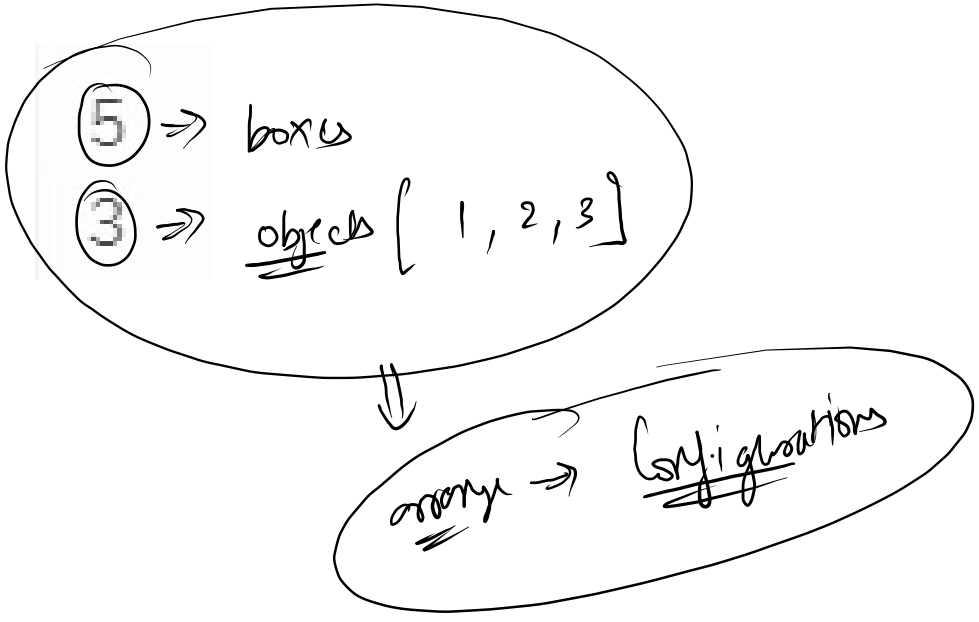
Permutation + Combination



Permutation-1

↳ (object chosen)

$n = 3$ [boxes]

