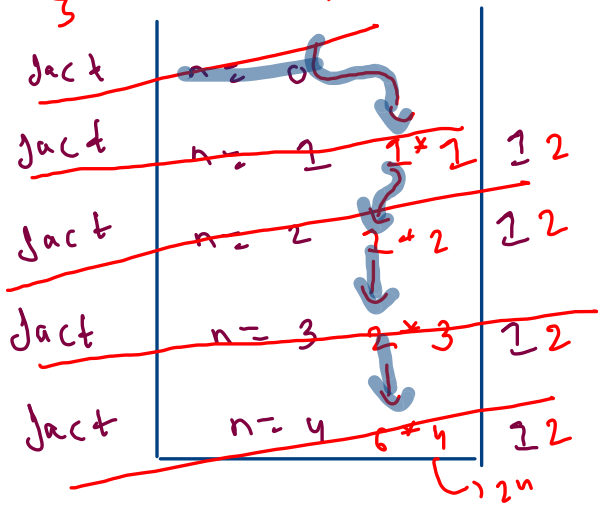
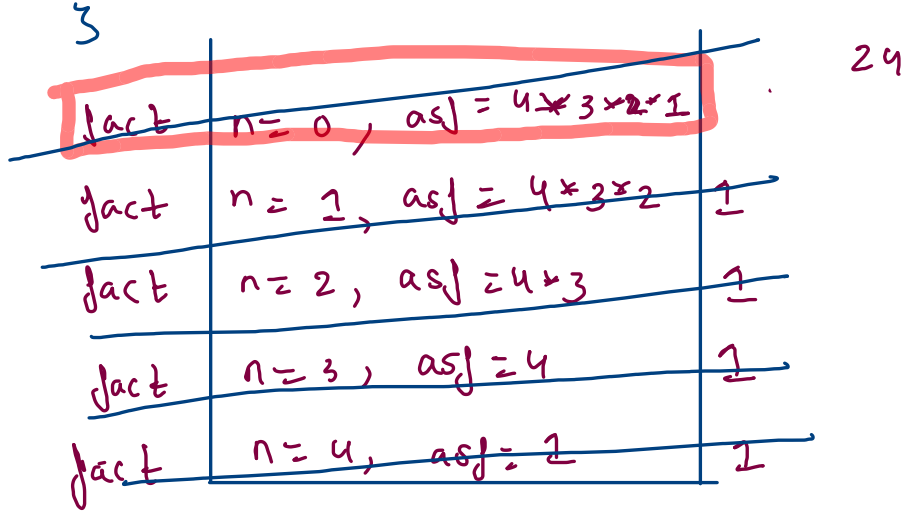
On the way down (Faith & expectation)

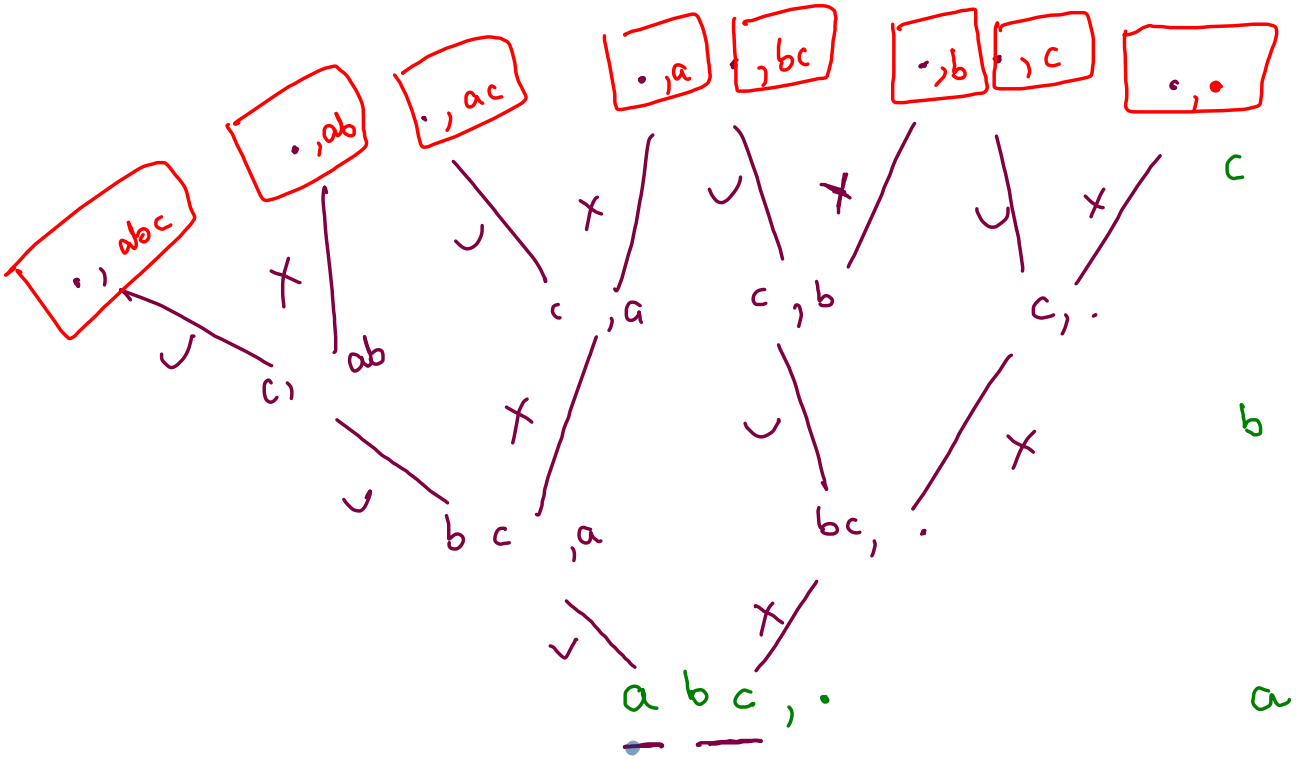
```
int fact (int n) {
  if (n == 0) { return 1; }

1 int fnm1 = fact (n-1);
2 [int fn = n * fnm1;
  return fn;
3 }
```



