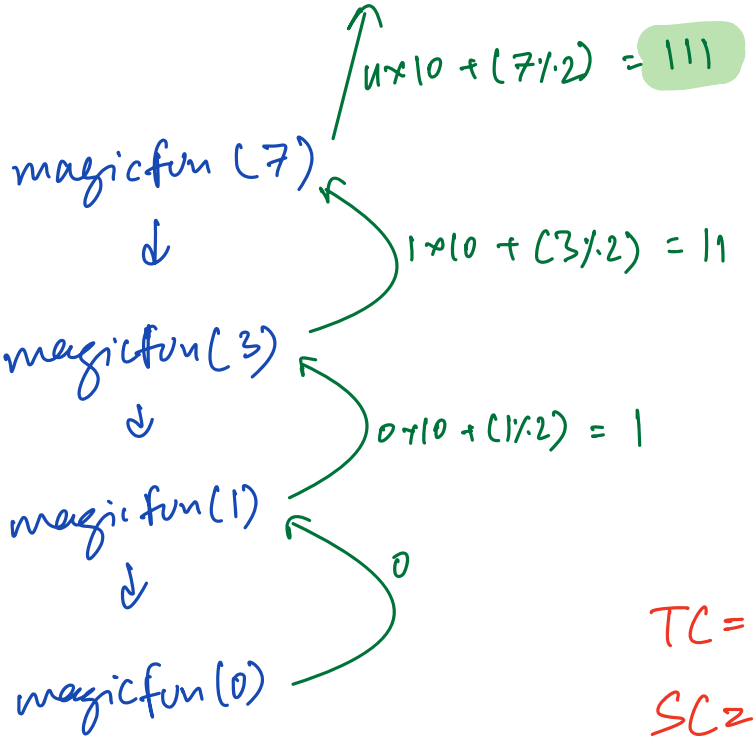
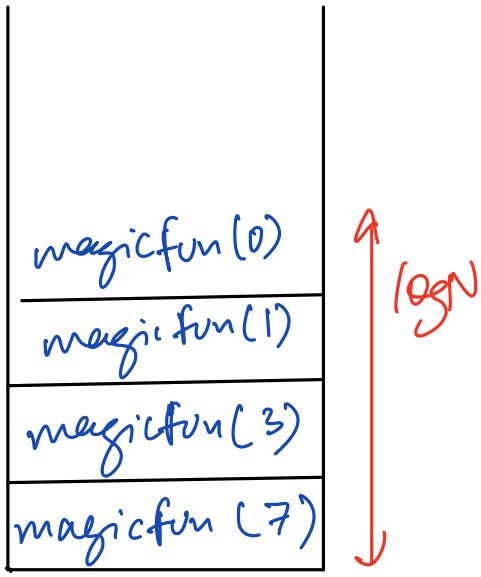
Backtracking

Quiz 1

N = 7

```
int magicfun(int N) {
    if (N == 0)  return 0
    else  return magicfun(N/2) * 10 + (N%2)
}
```





TC = O(logN)

SC = O(logN)

Quiz 2

fun("SCROLL", 0)

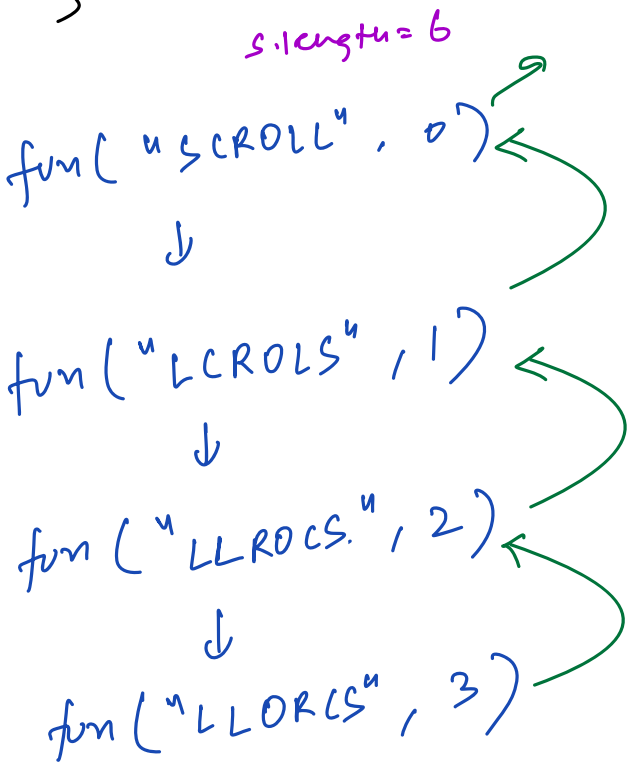
```
void fun(char s[], int x) {
    print(s)
    char temp
    if (x < s.length/2) {
        temp = s[x]
```

```
s[x] = s[s.length-x-1]

s[s.length-x-1] = temp

fun(s, x+1)

}
}
```



