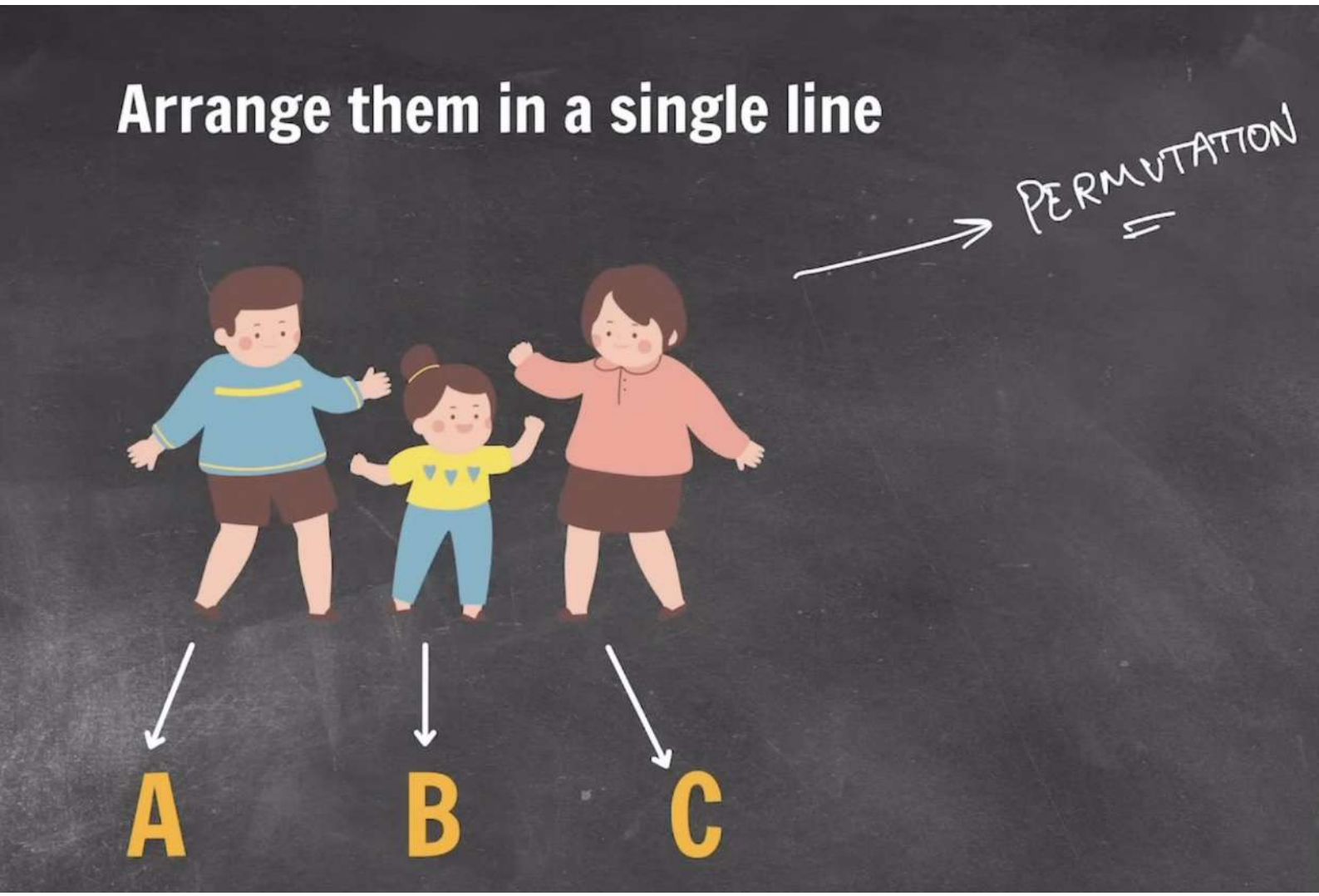
