|  |  |  |
| --- | --- | --- |
|  | Project report |  |
|  | on |  |
| **Solving N-Queens problem using Hill-Climbing Algorithm and its variants** | | |
|  | Project Guidance By |  |
|  | **Dr. Dewan Ahmed** |  |
|  | Team details |  |

|  |  |  |
| --- | --- | --- |
| **Priyanka Pandit Jadhav (801167642)** | **Maheswara Rao Lekkala**  **(801167702)** | **Bharadwaj Tirunagaru (801167910)** |
| [**pjadhav7@uncc.edu**](mailto:pjadhav7@uncc.edu) | [**mlekkala@uncc.edu**](mailto:mlekkala@uncc.edu) | [**btirunag@uncc.edu**](mailto:btirunag@uncc.edu) |
| **704-905-1666** | **980-833-5010** | **980-729-4465** |

**Date: 03/27/2019**

**AIM**

To solve n-queens problem using hill-climbing search and its variants.

**PROBLEM STATEMENT**

Implement Hill-climbing search, Hill-climbing search with sideway moves and Random-restart hill-climbing with and without sideways move and apply it to n-queens problem. List average number of steps when the algorithm succeeds and fails along with the success and failure rate for multiple iterations.

**N-QUEENS PROBLEM**

The N-queens puzzle is the problem of placing N queens on a N x N chessboard such that no two queens attack each other. The queen is the most powerful piece in chess and can attack from any distance horizontally, vertically, or diagonally. Thus, a solution requires that no two queens share the same row, column, or diagonal.

**PROBLEM FORMULATION**

**Initial State:** A random arrangement on n queens, with one in each column.

**Goal State:** N queens placed on the board such that no two queens can attack each other.

**States:** Any arrangement of n queens, one in each column.

**Actions:** Move any attacked queen to another square in the same column.

**Performance:** Number of steps and success rate to find a solution.

**HILL-CLIMBING ALGORITHM**

Hill Climbing is heuristic search used for mathematical optimization problems in the field of Artificial Intelligence. It is an [iterative algorithm](https://en.wikipedia.org/wiki/Iterative_algorithm) that starts with an arbitrary solution to a problem, then attempts to find a better solution by making an [incremental](https://en.wikipedia.org/wiki/Incremental_heuristic_search) change to the solution. If the change produces a better solution, another incremental change is made to the new solution, and so on until no further improvements can be found.

Steepest-Ascent Hill-climbing: It first examines all the neighboring nodes and then selects the node closest to the solution state as next node with best heuristic value. If no best successor is found then the search fails.

**f(n) = g(n) + h(n)**

**g(n) = cost so far to reach n**

**h(n) = estimated cost from n to goal**

**f(n) = estimated cost of path through n to goal**

**Heuristic Functions**

The heuristic function is a way to inform the search regarding the direction to a goal. It provides an information to estimate which neighboring node will lead to the goal. The two heuristic functions that we considered for solving 8-puzzle problem are

* **Misplaced Tile**

The number of misplaced tiles calculated by comparing the current state and goal state.

* **Manhattan Distance**

The distance between two tiles measured along the axes of right angles. It is the sum of absolute values of differences between goal state (i, j) coordinates and current state (l, m) coordinates respectively, i.e. |i - l|+ |j - m|

**HEURISTIC FUNCTION:**

The Heuristic function in the N queen problem is the number of pairs of queens that are attacking each other. The best successor is the state with low heuristic value.

|  |  |  |  |
| --- | --- | --- | --- |
| **Q** |  |  | **Q** |
|  | **Q** |  |  |
|  |  | **Q** |  |
|  |  |  |  |

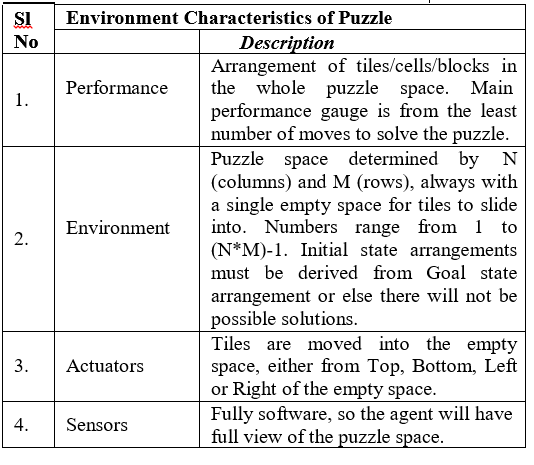
The heuristic value for the above problem is four since there are four pairs of queens that are attacking each other at this moment.

**N- Queens Puzzle**

For solution searching, it would be most useful to distil the possible arrangements of tiles as individual States. Thus, each State shows a possible combination of tile positions within the given puzzle space. The collection of all possible States is called the State Space. With the increase of N or M of the puzzle, the size of the State Space shall increase exponentially.

