Orçun Özdemir

54020

COMP341 Homework 1 Report

1. DFS is faster than BFS, but BFS is more optimal and consistent than DFS. DFS's space complexity is O(bm), time complexity is O(b^m). BFS's space and time complexities are O(b^S). If I want to find the shortest path, I would choose BFS. If I want to find any path that lead to solution but fast, I would choose DFS. Depth First Search chooses using stack as utility which implements the frontier as "Last In First Out" (LIFO) implementation, on the other hand Breadth First Search chooses using queue as utility which implements the frontier as "First In First Out" (FIFO) implementation.

2. A* explores less than UCS, however UCS has more volume since UCS explores more than A* search. Therefore, UCS needs more space and its space complexity is bigger than A* search. UFC expands equally in all directions, on the other hand A* expands through the goal generally. A* search focuses on expanding the most nodes which are low cost and idealistic. The most important trick for handling an A* search is finding a proper heuristic.

5. I used foodGrid state as current location of the node and appended the location with position state. Then, I created a variable called furthest_distance which is initially zero. Then I found the distance in between two points with mathematical formula and set it to a variable called distance. Then, if the distance is longer than furthest_distance, distance becomes the furthest_distance vice versa. Creating an furthest_distance, relatively relaxed my problem, therefore I believe that the problem is relaxed one. Subsequently, being a relaxed problem, handles this kind of problem led to my A* search is admissible and consistent.

6. We can prefer consistent heuristic over an inadmissible one when, the targets are relaxed problems which can be easily add new actions. Admissible heuristics are quite beneficial for relaxed problems. We can prefer inadmissible heuristics, when the speed is concern and failing constantly until finding the path was not an issue. When heuristics reach closer to the real cost, we need to expand fewer nodes, however we do more work for each node to compute heuristic itself.