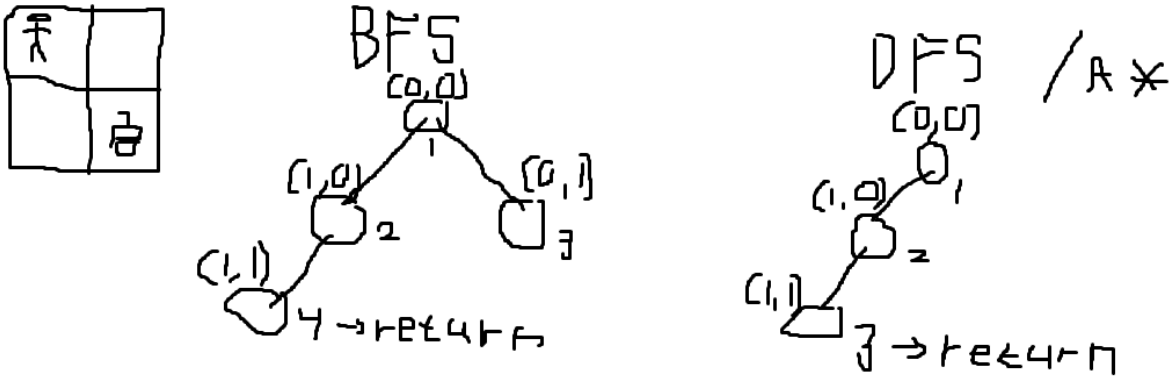
1. The states were position and direction. The actions were turning, moving forward and sucking. The branching factor is 4 at most if there are no walls.
2. No difference since BFS can be derived from uniform cost search with the condition that each action costs 1.
3. A, C
4. The heuristic could be the Manhattan distance to the square. The cost function could be the cost thus far + the heuristic cost. The Manhattan distance is an admissible heuristic since it always underestimates the cost or is equal to the cost. This is because the agent cannot move in a more efficient manner than moving east, west, north or south, i.e., it cannot move diagonally.
5. The BFS algorithm will evenly spread out from the starting position until it finds square (1,1). DFS will follow one path until it hits a dead end, so it will return faster than BFS in this case. A* with the Manhattan distance heuristic will queue squares in order of cost and in this case will be the same search tree as BFS (there is an error in the picture). This is the case in the base implementation of A* at least. The memory usage will be best with the DFS algorithm since it handles fewer branches of the tree at any given time. A* and BFS will have a higher memory usage because they will store all branches under consideration, which is a lot more branches than DFS since it does not deplete branches as DFS does.



6. Best first search is complete when the heuristic is admissible, breadth-first search is complete, uniform-cost search is complete, DFS and iterative deepening search is complete when the search space is finite, bidirectional search is complete if BFS is used for both searches, greedy best-first search is not complete unless the state spaces are finite, A* is complete. Only uniform cost search, bidirectional search (can be with the right conditions) and A* are optimal (with an admissible heuristic). Uniform cost search is optimal when the nodes have uniform path costs. BFS is optimal when nodes have uniform path costs. This is because they do not return a “good enough” path, they only return the shortest path. Iterative deepning search can be optimal too if the path cost does not decrease with depth.
7. We implemented a search algorithm in the first lab.