In every state, the empty space position determines which States can be transitioned to. For instance, when the empty space is in the middle of a 3x3 puzzle, tiles at the Top, Bottom, Left or Right can move into it. But if the empty space is at the top left corner, only the right or bottom tiles can slide into it.

Thus, after each slide, a new State is transitioned into. If puzzle is to begin with an Initial State of tile arrangements, then its subsequent transitions into other States can be represented by a Graph. A search attempt will need to begin with an Initial State and a Goal State to achieve. As puzzle traversal can often pass through the same state at different intervals. We will consider the instances of decisions as nodes. By aligning the node arrangements to start from the Initial Node to possible routes leading to the Goal nodes, a search tree is formed.



**Hill Climbing Algorithm**

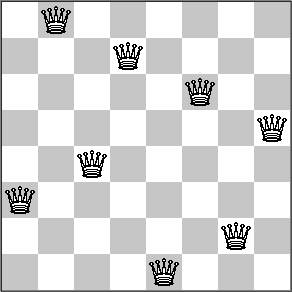
Hill Climbing works disregards memory of explored nodes. Therefore, it travels down the Search Tree by selecting the successor with the cheapest heuristics value, without retaining memory of explored states. This will ensure that the heuristics technique functions with minimal use of memory, least computation possible but still retain the advantage of an informed method of solution finding. The downside of Hill Climbing is that due to the absence of memory, resulting in the possibility of repeating the same states and getting stuck in some state of local maxima.

**Hill Climbing for 8 queens Problem**

Hill climbing search for this 8-Queen puzzle, in order to reach the goal state where h = 0, it will continue to loop to find moves in the direction of decreasing *h(n)*. It will terminate when there is no lower *h(n)* than the previous ones. Hill climbing search will randomly generate 8 random placement of the queen in the 8x8 board after the initial state it will then calculate the *h(n)* and then during the next state it will swap the Q position column by column in search of the *h(n)* that is lesser than the previous *h(n)* until it reach *h=0*.Hence, based on the evaluation function *f(n)*=*h(n)*, so the results will be *f(n)*1=0. The board will terminate if there is no *h(n)* that is less than the previous *h(n)*. When board is clear, a new random placement of the queens is placed again and the process is repeated until it reaches the goal state.

**N-Queens: Steepest Hill Climbing:**

The n-queens problem was first invented in the mid-1800s as a puzzle for people to solve in their spare time, but now serves as a good tool for discussing computer search algorithms. In chess, a queen is the only piece that can attack in any direction. The puzzle is to place a number of queens on a board in such a way that no queen is attacking any other. For example:



The N-queens problem is the problem of placing ‘n’ chess queens on an n×n chessboard so that no two queens threaten each other. This means that no two queens can be in same row, column or diagonal. We can find solutions for all natural numbers ‘n’ except for n=2 and n=3. Here the problem is solved using a complete-state formulation, which means we start with all 8 queens on the board and move them around to reach the goal state. We represent the n\*n chess board as a matrix.

The classic combinatorial problem is to place N-Queens on a chessboard so that no two attack each other. In the chess Queens attacking in three directions i.e. horizontally, vertically and diagonally. The problem can be generalized as placing ‘n’ non attacking queens on an N x N chessboard. Since each queen must be on a different row and column, we can assume that queen "i" is placed in ith column. All solutions to the NQP can therefore be represented as n-tuples (q1, q2, …, qn) that are permutations of an n-tuple (1, 2, 3, …, n). Position of a number in the tuple represents queen's column position, while its value represents queen's row position (counting from the bottom) using this representation, the solution space where two of the constraints (row and column conflicts) are already satisfied should be searched in order to eliminate the diagonal conflicts. Complexity of this problem is O (n!). The N-Queens problem is a generalization of the 8-Queens problem posed by a German chess player, Max Bezzel in 1848. The objective of the N-Queens problem is to arrange N-Queens so that no queen may attack each queen. Thus each column, row, diagonal, and anti-diagonal must contain one and only one queen.

**PROCESS OF SOLVING N-QUEENS**

* Suppose you have 8 chess Queens and chess board of size 8\*8.
* Queens can be placed on the chess board so no two queens are attacking each other.
* Two Queens are not allowed in the same column.
* Two Queens are not allowed in the same column, in the same row.
* Two Queens are not allowed in the same column, in the same row, or along the same diagonal.
* The number of Queens and the size of the board can differ.
* It looks like hard to generate one valid placement.
* The program uses a stack to keep track of where each Queens is placed.
* Each time the program decides to place a Queens on the board, the position of the new Queens is stored in a record which is placed in the stack.
* We also have an integer variable to keep track of how many rows have been filled so far.
* Each time we try to place a new Queens in the next row, we start by placing the Queens in the first column.
* If there is a clash with another Queens, then we shift the new Queens to the next column.
* If another clash occurs, the Queens is shifted rightward again.
* When there are no clash, we stop and add one to the value of filled.

