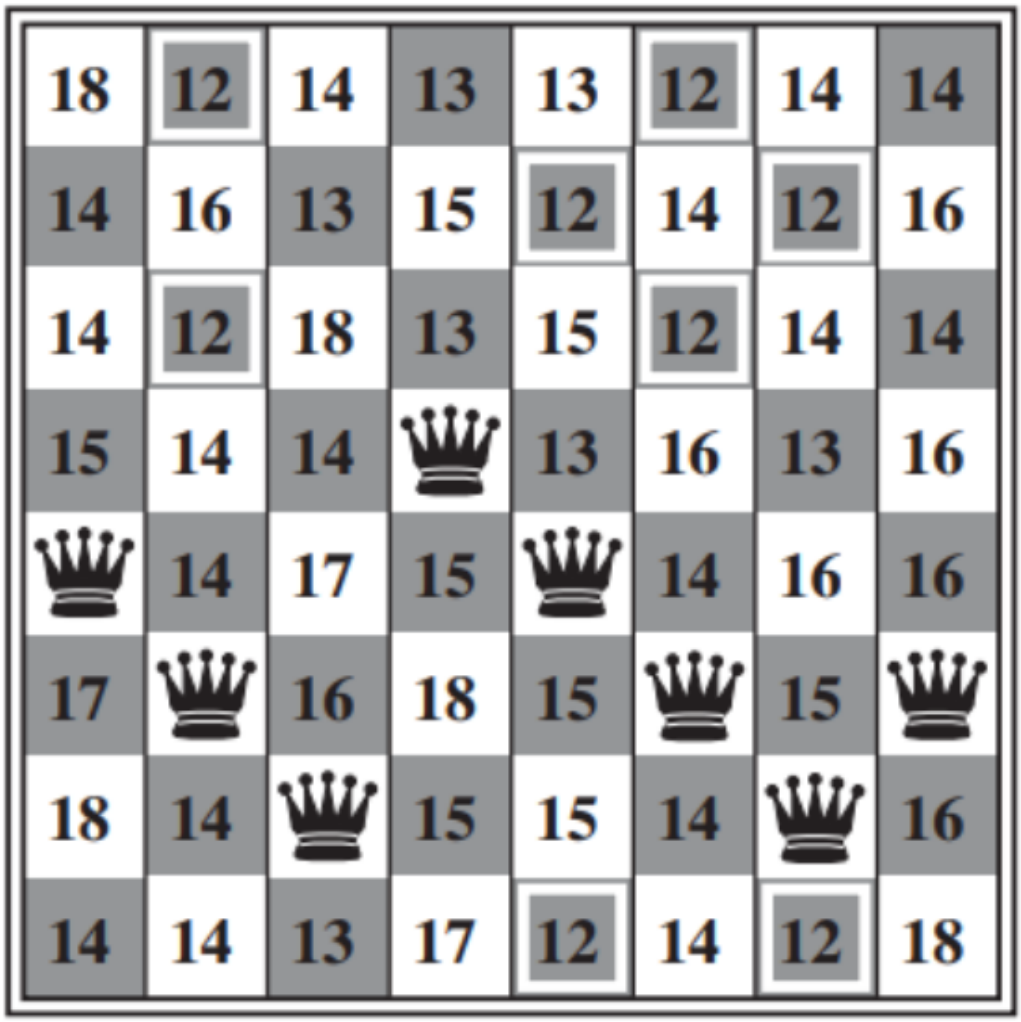


N-Queens Using Local Search Algorithm

Abhinav Pandey , Akhila U Hegde
(20GANSE002) (20GANSE012)

INFORMATION SCIENCE ENGINEERING
5th Sem 3rd Year
UVCE Bengaluru-560037

Overview

An attempt to find the N-Queens Solution Using Hill Climbing Search Algorithm.

Goals

1. To Illustrate N-Queens Solution by Greedy Technique.
2. Implementing Hill Climb Search and Find the Local Minima

Specifications

The project is dissected into two parts, The first part illustrates a few examples of the implementation for the N-Queens Problem using Hill Climb Search. The Second part is to cross-verify the solutions drawn out using C++ code

1) **Illustrations of the Solution :-** The Iterative boards are drawn and found out in this section
2) **C++ Implementation :-** The C++ Code to Implement the Same will be mapped out in this Section.

