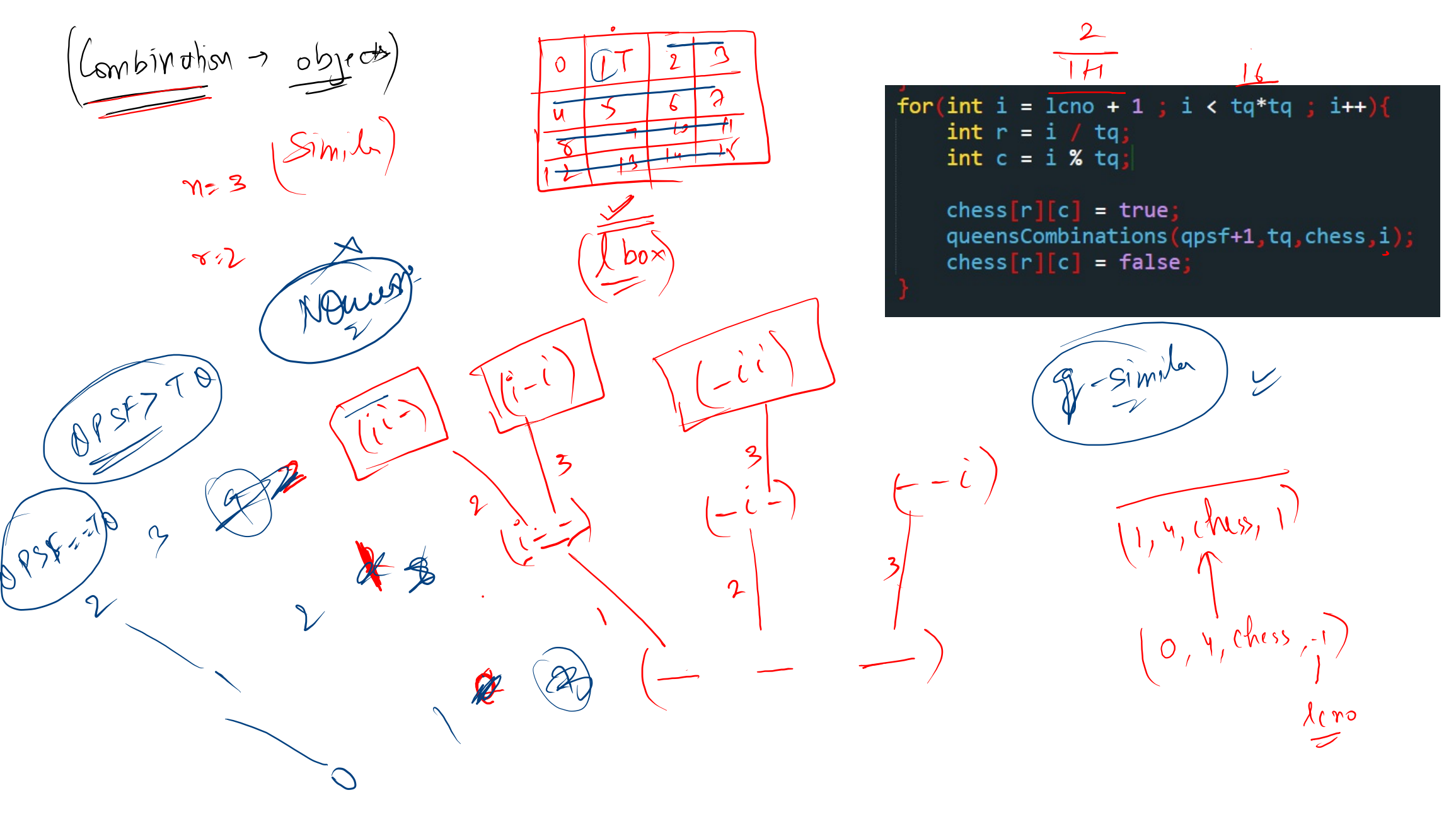
Queens → Combination → 2d cs id

(= → Permutation → id + N Queens)

(= → Combination → id + NQueen)

Word Permutation

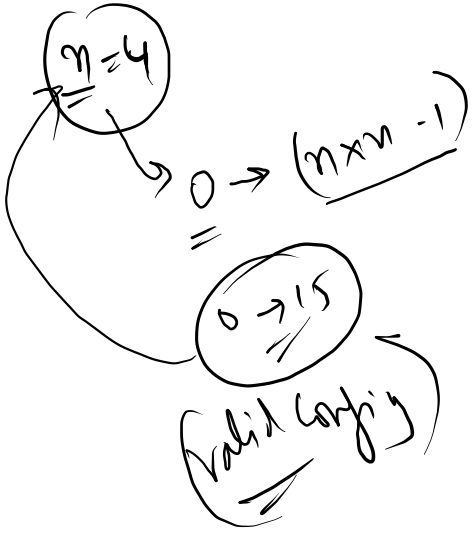
N Knights Combination.

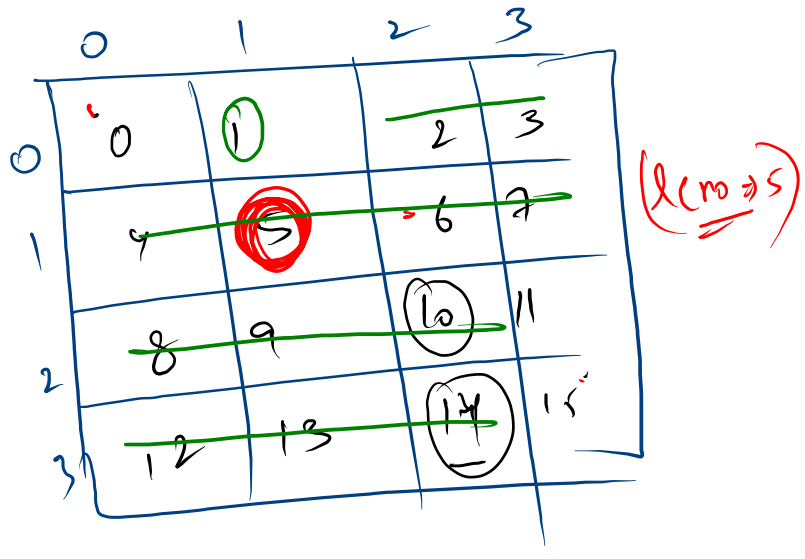

(Combination → object)
n = 3 (Similar)
r = 2
NQueens(2)
box
(box)
0 1T 2 3
4 5 6 7
8 9 10 11
12 13 14 15
for(int i = lcno + 1 ; i < tq*tq ; i++){
int r = i / tq;
int c = i % tq;
chess[r][c] = true;
queensCombinations(qpsf+1,tq,chess,i);
chess[r][c] = false;
}
2
TH
16
qpsf == tq
qpsf > tq
(i i -)
(i -i)
(-i i)
(-i -)
(- -i)
(- - -)
3
2
1
0
g-Similar
(1, 4, chess, 1)
(0, 4, chess, -1)
lcno

