Name: Satyam Jaiswal **UID:** 2021600028 **Batch**: B **Date:** 26/09/2023

EXP 5: Hill Climbing

Problem: Implement Random restart hill climbing to solve N queens problem.

1-show no of restarts required to solve the problem

2- show intermediate heuristics at every stage

3- show no of steps required to get the final solution

Program:

```
#include <iostream>
#include <vector>
#include <cmath>
#include <climits>
using namespace std;

vector<vector<int>> initial_board;
vector<vector<int>> final_board;
vector<int> queen;
vector<int> minimum_index = {-1, -1};
int minimum = INT_MAX;

void print_board(const vector<vector<int>>& board) {
   cout << "\n";
   for (const vector<int>& row : board) {
      for (int val : row) {
         cout << val << " ";
      }
      cout << endl;
   }
}
```

```
int count_clashes(const vector<vector<int>>& board) {
    int clashes = 0;
    for (int row = 0; row < board.size(); ++row) {
        for (int col = 0; col < board[row].size(); ++col) {
            if (board[row][col] == 1) {
                for (int i = 0; i < board.size(); ++i) {
                    if (i != row) {
                        if (board[i][col] == 1) {
                            clashes += 1;
                        }
                        int diff = abs(row - i);
                        if (col - diff >= 0 && board[i][col - diff] == 1) {
                            clashes += 1;
                        }
                        if (col + diff < board.size() && board[i][col + diff] == 1) {
                            clashes += 1;
                        }
                    }
                }
            }
        }
    }
    return clashes / 2;
}

vector<vector<int>> transpose(const vector<vector<int>>& matrix) {
    int rows = matrix.size();
    int cols = matrix[0].size();
```

```
    vector<vector<int>> result(cols, vector<int>(rows, 0));

    for (int i = 0; i < rows; ++i) {
        for (int j = 0; j < cols; ++j) {
            result[j][i] = matrix[i][j];
        }
    }

    return result;
}

int clashes(int i, int j, vector<int>& queen_copy, vector<vector<int>>& board_copy) {
    board_copy[queen_copy[i]][i] = 0;
    board_copy[j][i] = 1;
    queen_copy[i] = j;
    return count_clashes(transpose(board_copy));
}

int heuristic(const vector<int>& queen, const vector<vector<int>>& board) {
    int minimum = INT_MAX;
    for (int i = 0; i < queen.size(); ++i) {
        for (int j = 0; j < queen.size(); ++j) {
            vector<int> queen_copy = queen;
            vector<vector<int>> board_copy = board;
            final_board[j][i] = clashes(i, j, queen_copy, board_copy);
            if (minimum > final_board[j][i]) {
                minimum = final_board[j][i];
                minimum_index[0] = i;
                minimum_index[1] = j;
```

```
            }
        }
    }
    return minimum;
}

int main() {
    queen = {1, 3, 1, 3};
    int n = queen.size();
    initial_board = vector<vector<int>>(n, vector<int>(n, 0));
    final_board = initial_board;

    for (int i = 0; i < n; ++i) {
        initial_board[queen[i]][i] = 1;
    }

    print_board(initial_board);
    int iteration = 0;
    while (minimum != 0) {
        iteration += 1;
        minimum = heuristic(queen, initial_board);
        cout << "\n---------- ITERATION " << iteration << " ----------" << endl;
        cout << "\nminimum value is " << minimum << " at index (" << minimum_index[1]
<< ", " << minimum_index[0] << ")" << endl;
        cout << "\nBoard heuristic" << endl;
        print_board(final_board);
        cout << "\n";
        for (int j = 0; j < n; ++j) {
            initial_board[queen[j]][minimum_index[0]] = 0;
        }
```

```
        initial_board[minimum_index[1]][minimum_index[0]] = 1;
        queen[minimum_index[0]] = minimum_index[1];
        cout << "Final State" << endl;
        print_board(initial_board);
    }
    return 0;
}
```

Output:

0 0 0 0

1 0 1 0

0 0 0 0

0 1 0 1

---------- ITERATION 1 ----------

minimum value is 1 at index (0, 2)

Board heuristic

2 3 1 2

2 4 2 4

2 3 3 2

4 2 4 2

Final State

0 0 1 0

1 0 0 0

0 0 0 0

0 1 0 1

---------- ITERATION 2 ----------

minimum value is 0 at index (2, 3)

Board heuristic

3 2 1 1

1 3 2 3

3 1 3 0

3 1 4 1

Final State

0 0 1 0

1 0 0 0

0 0 0 1

0 1 0 0