# Assignment 0: N-queens and Python

**Ans 1.**
**Set of valid states** - Arrangement in which no rook attacks any other rook
**State Space -** All possible combinations of N rooks placed on an NxN chessboard
**Successor Function** - Given by finding the next square where the new rook can be placed so that it doesn't attack the rooks already placed
**Cost Function** - Number of conflicts when a rook is placed in a certain position(row, col)
**Goal State Definition** - Placing N rooks in such a way that no two rooks attack each other
**Initial State** - Empty chessboard of size 'NxN'

**Ans 2.**
In DFS (for N>3) the program was stuck in an infinite loop as it would pop the last state which would be the same as the initial state as N+1 states are generated.
After implementing BFS the program would check the nodes near to the root node and find the result. Therefore it wouldn't get stuck in an infinite loop and find the result. But as N would increase the answer would be further away from the root node and as BFS checks the nodes near the root node first it would take a lot of time to find the goal.

**Ans 3.**
After removing the states that have N+1 rooks and removing the states that involve not adding a rook the choice of BFS or DFS does matter. As we have now removed the condition of infinite loop explained in the above answer and also removed the states that do not add a rook we can conveniently use DFS.
As mentioned in the previous answer BFS checks the nodes near the root node first whereas DFS checks the node depth-wise. And as N increases the goal state will be further away from the initial state(root node).
Therefore, DFS is faster in finding the goal state.

**Ans 4.**
Largest Value of N = 205