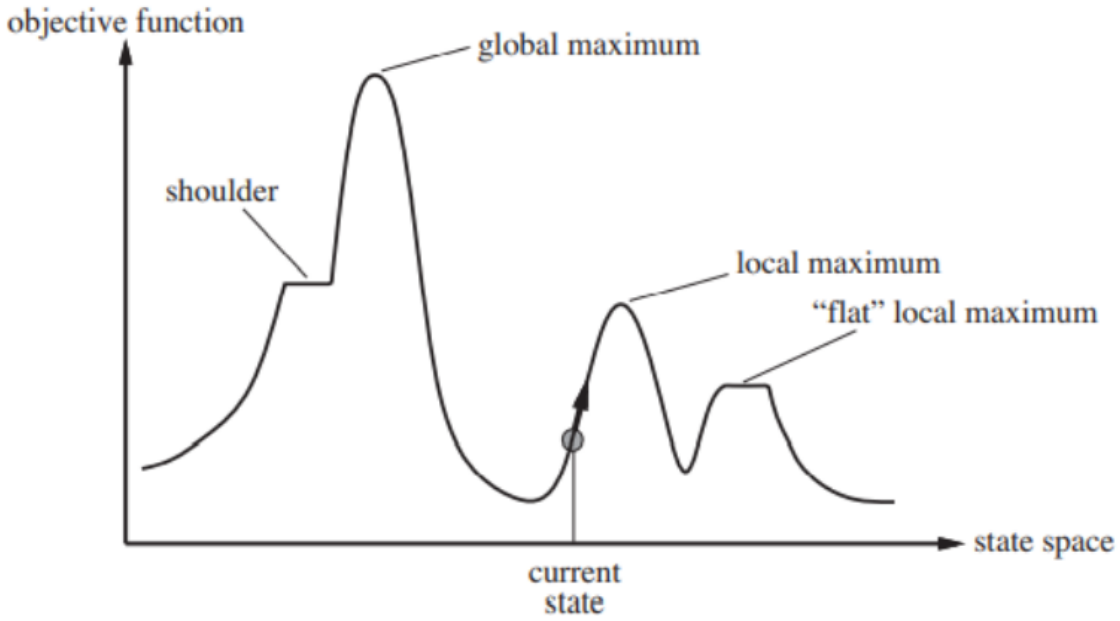
Rules of the Game

The N-Queens puzzle is the problem of placing n chess queens on an n×n chessboard so that no two queens attack each other. The Eight Queens problem is the n-queens problem for a standard size chessboard.

Approach for the Implementation

The approach we follow to try to find the solution for the problem is, we use "hill climbing algorithm".

We perform a local search on the Chess board to find the local minima that is available to us, and greedily place our queen of that column into that coordinates, and recalculate the board and continue.

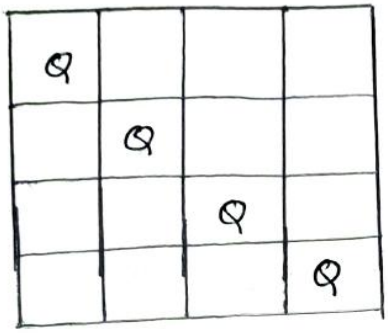


The Drawback with the Hill Climbing approach is we cannot ensure that the coordinate that we picking will make us reach the global minima(which is the feasible solution)

In out case if h(x)=0 we say we have solved the N-queens problem

PART-A (Illustrations of N-Queens Problem using hill climb Algoritum)

Eg 1> 4×4 N-queens



(distinct pairs

* The no. of attacking queens in this configuration is → 6

6	4	5	4
4	6	4	5
5	4	6	4
4	5	4	6

* We greedily choose 4, as thats the least minimum available to us.

