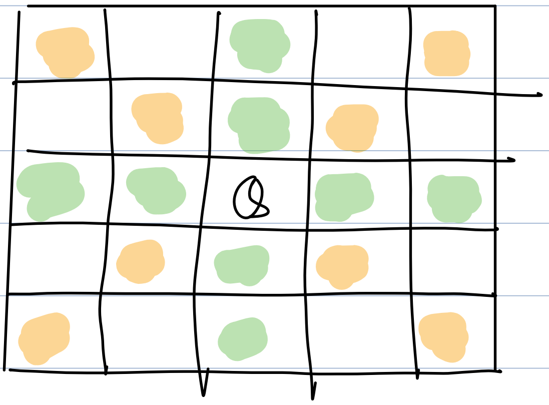
Q N Queens N×N chessboard.

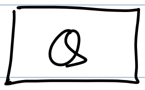
You have to place N Queens such that no Queen can attack /kill any other Queen. . Return the number of ways it can be done.

⇒ Power of a Queen.



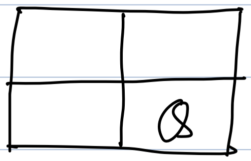
5x5

Ex1 N=1



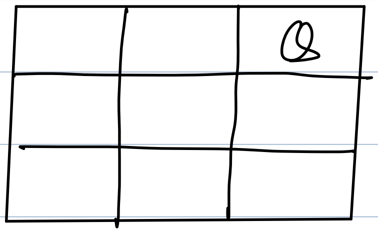
ans = 1.

Ex2 N=2



ans = 0.

Ex3 N=3



ans = 0

Observations

1) In a row, only 1 Queen can be placed.

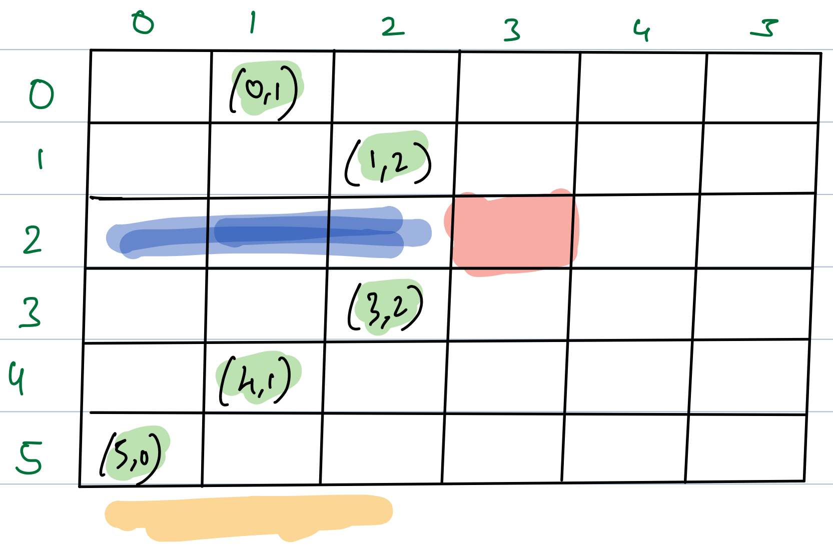
2) In a col, only 1 Queen can be placed.

3) In each row a Queen needs to placed.

4) In each col a Queen needs to placed.

Ex 4 N=4

	Q		
			Q
Q			
		Q	



	0	1	2	3	4	5
0		(0,1)				
1			(1,2)			
2						
3			(3,2)			
4		(4,1)				
5	(5,0)					

