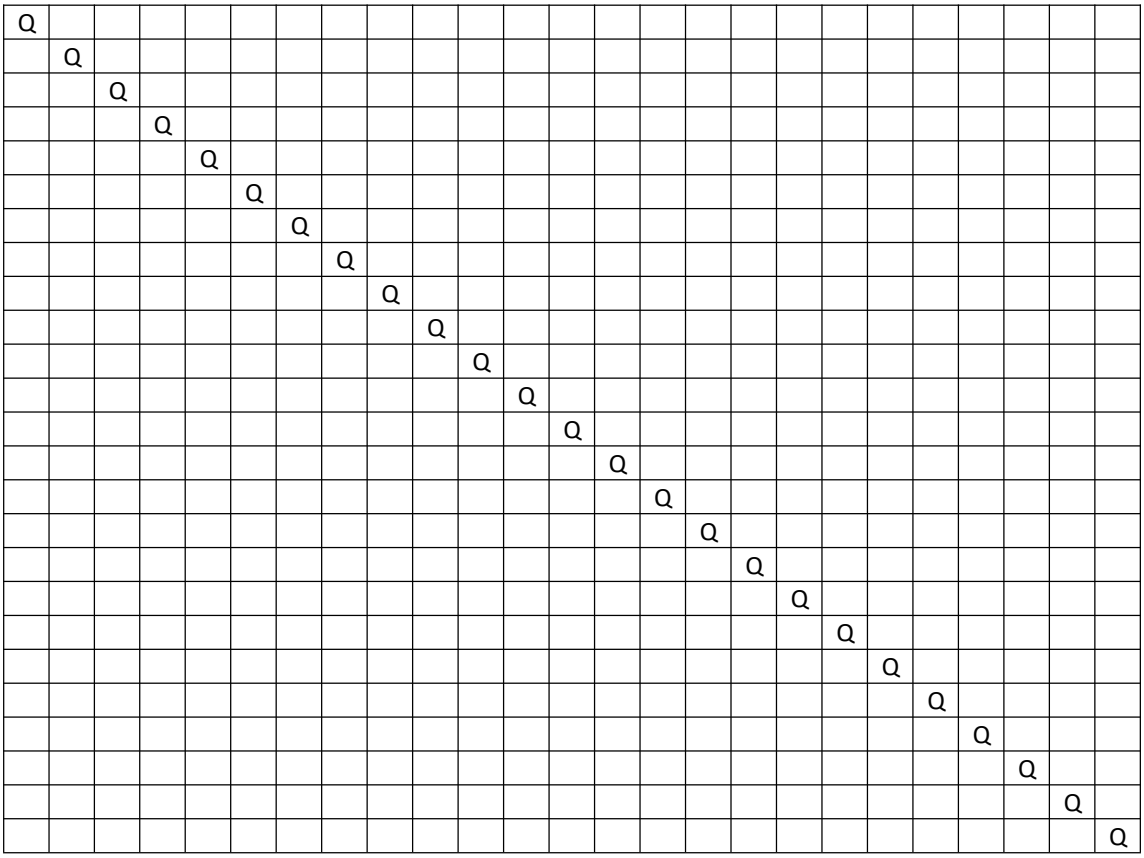
1 (Programming) Hill Climbing :

1. Considering our initial state as the one shown in the figure below.



I have applied the vanilla hill climbing algorithm to the initial state which is shown above where I have used the following parameters.

