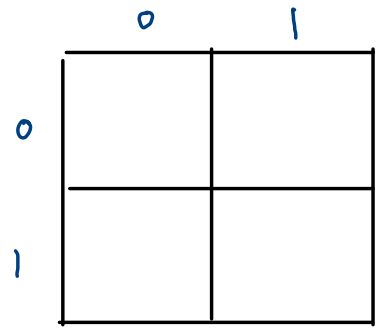
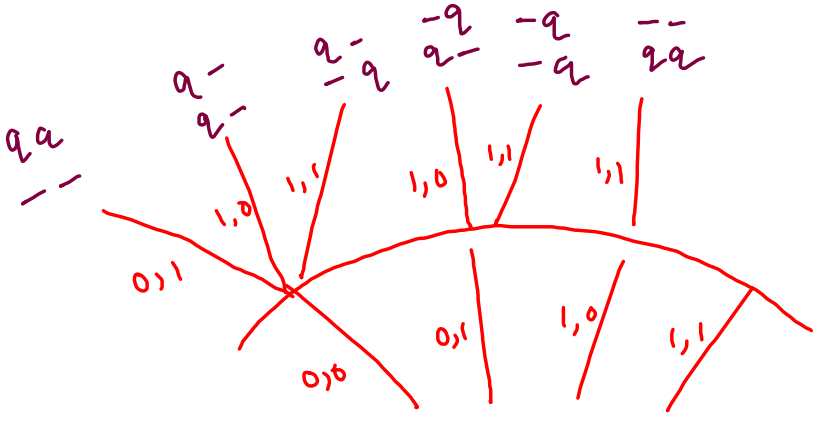


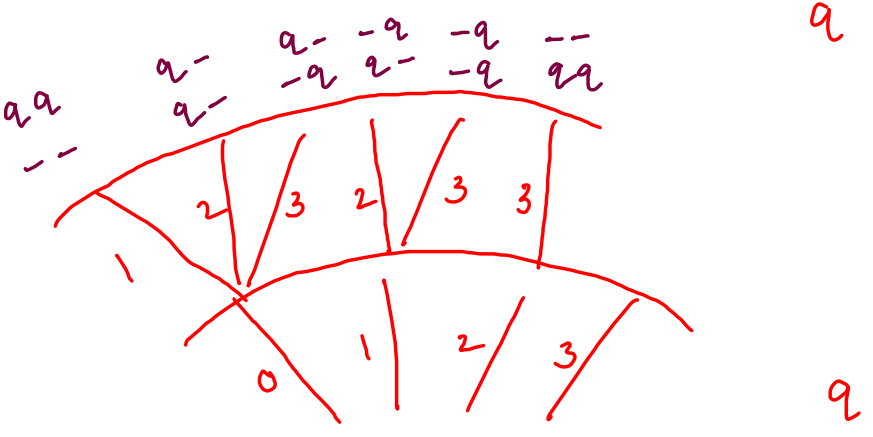
queen combinations

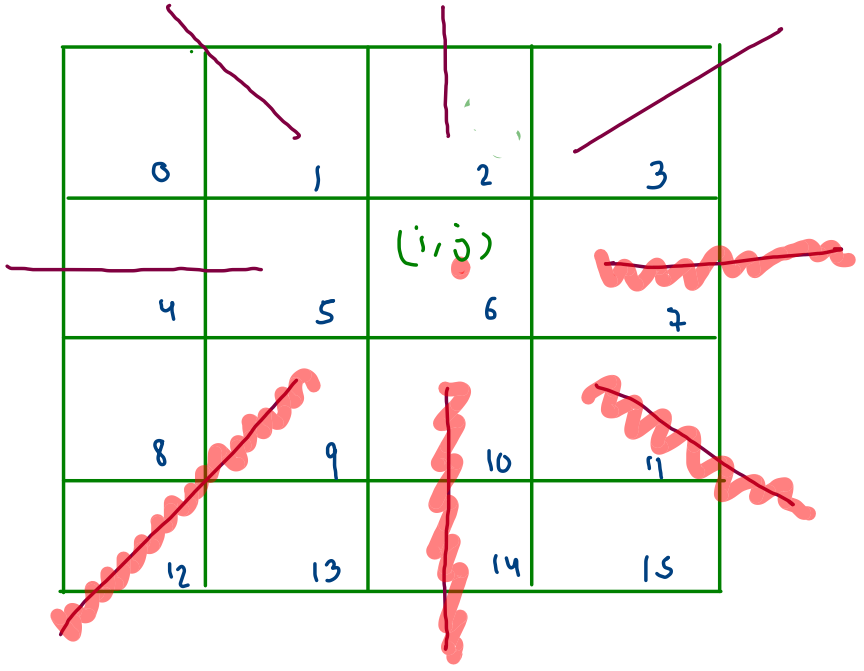
(queen choices)

