ITCS-6150 (Intelligent Systems)
Dr. Dewan T. Ahmed

N-Queen using Hill Climbing

Submitted By:
Hemanth Gaddipati
Mounika Rayana
Pradeep Chelamala
Durgesh Bharatbhai Patel

Problem Formulation:

The N-queens problem is the problem of placing ‘n’ chess queens on an n * n chessboard so that no two queens are attacking each other. This means no queen can be in the same row, column or diagonal. We can find the solutions for all the natural numbers except for n =2 or 3. Here in this report, we are choosing to solve the 8 queens problem by taking a random state by placing 8 queens in the 8*8 chessboard by placing each queen in a column.

There are different types of hill-climbing search techniques that can be used to solve this problem. The general hill-climbing search has less percent of success that is around 14%. So, to optimize this search there are a couple of updated searches like Hill Climbing sideways move and Random Restart Hill Climbing.

- Hill climbing search
- Hill climbing search with sideways movement allowed
- Random Restart hill climbing search
- Random restart hill climbing search with sideways movement allowed

Hill Climbing with a sideways move: This is an optimized version of the regular hill-climbing search algorithm. When a local minimum is reached, continuing search by non-improving “sideways” moves will lead to a significant improvement in the performance of the algorithm.
Random Restart Hill Climbing: This is built on top of the hill-climbing search algorithm.
It iteratively does hill-climbing, each time with a random initial condition. The best state is kept; if a new run of hill-climbing produces a better state than the store state, it replaces the stored state. This is the most effective algorithm in most of the cases.

We are using a heuristic function to determine the steps each queen takes. The heuristic cost function h calculates the number of pairs of queens that are attacking each other, either directly or indirectly.

Code Structure:

Class 1: HillClimbing

1. cells_at_state will return cells with queens in the given state.
2. print_Nqueen_matrix will print n queen state as a matrix
3. Horizontal_cells_right_to_current return the cells to the horizontal right of the current cell
4. diagonal_cells_right_to_current return the cells to the diagonal right of the current cell
5. total_cells_to_the_right will return all the horizontal and diagonal cells to the right of current cells
6. heuristic_value returns heuristic value for a given state (h calculates the number of pairs of queens that are attacking each other, either directly or indirectly)
7. heuristic_matrix calculates heuristic values of each cell and returns heuristic value matrix, least heuristic and numpy array with row and column with least heuristic
8. randon_state creates and returns a random state which will be used in random restart function.
9. hill_climbing_search function is a recursive implementation of the hill climbing search using steepest ascent. this method will return result and step towards the least heuristic value at each recursion and the result would contain flat local maxima, local maxima and success
10. hill_climbing_random_restart function implements random restart algorithm.

Class 2: project_analysis

1. analysis function will start iterating and performs hill climbing and randon-restart hill climbing with and without sideways movement.
2. print_results function prints stats of all 4 algorithms.
3. print_hillclimbing_stats will print report of the hill climbing search with and without sideways movement.
4. print_random_restart_stats will print report of the random restart hill climbing search with and without sideways movement.

Get N, Iterations and sideways movement as input from the user and run the above classes.

