PPS: N QUEEN PROBLEM

PRATEEK MOHANTY

20BCE1482

* **N QUEEN PROBLEM ( ONE QUEEN IN EACH COLUMN)**
* **ALGORITHM**

1) Start in the leftmost column

2) If all queens are placed

return true

3) Try all rows in the current column.

Do following for every tried row.

a) If the queen can be placed safely in this row

then mark this [row, column] as part of the

solution and recursively check if placing

queen here leads to a solution.

b) If placing the queen in [row, column] leads to

a solution then return true.

c) If placing queen doesn't lead to a solution then

unmark this [row, column] (Backtrack) and go to

step (a) to try other rows.

4) If all rows have been tried and nothing worked,

return false to trigger backtracking.

CODE

#include <iostream>

using namespace std;

void printBoard(int board[][20], int n)

{

    for (int i = 0; i < n; i++)

    {

        for (int j = 0; j < n; j++)

        {

            cout << board[i][j] << " ";

        }

        cout << endl;

    }

}

bool canPlace(int board[][20], int n, int i, int j)

{

    // row check

    for (int k = 0; k < j; k++)

    {

        if (board[i][k] == 1)

            return false;

    }

    // left diagonal

    int l = i;

    int m = j;

    while (l >= 0 && m >= 0)

    {

        if (board[l][m] == 1)

            return false;

        l--, m--;

    }

    // right diagonal

    l = i;

    m = j;

    while (m < n && l >= 0)

    {

        if (board[l][m] == 1)

            return false;

        l--, m++;

    }

    return true;

}

bool solveNQueen(int board[][20], int n, int x)

{

    // base case

    if (x == n)

    {

        printBoard(board, n);

        return true;

    }

    for (int i = 0; i < n; i++)

    {

        if (canPlace(board, n, i, x))

        {

            board[i][x] = 1;

            bool success = solveNQueen(board, n, x + 1);

            if (success)

                return true;

            // backtracking

            board[i][x] = 0;

        }

    }

    return false;

}

int main()

{

    int n;

    cout << "Enter the dimension of the chess board" << endl;

    cin >> n;

    int board[20][20] = {0};

    for (int i = 0; i < n; i++)

    {

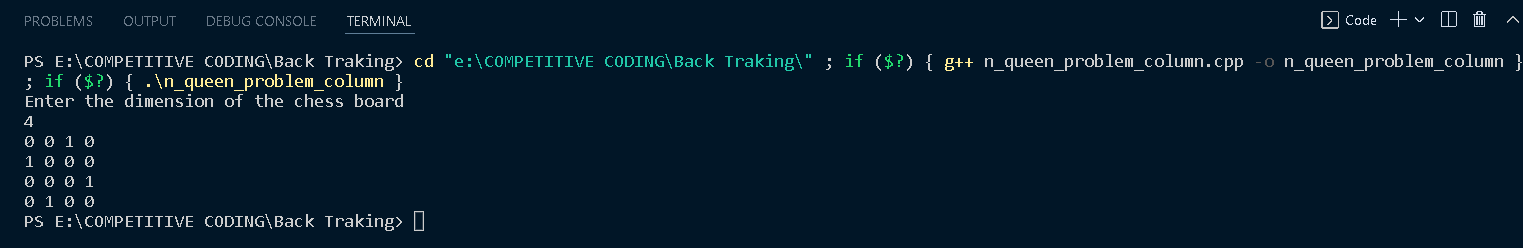
        solveNQueen(board, n, i);

    }

    return 0;

}

OUTPUT



TIME COMPLEXITY]

O(N^2)