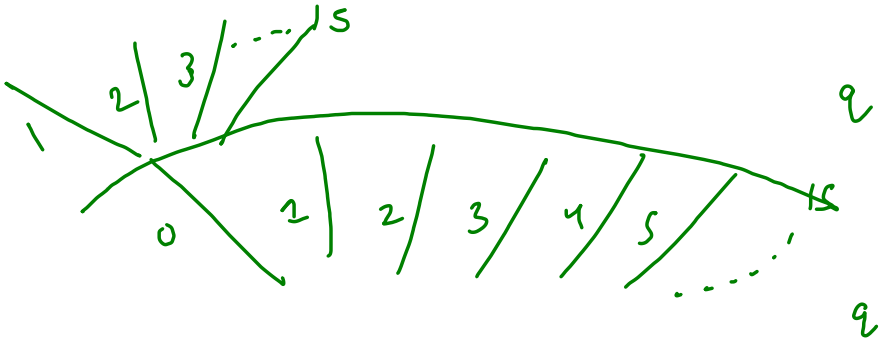
2d as 2d

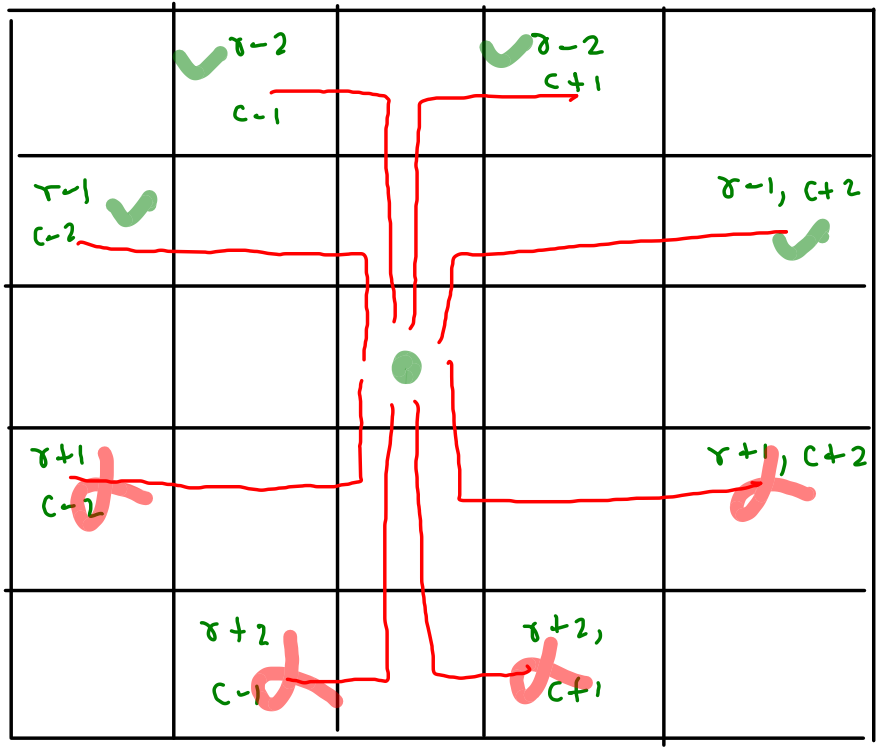


2d as 1d

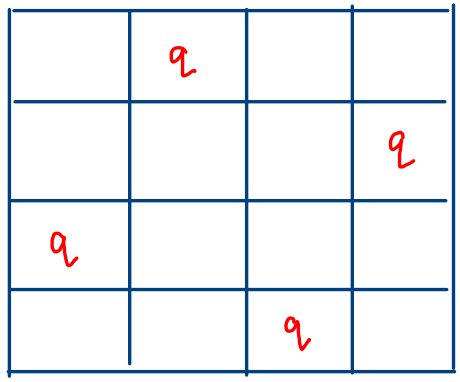



0
1
2
3
(i, j)
4
5
6
7
8
9
10
11
12
13
14
15

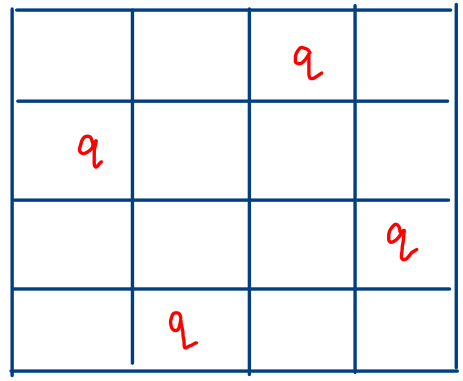

1
2
3
S
0
1
2
2
3
4
5
S


r-2
c-1
r-2
c+1
r-1
c-2
r-1, c+2
r+1
c-2
r+1, c+2
r+2
c-1
r+2,
c+1

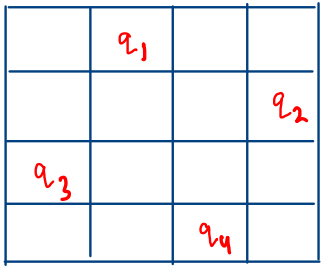
queen permutation

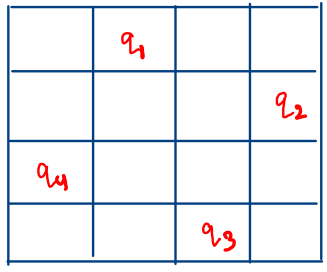


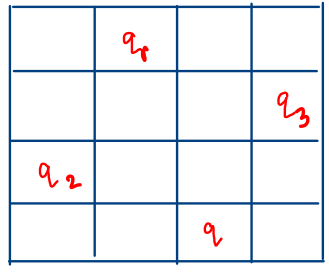
1 24



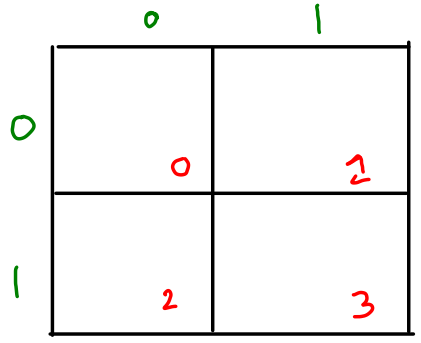
1 24



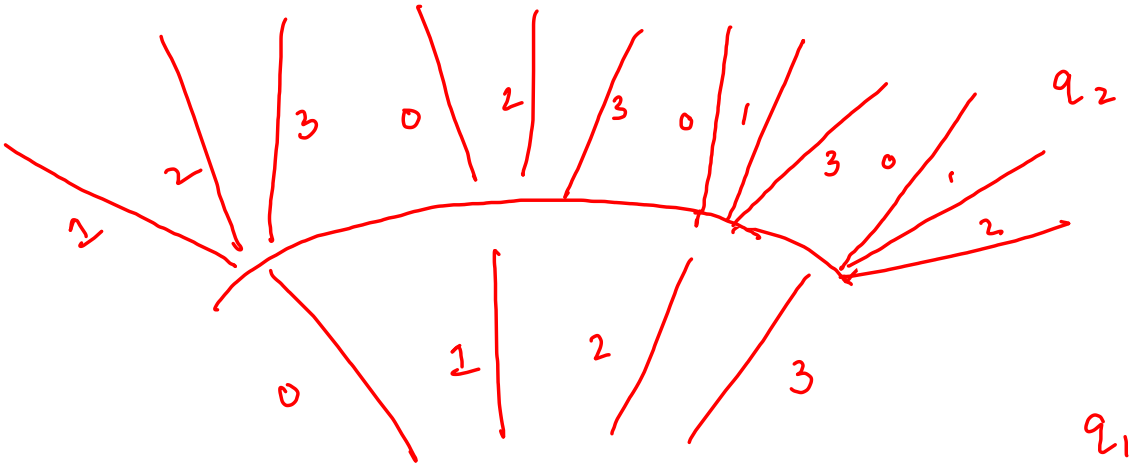




48 permutation



$q_{p_2} = 12$



top : -1, 0

down : 1, 0

left : 0, -1

right : 0, 1

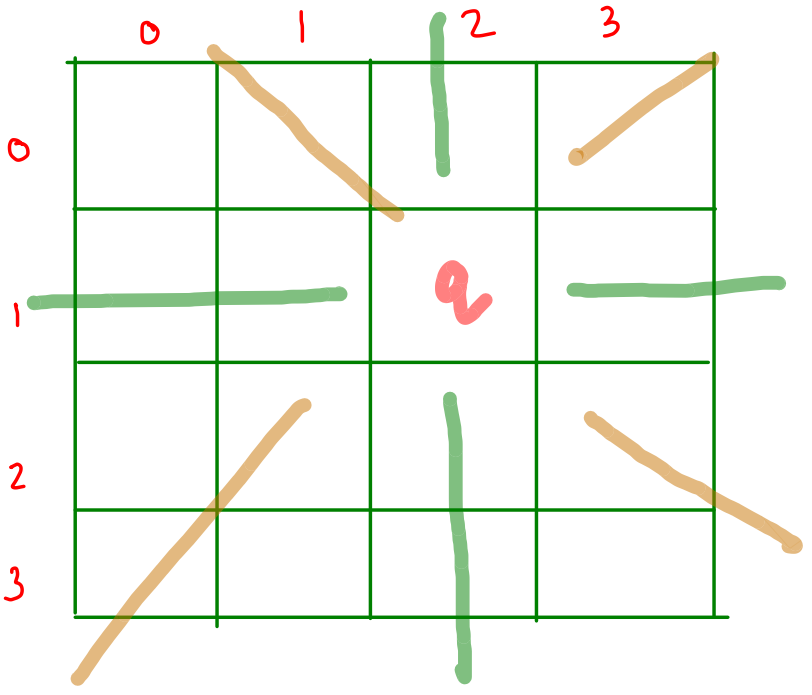
top-left diagonal : -1, -1

bottom-left diagonal : 1, -1

top-right diagonal : -1, 1

bottom-right diagonal : 1, 1

```java
public static void nqueens(int qpsf, int N, int[][] chess) {
    // write your code here
    if(qpsf == N) {
        for(int i=0; i < N;i++) {
            for(int j=0; j < N;j++) {
                if(chess[i][j] == 0) {
                    System.out.print("-\t");
                }
                else {
                    System.out.print("q"  + chess[i][j] + "\t");
                }
            }
            System.out.println();
        }
        System.out.println();
        return;
    }

    for(int b = 0; b < N*N ; b++) {
        int i = b / N;
        int j = b % N;

        if(chess[i][j] == 0 && isQueenSafe(chess,i,j) == true) {
            chess[i][j] = qpsf+1;
            nqueens(qpsf+1,N,chess);
            chess[i][j] = 0;
        }
    }
}
```