Sample Initial and Final configurations:

```
Please enter a value for N(number of queens): 5
Please enter a value for number of iterations: 100
Please enter a value for the maximum sideways moves allowed: 10
Hill climbing Search Analysis
Initial state is:
[(3, 0), (3, 1), (4, 2), (3, 3), (2, 4)]
|_|_|_|_|_|
|_|_|_|_|_|
|_|_|_|_|Q|
|Q|Q|_|Q|_|
|_|_|Q|_|_|
Step is:  2
[(3, 0), (0, 1), (4, 2), (3, 3), (2, 4)]
|_|Q|_|_|_|
|_|_|_|_|_|
|_|_|_|_|Q|
|Q|_|_|Q|_|
|_|_|Q|_|_|
Step is:  3
[(3, 0), (0, 1), (4, 2), (3, 3), (1, 4)]
|_|Q|_|_|_|
|_|_|_|_|Q|
|_|_|_|_|_|
|Q|_|_|Q|_|
|_|_|Q|_|_|
Step is:  4
[(3, 0), (0, 1), (4, 2), (1, 3), (1, 4)]
|_|Q|_|_|_|
|_|_|_|Q|Q|
|_|_|_|_|_|
|Q|_|_|_|_|
|_|_|Q|_|_|
The Search has failed.
Hill climbing Search with sideways movement Analysis
Initial state is:
[(3, 0), (3, 1), (4, 2), (3, 3), (2, 4)]
|_|_|_|_|_|
|_|_|_|_|_|
|_|_|_|_|Q|
|Q|Q|_|Q|_|
|_|_|Q|_|_|
Step is:  2
```

```
[(3, 0), (0, 1), (4, 2), (3, 3), (2, 4)]
|_|Q|_|_|_|
|_|_|_|_|_|
|_|_|_|_|Q|
|Q|_|_|Q|_|
|_|_|Q|_|_|
Step is:  3
[(3, 0), (0, 1), (4, 2), (4, 3), (2, 4)]
|_|Q|_|_|_|
|_|_|_|_|_|
|_|_|_|_|Q|
|Q|_|_|_|_|
|_|_|Q|Q|_|
Step is:  4
[(3, 0), (0, 1), (4, 2), (4, 3), (1, 4)]
|_|Q|_|_|_|
|_|_|_|_|Q|
|_|_|_|_|_|
|Q|_|_|_|_|
|_|_|Q|Q|_|
Successfully finished:
[(3, 0), (0, 1), (2, 2), (4, 3), (1, 4)]
|_|Q|_|_|_|
|_|_|_|_|Q|
|_|_|Q|_|_|
|Q|_|_|_|_|
|_|_|_|Q|_|
Random restart hill climbing search Analysis
Random-Restart start_state:  [3, 4, 2, 4, 1]
Initial state is:
[(3, 0), (4, 1), (2, 2), (4, 3), (1, 4)]
|_|_|_|_|_|
|_|_|_|_|Q|
|_|_|Q|_|_|
|Q|_|_|_|_|
|_|Q|_|Q|_|
Successfully finished:
[(3, 0), (0, 1), (2, 2), (4, 3), (1, 4)]
|_|Q|_|_|_|
|_|_|_|_|Q|
|_|_|Q|_|_|
|Q|_|_|_|_|
|_|_|_|Q|_|
Random restart hill climbing search with sideways movement Analysis
```

```
Random-Restart start_state:  [1, 2, 1, 2, 0]
Initial state is:
[(1, 0), (2, 1), (1, 2), (2, 3), (0, 4)]
|_|_|_|_|Q|
|Q|_|Q|_|_|
|_|Q|_|Q|_|
|_|_|_|_|_|
|_|_|_|_|_|
Step is:  2
[(1, 0), (2, 1), (4, 2), (2, 3), (0, 4)]
|_|_|_|_|Q|
|Q|_|_|_|_|
|_|Q|_|Q|_|
|_|_|_|_|_|
|_|_|Q|_|_|
Step is:  3
[(1, 0), (1, 1), (4, 2), (2, 3), (0, 4)]
|_|_|_|_|Q|
|Q|Q|_|_|_|
|_|_|_|Q|_|
|_|_|_|_|_|
|_|_|Q|_|_|
Successfully finished:
[(3, 0), (1, 1), (4, 2), (2, 3), (0, 4)]
|_|_|_|_|Q|
|_|Q|_|_|_|
|_|_|_|Q|_|
|Q|_|_|_|_|
|_|_|Q|_|_|
Hill climbing Search Analysis
Initial state is:
[(3, 0), (0, 1), (3, 2), (2, 3), (3, 4)]
|_|Q|_|_|_|
|_|_|_|_|_|
|_|_|_|Q|_|
|Q|_|Q|_|Q|
|_|_|_|_|_|
Step is:  2
[(3, 0), (0, 1), (3, 2), (2, 3), (4, 4)]
|_|Q|_|_|_|
|_|_|_|_|_|
|_|_|_|Q|_|
|Q|_|Q|_|_|
|_|_|_|_|Q|
Step is:  3
[(3, 0), (0, 1), (3, 2), (1, 3), (4, 4)]
|_|Q|_|_|_|
|_|_|_|Q|_|
|_|_|_|_|_|
|Q|_|Q|_|_|
|_|_|_|_|Q|
Successfully finished:
[(2, 0), (0, 1), (3, 2), (1, 3), (4, 4)]
|_|Q|_|_|_|
|_|_|_|Q|_|
|Q|_|_|_|_|
|_|_|Q|_|_|
|_|_|_|_|Q|
Hill climbing Search with sideways movement Analysis
Initial state is:
[(3, 0), (0, 1), (3, 2), (2, 3), (3, 4)]
|_|Q|_|_|_|
|_|_|_|_|_|
|_|_|_|Q|_|
|Q|_|Q|_|Q|
|_|_|_|_|_|
Step is:  2
[(3, 0), (0, 1), (3, 2), (2, 3), (4, 4)]
|_|Q|_|_|_|
|_|_|_|_|_|
|_|_|_|Q|_|
|Q|_|Q|_|_|
|_|_|_|_|Q|
Step is:  3
[(3, 0), (0, 1), (3, 2), (1, 3), (4, 4)]
|_|Q|_|_|_|
|_|_|_|Q|_|
|_|_|_|_|_|
|Q|_|Q|_|_|
|_|_|_|_|Q|
Successfully finished:
[(2, 0), (0, 1), (3, 2), (1, 3), (4, 4)]
|_|Q|_|_|_|
|_|_|_|Q|_|
|Q|_|_|_|_|
|_|_|Q|_|_|
|_|_|_|_|Q|
Random restart hill climbing search Analysis
```

```
Random-Restart start_state:  [2, 4, 3, 1, 0]
Initial state is:
[(2, 0), (4, 1), (3, 2), (1, 3), (0, 4)]
|_|_|_|_|Q|
|_|_|_|Q|_|
|Q|_|_|_|_|
|_|_|Q|_|_|
|_|Q|_|_|_|
The Search has failed.
Random-Restart start_state:  [3, 2, 0, 4, 1]
Initial state is:
[(3, 0), (2, 1), (0, 2), (4, 3), (1, 4)]
|_|_|Q|_|_|
|_|_|_|_|Q|
|_|Q|_|_|_|
|Q|_|_|_|_|
|_|_|_|Q|_|
Step is:  2
[(3, 0), (0, 1), (0, 2), (4, 3), (1, 4)]
|_|Q|Q|_|_|
|_|_|_|_|Q|
|_|_|_|_|_|
|Q|_|_|_|_|
|_|_|_|Q|_|
Successfully finished:
[(3, 0), (0, 1), (2, 2), (4, 3), (1, 4)]
|_|Q|_|_|_|
|_|_|_|_|Q|
|_|_|Q|_|_|
|Q|_|_|_|_|
|_|_|_|Q|_|
Random restart hill climbing search with sideways movement Analysis
Random-Restart start_state:  [4, 1, 0, 4, 0]
Initial state is:
[(4, 0), (1, 1), (0, 2), (4, 3), (0, 4)]
|_|_|Q|_|Q|
|_|Q|_|_|_|
|_|_|_|_|_|
|_|_|_|_|_|
