# N-Queens Problem

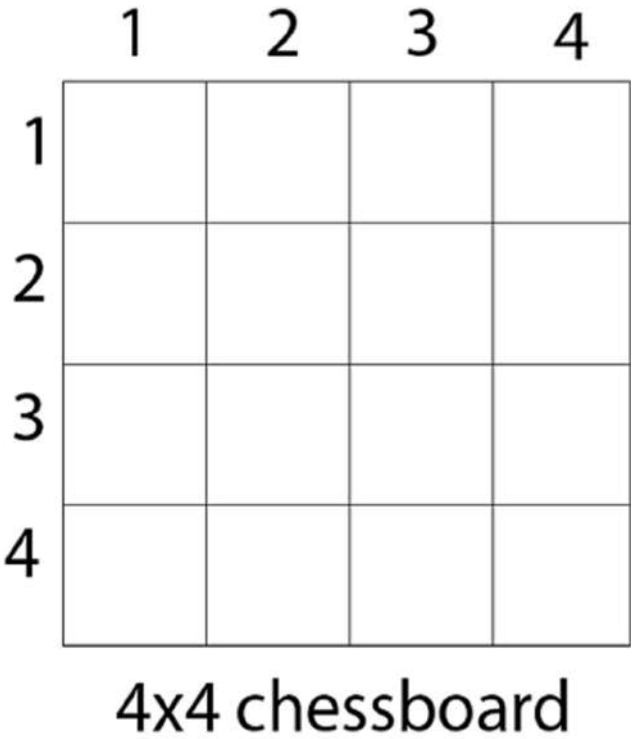
The N-Queens problem is a classic computer science problem that involves placing N chess queens on an N x N chessboard such that no two queens threaten each other. In other words, no two queens should be placed in the same row, column, or diagonal of the chessboard.

The problem can be solved using the backtracking algorithm. The basic idea is to start with an empty board and to recursively place queens on the board, one column at a time. At each step, the algorithm checks if the current queen can be placed on the board without threatening any of the previously placed queens. If a solution is found, it is returned. If not, the algorithm backtracks and tries the next position.

N - Queens problem is to place n - queens in such a manner on an n x n chessboard that no queens attack each other by being in the same row, column or diagonal.

It can be seen that for n =1, the problem has a trivial solution, and no solution exists for n =2 and n =3. So first we will consider the 4 queens problem and then generate it to n - queens problem.

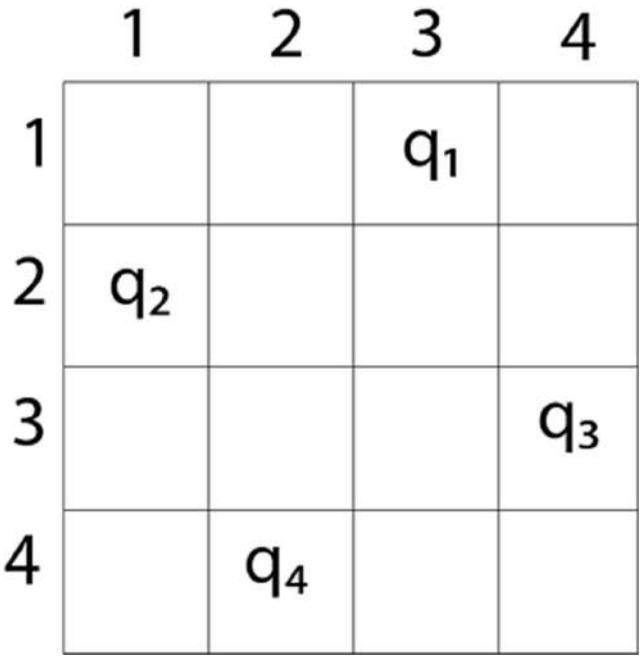
Given a 4 x 4 chessboard and number the rows and column of the chessboard 1 through 4.




4x4 chessboard

Since, we have to place 4 queens such as $q_1$ $q_2$ $q_3$ and $q_4$ on the chessboard, such that no two queens attack each other. In such a conditional each queen must be placed on a different row, i.e., we put queen "i" on row "i."

Now, we place queen $q_1$ in the very first acceptable position (1, 1). Next, we put queen $q_2$ so that both these queens do not attack each other. We find that if we place $q_2$ in column 1 and 2, then the dead end is encountered. Thus the first acceptable position for $q_2$ in column 3, i.e. (2, 3) but then no position is left for placing queen '$q_3$' safely. So we backtrack one step and place the queen '$q_2$' in (2, 4), the next best possible solution. Then we obtain the position for placing '$q_3$' which is (3, 2). But later this position also leads to a dead end, and no place is found where '$q_4$' can be placed safely. Then we have to backtrack till '$q_1$' and place it to (1, 2) and then all other queens are placed safely by moving $q_2$ to (2, 4), $q_3$ to (3, 1) and $q_4$ to (4, 3). That is, we get the solution (2, 4, 1, 3). This is one possible solution for the 4-queens problem. For another possible solution, the whole method is repeated for all partial solutions. The other solutions for 4 - queens problems is (3, 1, 4, 2) i.e.