**PROGRAM DESIGN AND EXPLANATION:**

**Global variables**

**int resetSum**: To keep track of total number of resets in random restart hill climbing  
**int randomGenerateSum**: To keep track of total runs in random restart in order to calculate average reset count

**Classes used**

**State:**

Class used to store the related information of each state. It contains constructor, getter and setter functions for the member variables. Also generate successors method which will generate all possible successors of the provided state.

**Main:**

Contains methods used to solve the n-Queen problem, to calculate the heuristics, to print the solution path, to find the possible next states.

**SOURCE CODE:**

**Main.java**

import java.util.ArrayList;

import java.util.Random;

import java.util.Scanner;

public class Main {

int resetSum=0;

int randomGenerateSum=0;

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

Main mainObj = new Main();

System.out.println("Enter number of Queens in n-Queen Problem :");

int numOfQueens = scanner.nextInt();

while(numOfQueens<4){

System.out.println("Invalid number of Queens. Please provide number greater than equal to 4");

numOfQueens = scanner.nextInt();

}

System.out.println("Provide RunTime: ");

int maxRun = scanner.nextInt();

while(maxRun<=0){

System.out.println("MaxRun should be greater than 0");

maxRun = scanner.nextInt();

}

int choice=0;

do {

System.out.println("Choose Search Type\n1.Hill Climbing Search\n2.Hill-climbing search with sideways move");

System.out.println("3.Random-restart hill-climbing search without sideway moves");

System.out.println("4.Random-restart hill-climbing search with sideway moves");

System.out.println("5.Exit");

choice = scanner.nextInt();

if(choice<1 ||choice>5){

System.out.println("Invalid choice: Please choose from provided options");

choice = scanner.nextInt();

}

if(choice==5)

System.exit(0);

int sucessCount = 0;

int successStepCount = 0;

int failureCount = 0;

int failureStepCount = 0;

int randomOutput = 0;

if (choice == 1)

System.out.println("------------------Hill Climbing Search-----------------------------");

else if (choice == 2)

System.out.println("---------------Hill-climbing search with sideways move--------------");

else if (choice == 3)

System.out.println("-----------------Random-restart hill-climbing search without sideway moves----------------");

else if (choice == 4)

System.out.println("-----------------Random-restart hill-climbing search with sideway moves----------------");

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

ArrayList<State> path = null;

State initialState = mainObj.generateRandomState(numOfQueens);

switch (choice) {

case 1:

path = mainObj.performHillClimbingSearch(initialState);

break;

case 2:

path = mainObj.performHillClimbingSearchWithSidewayMove(initialState);

break;

case 3:

path = mainObj.performHillClimbingSearchRandomRestartWithoutSidewayMoves(initialState);

break;

case 4:

path = mainObj.performHillClimbingSearchRandomRestartWithSidewayMoves(initialState);

break;

default:

}

State lastState = path.get(path.size() - 1);

if (lastState.getCost() == 0) {

sucessCount++;

successStepCount += path.size();

if (randomOutput < 4) {

System.out.println("The search sequences (" + (randomOutput + 1) + ") :");

System.out.println("Path cost: " + path.size());

printPath(path);

randomOutput++;

}

} else {

failureCount++;

failureStepCount += path.size();

}

}

System.out.println("success count " + sucessCount);

System.out.println("success rate " + ((double) sucessCount / maxRun) \* 100 + " %");

System.out.println("failure count " + failureCount);

System.out.println("failure rate " + ((double) failureCount / maxRun) \* 100 + " %");

System.out.println("Average no. of Steps when success: " + (double) successStepCount / sucessCount);

if (failureCount != 0)

System.out.println("Average no. of Steps when failure: " + (double) failureStepCount / failureCount);

else

System.out.println("Average no. of Steps when failure: 0");

if (choice == 3 || choice == 4) {

System.out.println("total reset count : " + mainObj.resetSum);

System.out.println("Avg reset count : " + (double) mainObj.resetSum / (maxRun + mainObj.randomGenerateSum));

}

}while(choice != 5);

}

**State.java**

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Collections;

public class State {

private int[] qRow;

private int cost;

public State(int[] qRow){

this.qRow=qRow;

this.cost=calculateStateCost(qRow);

}

private int calculateStateCost(int[] qRow) {

int cost=0;

for(int i=0;i<qRow.length;i++){

for(int j=i+1;j<qRow.length;j++){

if(qRow[i]==qRow[j] || i==j)

cost++;

else if(Math.abs(i-j)==Math.abs(qRow[i]-qRow[j]))

cost++;

}

}

return cost;

}

public int[] getqRow() {

return qRow;

}

public int getCost() {

return cost;

