**Experiment 3.4**

**Student Name:   Gaurav Kumar                                UID: 22MCC20177**

**Branch:   CC-DevOps                                                     Section/Group:- 1/B**

**Semester:   One                                                               Date of Performance: 06/01/2023**

**Subject Name:- Design & Analysis of Algorithms Lab                   Subject Code: 22CAP-646**

1. **Task to be done:**

**Implement 8-queen problem using branch and bound technique.**

1. **Steps for experiment/practical: copy and paste your code here/screenshots.**

#include <bits/stdc++.h>

#define N 8

using namespace std;

/\* A utility function to print solution \*/

void printSolution(int board[N][N])

{

    for (int i = 0; i < N; i++)

    {

        for (int j = 0; j < N; j++)

            if (board[i][j])

                cout << "Q ";

            else

                cout << ". ";

        printf("\n");

    }

}

bool isSafe(int board[N][N], int row, int col)

{

    int i, j;

    for (i = 0; i < col; i++)

        if (board[row][i])

            return false;

    for (i = row, j = col; i >= 0 && j >= 0; i--, j--)

        if (board[i][j])

            return false;

    for (i = row, j = col; j >= 0 && i < N; i++, j--)

        if (board[i][j])

            return false;

    return true;

}

bool solveNQUtil(int board[N][N], int col)

{

    if (col >= N)

        return true;

    for (int i = 0; i < N; i++)

    {

        if (isSafe(board, i, col))

        {

            board[i][col] = 1;

            if (solveNQUtil(board, col + 1))

                return true;

            board[i][col] = 0; // BACKTRACK

        }

    }

    return false;

}

bool solveNQ()

{

    int board[N][N] = {{0, 0, 0, 0},

                       {0, 0, 0, 0},

                       {0, 0, 0, 0},

                       {0, 0, 0, 0}};

    if (solveNQUtil(board, 0) == false)

    {

        cout << "Solution does not exist";

        return false;

    }

    printSolution(board);

    return true;

}

int main()

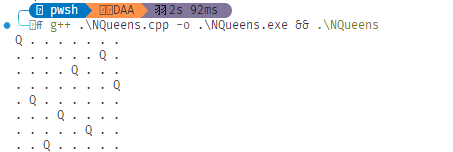
{

    solveNQ();

    return 0;

}

1. **Output (screenshots)**

****

**Evaluation Grid:**

|  |  |  |  |
| --- | --- | --- | --- |
| Sr. No. | Parameters | Marks Obtained | Maximum Marks |
| 1. | Demonstration and Performance  (Quiz) |  | 22 |
| 2. | Worksheet |  | 8 |