* The next Search tree will be as follows

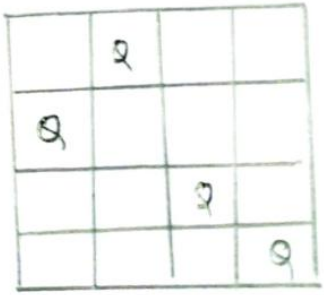
4	4	4	4
2	6	3	3
2	4	4	4
2	5	3	4

* Now, we greedily choose 2, the least minimum available to us

* The Next Search tree will be as follows

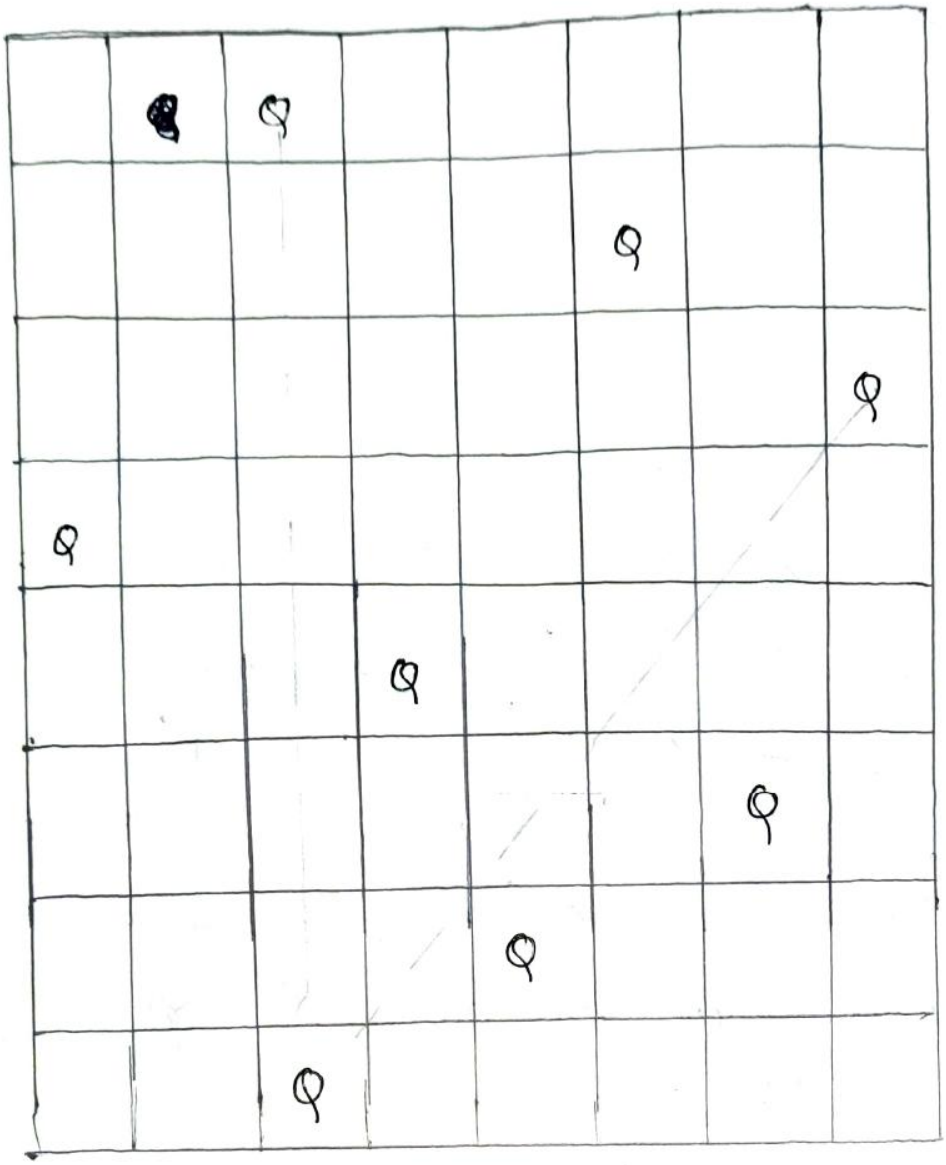
4	2	2	2
2	4	3	3
2	3	2	3
2	3	3	2

* Since we encountered same minima expanding it might give us the result as it can be a shoulder (or) flat local min



