Venkata Prudhvi Raj Indana.

1st Answer: -

For n rook problem -

- Valid States: - Any state having no two ones' in same row or column is a valid state and we can proceed future to Successor function until goal state is reached.
- Successor function: - From the initial board, start with an empty board (board list having all zeros), add a 1 starting from left column until last column and top row to bottom and check if it is a valid state, if valid call the Successor function to the new valid state until we reach goal state or no more possible valid state.
- Cost function: - Cost function is not applicable for N-rook problem.
- Goal state definition: - For n-rook problem we need to find goal state as a list such that each row contains only one 1 i.e. sum of board list always returns 1 when looped over row wise, and each column contain only one 1 i.e. sum of board list always returns 1 when looped over row wise.
- Initial State: - As an initial state we take an empty board having a predefined row of value n and column of n as a two-dimensional list initialized with zeros. Here zeros signify that no rook is present and 1 signify that there exists a rook at that when mapped to a chess board.

2nd Answer: -

Currently n-rooks code uses DFS since it uses lists as stake by using append and pop methods in python i.e. append and add elements from same side of the list, to convert it in such a way that it uses BFS, we can change fringe to use a deque and use popleft method and append method i.e. we add elements to one side of the list and remove elements from other side of the list (as queue).

If we change the code to use BFS nrooks.py code is working for n=4 and is running for a considerable amount of time in the case of n = 8.

3rd Answer: -

I see that for n = 8 both BFS and DFS are running for long time, so I guess choose of BFS and DFS does not matter when using successor2 function. Because code is generating and working on states that have to low no probability of generating goal state. Instead we can make changes to successor function such that we give priority to states that have good change of generating goal state.

4th Answer: -

In successor3 fucntion we are only adding rooks to row/ column that does not have a rook already by checking the sum on that row and column and making sure its zero by using functions count_on_row and count_on_col. The largest value of N that works using DFS and successor3 function is 85.

5th Answer: -

After modifying the code such that it generates output for both nrooks and nqueens I can see that my code can generate output for N=9 under one minute.