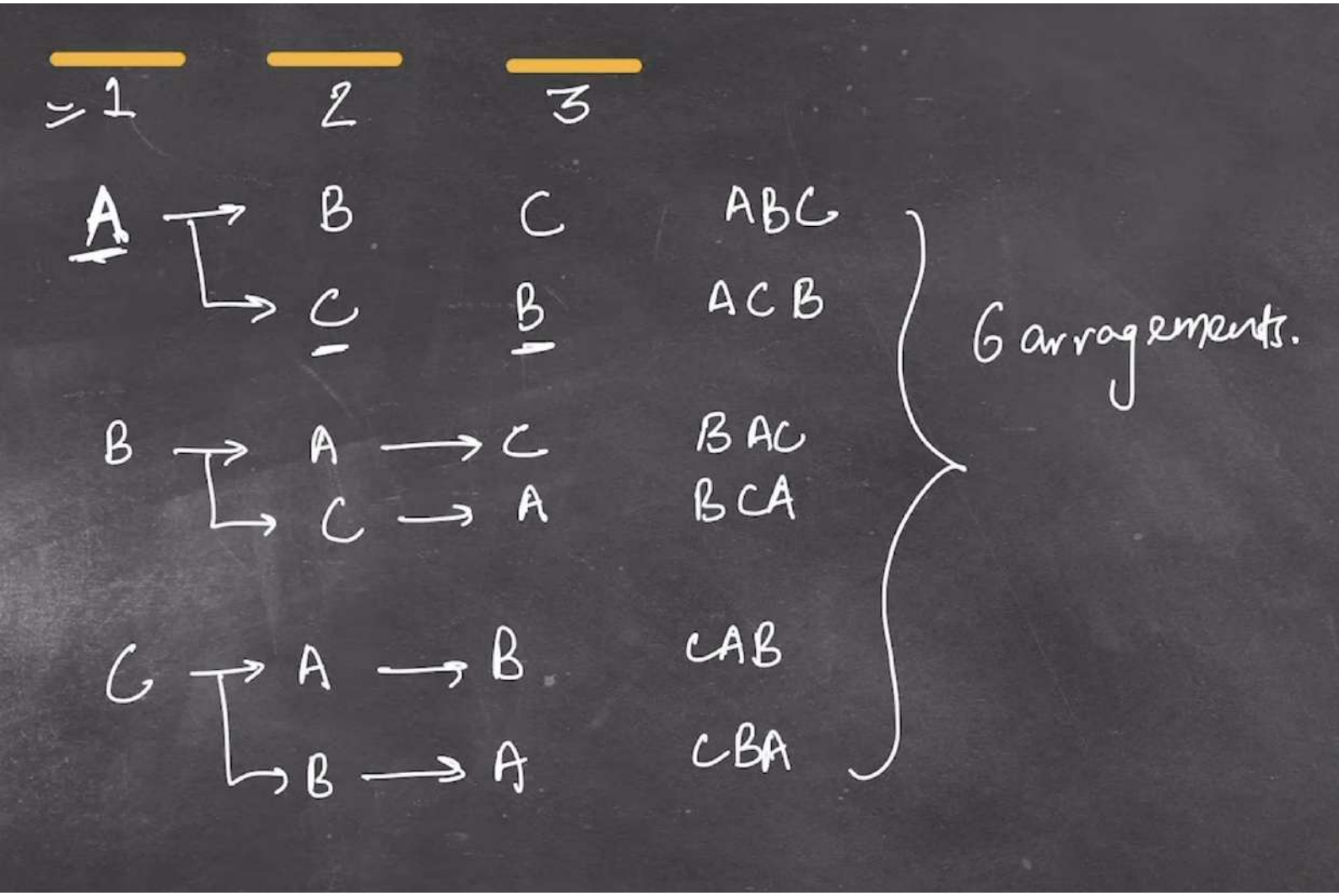