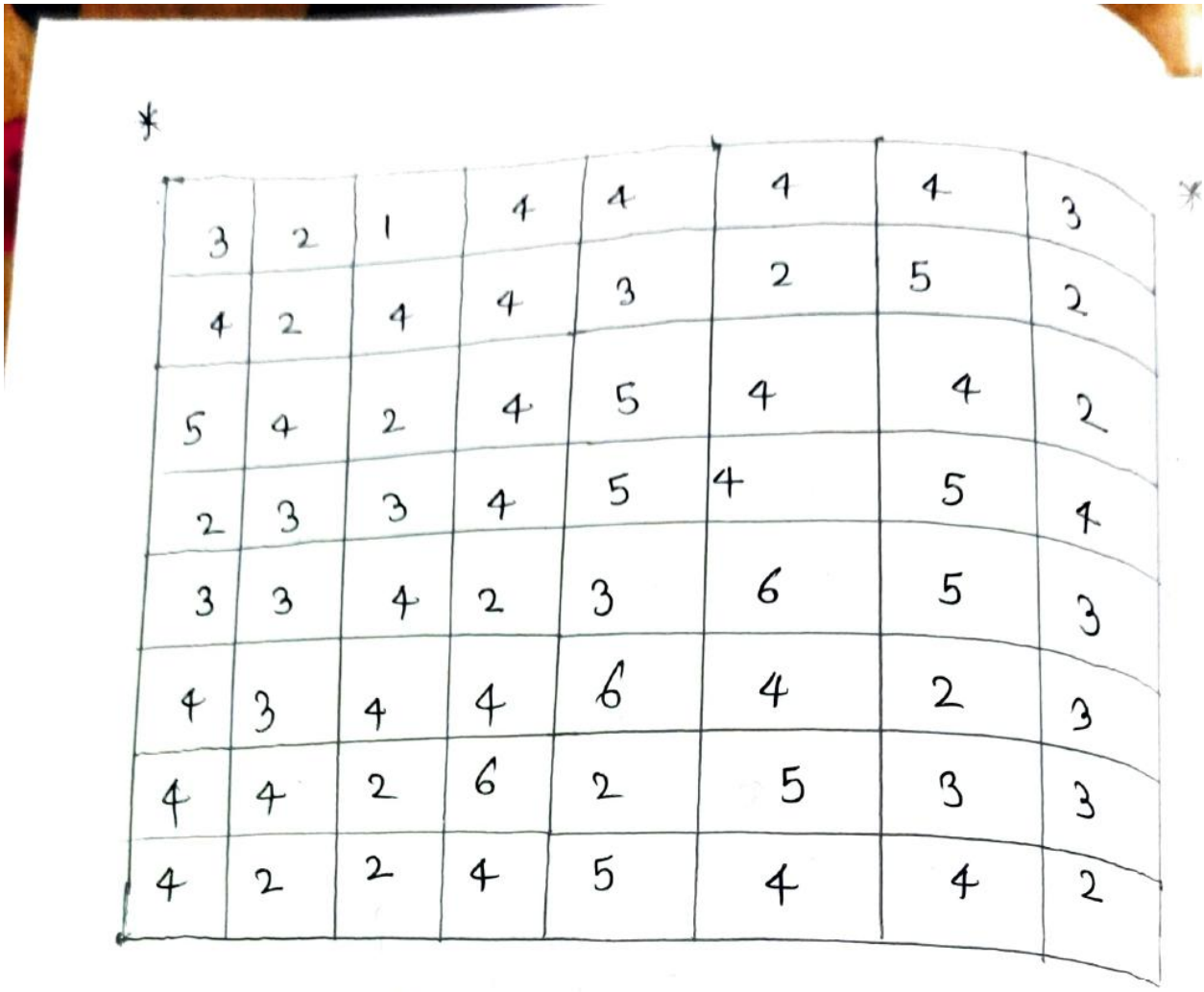
No. of attacking distinct pairs = 2.

eg 2> 8 × 8 N-queens



* No. of Attacking pairs = 2

* Constructing the Search tree for this configuration of Queens



3	2	1	4	4	4	4	3
4	2	4	4	3	2	5	2
5	4	2	4	5	4	4	2
2	3	3	4	5	4	5	4
3	3	4	2	3	6	5	3
4	3	4	4	6	4	2	3
4	4	2	6	2	5	3	3
4	2	2	4	5	4	4	2

* The minimum available is **1**

So we move our queen of 3rd column to (1,3)

* The Search Space tree obtained after moving our piece to (1,3) is obtained as follows,

*

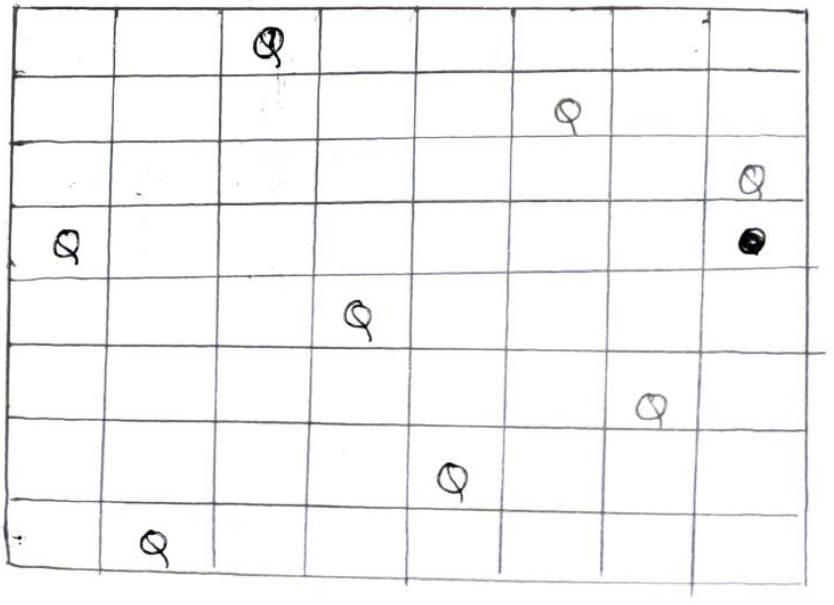
3	2	1	4	4	4	4	4
3	2	4	4	2	1	4	2
5	3	2	3	5	3	3	1
1	2	3	3	4	4	3	4
2	2	4	3	4	3	1	4
2	2	4	3	4	3	1	4
3	2	2	4	1	4	2	3
2	0	2	2	3	2	2	1