O/P :

SCROLL
LCROLS
LLROCS
LLORCS

TC = O(N)
SC = O(N)

Subarray vs Subset

arr[] = {1, 2, 3}

subarrays : [1] [2]
[1,2] [2,3]
[1,2,3] [3]

subsets :

[1] [2] [1,3]
[1,2] [2,3] []
[1,2,3] [3]

Question

Given an array. Print all of its subsets.

A = [1, 2, 3]

A[N] ⇒ 2^N subsets

o/p :

{ }

{1}

{1,2}

{1,3}

{2,3}

{2}

{3}

{1,2,3}

A = [1, 2, 3, 4]

o/p :

{ }

{1}

{1,2}

{1,3}

{2,3}

{2}

{3}

{4}

{1,4}

{1,2,4}

{1,3,4}

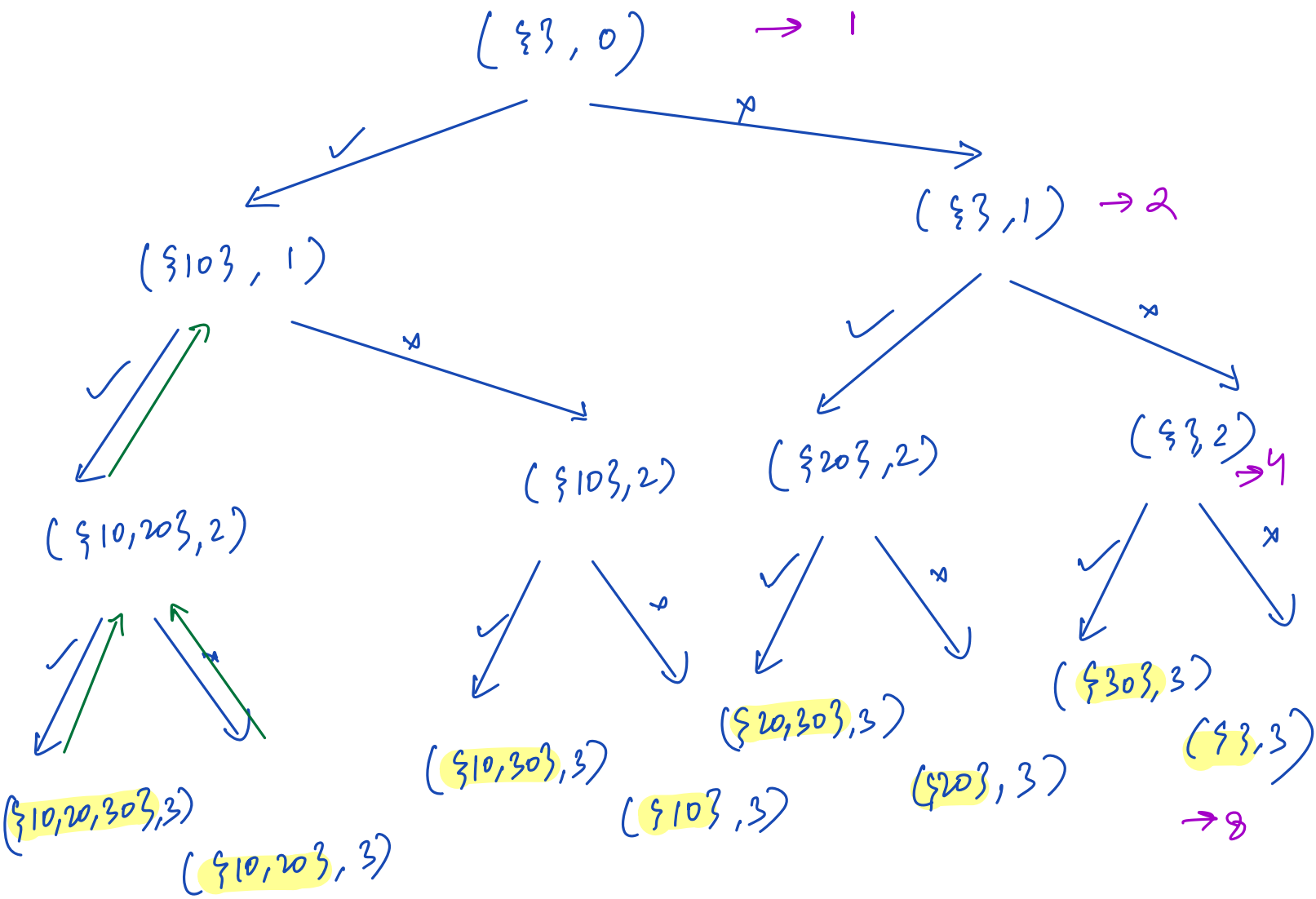
{2,3,4}

{2,4}

{3,4}

{1,2,3} {1,2,3,4}

arr[] → {10, 20, 30} (0, 1, 2)