|Q|_|_|Q|_|
Step is:  2
[(3, 0), (1, 1), (0, 2), (4, 3), (0, 4)]
|_|_|Q|_|Q|
|_|Q|_|_|_|
|_|_|_|_|_|
|Q|_|_|_|_|
|_|_|_|Q|_|
Step is:  3
[(3, 0), (1, 1), (4, 2), (4, 3), (0, 4)]
|_|_|_|_|Q|
|_|Q|_|_|_|
|_|_|_|_|_|
|Q|_|_|_|_|
|_|_|Q|Q|_|
Successfully finished:
[(3, 0), (1, 1), (4, 2), (2, 3), (0, 4)]
|_|_|_|_|Q|
|_|Q|_|_|_|
|_|_|_|Q|_|
|Q|_|_|_|_|
|_|_|Q|_|_|
Hill climbing Search Analysis
Initial state is:
[(3, 0), (3, 1), (0, 2), (0, 3), (0, 4)]
|_|_|Q|Q|Q|
|_|_|_|_|_|
|_|_|_|_|_|
|Q|Q|_|_|_|
|_|_|_|_|_|
Step is:  2
[(3, 0), (3, 1), (0, 2), (2, 3), (0, 4)]
|_|_|Q|_|Q|
|_|_|_|_|_|
|_|_|_|Q|_|
|Q|Q|_|_|_|
|_|_|_|_|_|
Step is:  3
[(3, 0), (3, 1), (0, 2), (2, 3), (4, 4)]
|_|_|Q|_|_|
|_|_|_|_|_|
|_|_|_|Q|_|
|Q|Q|_|_|_|
|_|_|_|_|Q|
Successfully finished:
[(1, 0), (3, 1), (0, 2), (2, 3), (4, 4)]
|_|_|Q|_|_|
|Q|_|_|_|_|
|_|_|_|Q|_|
|_|Q|_|_|_|
|_|_|_|_|Q|
Hill climbing Search with sideways movement Analysis
Initial state is:
[(3, 0), (3, 1), (0, 2), (0, 3), (0, 4)]
|_|_|Q|Q|Q|
|_|_|_|_|_|
|_|_|_|_|_|
|Q|Q|_|_|_|
|_|_|_|_|_|
Step is:  2
[(3, 0), (3, 1), (0, 2), (2, 3), (0, 4)]
|_|_|Q|_|Q|
|_|_|_|_|_|
|_|_|_|Q|_|
|Q|Q|_|_|_|
|_|_|_|_|_|
Step is:  3
[(3, 0), (3, 1), (0, 2), (2, 3), (4, 4)]
|_|_|Q|_|_|
|_|_|_|_|_|
|_|_|_|Q|_|
|Q|Q|_|_|_|
|_|_|_|_|Q|
Successfully finished:
[(1, 0), (3, 1), (0, 2), (2, 3), (4, 4)]
|_|_|Q|_|_|
|Q|_|_|_|_|
|_|_|_|Q|_|
|_|Q|_|_|_|
|_|_|_|_|Q|
Random restart hill climbing search Analysis
Random-Restart start_state:  [4, 0, 4, 2, 4]
Initial state is:
[(4, 0), (0, 1), (4, 2), (2, 3), (4, 4)]
|_|Q|_|_|_|
|_|_|_|_|_|
|_|_|_|Q|_|
|_|_|_|_|_|
|Q|_|Q|_|Q|
Step is:  2
[(3, 0), (0, 1), (4, 2), (2, 3), (4, 4)]
|_|Q|_|_|_|
|_|_|_|_|_|
|_|_|_|Q|_|
|Q|_|_|_|_|
|_|_|Q|_|Q|
Step is:  3
[(3, 0), (0, 1), (4, 2), (1, 3), (4, 4)]
|_|Q|_|_|_|
|_|_|_|Q|_|
|_|_|_|_|_|
|Q|_|_|_|_|
|_|_|Q|_|Q|
The Search has failed.
Random-Restart start_state:  [2, 1, 0, 0, 0]
Initial state is:
[(2, 0), (1, 1), (0, 2), (0, 3), (0, 4)]
|_|_|Q|Q|Q|
|_|Q|_|_|_|
|Q|_|_|_|_|
|_|_|_|_|_|
|_|_|_|_|_|
Step is:  2
[(2, 0), (1, 1), (3, 2), (0, 3), (0, 4)]
|_|_|_|Q|Q|
|_|Q|_|_|_|
|Q|_|_|_|_|
|_|_|Q|_|_|
|_|_|_|_|_|
The Search has failed.
Random-Restart start_state:  [4, 4, 2, 2, 1]
Initial state is:
[(4, 0), (4, 1), (2, 2), (2, 3), (1, 4)]
|_|_|_|_|_|
|_|_|_|_|Q|
|_|_|Q|Q|_|
|_|_|_|_|_|
|Q|Q|_|_|_|
Step is:  2
[(4, 0), (0, 1), (2, 2), (2, 3), (1, 4)]
|_|Q|_|_|_|
|_|_|_|_|Q|
|_|_|Q|Q|_|
|_|_|_|_|_|
|Q|_|_|_|_|
Step is:  3
[(4, 0), (0, 1), (2, 2), (4, 3), (1, 4)]
```

```
|_|Q|_|_|_|
|_|_|_|_|Q|
|_|_|Q|_|_|
|_|_|_|_|_|
|Q|_|_|Q|_|
Successfully finished:
[(3, 0), (0, 1), (2, 2), (4, 3), (1, 4)]
|_|Q|_|_|_|
|_|_|_|_|Q|
|_|_|Q|_|_|
|Q|_|_|_|_|
|_|_|_|Q|_|
Random restart hill climbing search with sideways movement Analysis
Random-Restart start_state:  [0, 0, 0, 4, 3]
Initial state is:
[(0, 0), (0, 1), (0, 2), (4, 3), (3, 4)]
|Q|Q|Q|_|_|
|_|_|_|_|_|
|_|_|_|_|_|
|_|_|_|_|Q|
|_|_|_|Q|_|
Step is:  2
[(0, 0), (2, 1), (0, 2), (4, 3), (3, 4)]
|Q|_|Q|_|_|
|_|_|_|_|_|
|_|Q|_|_|_|
|_|_|_|_|Q|
|_|_|_|Q|_|
Step is:  3
[(0, 0), (2, 1), (0, 2), (4, 3), (1, 4)]
|Q|_|Q|_|_|
|_|_|_|_|Q|
|_|Q|_|_|_|
|_|_|_|_|_|
|_|_|_|Q|_|
Step is:  4
[(0, 0), (3, 1), (0, 2), (4, 3), (1, 4)]
|Q|_|Q|_|_|
|_|_|_|_|Q|
|_|_|_|_|_|
|_|Q|_|_|_|
|_|_|_|Q|_|
Step is:  5
[(3, 0), (3, 1), (0, 2), (4, 3), (1, 4)]
```

```
|_|_|Q|_|_|
|_|_|_|_|Q|
|_|_|_|_|_|
|Q|Q|_|_|_|
|_|_|_|Q|_|
Step is:  6
[(0, 0), (3, 1), (0, 2), (4, 3), (1, 4)]
|Q|_|Q|_|_|
|_|_|_|_|Q|
|_|_|_|_|_|
|_|Q|_|_|_|
|_|_|_|Q|_|
Step is:  7
[(0, 0), (3, 1), (1, 2), (4, 3), (1, 4)]
|Q|_|_|_|_|
|_|_|Q|_|Q|
|_|_|_|_|_|
|_|Q|_|_|_|
|_|_|_|Q|_|
Successfully finished:
[(0, 0), (3, 1), (1, 2), (4, 3), (2, 4)]
|Q|_|_|_|_|
|_|_|Q|_|_|
|_|_|_|_|Q|
|_|Q|_|_|_|
|_|_|_|Q|_|
Hill climbing Search Analysis
Initial state is:
[(2, 0), (3, 1), (3, 2), (2, 3), (1, 4)]
|_|_|_|_|_|
|_|_|_|_|Q|
|Q|_|_|Q|_|
|_|Q|Q|_|_|
|_|_|_|_|_|
Step is:  2
[(0, 0), (3, 1), (3, 2), (2, 3), (1, 4)]
|Q|_|_|_|_|
|_|_|_|_|Q|
|_|_|_|Q|_|
|_|Q|Q|_|_|
|_|_|_|_|_|
Step is:  3
[(0, 0), (3, 1), (4, 2), (2, 3), (1, 4)]
|Q|_|_|_|_|
|_|_|_|_|Q|
|_|_|_|Q|_|
|_|Q|_|_|_|
|_|_|Q|_|_|
The Search has failed.
Hill climbing Search with sideways movement Analysis