}

@Override

public String toString() {

StringBuilder sb = new StringBuilder();

for (int i = 0; i < qRow.length; i++) {

for (int j = 0; j < qRow.length; j++) {

if (qRow[i] == j) {

sb.append('Q');

} else {

sb.append('.');

}

sb.append(' ');

}

sb.append('\n');

}

sb.deleteCharAt(sb.length() - 1);

return sb.toString();

}

public ArrayList<State> generateSuccessors() {

int originalValue;

int[] copy;

int[] board = Arrays.copyOf(qRow, qRow.length);

ArrayList<State> successors = new ArrayList<>((qRow.length \* qRow.length) - qRow.length);

for (int i = 0; i < board.length; i++) {

originalValue = board[i];

for (int j = 0; j < board.length; j++) {

if (j == originalValue) {

continue;

}

board[i] = j;

copy = Arrays.copyOf(board, board.length);

successors.add(new State(copy));

}

board[i] = originalValue;

}

if (successors.size() != (qRow.length \* qRow.length) - qRow.length) {

System.out.println("PROBLEM!!!!");

}

Collections.sort(successors, (State s1, State s2) -> {

return s1.cost - s2.cost;

});

return successors;

}

}

**SAMPLE INPUT/OUTPUT:**

Enter number of Queens in n-Queen Problem :

8

Provide RunTime:

500

Choose Search Type

1.Hill Climbing Search

2.Hill-climbing search with sideways move

3.Random-restart hill-climbing search without sideway moves

4.Random-restart hill-climbing search with sideway moves

5.Exit

1

------------------Hill Climbing Search-----------------------------

The search sequences (1) :

Path cost: 4

. . . . Q . . .

. . . . . . . Q

. . . . . . Q .

. . . . Q . . .

. . . Q . . . .

. Q . . . . . .

. . . . . . . Q

. . . . . Q . .

cost : 6

. . . . Q . . .

. . . . . . . Q

. . . . . . Q .

Q . . . . . . .

. . . Q . . . .

. Q . . . . . .

. . . . . . . Q

. . . . . Q . .

cost : 3

. . Q . . . . .

. . . . . . . Q

. . . . . . Q .

Q . . . . . . .

. . . Q . . . .

. Q . . . . . .

. . . . . . . Q

. . . . . Q . .

cost : 2

. . Q . . . . .

. . . . Q . . .

. . . . . . Q .

Q . . . . . . .

. . . Q . . . .

. Q . . . . . .

. . . . . . . Q

. . . . . Q . .

cost : 0

The search sequences (2) :

Path cost: 5

. . . Q . . . .

. Q . . . . . .

. Q . . . . . .

. . . . Q . . .

. . . . Q . . .

. . . . . . . Q

Q . . . . . . .

. . . . Q . . .

cost : 6

. . . Q . . . .

. Q . . . . . .

. Q . . . . . .

. . . . Q . . .

Q . . . . . . .

. . . . . . . Q

Q . . . . . . .

. . . . Q . . .

cost : 4

. . . Q . . . .

. Q . . . . . .

. . . . . . Q .

. . . . Q . . .

Q . . . . . . .

. . . . . . . Q

Q . . . . . . .

. . . . Q . . .

cost : 2

. . . Q . . . .

. Q . . . . . .

. . . . . . Q .

. . Q . . . . .

Q . . . . . . .

. . . . . . . Q

Q . . . . . . .

. . . . Q . . .

cost : 1

. . . Q . . . .

. Q . . . . . .

. . . . . . Q .

. . Q . . . . .

. . . . . Q . .

. . . . . . . Q

Q . . . . . . .

. . . . Q . . .

cost : 0

The search sequences (3) :

Path cost: 6

Q . . . . . . .

. . . . . Q . .

. . Q . . . . .

Q . . . . . . .

. . Q . . . . .

. . . . . . . Q

. . . . . Q . .

. . . . . Q . .

cost : 9

Q . . . . . . .

. . . . . Q . .

. . Q . . . . .

Q . . . . . . .

. . Q . . . . .

. . . . . . . Q

. . . . . Q . .

. Q . . . . . .

cost : 5

Q . . . . . . .

. . . . Q . . .

. . Q . . . . .

Q . . . . . . .

. . Q . . . . .

. . . . . . . Q

. . . . . Q . .

. Q . . . . . .

cost : 3

. Q . . . . . .

. . . . Q . . .

. . Q . . . . .

Q . . . . . . .

. . Q . . . . .

. . . . . . . Q

. . . . . Q . .

. Q . . . . . .

cost : 2

. Q . . . . . .

. . . . Q . . .

. . Q . . . . .

Q . . . . . . .

. . Q . . . . .

. . . . . . . Q

. . . . . Q . .

. . . Q . . . .

cost : 1

. Q . . . . . .

. . . . Q . . .

. . . . . . Q .

Q . . . . . . .