total calls = $2^0 + 2^1 + 2^2 + 2^3 + \ldots + 2^N$

$= 2^{N+1} - 1 = O(2^N)$

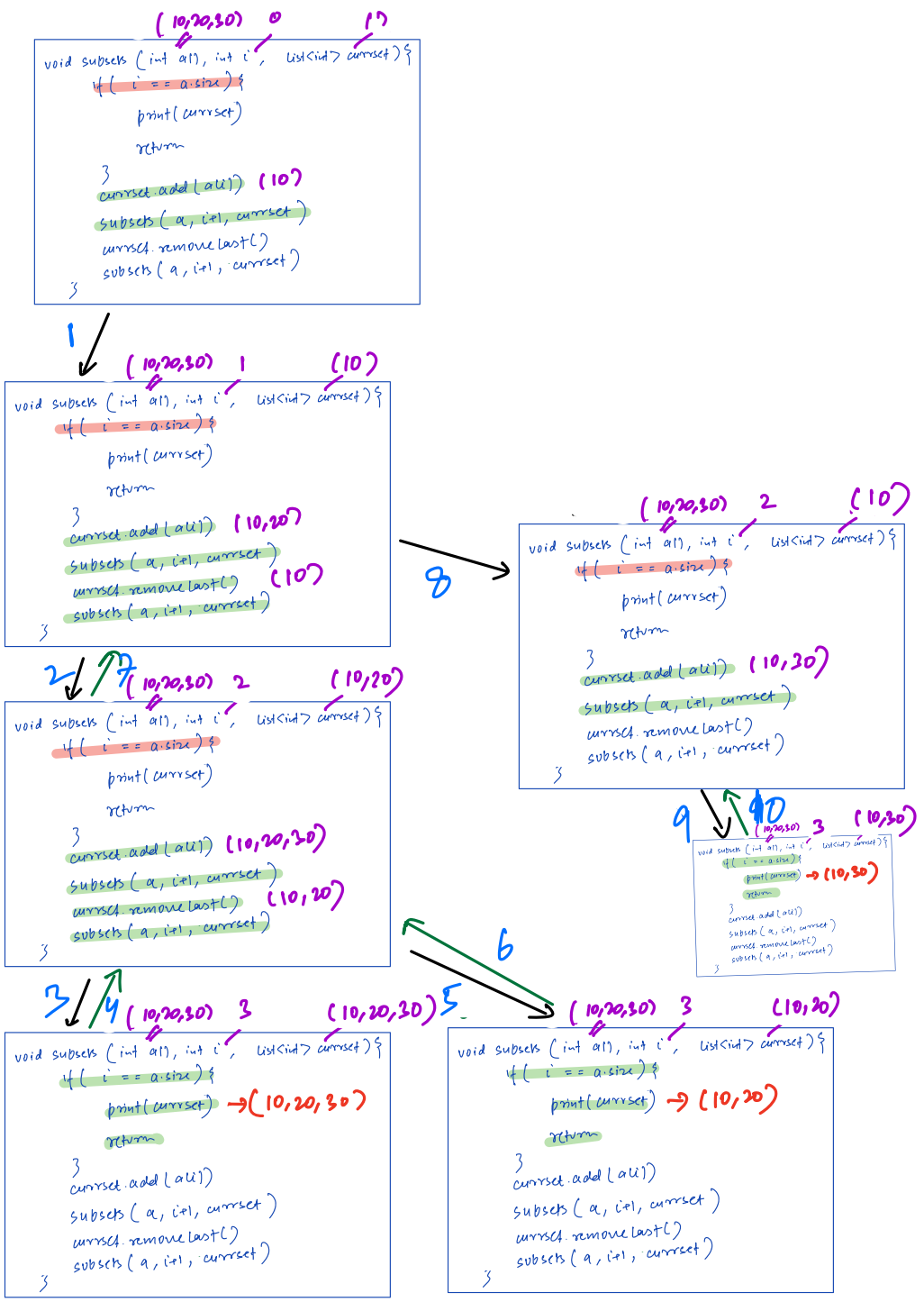
$= 2 \cdot 2^N$

subsets(a, 0, {})

```
void subsets (int a[], int i, List<int> currset) {
    if ( i == a.size ) {
        print(currset)
        return
    }
    // 2 options
    // 1. Pick a[i] in the subset