$r = 2$ [object] $\{1, 2\}$

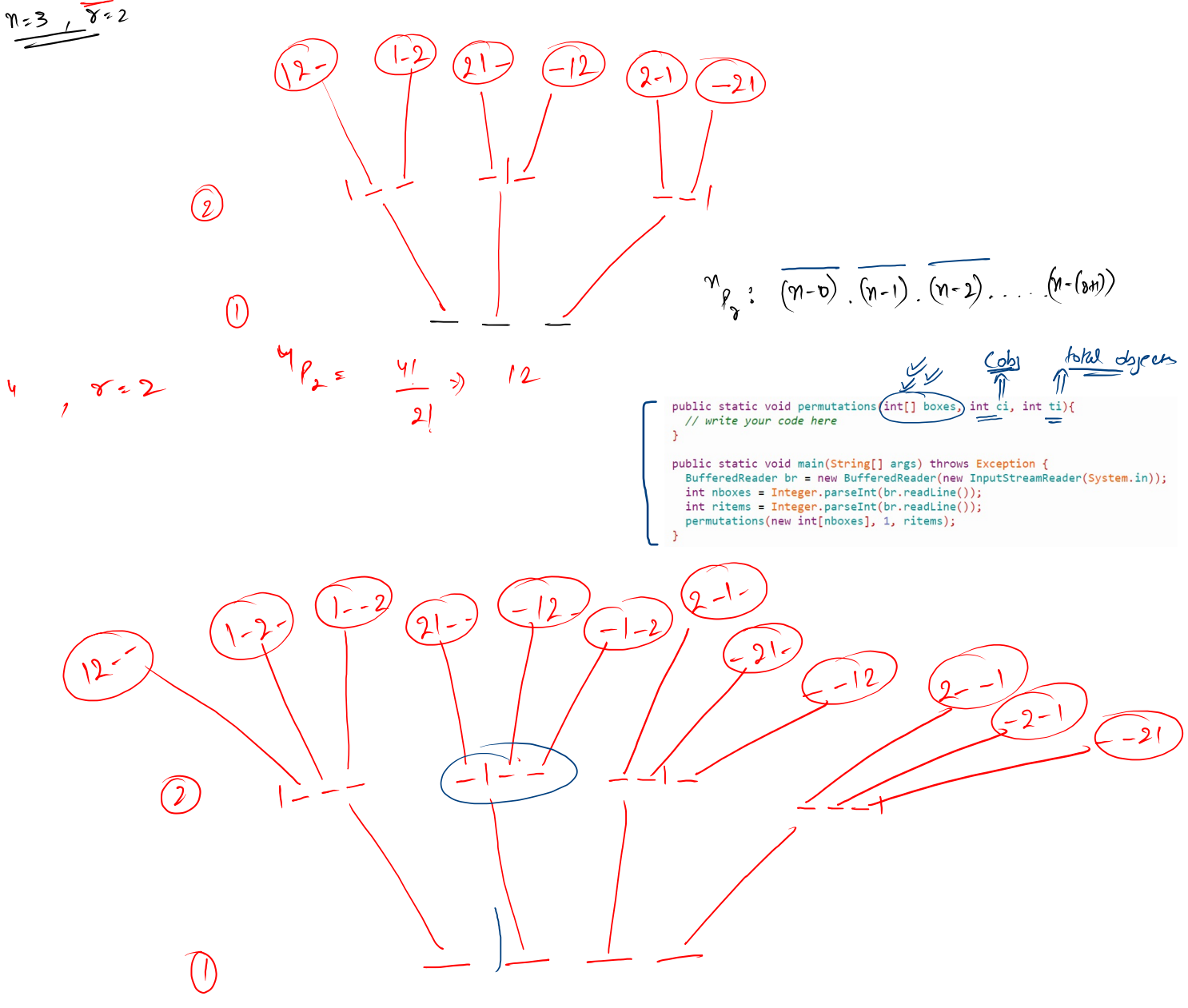
${}^3P_2 = \frac{3!}{1!} \Rightarrow 6$

Arrangement → Permutation

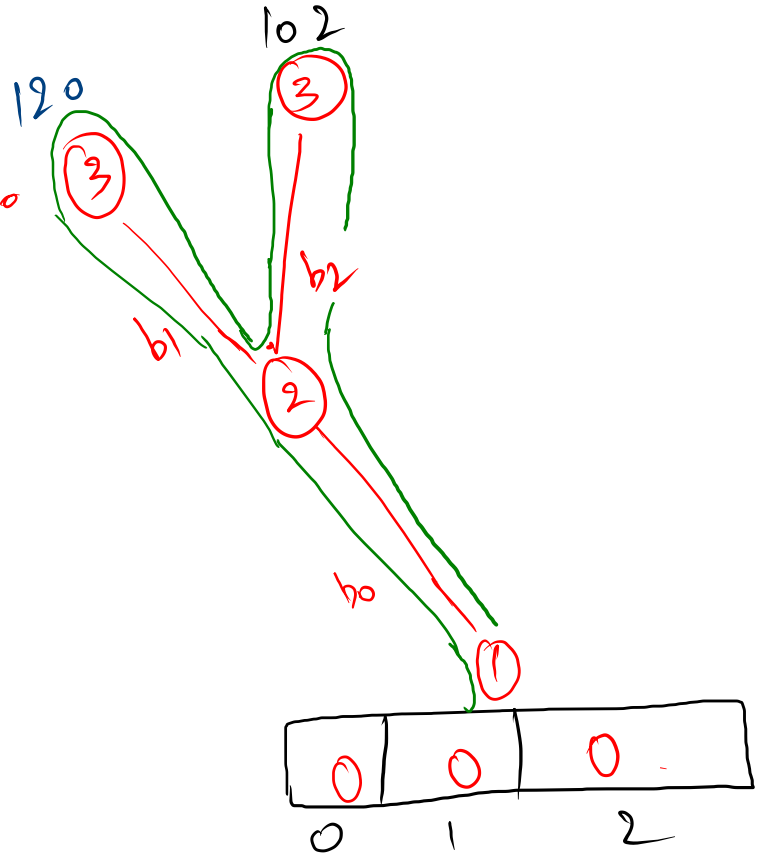
1 2 _	2 1 _
1 _ 2	2 _ 1
_ 1 2	_ 2 1

$${}^nP_r = \frac{n!}{(n-r)!} = \frac{n.(n-1).(n-2).(n-3)\ldots(n-(r+1))!.\cancel{(n-r)!}}{\cancel{(n-r)!}}$$

$$= n.(n-1).(n-2).(n-3)\ldots(n-r+1)$$


n=3 , r=2
12-
1-2
21-
-12
2-1
-21
2
1
nPr : (n-0) . (n-1) . (n-2) . . . . (n-(r-1))
n=4 , r=2
4P2 = 4!/2! => 12
Cobj
total objects
public static void permutations(int[] boxes, int ci, int ti){
// write your code here
}
public static void main(String[] args) throws Exception {
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
int nboxes = Integer.parseInt(br.readLine());
int ritems = Integer.parseInt(br.readLine());
permutations(new int[nboxes], 1, ritems);
}
12--
1-2-
1--2
21--
-12
-1-2
2-1-
-21-
--12
2--1
-2-1
--21
-1--
2
1
1

$n = 3$, $r = 2$




```java
public static void permutations(int[] boxes, int ci, int ti){
    if(ci > ti){
        for(int vl : boxes){
            System.out.print(vl);
        }
        System.out.println();
        return;
    }
    for(int bi = 0 ; bi < boxes.length ; bi++){
        if(boxes[bi] == 0){ // empty box
            boxes[bi] = ci; // object placed
            permutations(boxes,ci+1,ti);
            boxes[bi] = 0; // object unplaced
        }
    }
}
```