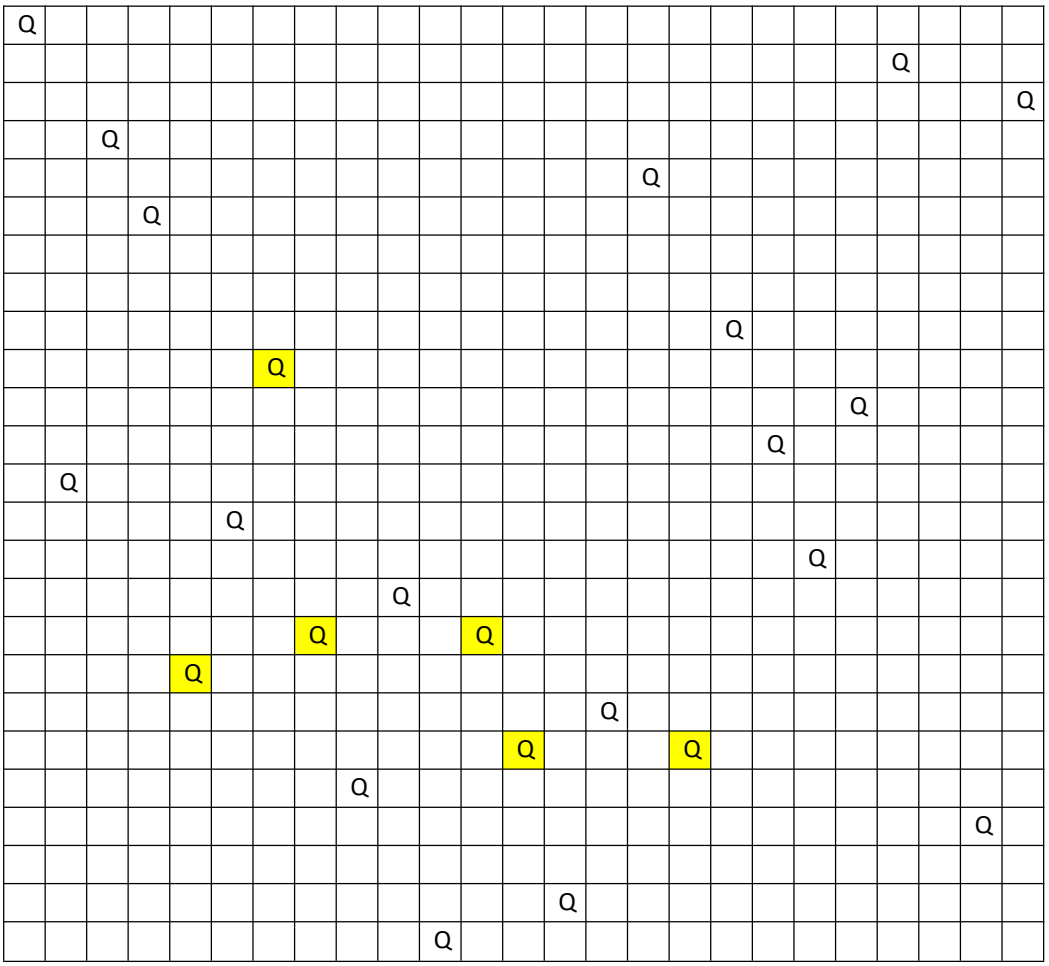
- Score Function : I have used the score function f(s) as no of attacking queens . i.e f(s) = Number of Attacking Queens .For Example in the initial state the score of queen present in column 25 is f(s) = 24 . I am considering the minimum value of f(s) to move in the neighbourhood.

- Neighbourhood : I have taken the neighbourhood of the queen as rows of its corresponding columns . For Example for a queen belonging to column 25 of the chess board the neighbourhood of the queen is the rows present in column 25 .

- APPROACH : I have started to apply the algorithm for the right most queen and I have calculated the score function to the right most queen in its neighbourhood and then changed the position of the queen to a point where score function f(s) is minimum . I am moving from right to left and for each column I am calculating the score function for each row in a column and I am moving the queen to the row for which f(s) is minimum . I will continue this process until I have minimum no of conflicting queens i.e zero in this case . If the no of conflicting queens is not zero and if I cannot find a position for which score function is not Zero for a queen in its neighbourhood then its a stop state.I had done the process by hand.

Results :

I have obtained the final state as follows .