Initial state is:
[(2, 0), (3, 1), (3, 2), (2, 3), (1, 4)]
|_|_|_|_|_|
|_|_|_|_|Q|
|Q|_|_|Q|_|
|_|Q|Q|_|_|
|_|_|_|_|_|
Step is:  2
[(2, 0), (3, 1), (3, 2), (0, 3), (1, 4)]
|_|_|_|Q|_|
|_|_|_|_|Q|
|Q|_|_|_|_|
|_|Q|Q|_|_|
|_|_|_|_|_|
Step is:  3
[(2, 0), (3, 1), (3, 2), (0, 3), (4, 4)]
|_|_|_|Q|_|
|_|_|_|_|_|
|Q|_|_|_|_|
|_|Q|Q|_|_|
|_|_|_|_|Q|
Step is:  4
[(2, 0), (0, 1), (3, 2), (0, 3), (4, 4)]
|_|Q|_|Q|_|
|_|_|_|_|_|
|Q|_|_|_|_|
|_|_|Q|_|_|
|_|_|_|_|Q|
Successfully finished:
[(2, 0), (0, 1), (3, 2), (1, 3), (4, 4)]
|_|Q|_|_|_|
|_|_|_|Q|_|
|Q|_|_|_|_|
|_|_|Q|_|_|
|_|_|_|_|Q|
Random restart hill climbing search Analysis
Random-Restart start_state:  [4, 3, 3, 2, 4]
Initial state is:
```

```
[(4, 0), (3, 1), (3, 2), (2, 3), (4, 4)]
|_|_|_|_|_|
|_|_|_|_|_|
|_|_|_|Q|_|
|_|Q|Q|_|_|
|Q|_|_|_|Q|
Step is:  2
[(4, 0), (3, 1), (0, 2), (2, 3), (4, 4)]
|_|_|Q|_|_|
|_|_|_|_|_|
|_|_|_|Q|_|
|_|Q|_|_|_|
|Q|_|_|_|Q|
Successfully finished:
[(1, 0), (3, 1), (0, 2), (2, 3), (4, 4)]
|_|_|Q|_|_|
|Q|_|_|_|_|
|_|_|_|Q|_|
|_|Q|_|_|_|
|_|_|_|_|Q|
Random restart hill climbing search with sideways movement Analysis
Random-Restart start_state:  [4, 2, 2, 0, 1]
Initial state is:
[(4, 0), (2, 1), (2, 2), (0, 3), (1, 4)]
|_|_|_|Q|_|
|_|_|_|_|Q|
|_|Q|Q|_|_|
|_|_|_|_|_|
|Q|_|_|_|_|
Step is:  2
[(4, 0), (0, 1), (2, 2), (0, 3), (1, 4)]
|_|Q|_|Q|_|
|_|_|_|_|Q|
|_|_|Q|_|_|
|_|_|_|_|_|
|Q|_|_|_|_|
Step is:  3
[(4, 0), (0, 1), (2, 2), (3, 3), (1, 4)]
|_|Q|_|_|_|
|_|_|_|_|Q|
|_|_|Q|_|_|
|_|_|_|Q|_|
|Q|_|_|_|_|
Step is:  4
```

```
[(4, 0), (0, 1), (0, 2), (3, 3), (1, 4)]
|_|Q|Q|_|_|
|_|_|_|_|Q|
|_|_|_|_|_|
|_|_|_|Q|_|
|Q|_|_|_|_|
Successfully finished:
[(4, 0), (2, 1), (0, 2), (3, 3), (1, 4)]
|_|_|Q|_|_|
|_|_|_|_|Q|
|_|Q|_|_|_|
|_|_|_|Q|_|
|Q|_|_|_|_|
Hill climbing Search Analysis
Initial state is:
[(4, 0), (1, 1), (2, 2), (3, 3), (0, 4)]
|_|_|_|_|Q|
|_|Q|_|_|_|
|_|_|Q|_|_|
|_|_|_|Q|_|
|Q|_|_|_|_|
Step is:  2
[(4, 0), (1, 1), (3, 2), (3, 3), (0, 4)]
|_|_|_|_|Q|
|_|Q|_|_|_|
|_|_|_|_|_|
|_|_|Q|Q|_|
|Q|_|_|_|_|
Step is:  3
[(4, 0), (1, 1), (3, 2), (2, 3), (0, 4)]
|_|_|_|_|Q|
|_|Q|_|_|_|
|_|_|_|Q|_|
|_|_|Q|_|_|
|Q|_|_|_|_|
The Search has failed.
Hill climbing Search with sideways movement Analysis
Initial state is:
[(4, 0), (1, 1), (2, 2), (3, 3), (0, 4)]
|_|_|_|_|Q|
|_|Q|_|_|_|
|_|_|Q|_|_|
|_|_|_|Q|_|
|Q|_|_|_|_|
Step is:  2
[(4, 0), (1, 1), (3, 2), (3, 3), (0, 4)]
|_|_|_|_|Q|
|_|Q|_|_|_|
|_|_|_|_|_|
|_|_|Q|Q|_|
|Q|_|_|_|_|
Step is:  3
[(4, 0), (1, 1), (3, 2), (0, 3), (0, 4)]
|_|_|_|Q|Q|
|_|Q|_|_|_|
|_|_|_|_|_|
|_|_|Q|_|_|
|Q|_|_|_|_|
Successfully finished:
[(4, 0), (1, 1), (3, 2), (0, 3), (2, 4)]
|_|_|_|Q|_|
|_|Q|_|_|_|
|_|_|_|_|Q|
|_|_|Q|_|_|
|Q|_|_|_|_|
Random restart hill climbing search Analysis
Random-Restart start_state:  [1, 2, 0, 0, 2]
Initial state is:
[(1, 0), (2, 1), (0, 2), (0, 3), (2, 4)]
|_|_|Q|Q|_|
|Q|_|_|_|_|
|_|Q|_|_|Q|
|_|_|_|_|_|
|_|_|_|_|_|
Step is:  2
[(1, 0), (3, 1), (0, 2), (0, 3), (2, 4)]
|_|_|Q|Q|_|
|Q|_|_|_|_|
|_|_|_|_|Q|
|_|Q|_|_|_|
|_|_|_|_|_|
Step is:  3
[(1, 0), (3, 1), (0, 2), (0, 3), (4, 4)]
|_|_|Q|Q|_|
|Q|_|_|_|_|
|_|_|_|_|_|
|_|Q|_|_|_|
|_|_|_|_|Q|
Successfully finished:
[(1, 0), (3, 1), (0, 2), (2, 3), (4, 4)]
|_|_|Q|_|_|
|Q|_|_|_|_|
|_|_|_|Q|_|
|_|Q|_|_|_|
|_|_|_|_|Q|
Random restart hill climbing search with sideways movement Analysis
Random-Restart start_state:  [4, 0, 0, 1, 1]
Initial state is:
[(4, 0), (0, 1), (0, 2), (1, 3), (1, 4)]
|_|Q|Q|_|_|
|_|_|_|Q|Q|
|_|_|_|_|_|
|_|_|_|_|_|
|Q|_|_|_|_|
Step is:  2
[(4, 0), (0, 1), (0, 2), (3, 3), (1, 4)]
|_|Q|Q|_|_|
|_|_|_|_|Q|
|_|_|_|_|_|
|_|_|_|Q|_|
|Q|_|_|_|_|
Successfully finished:
[(4, 0), (2, 1), (0, 2), (3, 3), (1, 4)]
|_|_|Q|_|_|
|_|_|_|_|Q|
|_|Q|_|_|_|
|_|_|_|Q|_|
|Q|_|_|_|_|
Hill climbing Search Analysis
Initial state is:
[(0, 0), (2, 1), (4, 2), (0, 3), (2, 4)]
|Q|_|_|Q|_|
|_|_|_|_|_|
|_|Q|_|_|Q|
|_|_|_|_|_|
|_|_|Q|_|_|
Step is:  2
[(0, 0), (2, 1), (4, 2), (0, 3), (3, 4)]
|Q|_|_|Q|_|
|_|_|_|_|_|
|_|Q|_|_|_|
|_|_|_|_|Q|
|_|_|Q|_|_|
Successfully finished:
[(0, 0), (2, 1), (4, 2), (1, 3), (3, 4)]
|Q|_|_|_|_|
|_|_|_|Q|_|
|_|Q|_|_|_|
|_|_|_|_|Q|
|_|_|Q|_|_|
Hill climbing Search with sideways movement Analysis
Initial state is:
[(0, 0), (2, 1), (4, 2), (0, 3), (2, 4)]
