Queens Combinations - 2D as 2D - Box Chooses

Thursday, 15 July 2021 7:14 PM

nxn → board, n - identical items } queen.

print all possible ways to place queens in chess board.

"behaviour of chess queen is not managed here"

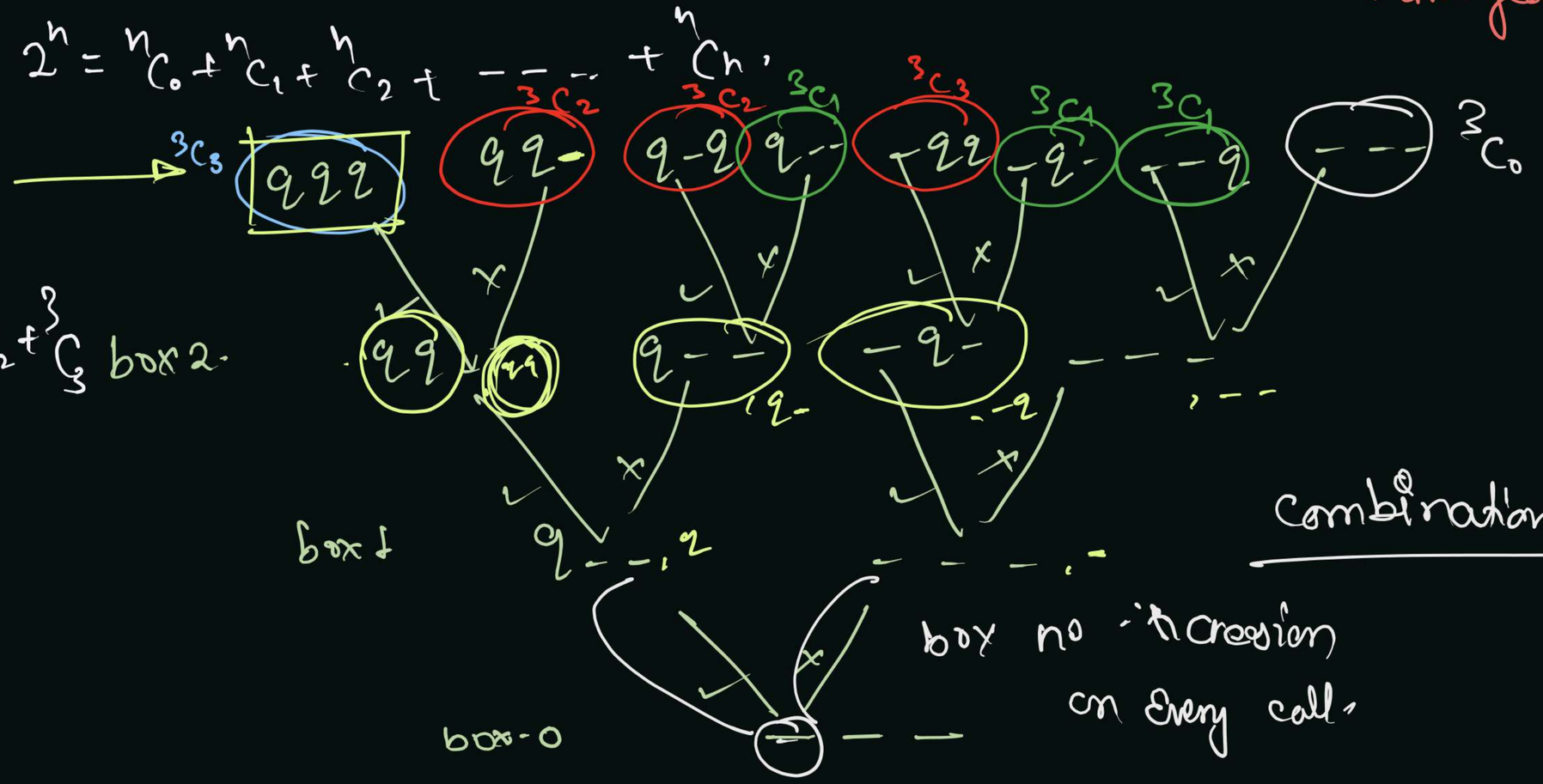
level - Box

option - Yes and No of queen.

$2^n = {}^nC_0 + {}^nC_1 + {}^nC_2 + \cdots + {}^nC_n$

n=3

r=2

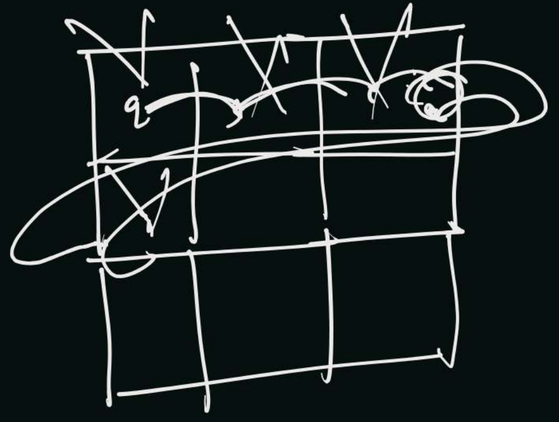


$2^3 = {}^3C_0 + {}^3C_1 + {}^3C_2 + {}^3C_3$ box 2.

