Aim-4-4. N-Queens II

#include <iostream>

#include <vector>

class NQueens {

public:

int totalNQueens(int n) {

std::vector<int> board(n, -1); // board[i] stores the column index for the queen in row i

int solutions = 0;

solve(0, n, board, solutions);

return solutions;

}

private:

void solve(int row, int n, std::vector<int>& board, int& solutions) {

if (row == n) { // All queens are placed successfully

solutions++;

return;

}

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

if (isSafe(row, col, board)) {

board[row] = col; // Place the queen

solve(row + 1, n, board, solutions); // Recurse for the next row

board[row] = -1; // Backtrack

}

}

}

bool isSafe(int row, int col, const std::vector<int>& board) {

for (int i = 0; i < row; ++i) {

// Check for queens in the same column or diagonals

if (board[i] == col || abs(board[i] - col) == abs(i - row)) {

return false;

}

}

return true;

}

};

int main() {

int n;

std::cout << "Enter the value of n: ";

std::cin >> n;

NQueens solver;

int result = solver.totalNQueens(n);

std::cout << "The number of distinct solutions for " << n << "-Queens is: " << result << "\n";

return 0;

}