row number

HSR

Sum of i & j

HSLL

Difference i & j

HSUL

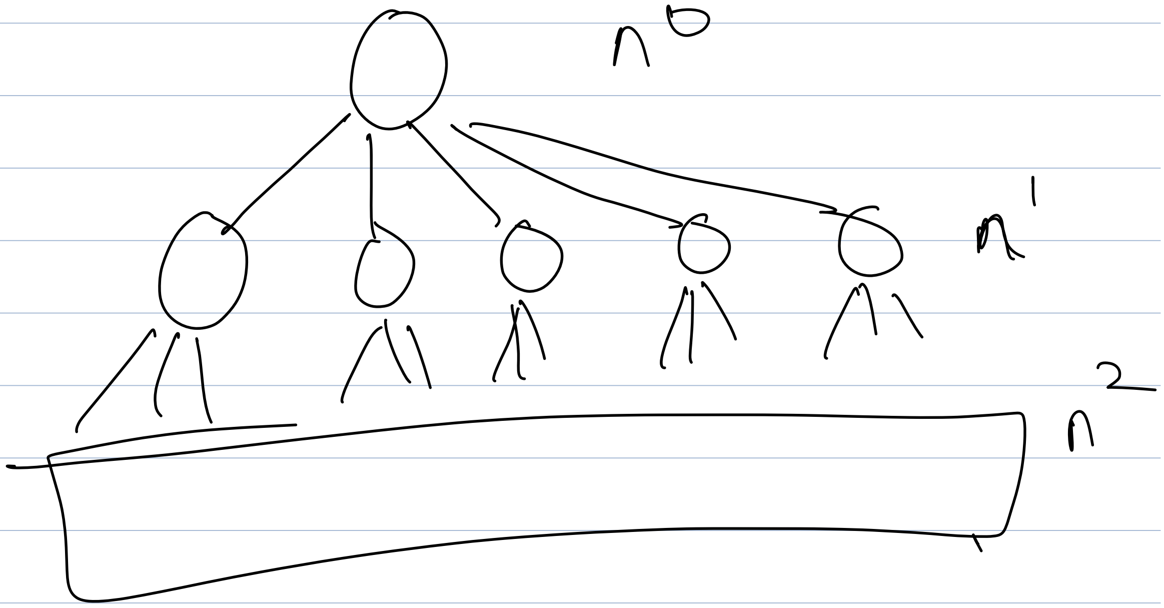
Attack from left

Attack from Upper left

Pseudo Code!

```
                Set : HSR , HSUL, HSLL
                      ↓             ↳ Attack
                  Hash set for        from lower
                    Rows                left
   int ans ⇒ 0;

   void countPossibilities (int n, int col) {

      if (col == n) {
            ans++
            return;
      }
                                        → (i, col)
      for (int i=0 ; i<n ; i++) {

            if (HSR.contains(i))
                  continue;
            if (HSLL.contains(i+col))
                  continue;
            if (HSUL.contains(i-col))
                  continue;

            HSR.insert(i);
            HSLL.insert(i+col);
            HSUL.insert(i-col);

            countPossibilities(n, col+1);
```

```
HSR.remove(i);
HSLL.remove(i+col);
HSUL.remove(i-col);
}
```

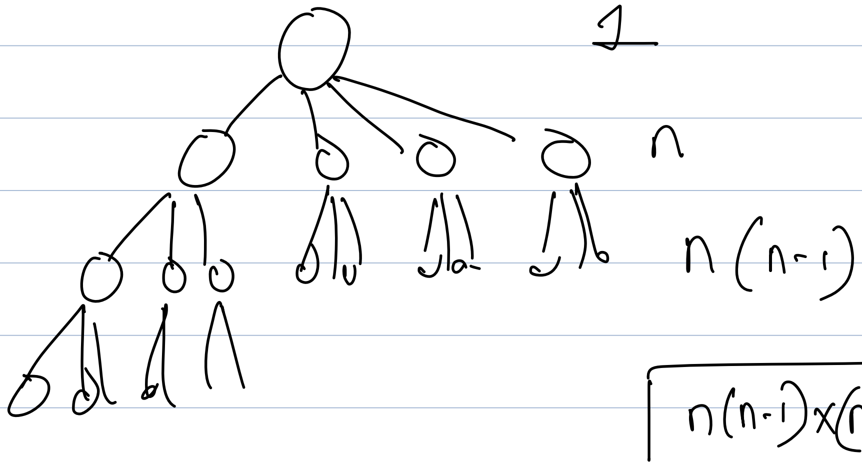


Total methods $\Rightarrow n^n$

Time taken by each method $\Rightarrow O(n)$

$\Rightarrow n^n \times n \Rightarrow \underline{\boxed{n^n}}$

SC $\Rightarrow \underline{O(n)}$



$$\boxed{n(n-1)\times(n-2)}$$

$$1 + n + n(n-1) + n(n-1)(n-2) + \dots + n!$$

$$n!\left\{\frac{1}{n!} + \frac{1}{(n-1)!} + \frac{1}{(n-2)!} \;\dots\dots\; \underline{1}\right)$$

$$\Downarrow$$

$$1 + \frac{1}{2} + \frac{1}{6} + \frac{1}{24} + \frac{1}{120} + \frac{1}{720}$$

⇒ Total function calls ⇒ $\underline{\underline{n!}}$

Q Given a Partially filled sudoko. Fill it Completely.