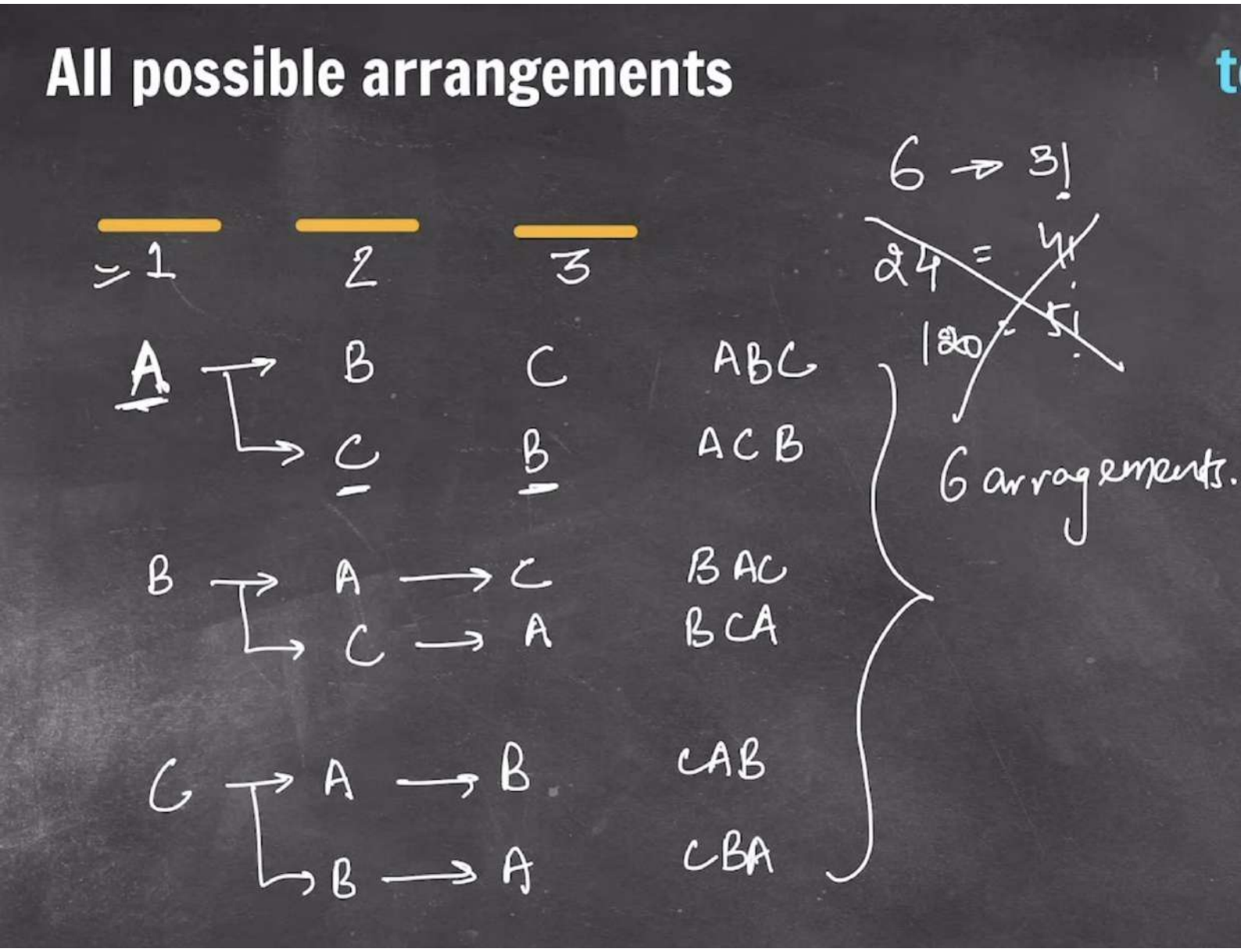
