/\*

    Name : Ayush Pandey

    Roll No : 3317

    ASSIGNMENT-4

    Problem Statement :

        Implement a solution for a Constraint Satisfaction Problem using Branch

        and Bound and Backtracking for n-queens problem or a graph coloring

        problem.

\*/

#include <iostream>

#include <vector>

#include <cstring>

using namespace std;

#define MAX\_N 20 // Maximum board size

// Function to print the solution board

void printSolution(vector<vector<int>> &board, int N) {

    for (int i = 0; i < N; i++) {

        for (int j = 0; j < N; j++) cout << (board[i][j] ? "Q " : ". ");

        cout << endl;

    }

    cout << endl;

}

// ---------------------------- BACKTRACKING SOLUTION ----------------------------

// Function to check if a queen can be placed safely

bool isSafeBacktracking(vector<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, j = col; i >= 0 && j >= 0; i--, j--) if (board[i][j]) return false;

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

    return true;

}

// Backtracking function to solve N-Queens

bool solveNQueensBacktracking(vector<vector<int>> &board, int row, int N) {

    if (row == N) {

        printSolution(board, N); return true;

    }

    bool foundSolution = false;

    for (int col = 0; col < N; col++) if (isSafeBacktracking(board, row, col, N)) {

            board[row][col] = 1;

            foundSolution |= solveNQueensBacktracking(board, row + 1, N);

            board[row][col] = 0; // Backtrack

        }

    return foundSolution;

}

// ---------------------------- BRANCH & BOUND SOLUTION ----------------------------

bool solveNQueensBranchBound(int col, int N, vector<int> &leftRow, vector<int> &upperDiag, vector<int> &lowerDiag, vector<vector<int>> &board) {

    if (col == N) {

        printSolution(board, N); return true;

    }

    bool foundSolution = false;

    for (int row = 0; row < N; row++)

        if (!leftRow[row] && !upperDiag[row + col] && !lowerDiag[row - col + N - 1]) {

            board[row][col] = 1;

            leftRow[row] = upperDiag[row + col] = lowerDiag[row - col + N - 1] = 1;

            foundSolution |= solveNQueensBranchBound(col + 1, N, leftRow, upperDiag, lowerDiag, board);

            board[row][col] = 0; // Backtrack

            leftRow[row] = upperDiag[row + col] = lowerDiag[row - col + N - 1] = 0;

        }

    return foundSolution;

}

// ---------------------------- MAIN FUNCTION ----------------------------

int main() {

    int choice, N;

    do {

        cout << "\n===== N-Queens Problem Solver =====\n";

        cout << "1. Solve using Backtracking\n";

        cout << "2. Solve using Branch & Bound\n";

        cout << "3. Exit\n";

        cout << "Enter your choice: ";

        cin >> choice;

        if (choice == 3) {

            cout << "Exiting program.\n";

            break;

        }

        cout << "Enter value of N (board size): "; cin >> N;

        if (N < 1 || N > MAX\_N) {

            cout << "Invalid input! Please enter N between 1 and " << MAX\_N << ".\n";

            continue;

        }

        vector<vector<int>> board(N, vector<int>(N, 0));

        if (choice == 1) {

            cout << "\nSolving using Backtracking:\n";

            if (!solveNQueensBacktracking(board, 0, N))

                cout << "No solution found for N = " << N << " using Backtracking.\n";

        }

        else if (choice == 2) {

            cout << "\nSolving using Branch & Bound:\n";

            vector<int> leftRow(N, 0), upperDiag(2 \* N - 1, 0), lowerDiag(2 \* N - 1, 0);

            if (!solveNQueensBranchBound(0, N, leftRow, upperDiag, lowerDiag, board))

                cout << "No solution found for N = " << N << " using Branch & Bound.\n";

        }

        else cout << "Invalid choice! Please select a valid option.\n";

    } while (choice != 3);

    return 0;

}

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.