Queen combinations - 2D as 2D - Box chooses:

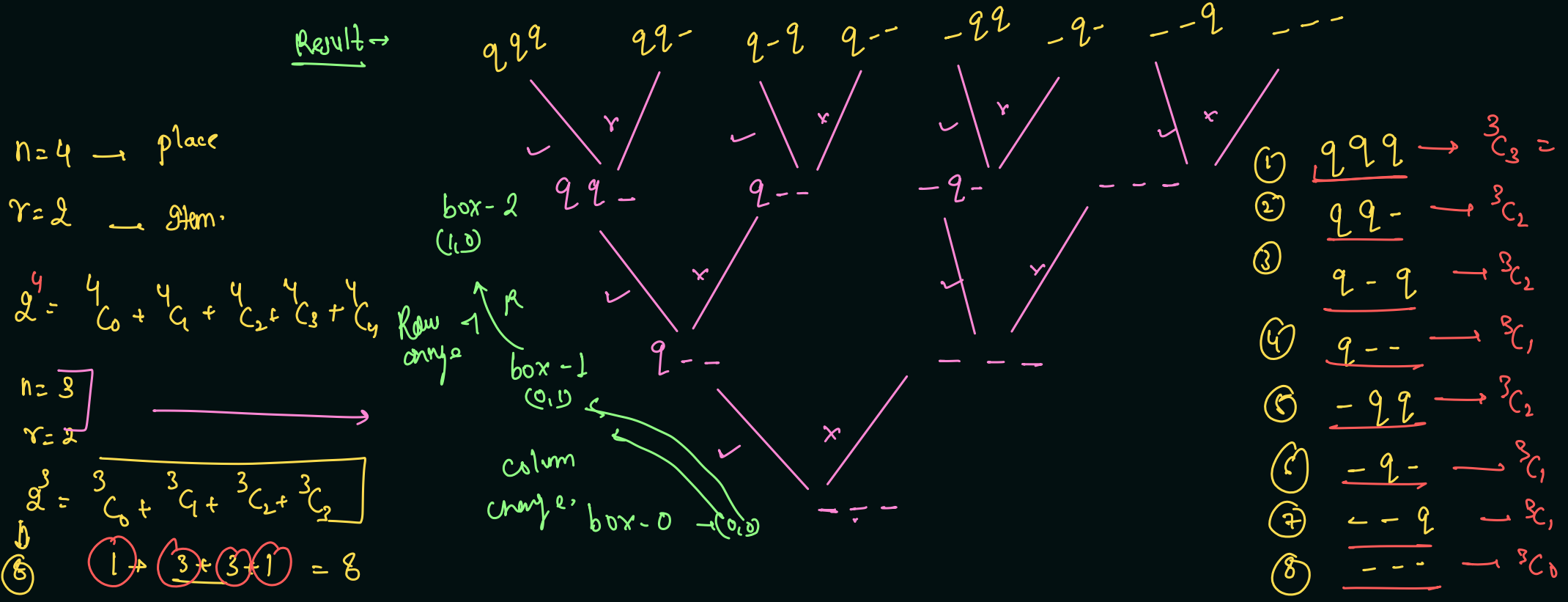
In nxn board, n - identical queens [behaviour is not as chess queen], print all possible ways to place queens in chessboard.

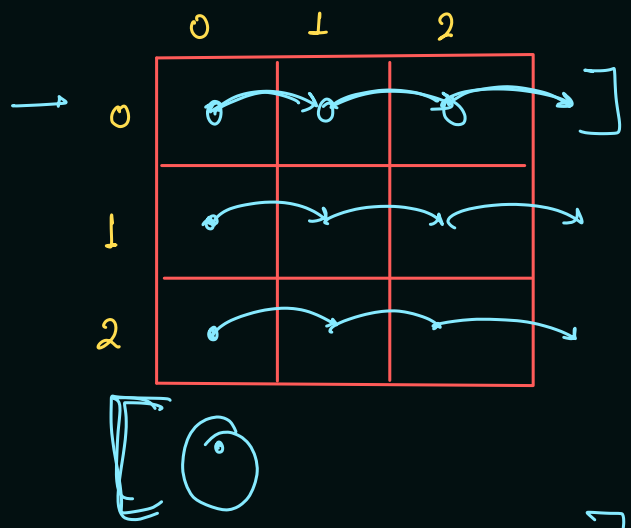
level - Box

options - choice of queens.

