Hamin Han

2 Graph Traversals

1. What is the branching factor for this graph?

S = 2

A = 2, B = 2, C = 1, D = 2

2. Based on the heuristic values, is the heuristic function:

- Admissible? If not, why?

  It is admissible because it never overestimates the actual cost to get to the goal. The heuristic values are all less than or equal to the actual costs.

- Consistent? If not, why?

  Yes, the heuristic value of the goal state is 0 and all states follow h(s) <= c(s,n)+h(n). The heuristic value of any state is less than or equal to the cost to a neighboring state plus the neighbor’s heuristic value.

3. What is the minimum cost of moving from S to the goal G? Provide an example of one of the paths that has the minimum cost.

11, S B D C G

4. What is the minimum length of all path(s) from S to the goal G? Provide an example of one of the paths that has the minimum length.

3, S B D G

5. What is the sequence of nodes in the stack/frontier when you apply Iterative Deepening on this graph to find a path from S to G? If a path is found, what is the cost of the path? How does it compare with the optimum cost?

Expanded Node	Frontier List
	{S}
S	{A,B}
A	{B}
B	{}
S	{A,B}
A	{C,D,B}
C	{D,B}
D	{B}
B	{C,D}
C	{D}
D	{D}
S	{A,B}
A	{C,D,B}
C	{G,D,B}
G	{D,B}

S->A->C->G Cost: 12 more than the actual optimum cost of 11.

3.3 Questions

1. Describe the heuristic function used in your A* search.

For my A* search, all the states have the heuristic value of 1 if it has the same total amount of water as the target state and if it's only one dump away from being the target state. The heuristic value is 2 for every other state with the exception of the goal state being 0. Of course I then add that value with the cost it took from the start state to the current state to get the A star value.

2. Is the heuristic you used admissible? Describe why.

Yes, because in the cases where it estimates a cost of 1, it will always be equal to or less than the actual cost. It's either one dump away or one pouring away from the target in which case the actual cost is 1, or it takes multiple pouring and that's fine since the heuristic is less. Naturally, everything with the heuristic value of 2 will also be less than or equal to the actual cost.

3. For each of the 5 initial to goal state pair, did BFS, DFS and/or A* reach a solution? If not, describe why?

BFS and A* reached a solution for all 5 states but DFS only reached the goal for two of it. DFS didn't find a goal since it was stuck in an infinite loop with no memory. The way my code is set up, it pours x to y, pours y to x, dump x, and dump y in that order. So with our initial state being (4,1), it will first pour x to y and make (3,2). There are three others that follow but those don't matter since we expand (3,2) and keep going into the depth. So with (3,2), pouring x to y isn't an option so it will do the next thing, which is pouring y to x and get (5,0). Again, there are other states expanded from (3,2) but those don't matter because we keep going depth first and expand (5,0) next. And with (5,0), we pour x to y and get (3,2). At this point, it will keep on looping between (3,2) and (5,0). And unless those two are your goal states, you will never reach your target state. Hence out of the five target states, DFS only managed to reach (5,0) and (3,2).

4. For each of the 5 initial to goal state pair, is the BFS, DFS and/or A* solution optimal? Describe why.

The solution costs are identical for all algorithms(other than DFS which didn't find solutions for certain). (3,2) is the same for all since it's the very first state in the queue after expanding the starting state. For everything else, A* is optimal since it finds the solution without expanding nodes as much as BFS.

5. How would you convert the A* search algorithm into Uniform Cost Search?

Instead of having a heuristic value on each state that estimates the cost to get to the goal state, it would only look at the actual cost it took to get to the current state from the start state and use those values to rank which ones to visit first.