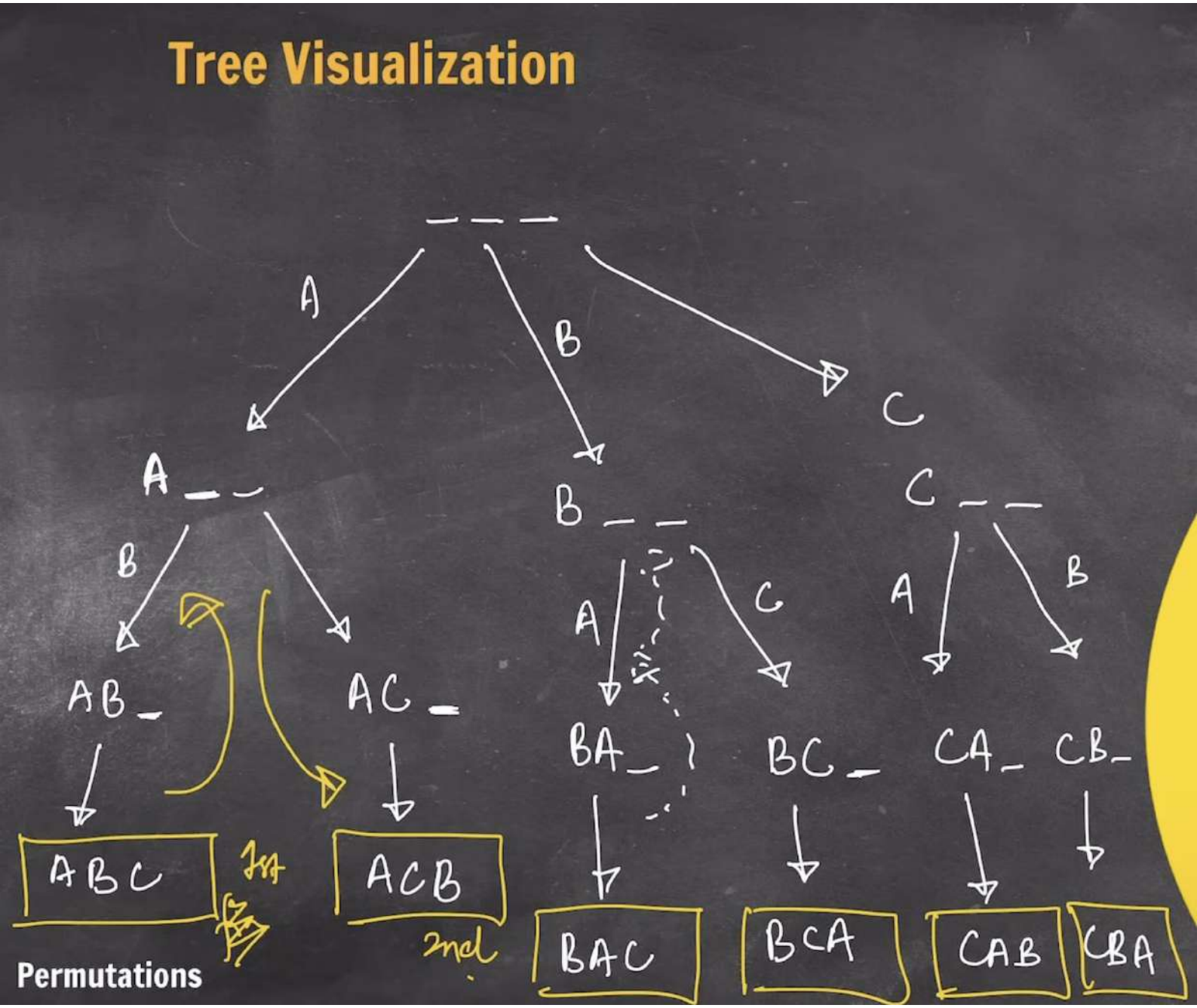