$q^n = {}^nC_0 + {}^nC_1 + {}^nC_2 + \dots + {}^nC_n$

q^n ← Idea from Subseq.





column == total column

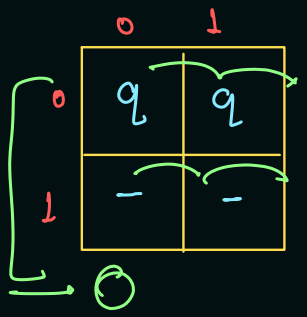
↳ Increment in Row & column begin from zero

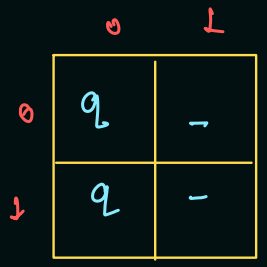
Chess board, 2×2 = 4 → box.

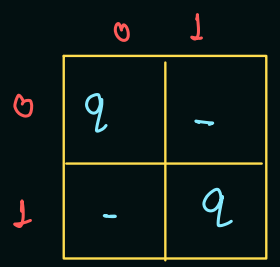
no. of items/queen = 2 → need to place.

Identical item = $^4C_2 = \frac{4!}{2!2!} = \frac{4\times3\times2!}{2!\times2!} = 6$

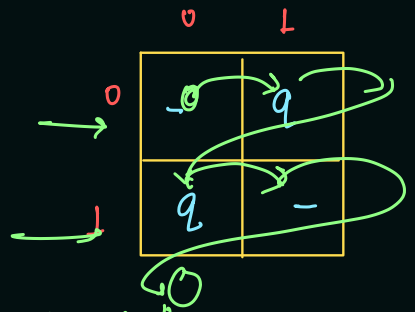
Row == total Row

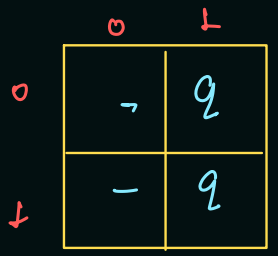


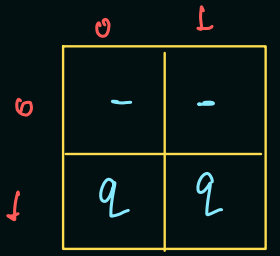




'\n' → newLine..







String → str = "Shreesh\n tripathi";