# Queens (2d → 1d) (Combination) → Queen chosen

↳ Combination objects

n = 4



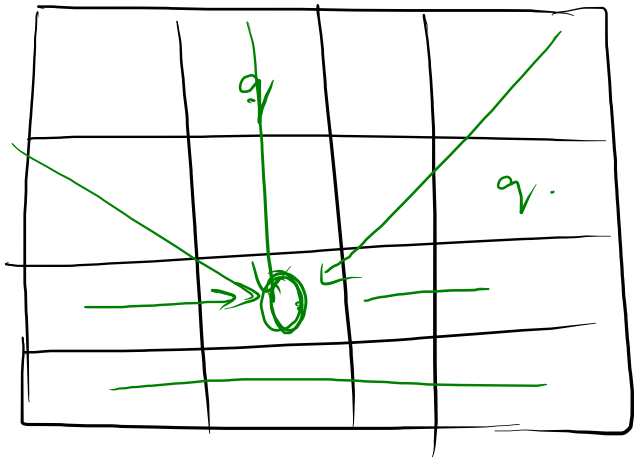




$$\begin{cases} 10 \to & 10/4 \Rightarrow 2 \text{ (row no.)} \\ 10 \to & 10\%4 \Rightarrow 2 \text{ (col no.)} \end{cases}$$