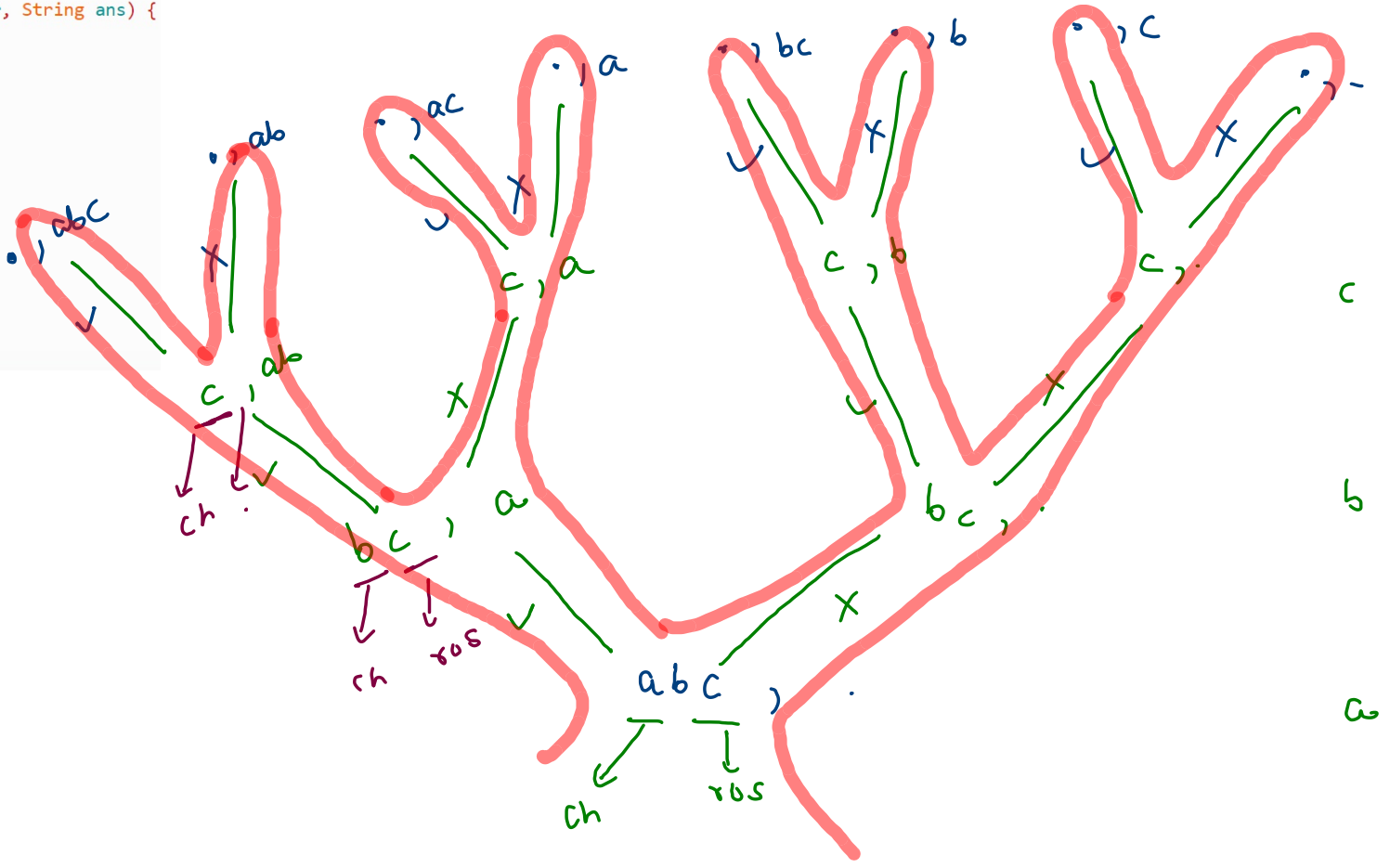
Recursion on the way up

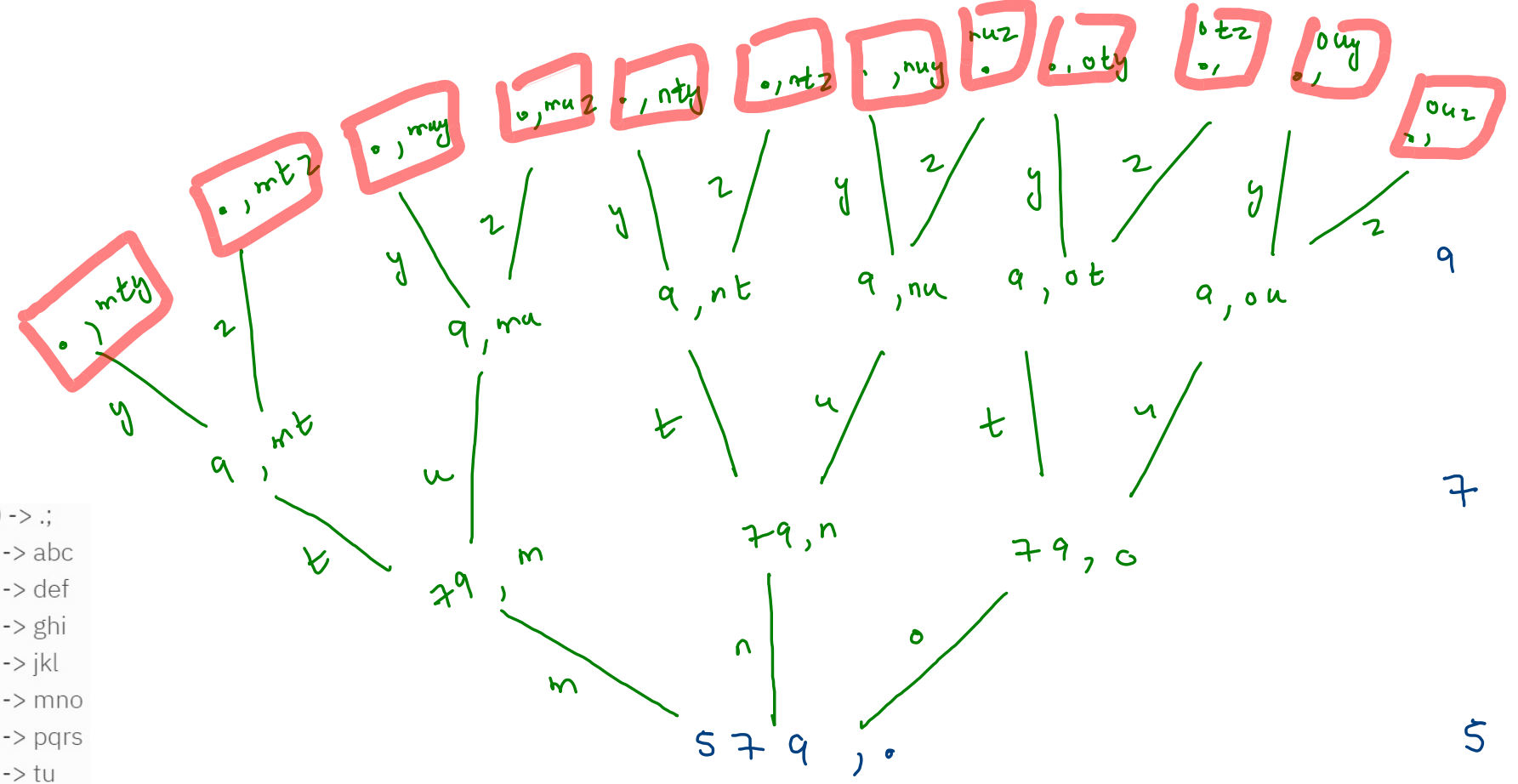
```
void fact (int n, int asf) {
  if (n == 0) {
    syso (asf);
    return;
  }

1. fact ( n-1 , asf * n);
```




•, abc
•, ab
•, ac
•, a
•, bc
•, b
•, c
•, •
c
b
a
c, •
c, b
c, a
ab
bc, •
bc, a
a b c, •

```java
public static void printSS(String str, String ans) {
    if(str.length() == 0) {
        System.out.println(ans);
        return;
    }


    char ch = str.charAt(0);
    String ros = str.substring(1);

    //ch -> present
    printSS(ros,ans + ch);

    //ch -> absent
    printSS(ros,ans);
}
```