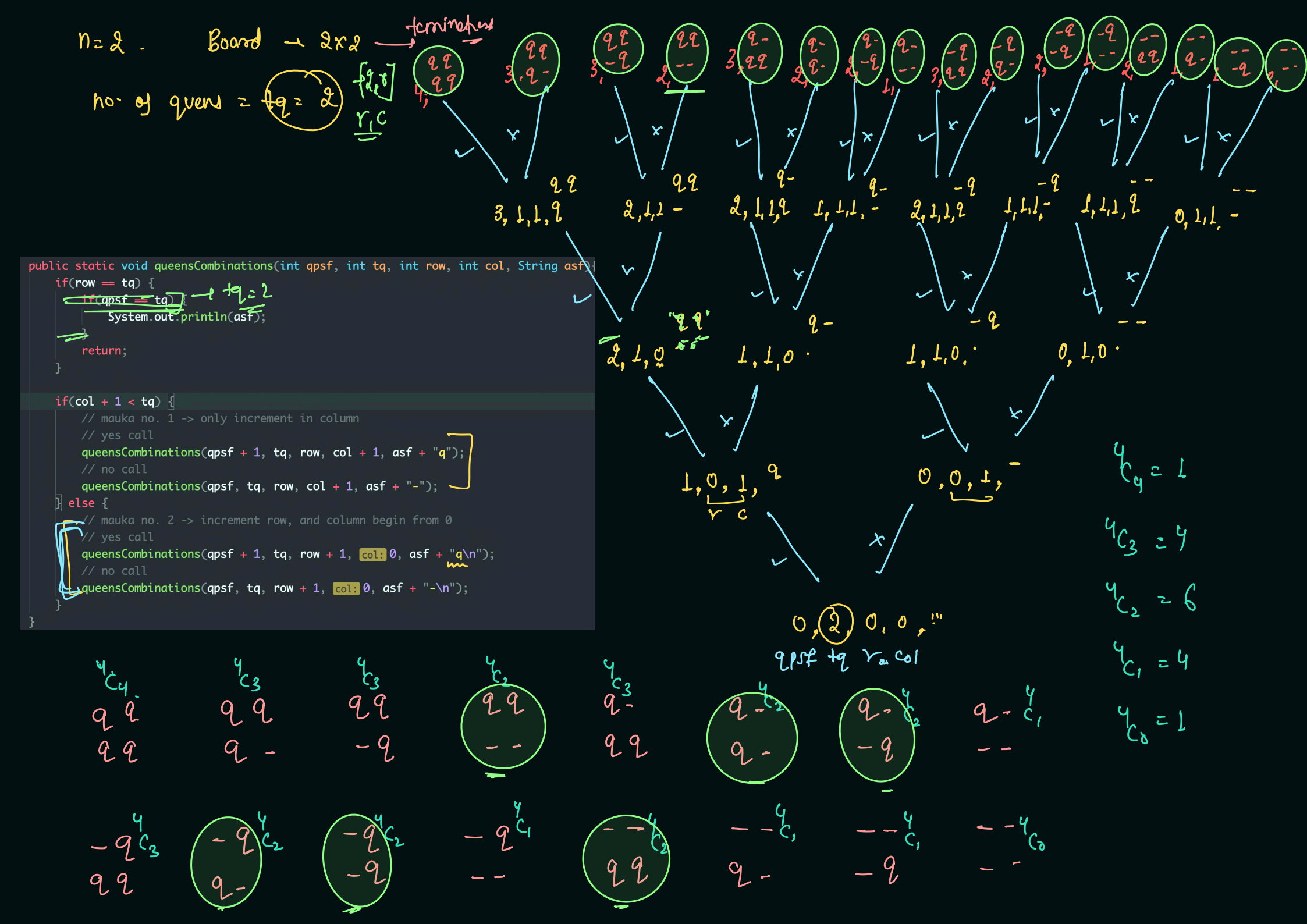
⇒ q q\n
- -

Syso(str); → Shreesh
tripath.

"q-\n
q-"

q-\n
-q

-q\n
q-




```java
public static void queensCombinations(int qpsf, int tq, int row, int col, String asf){
    if(row == tq) {
        if(qpsf == tq) {
            System.out.println(asf);
        }
        return;
    }

    if(col + 1 < tq) {
        // mauka no. 1 -> only increment in column
        // yes call
        queensCombinations(qpsf + 1, tq, row, col + 1, asf + "q");
        // no call
        queensCombinations(qpsf, tq, row, col + 1, asf + "-");
    } else {
        // mauka no. 2 -> increment row, and column begin from 0
        // yes call
        queensCombinations(qpsf + 1, tq, row + 1, col: 0, asf + "q\n");
        // no call
        queensCombinations(qpsf, tq, row + 1, col: 0, asf + "-\n");
    }
}
```

queens permutation - 2D as 2D - Queen chooses!

n×n → chess board, n → non identical items / queen

print all possible ways to place n items in chess board.

level → queen / item.

Options → box

n = 3 - boxs

r = 2 → items

$$n_{P_r} = (n-0) \times (n-1) \times (n-2) \times \dots \times (n-(r-1))$$

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

no. of possibs. ways to place 2 items on 3 box.