        currset.add(a[i])
        subsets ( a, i+1, currset )
        currset.removeLast()   → property of backtracking
    // 2. Don't Pick a[i]
        subsets ( a, i+1, currset )
}
```

$TC = O(2^N)$

$SC = O(N)$



Question

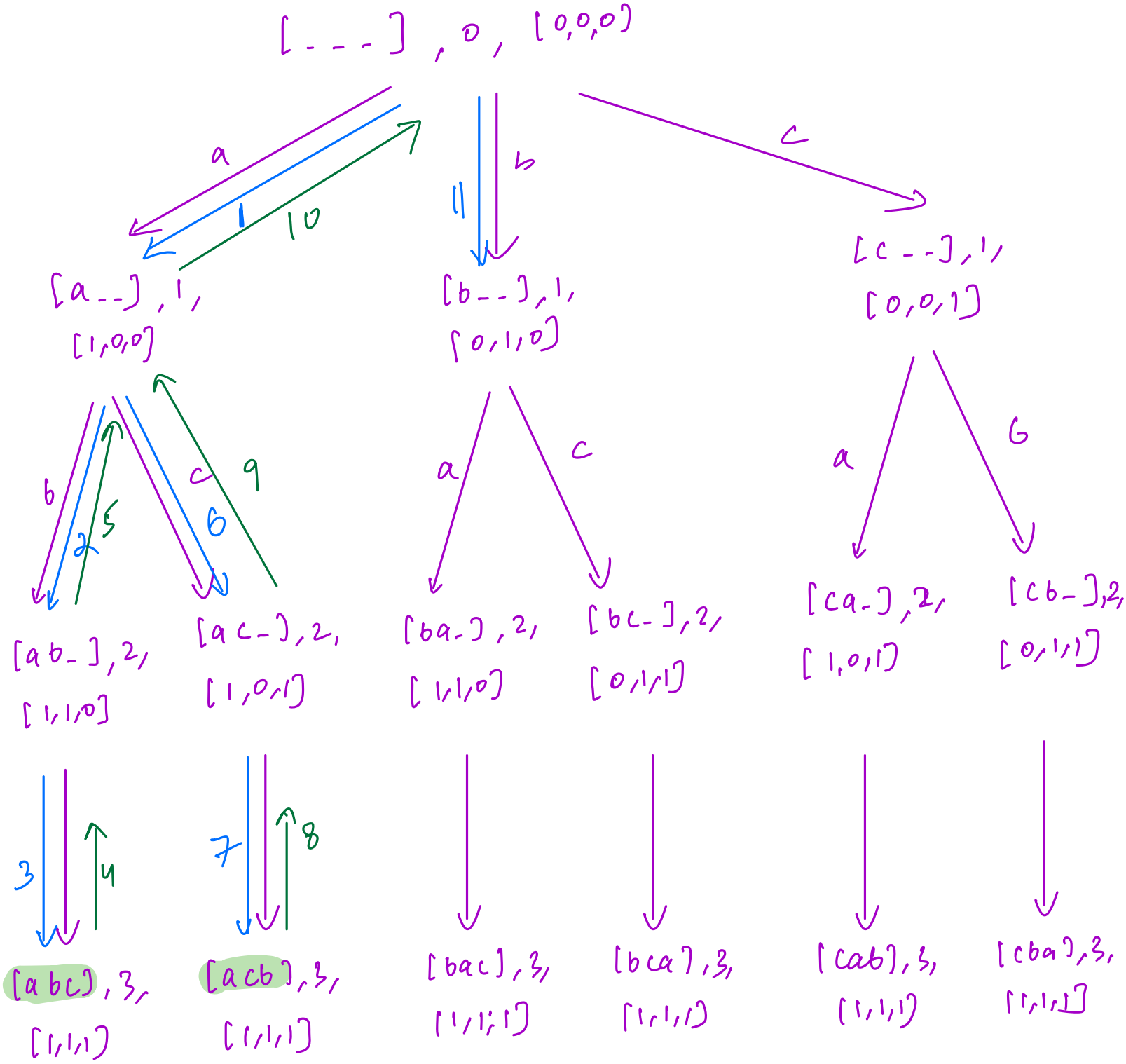
Given a string with distinct characters, print all permutations.

