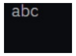
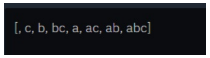
Recursion With ArrayLists { Get Variation }


abc


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

** Recursion tree levels ⇔ code / func parmeter

{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)

{ "" } → empty array will also have one subset i.e empty array / string itself
40↙ ↘x
return to lower level

{ "40", "" }   level = 40 getss(idx+1)
30↙ ↘x 30↙ ↘x

{ "30 40", "40", "30", "" }   level = 30 getss(idx+1)
20↙ ↘x 20↙ ↘x 20↙ ↘x 20↙ ↘x

{ "20 30 40", "30 40", "20 40", "40", "20 30", "30", "20", "" }   level = 20 getss(idx+1)
10↙ ↘x 10↙ ↘x 10↙ ↘x 10↙ ↘x 10↙ ↘x 10↙ ↘x 10↙ ↘x 10↙ ↘x

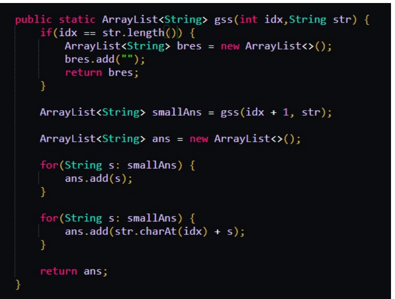
{ "10 20 30 40", "20 30 40", "10 30 40", "30 40", "10 20 40", "20 40", "10 40", "40", "10 20 30", "20 30", "10 30", "30", "10 20", "20", "10", "" }   level = 10 getss(idx+)

Here, we are applying yes and no call for each element in a single loop-

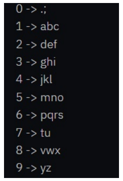
To maintain order as given in ques Apply 2 loops First for "No" Second for "YES" calls.

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

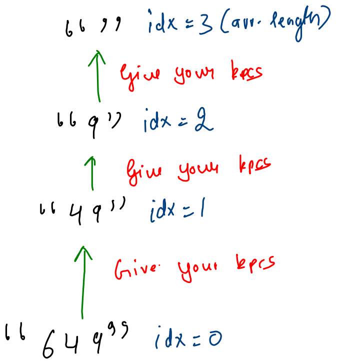
** This is the main reason that we can apply DP in get categories and not that efficiently in print categories.

