COMP 341 Homework #1 Report / Barış Aracı 62595

Written Q1: What are some differences between DFS and BFS in terms of path cost and number of expanded nodes? When and why would you prefer BFS over DFS? When and why would you prefer DFS over BFS?

Answer Q1: BFS expands more nodes compared to DFS since it expands from center to outer. But, it gives us shallowest solution. DFS expands less node so it's faster but it may give us a longer path. Also, BFS definitely finds an answer if there is one, but DFS could dive into the infinity if there is no protection such as iterative deepening search. We should prefer DFS if our priority is speed. On the other hand, we should prefer BFS if our priority is more accurate result.

Written Q2: What are some differences between UCS and A* in terms of path cost and number of expanded nodes? When and why would you prefer UCS over A*? When and why would you prefer A* over UCS?

Answer Q2: UCS is an uninformed search. It does not use any pre-knowledge. On the other hand, A* is an informed search. It uses a function called heuristic function to guess the distance to destination. So, A* tries to minimize the number of expanded nodes. If a good heuristic function can be created, then using A* is a better option. If a heuristic function cannot be created, then UCS should be used.

Written Q3: Comment on your choice of state in the four corners problem. Why does it allow you to solve the problem?

Answer Q3: My state is a tuple object. I save the position in the first index and number of visited corners as an array in the second index. I do it to track the status of the corners(visited/unvisited) in successors. Another alternative is keeping the number of unvisited corners in the second index and remove corners one by one as they are explored. One of those options is necessary to know whether the goal is achieved or not.

Written Q4: Comment on your choice of heuristic in the four corners problem. Why did you settle on that heuristic? Why is it admissible and consistent?

Answer Q4: I used manhattan distance as a heuristic function. It calculates the sum of manhattan distances between the Pacman and all of the unvisited corners from the minimum to maximum. Of course, it updates the Pacman's position to the visited corner before calculating the next distance to next corner. Another alternative that I know is the euclidian distance, but it expanded more node when I tried it.

Written Q5: Comment on your choice of heuristic in the eating all the dots problem. Why did you settle on that heuristic? Why is it admissible and consistent?

Answer Q5: I calculated the sum of maze distances (which returns the real path cost by doing search) between Pacman and foods. Then I always returned the max to make the heuristic cost closer to real cost. It is admissible because even if I use the max value, it is a distance to single food and it is definitely less than the real cost since the real cost includes the whole path containing all foods.

Written Q6: What are some practical differences between a consistent and an inadmissible heuristic, in terms of path cost and number of expanded nodes? When and why would you prefer an inadmissible heuristic over a consistent one? When and why would you prefer a consistent heuristic over an inadmissible one?

Answer Q6: If the heuristic function overestimates the cost of reaching the goal, then it is inadmissible, and the solution is not guaranteed to be optimal. If the real cost to the successor + estimated cost from the successor to the goal is less than the estimated cost to the goal, then the heuristic function is consistent. If a heuristic function is consistent, then it is also admissible, and it finds the optimal solution.