**GROUP – A**

**Assignment No: 05**

====================================================================

Title:-

Design N-Queen matrix having first queen placed. Use backtracking to place remaining queen to generate the final N-Queen matrix.

=====================================================================Objective:-

-To learn about N-Queen problem

-To understand Backtracking method

=====================================================================

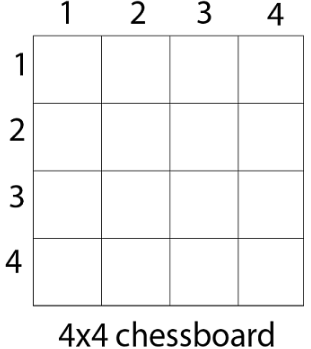
Theory:-

* In Backtracking method is finding the solution of a problem whereby the solution depends on the previous steps taken. For example, in a maze problem, the solution depends on all the steps take one-by-one. If any of those steps is wrong, then it will not lead us to the solution. In a maze problem, first choose a path and continue moving along it. But once it understand that the particular path is incorrect, then just come back and change it.
* In backtracking, first take a step and then see if that step taken is correct or not. Which means whether it will give a correct answer or not. If it doesn’t give correct solution, then just come back and change first step.
* The general steps of backtracking are:

1. Start with a sub-solution.
2. Check if this sub-solution will lead to the solution or not.
3. If not, then come back and change the sub-solution and continue again

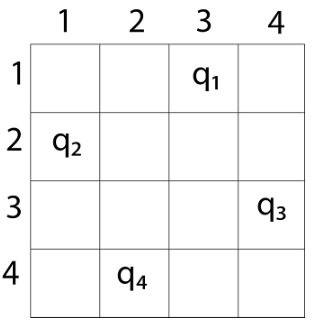
* What is N-Queen Problem?

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. First place the first queen anywhere arbitrarily and then place the next queen in any of the safe places. 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 it change the position of the previously placed queen.



The above picture shows an NxN chessboard and have to place N queens on it. So, it will start by placing the first queen.

Now, we place queen q1 in the very first acceptable position (1, 1). Next, we put queen q2 so that both these queens do not attack each other. We find that if we place q2 in column 1 and 2, then the dead end is encountered. Thus the first acceptable position for q2 in column 3, i.e. (2, 3) but then no position is left for placing queen 'q3' safely. So we backtrack one step and place the queen 'q2' in (2, 4), the next best possible solution. Then we obtain the position for placing 'q3' which is (3, 2). But later this position also leads to a dead end, and no place is found where 'q4' can be placed safely. Then we have to backtrack till 'q1' and place it to (1, 2) and then all other queens are placed safely by moving q2 to (2, 4), q3 to (3, 1) and q4 to (4, 3). That is, we get the solution (2, 4, 1, 3). This is one possible solution for the 4-queens problem. For another possible solution, the whole method is repeated for all partial solutions. The other solutions for 4 - queens problems is (3, 1, 4, 2) i.e.



The implicit tree for 4 - queen problem for a solution (2, 4, 1, 3) is as follows:

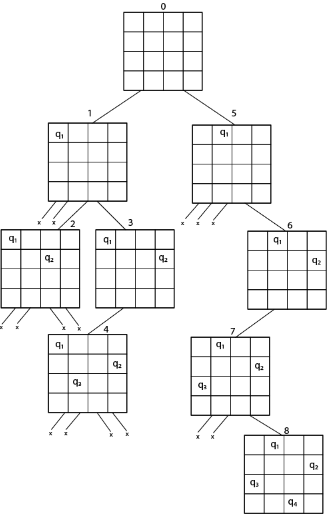


Fig shows the complete state space for 4 - queens problem. But we can use backtracking method to generate the necessary node and stop if the next node violates the rule, i.e., if two queens are attacking.

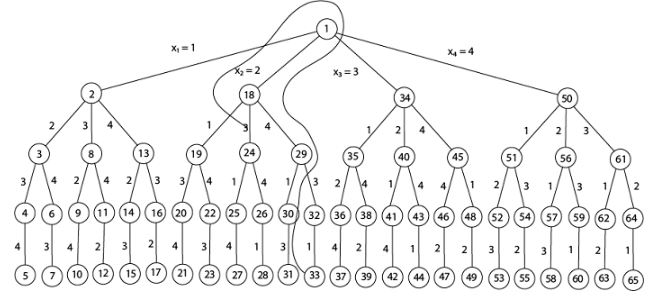
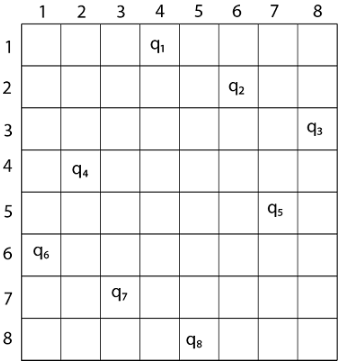


Figure: 4-Queen solution space with nodes numbered in DFS

It can be seen that all the solutions to the 4 queens problem can be represented as 4 - tuples (x1, x2, x3, x4) where xi represents the column on which queen "qi" is placed.

One possible solution for 8 queens problem is shown in figure:



**Algorithm**:

* Step 1 - Place the queen row-wise, starting from the left-most cell.
* Step 2 - If all queens are placed then return true and print the solution matrix.
* Step 3 - Else try all columns in the current row.
* Condition 1 - Check if the queen can be placed safely in this column then mark the current cell [Row, Column] in the solution matrix as 1 and try to check the rest of the problem recursively by placing the queen here leads to a solution or not.
* Condition 2 - If placing the queen [Row, Column] can lead to the solution return true and print the solution for each queen's position.
* Condition 3 - If placing the queen cannot lead to the solution then unmark this [row, column] in the solution matrix as 0, BACKTRACK, and go back to condition 1 to try other rows.
* Step 4 - If all the rows have been tried and nothing worked, return false to trigger backtracking.

The time complexity of N-Queen problem using backtracking method is O(N!). N choices in the first row, N-1 in the second row, N-2 in the next .

=====================================================================

Conclusion:-

Thus we have studied how to solve N-Queen problem. The use of backtracking method is find a feasible solution to a decision problem.

=====================================================================