**Artificial Intelligence (MCA573D)**

**LAB-1**

**1. Algorithm –** N-queens by using backtracking.

**2. Definition and Example -**

In backtracking, we first take a step and then we see if this step taken is correct or not i.e., whether it will give a correct answer or not. And if it doesn’t, then we just come back and change our first step. In general, this is accomplished by recursion. Thus, in backtracking, we first start with a partial sub-solution of the problem (which may or may not lead us to the solution) and then check if we can proceed further with this sub-solution or not. If not, then we just come back and change it.

Thus, the general steps of backtracking are:

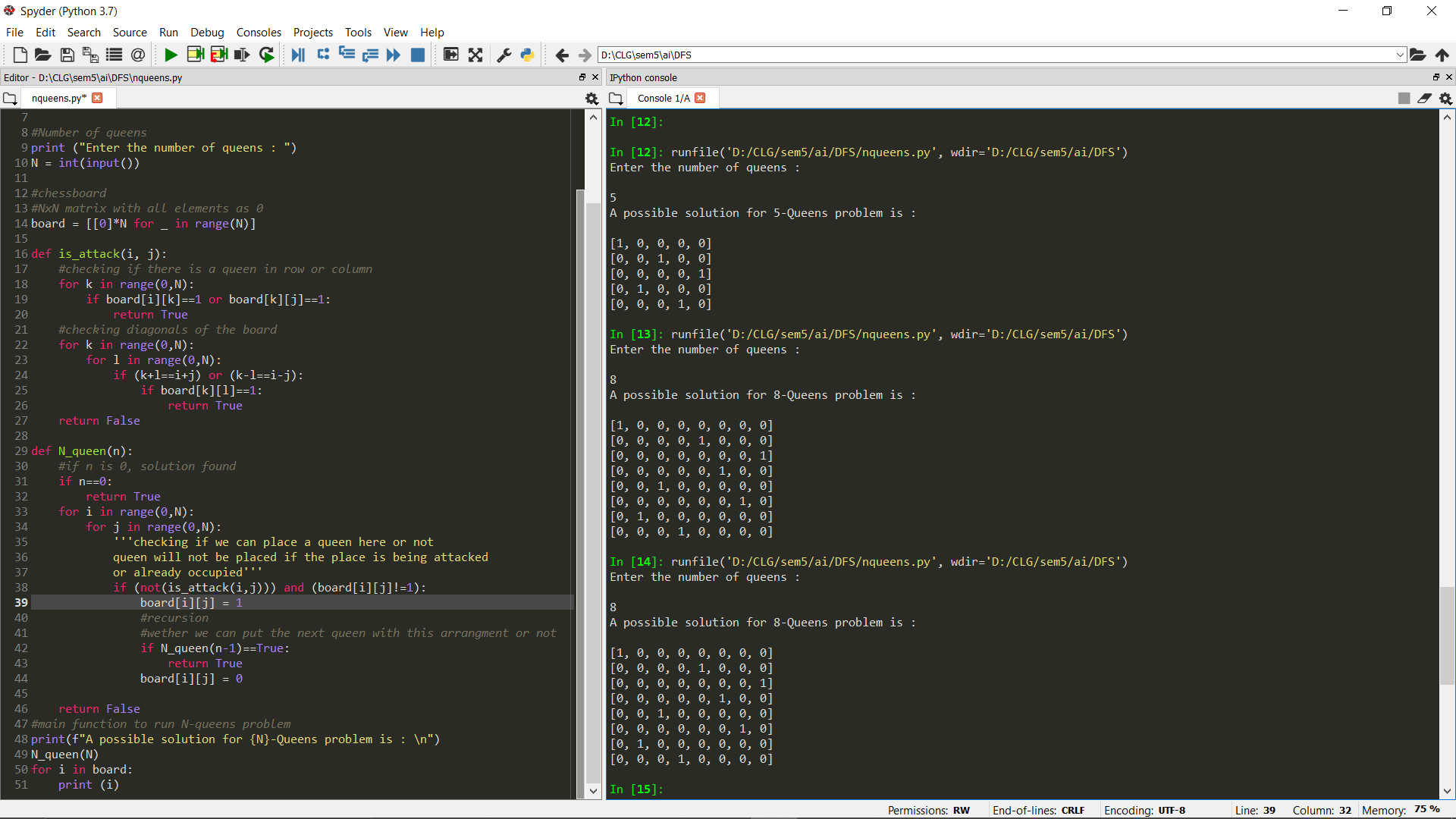
* start with a sub-solution
* check if this sub-solution will lead to the solution or not
* If not, then come back and change the sub-solution and continue again

For e.g. if we look at the n-queens problem here, we can say that :

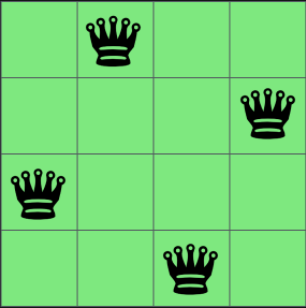
One of the most common examples of the backtracking is to arrange N queens on an NxN chessboard such that no queen can strike down any other queen. A queen can attack horizontally, vertically, or diagonally. The solution to this problem is also attempted in a similar way. We first place the first queen anywhere arbitrarily and then place the next queen in any of the safe places. We continue this process until the number of unplaced queens becomes zero (a solution is found) or no safe place is left. If no safe place is left, then we change the position of the previously placed queen.

**3. Implementation -**

Following is the code implementation of n-queens in Python :



**4. Explanation –**

****

Following is the explaination of the python code used in implementation:

**is\_attack(int i,int j)** →  This is a function to check if the cell (i,j) is under attack by any other queen or not. We are just checking if there is any other queen in the row ‘i’ or column ‘j’. Then we are checking if there is any queen on the diagonal cells of the cell (i,j) or not. Any cell (k,l) will be diagonal to the cell (i,j) if k+l is equal to i+j or k-l is equal to i-j.

**N\_queen** → This is the function where we are really implementing the backtracking algorithm.

**if(n==0)** → If there is no queen left, it means all queens are placed and we have got a solution.

**if((!is\_attack(i,j)) && (board[i][j]!=1))** → We are just checking if the cell is available to place a queen or not. is\_attack function will check if the cell is under attack by any other queen and board[i][j]!=1 is making sure that the cell is vacant. If these conditions are met then we can put a queen in the cell – board[i][j] = 1.

**if(N\_queen(n-1)==1)** → Now, we are calling the function again to place the remaining queens and this is where we are doing backtracking. If this function (for placing the remaining queen) is not true, then we are just changing our current move – board[i][j] = 0 and the loop will place the queen on some another position this time.

**5. Time Complexity -** The time complexity of n-queens problem is O(2^n), where n is the number of queens entered

For E.g. Imagine a n-queens with 4 queens problem then the time complexity will be O(2^4) = O(16).