qq

qq

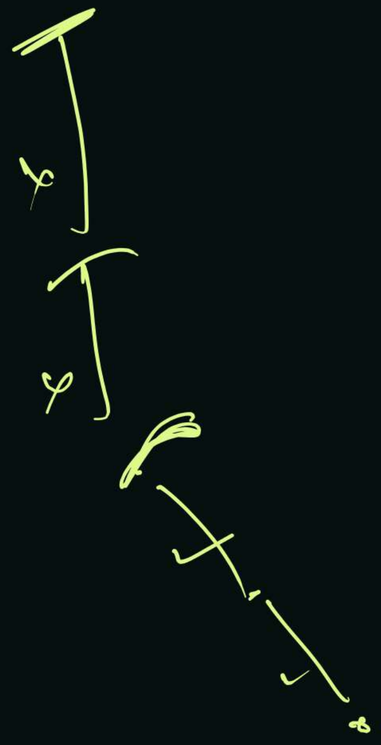
Combination:

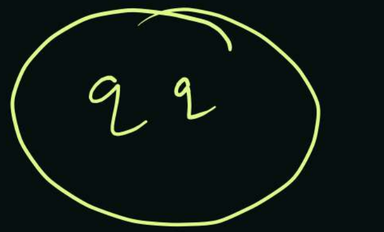


if col cross its range:

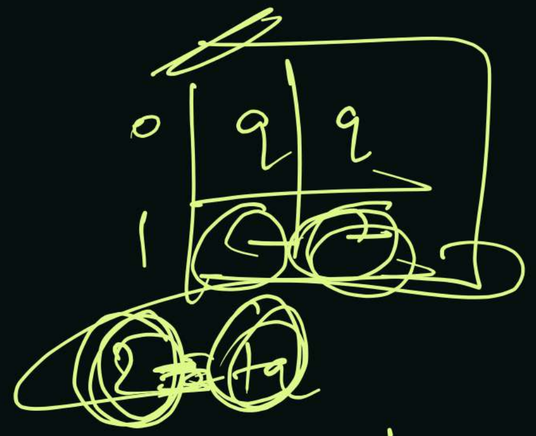
↳ row - increment

↳ col. begin with '0'