|Q|_|_|Q|_|
|_|_|_|_|_|
|_|Q|_|_|Q|
|_|_|_|_|_|
|_|_|Q|_|_|
Step is:  2
[(0, 0), (2, 1), (4, 2), (0, 3), (3, 4)]
|Q|_|_|Q|_|
|_|_|_|_|_|
|_|Q|_|_|_|
|_|_|_|_|Q|
|_|_|Q|_|_|
Successfully finished:
[(0, 0), (2, 1), (4, 2), (1, 3), (3, 4)]
|Q|_|_|_|_|
|_|_|_|Q|_|
|_|Q|_|_|_|
|_|_|_|_|Q|
|_|_|Q|_|_|
Random restart hill climbing search Analysis
Random-Restart start_state:  [4, 1, 0, 4, 4]
Initial state is:
[(4, 0), (1, 1), (0, 2), (4, 3), (4, 4)]
|_|_|Q|_|_|
|_|Q|_|_|_|
|_|_|_|_|_|
|_|_|_|_|_|
|Q|_|_|Q|Q|
Step is:  2
[(4, 0), (1, 1), (0, 2), (4, 3), (2, 4)]
|_|_|Q|_|_|
|_|Q|_|_|_|
|_|_|_|_|Q|
|_|_|_|_|_|
|Q|_|_|Q|_|
Step is:  3
[(4, 0), (1, 1), (1, 2), (4, 3), (2, 4)]
|_|_|_|_|_|
|_|Q|Q|_|_|
|_|_|_|_|Q|
|_|_|_|_|_|
|Q|_|_|Q|_|
The Search has failed.
Random-Restart start_state:  [2, 3, 1, 0, 3]
Initial state is:
[(2, 0), (3, 1), (1, 2), (0, 3), (3, 4)]
|_|_|_|Q|_|
|_|_|Q|_|_|
|Q|_|_|_|_|
|_|Q|_|_|Q|
|_|_|_|_|_|
Step is:  2
[(2, 0), (4, 1), (1, 2), (0, 3), (3, 4)]
|_|_|_|Q|_|
|_|_|Q|_|_|
|Q|_|_|_|_|
|_|_|_|_|Q|
|_|Q|_|_|_|
Step is:  3
[(2, 0), (4, 1), (2, 2), (0, 3), (3, 4)]
|_|_|_|Q|_|
|_|_|_|_|_|
|Q|_|Q|_|_|
|_|_|_|_|Q|
|_|Q|_|_|_|
Successfully finished:
[(1, 0), (4, 1), (2, 2), (0, 3), (3, 4)]
|_|_|_|Q|_|
|Q|_|_|_|_|
|_|_|Q|_|_|
|_|_|_|_|Q|
|_|Q|_|_|_|
Random restart hill climbing search with sideways movement Analysis
Random-Restart start_state:  [3, 1, 0, 0, 2]
Initial state is:
[(3, 0), (1, 1), (0, 2), (0, 3), (2, 4)]
|_|_|Q|Q|_|
|_|Q|_|_|_|
|_|_|_|_|Q|
|Q|_|_|_|_|
|_|_|_|_|_|
Step is:  2
[(3, 0), (1, 1), (0, 2), (4, 3), (2, 4)]
|_|_|Q|_|_|
|_|Q|_|_|_|
|_|_|_|_|Q|
|Q|_|_|_|_|
|_|_|_|Q|_|
Step is:  3
[(3, 0), (1, 1), (0, 2), (4, 3), (1, 4)]
|_|_|Q|_|_|
|_|Q|_|_|Q|
|_|_|_|_|_|
|Q|_|_|_|_|
|_|_|_|Q|_|
Step is:  4
[(3, 0), (3, 1), (0, 2), (4, 3), (1, 4)]
|_|_|Q|_|_|
|_|_|_|_|Q|
|_|_|_|_|_|
|Q|Q|_|_|_|
|_|_|_|Q|_|
Step is:  5
[(0, 0), (3, 1), (0, 2), (4, 3), (1, 4)]
|Q|_|Q|_|_|
|_|_|_|_|Q|
|_|_|_|_|_|
|_|Q|_|_|_|
|_|_|_|Q|_|
Step is:  6
[(0, 0), (3, 1), (1, 2), (4, 3), (1, 4)]
|Q|_|_|_|_|
|_|_|Q|_|Q|
|_|_|_|_|_|
|_|Q|_|_|_|
|_|_|_|Q|_|
Successfully finished:
[(0, 0), (3, 1), (1, 2), (4, 3), (2, 4)]
|Q|_|_|_|_|
|_|_|Q|_|_|
|_|_|_|_|Q|
|_|Q|_|_|_|
|_|_|_|Q|_|
Hill climbing Search Analysis
Initial state is:
[(0, 0), (1, 1), (0, 2), (2, 3), (3, 4)]
|Q|_|Q|_|_|
|_|Q|_|_|_|
|_|_|_|Q|_|
|_|_|_|_|Q|
|_|_|_|_|_|
Step is:  2
[(0, 0), (1, 1), (4, 2), (2, 3), (3, 4)]
|Q|_|_|_|_|
|_|Q|_|_|_|
|_|_|_|Q|_|
|_|_|_|_|Q|
|_|_|Q|_|_|
The Search has failed.
Hill climbing Search with sideways movement Analysis
Initial state is:
[(0, 0), (1, 1), (0, 2), (2, 3), (3, 4)]
|Q|_|Q|_|_|
|_|Q|_|_|_|
|_|_|_|Q|_|
|_|_|_|_|Q|
|_|_|_|_|_|
Step is:  2
[(4, 0), (1, 1), (0, 2), (2, 3), (3, 4)]
|_|_|Q|_|_|
|_|Q|_|_|_|
|_|_|_|Q|_|
|_|_|_|_|Q|
|Q|_|_|_|_|
Step is:  3
[(4, 0), (1, 1), (4, 2), (2, 3), (3, 4)]
|_|_|_|_|_|
|_|Q|_|_|_|
|_|_|_|Q|_|
|_|_|_|_|Q|
|Q|_|Q|_|_|
Step is:  4
[(4, 0), (1, 1), (4, 2), (0, 3), (3, 4)]
|_|_|_|Q|_|
|_|Q|_|_|_|
|_|_|_|_|_|
|_|_|_|_|Q|
|Q|_|Q|_|_|
Step is:  5
[(4, 0), (1, 1), (3, 2), (0, 3), (3, 4)]
|_|_|_|Q|_|
|_|Q|_|_|_|
|_|_|_|_|_|
|_|_|Q|_|Q|
|Q|_|_|_|_|
Successfully finished:
[(4, 0), (1, 1), (3, 2), (0, 3), (2, 4)]
|_|_|_|Q|_|
|_|Q|_|_|_|
|_|_|_|_|Q|
|_|_|Q|_|_|
|Q|_|_|_|_|
Random restart hill climbing search Analysis
Random-Restart start_state:  [2, 1, 3, 1, 2]
Initial state is:
[(2, 0), (1, 1), (3, 2), (1, 3), (2, 4)]
|_|_|_|_|_|
|_|Q|_|Q|_|
|Q|_|_|_|Q|
|_|_|Q|_|_|
|_|_|_|_|_|
Step is:  2
[(2, 0), (1, 1), (3, 2), (0, 3), (2, 4)]
|_|_|_|Q|_|
|_|Q|_|_|_|
|Q|_|_|_|Q|
|_|_|Q|_|_|
|_|_|_|_|_|
Successfully finished:
[(4, 0), (1, 1), (3, 2), (0, 3), (2, 4)]
|_|_|_|Q|_|
|_|Q|_|_|_|
|_|_|_|_|Q|
|_|_|Q|_|_|
|Q|_|_|_|_|
Random restart hill climbing search with sideways movement Analysis
Random-Restart start_state:  [2, 2, 2, 0, 3]
Initial state is:
[(2, 0), (2, 1), (2, 2), (0, 3), (3, 4)]
```

```
|_|_|_|Q|_|
|_|_|_|_|_|
|Q|Q|Q|_|_|
|_|_|_|_|Q|
|_|_|_|_|_|
Step is:  2
[(2, 0), (4, 1), (2, 2), (0, 3), (3, 4)]
|_|_|_|Q|_|
|_|_|_|_|_|
|Q|_|Q|_|_|
|_|_|_|_|Q|
|_|Q|_|_|_|
Successfully finished:
[(1, 0), (4, 1), (2, 2), (0, 3), (3, 4)]
|_|_|_|Q|_|
|Q|_|_|_|_|
|_|_|Q|_|_|
|_|_|_|_|Q|
|_|Q|_|_|_|
```