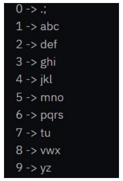

```java
public static ArrayList<String> gss(int idx,String str) {
    if(idx == str.length()) {
        ArrayList<String> bres = new ArrayList<>();
        bres.add("");
        return bres;
    }

    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

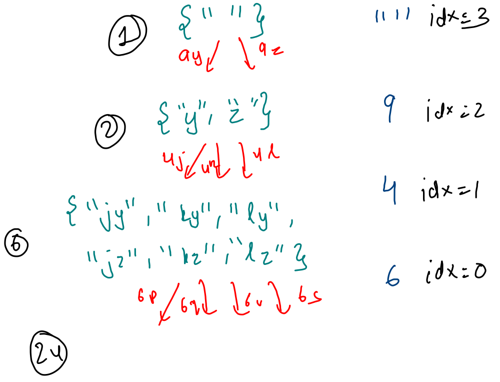


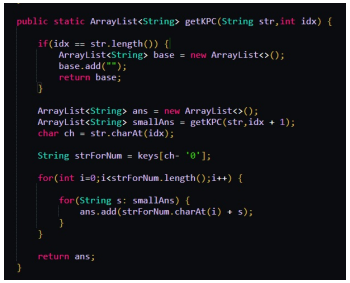


0	->	.;
1	->	abc
2	->	def
3	->	ghi
4	->	jkl
5	->	mno
6	->	pqrs
7	->	tu
8	->	vwx
9	->	yz

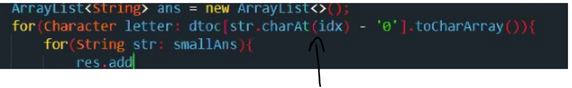
6 4 9

4×3×2 = 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)

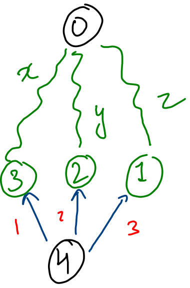
A<S> grs(n)

source => n

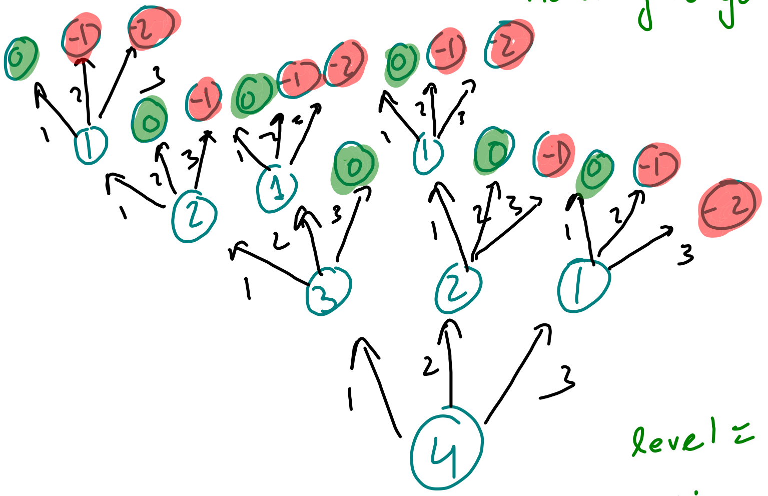
dest => 0th

0-0 -> +ve base case
one way i.e stay there only

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

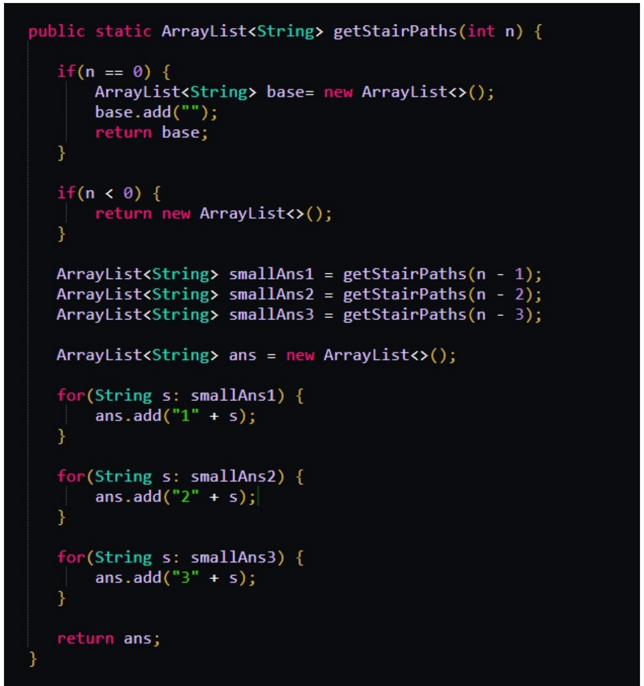


total paths from 4 = x+y+z

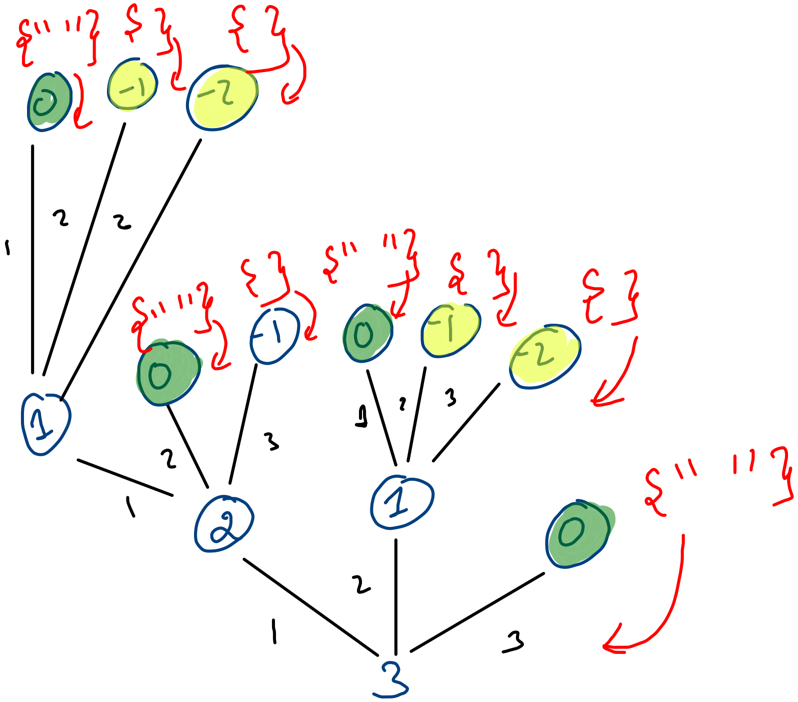


level = curr stair

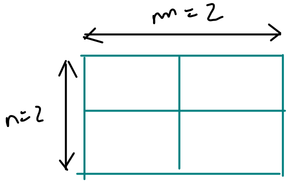
options = moves



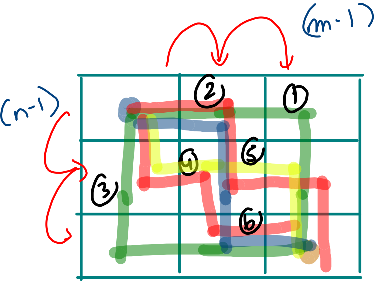
```
public static ArrayList<String> getStairPaths(int n) {