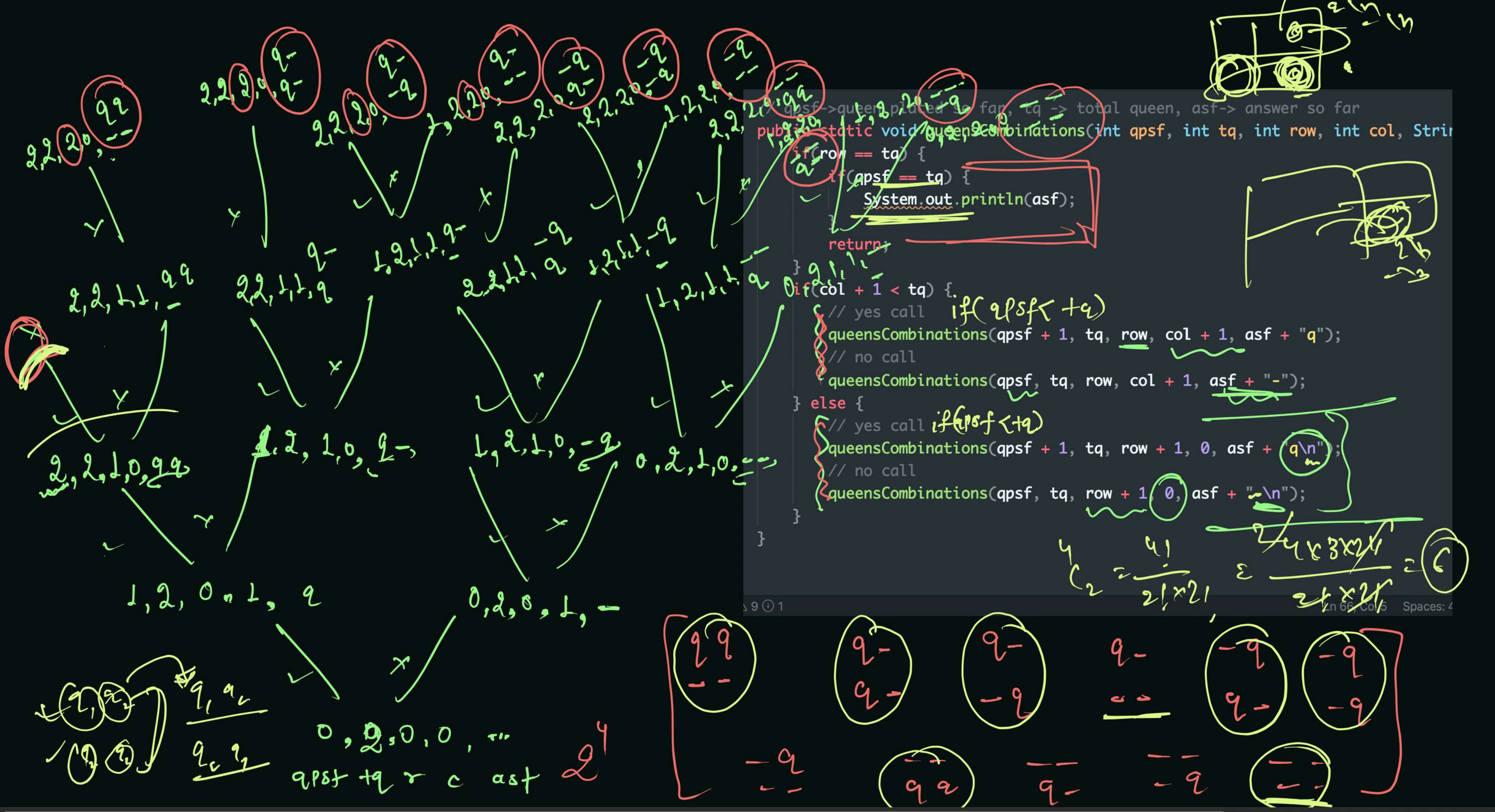




(row == tq)



row == tq


// qpsf->queen placed so far, tq -> total queen, asf-> answer so far
public static void queensCombinations(int qpsf, int tq, int row, int col, Strin
if(row == tq) {
if(qpsf == tq) {
System.out.println(asf);
}
return;
}
if(col + 1 < tq) {
// yes call
queensCombinations(qpsf + 1, tq, row, col + 1, asf + "q");
// no call
queensCombinations(qpsf, tq, row, col + 1, asf + "-");
} else {
// yes call
queensCombinations(qpsf + 1, tq, row + 1, 0, asf + "q\n");
// no call
queensCombinations(qpsf, tq, row + 1, 0, asf + "-\n");
}
}

Queens Permutations- 2D as 2D - Queen Chooses

Thursday, 15 July 2021 7:14 PM

nxn chessboard, n- non identical queens.
print all possible arrangement

level = queen / items

option → box

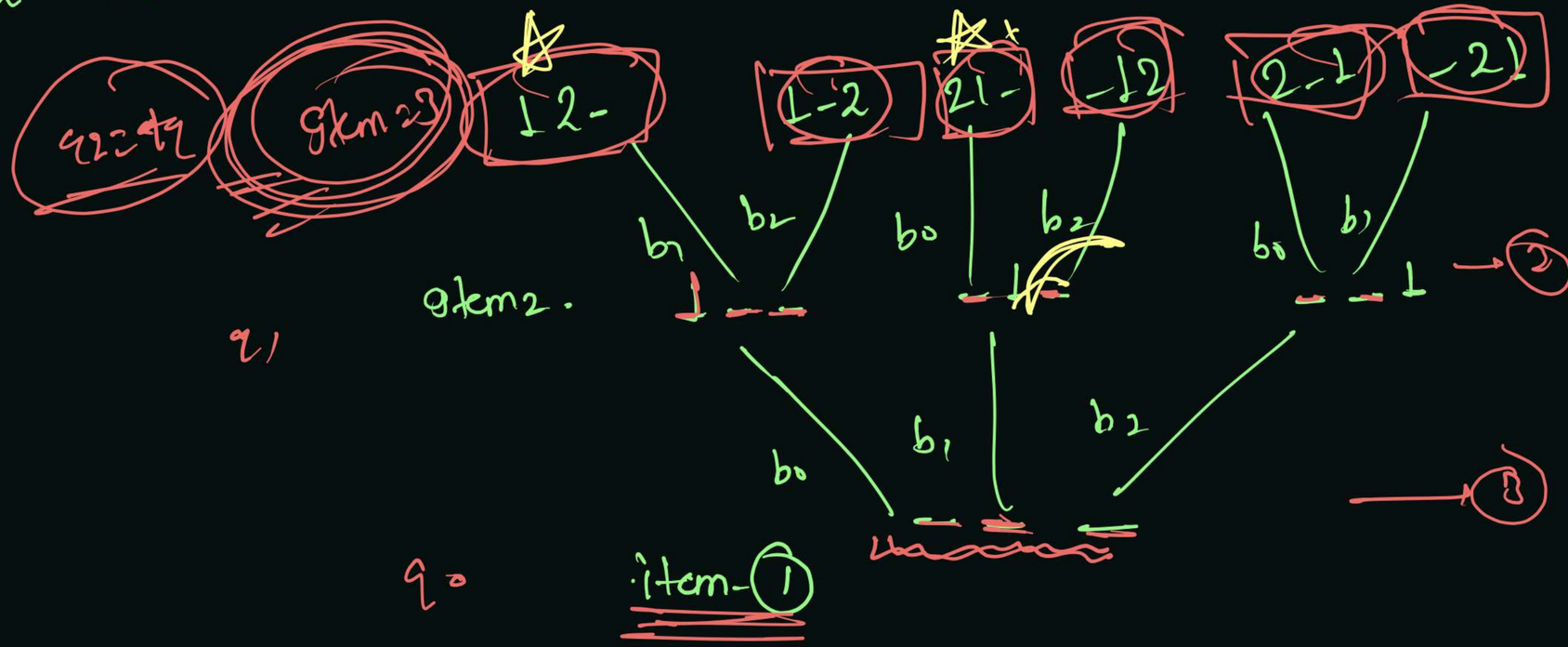
n = 3 → boxes

r = 2 → items

$n \times (n-1) \times (n-2) \times \cdots \times (n-r+1)$

↳ $\frac{n!}{(n-r)!} = {}^nP_r$.