$$\begin{cases} 14 \to & 14/4 \to 3 \\ 14 \to & 14\%4 \to 2 \end{cases}$$

N=4

Backtracking

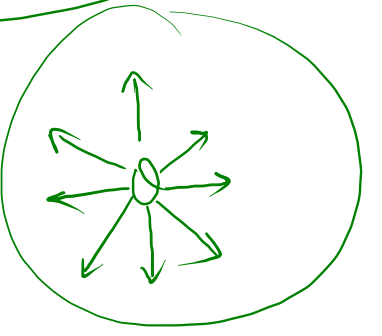


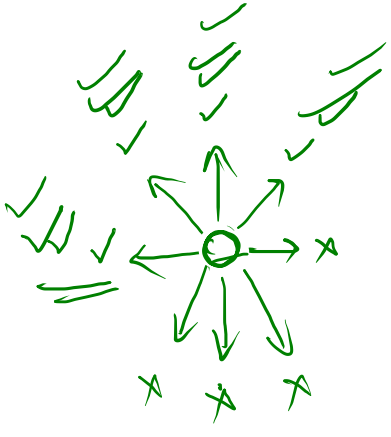
4 Queens

+

N Queens

Combination

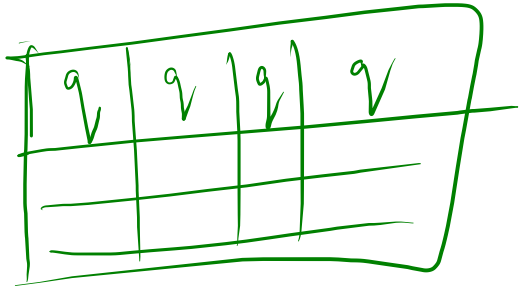


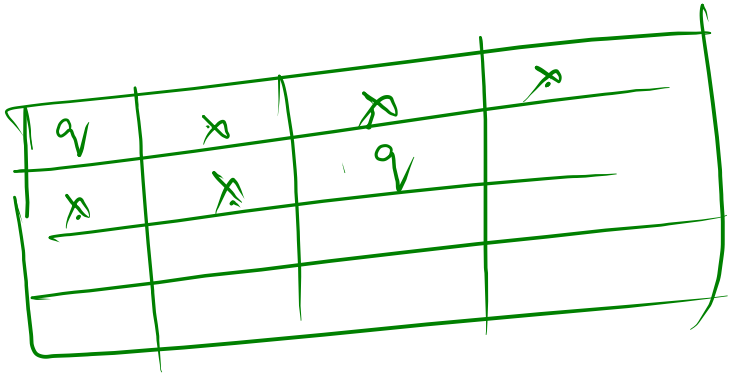


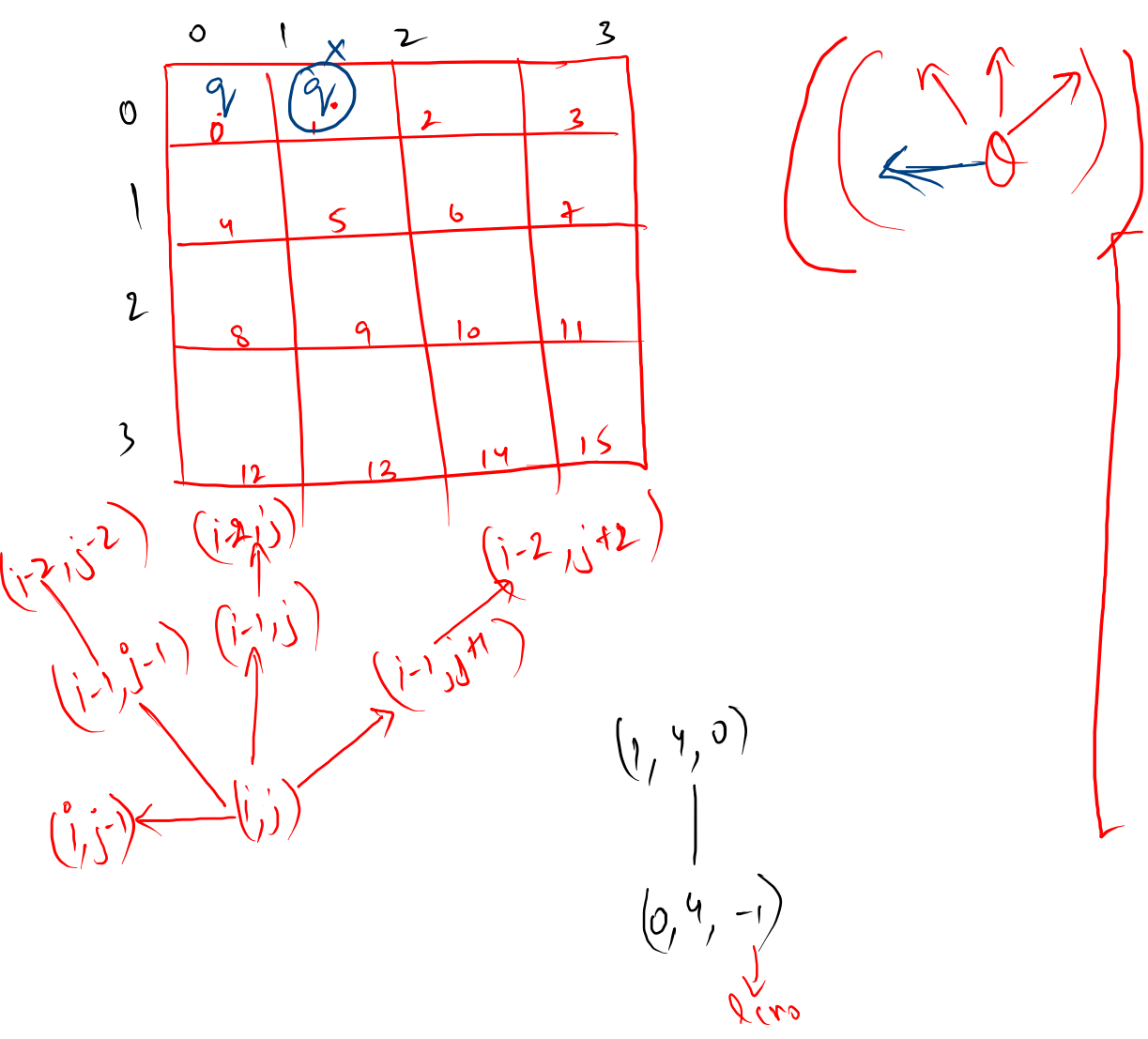
Empty Space + Combinations

Is Safe

N Queens





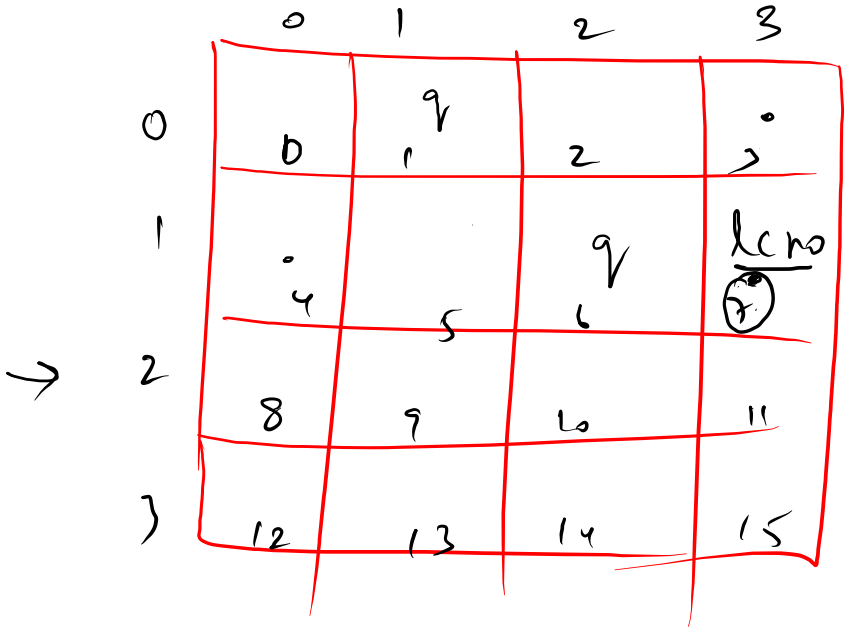





```java
public static void nqueens(int qpsf, int tq, boolean[][] chess, int lcno) {
    if (qpsf == tq) {
        for (int row = 0; row < chess.length; row++) {
            for (int col = 0; col < chess.length; col++) {
                System.out.print(chess[row][col] ? "q\t" : "-\t");
            }
            System.out.println();
        }
        System.out.println();
        return;
    }

    for (int i = lcno + 1; i < chess.length * chess.length; i++) {
        int row = i / chess.length;
        int col = i % chess.length;

        if (IsQueenSafe(chess, row, col)) {
            chess[row][col] = true;
            nqueens(qpsf + 1, tq, chess, i);
            chess[row][col] = false;
        }
    }
}
```




(cellno ⇒ 5)

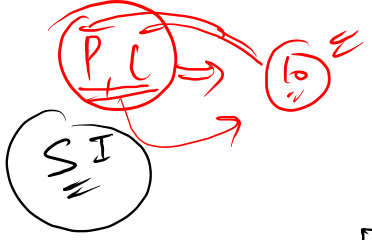
↳ row ⇒ 5/4 ⇒ 1

col ⇒ 5 % 4 ⇒ 1

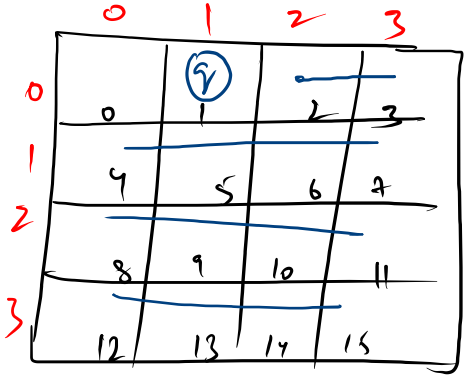
4[row + 1] − 1 ⇒ 4[2] − 1 ⇒ 7

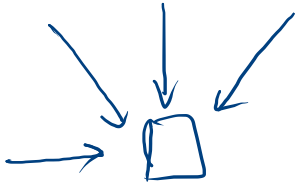
4[0+1] − 1 ⇒ 3

4[1+1] − 1 ⇒ 7

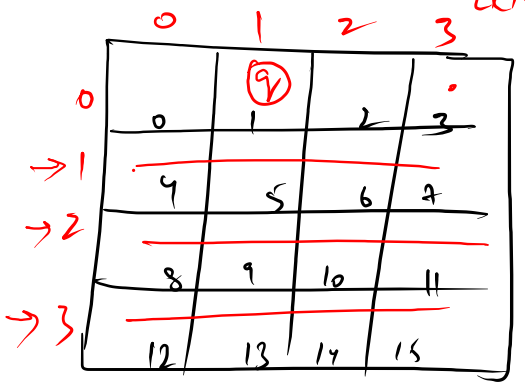


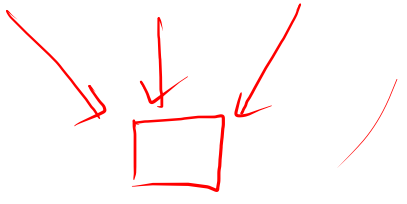

L(NO) ⇒ i(pos)




L(NO) ⇒ 3 (tq-[800w+1]-1)

