**PUNE INSTITUTE OF COMPUTER TECHNOLOGY**
**DHANKAWADI, PUNE**

ARTIFICIAL INTELLIGENCE ROBOTICS MINI-PROJECT
REPORT ON

# "N-QUEENS PROBLEM USING HILL CLIMBING RANDOM RESTART ALGORITHM"

**SUBMITTED BY**


| | |
|---|---|
| Atharva Shastri | 41161 |
| Prathamesh Thombre | 41172 |
| Prathamesh Sonawane | 41166 |

**Under the guidance of**
Prof. Parag Jambhulkar



DEPARTMENT OF COMPUTER ENGINEERING
**Academic Year 2021-22**

# Contents

# 1 Problem Statement

The problem formulation consists of finding an arrangement of n queens on a nxn board such that they do not pose any threat to any other queen on the board.

# 2 Abstract

Hill climbing algorithm is a local search algorithm which continuously moves in the direction of increasing elevation/value to find the peak of the mountain or best solution to the problem. It terminates when it reaches a peak value where no neighbor has a higher value. Hill climbing algorithm is a technique which is used for optimizing the mathematical problems. One of the widely discussed examples of Hill climbing algorithm is Traveling- salesman Problem in which we need to minimize the distance traveled by the salesman.

In this algorithm, we don't need to maintain and handle the search tree or graph as it only keeps a single current state. Nqueens can be solved by different approaches like BackTracking,Bit-Masking,Parallel Implementation of Bit-Masking,Hill-Climbing etc. In this miniproject we are solving it using Hill-Climbing approach.

# 3 Hardware and Software Requirements

## 3.1 Hardware Requirements

1. 500 GB HDD
2. 4GB RAM
3. Monitor
4. Keyboard

## 3.2 Software Requirements

1. 64 bit Open Source Operating System like Ubuntu 18.04
2. Java
3. Eclipse
4. JDK

# 4 INTRODUCTION

- Hill Climbing with random restart is a local search algorithm. It is an iterative algorithm that starts with an arbitrary solution to a problem, then attempts to find a better solution by incrementally changing a single element of the solution.
- If the change produces a better solution, an incremental change is made to the new solution, repeating until no further improvements can be found.
- The algorithm does not maintain a search tree, so the data structure for the current node need only record the state and the value of the objective function.
- The hill-climbing algorithms often fail to find a goal when one exists because they can get stuck on local maxima.
- Random-restart can be used to solve the problem of local maxima, as it conducts a series of hill-climbing searches from randomly generated initial states, until a goal is found.

# 5 OBJECTIVE

- Learn to implement Hill Climbing Algorithm to solve N-Queens Problem
- To build system that will take number of queens as input and predict steps required to generate actual n*n board.

# 6 Scope

- In this we are finding out the board state for given number of queens.
- The scope of this is to check teh performance of the hill climbing algorithm for n-queens problem in comparison to Parallel-Implementation and backtracking.

# 7 System Architecture

system consists of two Classes:

1. HillClimbingRandomRestart: The main class which has the hill climbing algorithm with random restart and the main method of the program.

2. NQueen: The NQueen Class is used to represent the N-queens state.

NQueen Class:

The NQueen class is used to represent the N-queens state. It has the following components:

- row: Current row of the queen
- column: Current column of the queen The class has getter methods for both components and a parameterized constructor. This class has following methods:
- ifConflict(NQueen q): This method checks if the queen has any conflicts or not and returns true if there are conflicts otherwise returns false.
- move(): Move the queen one row down

The HillClimbingRandomRestart class is the main class which accepts the input from user, contains the Hill climbing with random restart algorithm logic and prints the desired output to the console.

This class has the following methods:

1. Main() : This accepts the input from user and checks if the input is valid, then generate an initial board randomly and then perform hill climbing. Finally print the number of steps climbed and total number of random restarts used.

2. generateBoard(): This method generates a board and fills the n-queens randomly. This is used at the beginning to generate the initial board and is called for each random restart thereafter.

3. printState( Nqueen[] State): This is a helper method to print the current configuration.

4. findHeuristic( NQueen[] State): This method finds the number of conflicts of the queen.

5. nextBoard (NQueen[] presentBoard): This method finds the next configuration i.e. the step to be climbed if there is any.