Permutation

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

$n = 3$, $r = 2$ [distinct $\Rightarrow$ 1, 2]

1 2 _	2 1 _
1 _ 2	2 _ 1
_ 1 2	_ 2 1

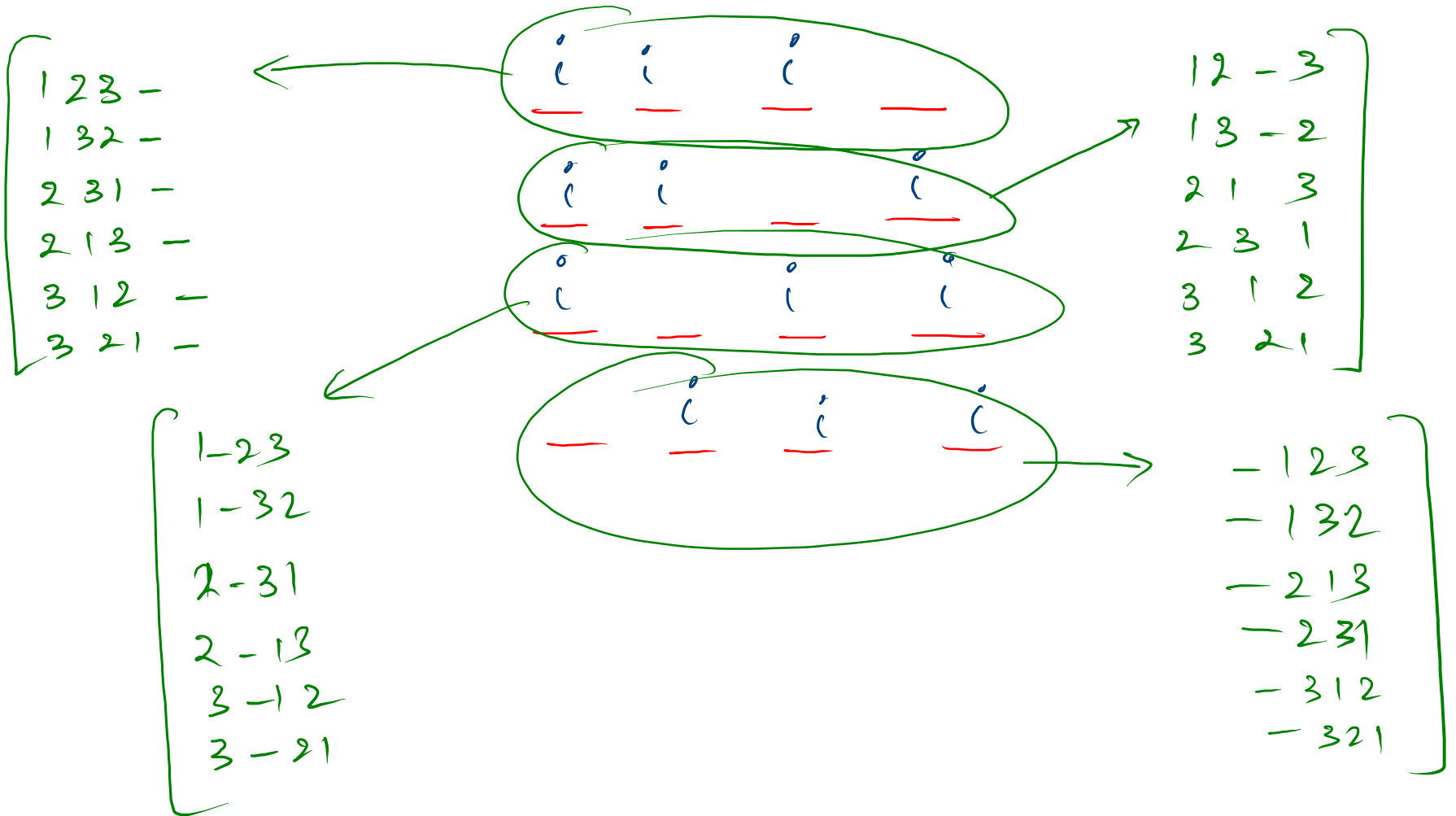
Combination

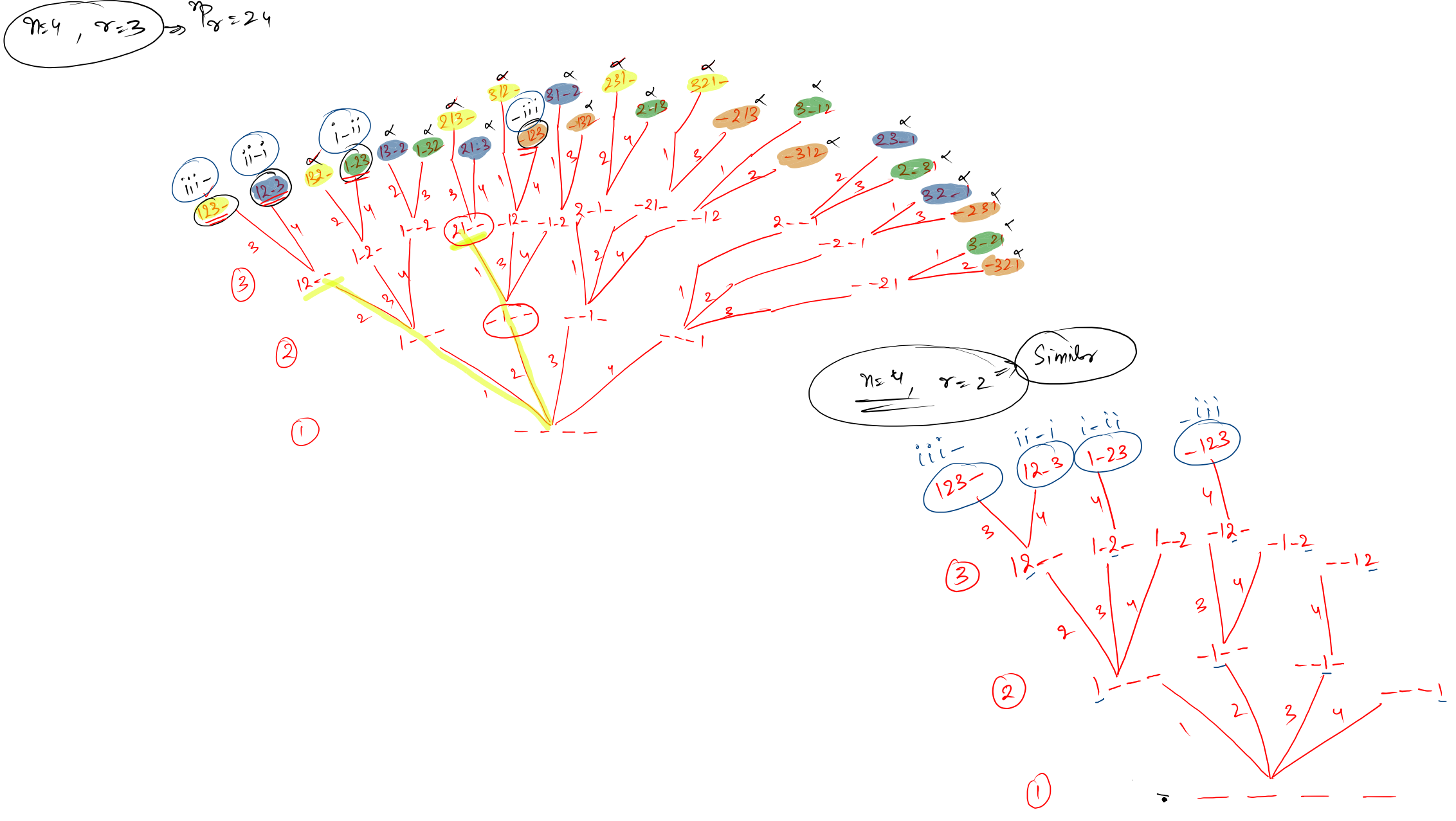
$$^{n}C_{r} = \frac{n!}{(n-r)!\,r!} = \frac{^{n}P_{r}}{r!}$$

$n = 3$, $r = 2$ [similar $\Rightarrow$ i, i]

i i	i _	_
i _	_	i
_	i	i

$n = 4$, $r = 2$, ${}^{n}P_r \Rightarrow {}^{4}P_3 = 24$, ${}^{n}C_r \Rightarrow {}^{4}C_3 = 4$