1 2 -

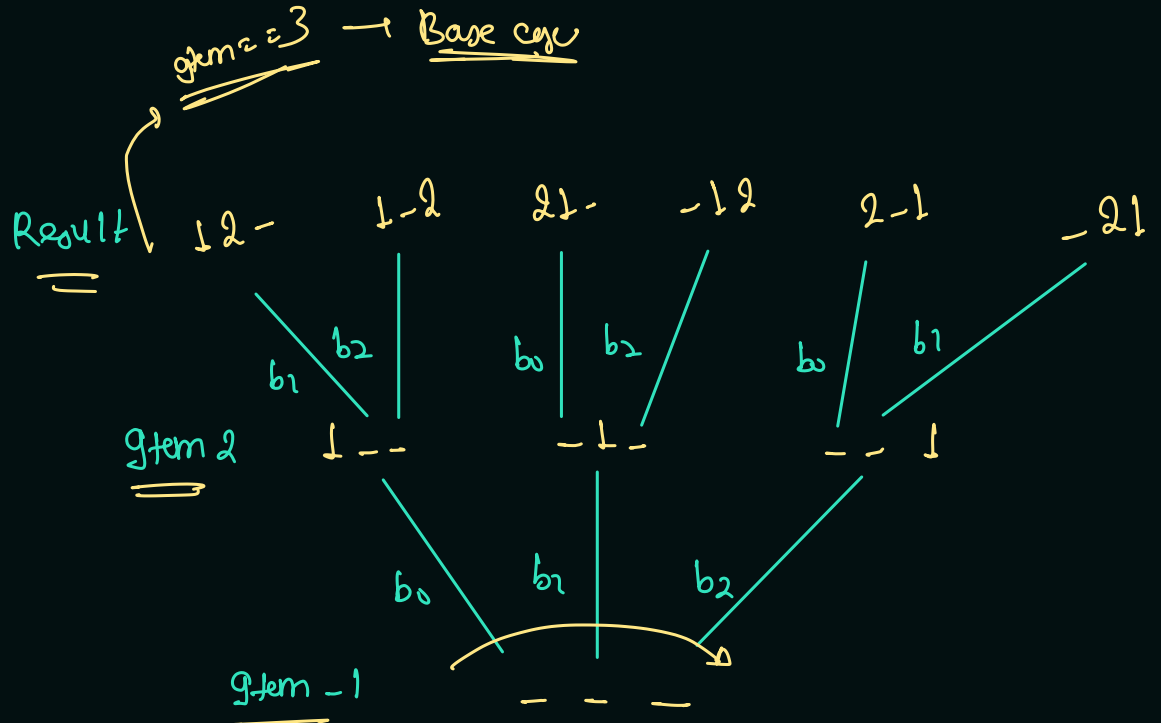
1 - 2

2 1 -

- 1 2

2 - 1

- 2 1



n = 2, Board = 2x2.