Arrange them in a single line
PERMUTATION
A
B
C


1
2
3
A
B
C
ABC
C
B
ACB
B
A
C
BAC
C
A
BCA
C
A
B
CAB
B
A
CBA
6 arrangements.

All possible arrangements




Tree Visualization
A
B
C
A _ _
B _ _
C _ _
B
A
C
A
B
AB _
AC _
BA _
BC _
CA _
CB _
ABC
ACB
BAC
BCA
CAB
CBA
1st
2nd
Permutations

```java
public static void printPermutation(String str, int idx, String perm) {
    if(str.length() == 0) {
        System.out.println(perm);
        return;
    }

    for(int i=0; i<str.length(); i++) {
        char currChar = str.charAt(i);
        String newStr = str.substring(0, i) + str.substring(i+1);
        printPermutation(newStr, idx+1, perm+currChar);
    }
}
```

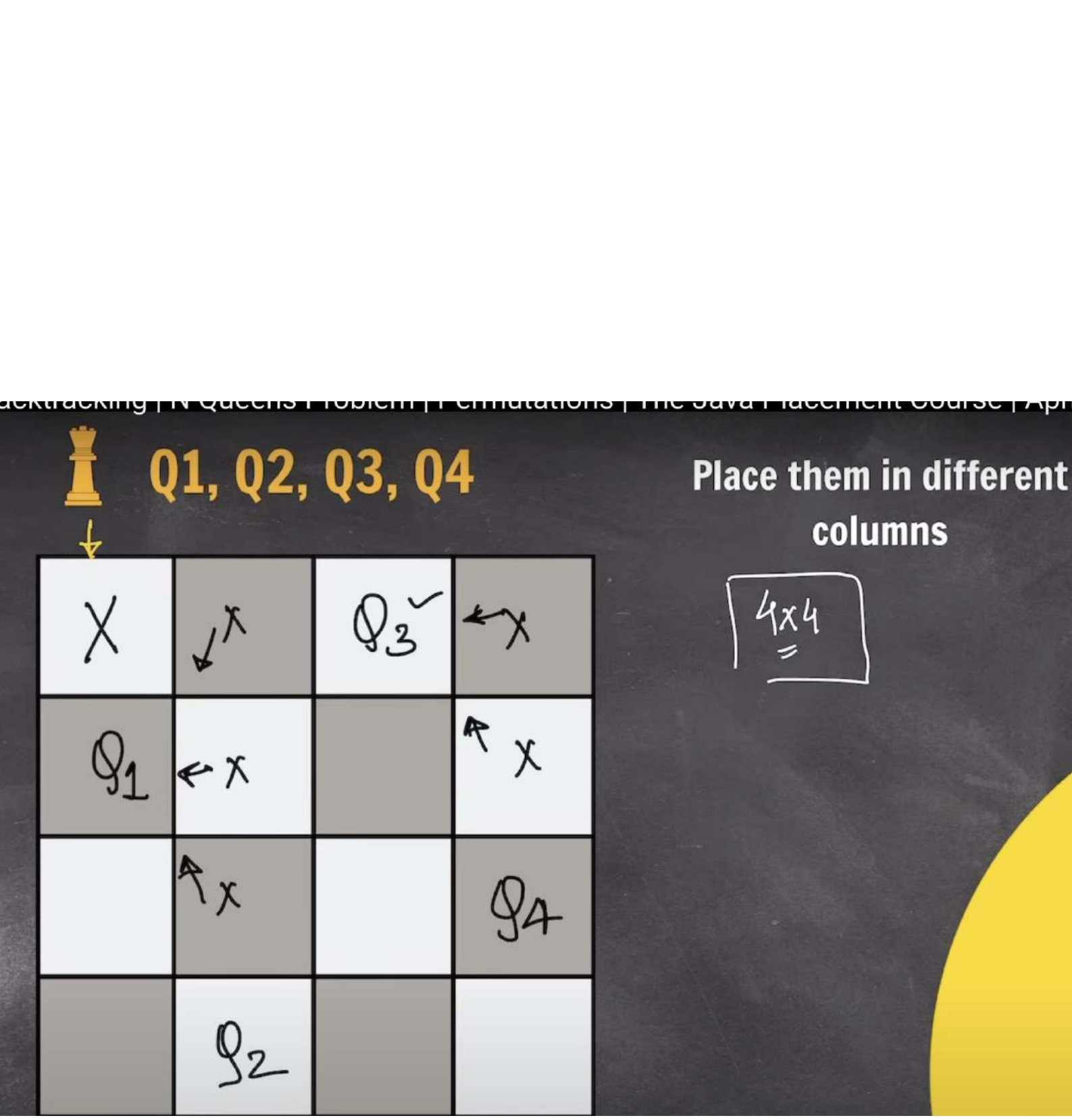
time complexity = O(n*n!)