* The minimum we obtaining here is at (8,2) with h(n) = 0.

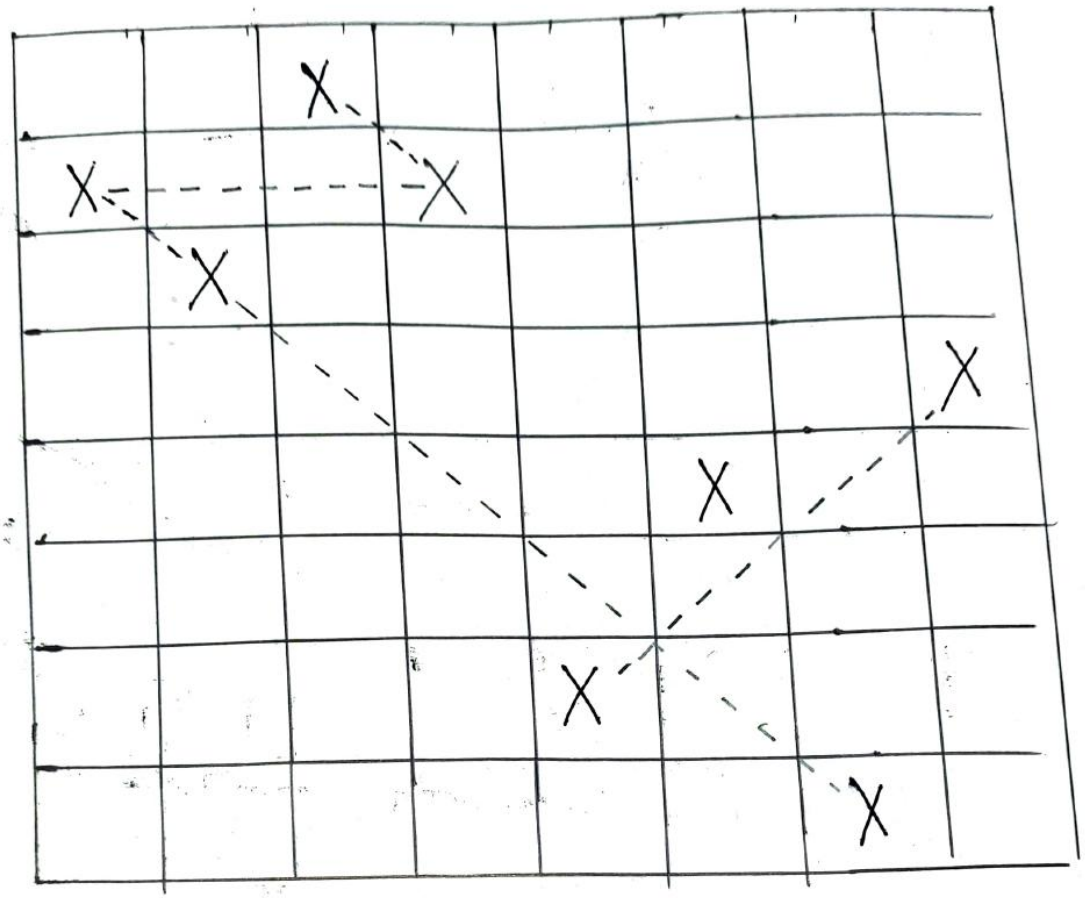
So we move our 2nd column Queen to (8,2) & recalculate the