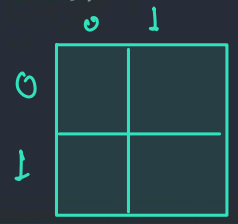
no. of non-Identical Items / queen = 2

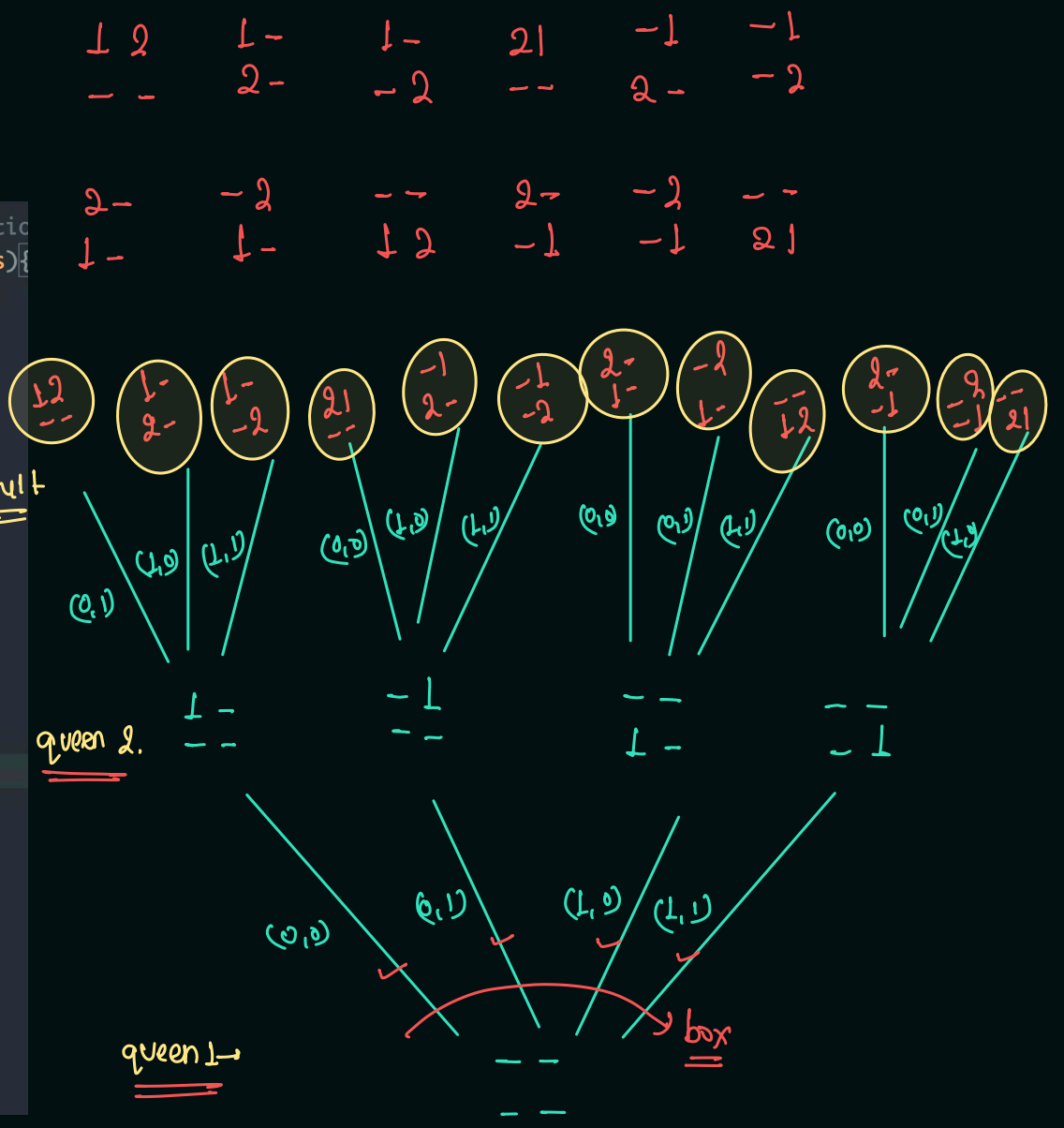
$^4P_2 = \frac{4!}{2!} = \frac{4 \times 3 \times 2!}{2!} = 12$

```java
// qpsf -> queen place so far, tq -> total queen, level -> queen, optic
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.length; j++) {
                if(chess[i][j] != 0)
                    System.out.print("q" + chess[i][j] + "\t");
                else
                    System.out.print(s: "-\t");
            }
            System.out.println();
        }
        System.out.println();
        return;
    }
    // option -> box
    for(int i = 0; i < chess.length; i++) {
        for(int j = 0; j < chess.length; j++) {
            if(chess[i][j] == 0) {
                chess[i][j] = qpsf + 1;
                queensPermutations(qpsf + 1, tq, chess);
                chess[i][j] = 0;
            }
        }
    }
}
```

Result

result





Queen permutation 2D as 2D - Box choosen →

nxn → chess board, n → non identical items/queen

print all possible ways to place n- items in chess board.

level → Box

option → queens.

