**ASSIGNMENT NO: 1**

**TITLE: N Queens Problem**

**(Backtracking)**

**Problem Statement:-**

Write a recursive program to find the solution of placing n queens on chess board so that no two queens attack each other using backtracking.

**Objective:-**

To learn:

* Need and significance of Backtracking
* General method of Backtracking
* Recursive and Iterative algorithms
* 8-queens’ problem and its analysis
* Bounding function for 8-queens’ problem
* Recursive backtracking algorithm for 8-queens’ problem

**Theory:-**

**A) Backtracking strategy:**

Suppose you have to make a series of *decisions,* among various *choices,* where

* *You don’t have enough information to know what to choose*
* *Each decision leads to a new set of choices*
* *Some sequence of choices (possibly more than one) may be a solution to your problem*

Backtracking is a methodical way of trying out various sequences of decisions, until you find one that “works”.

General steps followed in Backtracking:

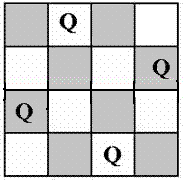
* Construct the state space tree:
  + *Root represents an initial state*
  + *Nodes reflect specific choices made for a solution’s components.*
    - Promising and nonpromising nodes
    - leaves
* Explore the state space tree using depth-first search
* “Prune” non-promising nodes
  + *dfs stops exploring subtree rooted at nodes leading to no solutions and...*
  + *“backtracks” to its parent node*

**B) Algorithm structure:**

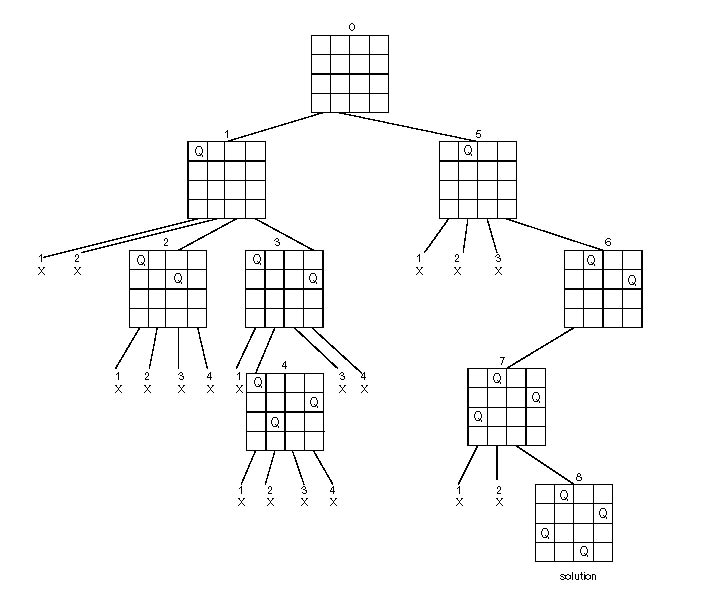
* ***Iterative*** Þ ***execute action in loop***
* ***Recursive*** Þ ***reapply action to subproblem(s)***

**C) N Queens’ problem:**

The N Queen is the problem of placing N chess queens on an N×N chessboard so that no two queens attack each other. For example, following is a solution for 4 Queen problem.



State space tree for 4 Queens’ problem:



**Algorithm:**

**Algorithm Place(k,i)**

//Returns true if a queen can be placed in kth row & ith column.Otherwise it returns false.

//x[i] is a global array whose first (k-1)values have been set.

//Abs(r) returns the absolute value of r.

{

for j:=1 to k-1 do

if ((x[j]=i)// Two in the same row

or (Abs(x[j]-i) = Abs(j-k))) // Two in the same diagonal

then return false;

return true;

}

**Algorithm NQueen(k,n)**

// Using backtracking, this procedure prints all possible placements of n queen on an

// n x 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 NQueen(k+1 ,n);

}

}

}

**Analysis:**

The backtracking algorithm is more effective over the brute force approach. For 8 \* 8 chess board, there are 64C8 ways to place 8 queens or 4.4 billion 8 tuples to examine. By allowing only placements of queens on distinct rows and columns, we require to examine at most 8! tuples or 40,320 tuples.

**Input and Output:**

**Input :** N=no.of queens to be placed on N x N chessboard.

**Output=**all possible solutions to N Queens’ problem

**Example:** The expected output is a binary matrix which has 1s for the blocks where queens are placed. For example following is the output matrix for one of the solution of 4 queens’ solution.

{ 0, 1, 0, 0} OR (-- Q -- --)

{ 0, 0, 0, 1} OR (-- -- -- Q)

{ 1, 0, 0, 0} OR (Q -- -- --)

{ 0, 0, 1, 0} OR (-- -- Q --)

**Graphical output is not expected, if done most welcome!**

**Conclusion:**

N Queens’ problem is studied and implemented using recursive backtracking strategy.

**FAQ :**

1. Explain backtracking strategy, its need and characteristics.
2. Explain the significance of bounding function in backtracking.
3. Explain terms: a) Solution space b) State space c) Answer state d) Static trees e) Dynamic trees f) Live nodes g) Bounding function h) Explicit constraint i) Implicit constraint
4. What are bounding conditions used in solving N Queens’ problem?
5. Analyze the recursive backtracking solution for N Queens’ problem.
6. Is there any another algorithmic strategy to solve N Queens’ problem? If yes , explain the same in brief.