B551-ASSIGNMENT 0

Q1.

N-Rooks

Valid States: All the states where none of the rooks can take over each other, and there should be a total of N rooks for a NxN board.

Successor function: Place first rook on any arbitrary square on the board, then place the other rooks at a position where no rook can take over each other and a NxN board is filled with N rooks.

For Ex: For a Board of size 3x3 we consider a 3x3 matrix A where each block is denoted by A_{ij} where i=>the row number and j=>the column number on the board. Now suppose we place a rook on A_{11}(rows and columns starting from 1) no rook can be placed on A_{12}, A_{13}, A_{21}, A_{31} so the next rook can be on any of the blocks except the above 4

Cost Function: Moving N blocks forward/backwards or sideways, so the cost of each step would be N

Goal State: A state where N rooks are placed on a NxN board where no rook can kill each other, so in this case the only possible solution is placing all the rooks diagonally on blocks where i=j where i=>row number and j=>column number(for a board with unrestricted blocks).

Initial State: Empty NxN board.

N-Queens

Valid States: All the states on the board where no queen can kill other queen and there should be a total of N queens on a NxN board

Successor function: Place one queen on the leftmost column of the board, eliminate the positions where that queen can attack, then place the next queen on the next column where the previous queen can't attack. Repeat this process till all N queens are placed on the board.

Cost Function: Moving N blocks forward/ backward/ sideways or diagonally.

Goal State: A state where N queens are placed on a NxN board where no queen can kill each other.

Q2

The changes we observe after switching to BFS from DFS are:

- The code now uses a queue data structure instead of stack so the elements of the algorithm are search on a basis of first in first out basis instead of first in last out basis. So the algorithm no longer pops out the top most element in the stack but in fact pops out the last element of the stack.

- The code now becomes faster and works till N=6 the memory error thrown in the BFS is removed as in BFS there were too many branches and all we needed to do is to search in one particular depth.

Q3

After creating a successors2 function which solves the problems of N+1 and creating states which allows addition of no rooks at all. The choice of BFS and DFS matters because:

-In BFS the condition of N+1 doesn't get solved as the final state generated will have N+1 condition which is not required for our implementation so there is a state overhead, and in the case of DFS the state with N+1 is not generated and we stop at the goal state.

Q4

-The Largest value of N my new code can run is upto 1200 rooks.