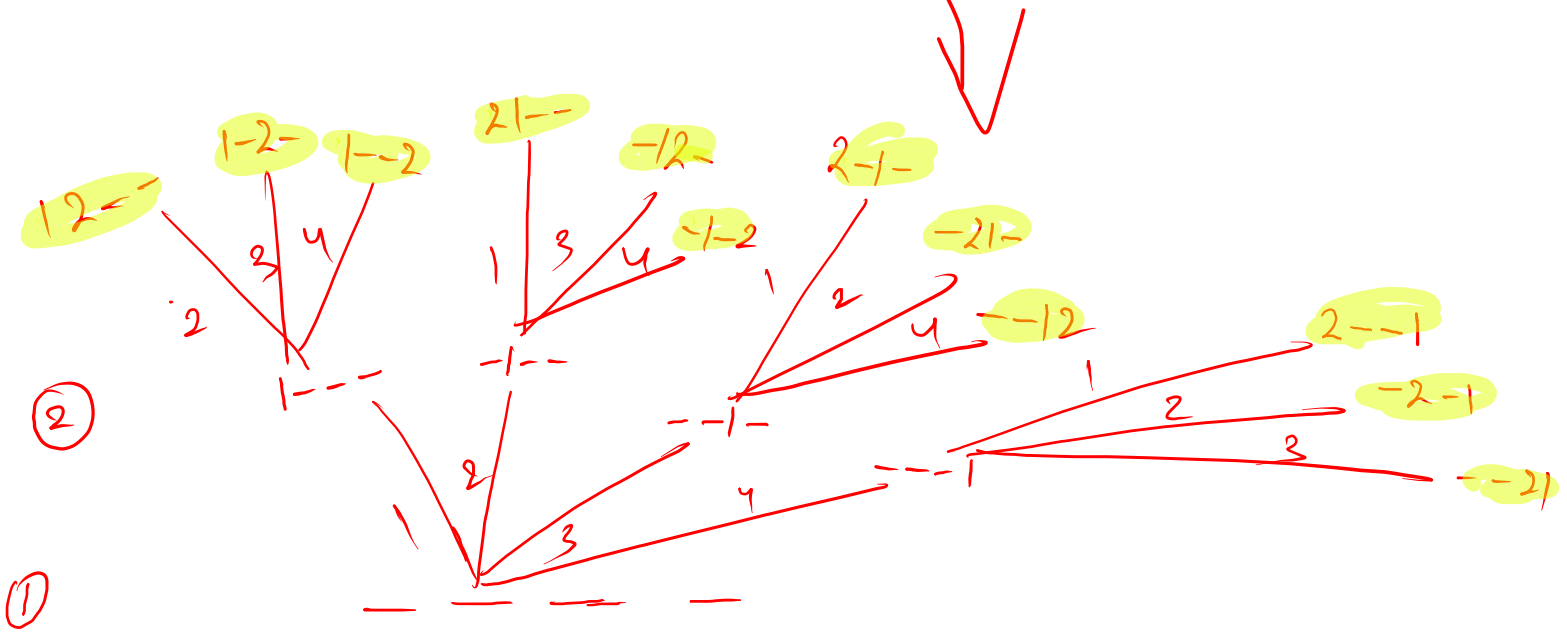


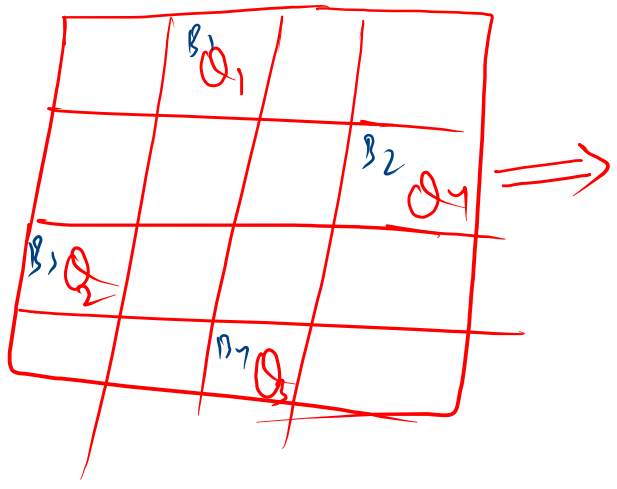
Permutation → object

$n = 4$

$r = 2$



n=4

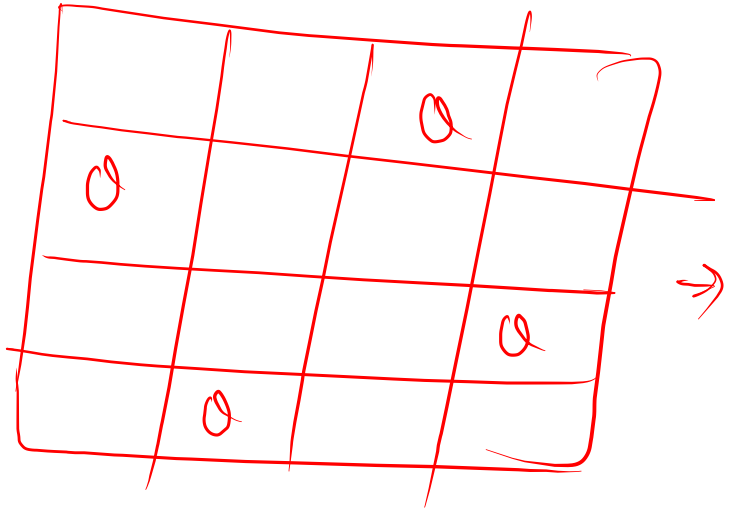


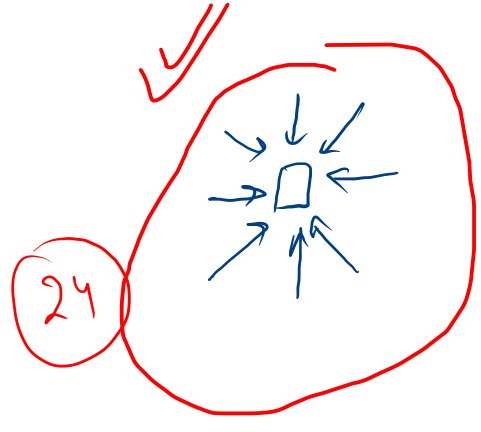

$B_1 \to \overline{4}$ , $B_2 \to \overline{3}$, $B_3 \to \overline{2}$, $B_4 \to \overline{1}$ ⇒ 4! = 24

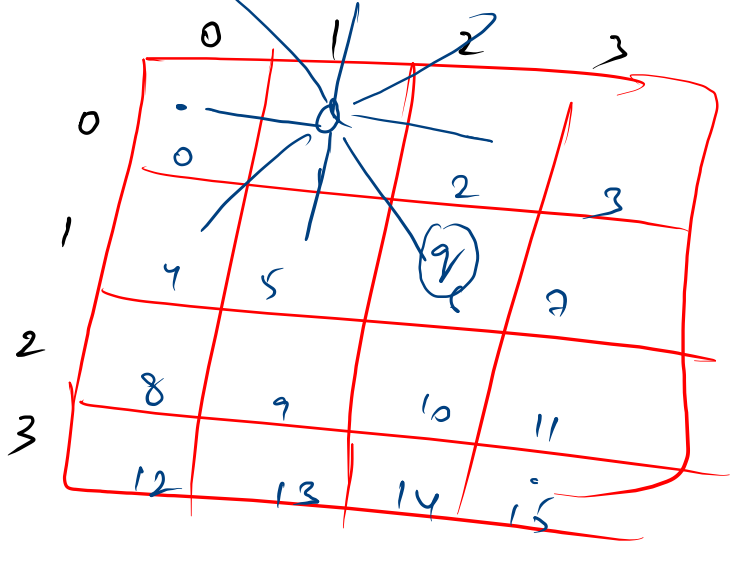
(cell no → row ⇒ cell no./tq
→ Col ⇒ cell no % tq)