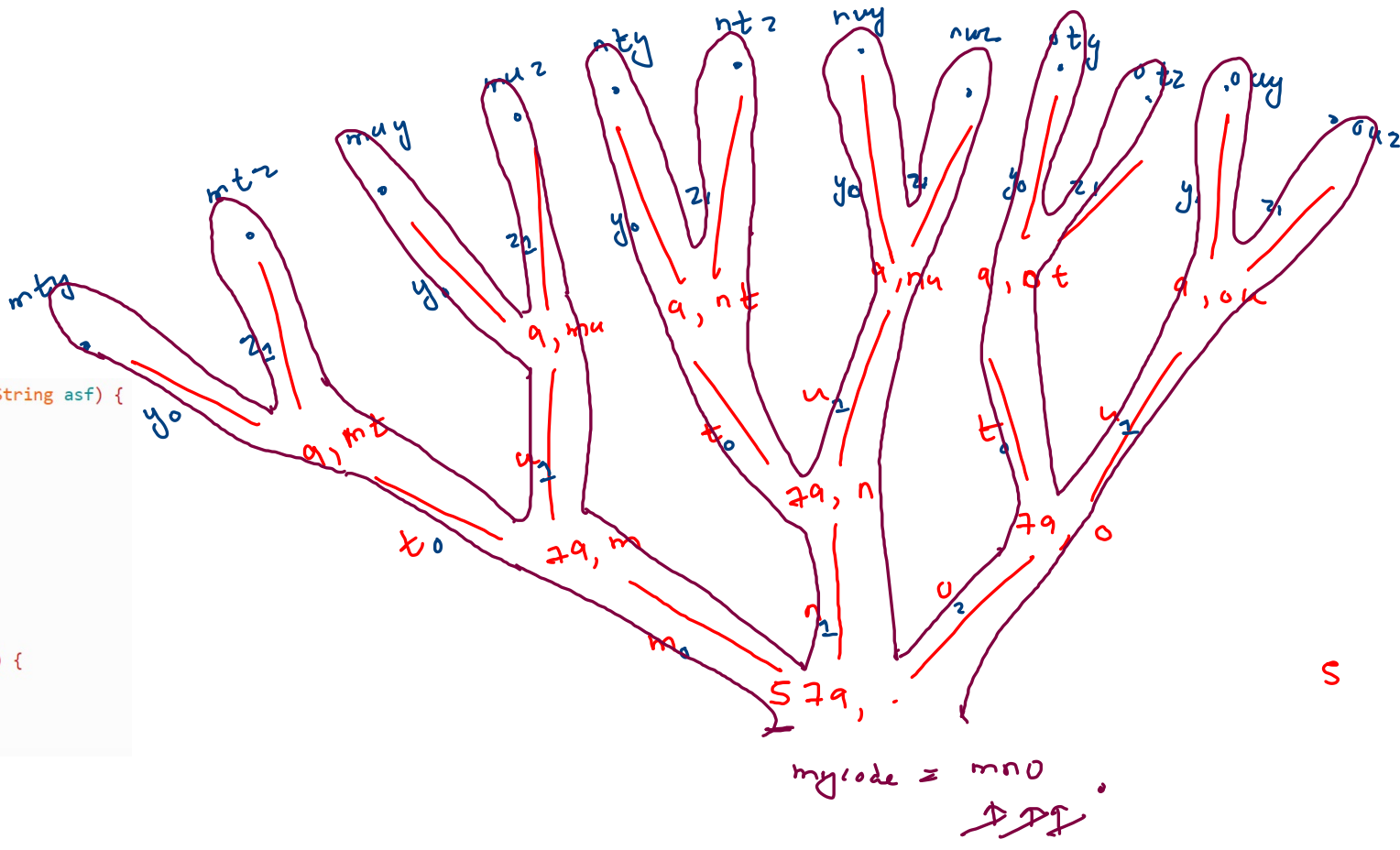


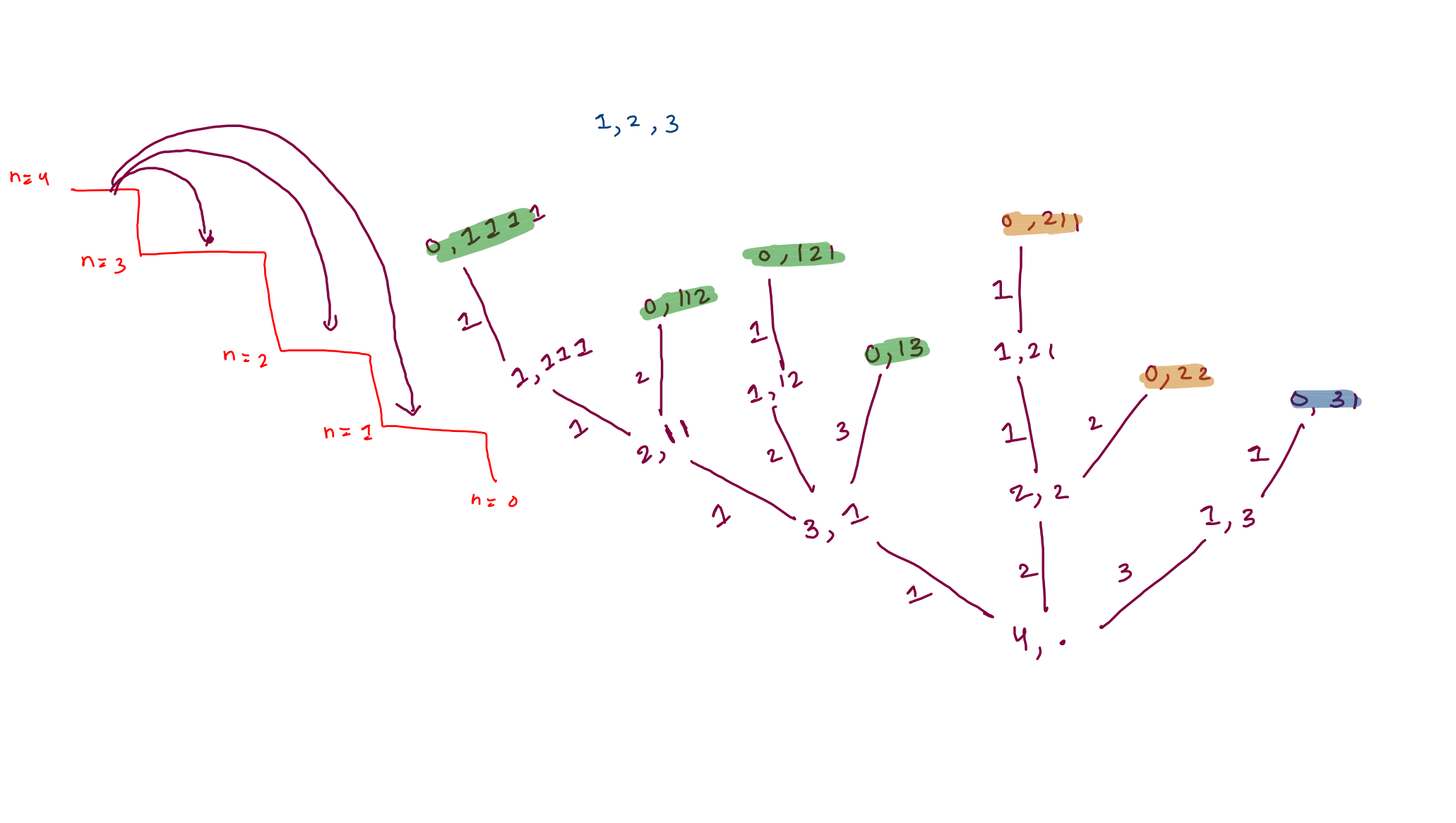


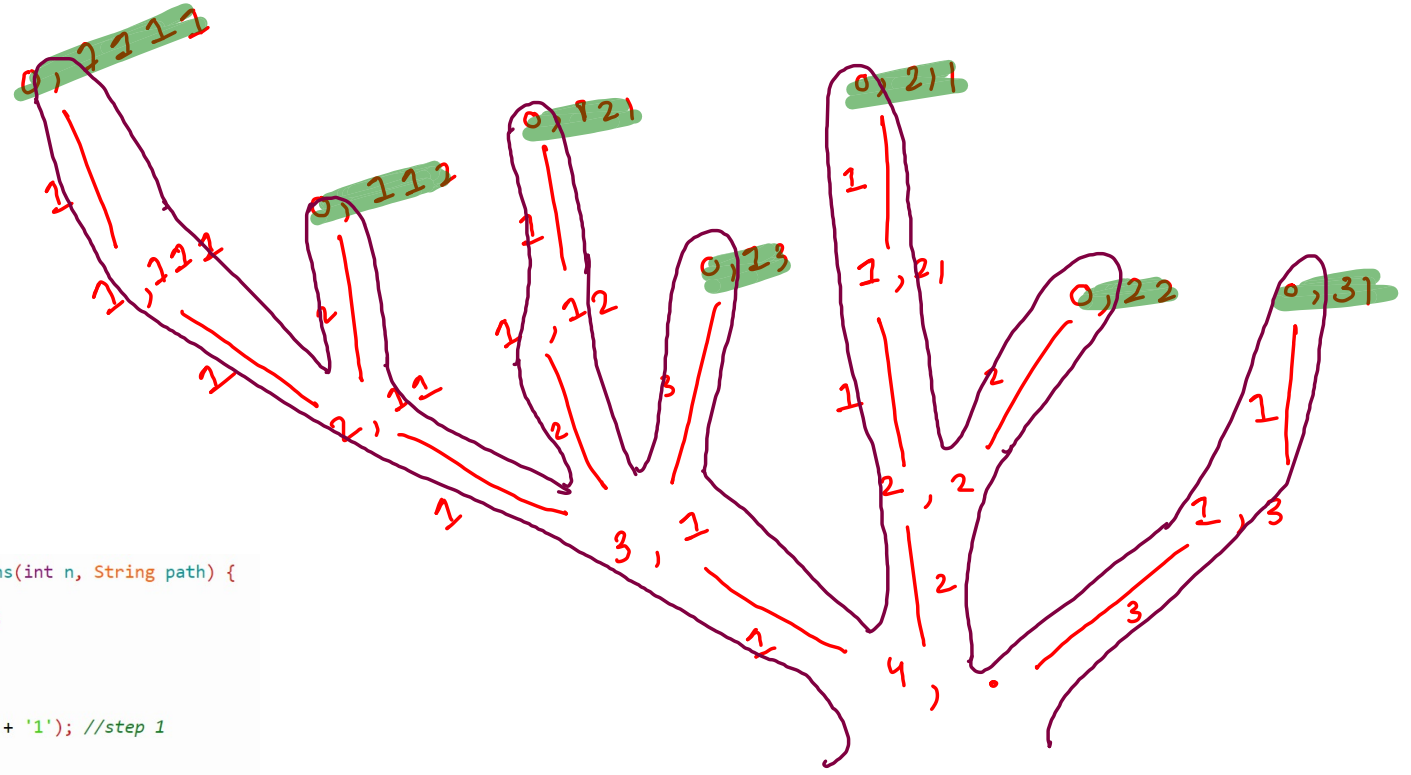
0 -> .;
1 -> abc
2 -> def
3 -> ghi
4 -> jkl
5 -> mno
6 -> pqrs
7 -> tu
8 -> vwx
9 -> yz

