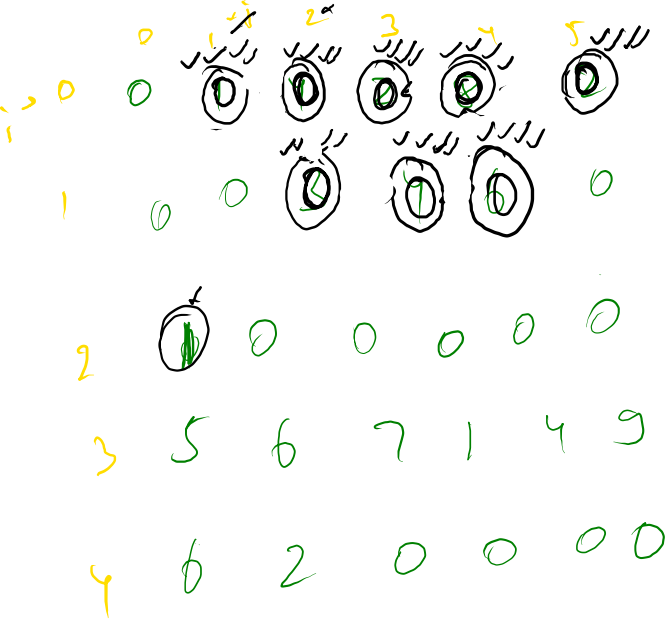
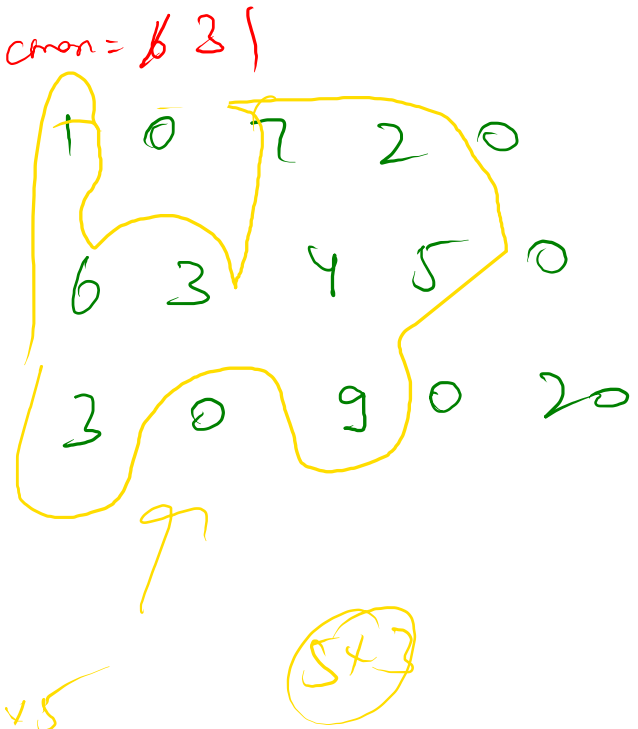
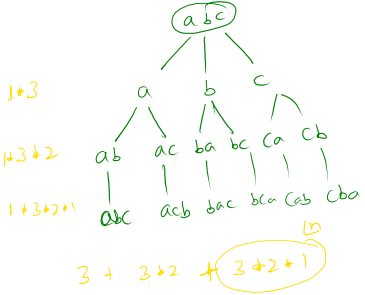


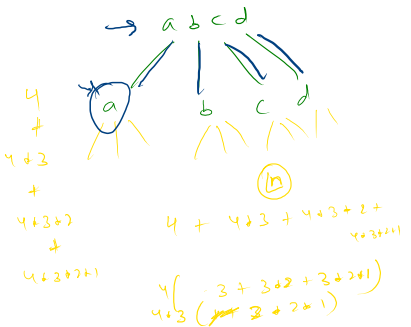
```
static int max = 0;
static int cmax = 0;
public static void getMaxGold(int[][] arr){
    //write your code here

    for(int i=0;i<arr.length;i++){
        for(int j=0;j<arr[0].length;j++){
            if(arr[i][j] != 0){
                traversal(arr,i,j);

                max = Math.max(max,cmax);
                cmax = 0;
            }
        }
    }
}
```

```
public static void traversal(int[][]arr,int row,int col){
    if(row < 0 || col < 0 || row == arr.length||col == arr[0].length||arr[row][col] == 
{
        return;
    }

    cmax = cmax + arr[row][col];
    arr[row][col] = 0;

    traversal(arr,row+1,col);
    traversal(arr,row,col+1);
    traversal(arr,row-1,col);
    traversal(arr,row,col-1);

}
```