n=4, r=3 ⇒ nPr = 24
Similar
n=4, r=2

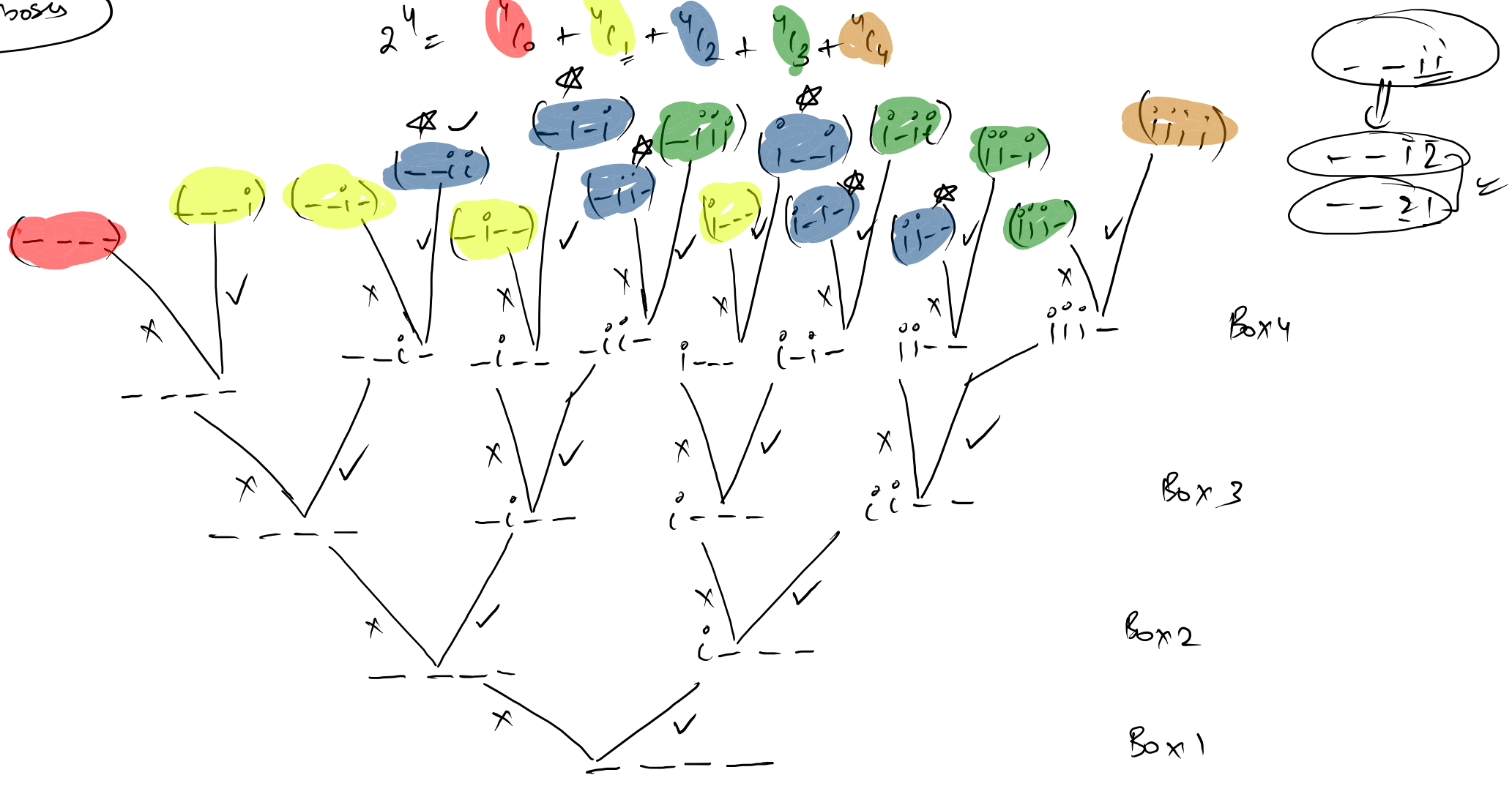
Combination

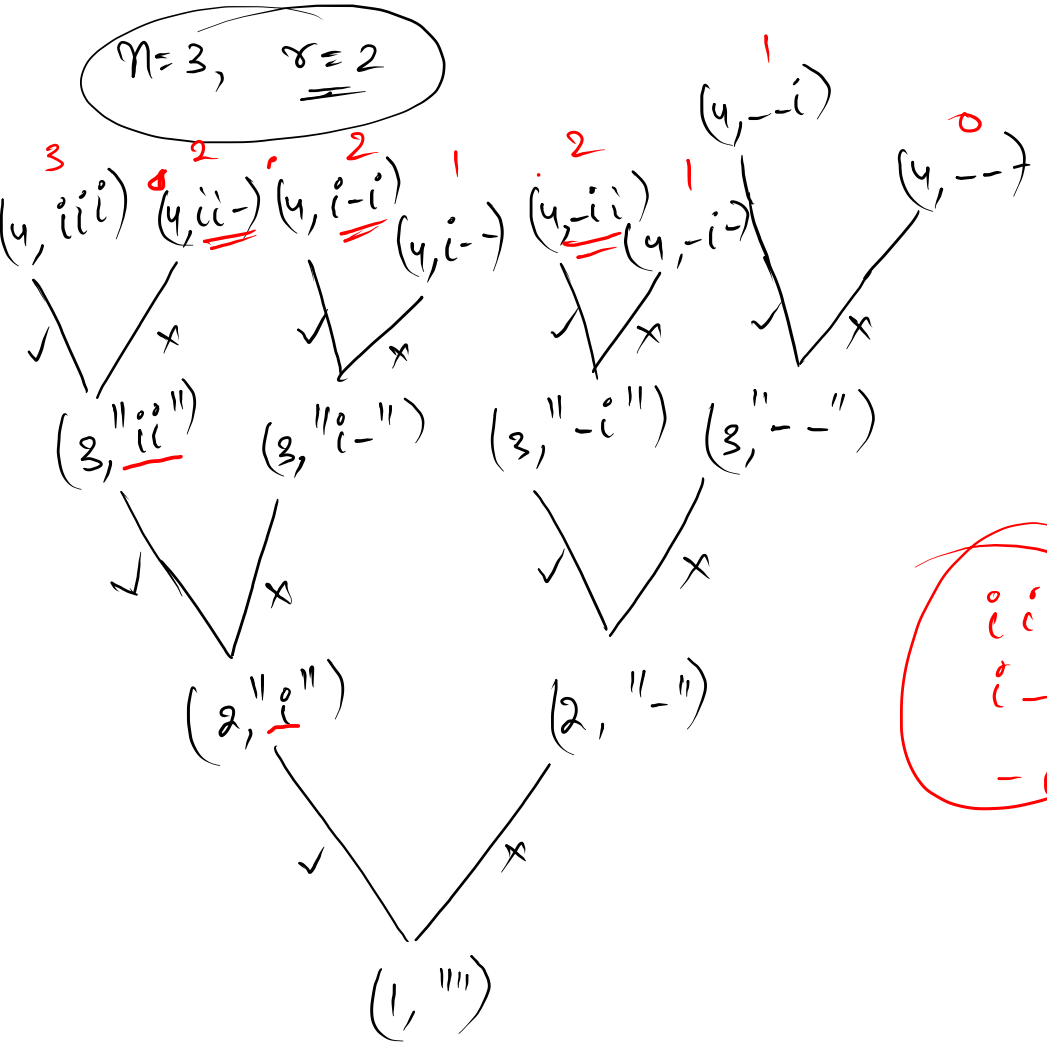
Box - chosy

$$2^n = {}^nC_0 + {}^nC_1 + {}^nC_2 + {}^nC_3 + \dots + {}^nC_n$$

$${}^4C_2 \Rightarrow 6$$

$$2^4 = {}^4C_0 + {}^4C_1 + {}^4C_2 + {}^4C_3 + {}^4C_4$$





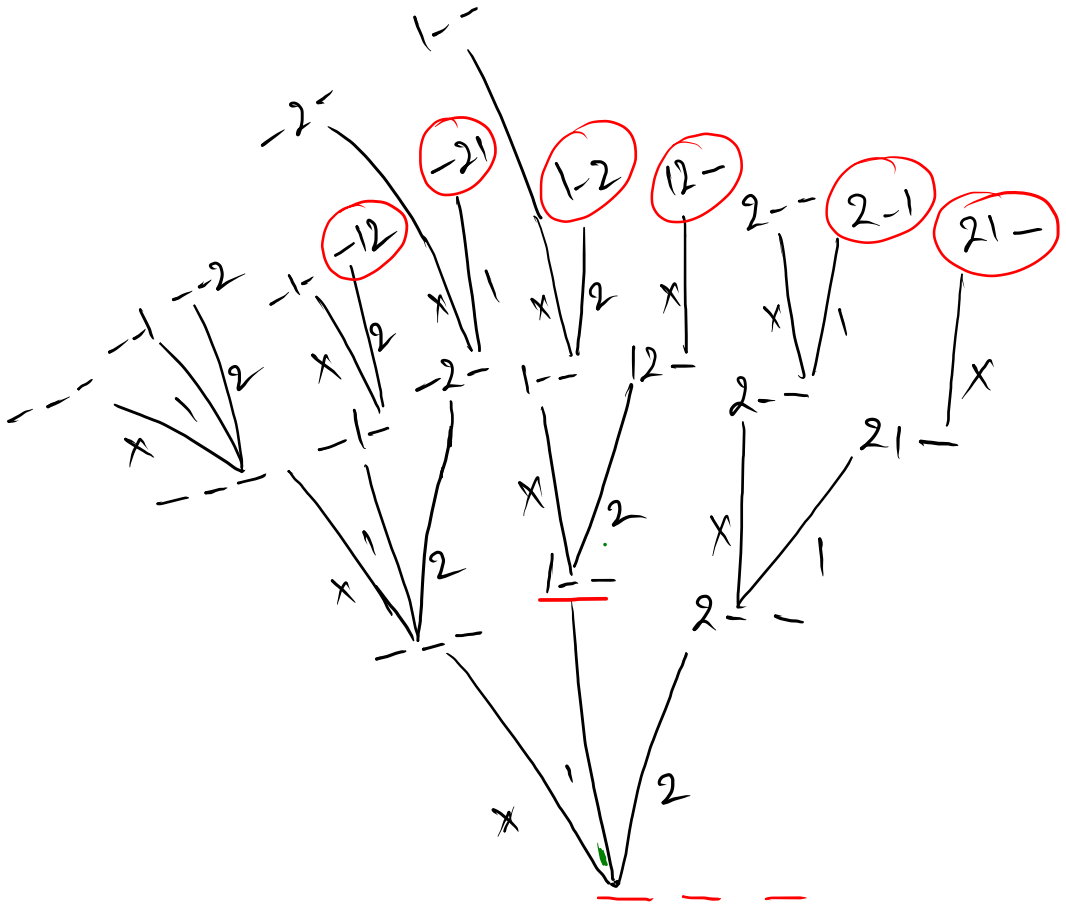
1 3 0 2 "iii"

