Assignment 0

Ans 1.

SET OF VALID STATES:
The set of valid states of a n*n matrix are $2^{n^2}$. As the number of rows of board is 'n' and the number of columns of the board is also 'n'. So, the size of board is n*n, i.e. $n^2$.
Now since there are two possible states that a blank chess block can take, i.e. either the chess block has a rook placed on it or not. So, the set of valid states will be $2^{n^2}$.

SUCCESSOR FUNCTION:
Successor functions of the given problem is the chess block in which we can successfully place one chess piece on the chess board.

COST FUNCTION:
Cost function in the given problem will be the cost to place one chess piece on the given board.

GOAL STATE:
Our goal state is to place N-rooks/N-queens such that they do not hit/kill each other on an N*N board.
i.e. if our board is of the size 8*8 they we have to place 8 rooks / 8 queens such that the rooks or queens do not kill each other respectively.

INITIAL STATE:
Our initial state is a blank board of size N*N.

Ans 2.

To modify from DFS to BFS is same as saying to modify from stack to list. In a stack we take out the last element and in a list we take out the first element i.e. popping out the last and first element respectively.

To do this we use *fringe.pop()* in DFS to that last element is popped out and we use *fringe.pop(0)* in BFS to pop out the first element.

Here we observe that the code runs for BFS and DFS for N = 2. The code in both DFS and BFS takes almost the same time. But when we increase N = 3 or 4 in DFS the program goes in an infinite loop, unlike in BFS.

In DFS, for N = 3, 4 the code goes in an infinite loop because the same successor is created again and again the program is never able to find the solution.

Ans 3.

Creating successor2() solves the above-mentioned problem. Now for larger values of N i.e. N>5, BFS takes significantly more time than DFS. So to solve the given problem in faster amount of time, we will use DFS.

The increase in time happens when we use the same value of N and check for the time taken in DFS and BFS, BFS takes more time as it visits all the nodes in each layer. As our sample space is very large, visiting every node will take much more time and memory.

In DFS only one node is expanded so the program has to remember only a single path, unlike BFS which keeps all the paths in its memory thus making BFS slower.

Ans 4.

After creating successor3() and making some other modifications, the largest value of N which runs on my nrooks.py is N = 320