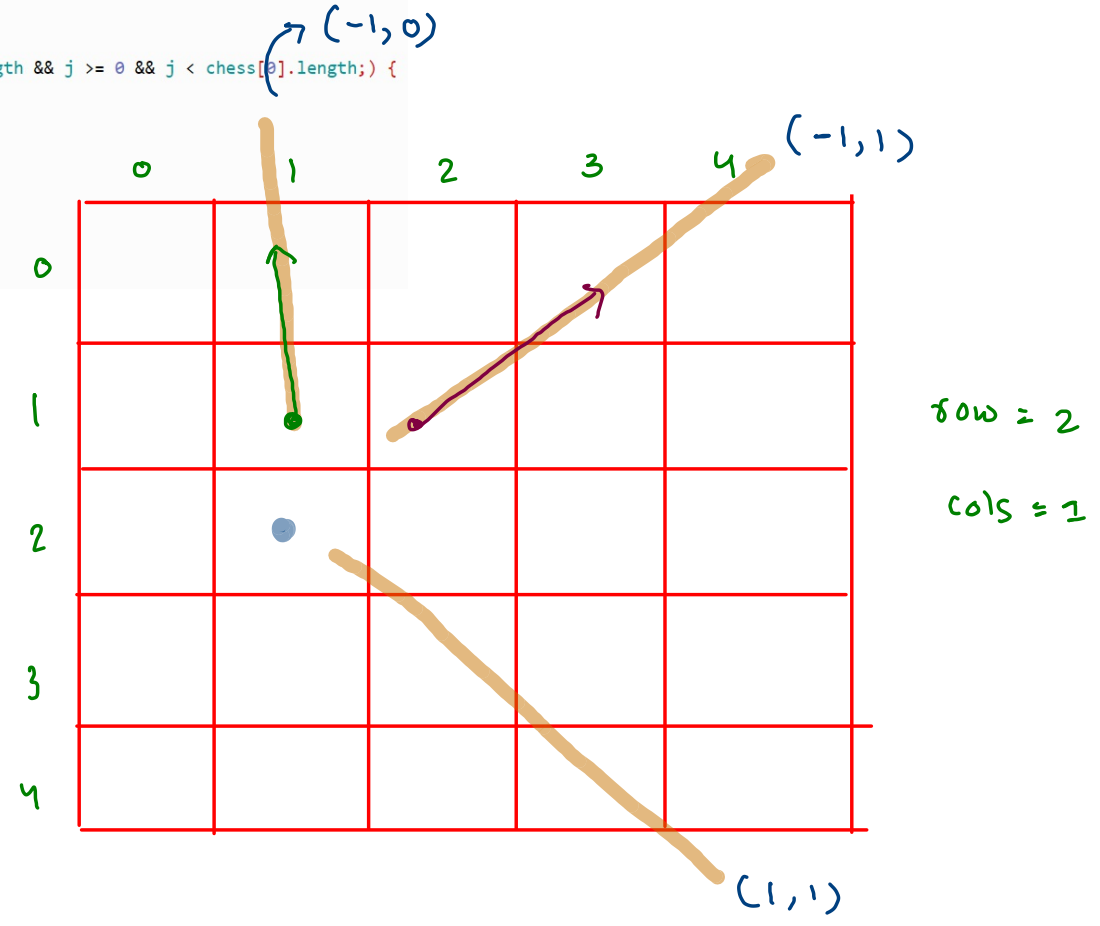
```java
public static boolean IsQueenSafe(int[][] chess, int row, int col) {
    int[][]dir = {{-1,0},{1,0},{0,-1},{0,1},{-1,-1},{1,-1},{-1,1},{1,1}};

    for(int k=0; k < 8;k++) {
        for(int i = row + dir[k][0], j = col + dir[k][1]; i >= 0 && i < chess.length && j >= 0 && j < chess[0].length;) {
            if(chess[i][j] == true) {
                return false;
            }
            i += dir[k][0];
            j += dir[k][1];
        }
    }

    return true;
}
```

