#---Elements of AI--

#Assignment0

#1.
 # There are Abstraction used by the supplied code
#a) Set of valid steps
A state space contains all the possible states from initial state to goal state. In other words, the set of all permutations from initial state to goal state.
The set of valid sets contains only those states which are possible from a given state. For example in this problem, we avoided the steps where the add function trying to add a piece outside of the board, rooks has to be placed where no rooks attacking it etc. which were invalid steps.

#b)Initial State
Initial state is the presented state of the problem that is provided to us before any kind of calculations. It does not contain any information of the next possible states.
For example if a person wants to go from Bloomington to San Francisco , the initial state will be Bloomington.

#c)Successor function
This function gives the set of all possible moves from one step to the next. For example if there is a path exist from point A to point B and point C. Then the successor function will give us values(A,B) and (A,C).

#d)Cost function
Cost function decides the cost of every possible move we take from one state to the next state. In other words it tells us how close are we from our goal state. For example if a person wants to go from Bloomington to San Francisco , he/she can reach there with different paths , service(bus, car, flight, train or all or some of them). All the paths have different cost values and if we are taking all or some of the services mentioned above then the cost function has to calculated for every step it takes.
It is of 2 types.
1.Uniform cost function
If the cost of the path we take from one state to another does not vary then it is called as uniform cost function.
2.Non-uniform cost function
If the cost of the path we take from one state to another varies then it is called as non-uniform cost function. In this case, we need to choose the path that is economical.

#e)Goal state
This state tells us if we have achieved what we were trying to achieve or not. It is the final state and also our destination. For example if a person wants to go from Bloomington to San Francisco then San Francisco is the goal state. In the nrook problem, if we are achieving a state where we can add n rooks that are not attacking each other is a goal state.

#2.

BFS uses FIFO(First in First Out) queue and DFS uses LIFO(Last In First Out) stack. So in order to switch the function from DFS to BFS ,we just needs to put pop(0) in solve function.

By calling the successor function with BFS, I am getting results for N<=5 but with DFS even N=3 takes too much time. This is happening because with DFS we are removing elements from fringe from last and we have not handled the (N+1) and the steps when it is actually not adding anything on the board. So, we are pulling out the inefficient steps and that's why it takes time to reach the goal state.

#3.

I have implemented the logic of discarding (N+1) moves and not allowing moves where it is not adding anything to the board. Yes, for my program the DFS and BFS still matters. With DFS I am getting values but it's slow. This is happening because BFS is an iterative program where as DFS is a recursive program. BFS can find the shortest path and able to reach goal state early.

#4.

My version of successor 3() runs for N=190 within 1 minute.

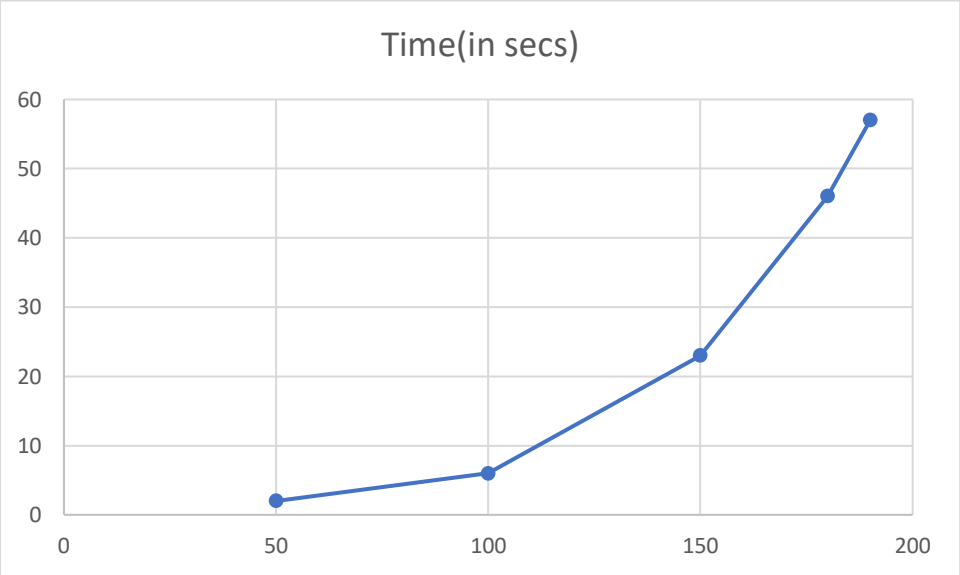
The running time of N varies exponentially.

Suppose b=branching factor

d=depth of node

Then time complexity of the program will be b^d. In this case, it is N^N .

Serial No	N	Time(in secs)
1	50	2
2	100	6
3	150	23
4	180	46
5	190	57



The values in table and in graph may change with different systems, hardware etc.