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

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

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

^nC_r → Yes or No

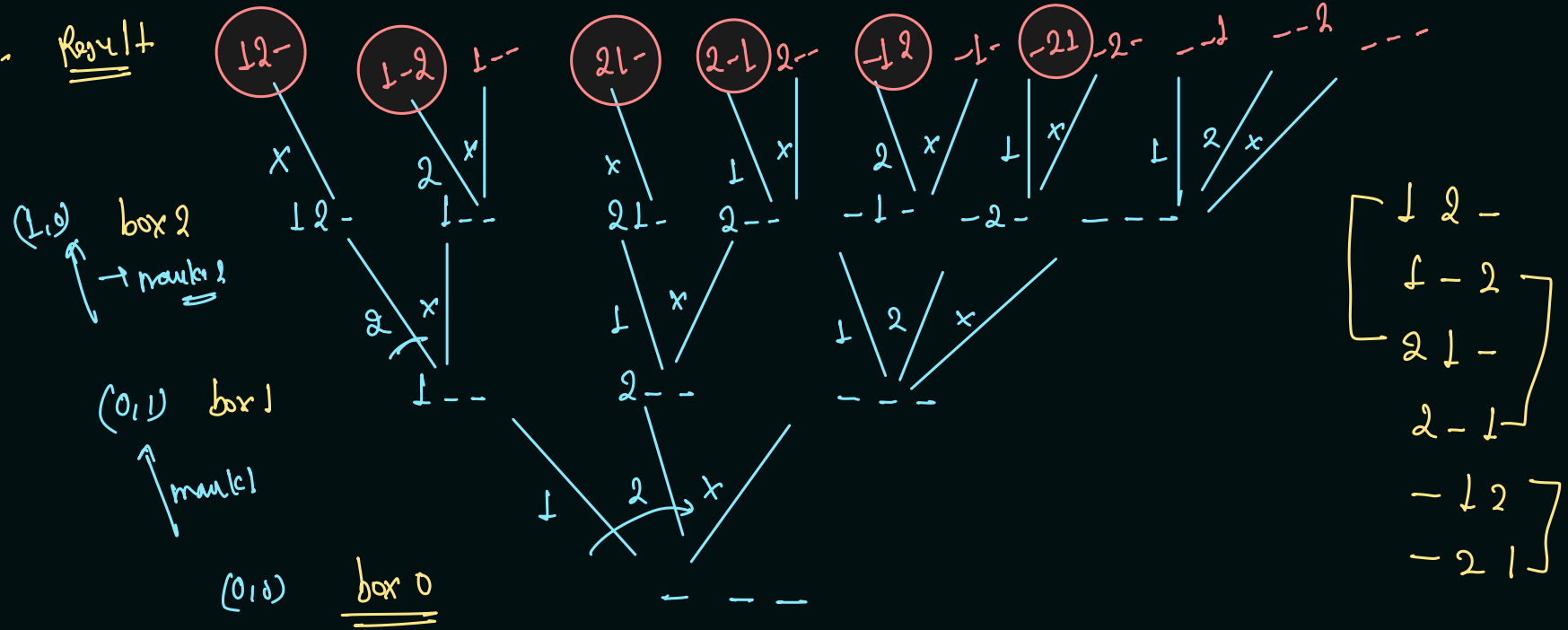
$r!$ → items permutation

2^n → Yes or No for each item

n=3 → box

r=2 → items

Result



[1 2 -
1 - 2
2 1 -
2 - 1
- 1 2
- 2 1]

queens combination - 2D as 2D- queen chooses :

n×n → chess board

n- identical items / queen.

print all possible ways.