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

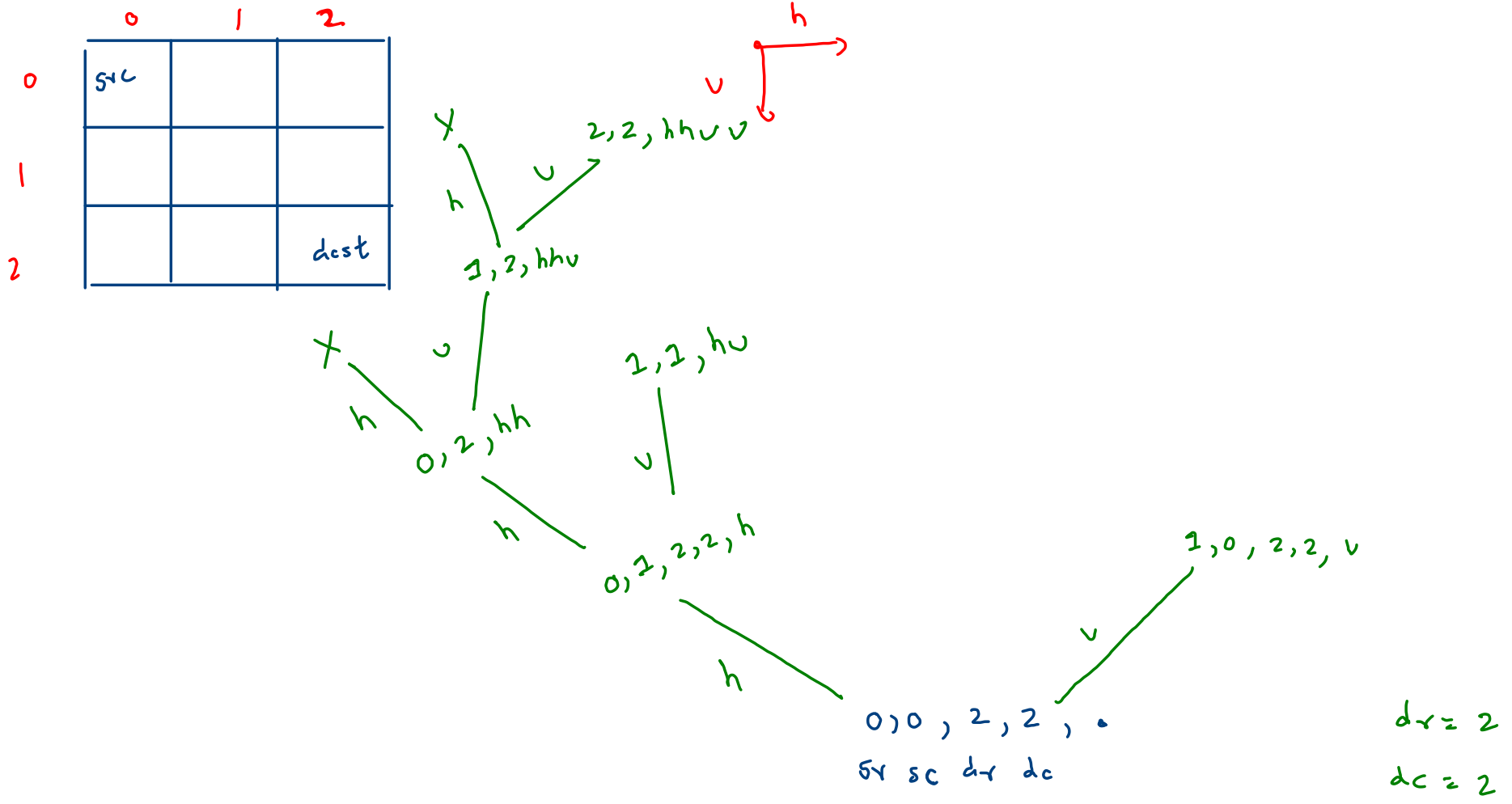
```java
public static void printKPC(String str, String asf) {
    if(str.length() == 0) {
        System.out.println(asf);
        return;
    }

    char ch = str.charAt(0);
    String ros = str.substring(1);

    //ch -> choices
    String mycode = codes[ch-'0'];

    for(int i=0; i < mycode.length();i++) {
        char mch = mycode.charAt(i);
        printKPC(ros,asf + mch);
    }
}
```

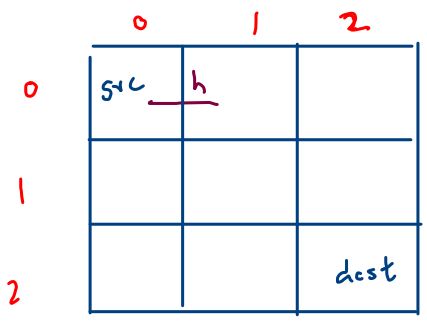



1, 2, 3
n = 4
n = 3
n = 2
n = 1
n = 0
0, 1111
1
1, 111
1
2, 11
1
3, 1
1
4, .
0, 112
2
0, 121
1
1, 12
2
0, 13
3
0, 211
1
1, 21
1
2, 2
2
0, 22
2
0, 31
1
1, 3
3



