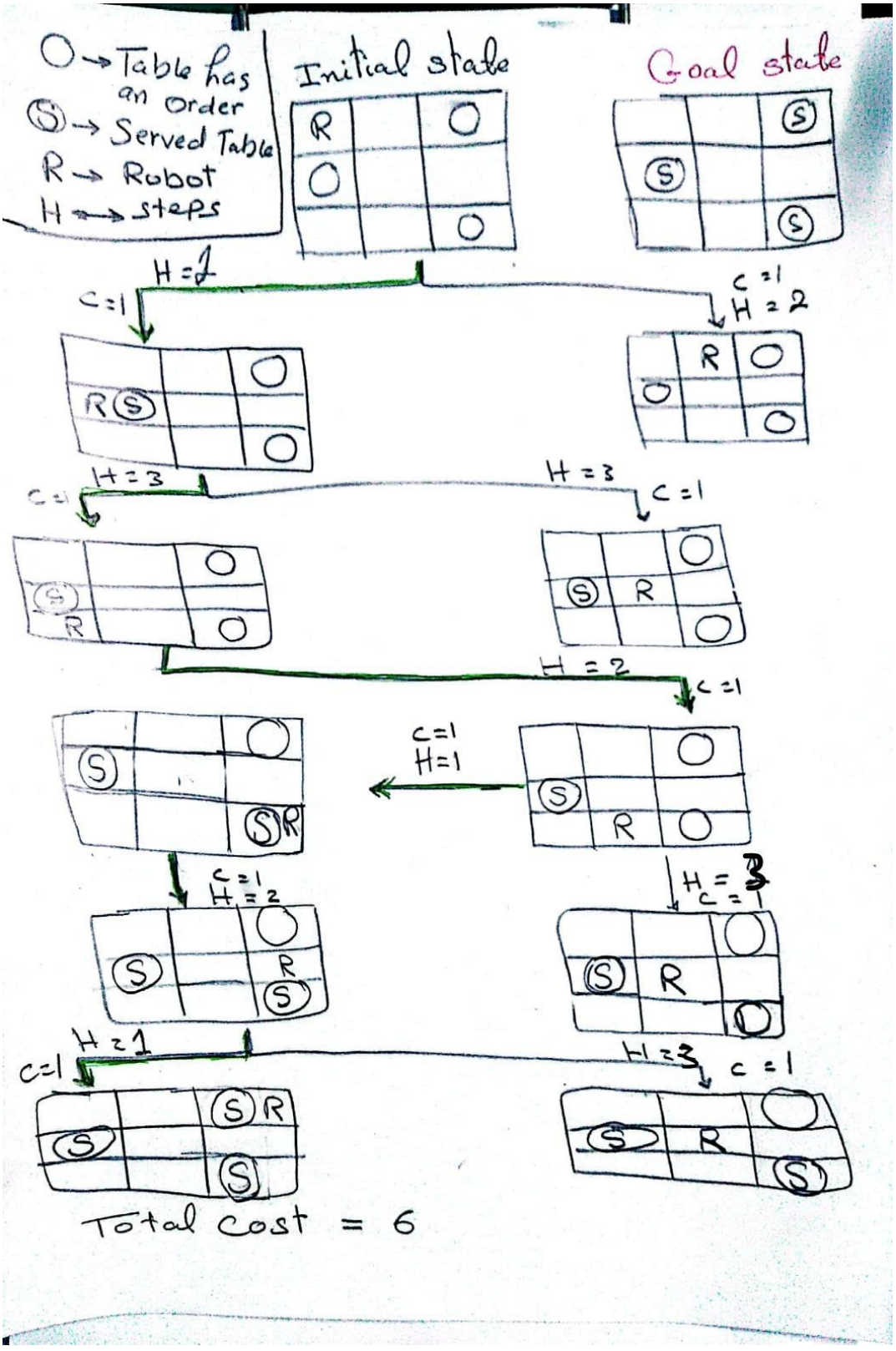
4. The assumption that the waiter moves horizontally or vertically: To calculate the Manhattan distance heuristic function, we assume that the waiter can only move horizontally or vertically, and not diagonally. This assumption is necessary to ensure that the heuristic function is admissible, meaning it never overestimates the actual cost to reach the goal state.

5. The assumption that the order of service does not matter: We assume that the order in which the customers are served and their orders are delivered does not matter. This assumption simplifies the

problem and allows us to focus on finding the shortest path that serves all customers and delivers all orders.

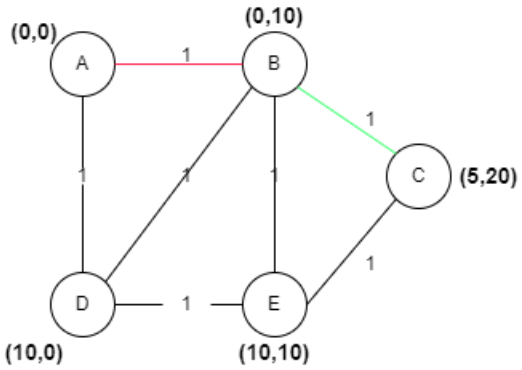
Based on these assumptions, we can calculate the heuristic function for A* algorithm as the Manhattan distance between the waiter and the nearest unserved customer. The Manhattan distance is calculated by adding up the absolute differences in the x and y coordinates between the two positions. This heuristic function is admissible because it never overestimates the actual cost to reach the goal state. We can use this heuristic function along with the cost of each step to estimate the optimal path for the waiter to serve all customers and deliver all orders.

2) Apply A* search algorithm on the state diagram and / or state tree



Stage#2:

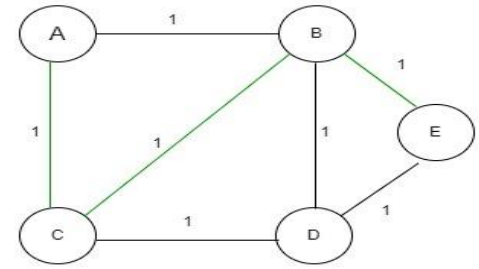
A*

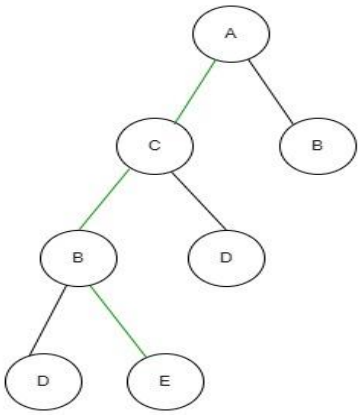


A ---> B --->C
Total Cost = 2

Initial A goal C

DFS





Path: A --> C --> B --> E, Total cost: 3

Initial A goal E

Using Depth-First Search (DFS), the robot would explore one path at a time, visiting the deepest node before backtracking to the next possible path. While DFS may be faster than BFS in some cases, it does not guarantee that it will find the shortest path.

On the other hand, A* algorithm combines the benefits of both BFS and DFS by using a heuristic function to estimate the distance from the

current node to the goal node. A* algorithm expands nodes in the order of their total estimated cost from the start node to the goal node. A* algorithm is guaranteed to find the shortest path as long as the heuristic function is admissible and consistent. In the case of the waiter robot food delivery, A* algorithm would be a better choice than BFS or DFS because it would quickly find the shortest path using the heuristic function that estimates the distance between the robot's current position and the table where the food needs to be delivered.

A* algorithm would be faster than BFS in terms of exploration time and would be more reliable than DFS in finding the shortest path.

So from the above example and according to our result from our code we can demonstrate that A* from our point of view was useful to solve our problem and get best solution.