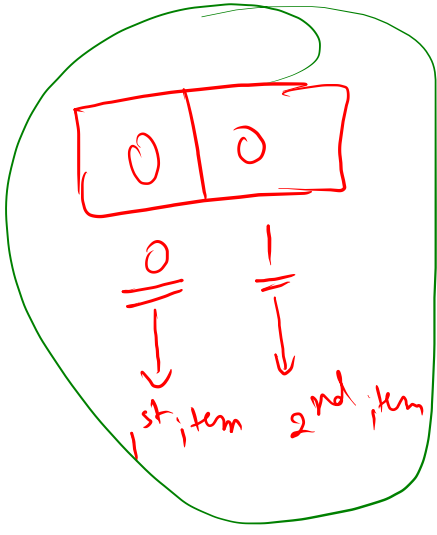
```
public static void combinations(int cb, int tb, int ssf, int ts, String asf){
  if(cb > tb){
    if(ssf == ts){
        System.out.println(asf);
    }
    return;
  }

  combinations(cb+1,tb,ssf+1,ts,asf+"i");
  combinations(cb+1,tb,ssf,ts,asf+"-");
}
```

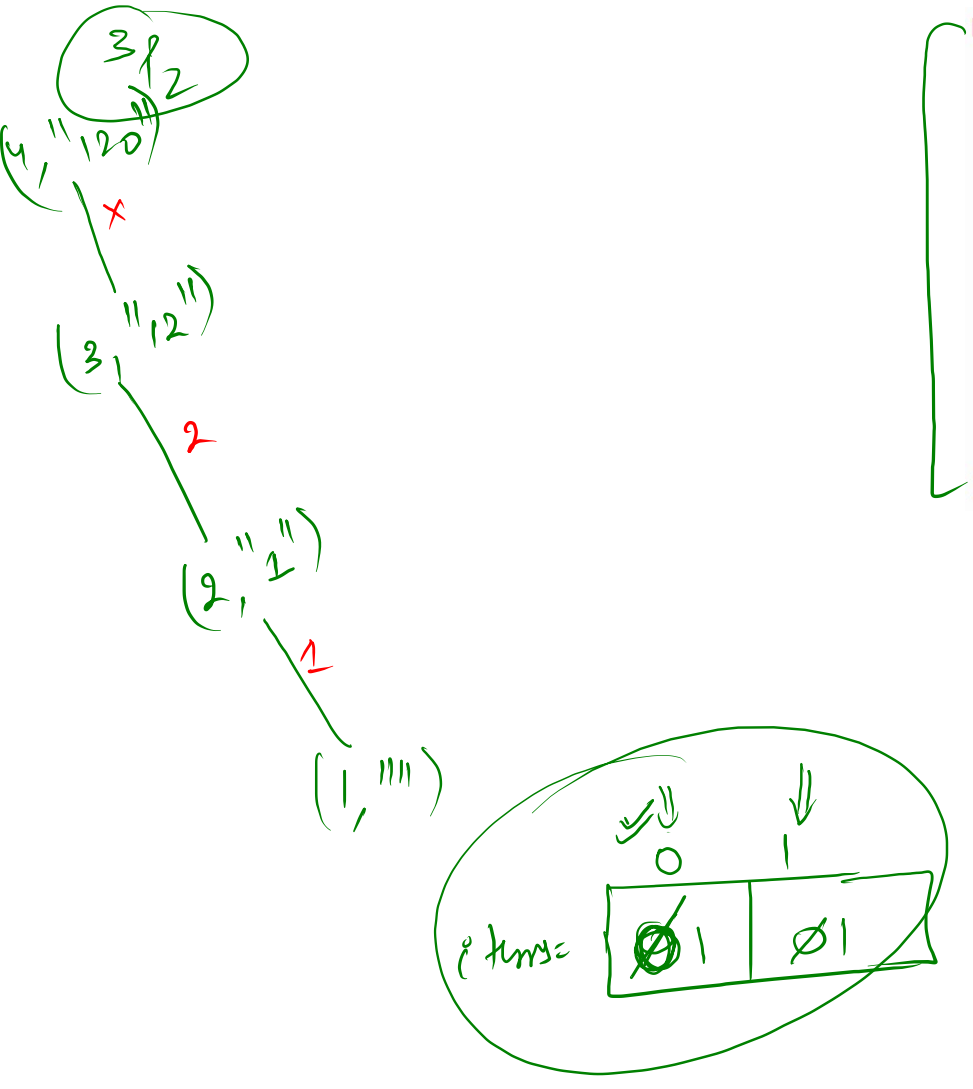
iii
ii-
i-i
-ii

Permutation - 2

$^3P_2$ →





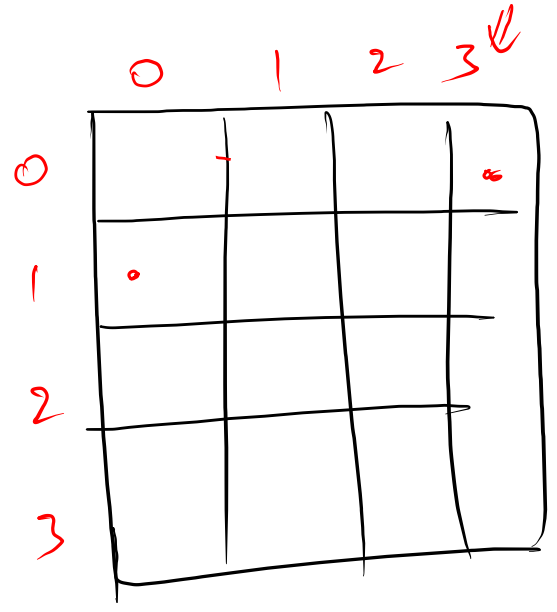



```java
public static void permutations(int cb, int tb, int[] items, int ssf, int ts, String asf){
    if(cb > tb){
        if(ssf == ts){
            System.out.println(asf);
        }
        return;
    }
    for(int i  = 0 ; i < items.length ; i++){
        if(items[i] == 0){ // object/item is unplaced
            items[i] = 1;
            permutations(cb+1,tb,items,ssf+1,ts,asf+(i+1));
            items[i] = 0;
        }
    }
    permutations(cb+1,tb,items,ssf,ts,asf+"0");
}
```

2

120

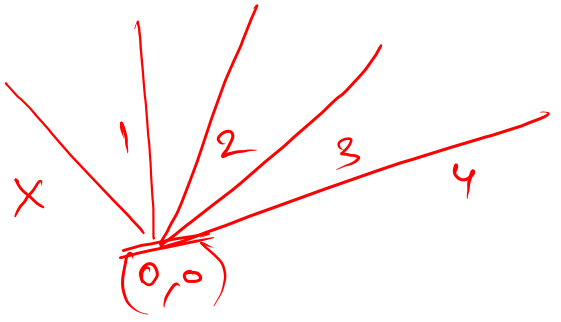
n = 4



4 queens

N queens

```
int nr,nc;
if(col < tq-1){
    nr = row;
    nc = col+1;
}else{
    nr = row+1;
    nc = 0;
}
```



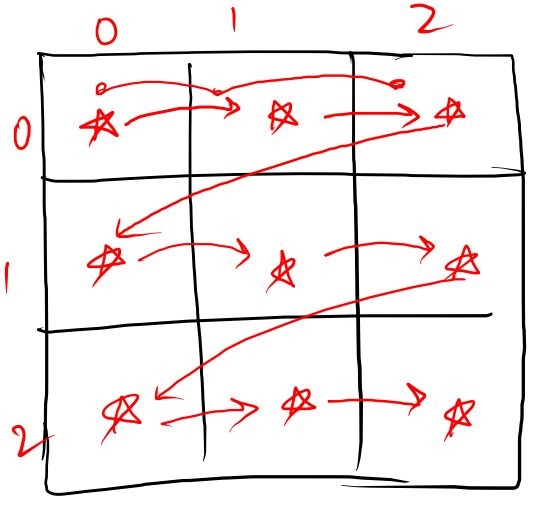

```
public static void queensPermutations(int qpsf, int tq, int row, int col, String asf, boolean[] queens) {
    // write your code here
}
```

n = 3

col = 1

tq = 3

tq - 1 = 2



row = 3

0 3 0 0

```
public static void queensPermutations(int qpsf, int tq, int row, int col, String asf, boolean[] queens) {

    int nr,nc;
    if(col < tq-1){
        nr = row;
        nc = col+1;
    }else{
        nr = row+1;
        nc = 0;
    }