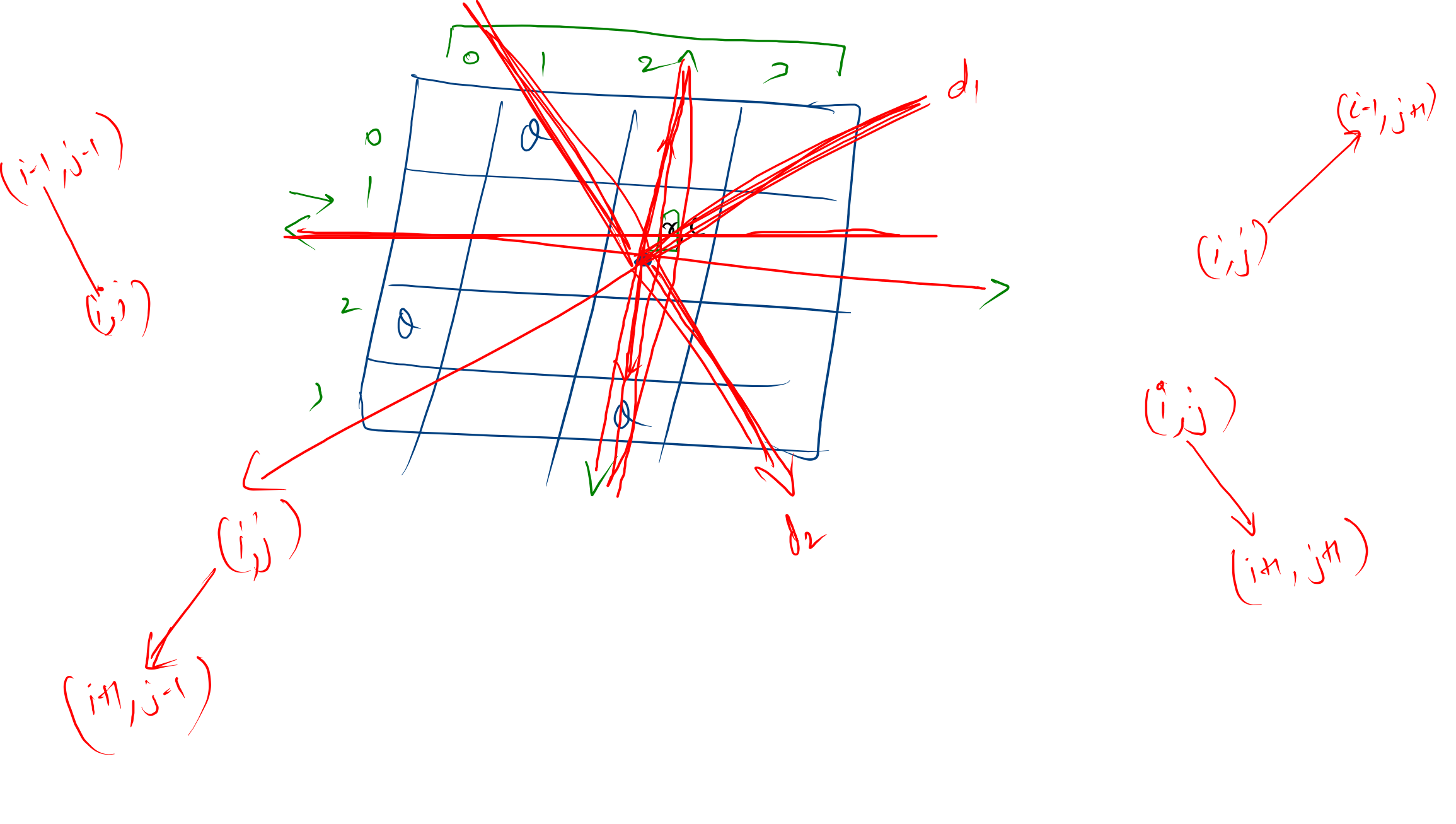
Permutation → 48

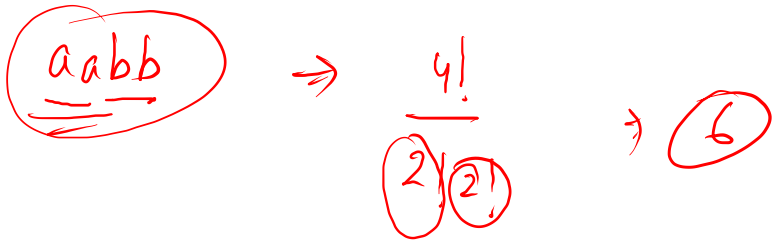
((tq * row) + Col → cell no.)




0 1 2 3
0
1
2
3
$d_1$
$d_2$
$(i-1, j-1)$
$(i, j)$
$(i-1, j+1)$
$(i, j)$
$(i, j)$
$(i+1, j-1)$
$(i, j)$
$(i+1, j+1)$


aabb
4!
2!2!
6

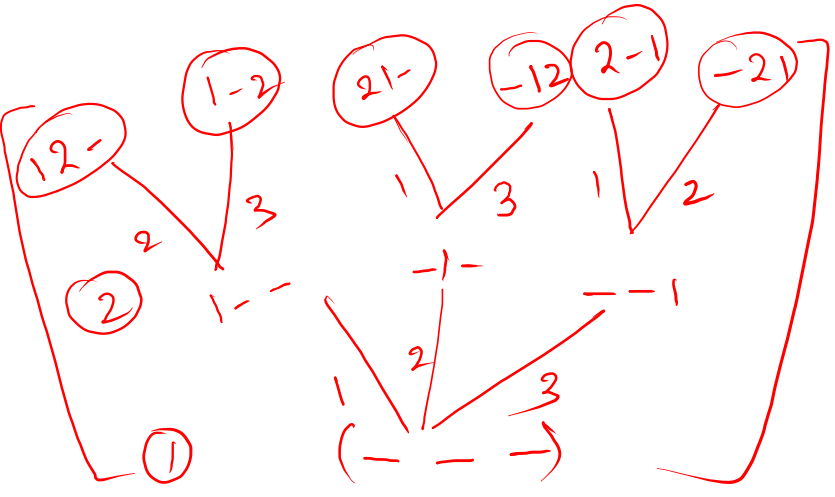
Permutation

(object choose)