. . Q . . . . .

. . . . . . . Q

. . . . . Q . .

. . . Q . . . .

cost : 0

The search sequences (4) :

Path cost: 6

Q . . . . . . .

. . . . . Q . .

. . . . . . . Q

. . . . . . Q .

. . Q . . . . .

Q . . . . . . .

Q . . . . . . .

. . . Q . . . .

cost : 7

Q . . . . . . .

. . . . . Q . .

. . . . . . . Q

. . . . . . Q .

. . Q . . . . .

Q . . . . . . .

. Q . . . . . .

. . . Q . . . .

cost : 4

Q . . . . . . .

. . . . . Q . .

. . . . . . . Q

. . . . . . Q .

. . Q . . . . .

. . . . . . Q .

. Q . . . . . .

. . . Q . . . .

cost : 3

Q . . . . . . .

. . . . . Q . .

. . . . . . . Q

Q . . . . . . .

. . Q . . . . .

. . . . . . Q .

. Q . . . . . .

. . . Q . . . .

cost : 2

Q . . . . . . .

. . . . Q . . .

. . . . . . . Q

Q . . . . . . .

. . Q . . . . .

. . . . . . Q .

. Q . . . . . .

. . . Q . . . .

cost : 1

Q . . . . . . .

. . . . Q . . .

. . . . . . . Q

. . . . . Q . .

. . Q . . . . .

. . . . . . Q .

. Q . . . . . .

. . . Q . . . .

cost : 0

**success count 74**

**success rate 14.799999999999999 %**

**failure count 426**

**failure rate 85.2 %**

**Average no. of Steps when success: 5.0675675675675675**

**Average no. of Steps when failure: 4.084507042253521**

Choose Search Type

1.Hill Climbing Search

2.Hill-climbing search with sideways move

3.Random-restart hill-climbing search without sideway moves

4.Random-restart hill-climbing search with sideway moves

5.Exit

2

---------------Hill-climbing search with sideways move--------------

The search sequences (1) :

Path cost: 9

. . . Q . . . .

. . . . . . . Q

. . . Q . . . .

. . . . Q . . .

. . . . Q . . .

. . . Q . . . .

. . . Q . . . .

. . . . Q . . .

cost : 14

. . . Q . . . .

. . . . . . . Q

. . . Q . . . .

. . . . Q . . .

. . . . Q . . .

. Q . . . . . .

. . . Q . . . .

. . . . Q . . .

cost : 9

. . . Q . . . .

. . . . . . . Q

Q . . . . . . .

. . . . Q . . .

. . . . Q . . .

. Q . . . . . .

. . . Q . . . .

. . . . Q . . .

cost : 6

. . . Q . . . .

. . . . . . . Q

Q . . . . . . .

. . . . Q . . .

. . . . . . Q .

. Q . . . . . .

. . . Q . . . .

. . . . Q . . .

cost : 3

. . . . . Q . .

. . . . . . . Q

Q . . . . . . .

. . . . Q . . .

. . . . . . Q .

. Q . . . . . .

. . . Q . . . .

. . . . Q . . .

cost : 2

. . . . . Q . .

. . . . . . . Q

Q . . . . . . .

. . . . Q . . .

. . . . . . Q .

. Q . . . . . .

. . . Q . . . .

. . Q . . . . .

cost : 1

. . . . . Q . .

. . . . . . . Q

Q . . . . . . .

. . . . Q . . .

. . . . . . Q .

. Q . . . . . .

. . . . . . Q .

. . Q . . . . .

cost : 1

. . . . . Q . .

. . . . . . . Q

Q . . . . . . .

. . . . Q . . .

. . . . . . Q .

. Q . . . . . .

. . . . . Q . .

. . Q . . . . .

cost : 1

. . . Q . . . .

. . . . . . . Q

Q . . . . . . .

. . . . Q . . .

. . . . . . Q .

. Q . . . . . .

. . . . . Q . .

. . Q . . . . .

cost : 0

The search sequences (2) :

Path cost: 6

. . . . . . . Q

. . . . Q . . .

. . . . . . Q .

. Q . . . . . .

. Q . . . . . .

. . Q . . . . .

. . . . . Q . .

. Q . . . . . .

cost : 7

. . . . . . . Q

. . . . Q . . .

. . . . . . Q .

. Q . . . . . .

Q . . . . . . .

. . Q . . . . .

. . . . . Q . .

. Q . . . . . .

cost : 4

. . . . . . . Q

. . . . Q . . .

. . . . . . Q .

. . . Q . . . .

Q . . . . . . .

. . Q . . . . .

. . . . . Q . .

. Q . . . . . .

cost : 2

. . . . . . . Q

. . . . Q . . .

. . . . . . Q .

. . . Q . . . .

Q . . . . . . .

. . . . . . . Q

. . . . . Q . .

