Program 2

Aravind Kumar Reddy Yempada
ITCS 6150
800877797

Program Details

The program has been developed in C# using the Visual Studio IDE. It contains four .cs files:

- Program.cs
- NQueensProblem.cs
- HillClimbing.cs
- CSP.cs

Program.cs contains the Main() method from which the execution starts. Main() method accepts the input from user which is the size of the N queens problem. It then creates an object of NQueensProblem class and calls the methods in HillClimbing and CSP classes to find the solution to the n-queens problem. It then displays the solution, number of random initializations and number of state changes to reach the solution.

The NQueensProblem class represents a state of n-queens problem. It contains the following properties and methods:

- Size – stores the size of the n queens problem
- BoardConfig – stores the current configuration of the board
- QueenPositions – it is an 'n' size array that stores the position of the queen for each row. It is used as the array to represent the Variables for the Constraint Satisfaction Problem.
- Conflicts – it gives the number of conflicts for the current board configuration
- NQueensProblem() – constructor which gets the size of the problem from the Program.cs and creates the BoardConfig and QueenPositions based on that size
- InitializeBoard() – Method to randomly initialize the Board
- Successors() – Method that generates the successor states. It calls the Operations() method to generate each state and stores the states in a list
- HCost() – calculates the heuristic cost for the current board configuration which is the number of conflicts between the queens.
- PrintState() – prints state on the console

HillClimbing.cs has the implementation of hill climbing approach to solve the n-queens problem. The pseudo code is below:

```
function HILL-CLIMBING(problem) returns a state that is a local maximum
   current ← MAKE-NODE(problem.INITIAL-STATE)
   loop do
      neighbor ← a highest-valued successor of current
      if neighbor.VALUE ≤ current.VALUE then return current.STATE
      current ← neighbor
```

Source: http://aima.cs.berkeley.edu/algorithms.pdf

Hill-Climbing approach get stuck when it reaches a local maximum or plateau or ridge, random initialization is used whenever such scenario occurs.

CSP.cs has the implementation of Constraint Satisfaction Problem approach to solve the n-queens problem. Two constraints are used to determine the next state:

- Each row has one and only one queen
- There are no conflicts between the queens

The pseudo code for the CSP approach is below:

```
function MIN-CONFLICTS(csp, max_steps) returns a solution or failure
   inputs: csp, a constraint satisfaction problem
           max_steps, the number of steps allowed before giving up

   current ← an initial complete assignment for csp
   for i = 1 to max_steps do
      if current is a solution for csp then return current
      var ← a randomly chosen conflicted variable from csp.VARIABLES
      value ← the value v for var that minimizes CONFLICTS(var, v, current, csp)
      set var = value in current
   return failure
```

Source: http://aima.cs.berkeley.edu/algorithms.pdf

CSP approach gets stuck when no variable (i.e., queen) can be changed to a value (i.e., a position within its column) that will reduce the conflicts, random initialization is used whenever such scenario occurs.