$n = 3$

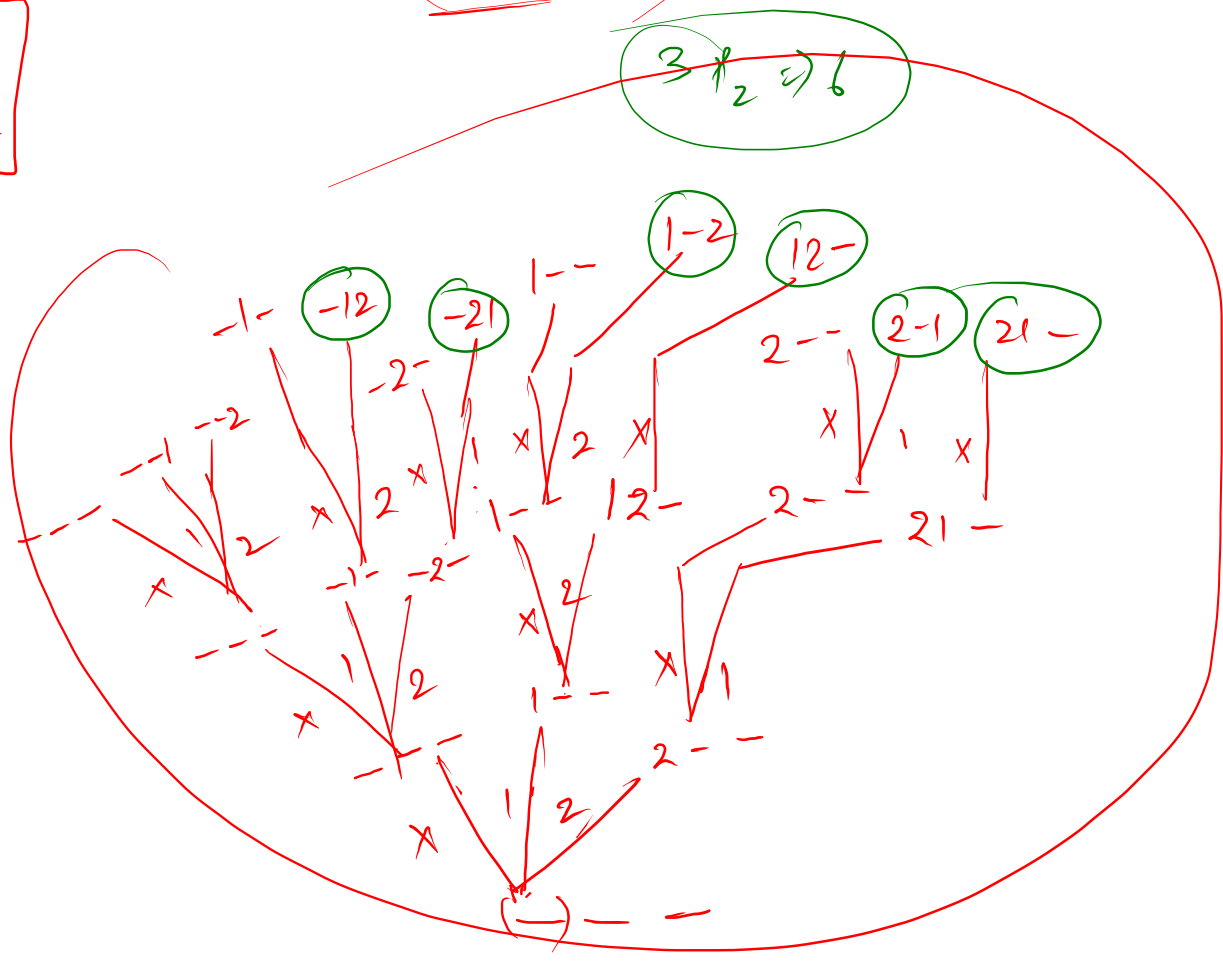
$r = 2$

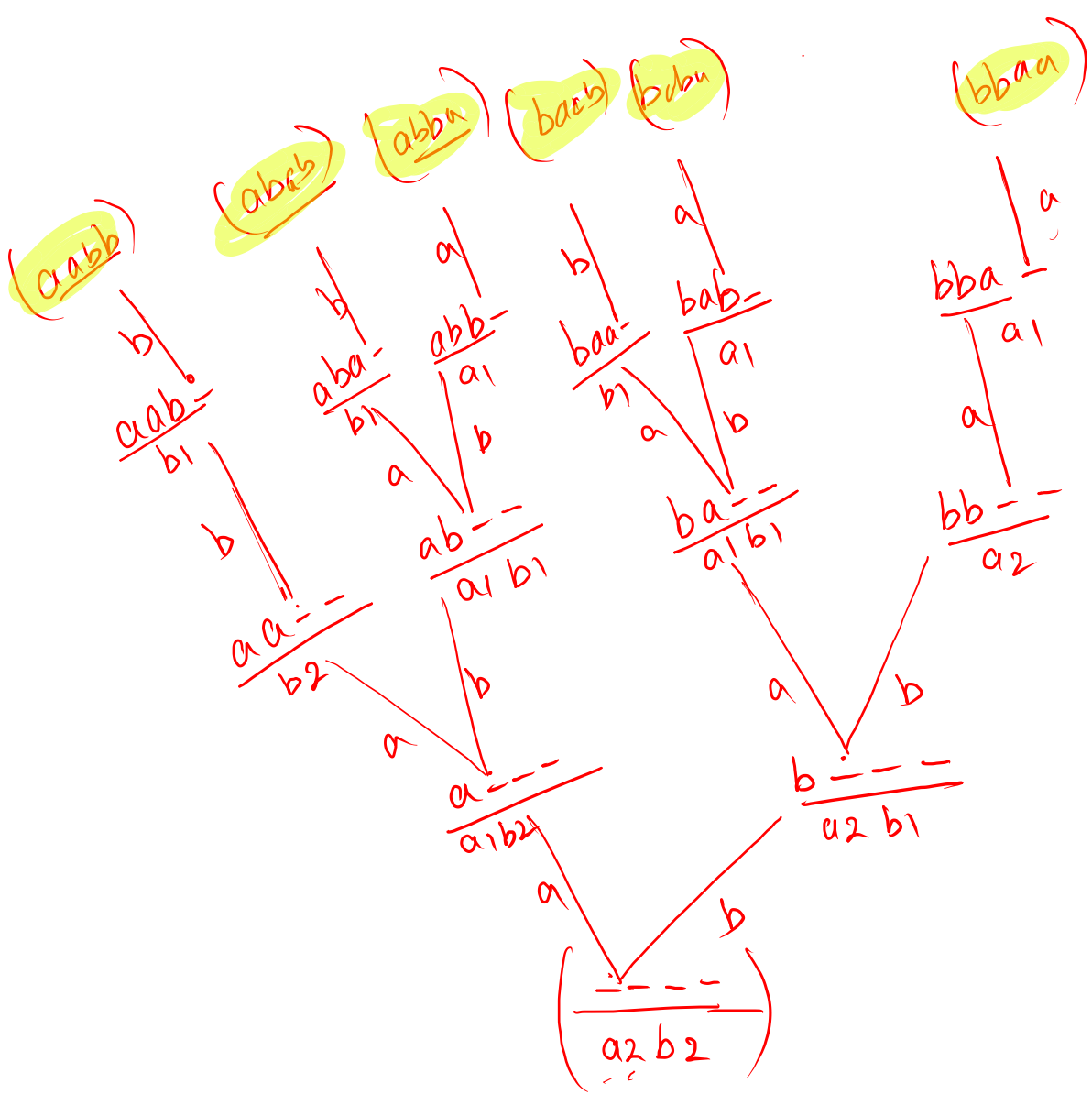
$^3P_2 = 6$

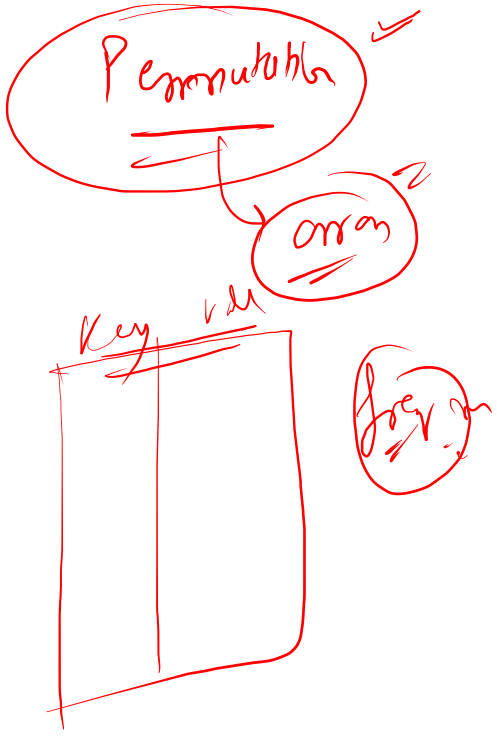


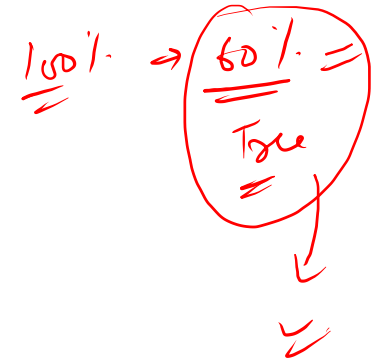

(box choose)