abc →

abc
acb
bac
bca
cab
cba

string of size N → N! permutations.

Inp: a b c (0 1 2)



```
void permutations( str, i, ans, vis ) {        // vis: type bool
    if( i == str.size() ) {
        print( ans )
        return
    }
    for( j=0 to n-1 ) {                        // n = vis.size()
        if( vis[j] == false ) {
            ans[i] = str[j]
            vis[j] = true
            permutations( str, i+1, ans, vis )
            vis[j] = false
            ~~ans[i] = '\0'~~                  not needed
        }
    }
}
```

permutations("abc", 0, [...], [0,0,0])

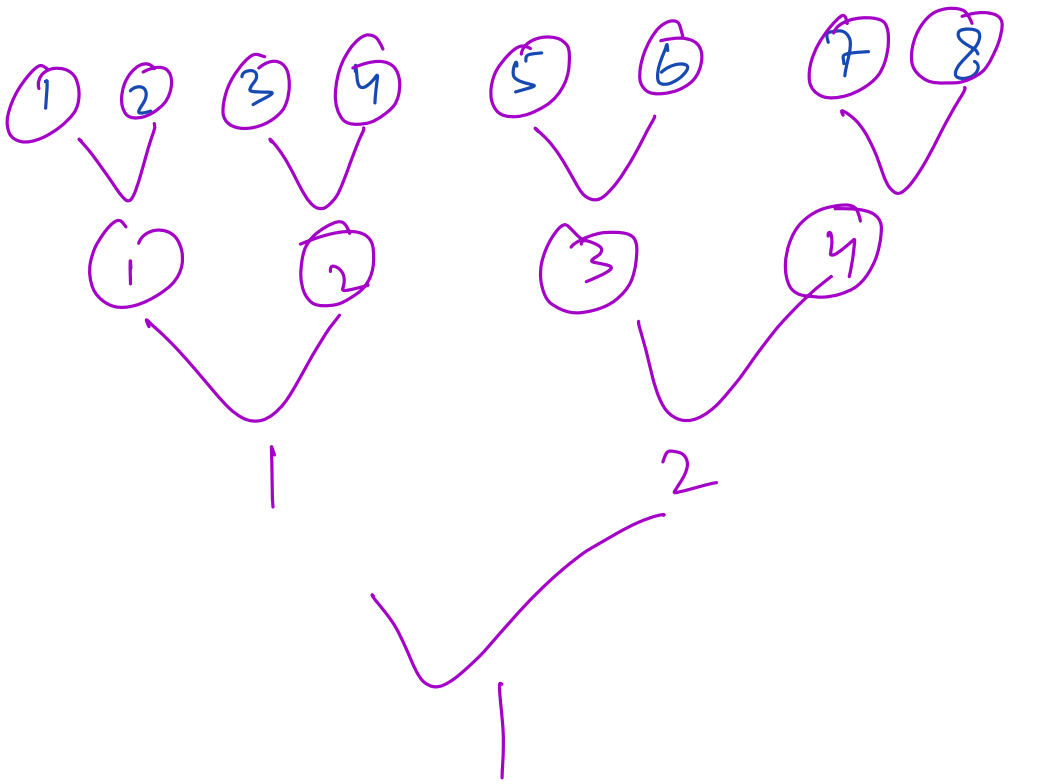
TC → total func. calls × TC of one func. call