Complexity

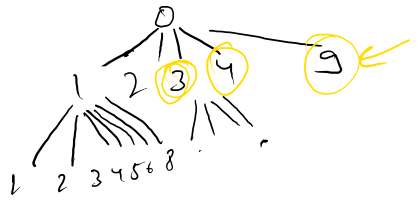




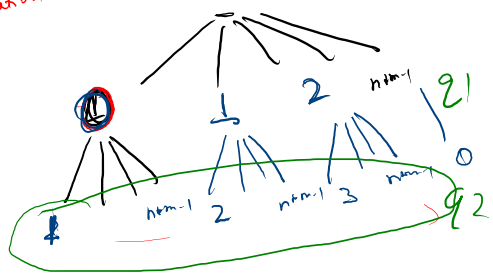
$O(n \cdot n)$

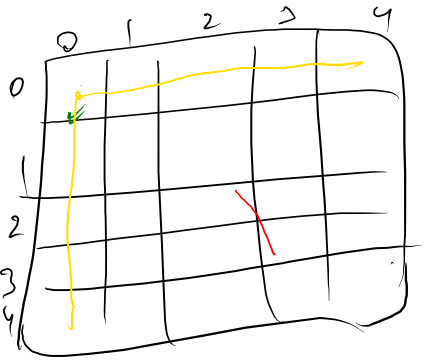
$O(9^{n^2} \cdot n)$

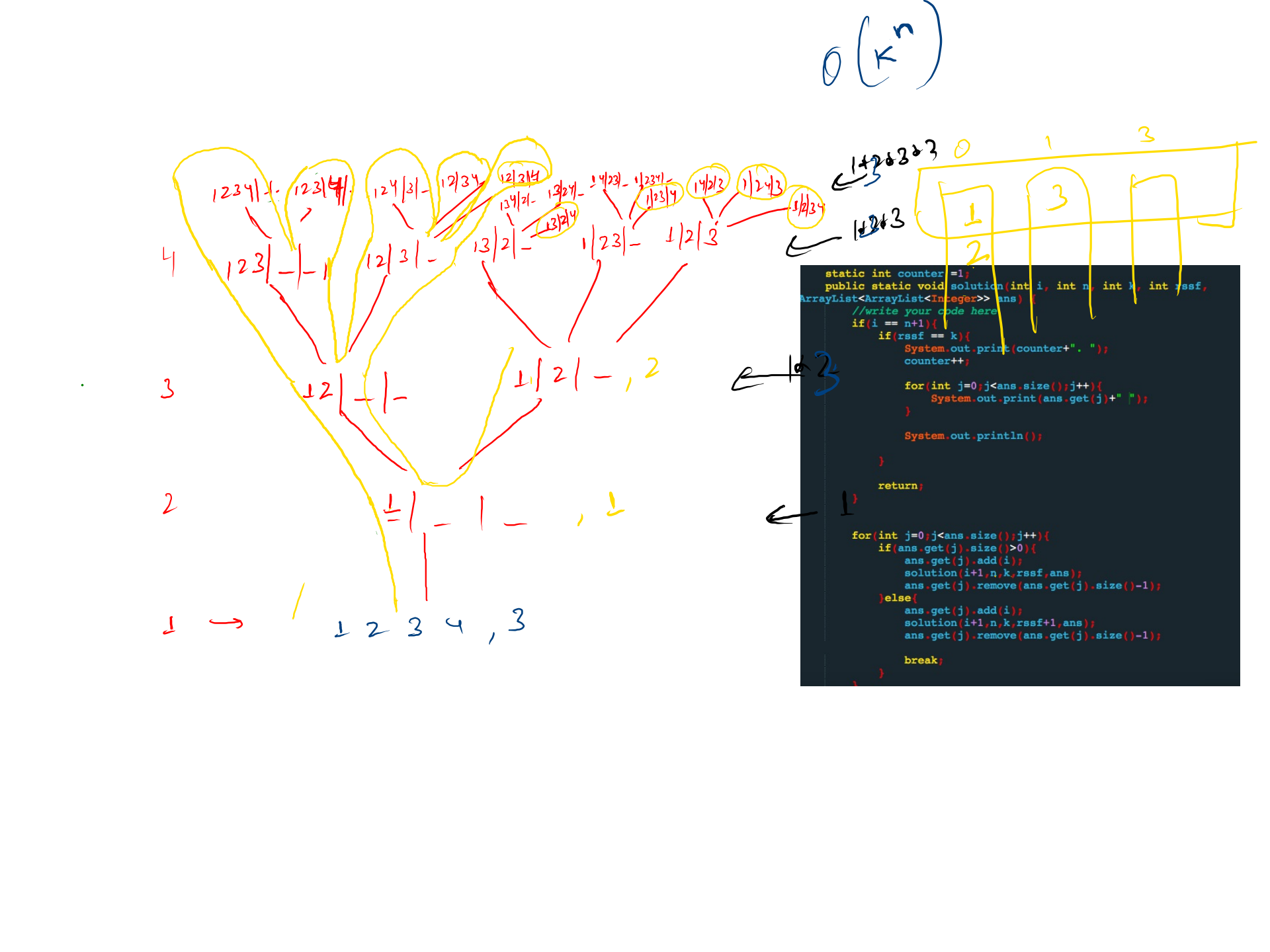
$O(9^{n^2} \cdot m)$

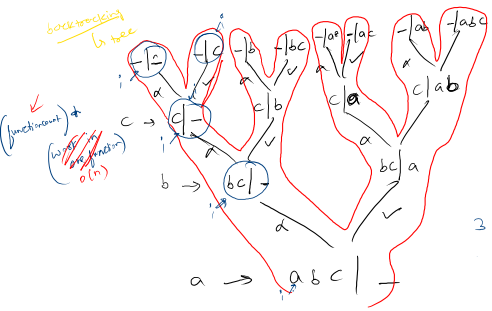


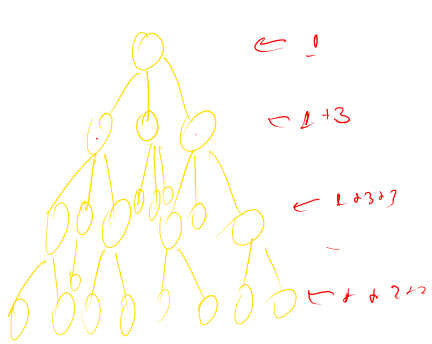
Permutation

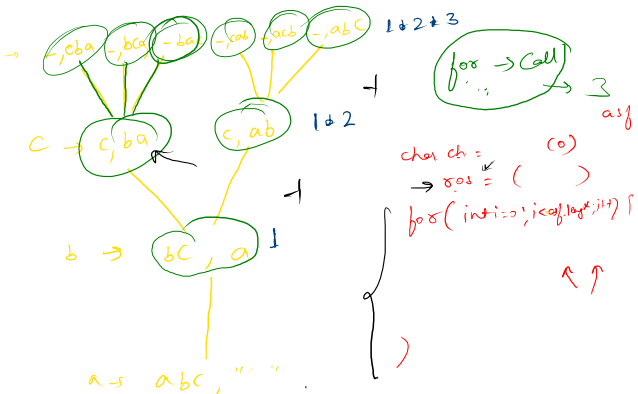





O(K^n)
1234|-|-
123|4|-
124|3|-
12|34
12|3|4
123|-|-|1
12|3|-
13|2|-
1|23|-
1|2|3
1|2|-,2
1|-|-|-,1
1 → 1 2 3 4 , 3
0 1 3
1 3
2
static int counter =1;
public static void solution(int i, int n, int k, int rssf,
ArrayList<ArrayList<Integer>> ans) {
//write your code here
if(i == n+1){
if(rssf == k){
System.out.print(counter+". ");
counter++;
for(int j=0;j<ans.size();j++){
System.out.print(ans.get(j)+" ");
}
System.out.println();
}
return;
}
for(int j=0;j<ans.size();j++){
if(ans.get(j).size()>0){
ans.get(j).add(i);
solution(i+1,n,k,rssf,ans);
ans.get(j).remove(ans.get(j).size()-1);
}else{
ans.get(j).add(i);
solution(i+1,n,k,rssf+1,ans);
ans.get(j).remove(ans.get(j).size()-1);
break;
}