Output:

1) N = 5, Iterations = 100 and sideways movement = 10

```
Hill climbing Search Analysis
=============================

N(number of queens) value:  5  (i.e  5 x 5 )
Total number of Runs:  100

Successful Runs:  62
Success Rate:  62.0 %
Average Steps to success:  3.56

Failure Runs:  38
Failure Rate:  38.0 %
Average Steps to failure:  2.63

Flat local maxima:  38
```

```
Hill climbing Search with sideways movement Analysis
====================================================

N(number of queens) value:  5  (i.e  5 x 5 )
Total number of Runs:  100

Successful Runs:  100
Success Rate:  100.0 %
Average Steps to success:  4.11

Failure Runs:  0
Failure Rate:  - %
Average Steps to failure:  -

Flat local maxima:  0

Random restart hill climbing search Analysis
============================================

N(number of queens) value:  5  (i.e  5 x 5 )
Total number of Runs:  100

Average Restarts:  1.48
Average Steps on last restart:  3.46
Average steps on all restarts:  4.61

Random restart hill climbing search with sideways movement Analysis
===================================================================

N(number of queens) value:  5  (i.e  5 x 5 )
Total number of Runs:  100

Average Restarts:  1.0
Average Steps on last restart:  4.12
Average steps on all restarts:  4.12
```