    for(int i = 0 ; i < tq ; i++){
        if(queens[i] == false){
            queens[i] = true;
            queensPermutations(qpsf+1,tq,nr,nc,queens);
            queens[i] = false;
        }
    }
    queensPermutations(qpsf,tq,nr,nc,queens);
}
```

```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);
        }
            // System.out.println();
        return;
    }
    int nr,nc;
    String sep = "";
    if(col < tq-1){
        nr = row;
        nc = col+1;
        sep = "";
    }else{
        nr = row+1;
        nc = 0;
        sep = "\n";
    }
    queensCombinations(qpsf+1,tq,nr,nc,asf+"q"+sep);
    queensCombinations(qpsf,tq,nr,nc,asf+"-"+sep);
}
public static void main(String[] args) throws Exception {
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    int n = Integer.parseInt(br.readLine());

    queensCombinations(0, n, 0, 0, "");
}
```

```java
public static void queensPermutations(int qpsf, int tq, int row, int col, String asf, boolean[] queens) {
    if(row == tq){
        if(qpsf == tq){
            System.out.println(asf);
            System.out.println();
        }
        return;
    }
    int nr,nc;
    char sep;
    if(col == tq-1){
        nr = row+1;
        nc = 0;
        sep = '

    }else{
        nr = row;
        nc = col+1;
        sep = ' ';
    }

    for(int i = 0 ; i < queens.length ; i++){
        if(queens[i] == false){
            queens[i] = true;
            queensPermutations(qpsf+1,tq,nr,nc,asf+"q"+(i+1)+sep,queens);
            queens[i] = false;
        }
    }
    queensPermutations(qpsf,tq,nr,nc,asf+"-"+sep,queens);
}
```