${}^3P_2 = \frac{3!}{1!} = \frac{3 \times 2 \times 1}{1} = 6$

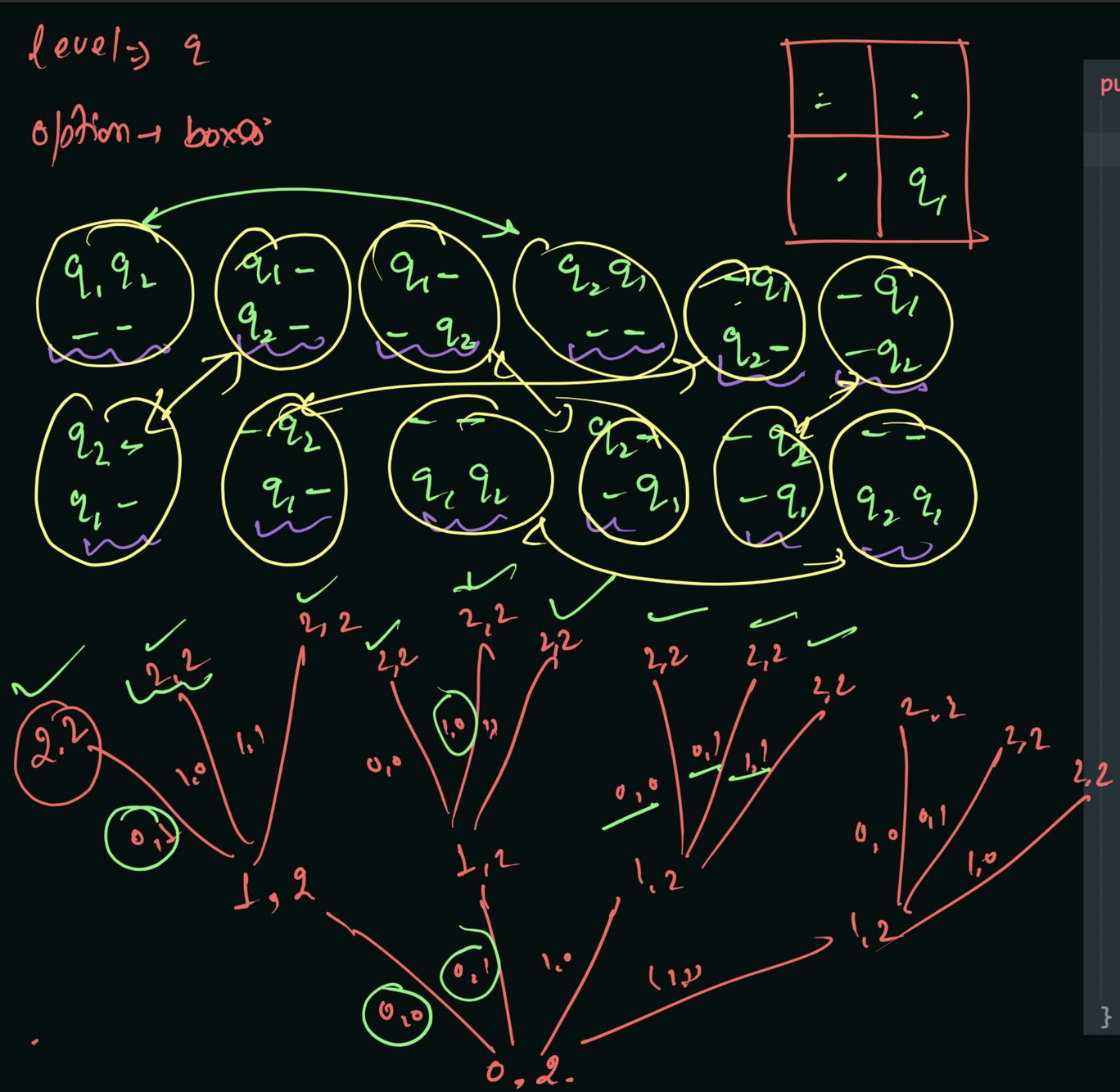


↳ 12-
- 1-2
- 21-
- -12
- 2-1
- -21

if(qpsf == tq)

print tqpsf

}




```java
public static void queensPermutations(int qpsf, int tq, int[][] chess){
    if(qpsf == tq) {
        // print result
        for(int i = 0; i < chess.length; i++) {
            for(int j = 0; j < chess[0].length; j++) {
                if(chess[i][j] != 0) {
                    System.out.print("q" + chess[i][j] + "\t");
                } else {
                    System.out.print("-\t");
                }
            }
            System.out.println();
        }
        System.out.println();
        return;
    }

    for(int i = 0; i < chess.length; i++) {
        for(int j = 0; j < chess[0].length; j++) {
            if(chess[i][j] == 0) {
                // place queen
                chess[i][j] = qpsf + 1;
                queensPermutations(qpsf + 1, tq, chess);
                // unplace queen
                chess[i][j] = 0;
            }
        }
    }
}
```

Queens Permutations- 2D as 2D - Box Chooses

Thursday, 15 July 2021 7:15 PM

n x n - chessboard, n-queens, (non identical) print all possible ways to arrange these queens.