1	2	[0]	2	2	2	2	2
2	2	3	2	1	[0]	3	2
3	3	1	2	3	2	3	[0]
[0]	2	2	2	3	3	2	3
1	2	3	[0]	2	3	3	2
1	2	3	3	3	2	[0]	2
3	2	2	3	[0]	3	1	2
2	[0]	2	2	3	2	2	1

The final solution to the problem is,



Ex 3) 8 x 8 N- Queens



Initially, the distinct pair of Attacking queens are :- 6

→ Constructing the search tree for this configuration of Queens

4	7	6	6	8	7	5	6
6	7	9	6	7	9	6	7
6	6	7	6	8	7	6	7
5	7	9	5	7	9	6	6
5	5	7	8	6	6	9	6
3	4	5	5	9	8	5	7
4	5	6	6	6	10	5	8
4	5	7	7	6	8	6	6

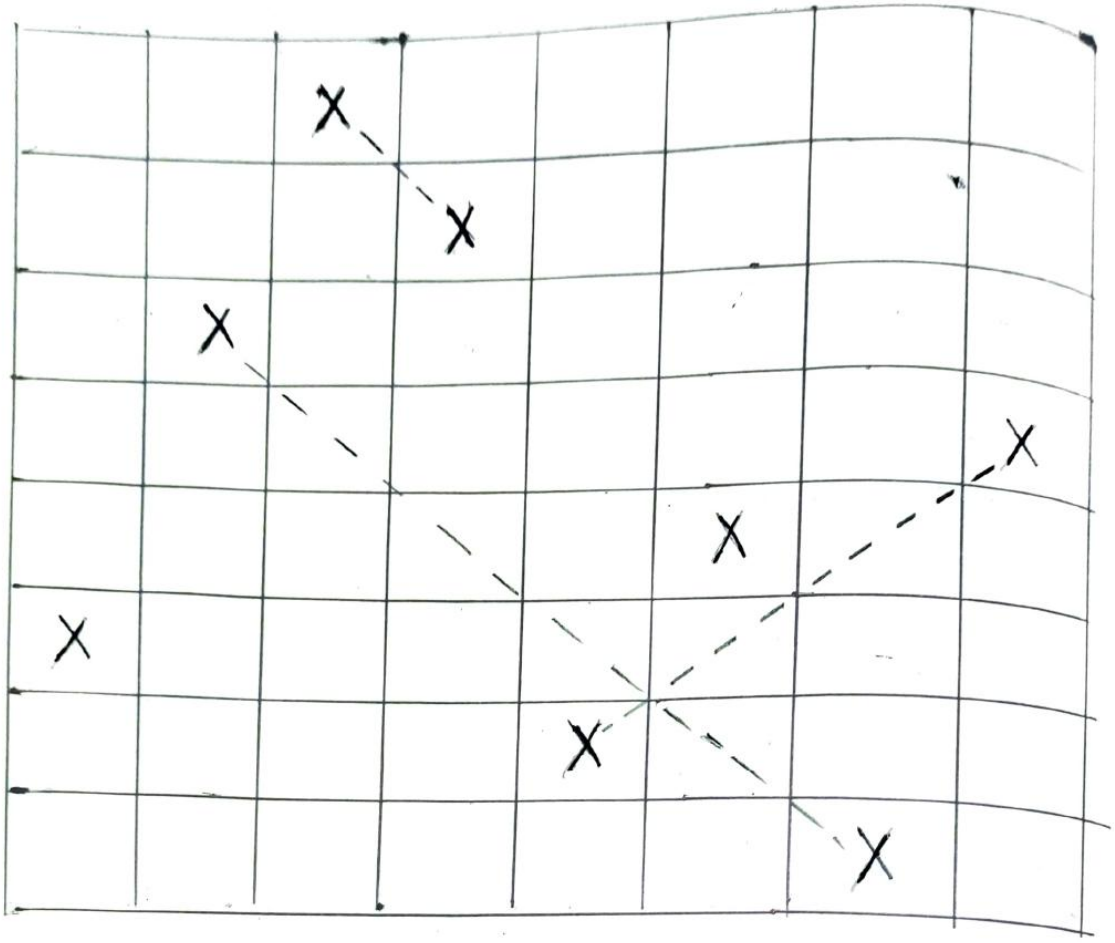
→ The minimum available ps 3, So, we have to move our queen of first column to 6th row from second row.