. Q . . . . . .

cost : 2

. . . . . . . Q

. . . . Q . . .

. . . . . . Q .

. . . Q . . . .

Q . . . . . . .

. . . . . . . Q

. . . . . Q . .

. . Q . . . . .

cost : 1

. Q . . . . . .

. . . . Q . . .

. . . . . . Q .

. . . Q . . . .

Q . . . . . . .

. . . . . . . Q

. . . . . Q . .

. . Q . . . . .

cost : 0

The search sequences (3) :

Path cost: 3

. . . . . Q . .

. . Q . . . . .

. . Q . . . . .

. Q . . . . . .

Q . . . . . . .

. . . . . . . Q

Q . . . . . . .

. . . . Q . . .

cost : 5

. . . . . Q . .

. . Q . . . . .

. . . . . . Q .

. Q . . . . . .

Q . . . . . . .

. . . . . . . Q

Q . . . . . . .

. . . . Q . . .

cost : 2

. . . . . Q . .

. . Q . . . . .

. . . . . . Q .

. Q . . . . . .

. . . Q . . . .

. . . . . . . Q

Q . . . . . . .

. . . . Q . . .

cost : 0

The search sequences (4) :

Path cost: 4

. . Q . . . . .

. . . . . Q . .

. . . . . . . Q

. . . . . . Q .

. . Q . . . . .

. . Q . . . . .

. . . . . . . Q

. . . . Q . . .

cost : 7

. . Q . . . . .

. . . . . Q . .

. . . . . . . Q

. . . . . . Q .

Q . . . . . . .

. . Q . . . . .

. . . . . . . Q

. . . . Q . . .

cost : 4

. . Q . . . . .

. . . . . Q . .

. Q . . . . . .

. . . . . . Q .

Q . . . . . . .

. . Q . . . . .

. . . . . . . Q

. . . . Q . . .

cost : 2

. . Q . . . . .

. . . . . Q . .

. Q . . . . . .

. . . . . . Q .

Q . . . . . . .

. . . Q . . . .

. . . . . . . Q

. . . . Q . . .

cost : 0

**success count 467**

**success rate 93.4 %**

**failure count 33**

**failure rate 6.6000000000000005 %**

**Average no. of Steps when success: 19.946466809421842**

**Average no. of Steps when failure: 59.21212121212121**

Choose Search Type

1.Hill Climbing Search

2.Hill-climbing search with sideways move

3.Random-restart hill-climbing search without sideway moves

4.Random-restart hill-climbing search with sideway moves

5.Exit

3

-----------------Random-restart hill-climbing search without sideway moves----------------

The search sequences (1) :

Path cost: 5

. . Q . . . . .

. Q . . . . . .

. . . Q . . . .

. . Q . . . . .

. . . . Q . . .

. . . Q . . . .

. . . . . . . Q

. . . . . Q . .

cost : 8

. . . . . . Q .

. Q . . . . . .

. . . Q . . . .

. . Q . . . . .

. . . . Q . . .

. . . Q . . . .

. . . . . . . Q

. . . . . Q . .

cost : 6

. . . . . . Q .

. Q . . . . . .

. . . . . Q . .

. . Q . . . . .

. . . . Q . . .

. . . Q . . . .

. . . . . . . Q

. . . . . Q . .

cost : 4

. . . . . . Q .

. Q . . . . . .

. . . . . Q . .

. . Q . . . . .

Q . . . . . . .

. . . Q . . . .

. . . . . . . Q

. . . . . Q . .

cost : 2

. . . . . . Q .

. Q . . . . . .

. . . . . Q . .

. . Q . . . . .

Q . . . . . . .

. . . Q . . . .

. . . . . . . Q

. . . . Q . . .

cost : 0

The search sequences (2) :

Path cost: 5

. . . . . Q . .

. . . . . . Q .

. Q . . . . . .

. . . . . Q . .

. . . . Q . . .

. . . Q . . . .

Q . . . . . . .

. . . . . Q . .

cost : 8

. . . . . Q . .

. . . . . . Q .

. Q . . . . . .

. . . . . . . Q

. . . . Q . . .

. . . Q . . . .

Q . . . . . . .

. . . . . Q . .

cost : 4

. . Q . . . . .

. . . . . . Q .

. Q . . . . . .

. . . . . . . Q

. . . . Q . . .

. . . Q . . . .

Q . . . . . . .

. . . . . Q . .

cost : 2

. . Q . . . . .

. . . . . . Q .

. Q . . . . . .

. . . . . . . Q

. . . . Q . . .

Q . . . . . . .

Q . . . . . . .

. . . . . Q . .

cost : 1

. . Q . . . . .

. . . . . . Q .

. Q . . . . . .

. . . . . . . Q

. . . . Q . . .

Q . . . . . . .

. . . Q . . . .

. . . . . Q . .

cost : 0

The search sequences (3) :

Path cost: 6

. . . . Q . . .

