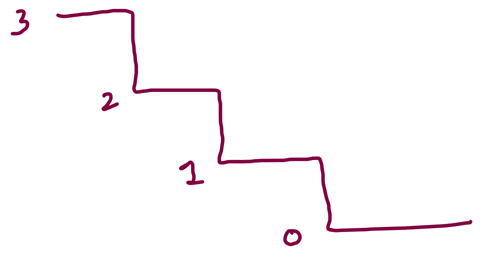
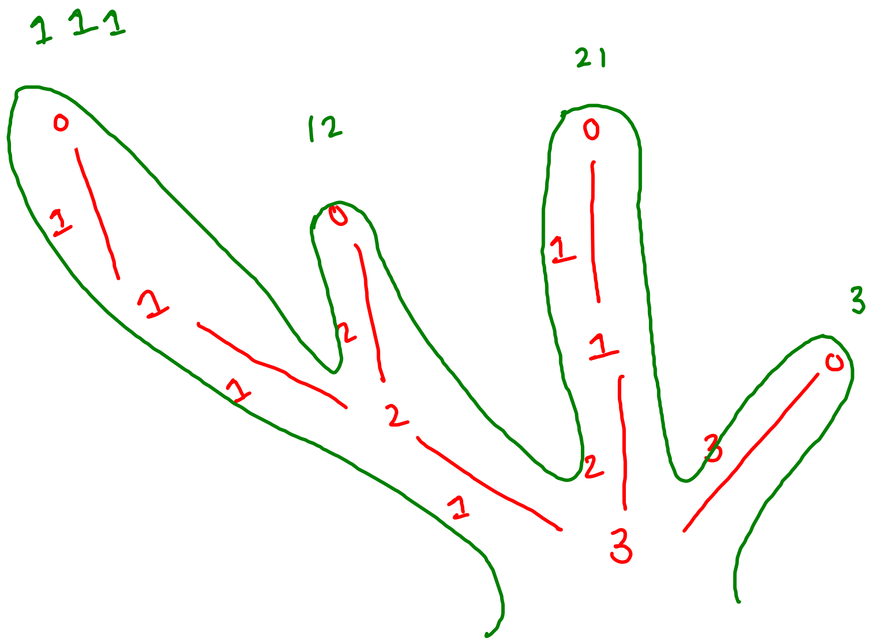
print stair paths



