WRITING ASSIGNMENT 1

Yahya Alhinai
alhin010@umn.edu
CSCI4511W

1 Figures

The following figure describe the run-time for different N-Queen size using two different search algorithms:

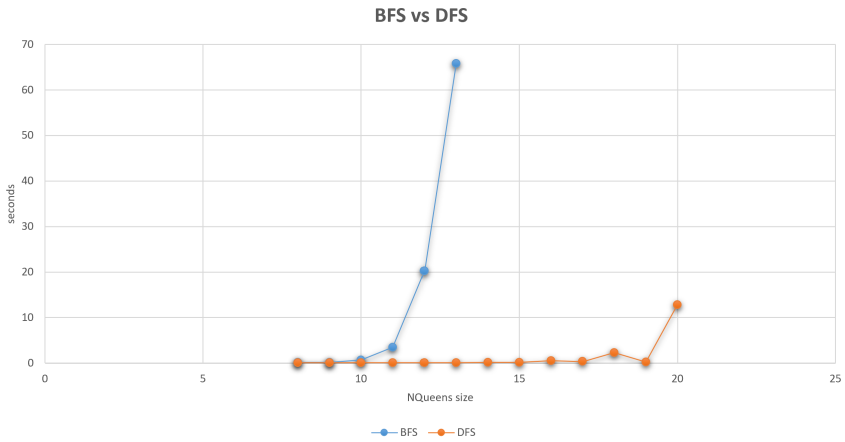

Figure 1: BFS vs DFS Runtime

The depth first search algorithm (DFS) is a superior algorithm to breadth first search (BFS). The superiority of DFS can be seen at the amount of time it takes to solve N-queens problem as well as the superiority of DFS is shown in the usage of memory in investigating game's elements. Even though both algorithms have similar performance in the aspect of time spent to complete a task and the efficiency of memory usage when the size of N-queens is small, the time gap and memory between the two algorithms increases rapidly as the sizes of N-queens gets bigger. For instance, once the size of N-queens reaches 13, the DFS becomes 10 times more efficient in the usage of memory and 700 times faster in comparisons to BFS.

The superiority of DFS algorithm in solving N-queens over BFS can be sum in two features. First, the superiority of DFS is due to the fact that this algorithm takes one path to investigate out of the possible available action and then proceed to investigate another in the next step without going back to investigate the possible path that was not taken. This will cause the search to be really fast. On the other hand, BFS algorithm will investigate every single possible path of every single state; therefore, the time it takes increases exponentially as N gets bigger. The second feature is the memory usage. At any state, DFS only keep track of the children that the node is at and

choose one of them to investigate. Whereas BFS keep track of all the children of all the nodes in any given state. This is way BFS has space complexity of $O(b^d)$ in comparison to DFS which has $O(b \times d)$. Because BFS investigate all the children of all the nodes in any given state, BFS Excel at the end results produced over DFS. As DFS gives a solution, BFS gives the optimal solution with the least amount of moves to solve the game.

2 Tables

The following figure describe the run-time and the memory usage of running Eight Puzzle problem using depth limited search algorithm and breadth first search algorithm:

Table 1: Time-run and Memory for BFS and DLS

	Time (second)	Memory (kB)
BFS	13.728	1,662,136
DLS	9.047	99,080

Starting with breadth first search (BFS). In eight-puzzle problem, The benefit of using BFS is seen in the results it produces at the end. The produced result seems to be optimal for this problem as the BFS algorithm returns the least amount of moves to solve the puzzle. Due to the face it investigate ever single child of every single node level by level in a tree. This is what made time complexity of using BFS to solve the puzzle increases exponentially because it investigates ever possible action at ever level of the tree. Also, BFS consume a great deal of memory as the algorithm uses $O(b^d)$ to keep track of ever single child of every single node at any state. To put this into a perspective, BFS required 16 times more memory than DLS to solve eight-puzzle problem.

The algorithm depth limited search (DLS) has the advantage of time and memory in comparison to BFS. We can find the superiority of DLS algorithm in the area of memory usage. This can be represented in the big-O notation for memory complexity which happened to be $O(b \times d)$. Unlike BFS, DLS only need to save the children of only a single node instead of all the nodes at a particular state. The time is correlated to the memory usage as this algorithm solves the puzzle 30% faster in comparison to BFS because DLS search as deep as possible every time it iterates which leads to the goal faster than searching all possibility that leads to the goal at the lowest depth in the tree. The downside of DLS is the produced results. The result that DLS gives is usually not the optimal solution because this search goes to investigate a branch of a tree as deeply as possible and stop once it reach the end goal which does not give the minimum moves to solve eight-puzzle problem.

It is worth mentioning that depth first search algorithm (DFS) would not work to solve this eight-puzzle problem. Since DFS investigate a branch of a tree as deeply as possible might lead to be stuck in a loop forever because each element in the puzzle can be a child of another any other element which create a circle. Therefore, for a lot of cases the algorithm will never be able to give a solution to the puzzle.