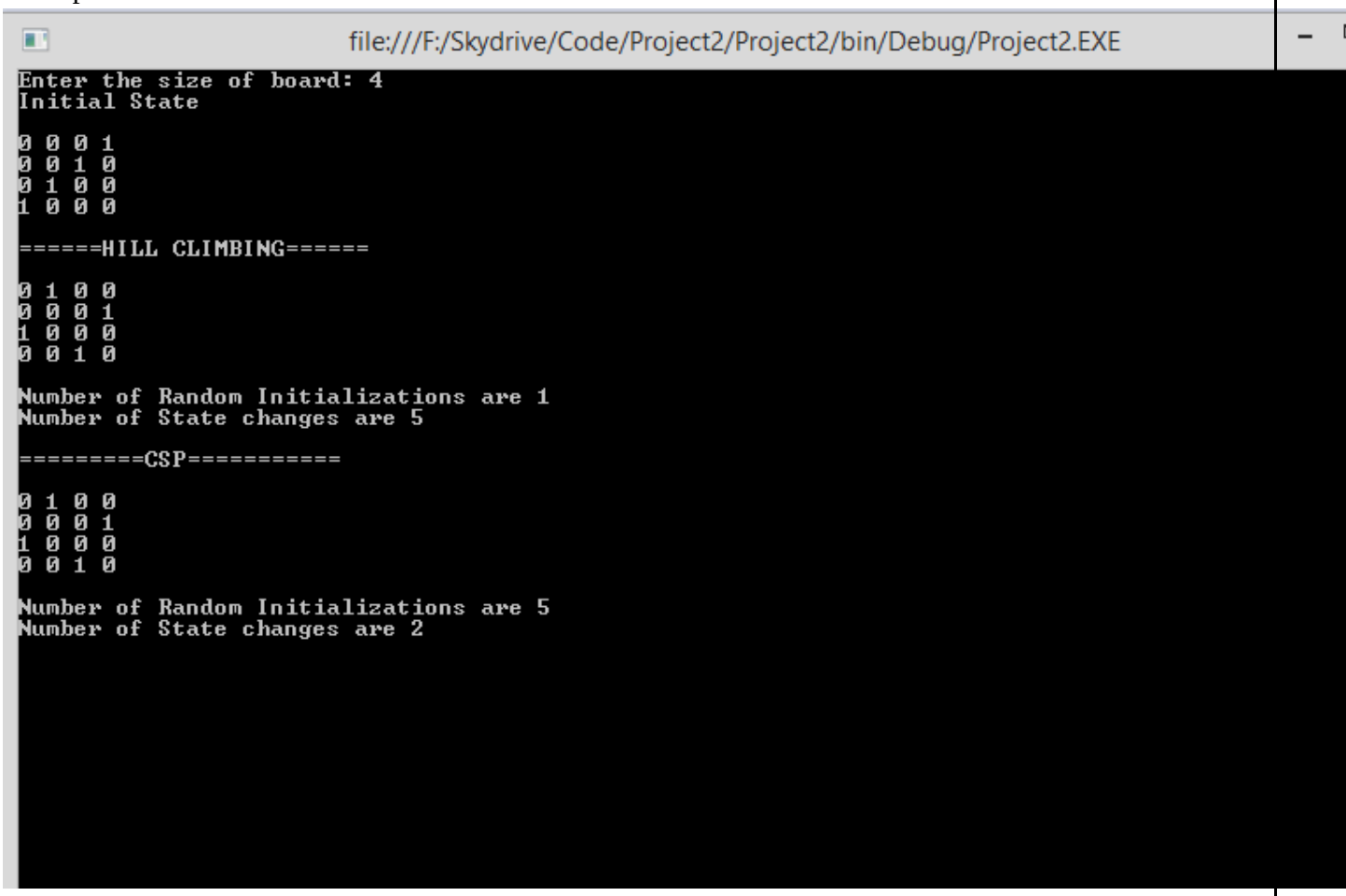
Following two global variables are used in the program:

- numberOfInits – keeps track of number of random initializations
- numberOfSteps – keeps track of number of state changes