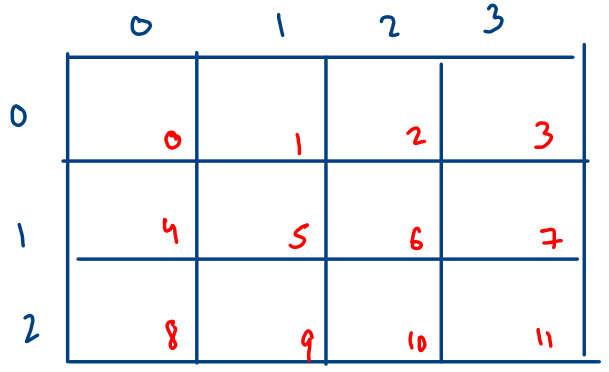
dir[k][0] = -1

dir[k][1] = 1

i = -1

j = 4





bno to i, j

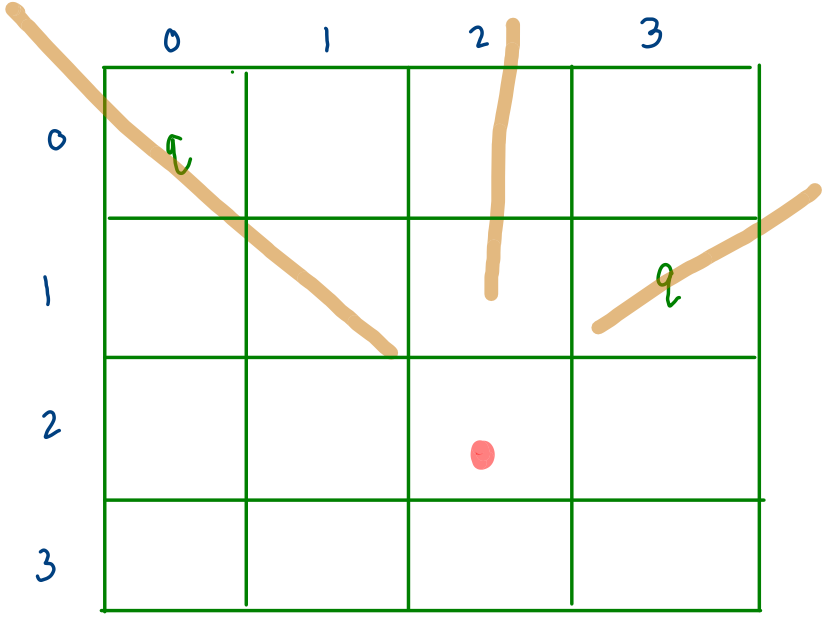
i = bno / cols;

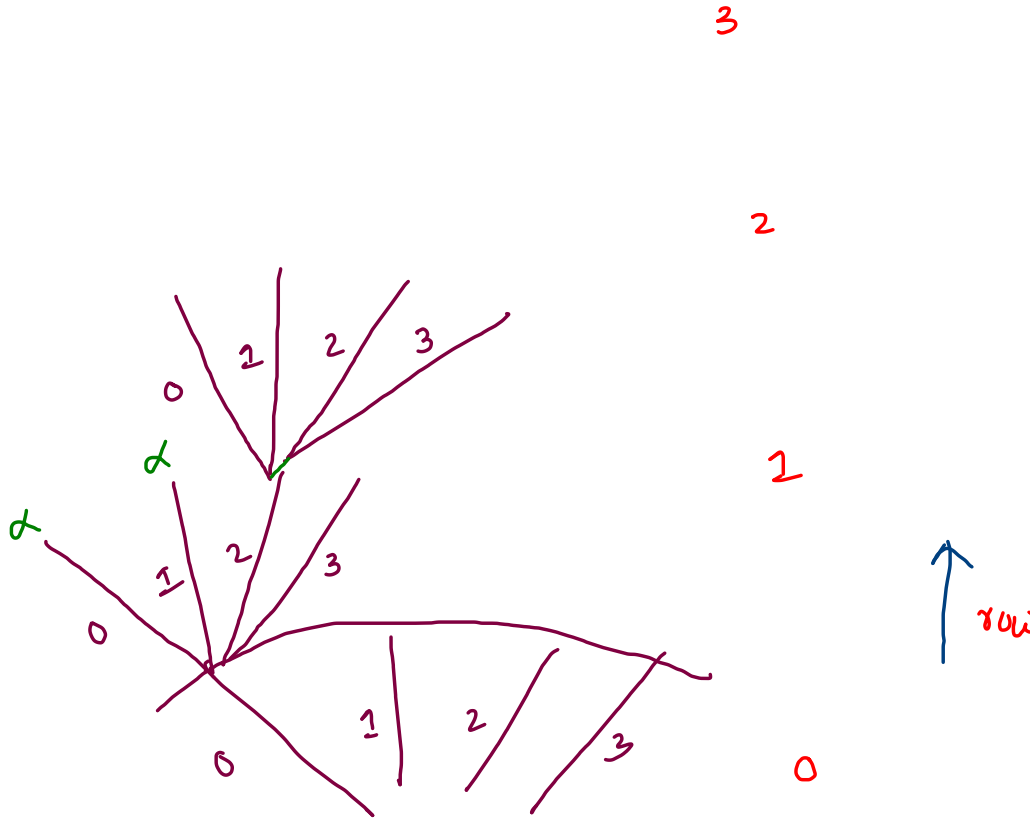
j = bno % cols;