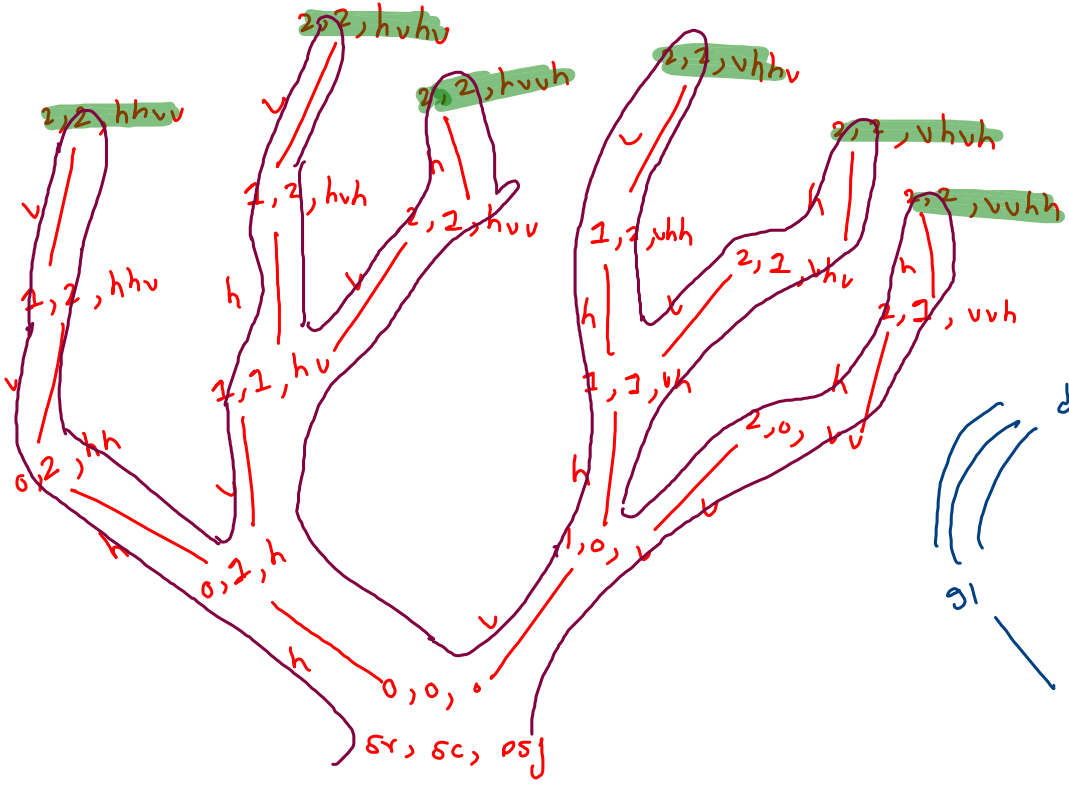

```java
public static void printStairPaths(int n, String path) {
    if(n == 0) {
        System.out.println(path);
        return;
    }

    if(n >= 1) {
        printStairPaths(n-1,path + '1'); //step 1
    }
    if(n >= 2) {
        printStairPaths(n-2,path + '2'); //step 2
    }
    if(n >= 3) {
        printStairPaths(n-3,path + '3'); //step 3
    }
}
```


0
1
2
src
dest
h
v
X
2,2,hhvv
1,2,hhv
2,2,hv
0,2,hh
0,2,2,h
1,0,2,2,v
0,0,2,2,.
sr sc dr dc
dr = 2
dc = 2

