**East WestUniversity**

**Department of Computer Science and Engineering**

**CSE 246**

**Project: N-Queen Problem**

**Sec: 03**

**Course Title: Algorithm**

**Semester: Fall-2021**

**Course Instructor: Jeson Ahammed Ovi, Senior Lecturer, Dept. of**

**Computer Science & Engineering.**

**Submitted To:**

**Jeson Ahammed Ovi**

**Senior Lecturer, Dept. of**

**Computer Science & Engineering**

**Submitted By:**

**Group Member1: Tarpon Datta**

**ID: 2020-1-60-202**

**Group Member2: Zarin Tasnim Nuzhat**

**ID: 2020-1-60-211**

**Problem Statement:**

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, to solve the eight queens puzzle (when N=8), the problem is to place the eight chess queens on an 8×8 chessboard so that no two queens threaten each other. Thus, a solution requires that no two queens share the same row, column, or diagonal. To solve the problem, keep a simple rule: last placed, first displaced. In other words, if you place successfully a queen on the ith column but cannot find solution for (i+1)th queen, then going backwards you will try to find other admissible solution for the ith queen first. This is a Backtrack approach where you try to build a solution one-step at a time. If at some step, it becomes clear that the current path that you are on cannot lead to a solution, you go back to the previous step (backtrack) and choose a different path. Briefly, once you exhaust all your options at a certain step you go back.

**System Requirements:**

Here, the processor “AMD Ryzen 7 PRO 2700U w/ Radeon Vega Mobile Gfx 2.20 GHz” is used. The operating system “Windows” is used here. We use for C++ code, Codeblock IDE.

**System Design:**

Begin within the leftmost column. In case all queens have set, return true Try all columns within the current column. On the off chance that placing the queen in [row, column] leads to an arrangement at that point return genuine. In case the placing queen does not lead to an arrangement at that point unmark this [row, column] (Backtrack) and go to first step to undertake other rows.

**Implementation:**

**#include<iostream>**

**using namespace std;**

**int row, col;**

**bool isSafe(int\*\* arr, int x, int y, int n)**

**{**

**for(int row=0; row<x; row++)**

**{**

**if(arr[row][y]==1)**

**{**

**return false;**

**}**

**}**

**int row= x;**

**int col= y;**

**while(row>=0 && col>=0)**

**{**

**if( arr[row][col] == 1)**

**{**

**return false;**

**}**

**row--;**

**col--;**

**}**

**row= x;**

**col= y;**

**while(row>=0 && col<n)**

**{**

**if( arr[row][col] == 1)**

**{**

**return false;**

**}**

**row--;**

**col++;**

**}**

**return true;**

**}**

**bool nQueen(int\*\* arr, int x, int n)**

**{**

**if(x>=n)**

**{**

**return true;**

**}**

**for(int col=0; col<n; col++)**

**{**

**if(isSafe(arr,x,col,n))**

**{**

**arr[x][col]=1;**

**if(nQueen(arr,x+1,n))**

**{**

**return true;**

**}**

**arr[x][col]=0; //backtracking**

**}**

**}**

**return false;**

**}**

**int main()**

**{**

**int n;**

**cin>>n;**

**int\*\* arr=new int\* [n];**

**for(int i=0; i<n; i++)**

**{**

**arr[i]=new int [n];**

**for(int j=0; j<n; j++)**

**{**

**arr[i][j]=0;**

**}**

**}**

**if(nQueen(arr,0,n))**

**{**

**for(int i=0; i<n; i++)**

**{**

**for(int j=0; j<n; j++)**

**{**

**cout<< arr[i][j]<<"";**

**}**

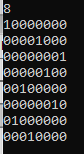
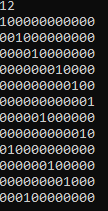
**cout<< endl;**

**}**

**}**

**}**

**Testing Results:**

**  **

**Future Scope:**

In this project, we have designed a suitable algorithm that solves the N-queen problem and shows all possible placements of N queens. We have to use **Backtracking** algorithm in this regard. We should not use **Brute Force** as using it will increase the runtime and complexity with the increasing value of N.