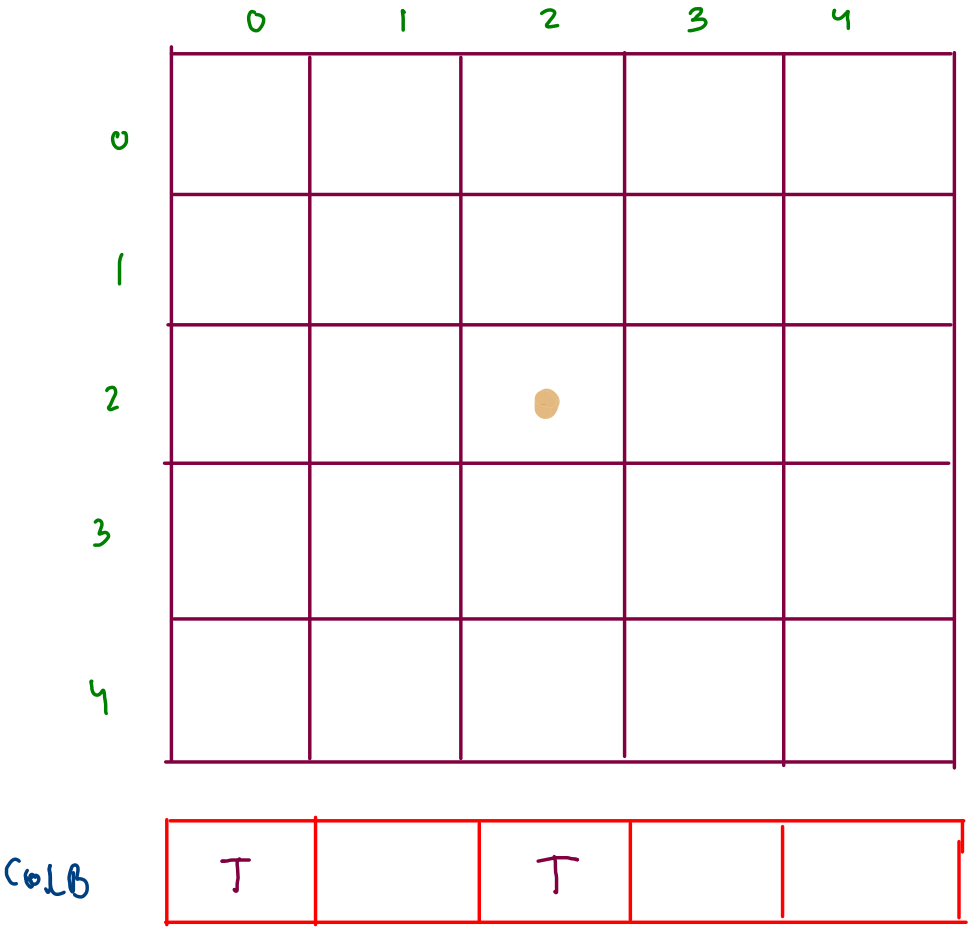
i, j to bno.

