SESSIONAL REPORT

Course No: CSE-2202

Experiment No : 05

Name of Experiment:

Introduction to Backtracking problem by solving N-Queen problem

Objectives:

- To understand what backtracking is and why it is used.
- To solve N-Queen problem using backtracking technique.
- To analyze the time and space complexity of backtracking for N-Queen.

Introduction:

Backtracking is a problem solving method where solution is found step by step, and if any step says the solution is invalid, we get back to last point (back-track) and try to find another option. It is like trial and error for solving problem but in a smart way.

The **N-Queen problem** is a example of backtracking.
The problem says that, we have to place **N queens on an N×N chessboard** in such a way that:

- No two queens can attack each other
- That means for 2 queens - no same row, same column or same diagonal

Backtracking Is perfect for this because at each row we try to place a queen in a safe column. If placing queen causes problem that means if it attack or it is attacked by another queen, we simply remove it and try to put in next position.
By repeating this process, we can find all possible ways to place the queens safely in the chessboard.

Algorithm:

Step 1: Start from first row, row = 0

Step 2: Try to place a queen in each column of this row

Step 3: Before placing a queen, check if that position (row, col) is safe or not
A position is safe if:
- There is no queen in same column
- There is no queen in left diagonal
- There is no queen in right diagonal

Step 4: If the place is safe, then place the queen and move to next row (row + 1).

Step 5: If not safe , that means placing queen in next row is not possible, remove the queen (backtrack) and try next column.

Step 6: Repeat until all queens are placed or all possibilities are checked.

Code for implementation:

Safe function: Returns true if the position is safe for queen. Returns false otherwise

```cpp

bool safe(vector<int>& board, int row, int col, int n) {
    for (int i = 0; i < row; i++) {
        if (board[i] == col) {
            return false;
        }
    }

    for (int i = row - 1,j = col - 1; i >= 0 && j >= 0; i--, j--) {
        if (board[i] == j) {
            return false;
        }
    }

    for (int i = row - 1, j = col + 1; i >= 0 && j < n; i--, j++) {
        if (board[i] == j) {
            return false;
        }
    }

    return true;
}

```

soolveNQueens function: solves the problem

```cpp
bool solveNQueens(vector<int>& board, int row, int n) {
    if (row == n) {
        return true;
    }
    for (int col = 0; col < n; col++) {
        if (safe(board, row, col, n)) {
            board[row] = col;

            if (solveNQueens(board, row + 1, n)) {
                return true;
            }

            board[row] = -1;
        }
    }

    return false;
}
```

Time Complexity:

O(N!) - worst case (Using Backtracking)

because we try many combinations

Space Complexity:

O(N)

as we store queen positions only for one row at a time

Conclusion:

In this lab, we learned about backtracking and solved a problem named N-Queen problem with this technique. Backtracking is helpful when we need to trypossiblities and get back if we are wrong. N-Queen problem is a very good example to understand how backtracking works in real life. By implementing the solution, we understood how careful checking and step-by-step building helps us reach the correct answer.There are many more examples of backtracking problems like sum of subarray, knights tour etc. All these problems can be solved easily using backtracking method.

Reference :

1. Cormen, Leiserson, Rivest, and Stein, *Introduction to Algorithms*, MIT Press.
2. GeeksforGeeks: https://www.geeksforgeeks.org/n-queen-problem-backtracking-3/
3. Programiz: https://www.programiz.com/dsa/backtracking
4. Class Lecture

Submitted By,

Name : Saleh Sadid Mir
Roll : 2207024
Batch : 2k22
Group : A1
Year : Second
Term : Second

Date of performance : 05-11-25
Date of Submission : 05-12-25