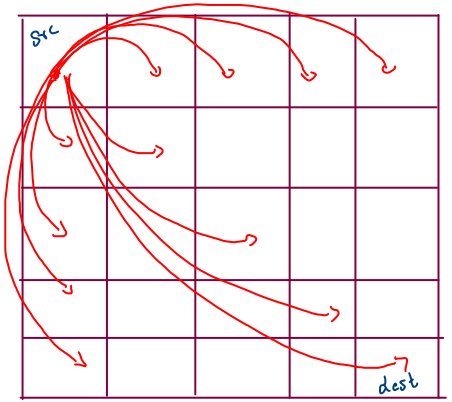


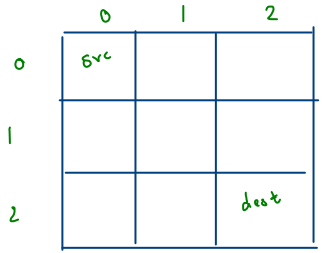
dr = 2

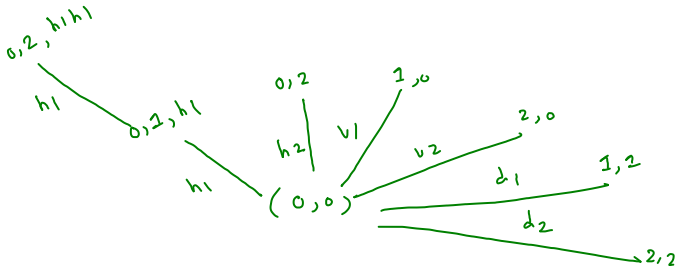
dc = 2



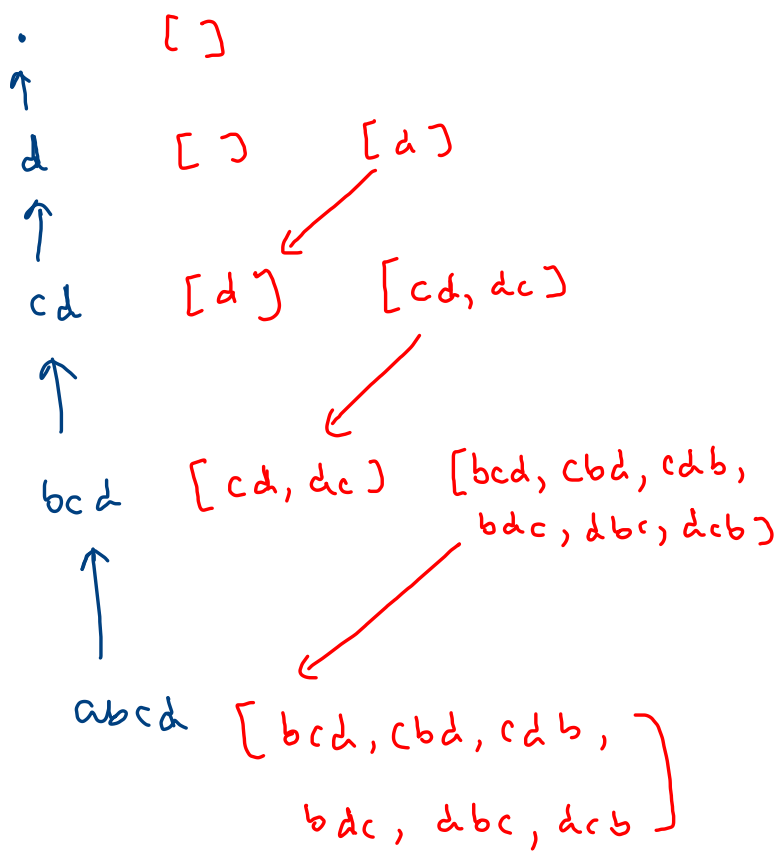

```java
public static void printMazePaths(int sr, int sc, int dr, int dc, String psf) {
    if(sr == dr && sc == dc) {
        System.out.println(psf);
        return;
    }

    if(sc + 1 <= dc) {
        printMazePaths(sr,sc+1,dr,dc,psf + 'h'); //horizontal
    }

    if(sr + 1 <= dr) {
        printMazePaths(sr+1,sc,dr,dc,psf + 'v'); //vertical
    }
}
```


src
dest


0
1
2
0
1
2
src
dest


0,2, h1h1
h1
0,1, h1
h1
0,2
h2
v1
1,0
v2
2,0
d1
1,2
d2
2,2
(0,0)

abc
acb
bac
bca
cab
cba



i = 1

left = (0 to i)

right = (i to end)

perm = left + ch + right

[abcd
bacd
bcad
bcda]

c b d

0 1 2

left = (0 to i)

right = (i to end)

perm = left + ch + right

i = 0

perm = "" + a + cbd = acbd

i = 1

perm = c + a + bd = cabd

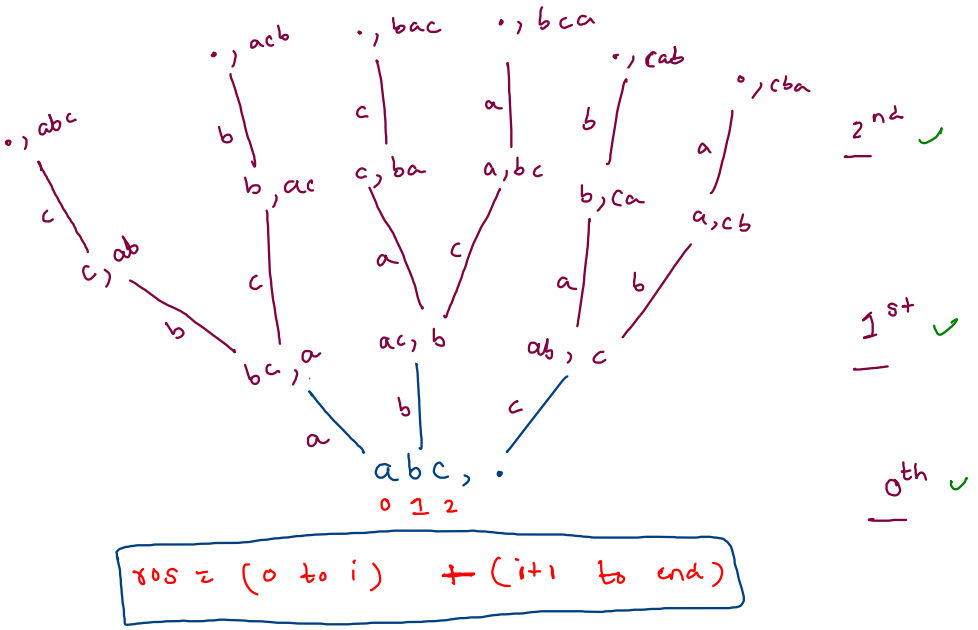
i = 2

perm = cb + a + d = cbad

i = 3

perm = cbd + a + "" → cbda

abc
acb
bac
bca
cab
cba



i = 0 , ros = (0 to i) + (i+1 to end)

= "" + bc = bc

i = 1 , ros = a + c = ac

i = 2, ros = ab + "" = ab

```java
public static void printPermutations(String str,String asf) {
    if(str.length() == 0) {
        System.out.println(asf);
        return;
    }

    for(int i=0; i < str.length();i++) {
        char ch = str.charAt(i);
        String ros = str.substring(0,i) + str.substring(i+1);
        printPermutations(ros,asf + ch);
    }
}
```