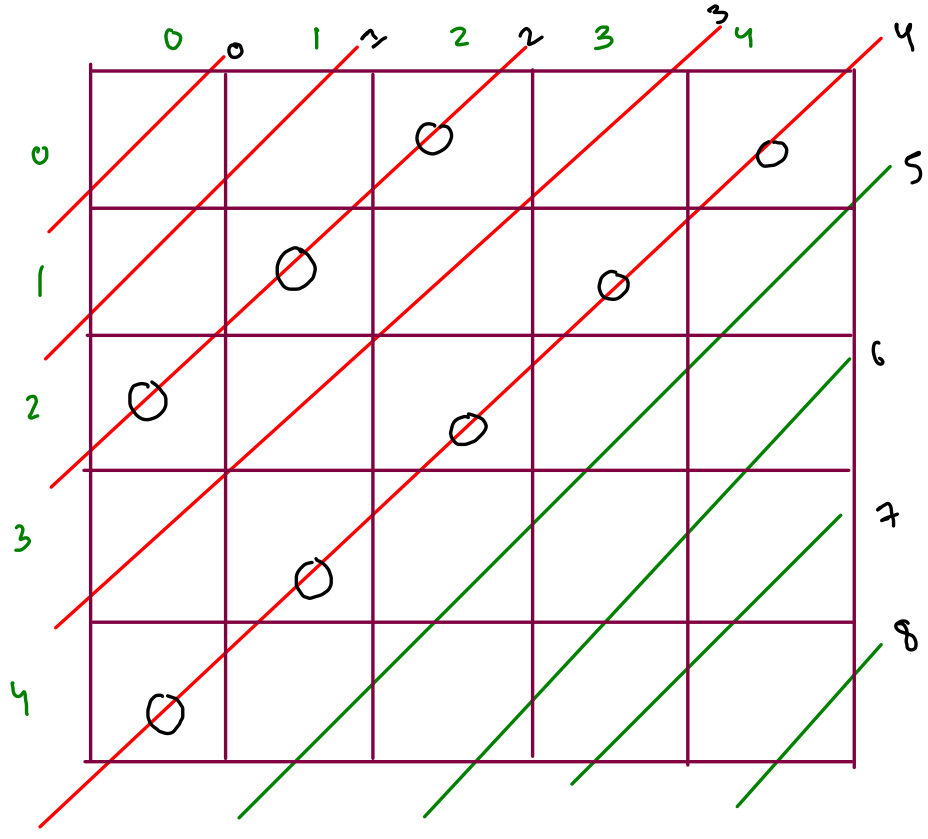
bno = i * cols + j

N- queens branch and bound



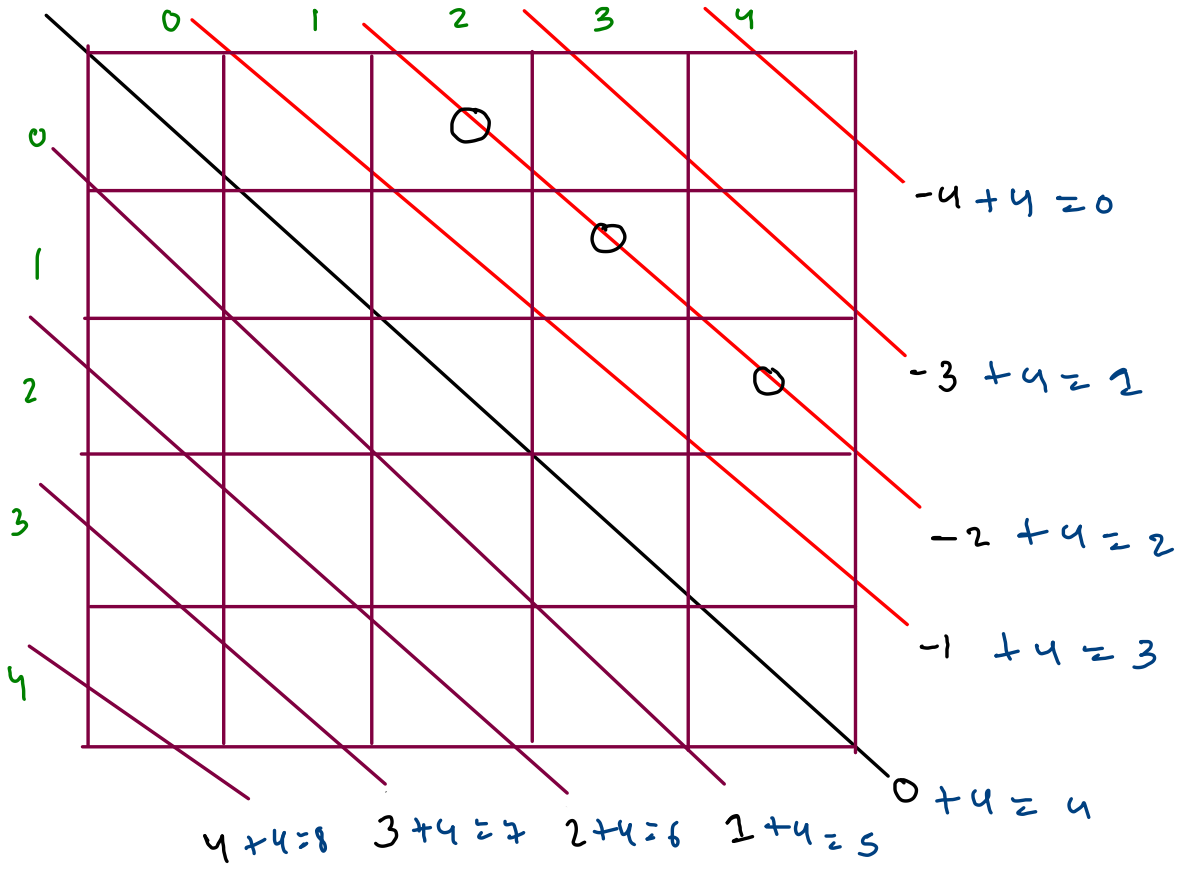



0 1 2 3 4
0
1
2
3
4
ColB
T
T



i + j = reg. - diagonal no.

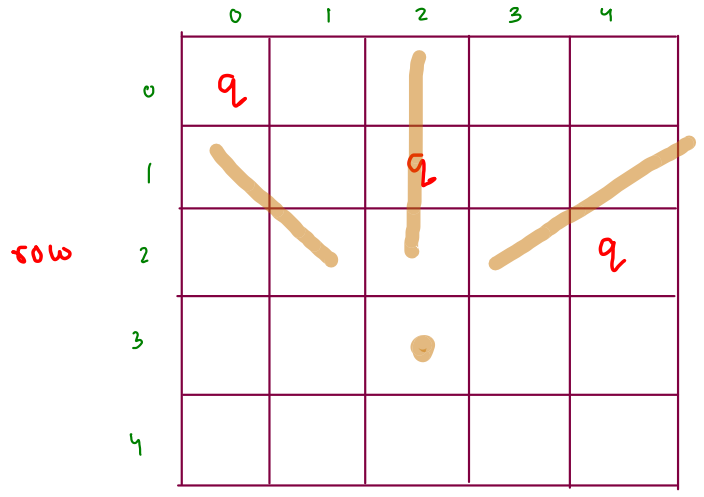
2 * N - 1



rev_diagonal

rev_diag = (i - j + N - 1)

shift



(i, j)

colB [j]

regdB [i+j]

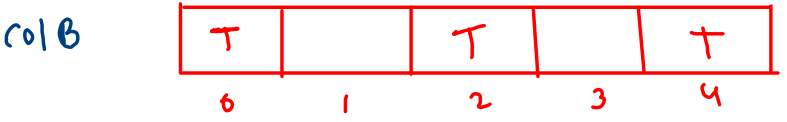
rwdB [i-j+N-1]