N = 8, Iterations = 500 and sideways movement = 100

```
Hill climbing Search Analysis
=============================

N(number of queens) value:  8  (i.e  8 x 8 )
Total number of Runs:  500
```

```
Successful Runs:  55
Success Rate:  11.0 %
Average Steps to success:  5.25

Failure Runs:  445
Failure Rate:  89.0 %
Average Steps to failure:  4.0

Flat local maxima:  442


Hill climbing Search with sideways movement Analysis
====================================================

N(number of queens) value:  8  (i.e  8 x 8 )
Total number of Runs:  500

Successful Runs:  469
Success Rate:  93.8 %
Average Steps to success:  19.54

Failure Runs:  31
Failure Rate:  6.2 %
Average Steps to failure:  60.03

Flat local maxima:  17


Random restart hill climbing search Analysis
============================================

N(number of queens) value:  8  (i.e  8 x 8 )
Total number of Runs:  500

Average Restarts:  7.056
Average Steps on last restart:  5.088
Average steps on all restarts:  29.642



Random restart hill climbing search with sideways movement Analysis
===================================================================

N(number of queens) value:  8  (i.e  8 x 8 )
Total number of Runs:  500

Average Restarts:  1.252
Average Steps on last restart:  19.186
Average steps on all restarts:  24.052
```