The implicit tree for 4 - queen problem for a solution (2, 4, 1, 3) is as follows:

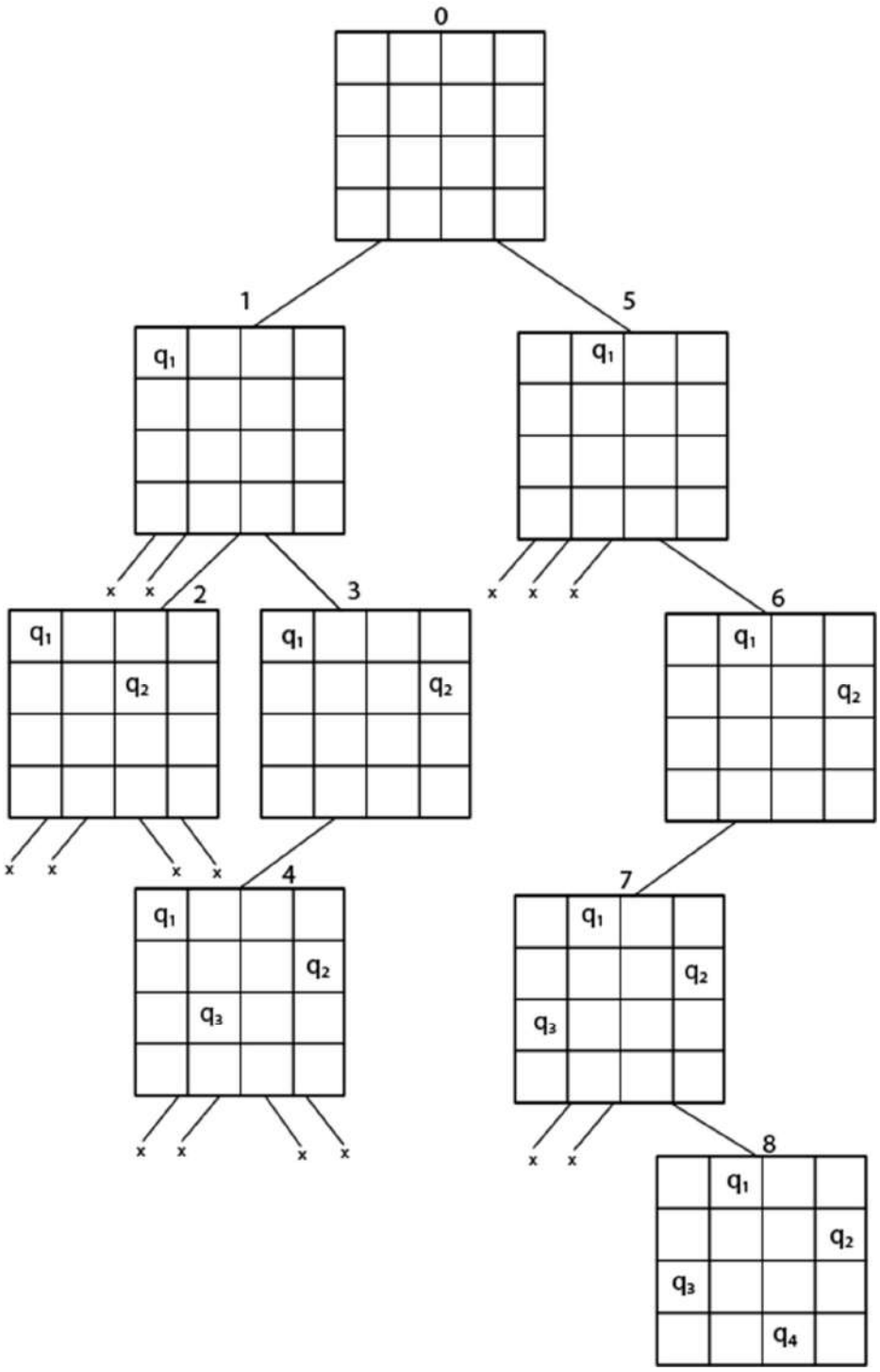
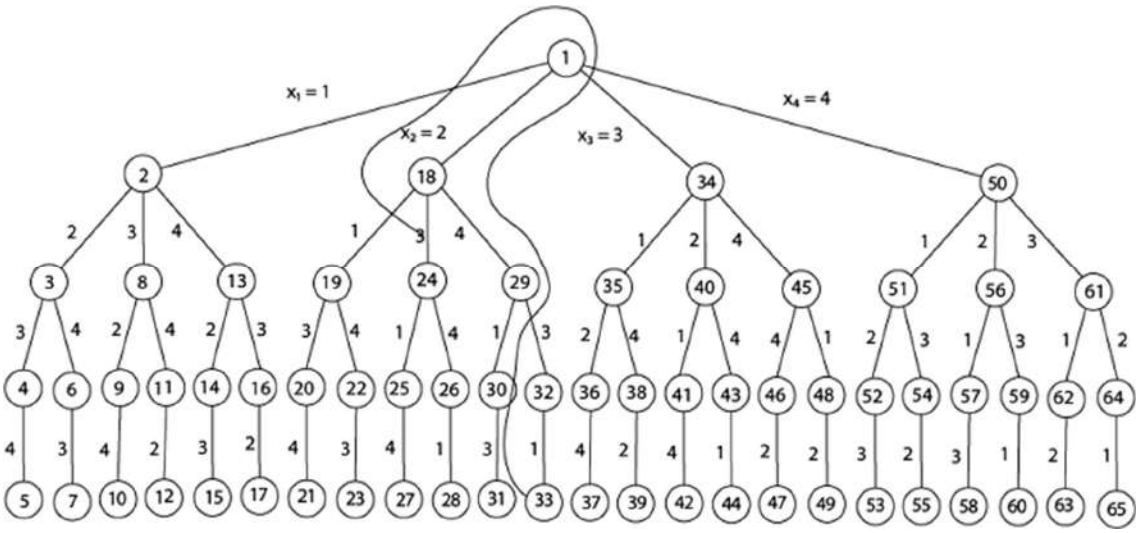