3	1	6	5	2	8	4	9	7
5	2	4						
	8	7					3	1
		3		1			8	5
9			8	6	3			
	5			5		6		
1	3					2	5	
							7	4
		5	2		6	3		

Note: Every row should have all values from 1 to 9 only once

Every col should have all values from 1 to 9 only once.

Every 3x3 grid should have all values from 1 to 9 only once.

The Solution is unique.

	0	1	2	3	4	5	6	7	8
0	0	1	2	3	4	5	6	7	8
1	9	10	11	12	13	14	15	16	17
2								3	1
3								8	5
4	9								
5		5			49		6		
6	1	3					2	5	
7								7	4
8			5	2		6	3		

(3,3)

index 0 → 80

row → index/9

col → index %9

index = 13

row → 1

col → 4.

index = 45

row = 49/9 = 5

Col = 49%9 → 4

row − (row%3)

5 − (5%3)

5 − 2 → 3

4 − (4%3)

4 − 1 → 3

cube index start position

r → row − (row%3)

1 − (1%3)

1−1 → 0

c → col − (col%3)

4 − (4%3)

4 − 1 → 3

$$\frac{\text{row}}{\cancel{3}} \times \cancel{3} = \text{row}$$

$$\frac{\text{col}}{\cancel{3}} \times \cancel{3} = \text{col}$$

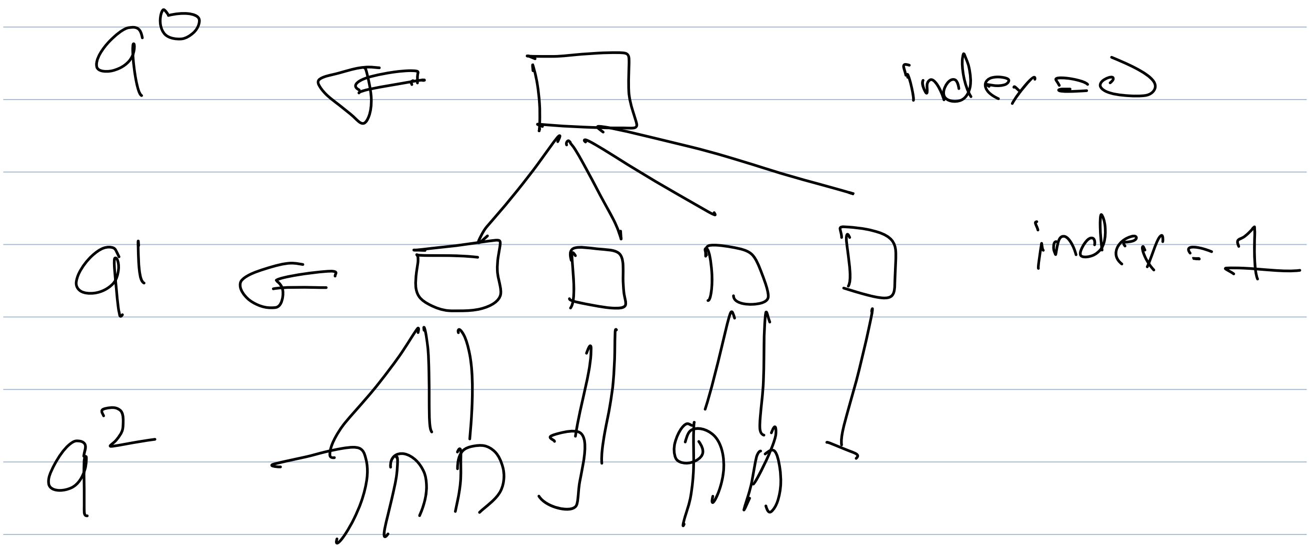
$$7/3 \times 3$$

$$\boxed{2 \times 3 = 6}$$

```
sodoku ( mat[][] , int index) {.
      if (index = 81)
                    ⇒ Sodoku solved.
      int row ⇒ index/9;
      int col ⇒ index %9;
    if ( mat[row][col] != 0) {.
                sudoku (mat, index+1);
   }
                                                  check if
                                                  i can be
   for(int i=1 ; i≤9 ; i++) {.  ⇒                placed at
                                                  (row,col).
        if (checkValidity(row, col, i)) {
                 mat[row][col] = i
                 sudoku (mat, index+1)
          }      mat [row] [col] = 0;
   }
}
```

```
bool checkValidity ( int row, int col, int val){

}
```

$\boxed{TC: O(1)}$



9^3

⋮

9^{80}

$\Rightarrow \boxed{9^{81}}$

$\downarrow$

$9^{\text{empty cells}}$