. Q . . . . . .

. . Q . . . . .

. . . . Q . . .

. . . . . Q . .

. . . . . Q . .

Q . . . . . . .

. . . Q . . . .

cost : 8

. . . . Q . . .

. Q . . . . . .

. . Q . . . . .

. . . . Q . . .

. . . . . Q . .

. . . . . . . Q

Q . . . . . . .

. . . Q . . . .

cost : 4

. . . Q . . . .

. Q . . . . . .

. . Q . . . . .

. . . . Q . . .

. . . . . Q . .

. . . . . . . Q

Q . . . . . . .

. . . Q . . . .

cost : 3

. . . Q . . . .

. Q . . . . . .

. . . . . . Q .

. . . . Q . . .

. . . . . Q . .

. . . . . . . Q

Q . . . . . . .

. . . Q . . . .

cost : 2

. . . Q . . . .

. Q . . . . . .

. . . . . . Q .

. . Q . . . . .

. . . . . Q . .

. . . . . . . Q

Q . . . . . . .

. . . Q . . . .

cost : 1

. . . Q . . . .

. Q . . . . . .

. . . . . . Q .

. . Q . . . . .

. . . . . Q . .

. . . . . . . Q

Q . . . . . . .

. . . . Q . . .

cost : 0

The search sequences (4) :

Path cost: 4

. . . Q . . . .

Q . . . . . . .

Q . . . . . . .

. Q . . . . . .

. . . . . Q . .

. . . . . Q . .

. . Q . . . . .

. . . . . . . Q

cost : 4

. . . Q . . . .

Q . . . . . . .

. . . . Q . . .

. Q . . . . . .

. . . . . Q . .

. . . . . Q . .

. . Q . . . . .

. . . . . . . Q

cost : 2

. . . Q . . . .

Q . . . . . . .

. . . . Q . . .

. Q . . . . . .

. . . . . Q . .

Q . . . . . . .

. . Q . . . . .

. . . . . . . Q

cost : 1

. . . Q . . . .

. . . . . . Q .

. . . . Q . . .

. Q . . . . . .

. . . . . Q . .

Q . . . . . . .

. . Q . . . . .

. . . . . . . Q

cost : 0

**success count 500**

**success rate 100.0 %**

**failure count 0**

**failure rate 0.0 %**

**Average no. of Steps when success: 5.214**

**Average no. of Steps when failure: 0**

**total reset count : 2791**

**Avg reset count : 5.582**

Choose Search Type

1.Hill Climbing Search

2.Hill-climbing search with sideways move

3.Random-restart hill-climbing search without sideway moves

4.Random-restart hill-climbing search with sideway moves

5.Exit

4

-----------------Random-restart hill-climbing search with sideway moves----------------

The search sequences (1) :

Path cost: 5

. . . . Q . . .

. . . . . Q . .

. . . . . . . Q

. . . . . Q . .

. . . . . Q . .

. . . . . . Q .

. . Q . . . . .

. . Q . . . . .

cost : 10

. . . . Q . . .

. . . . . Q . .

. . . . . . . Q

. . . . . Q . .

. Q . . . . . .

. . . . . . Q .

. . Q . . . . .

. . Q . . . . .

cost : 5

. . . . Q . . .

Q . . . . . . .

. . . . . . . Q

. . . . . Q . .

. Q . . . . . .

. . . . . . Q .

. . Q . . . . .

. . Q . . . . .

cost : 3

. . . . Q . . .

Q . . . . . . .

. . . . . . . Q

. . . Q . . . .

. Q . . . . . .

. . . . . . Q .

. . Q . . . . .

. . Q . . . . .

cost : 2

. . . . Q . . .

Q . . . . . . .

. . . . . . . Q

. . . Q . . . .

. Q . . . . . .

. . . . . . Q .

. . Q . . . . .

. . . . . Q . .

cost : 0

The search sequences (2) :

Path cost: 5

. . . Q . . . .

. . Q . . . . .

. Q . . . . . .

. . . . . Q . .

. . . . . Q . .

. . . . . . Q .

. . Q . . . . .

. . . . . Q . .

cost : 11

. . . Q . . . .

Q . . . . . . .

. Q . . . . . .

. . . . . Q . .

. . . . . Q . .

. . . . . . Q .

. . Q . . . . .

. . . . . Q . .

cost : 7

. . . Q . . . .

Q . . . . . . .

. Q . . . . . .

. . . . . . . Q

. . . . . Q . .

. . . . . . Q .

. . Q . . . . .

. . . . . Q . .

cost : 4

. . . Q . . . .

Q . . . . . . .

. . . . Q . . .

. . . . . . . Q

. . . . . Q . .

. . . . . . Q .

. . Q . . . . .

. . . . . Q . .

cost : 2

. . . Q . . . .

Q . . . . . . .