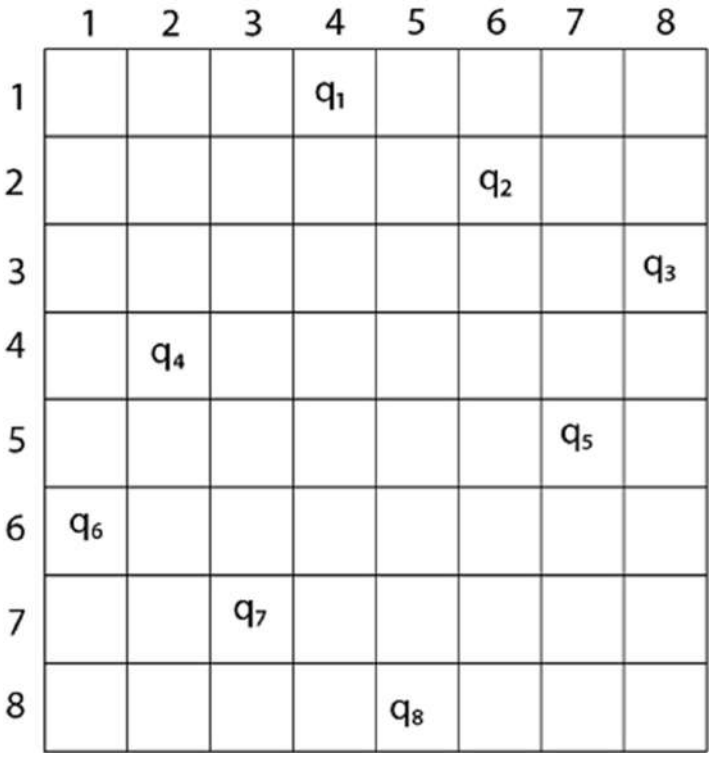

Fig shows the complete state space for 4 - queens problem. But we can use backtracking method to generate the necessary node and stop if the next node violates the rule, i.e., if two queens are attacking.




**4 - Queens solution space with nodes numbered in DFS**

It can be seen that all the solutions to the 4 queens problem can be represented as 4 - tuples ($x_1$, $x_2$, $x_3$, $x_4$) where $x_i$ represents the column on which queen "$q_i$" is placed.

One possible solution for 8 queens problem is shown in fig:




1. Thus, the solution **for** 8 -queen problem **for** (4, 6, 8, 2, 7, 1, 3, 5).
2. If two queens are placed at position (i, j) and (k, l).
3. Then they are on same diagonal only **if** (i - j) = k - l or i + j = k + l.

4. The first equation implies that j - l = i - k.
5. The second equation implies that j - l = k - i.
6. Therefore, two queens lie on the duplicate diagonal **if** and only **if** |j-l|=|i-k|

Place (k, i) returns a Boolean value that is true if the kth queen can be placed in column i. It tests both whether i is distinct from all previous costs $x_1$, $x_2$,....$x_{k-1}$ and whether there is no other queen on the same diagonal.

Using place, we give a precise solution to then n- queens problem.

```
1.  Place (k, i)
2.    {
3.      For j  ←  1 to k - 1
4.       do if (x [j] = i)
5.        or (Abs x [j]) - i) = (Abs (j - k))
6.     then return false;
7.      return true;
8.  }
```

Place (k, i) return true if a queen can be placed in the kth row and ith column otherwise return is false.

x [] is a global array whose final k - 1 values have been set. Abs (r) returns the absolute value of r.

```
1.  N - Queens (k, n)
2.  {
3.    For i  ←  1 to n
4.        do if Place (k, i) then
5.    {
6.      x [k]  ←  i;
7.      if (k ==n) then
8.        write (x [1....n));
9.      else
10.     N - Queens (k + 1, n);
11.   }
12. }
```