**Greedy search for N queens:**

* We created a board with queens with random weights and placed them randomly one in each column.
* Then we find the No. of attacks on each queen to find the H value of the board and we divide no of attacks by two because **H** is no of attacking **pairs.**
* We created a H Matrix for the board. It will give the H value of board if we move Queen to each position in its own column.
* Then we find the good position where our H value is minimum and move the queen to that position
* We run this loop until we get the H value of the board turns out to be 0.

Effective Branching Factor:

* the Effective Branching Factor for this Algorithm is N (no. of queens on the board).

Board size solved in 1 Min.:

* we are able to solve **9** queen problem in a minute.

**A\* Algorithm for N queens:**

* In this approach we created a random board with N queens of random weights, one in each column.
* We defined the heuristic H to be the number of pair of queens under attack. To find the H value, for each of the queen, we traverse through all its moving directions. We increment a count variable if there is any other queen in its attacking range. After continuing this process for all the queens, we finally divide the count by 2, since we require the pair of queens under attack.
* We iterate through each column, check the presence of the queen, and store the positions and the weights of the queen in separate lists.
* We form a H matrix, which gives us the H value of the board if the queen in each column in moved to all the positions.
* Similarly, we also form a Cost matrix, whose values helps us to determine the cost to move a queen to all positions in its column. The formula for the cost calculation of each queen is, Weight^2 \* No. of tiles moved.
* We calculate the Total cost matrix by adding the Cost and H matrices.
* From this Total cost matrix, we find the least value. And in the column in which this least value is present, we move the queen in that column to the position of the least value.
* We then recalculate the H value to this modified board.
* We continue this process until none of the queens are in attacking positions, i.e., we get a H value of 0.

Effective Branching Factor:

* Effective Branching Factor for this Algorithm is N\*(N-1), where N is the number of queens on the board.

Board size solved in 1 minute:

* we are able to solve **6** queen problem in a minute.

**Hill Climbing algorithm for 9/8 heavy queens:**

* We chose the random restart method of the hill climbing algorithm
* We create a random board of size 8 with 9 queens in it.
* Set this board as the initial node.
* Set the time limit for the process.
* Find all the neighbours of the initial node and calculate the cost for each neighbour, with the formula given, 100\*attacking pairs + cost to move each weighted queen
* Find the neighbour with the least cost. If this cost in lower than the cost of the initial node, then this neighbour becomes the initial node. If not, we perform a random restart and create a new random board and set it to the initial node.
* We continue the same process until the time limit is exceeded.
* The final board which was set as node will be the outcome of the algorithm.