level → queen

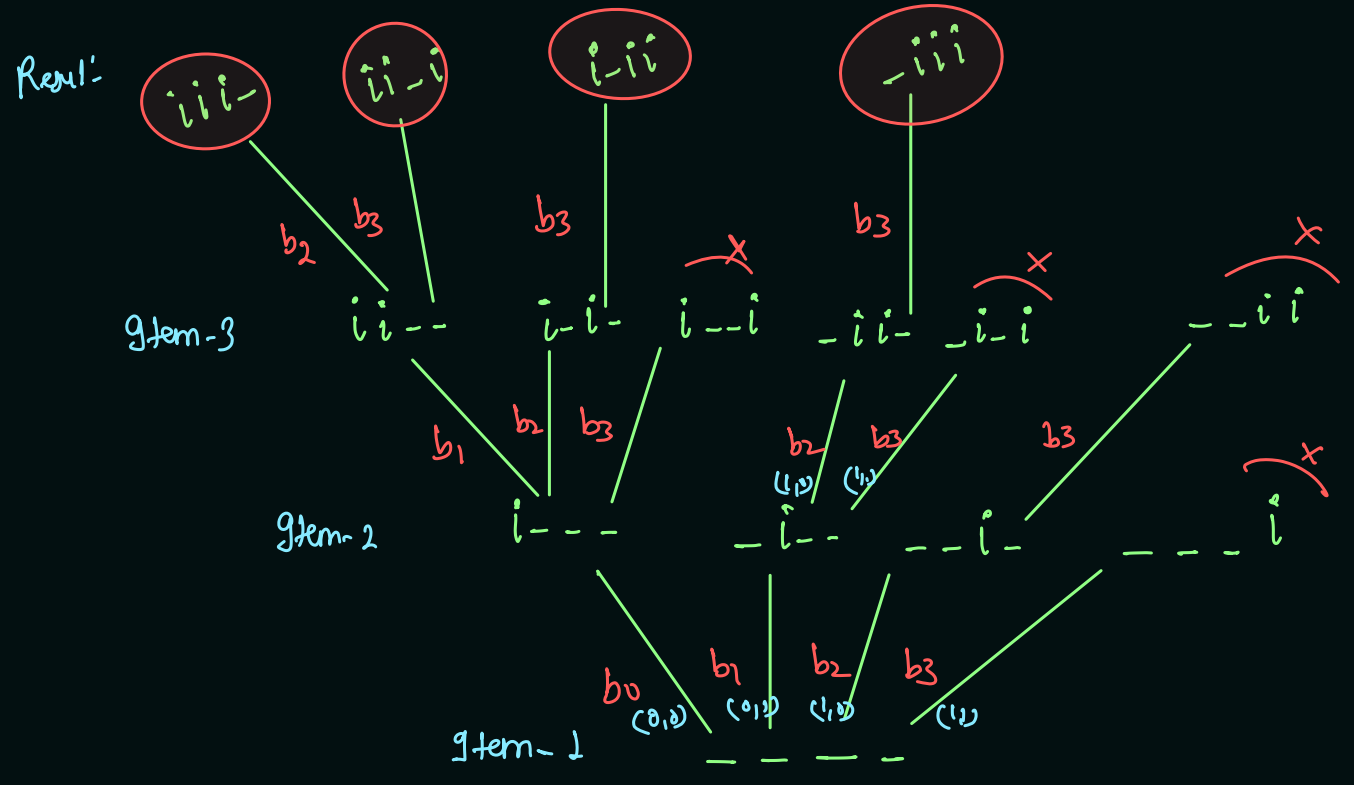
option → box

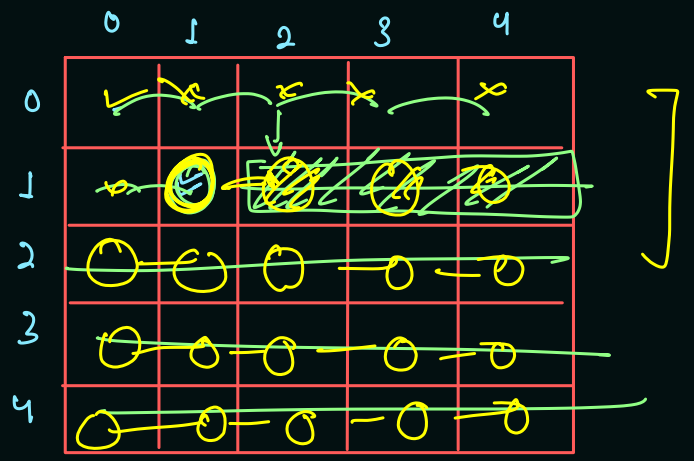
n = 4- box

r = 3 → items.

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

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





to present the Sorted order calling protocol.

current R.

(1, 1) → Row

2, 0

complete loop

```
// finish current row
for(int c = j + 1; c < chess.length; c++) {
    int r = i;
    chess[r][c] = true;
    queensCombinations(qpsf + 1, tq, chess, r, c);
    chess[r][c] = false;
}

// fininsh remaining board
for(int r = i + 1; r < chess.length; r++) {
    for(int c = 0; c < chess.length; c++) {
        chess[r][c] = true;
        queensCombinations(qpsf + 1, tq, chess, r, c);
        chess[r][c] = false;
    }
}
```

in next Row, from zeroth colum