GROUP B: Design & Analysis of Algorithms
TITLE: N Queens
(Backtracking)

Problem Statement:-

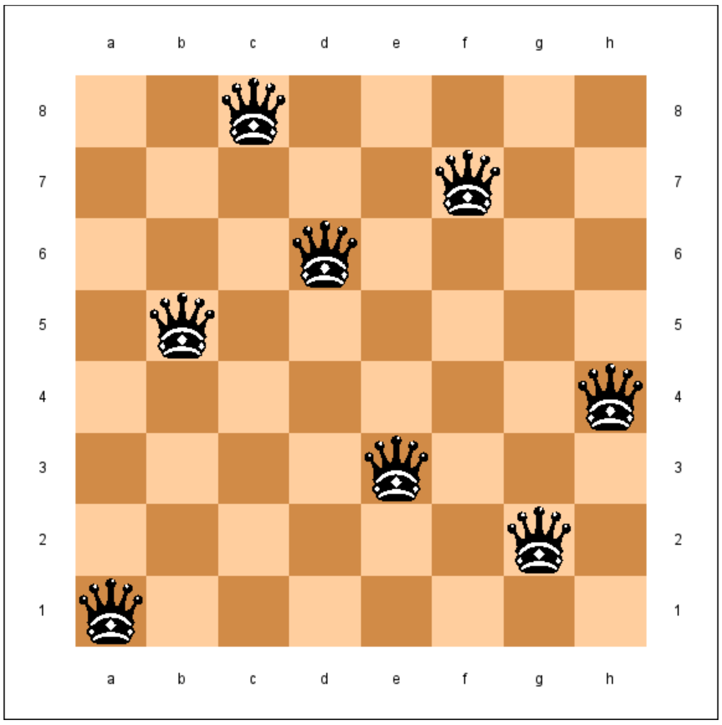
Write a recursive program to find the solution of placing n queens on a chessboard so that no queen takes each other.

Objective:-

To understand the n queens problem and learn backtracking method, to find the solution for the n queens problem

Theory:-

The **eight queens puzzle** is the problem of placing eight chess queens on an 8×8 chessboard so that none of them can capture any other using the standard chess queen's moves. The queens must be placed in such a way that no two queens attack each other. Thus, a solution requires that no two queens share the same row, column, or diagonal. The eight queens puzzle is an example of the more general ***n*-queens problem** of placing *n* queens on an *n*×*n* chessboard.



Backtracking Method:

Backtracking is a general algorithm for finding all (or some) solutions to some computational problem that incrementally builds candidates to the solutions, and abandons

each partial candidate c ("backtracks") as soon as it determines that c cannot possibly be completed to a valid solution.

Backtracking can be applied only for problems which admit the concept of a "partial candidate solution" and a relatively quick test of whether it can possibly be completed to a valid solution. It is useless, for example, for locating a given value in an unordered table. When it is applicable, however, backtracking is often much faster than brute force enumeration of all complete candidates, since it can eliminate a large number of candidates with a single test.
Backtracking depends on user-given "black box procedures" that define the problem to be solved, the nature of the partial candidates, and how they are extended into complete candidates. It is therefore a met heuristic rather than a specific algorithm --- although, unlike many other meta-heuristics, it is guaranteed to find all solutions to a finite problem in a bounded amount of time.

Description of the method
The backtracking algorithm enumerates a set of partial candidates that, in principle, could be completed in various ways to give all the possible solutions to the given problem. The completion is done incrementally, by a sequence of candidate extension steps.
Conceptually, the partial candidates are the nodes of a tree structure, the potential search tree. Each partial candidate is the parent of the candidates that differ from it by a single extension step; the leaves of the tree are the partial candidates that cannot be extended any further.
The backtracking algorithm traverses this search tree recursively, from the root down, in depth-first order. At each node c, the algorithm checks whether c can be completed to a valid solution. If it cannot, the whole sub-tree rooted at c is skipped (pruned). Otherwise, the algorithm (1) checks whether c itself is a valid solution, and if so reports it to the user; and (2) recursively enumerates all sub-trees of c. The two tests and the children of each node are defined by user-given procedures.
Therefore, the actual search tree that is traversed by the algorithm is only a part of the potential tree. The total cost of the algorithm is basically the number of nodes of the actual tree times the cost of obtaining and processing each node. This fact should be considered when choosing the potential search tree and implementing the pruning test.
Solution to 8 queens problem:
8 queens problem can be solved using backtracking method. The problem can be generalized. Consider an n×n chessboard and try to find all ways to place n non-attacking queens. Let (x1, …., xn) represent solution in which xi is the column of the ith row where the ith queen placed. The xi's will all be distinct since no two queens can be placed in the same column.

If we imagine the chessboard squares being numbered as the indices of the two-dimensional array a[1:n, 1:n], then we observe that every element on the same diagonal that runs from the upper left to the lower right has the same row – column value.