. . . . Q . . .

. . . . . . . Q

. Q . . . . . .

. . . . . . Q .

. . Q . . . . .

. . . . . Q . .

cost : 0

The search sequences (3) :

Path cost: 4

. . . . . Q . .

. Q . . . . . .

. . . . . . Q .

Q . . . . . . .

Q . . . . . . .

. Q . . . . . .

. . . . . . . Q

Q . . . . . . .

cost : 5

. . . . . Q . .

. Q . . . . . .

. . . . . . Q .

Q . . . . . . .

. . Q . . . . .

. Q . . . . . .

. . . . . . . Q

Q . . . . . . .

cost : 3

. . . . . Q . .

. Q . . . . . .

. . . . . . Q .

Q . . . . . . .

. . Q . . . . .

. . . . Q . . .

. . . . . . . Q

Q . . . . . . .

cost : 1

. . . . . Q . .

. Q . . . . . .

. . . . . . Q .

Q . . . . . . .

. . Q . . . . .

. . . . Q . . .

. . . . . . . Q

. . . Q . . . .

cost : 0

The search sequences (4) :

Path cost: 6

. . . . . . . Q

. . Q . . . . .

. . . Q . . . .

. . . Q . . . .

. . Q . . . . .

. . . . . . Q .

. . . . . Q . .

Q . . . . . . .

cost : 8

. . . . . . . Q

. Q . . . . . .

. . . Q . . . .

. . . Q . . . .

. . Q . . . . .

. . . . . . Q .

. . . . . Q . .

Q . . . . . . .

cost : 6

. . . . . . . Q

. Q . . . . . .

. . . Q . . . .

Q . . . . . . .

. . Q . . . . .

. . . . . . Q .

. . . . . Q . .

Q . . . . . . .

cost : 4

. . . . Q . . .

. Q . . . . . .

. . . Q . . . .

Q . . . . . . .

. . Q . . . . .

. . . . . . Q .

. . . . . Q . .

Q . . . . . . .

cost : 3

. . . . Q . . .

. Q . . . . . .

. . . Q . . . .

Q . . . . . . .

. . Q . . . . .

. . . . . . . Q

. . . . . Q . .

Q . . . . . . .

cost : 1

. . . . Q . . .

. Q . . . . . .

. . . Q . . . .

. . . . . . Q .

. . Q . . . . .

. . . . . . . Q

. . . . . Q . .

Q . . . . . . .

cost : 0

**success count 500**

**success rate 100.0 %**

**failure count 0**

**failure rate 0.0 %**

**Average no. of Steps when success: 5.094**

**Average no. of Steps when failure: 0**

**total reset count : 5608**

**Avg reset count : 1.6906843533313234**

**STATISTICS:**

**Hill Climbing:**

success count 74

success rate 14.799999999999999 %

failure count 426

failure rate 85.2 %

Average no. of Steps when success: 5.0675675675675675

Average no. of Steps when failure: 4.084507042253521

**Hill Climbing using sideway moves:**

success count 467

success rate 93.4 %

failure count 33

failure rate 6.6000000000000005 %

Average no. of Steps when success: 19.946466809421842

Average no. of Steps when failure: 59.21212121212121

**Random Restart without sideway moves:**

success count 500

success rate 100.0 %

failure count 0

failure rate 0.0 %

Average no. of Steps when success: 5.214

Average no. of Steps when failure: 0

total reset count : 2791

Avg reset count : 5.582

**Random Restart with sideway moves:**

success count 500

success rate 100.0 %

failure count 0

failure rate 0.0 %

Average no. of Steps when success: 5.094

Average no. of Steps when failure: 0

total reset count : 5608

Avg reset count : 1.6906843533313234

**RESULTS:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number of Queens** | **Search Used** | **Success Rate and Number of steps** | **Failure Rate and Number of steps** | **Avg Number of Restarts** |
| 8 | Hill-Climbing | Rate: 14.80% | Rate: 85.20% | No Restarts |
| Steps: 5.07 | Steps: 4.0 |
| 8 | Hill-Climbing with Sideway moves | Rate: 93.4 % | Rate: 6.60% | No Restarts |
| Steps: 19.95 | Steps: 59.21 |
| 8 | Random-restart without Sideway moves | Rate: 100% | Rate: 0.00% | 5.58 |
| Steps: 5.02 | Steps: 0.00 |
| 8 | Random-restart with Sideway moves | Rate: 100% | Rate: 0.00% | 1.69 |
| Steps: 5.00 | Steps: 0.00 |

**OBESERVATIONS:**

The success rate is highest when Hill Climbing with sideways method is used and it reduces drastically from 93% to 15% when Hill climbing without sideway moves is used. The failure rate reduces from 85% to 6% when Hill Climbing with sideways method is used. Also, the success rate for hill climbing using random restart with or without sideway moves is 100%.