level → Boxes

option → queen.

$${}^nC_r = \frac{n!}{(n-r)!r!}$$

$${}^nP_r = \frac{n!}{(n-r)!}$$

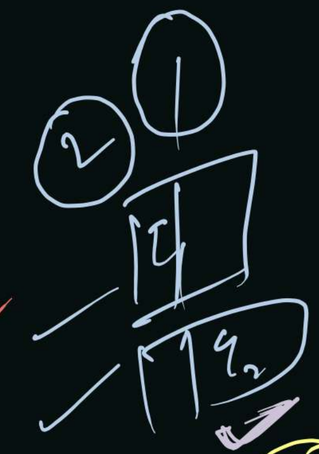
$${}^nP_r = {}^nC_r \times r!$$

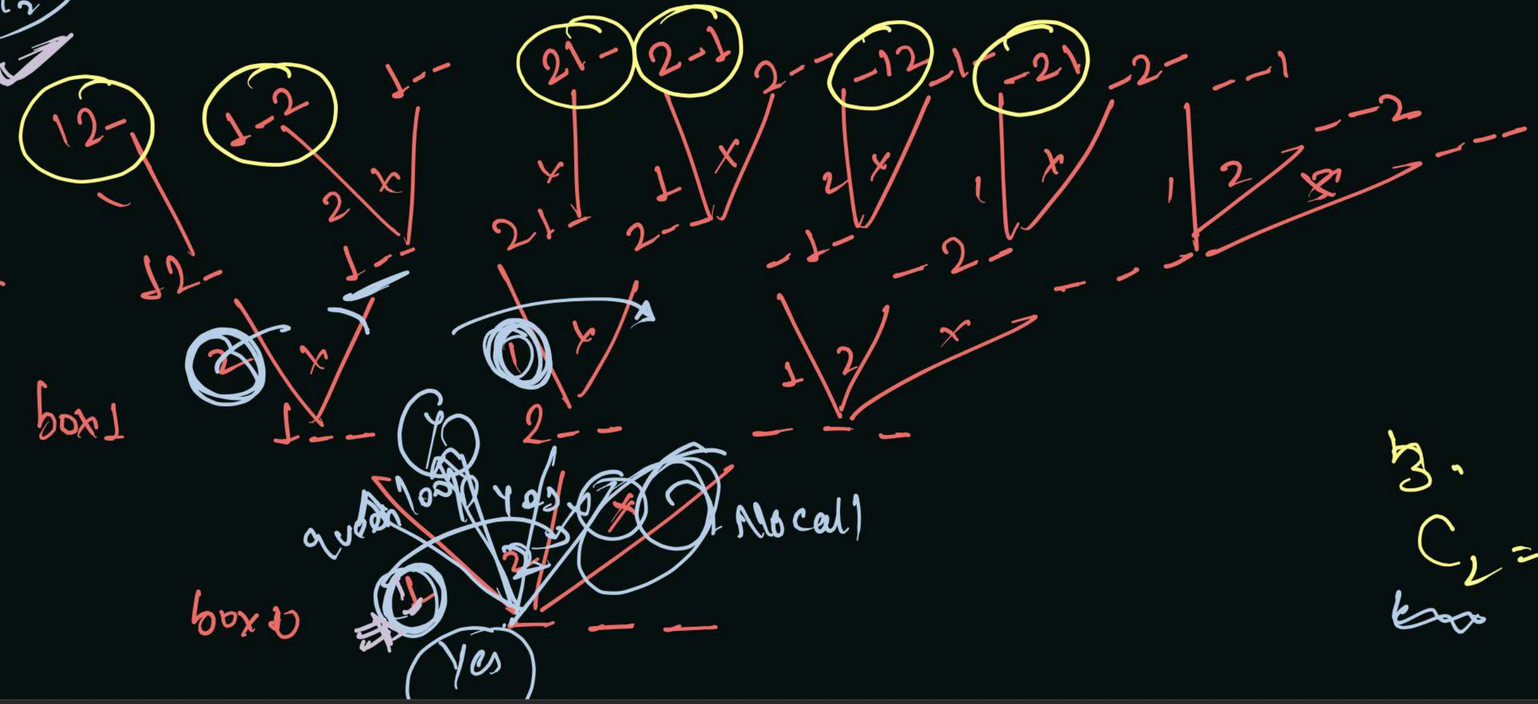
Yes or No

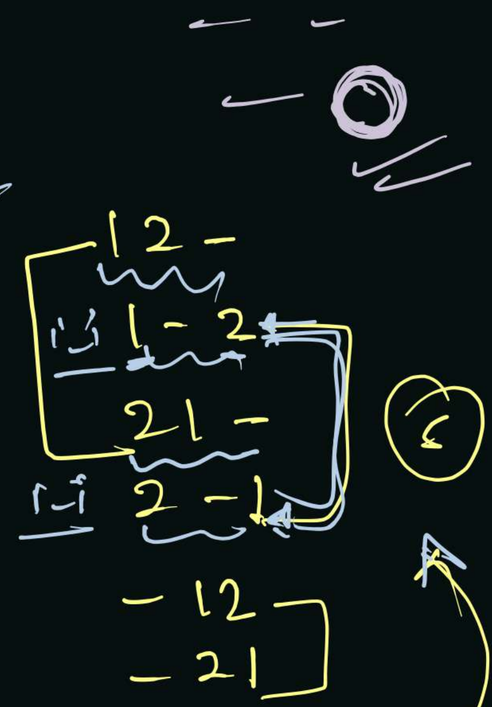
n=3

r=2

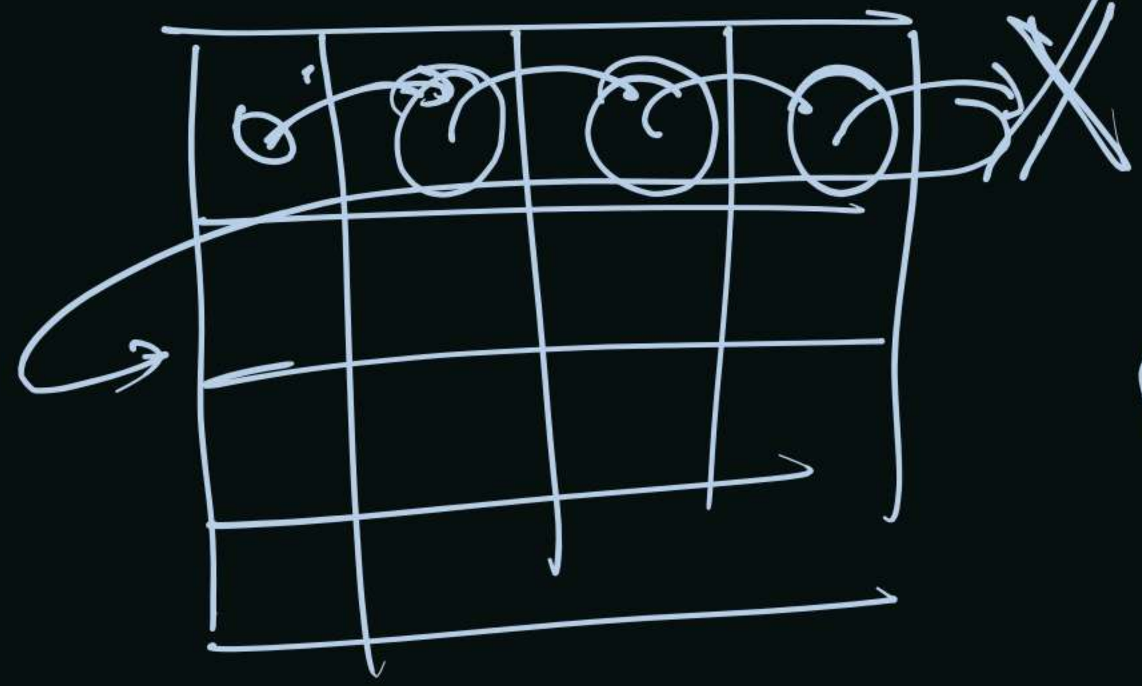
${}^3P_2$







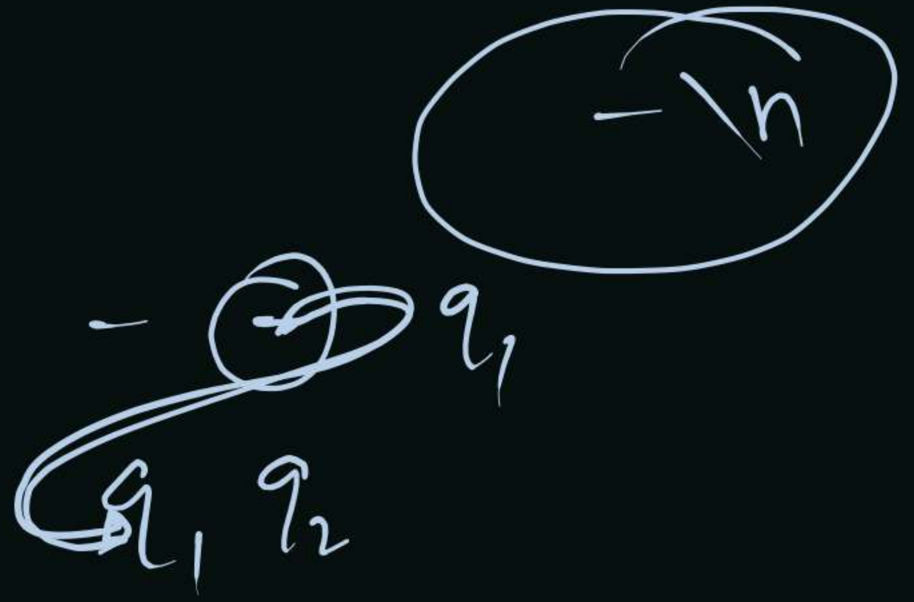
3. ${}^3C_2 = \frac{3!}{1!2!} =$ 3 ways



index out bn

col+1 — out of bn

row+1, col=0



String, row, col, queen array;

asf

row, col → box for current level

queen array → to figure out which item/queen was already placed in past

Queens Combinations - 2D as 2D - Queens Chooses

Thursday, 15 July 2021 7:15 PM

n x n - chessboard, n - queens (identical)