→ Now the search space tree obtained as follo

4	4	3	4	5	5	3	3
6	4	5	3	4	5	3	3
6	3	4	5	5	4	4	4
5	5	6	3	4	6	4	3
5	4	4	5	3	3	7	3
3	3	3	4	6	6	4	5
4	4	3	4	3	6	3	5
4	3	5	5	3	5	3	3

→ Even here also we are obtaining minimum as 3. Hence local minima that was encountered was 3 which can be a shoulder or flat local maxima

→ The final solution of the problem is as follows



The attacking pairs of queens are shown, which is final solution

PART-B (C++ IMPLEMENTATION OF THE SOLUTION)

```
#include <bits/stdc++.h>

using namespace std;
bool check(pair<int,int>a,pair<int,int>b)
{
   if(a.first==b.first||a.second==b.second)
      return true;
   else if(abs(a.first-b.first)==abs(a.second-b.second))
      return true;
   return false;
}
long long findpairs(vector<pair<int,int>> coordinates,int flag)
{ long long cnt=0;
   for(int i=0;i<coordinates.size();i++)
 {
   for(int j=i+1;j<coordinates.size();j++)
   {
      if(check(coordinates[i],coordinates[j])==true)
         {
           if(flag)
          cout<<coordinates[i].second<<"
"<<coordinates[i].first<<"---"<<coordinates[j].second<<" "<<coordinates[j].first<<"\n";
         cnt++;
      }
   }
 }
 return cnt;
```

```
}
int main(){
 int n;
 cin>>n;
 char a[n][n];
 cout<<"Enter The Initial Arrangement of the Board\n";
 cout<<"Mark X where the queens are placed and * where its empty";
for(int i=0;i<n;i++)
    for(int j=0;j<n;j++)
        cin>>a[i][j];
std::vector<pair<int,int>> coordinates;
for(int i=0;i<n;i++)
    for(int j=0;j<n;j++)
    {
        if(a[i][j]=='X'||a[i][j]=='x')
            coordinates.push_back({j,i});
    }
    cout<<"\n";
/* Display of all the coordinates Entered
for(int i=0;i<coordinates.size();i++)
    cout<<coordinates[i].first<<" "<<coordinates[i].second<<"\n";
*/
 sort(coordinates.begin(),coordinates.end());
long long cnt=0;
 cnt=findpairs(coordinates,0);

 cout<<"Initially for the Inputted Board, the no. Distinct pairs of attacking queens are:- ";
 cout<<cnt<<"\n";
```

```
 long long tempcnt=cnt;
 long long mini=INT_MAX;
 int t=5;
 long long prevtempcnt=INT_MAX;
while(tempcnt<prevtempcnt)
{ long long ix=-1,iy=-1;
prevtempcnt=tempcnt;
for(int i=0;i<n;i++)
   { for(int j=0;j<n;j++)
    {  long long tempx=coordinates[j].first,tempy=coordinates[j].second;
       coordinates[j].first=j;
       coordinates[j].second=i;
       cnt=findpairs(coordinates,0);
       cout<<cnt<<" ";
       if(tempcnt>cnt)
      {
       tempcnt=cnt;
       ix=i;
       iy=j;
      }
       coordinates[j].first=tempx;
       coordinates[j].second=tempy;
    }
   cout<<"\n";}
   coordinates[iy].first=iy;
   coordinates[iy].second=ix;
   cout<<"\n";

}
```

```
for(int i=0;i<n;i++)
{for(int j=0;j<n;j++)
 {
   if(coordinates[j].first==j&&coordinates[j].second==i)
      cout<<"X";
   else cout<<"*";
 }
cout<<"\n";
}
cout<<"LOCAL MINIMA THAT WAS ENCOUNTERED WAS :- "<<tempcnt<<"\n";
 cout<<"The current Attacking pairs are :- \n";
 int flush=findpairs(coordinates,1);

}
```