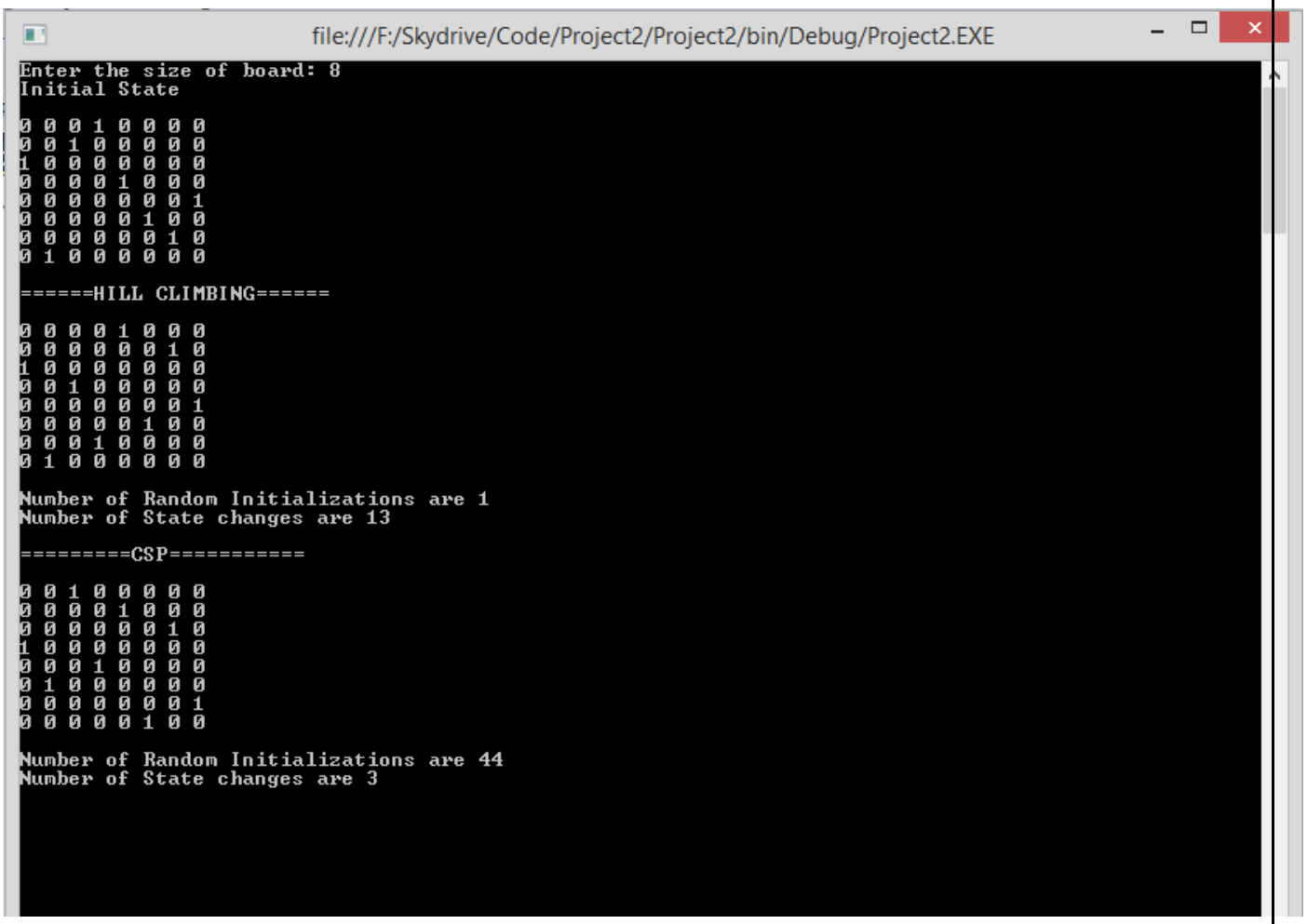
On execution the program asks for user to enter the size of the problem. It accepts only numbers. The program displays solution for Hill Climbing and CSP approaches with the number of random initializations and number of state changes.

Execution Screenshots

Example I



Example II




```
Enter the size of board: 8
Initial State

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

======HILL CLIMBING======

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

Number of Random Initializations are 1
Number of State changes are 13

=========CSP===========

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

Number of Random Initializations are 44
Number of State changes are 3
```

Example III

```
file:///F:/Skydrive/Code/Project2/Project2/bin/Debug/Project2.EXE
Enter the size of board: 20
Initial State

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

======HILL CLIMBING======

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

Number of Random Initializations are 1
Number of State changes are 80
```

```
=========CSP===========

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

Number of Random Initializations are 1
Number of State changes are 12
```

Comparison between Hill Climbing and CSP:

Size	**Hill Climbing**	**CSP**
	Number of State Changes	**Number of State Changes**
4	5	2
5	4	5
6	3	3
7	15	46
8	13	3
9	10	2
10	55	20
11	18	28
12	48	65
13	44	56
14	10	12
15	15	43
20	80	12
25	34	47
30	71	71
35	26	61
40	34	76
45	38	41
50	53	56

The average number of steps in CSP are greater than the average steps in Hill Climbing but the execution time of Hill Climbing increases rapidly with the increase in the size of the problem since we have generate (n* n-1) successors to get the next state. Whereas in CSP, the execution time increases slowly as we only generate n-1 states to get the next state. So the CSP approach is efficient than the Hill Climbing approach.