Assignment- 0

1. The program follows the 5-step abstraction

Sl.No	Abstraction	Code logic
1	Valid states	A board with N or fewer than N rooks on a chess board in any arrangement are the valid states in the starter code.
2	Initial State	A blank chess board is the initial state. Represented by a list of lists filled only with zeroes.
3	Successor Function	To add one rook at a given coordinate of a square on the N*N chess board.
4	Goal State Definition	The state where there are N rooks on an N*N sized chess board such that no rook kills another.
5	Cost Function	N/A

2. Upon changing the code to solve the nrooks using BFS the solution got generated for N=3, N=4 and N=6. It starts to take a lot of time from N=6. Whereas the solution doesn't get generated after N=3 for DFS as the code got into a infinite loop due to successor function generating duplicate states.

   The BFS modification works for a small N but as N increases the BFS would take a lot of time since it must branch out from a lot of states at a level in the tree if there is no solution found in an earlier state. This time would only increase rapidly with increasing N as there would be lot of states to go through.

   The solver was changed to BFS by popping (pop(0)) the first element from the fringe instead of the last element (pop()) in the case of DFS. Upon changing the above the fringe is now transformed from a stack to queue for BFS.

3. Upon fixing the two issues both BFS and DFS work at a similar speed till N=5. But from N=6, BFS starts to take a lot of time to arrive at the solution. And the time would only increase rapidly further for N>6. This is because the number of nodes to search at a level increase dramatically for BFS and that would consume a lot of time if the solution is deep inside the tree.

   On the other hand DFS would find the solution (assuming to deep for big N) at much faster rate as the search progresses to all the branches at single node at a level in the tree. Whereas BFS must go through all the nodes at a level and then proceed to the further level. Also, memory requirements would be be much higher for BFS with higher N. So, with increasing N, DFS makes more sense to find a solution than BFS.

4. The successor function has been modified to add a piece only to the leftmost empty column. The drop in runtime is due to decrease in the number of states to search for a solution. The runtime has decreased drastically compared to the earlier versions. The largest N that could be solved successfully under a minute using successors3 is N=250 which takes approximately 60s. The fit appears to be a polynomial of $O(n^2)$.