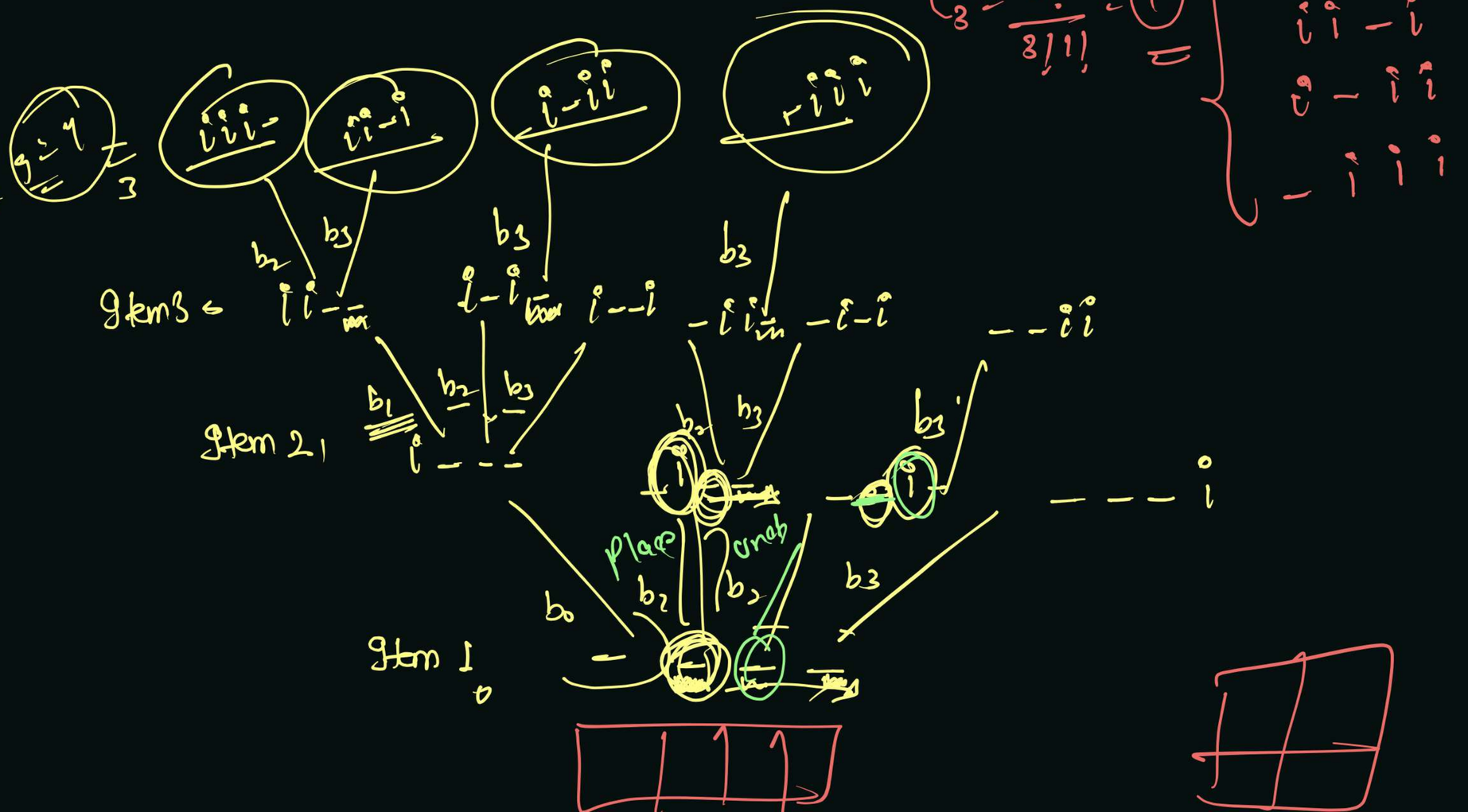
print all possible arrangement

level → queens

option → boxes

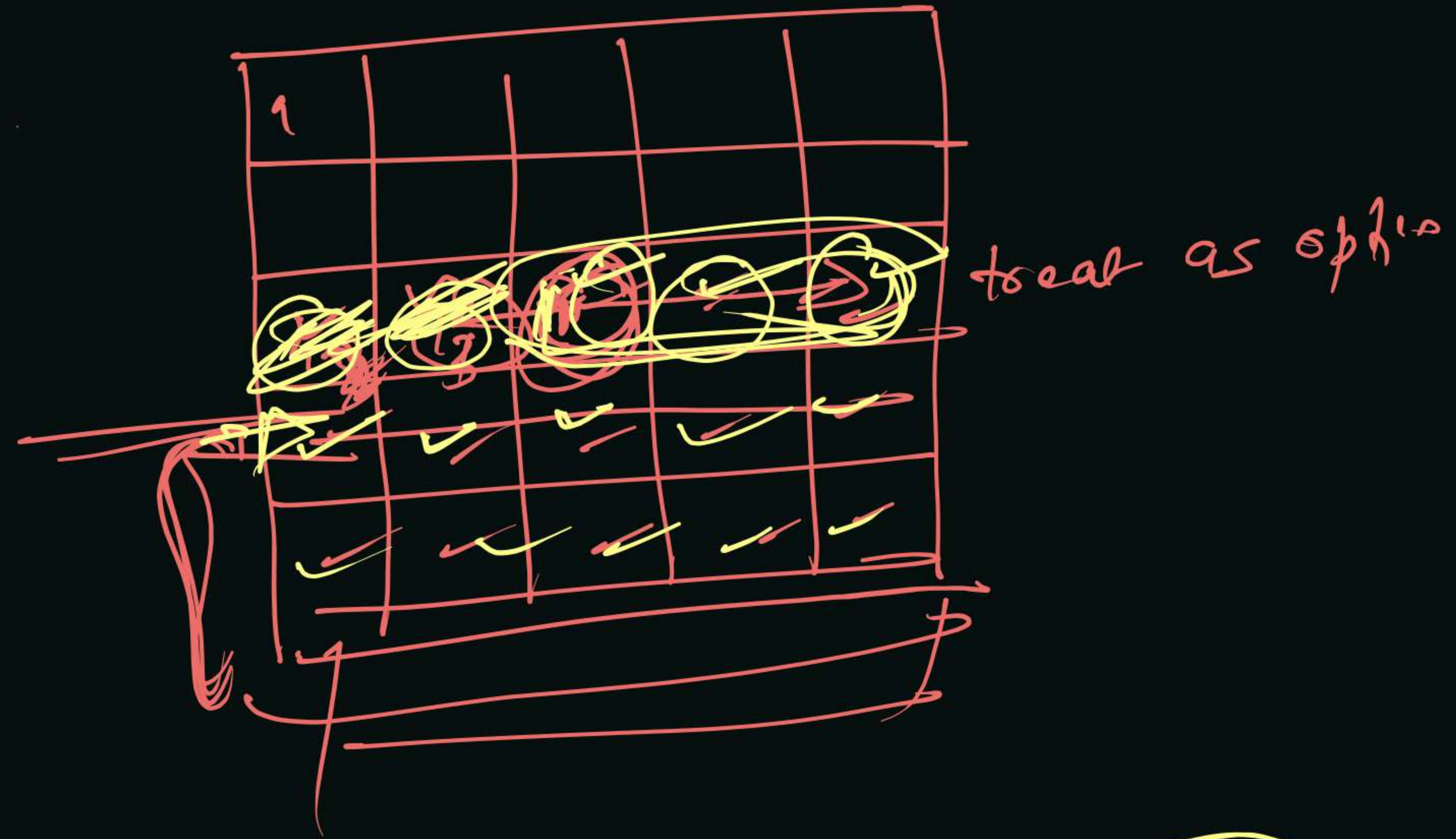
4 → boxes

3 - items



$^4C_3 = \frac{4!}{3!1!} = 4$

- i i i -
- i i - i
- i - i i
- - i i i

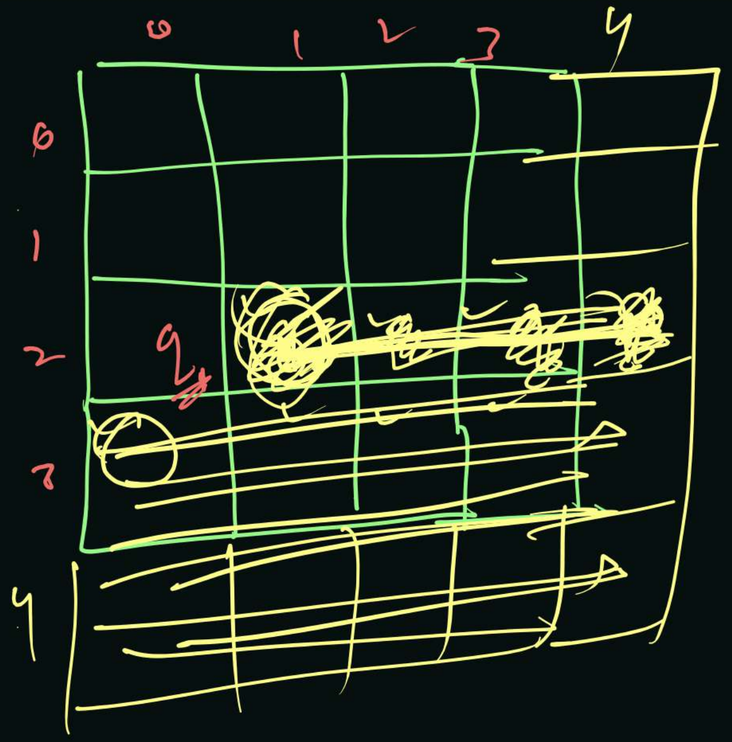


c9 , r9 row, col , String

Booleans

in a current row remaining col.

remaing row X colum



```
        // travel in remaining columns in current row
        for(int c = j + 1; c < chess[0].length; c++) {
            int r = i;
            // place
            chess[r][c] = true;
            queensCombinations(qpsf + 1, tq, chess, r, c);
            // unplace
            chess[r][c] = false;
        }

        // travel in remaining rows and columns
        for(int r = i + 1; r < chess.length; r++) {
            for(int c = 0; c < chess[0].length; c++) {
                // place
                chess[r][c] = true;
                queensCombinations(qpsf + 1, tq, chess, r, c);
                // unplace
                chess[r][c] = false;
            }
        }
```