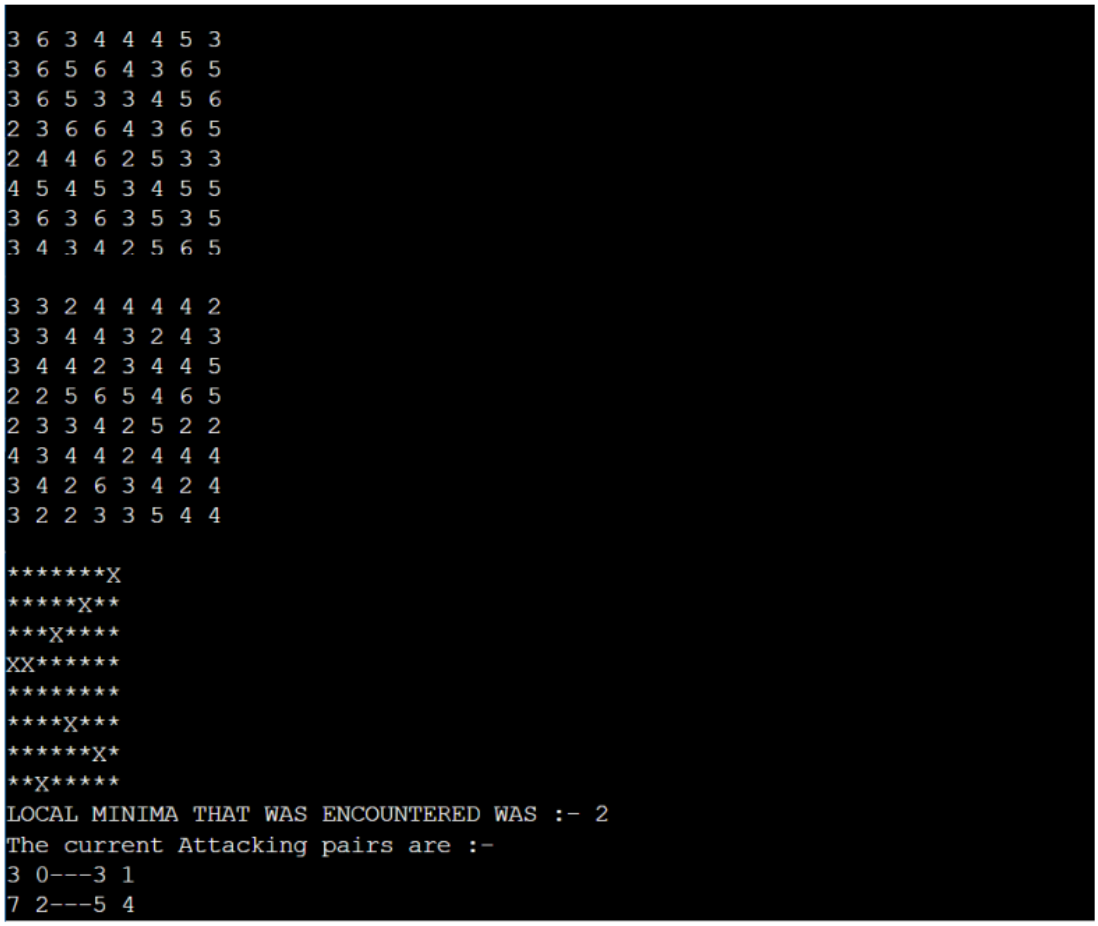
CODE LINK:- https://github.com/AbhinavPandey1911/Hill-Climb-Nqueens-Local-Search

OUTPUT :-

```
8
Enter The Initial Arrangement of the Board
Mark X where the queens are placed and * where its empty*******X
X*******
*****X**
*X******
***X****
****X***
******X*
**X*****


Initially for the Inputted Board, the no. Distinct pairs of attacking queens are:- 7
7 9 7 5 6 6 8 7
7 9 7 5 6 6 11 6
7 11 9 4 5 7 8 8
6 7 10 5 8 6 11 6
7 8 8 7 6 8 8 7
7 8 10 5 7 7 8 7
6 12 7 6 6 9 7 6
9 8 7 4 5 8 11 7

5 7 4 5 4 4 5 4
4 6 5 5 5 3 7 4
6 8 7 4 4 4 6 7
4 4 7 5 6 3 8 4
4 5 4 7 3 5 4 4
6 5 6 5 4 4 6 5
4 8 4 6 4 5 4 5
6 5 4 4 3 5 7 5
```


3 6 3 4 4 4 5 3
3 6 5 6 4 3 6 5
3 6 5 3 3 4 5 6
2 3 6 6 4 3 6 5
2 4 4 6 2 5 3 3
4 5 4 5 3 4 5 5
3 6 3 6 3 5 3 5
3 4 3 4 2 5 6 5
3 3 2 4 4 4 4 2
3 3 4 4 3 2 4 3
3 4 4 2 3 4 4 5
2 2 5 6 5 4 6 5
2 3 3 4 2 5 2 2
4 3 4 4 2 4 4 4
3 4 2 6 3 4 2 4
3 2 2 3 3 5 4 4
*******X
*****X**
X*
XX******
****
****X***
******X*
X***
LOCAL MINIMA THAT WAS ENCOUNTERED WAS :- 2
The current Attacking pairs are :-
3 0---3 1
7 2---5 4