# 8 Test Cases

Enter the number of Queens : 4 Solution to 4 queens using hill climbing with random restart:
Heuristic Value : 5

0 0 0 0
1 1 0 1
0 0 1 0
0 0 0 0

Heuristic Value : 2

0 0 0 1
1 1 0 0
0 0 1 0
0 0 0 0

Heuristic Value : 1

0 0 0 1
1 0 0 0
0 0 1 0
0 1 0 0

Heuristic Value : 3

0 0 0 0
0 0 1 1
1 0 0 0
0 1 0 0

Heuristic Value : 2

1 0 0 0
0 0 1 1
0 0 0 0
0 1 0 0

Heuristic Value : 1

1 0 0 0
0 0 1 0
0 0 0 1
0 1 0 0

Heuristic Value : 4

1 1 1 0
0 0 0 1
0 0 0 0
0 0 0 0

Heuristic Value : 1

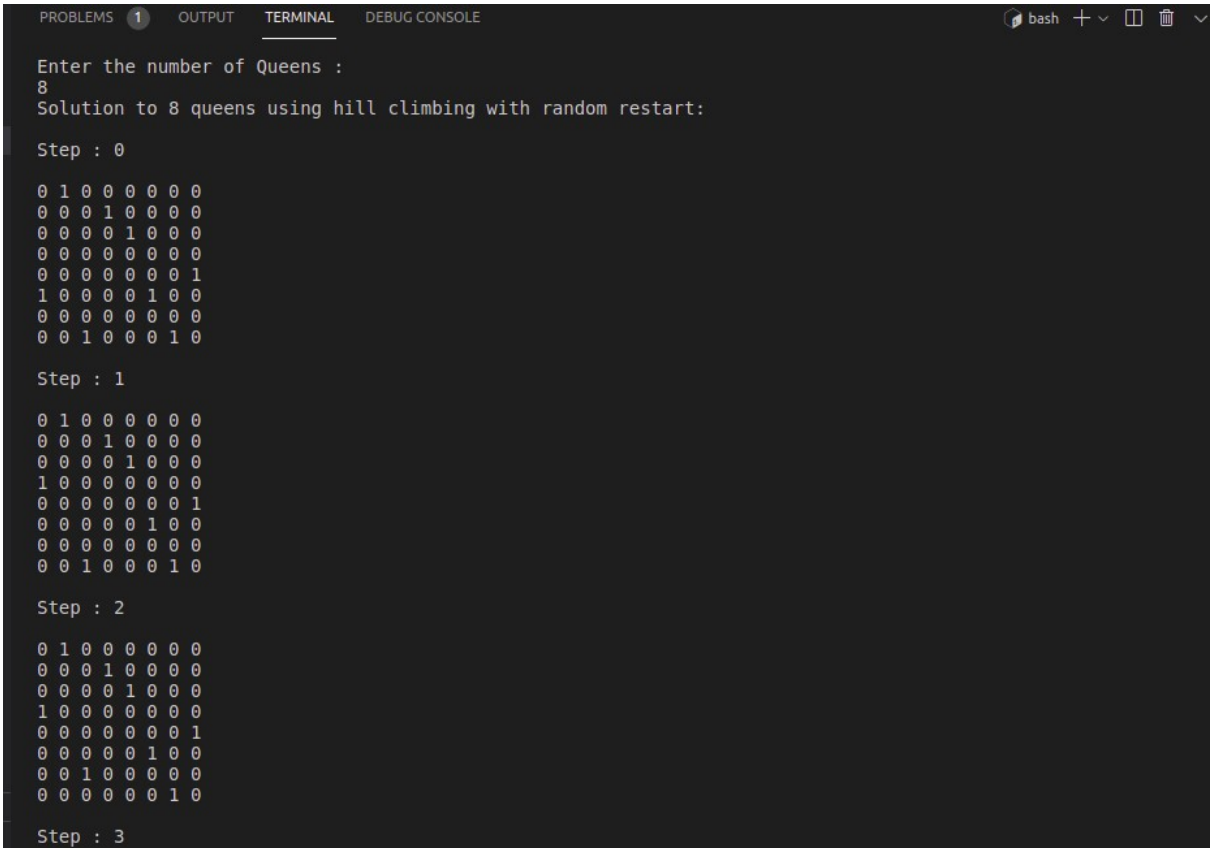
1 1 0 0
0 0 0 1
0 0 0 0
0 0 1 0

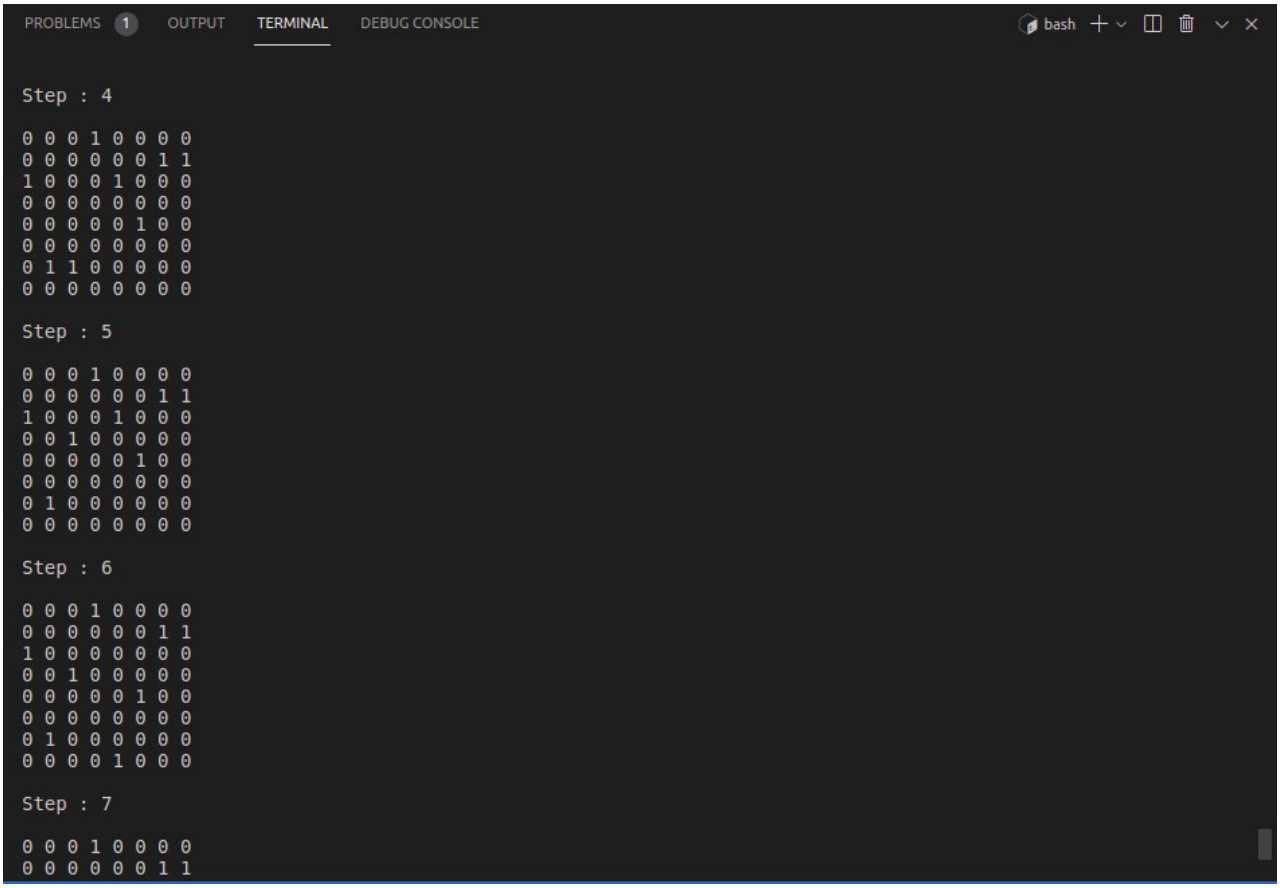
Final Solution
0 1 0 0
0 0 0 1
1 0 0 0
0 0 1 0

