Recursion With ArrayLists { Get Variation }

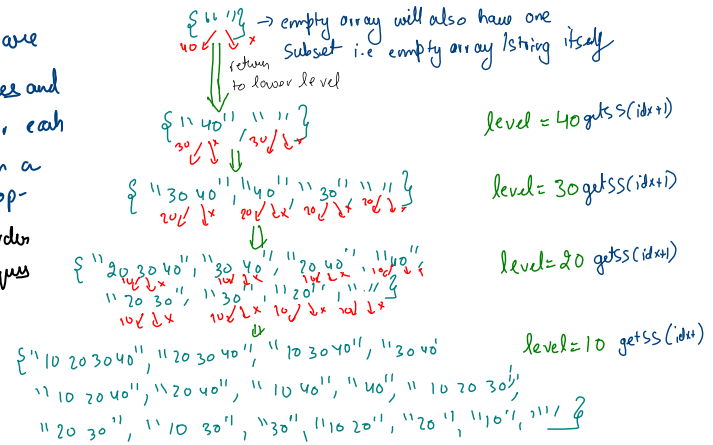
abc

[, c, b, bc, a, ac, ab, abc]

** Recursion tree levels ⇔ code / func parameter

{10, 20, 30, 40} level → Item / No
options → include / exclude

A<S> getSS(idx, arr) ↝ Expectation
A<S> getSS(idx+1, arr) ↝ faith (idx+1 to end elements will fill an arraylist of their subsets & then return it)



Here, we are applying yes and no call for each element in a single loop.
To maintain order as given in ques Apply 2 loops first for "NO" Second for "YES" calls.

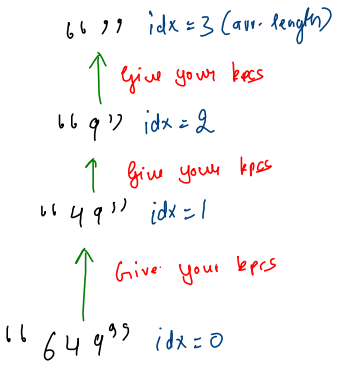
In getSS, the smallest problem will lie in the base case. and largest on the level 0 / main method / calling method.
↓
all get 0's

** This is the main reason that we can apply DP in get categories and not that efficiently in print categories.

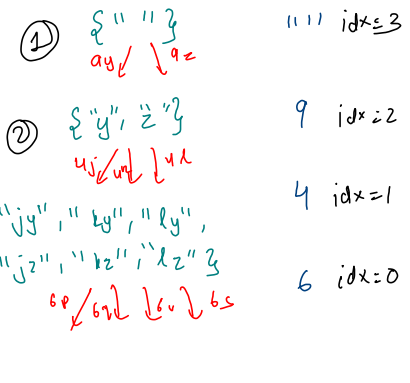
```java
public static ArrayList<String> gss(int idx,String str) {
    if(idx == str.length()) {
        ArrayList<String> bres = new ArrayList<>();
        bres.add("");
        return bres;
    }

    ArrayList<String> smallAns = gss(idx + 1, str);

    ArrayList<String> ans = new ArrayList<>();

    for(String s: smallAns) {
        ans.add(s);
    }

    for(String s: smallAns) {
        ans.add(str.charAt(idx) + s);
    }

    return ans;
}
```

Get KPC

```
0 -> .;
1 -> abc
2 -> def
3 -> ghi
4 -> jkl
5 -> mno
6 -> pqrs
7 -> tu
8 -> vwx
9 -> yz
```




```
0 -> .;
1 -> abc
2 -> def
3 -> ghi
4 -> jkl
5 -> mno
6 -> pqrs
7 -> tu
8 -> vwx
9 -> yz
```

649
4 x 3 x 2 = 24

⑤

24

```java
public static ArrayList<String> getKPC(String str,int idx) {

    if(idx == str.length()) {
        ArrayList<String> base = new ArrayList<>();
        base.add("");
        return base;
    }

    ArrayList<String> ans = new ArrayList<>();
    ArrayList<String> smallAns = getKPC(str,idx + 1);
    char ch = str.charAt(idx);

    String strForNum = keys[ch- '0'];

    for(int i=0;i<strForNum.length();i++) {

        for(String s: smallAns) {
            ans.add(strForNum.charAt(i) + s);
        }
    }

    return ans;
}
```

```java
ArrayList<String> ans = new ArrayList<>();
for(Character letter: dtoc[str.charAt(idx) - '0'].toCharArray()){
    for(String str: smallAns){
        res.add
```

for applying for-each loop on each character of a String convert it to character array.

Get Stair Paths

(only n-1, n-2 & n-3 jumps allowed)

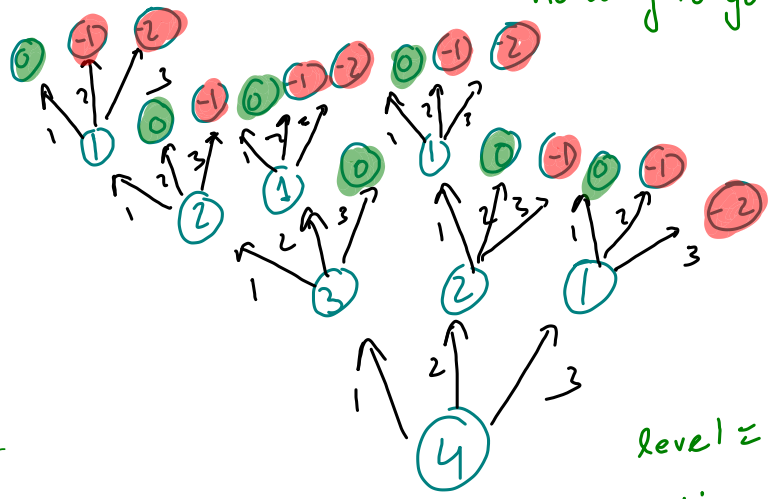
0-0 → true base case
one way i.e stay there only

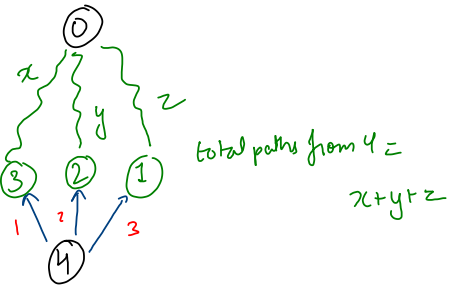
-1 to 0 → -ve base case
no way to go back to 0.

AKS gsp(n)

source ⇒ n

dest ⇒ 0th





level ⇒ curr stair

options ⇒ moves

```java
public static ArrayList<String> getStairPaths(int n) {

    if(n == 0) {
        ArrayList<String> base= new ArrayList<>();
        base.add("");
        return base;
    }

    if(n < 0) {
        return new ArrayList<>();
    }

    ArrayList<String> smallAns1 = getStairPaths(n - 1);
    ArrayList<String> smallAns2 = getStairPaths(n - 2);
    ArrayList<String> smallAns3 = getStairPaths(n - 3);

    ArrayList<String> ans = new ArrayList<>();

    for(String s: smallAns1) {
        ans.add("1" + s);
    }

    for(String s: smallAns2) {
        ans.add("2" + s);
    }

    for(String s: smallAns3) {
        ans.add("3" + s);
    }

    return ans;
}
```