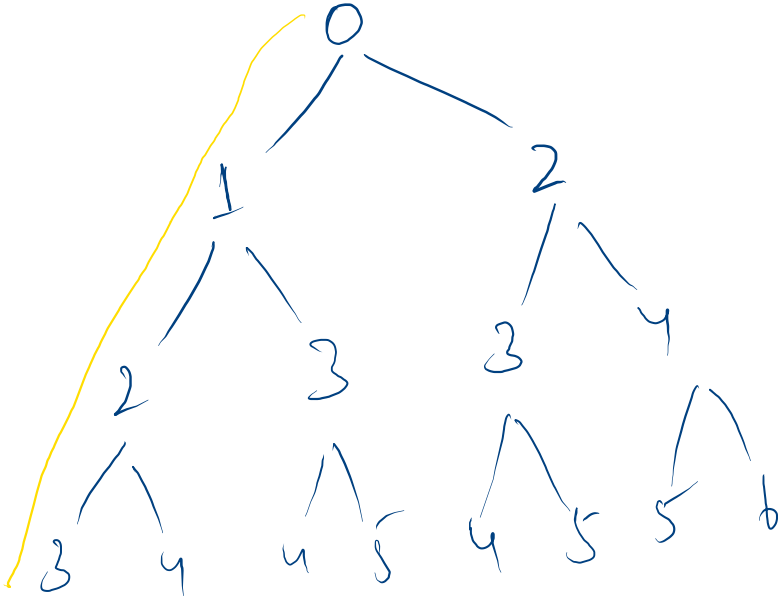






n*

O(2^n)

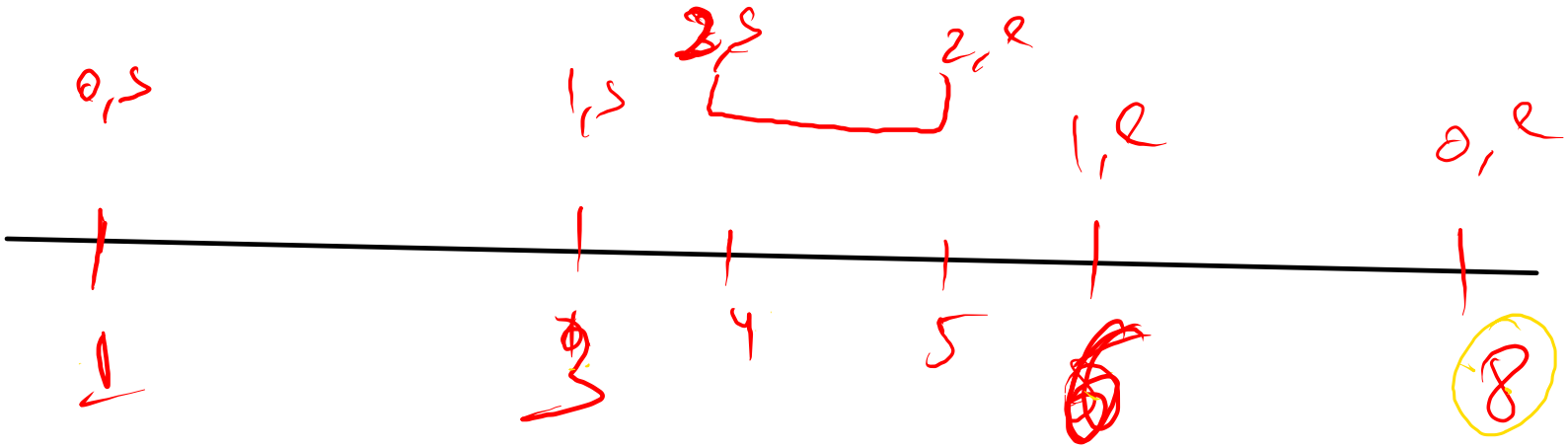


1+2 + 1+2+3 + 1+2+3+4

2(3 + 3+4)

2+3 (4) => 2+3+4

O(n) * (n)


0,s
1,s
2,s
2,e
1,e
0,e
1
3
4
5
6
8

2,s,4
1,s,3
0,s,1

0
1
2
-3
-1+3
1