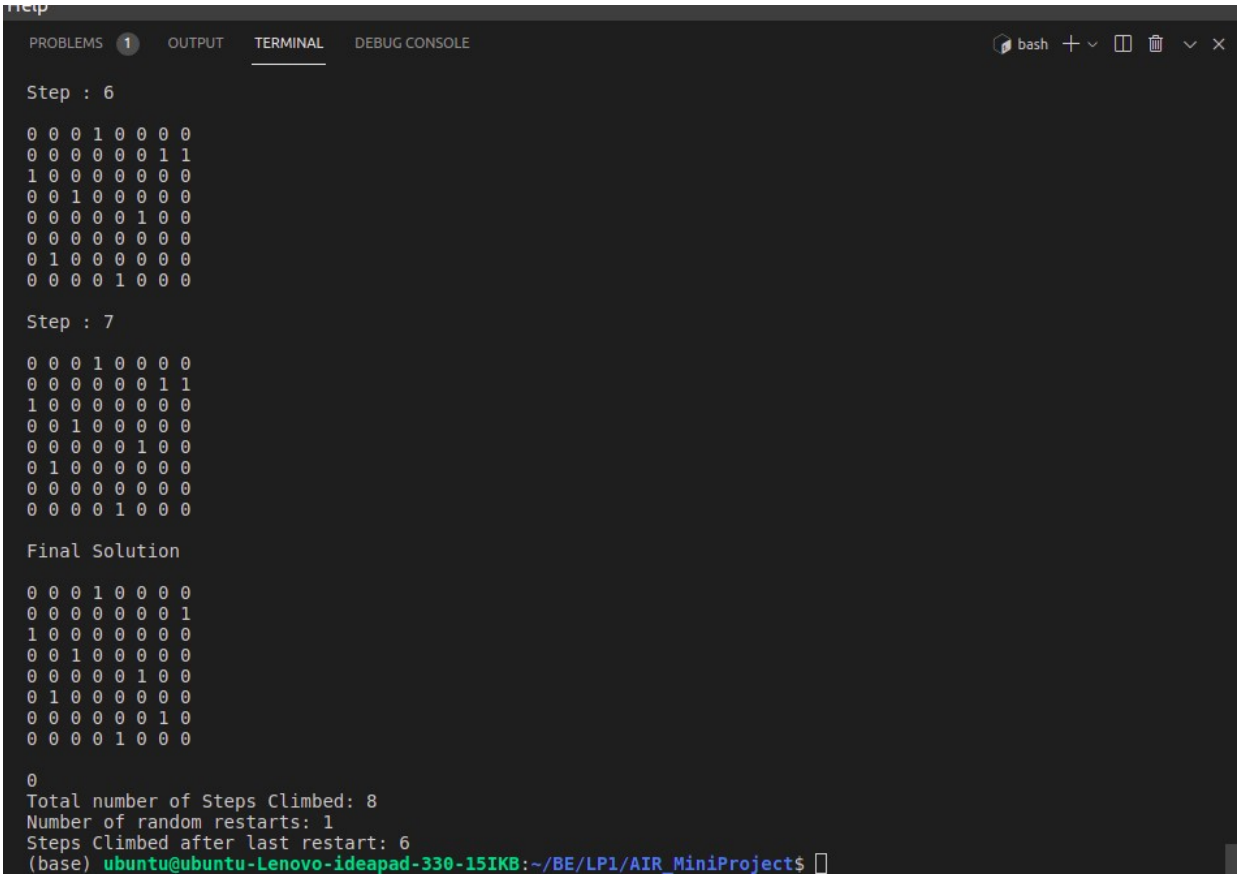
Total number of Steps Climbed: 8
Number of random restarts: 2
Steps Climbed after last restart: 3

# 9 Result

For n = 8




PROBLEMS 1
OUTPUT
TERMINAL
DEBUG CONSOLE
bash
Step : 4
0 0 0 1 0 0 0 0
0 0 0 0 0 0 1 1
1 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0
0 1 1 0 0 0 0 0
0 0 0 0 0 0 0 0
Step : 5
0 0 0 1 0 0 0 0
0 0 0 0 0 0 1 1
1 0 0 0 1 0 0 0
0 0 1 0 0 0 0 0
0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0
Step : 6
0 0 0 1 0 0 0 0
0 0 0 0 0 0 1 1
1 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0
0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0
0 0 0 0 1 0 0 0
Step : 7
0 0 0 1 0 0 0 0
0 0 0 0 0 0 1 1


PROBLEMS 1
OUTPUT
TERMINAL
DEBUG CONSOLE
bash
Step : 6
0 0 0 1 0 0 0 0
0 0 0 0 0 0 1 1
1 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0
0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0
0 0 0 0 1 0 0 0
Step : 7
0 0 0 1 0 0 0 0
0 0 0 0 0 0 1 1
1 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0
0 0 0 0 0 1 0 0
0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0
Final Solution
0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0
0 0 0 0 0 1 0 0
0 1 0 0 0 0 0 0
0 0 0 0 0 0 1 0
0 0 0 0 1 0 0 0
0
Total number of Steps Climbed: 8
Number of random restarts: 1
Steps Climbed after last restart: 6
(base) ubuntu@ubuntu-Lenovo-ideapad-330-15IKB:~/BE/LP1/AIR_MiniProject$

# 10 Conclusion

Thus , we have successfully implemented NQueens Problem using hill climbing algorithm and have understood it properly.