    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;
}
```



Get Maze Paths



Only 1 sized horizontal and 1 sized vertical in right & down directions respectively are possible. (ATQ)



① r r d d

② r d r d

③ d d r r

④ d r d r

⑤ d r r d

⑥ r d d r

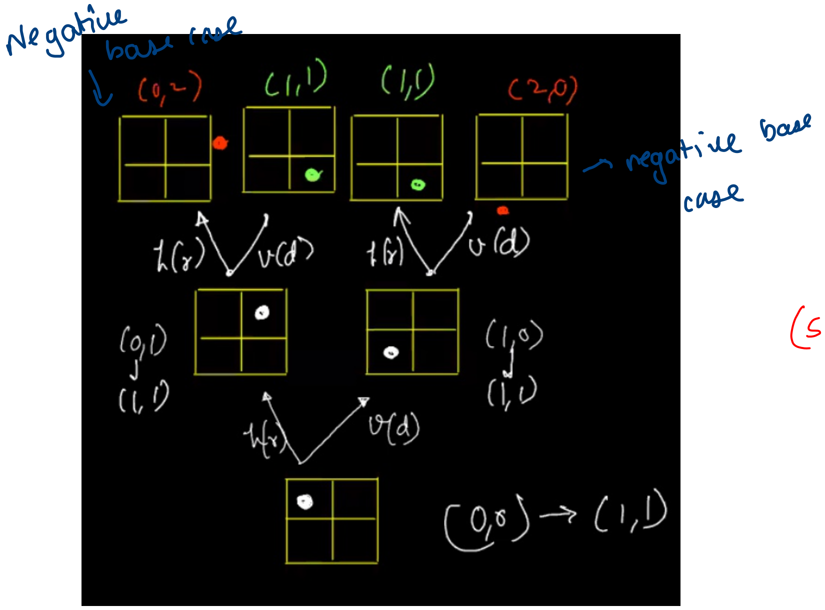
A L<S> gMP (sr, sc, dr, dc)

Path Length: (n-1)+(m-1)
↳ This is greedy

No of Paths:
↳ This will be DP
(catalan No)

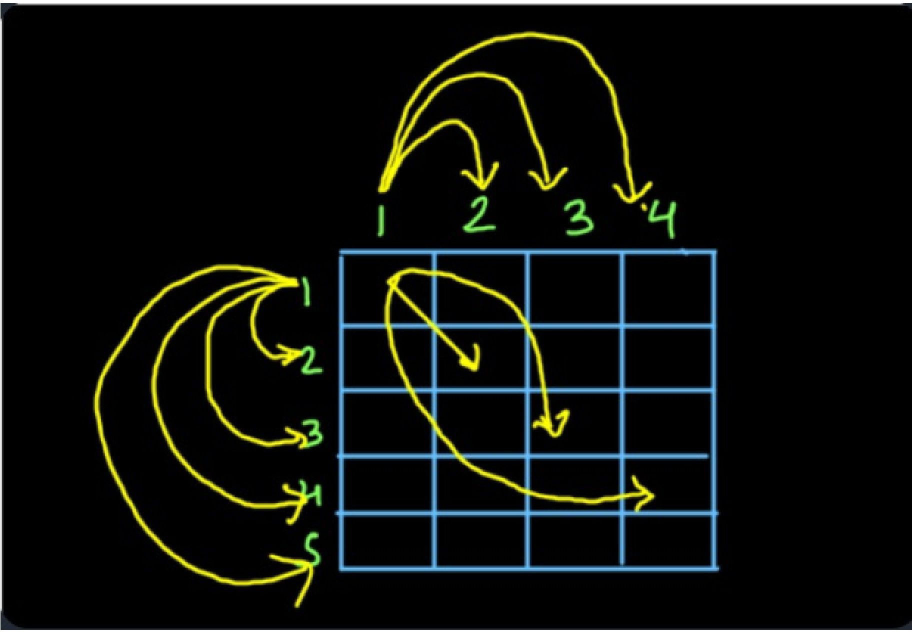