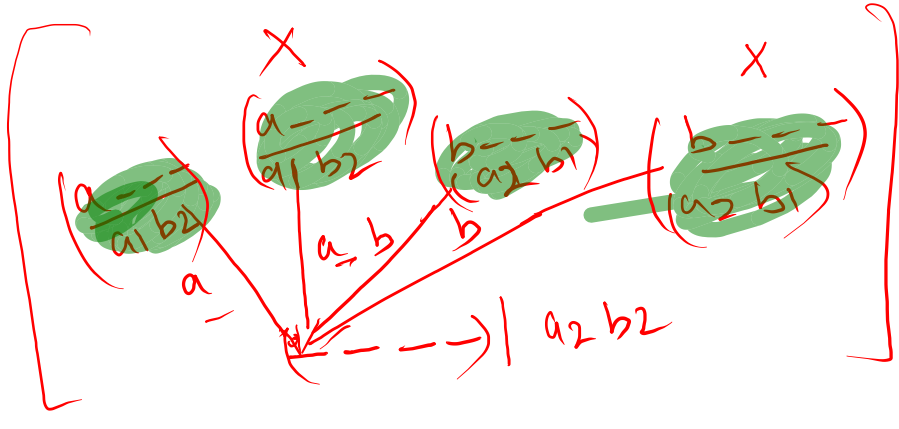
$^3P_2 = 6$

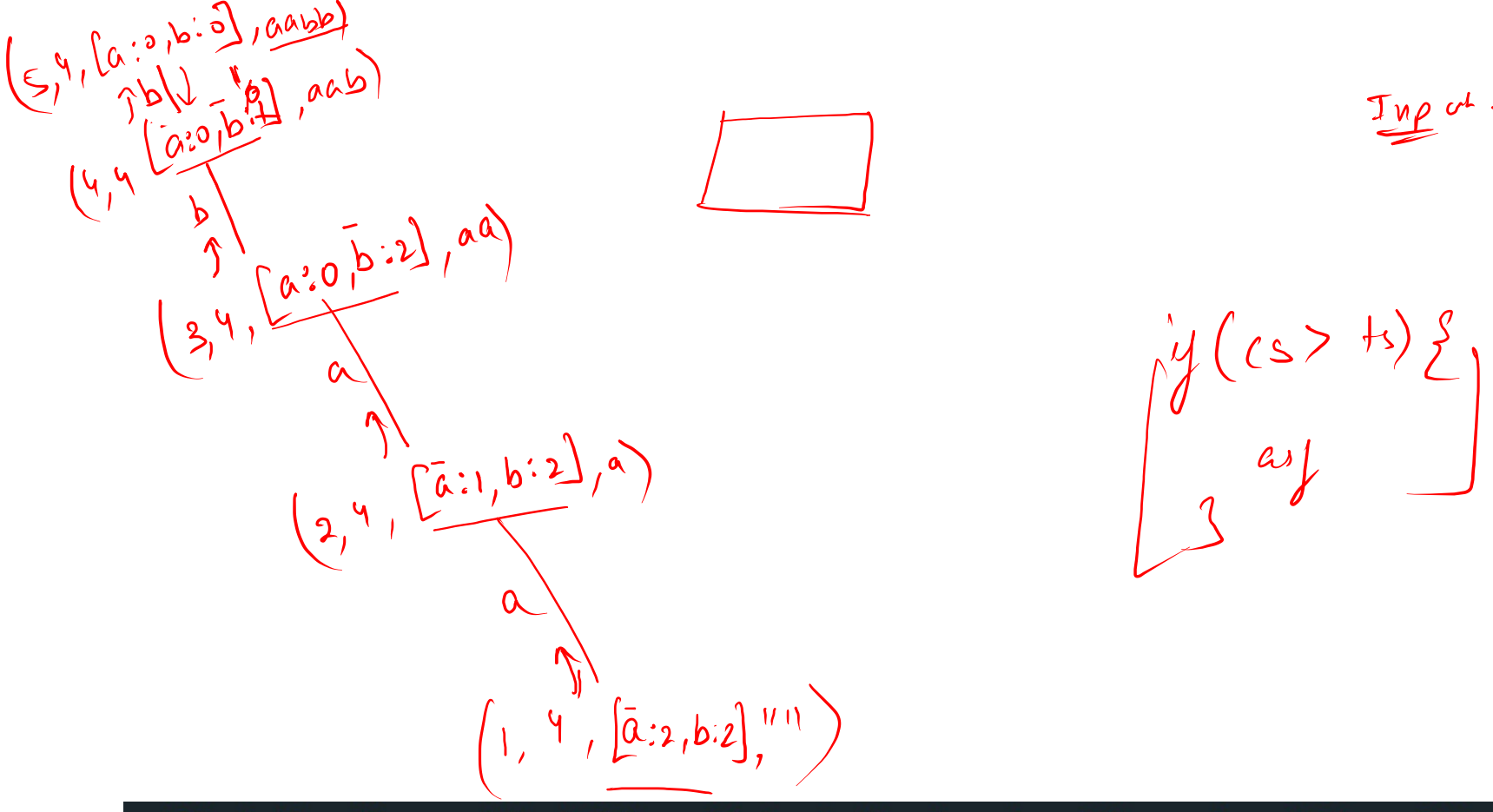



aabb
abab
abba
baab
bbaa
aab
aba
abb
baa
bab
bba
aa
ab
ba
bb
a
b
aa
ab
ba
bb
a
b
a
b
a
b
a
b
a
b
a
b
a
b
a
b