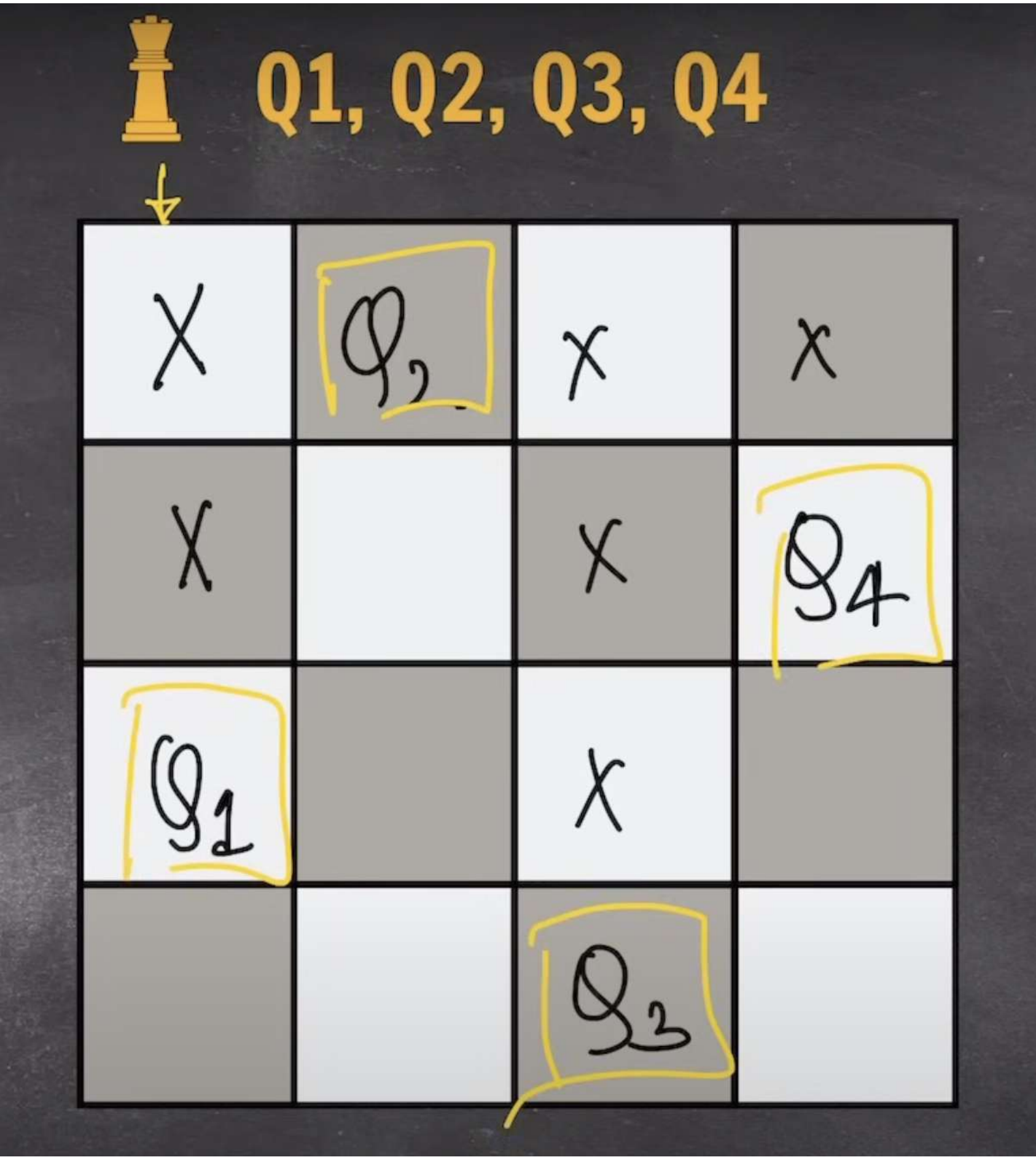
N-Queens

N x N chessboard

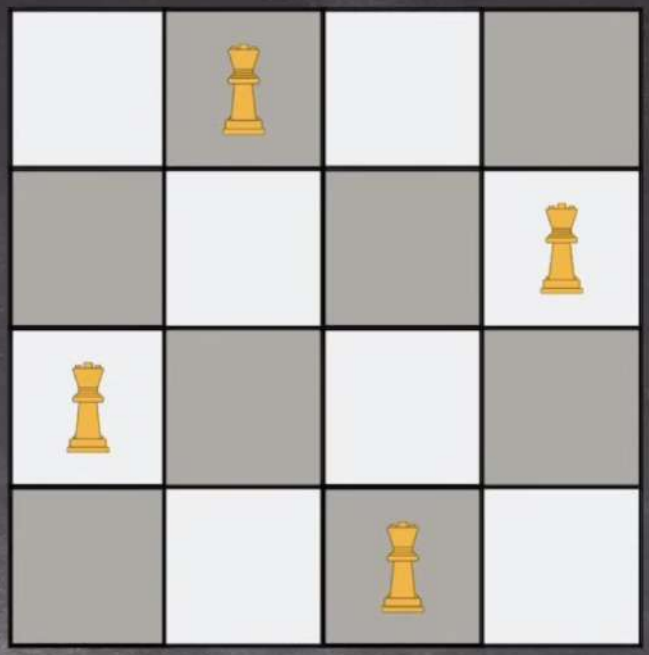
N Queens

Print all solutions where queens are safe

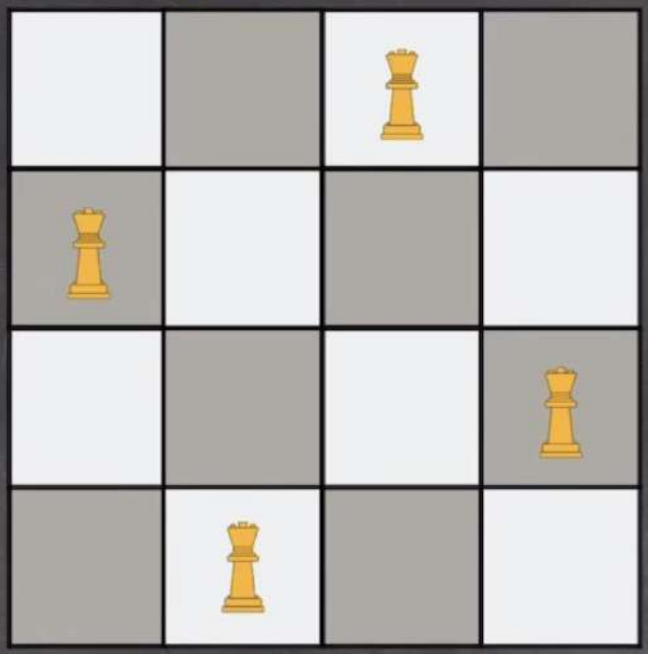

Q1, Q2, Q3, Q4
Place them in different columns
4x4
Q1
Q2
Q3
Q4


Q1, Q2, Q3, Q4
X
Q2
X
X
X
X
Q4
Q1
X
Q3

Result 2



Result 1



Possible Solutions

Code

```java
public boolean isSafe(char[][]board, int row, int col) {
    //horizontal
    for(int j=0; j<board.length; j++) {
        if(board[row][j] == 'Q') {
            return false;
        }
    }

    //vertical
    for(int i=0; i<board.length; i++) {
        if(board[i][col] == 'Q') {
            return false;
        }
    }

    //left upper
    int r = row;
    for(int j=col; j>=0 && r>=0; j--,r--) {
        if(board[r][j] == 'Q') {
            return false;
        }
    }

    //left lower
    r = row;
    for(int j=col; j>=0 && r<board.length; j--, r++) {
        if(board[r][j] == 'Q') {
            return false;
        }
    }
```

time complexity = O(n^n)

n^n