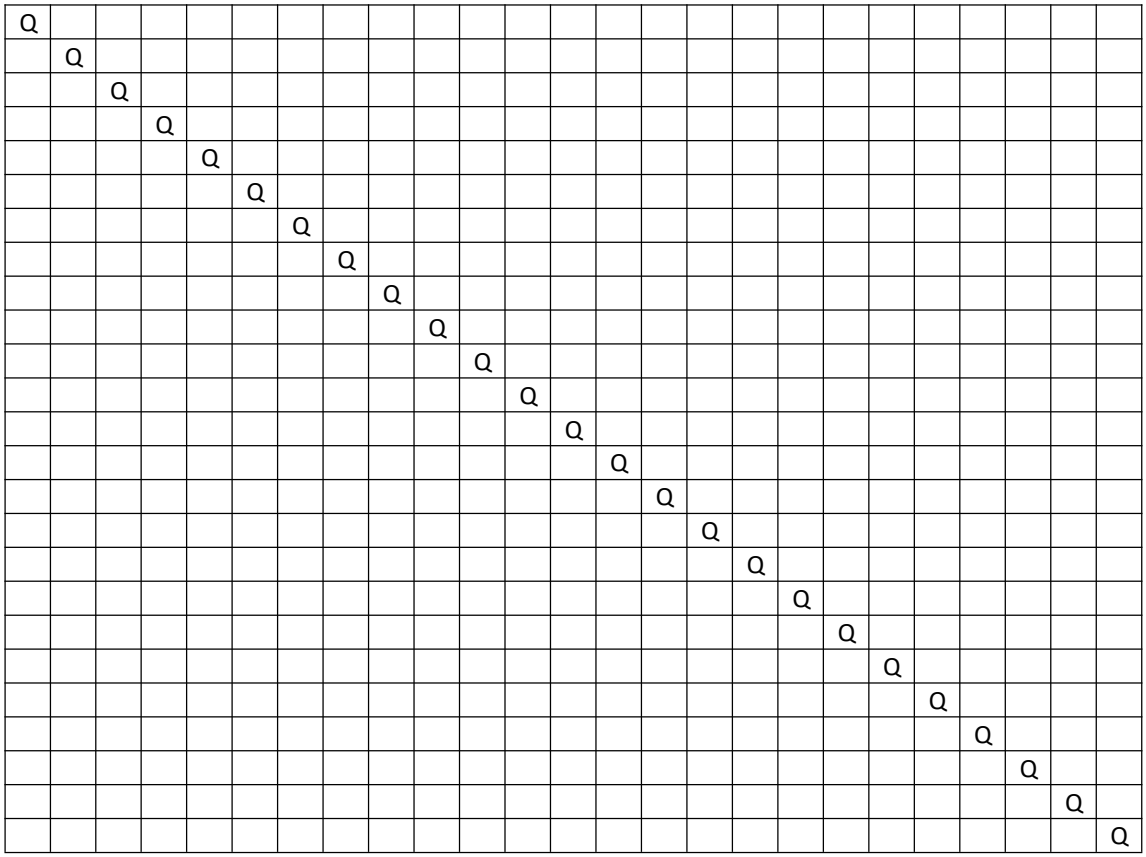


This is the best state I have obtained using the vanilla hill climbing algorithm . I have used the approach stated above to reach to this state . Beyond this state we cannot move a queen in its neighbourhood to a point where the score function is minimum and the no of conflicting queens is less than the obtained .

In my best state the no of conflicting queens = 6 , The 6 conflicting queens are highlighted in yellow

P.T.O

2. The Initial state is as follows



For this initial state I have applied Stochastic Hill Climbing which is variation 2 . The parameters I have used are as follows.