Suppose two queens are placed at positions (I,j) and (k,l). Then by the above they are on the same diagonal only if

$$i - j = k - l \qquad \text{or} \qquad I + j = k + l$$

The first equation implies

j – l = i – k

The second implies

J – l = k – i

Therefore two queens lie on the same diagonal if and only if | j – l | = | i – k |.

Algorithm Place(k, i) returns a Boolean value that is true if the kth queen can be placed in column i. It tests both whether I is distinct from all previous values x[1], …., x[k-1] and whether there is no other queen on the same diagonal. Its computing time is O(k-1).

```
Algorithm Place(k,i)
    // Returns true if a queen can be placed in kth row and ith column. Otherwise it
returns false.
    //X[] is a global array whose first (k-1) values have been set.
    //Abs( ) returns absolute value of r

    {
        for j := 1 to k-1 do
            if (( x[j] =i)  //Two in the same column
                or (Abs(x[j] – i) = Abs(j – k))) // or in the same diagonal
                then return false;
        return true;
    }
```

Using Place, we can give a precise solution to n- queens problem. Algorithm NQueens is the solution to the problem.

The array x[ ] is global. The algorithm is invoked by NQueens(1,n);

```
Algorithm NQueens(k,n)
    //Using backtracking, this procedure prints all possible placements of n queens on
    an n×n chessboard so that they are nonattacking.
    {
        for i:= 1 to n do
        {
            if Place(k,i) then
            {
                x[k] := I;
                if (k==n) then write (x[1:n]);
                else NQueens(k+1,n);
            }
        }
    }
```

For an 8×8 chessboard there are $\binom{64}{8}$ possible ways to place 8 pieces, or approximately 4.4 billion 8- tuples to examine. However, by allowing only placements of queens on distinct rows and columns, we require the examination of the most 8!, or only 40,320 8-tuples.

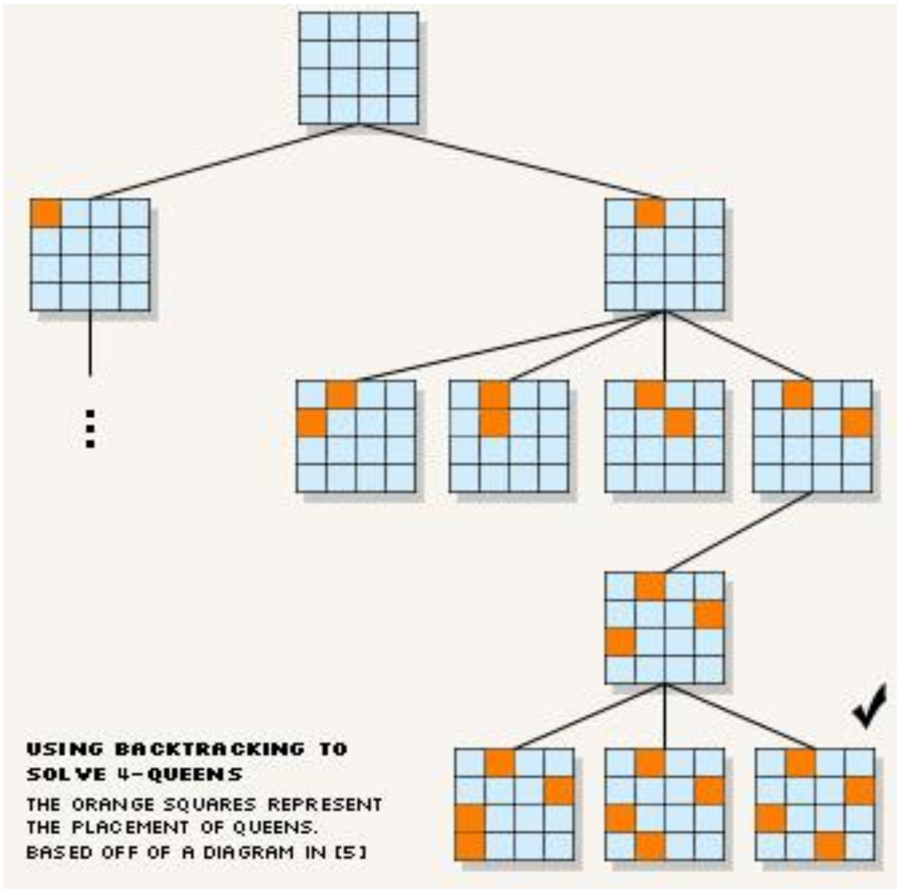

USING BACKTRACKING TO SOLVE 4-QUEENS
THE ORANGE SQUARES REPRESENT THE PLACEMENT OF QUEENS.
BASED OFF OF A DIAGRAM IN [5]

