Homework 1 Solutions
CSCE4613

Morgan Maness and Chris Troupe

Background Questions

1.a

An algorithm is complete if there is at least one acceptable solution the algorithm can find.

1.b

An algorithm is optimal when the solution found is the best possible solution.

1.c

Depth first search is more efficient with its memory when compared with BFS. Additionally, it is better for searching trees where the node is down many levels. Therefore, DFS excels with when the solution is deep.

1.d

Breath first search is better for searching trees where the solution node is very close to the start. Additionally, for finite graphs, BFS will always find a solution and DFS can get stuck in an infinite loop.

2. Uniformed Search

2.a

In figure 1, we would prefer to use BFS. If DFS does not know when a node is visited, it would get stuck in an infinite loop on the left side. knows when a node is visited or not, they will both achieve the goal in 6 moves. For BFS the moves will be (a b e c d f) and for DFS the moves will be (a b c d e f). However, DFS achieves this problem at a faster rate. When programmed in Matlab using ”tic” ”toc” BFS finishes the test in .001555 and DFS finishes it in .001073 or roughly 60 percent of the time of BFS.

2.b.

For BFS the sequence of paths would be, 1. A B E D C F
The DFS sequence of paths would be, 1. A B C D E F

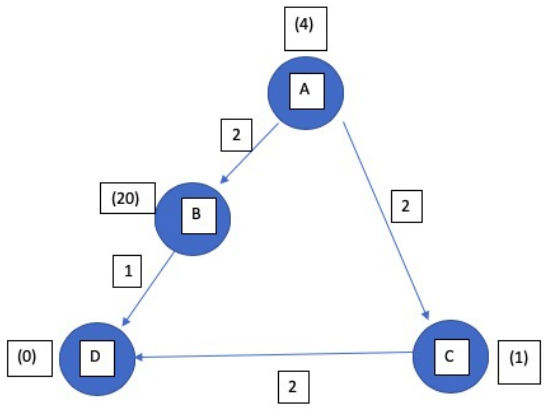

Figure 1: A* Search Tree

3. A* Search Tree

Figure 1 shows an example of a non optimal search tree. This graph is not optimal because the heuristic is not perfect. In this example, the heuristic is shown in ”()” and is the distance from the current node to the goal node. The starting node is A and the goal node is D. The A* search algorithm would go from A-C-D. The algorithm would never search out the B node because the heuristic of 20 is higher than the other edge. This solution is not optimal because it expanded the B branch of the tree, therefore not finding the best solution. The A* search algorithm gives a distance of 4 while the optimal distance would be 3.

4. A* Graph Search

This A* graph search in figure 2 is similar to the search tree. The heuristic used here is admissible, all of the heuristics are equal or less than the total distance. Like the other example, ”()” represents the heuristic and the number by the line represents the cost of the path. ’A’ is the starting node and ’G’ is the goal node. A* graph search would say give A-C-D-G as the path with a path cost of 34. The actual optimal path would be A-C-G with a cost of 32. This utilizes the fact that

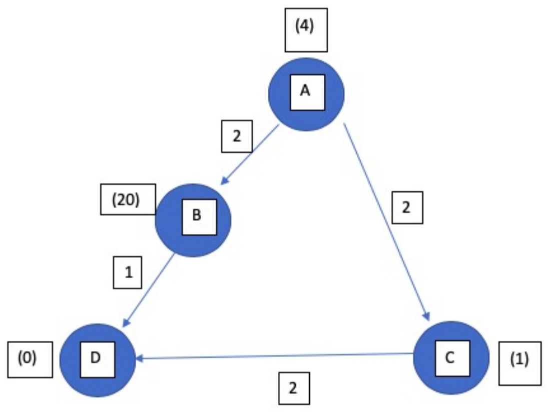

Figure 2: A* Search Tree

with graph search doesn't expand a node twice and won't go back to itself after it's chosen a path and has the lowest total value.

Search Algorithm Implementation

2. Traffic time

The heuristic function for this would be the distance from the desired node plus a designated weight for the amount of traffic on that road.

3. Real World Routing

One problem for this code is that there is no way to reroute for road closures. One way to solve that is to make a section where you can remove certain roads from the list so that it is no longer an option for the path.