Analysis:

	Hill Climbing	Hill Climbing with sideways movement	Random Restart Hill climbing	Random restart hill climbing with sideways movement
Success Rate	62%	100	100	100
Average Steps to success	3.56	4.11	Last restart: 3.46 All restarts: 4.61	Last restart: 4.12 All restarts: 4.12
Average Number of Restarts	Not applicable	Not applicable	1.48	1.0
Flat local maxima	38	0	Not applicable	Not applicable
Failure Rate	38%	0	0	0
Average steps to failure	2.63	0	0	0
Total Runs	100	100	100	100

	Hill Climbing	Hill Climbing with sideways movement	Random Restart Hill climbing	Random restart hill climbing with sideways movement
Success Rate	11%	93.8%	100	100
Average Steps to success	5.25	19.54	Last restart: 5.088 All restarts: 29.64	Last restart: 19.18 All restarts: 24.05
Average Number of Restarts	Not applicable	Not applicable	7.056	1.0
Flat local maxima	442	17	Not applicable	Not applicable
Failure Rate	89.0%	6.2%	0	0
Average steps to failure	4.0	60.03	0	0
Total Runs	500	500	500	500

Program Code:

Program code:

```
import random

import copy

import numpy as np

print_states = True


class HillClimbing:

    def __init__(self, state = None, sideways_moves = 0, Number_of_queens = 0):

        self.start_state = state


        if(state == None and Number_of_queens == 0):

            print("Invalid Number of queens value provided so we are going to proceed with 8
queens.")

            self.Number_of_queens = 8

        elif(state == None and Number_of_queens):

            self.Number_of_queens = Number_of_queens

        else:

            self.Number_of_queens = len(state)


        self.sideways_moves = sideways_moves

        self.sideways_moves_remaining = sideways_moves

        self.Number_of_steps = 0


    # 1. cells_at_state will return cells with queens in the given state.

    def cells_at_state(self,state):

        cells = []
        for column, row in enumerate(state):
            cells.append((row,column))
        return cells

    # 2. print_Nqueen_matrix will print n queen state as a matrix
    def print_Nqueen_matix(self, cellsAtState):
        global print_states
        if print_states:
            print(cellsAtState)
            for r in range(self.Number_of_queens):
                cell = '|'
                for c in range(self.Number_of_queens):
                    if (r,c) in cellsAtState:
                        cell += 'Q|'
                    else:
                        cell += '_|'
                print(cell)

    # 3. Horizontal_cells_right_to_current return the cells to the horizontal right of the current cell
    def horizontal_cells_right_to_current(self,row,column):
        j = column+1
        cells =[]
        while j < self.Number_of_queens:
            cells.append((row,j))
            j = j+1
        return cells
    # 4. diagonal_cells_right_to_current return the cells to the diagonal right of the current cell
    def diagonal_cells_right_to_current(self,row,column):
        j = column+1
        cells =[]
        while j < self.Number_of_queens:
            # top diagonal cells on  the right of current cell
            if row - (j-column) >= 0:
                cells.append((row-(j-column),j))
            # bottom diagonal cells on  the right of current cell
            if row + (j-column) <= self.Number_of_queens - 1:
                cells.append((row+(j-column),j))
            j = j+1
        return cells

    # 5. total_cells_to_the_right will return all the horizontal and diagonal cells to the right of
current cells
    def total_cells_to_the_right(self,row,column):
        total = self.horizontal_cells_right_to_current(row,column) +
self.diagonal_cells_right_to_current(row,column)
        return total

    # 6. heuristic_value returns heuristic value for a given state
    def heuristic_value(self,cellsAtState):
        heuristic_val = 0
        for row, column in cellsAtState:
            q_cells = set(cellsAtState)
            r_cells = set(self.total_cells_to_the_right(row,column))
            intersection = q_cells.intersection(r_cells)
            heuristic_val += len(intersection)

        return heuristic_val

    #7. heuristic_matrix calculates heuristic vlues of each cell and returns heuristic value matrix,
least heuritic
    #    and umpy array with row and column with least heuristic
    def heuristic_matrix(self,cellsAtState):
        heuristicMatrix = np.zeros((self.Number_of_queens, self.Number_of_queens), int) + (-1)
        least_heuristic = sum(range(self.Number_of_queens))+1
        least_heuristic_state = None
        for (row,column) in cellsAtState:
            for i in range(self.Number_of_queens):
                if row == i:
                    pass
                else:
                    new_state = copy.deepcopy(cellsAtState)
                    new_state[column] = temp = (i, column)
                    heuristicMatrix[i,column] = self.heuristic_value(new_state)
                    least_heuristic = min(least_heuristic, heuristicMatrix[i, column])
                    least_heuristic_state = new_state

        return heuristicMatrix, least_heuristic, np.where(heuristicMatrix == least_heuristic)

    #8. randon_state creates and returns a random state which will be used in random restart
function.
    def random_states(self):
        state = []
        for i in range(self.Number_of_queens):
            state.append(random.randint(0, self.Number_of_queens - 1))
        return state

    #9. hill_climbing_search function is a recursive implementation of the hill climbing search using sttepest ascent.
    #    this method will return result and step towards the least heuristic value at each recursion and the result would
    #    contain flat local maxima, local maxima and success
    def hill_climbing_search(self, state = None, heuristicVal = None, step = 0):
        cellsAtState = None
        if(step == 0):
            state = self.start_state
            cellsAtState = self.cells_at_state(state)
            heuristicVal = self.heuristic_value(cellsAtState)
        else:
            cellsAtState = self.cells_at_state(state)

        step = step + 1
        self.Number_of_steps += 1
        if heuristicVal == 0:
            if print_states:
                print("Successfully finished: ")
            self.print_Nqueen_matix(cellsAtState)
            return 3, step
        if step == 1:
            if print_states:
```

```
        print("Initial state is: ")
    self.print_Nqueen_matix(cellsAtState)
else:
    if print_states:
        print("Step is: ", step)
    self.print_Nqueen_matix(cellsAtState)

heuristicMatrix = self.heuristic_matrix(cellsAtState)
leastHeuristic = heuristicMatrix[1]
shuffledMatrix = random.randint(0,len(heuristicMatrix[2][0])-1)
shuffledRow = heuristicMatrix[2][0][shuffledMatrix]
shuffledCol = heuristicMatrix[2][1][shuffledMatrix]
newState = copy.deepcopy(state)
newState[shuffledCol] = shuffledRow

# result -> 1 => (flat,flat local maxima)
# result -> 2 => Local Maxima

if leastHeuristic < heuristicVal:
    return self.hill_climbing_search(newState, leastHeuristic, step)
#local Maxima condition
elif leastHeuristic > heuristicVal:
    return 2, step
#flat condition
elif leastHeuristic == heuristicVal:
    if self.sideways_moves_remaining:
        self.sideways_moves_remaining -= 1
```

```
                return self.hill_climbing_search(newState, leastHeuristic, step)
            else:
                if print_states:
                    print("The Search has failed.")
                return 1, step


    # 10. hill_climbing_random_restart function implements random restart algorithm.
    def hill_climbing_random_restart(self):
        number_of_restarts = 0
        while True:
            number_of_restarts += 1
            self.start_state = self.random_states()
            print('Random-Restart start_state: ', self.start_state)
            result = self.hill_climbing_search()
            if result[0] == 3:
                return number_of_restarts, result[1], self.Number_of_steps
                break


class project_analysis:
    def __init__(self, n, maxIterations, maxSide_moves = 0):
        self.n = n
        self.maxIterations = maxIterations
        self.maxSide_moves = maxSide_moves
        self.hillclimbing_stats = [[0,[]],[0,[]],[0,[]],[0,[]]]
        self.hillclimbing_with_sideways_stats = [[0,[]],[0,[]],[0,[]],[0,[]]]
        self.random_restart_stats = [0, [], [], []]
        self.random_restart_with_sideways_stats = [0, [], [], []]

    # 1. analysis function will start iterating and performs hill climbing and randon-restart hill climbing with and without
    #     side ways movement.
    def analysis(self):
        if(self.n in range(4)):
            print('Invalid Number of queens(N) value. Number of queens shoule be above 3 !!!')
            return

        if(self.maxIterations < 1):
            print('Invalid number of iterations provided. Number of iterations should be above 1 !!!')
            return

        for n in range(self.maxIterations):
            self.hillclimbing_stats[0][0] += 1
            self.hillclimbing_with_sideways_stats[0][0] += 1
            self.random_restart_stats[0] += 1
            self.random_restart_with_sideways_stats[0] += 1
            state = []

            # The for loop below generates random state
            for i in range(self.n):
                state.append(random.randint(0,self.n-1))
            if(print_states):
                print("Hill climbing Search Analysis")
```

```
hillClimbing = HillClimbing(state)
result = hillClimbing.hill_climbing_search()
self.hillclimbing_stats[result[0]][0] += 1
self.hillclimbing_stats[result[0]][1].append(result[1])

if(print_states):
   print("Hill climbing Search with sideways movement Analysis")
hillClimbing_sideways = HillClimbing(state, self.maxSide_moves)
result = hillClimbing_sideways.hill_climbing_search()
self.hillclimbing_with_sideways_stats[result[0]][0] += 1
self.hillclimbing_with_sideways_stats[result[0]][1].append(result[1])

if(print_states):
   print("Random restart hill climbing search Analysis")
hillClimbing_randomRestart = HillClimbing(None, 0, self.n)
result = hillClimbing_randomRestart.hill_climbing_random_restart()
self.random_restart_stats[1].append(result[0])
self.random_restart_stats[2].append(result[1])
self.random_restart_stats[3].append(result[2])

if(print_states):
   print("Random restart hill climbing search with sideways movement Analysis")
hillClimbing_randomRestart_sideways = HillClimbing(None, self.maxSide_moves, self.n)
result = hillClimbing_randomRestart_sideways.hill_climbing_random_restart()
self.random_restart_with_sideways_stats[1].append(result[0])
self.random_restart_with_sideways_stats[2].append(result[1])
self.random_restart_with_sideways_stats[3].append(result[2])
```

```
        self.print_results()


    # 2. print_results function prints stats of all 4 algorithms.

    def print_results(self):


        self.print_hillclimbing_stats(self.hillclimbing_stats, "Hill climbing Search Analysis")

        self.print_hillclimbing_stats(self.hillclimbing_with_sideways_stats, "Hill climbing Search with sideways movement Analysis")

        self.print_random_restart_stats(self.random_restart_stats, "Random restart hill climbing search Analysis")

        self.print_random_restart_stats(self.random_restart_with_sideways_stats, "Random restart hill climbing search with sideways movement Analysis")



    # 3. print_hillclimbing_stats will print report of the hill climbing search with and without sideways movement.

    def print_hillclimbing_stats(self, result, title):


        total_number_of_Runs = result[0][0]

        successful_runs = result[3][0]

        if successful_runs:

            success_rate = round((successful_runs/total_number_of_Runs)*100,2)

            steps_to_success = result[3][1]

            average_steps_to_success = round(sum(steps_to_success)/successful_runs, 2)

        else:

            success_rate = steps_to_success = average_steps_to_success = '-'

        failure_runs = result[1][0] + result[2][0]
        if failure_runs:
            failure_rate = round((failure_runs/total_number_of_Runs)*100,2)
            steps_to_failure = result[1][1]+result[2][1]
            average_steps_to_failure = round(sum(steps_to_failure)/failure_runs,2)
        else:
            failure_rate = steps_to_failure = average_steps_to_failure = '-'
        flat_runs = result[1][0]
        print("\n\n")
        print(title)
        underline = ''
        for i in range(len(title)):
            underline+="="

        print(underline,"\n")
        print("N(number of queens) value: ", self.n, " (i.e ",self.n,"x",self.n,")")
        print("Total number of Runs: ", total_number_of_Runs,"\n")
        print("Successful Runs: ", successful_runs)
        print("Success Rate: ", success_rate, "%")
        print("Average Steps to success: ", average_steps_to_success, "\n")
        print("Failure Runs: ", failure_runs)
        print("Failure Rate: ", failure_rate, "%")
        print("Average Steps to failure: ", average_steps_to_failure, "\n\n")
        print("Flat local maxima: ", flat_runs)
        return

    # 4. print_random_restart_stats will print report of the random restart hill climbing search with and without sideways movement.
    def print_random_restart_stats(self, result, title):

        total_number_of_runs = result[0]
        average_number_of_restarts = sum(result[1]) / total_number_of_runs
        average_last_steps = sum(result[2]) / total_number_of_runs
        average_total_steps = sum(result[3]) / total_number_of_runs
        print("\n\n")
        print(title)
        underline = ''
        for i in range(len(title)):
            underline+="="
        print(underline, "\n")
        print("N(number of queens) value: ", self.n, " (i.e ",self.n,"x",self.n,")")
        print("Total number of Runs: ", total_number_of_runs, "\n")
        print("Average Restarts: ", average_number_of_restarts)
        print("Average Steps on last restart: ", average_last_steps)
        print("Average steps on all restarts: ", average_total_steps)

N = iterations = sideways_movement = 0

# Getting number of queens "N" value
while(True):
    try:
        N = int(input("Please enter a value for N(number of queens): "))
        if(N < 4):
            print("Please enter a N(number of queens) greater than 3.")
        else:
            break
    except ValueError:
        print("Please provide a valid input")

# Getting iterations value
while(True):
    try:
        iterations = int(input("Please enter a value for number of iterations: "))
        if(iterations < 1):
            print("Please enter an iterations value that greater than or equal to 1.")
        else:
            break
    except ValueError:
        print("Please provide a valid input")

# Getting maximum sideways movement allowed value
while(True):
    try:
        sideways_movement = int(input("Please enter a value for the maximum sideways moves allowed: "))
        if(sideways_movement < 1):
            print("Please enter a sideways moves value that greater than or equal to 1.")
        else:
            break
    except ValueError:
        print("Please provide a valid input")
```

```
hill_climbing_search_analysis = project_analysis(N, iterations, sideways_movement)

hill_climbing_search_analysis.analysis()
```