Saagar Sanghavi, Jin Seok Won, Kevin Zhu

Project T Final

State Spaces and Search Quiz Questions

Question 1: Recall in the coding assignment, the robot could only move one joint at a time, at a specific amount, 30 degrees. If the angle the robot could move at one time is changed to 15 degrees, how would this affect the size of the state space?

Answer: The state space will get 2x bigger. Now instead of 12 possible angles, there are 24 angles of 15 degree increments from 0 to 360 degrees.

Question 2: What is the state space size of a baseball game when you only consider the runners in each of the bases, number of outs, balls, and strikes?

Answer: 288.

From bases - 3 bases empty or not empty -> 2^3 = 8

From number of outs - 0,1,2 -> 3

From number of balls - 0, 1, 2, 3 -> 4

From number of strikes - 0, 1, 2 -> 3

*Multiply them all to get 8 * 3 * 4 * 3 = 288*

Question 3: What is the runtime complexity of DFS and what is the worst case scenario in terms of runtime?

Answer: The worst case runtime of DFS happens when the first goal state it encounters is on the right most branch, at the deepest level. In this situation, the runtime would be $O(b^m)$

Question 4: Suppose we knew the structure of the state space and all of the costs were equal. When would we use DFS versus BFS?

Answer: DFS is better for a "dense" graph, which is a graph with a higher node to edge ratio. BFS is better for a "sparse" graph, which is a graph with a higher edge to node ratio.

Question 5: Suppose your search graph has negative edge costs. Is UCS optimal in this case? Explain.

Answer: No. Consider a graph with vertices A, B, C and edges A -> B with weight 3, B -> C with weight -2 and A -> C with weight 2. UCS will return A->C, however the optimal path is A -> B -> C. Intuitively, our invariant of expanding our lowest cost space at every iteration is broken.

Question 6: The fringe of UCS is usually implemented with which type of data structure?

a. Priority queue, ranked by backward cost
b. Priority queue, ranked by heuristic
c. Stack
d. Hashmap

Answer: a. This is true by construction of UCS. By having a priority queue ranked by backward cost, we explore nodes in order of increasing path cost, which will iteratively reach us closer to the smallest cost path to our goal state.

Question 7: For A* Search to be optimal and complete, should the heuristic be an underestimate or overestimate of the actual distance to the goal node? Explain.

Answer: Underestimate. Suppose we had an overestimating heuristic. Since A Search considers backwards cost summed with the heuristic, our heuristic can cause the algorithm to consider the optimal path after some other path that had a better heuristic. However, if the heuristic is an underestimate, the path with lowest cost is guaranteed to be considered first when reaching the goal node. An intuitive answer would also suffice.*

Question 8: Which of the following is an example of a difference between UCS and Dijkstra's Algorithm? Choose all that apply.

a. UCS generates the graph on demand, only adding encountered nodes to the PQ
b. UCS terminates when it finds a goal node, while Dijkstra's Algorithm finishes exploring.
c. UCS takes heuristic into account, usually resulting in better runtime than Dijkstra's Algorithm
d. UCS can fail to find a goal state depending on edge weights while Dijkstra's Algorithm is always complete

Answer: a and b. UCS generates nodes on demand as many search spaces are too big to be stored fully in memory. UCS also terminates when it finds a solution. However, UCS is an uninformed search algorithm which does not take heuristics into consideration and it is always complete, like Dijkstra's Algorithm.

Question 9: When all of the edge weights in the state space graph are equal, which search algorithm would perform best, given no information about the goal node?

Answer: BFS will have the fastest runtime while being complete and optimal. In this case, the FIFO queue of BFS will work in the same way as the UCS's priority queue, while performing

better due to lack of sinking and swimming operations of priority queue. DFS is not optimal in this case.

Question 10: Unlike the graph search algorithms you may have seen in a class like CS 61B, the search algorithms here generate the graph "on demand." Why would this be beneficial?

Answer: State spaces grow exponentially. Thus, the graphs that we work with in this setting are generally extremely large and thus might not fit on the memory state.