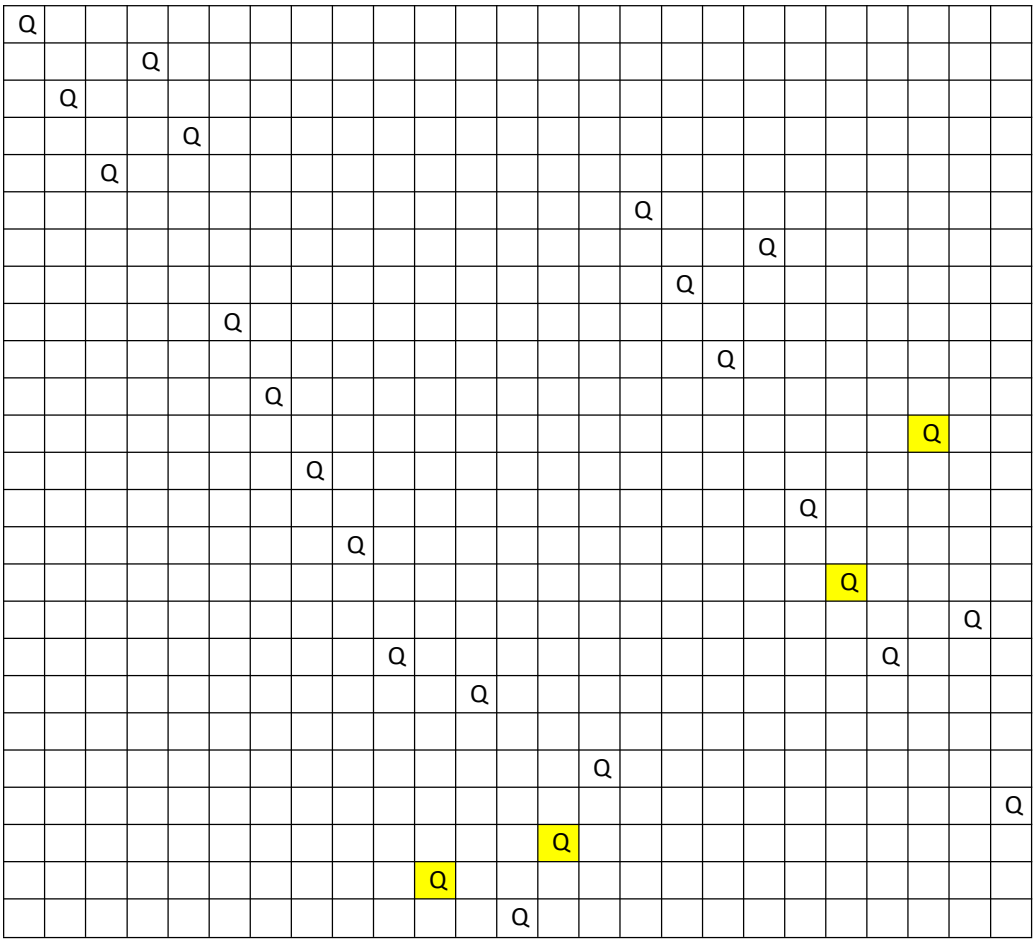
- Score Function : I have used the score function f(s) as no of attacking queens . i.e f(s) = Number of Attacking Queens .For Example in the initial state the score of queen present in column 25 is f(s) = 24 . I am considering the minimum value of f(s) to move in the neighbourhood.

- Neighbourhood : I have taken the neighbourhood of the queen as rows of its corresponding columns . For Example for a queen belonging to column 25 of the chess board the neighbourhood of the queen is the rows present in column 25 .

- APPROACH : I have used the Stochastic Hill Climbing algorithm to solve the problem. I have applied the algorithm to the right most Queen and I have calculated the score function of the queen in its neighbourhood and I have taken the subset of the neighbourhood consisting of the best 5 minimum score function and is greater than the current score function. I had randomly selected a position from that subset consisting of the best 5 minimum score function . I had moved from right to left in the chess board and had done the same process for every queen . I had done 10 iterations before I stopping the process . I had done the process by hand.

Results :

The best state obtained is as follows



I have used the approach stated above to reach to this state .
In my best state No of Conflicting Queens = 4 , the 4 queens are highlighted in yellow .

Vanilla Hill Algorithm works fine if the Neighbourhood is large , but for a smaller neighbourhood the best results depend on the initial start point and the score function used . If the initial start point leads us to a local optima we will be stuck at the local optima due to greediness of the score function . For this problem I am not able to reach the optimal solution which is no of conflicting queens = 0 because of the greediness of the score function . I have chosen a neighbourhood as the rows of a column which is neither too small or too large , its sufficient to find the optimal solution if the vanilla hill climbing algorithm is not greedy .I have chosen Stochastic Hill Climbing to reduce the greediness to find an optimal solution . For a good neighbourhood Stochastic process is effective when the subset is not too large nor too small , if subset is large the we need more and more no of iterations and we may not find the optimal solution , if the subset is too small it will be behaving almost equivalent to vanilla hill climbing which may not find the optimal solution. For my case I have taken the subset as 5 best minimum score function values ,so it has taken a large no of iteration , if we continue the process for more iterations we might land up with a optimal solution as well . Stochastic Hill Climbing is better compared to

vanilla Hill Climbing when the neighbourhood is not too large nor too small , but in Stochastic Hill Climbing the no of iterations required to reach optimal solution might be high, depending on the subset we choose .