For this ques

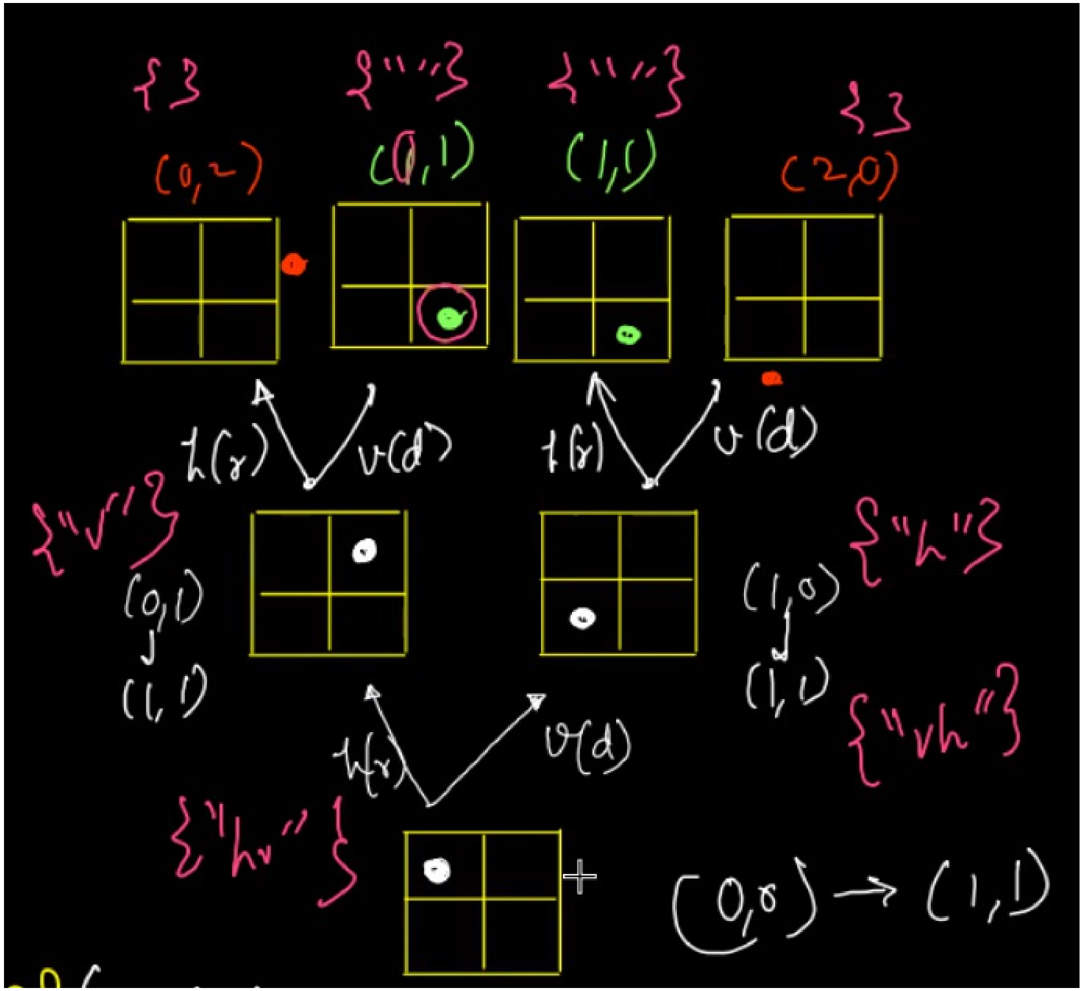
$$\frac{(n-1+m-1)}{(n-1)!\ (m-1)!}$$



$$\begin{tikzcd} & (sr, sc+1) \arrow[dr, "x \text{ paths}"] & \\ (sr, sc) \arrow[ur, "h(r)"] \arrow[dr, "v(d)"'] & & (dr, dc) \\ & (sr+1, sc) \arrow[ur, "y \text{ paths}"'] & \end{tikzcd}$$

Get Maze Paths with Jumps





{}
{" "}
{" "}
{}
(0,2)
(1,1)
(1,1)
(2,0)
h(r)
v(d)
h(r)
v(d)
{"v"}
(0,1)
(1,1)
(1,0)
(1,1)
{"h"}
{"vh"}
h(r)
v(d)
{"hv"}
(0,0) → (1,1)