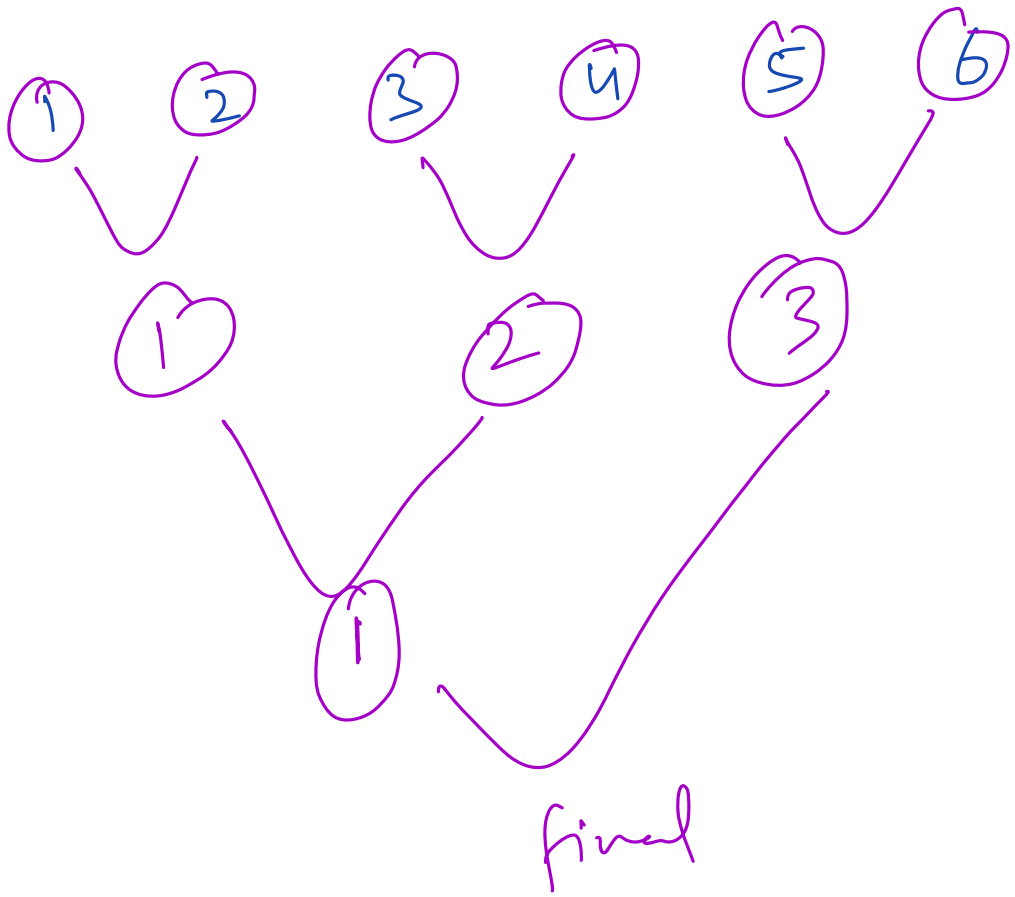
↓

N! × O(N)

TC = O(N × N!) ⇒ O((N+1)!)

SC = O(N)


1
2
3
4
5
6
7
8
1
2
3
4
1
2
1


1
2
3
4
5
6
1
2
3
1
final