Figure: 8 – queens solution space

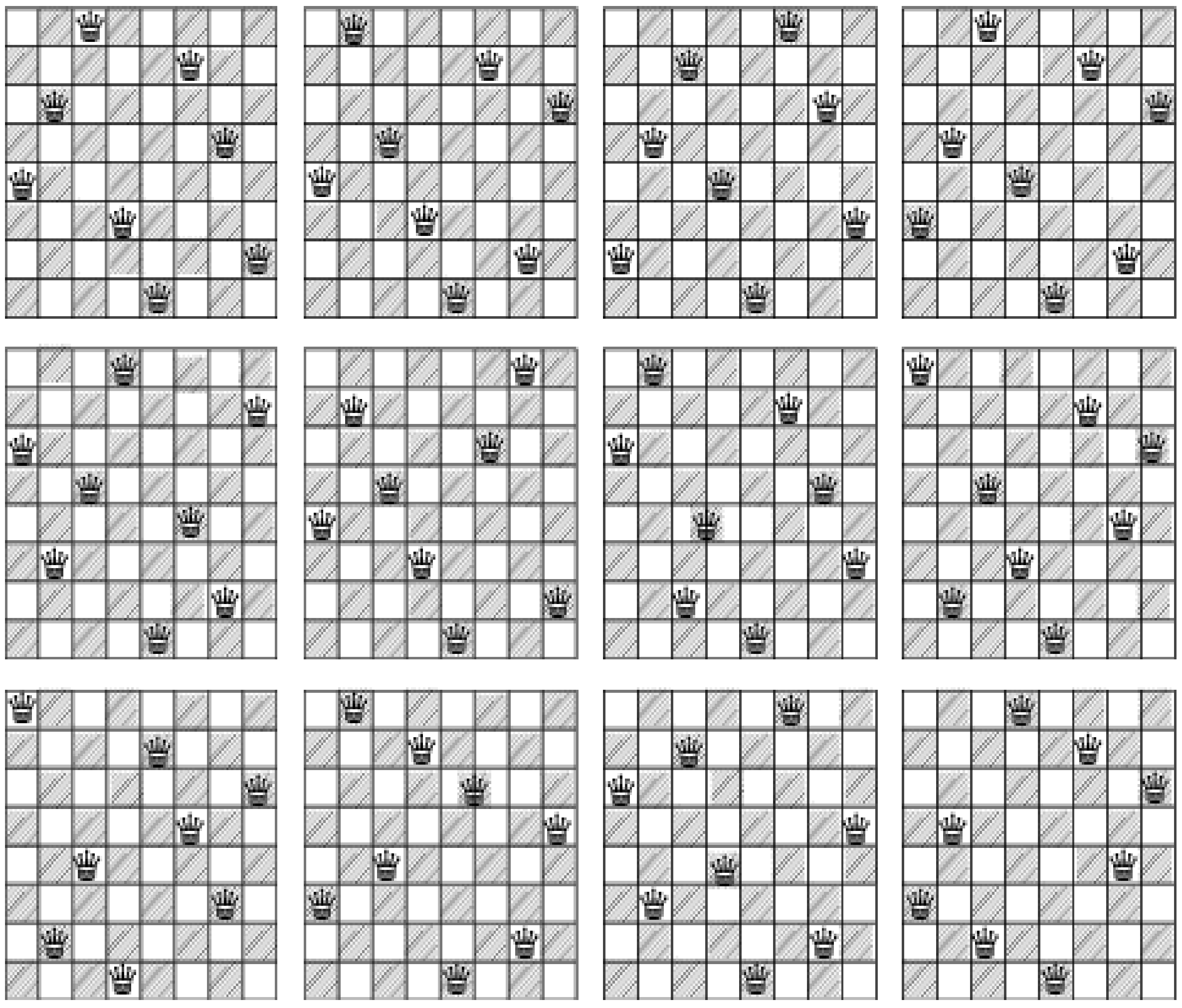
We can use Estimate to estimate the number of nodes in the solution space that will be generated using NQueens.

```
Algorithm Estimate( )
        //This algorithm follows a random path in a state space tree and produces an
estimate of the number of nodes in the tree.
        {
                k := 1; m := 1; r :=1;
                repeat
                {
                        Tk := {x[k] | x[k] ε T( x[1], x[2], …, x[k-1])
                                and Bk(x[1],…, x[k]) is true};
                        if (Size(Tk) = 0) then return m;
                        r := r * Size(Tk); m:= m + r;
                        x[k] := Choose(Tk);  k:= k + 1;
                } until (false);
        }
```

The assumptions that are needed for Estimate do hold for NQueens. The bounding function is static. No change is made to the function as the search proceeds. In addition, all nodes on the same level of the state space tree have the same degree. In following figure, five 8×8 chessboards are shown that are crated using Estimate.

INPUT:
Number of queens on the chessboard (eg n = 8)

OUTPUT:
The 8 queen puzzle has 92 solutions. If the solutions that differ by only symmetry operations (rotations and reflections) of the board are counted as one, the puzzle has 12 unique solutions.



FAQS:

1. What is n queens problem?
2. What is solution space?
3. Explain solution space for 4- queens problem.
4. Explain backtracking method
5. Differentiate between brute force method and backtracking method