Permutation
Comb
Key
Value

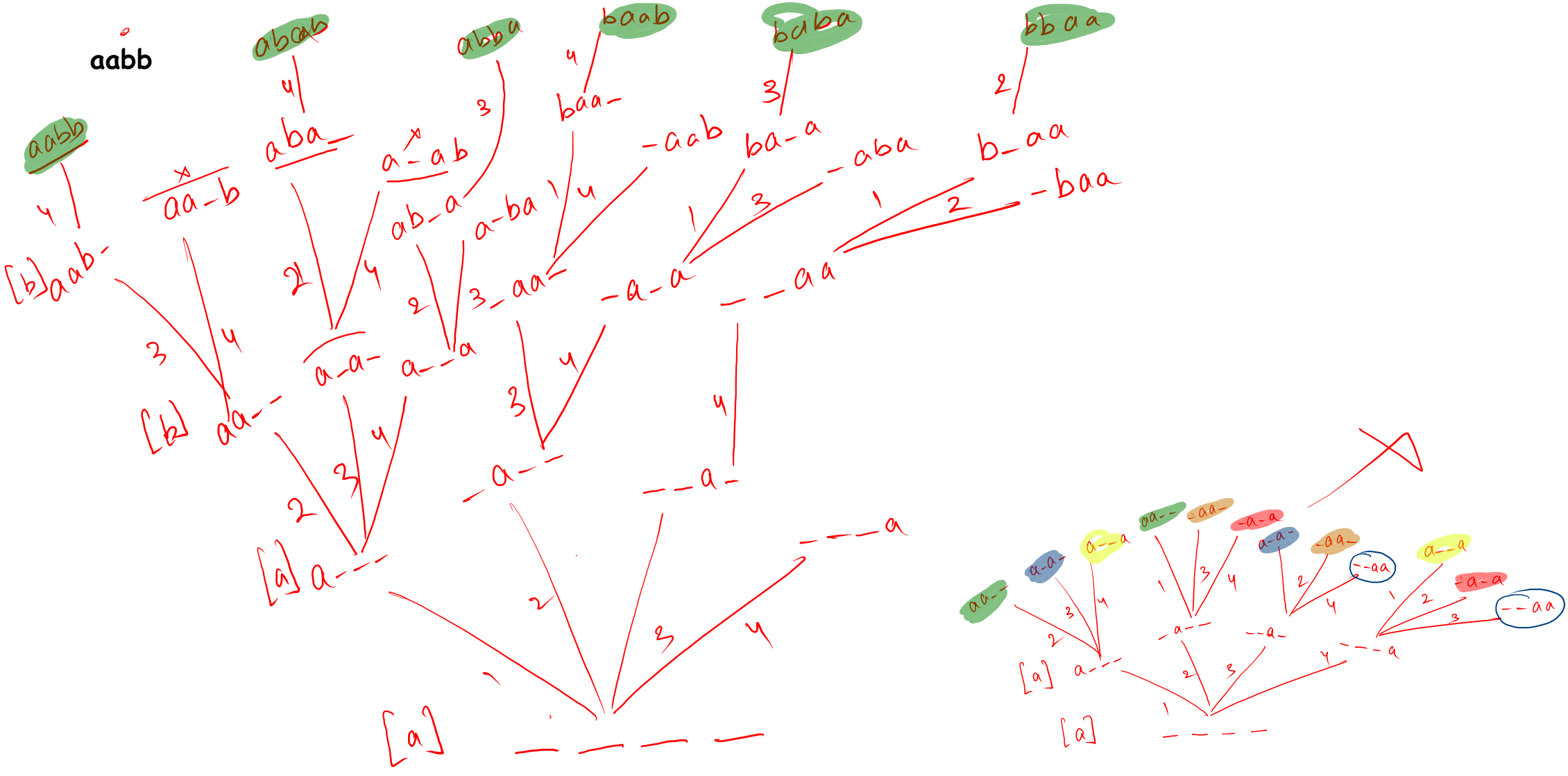

Is it
So it
True

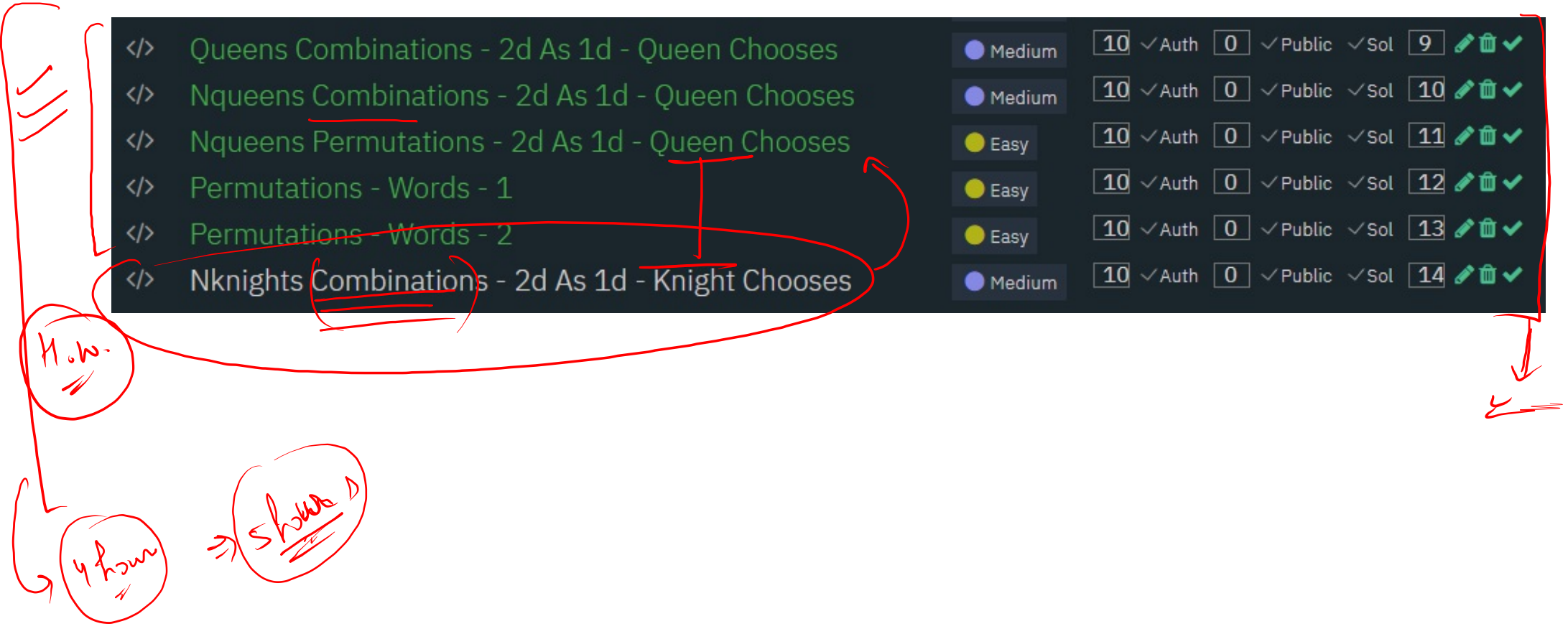

X
X
a
b
a
b
a
b

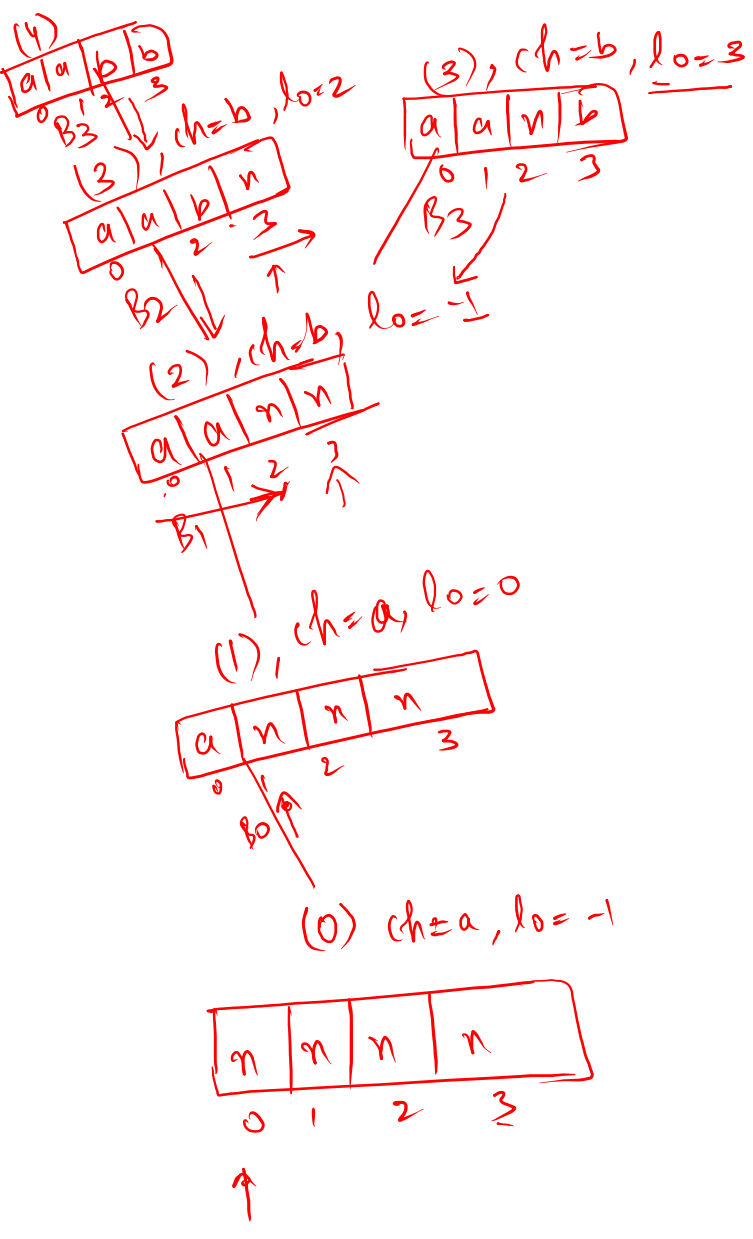



```java
public static void generateWords(int cs, int ts, HashMap<Character, Integer> fmap, String asf) {
  for(char ch : fmap.keySet()){
      if(fmap.get(ch) > 0){
          fmap.put(ch,fmap.get()-1);
          generateWords(cs+1,ts,fmap,asf+ch);
          fmap.put(ch,fmap.get()+1);
      }
  }
}
```

aabb




Queens Combinations - 2d As 1d - Queen Chooses
Medium
10 Auth 0 Public Sol 9
Nqueens Combinations - 2d As 1d - Queen Chooses
Medium
10 Auth 0 Public Sol 10
Nqueens Permutations - 2d As 1d - Queen Chooses
Easy
10 Auth 0 Public Sol 11
Permutations - Words - 1
Easy
10 Auth 0 Public Sol 12
Permutations - Words - 2
Easy
10 Auth 0 Public Sol 13
Nknights Combinations - 2d As 1d - Knight Chooses
Medium
10 Auth 0 Public Sol 14




aabb

str = aabb

lastOccurrence = { a : 1, b : -1 }

```
public static void generateWords(int cc, String str, Character[] spots,HashMap<Character, Integer> lastOccurence)
{
    if(cc == str.length()){
        for(Character ch: spots){
            System.out.print(ch);
        }
        System.out.println();
        return;
    }
    char ch = str.charAt(cc);
    int lo = lastOccurence.get(ch);

    for(int i = lo+1 ; i < spots.length ; i++){
        if(spots[i] == null){
            spots[i] = ch;
            lastOccurence.put(ch,i);
            generateWords(cc+1,str,spots,lastOccurence);
            lastOccurence.put(ch,lo);
            spots[i] = null;
        }
    }
}
```