CS 380 Homework 2 – Uniformed Search

Yena Lee (Student ID : 14531969)

1. Written problems to be turned in

A. Forward-Backward Search –

I think by showing that Franklin in one of Eloise's ancestors would be the easier way to verify Eloise's claim. Because if we search Eloise starting from Eloise's ancestors which means top-down approach, there would be another descendant except for Eloise so that we have lots of branches to search that means we must figure out a greater number of cases than another method. However, if we show that Franklin is one of Eloise's ancestors by starting from Eloise and going up using bottom-up approach, Eloise has only one parent for the tree structure, so there is limited route for search so that we can find out Franklin is Eloise's ancestor with not much effort on it.
Additionally, if we apply Forward-Backward Search that starts searching started at both side, it will be the most efficient way because we can search faster with less amounts of searching.

B. Sliding-Tile Puzzle –

a. Using this notation, show a tree depicting all possible moves from the initial state.

- Initial state : BBB_WWW

- All possible moves :

BB_BWWW
B_BBWWW
_BBBWWW
BBBW_WW
BBBWW_W
BBBWWW_

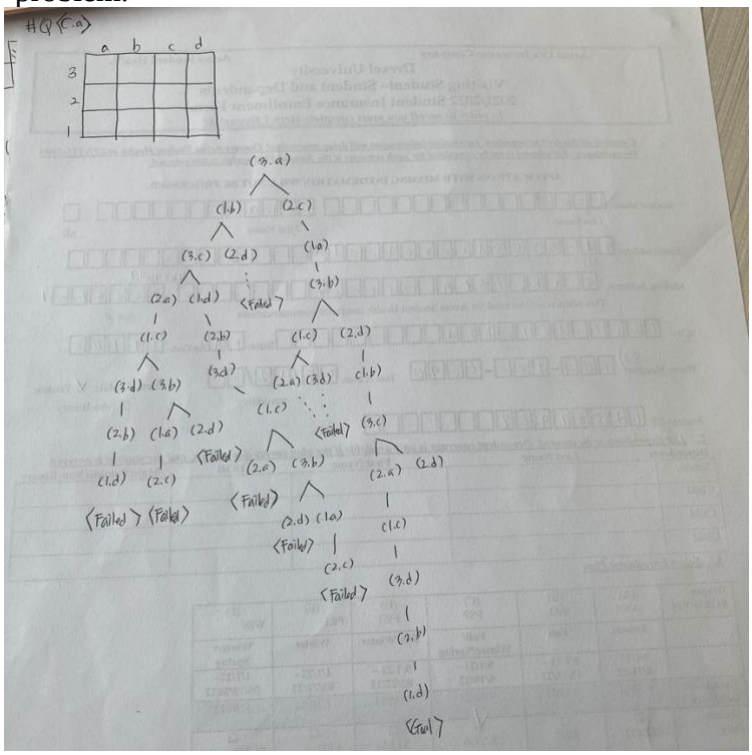
b. Note the inherent symmetry in this problem found by switching the roles of B and W tiles. Explain how to take advantage of this symmetry in order to accomplish forward-backward search.
We can use it by checking if we find all the cases correctly just by one side even though we don't have to do both sides because it is the inherent symmetry in this problem.

c. Is there an advantage to using reflectional symmetry also? (i.e., the reflected image of (**BB_BWWW**) is (**WWWB_BB**)). Explain.
I don't think so, because I can't find repetitive rules in there, so I should search all the cases with forward-backward search to switch the roles of B and W tiles.

C. A Hard Day's Knight's Tour –

a. Using **backTrack**, sketch out a solution tree, and count the number of times you backtracked (i.e., returned a "FAILED" value). You can do this by hand; your tree need not contain fastidious detail, but should give a reasonable accounting for the number of failures.

I tried about 8 backtrackings while sketching out a solution tree. I should have done backtracking because by searching, there were no places for next moves, and it made me failed value. And while trying backtracking again, I could finally solve the problem.



b. With a branching factor of 8, how many terminal nodes will a tree of depth 36 contain?

It will contain 8^36 terminal nodes, because the branching factor is 8, there will be 8 nodes one level down from the current position, so a tree of depth 36 will have that amount of nodes.

c. Suppose, on the average, 4 moves is more typical. Re-calculate your answer to part (b) with a branching factor of 4.

With 4 moves, it will be 4^36 terminal nodes, because the branching factor is 4, and there will be 4 nodes one level down from the current position.

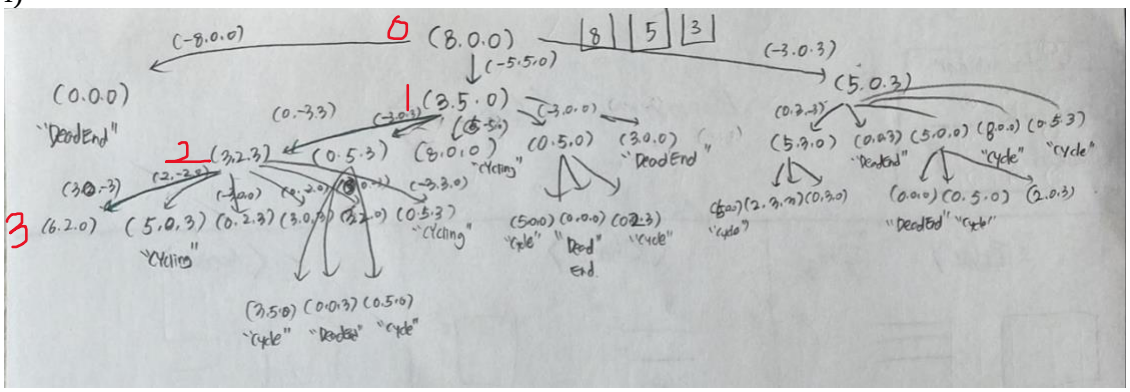
d. Note that this problem can also be solved using Forward-Backward (Bidirectional) search. How does this compare with the forward search used in part (b)?

If we do forward backward search, forward and backward search meet at depth 18, so by comparing with the forward search, it can be faster, and reduce the amount of

required exploration. With forward search, searching complexity will be O(8^36), but with bidirectional search, it will be O(4^18+4^18), so it is far less then forward search.

D. Water Jugs, Revisited –

i)



ii)