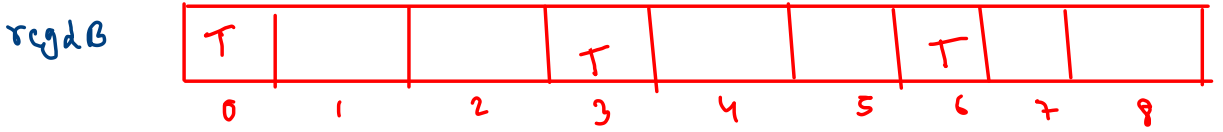
(2,4)

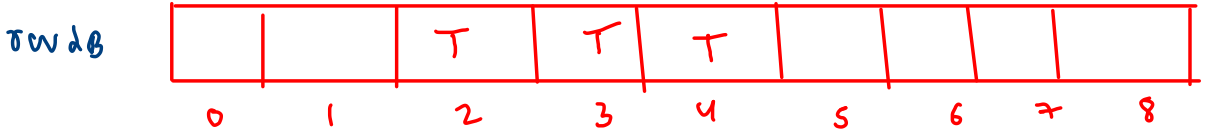
col =

regDb =

rwDb =



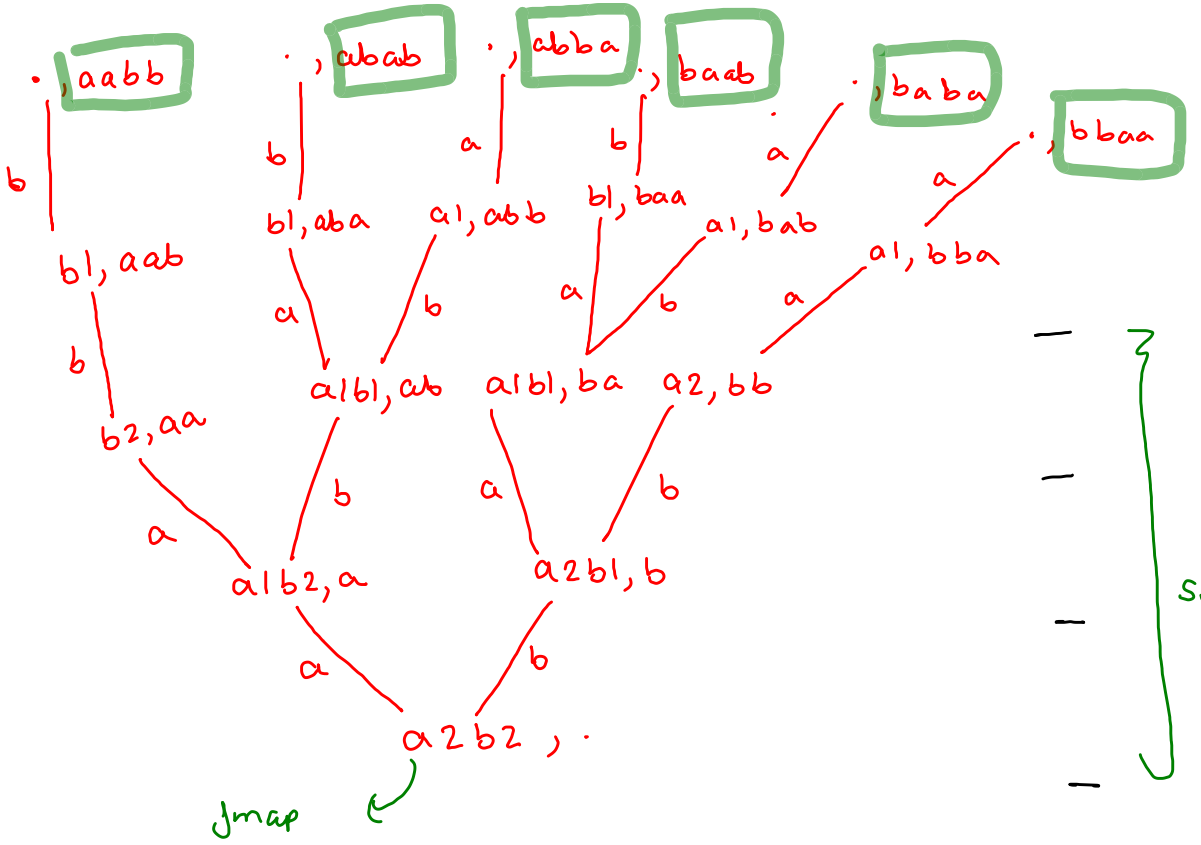




Permutations words - 1

$$\frac{4!}{2! \times 2!} = \frac{24}{4} = 6$$

aabb



```java
public static void generateWords(int cs, int ts, HashMap<Character, Integer> fmap, String asf) {
    if(cs > ts) {
        System.out.println(asf);
        return;
    }

    for(char uch : fmap.keySet()) {
        if(fmap.get(uch) > 0) {
            fmap.put(uch,fmap.get(uch)-1);
            generateWords(cs+1,ts,fmap,asf + uch);
            fmap.put(uch,fmap.get(uch)+1);
        }
    }
}
```

abab

a - 2

b - 2