1 jump

2 jump

3 jump



1111

112

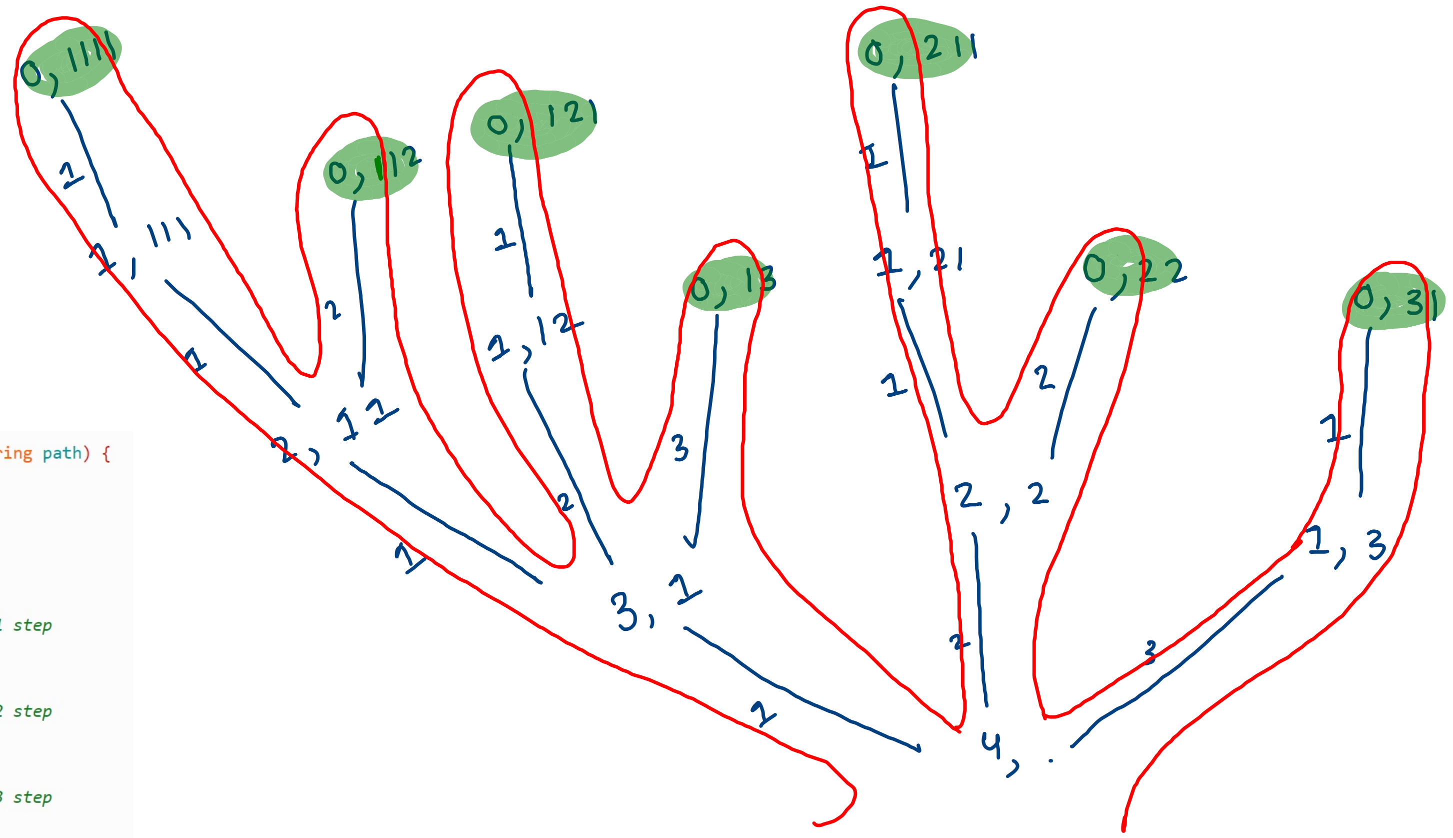
121

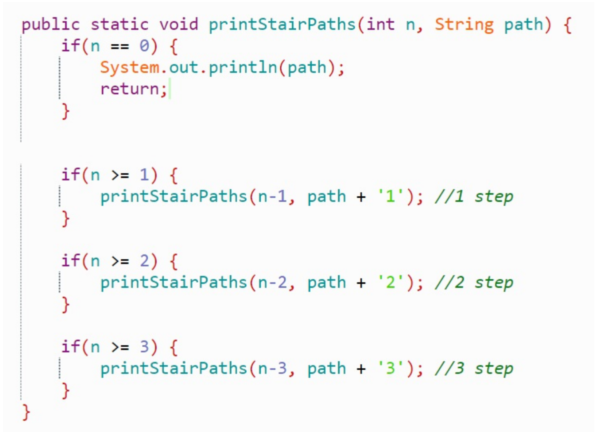
13

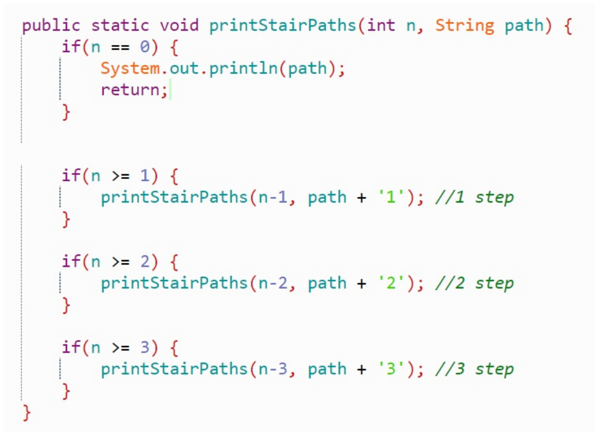
211

22

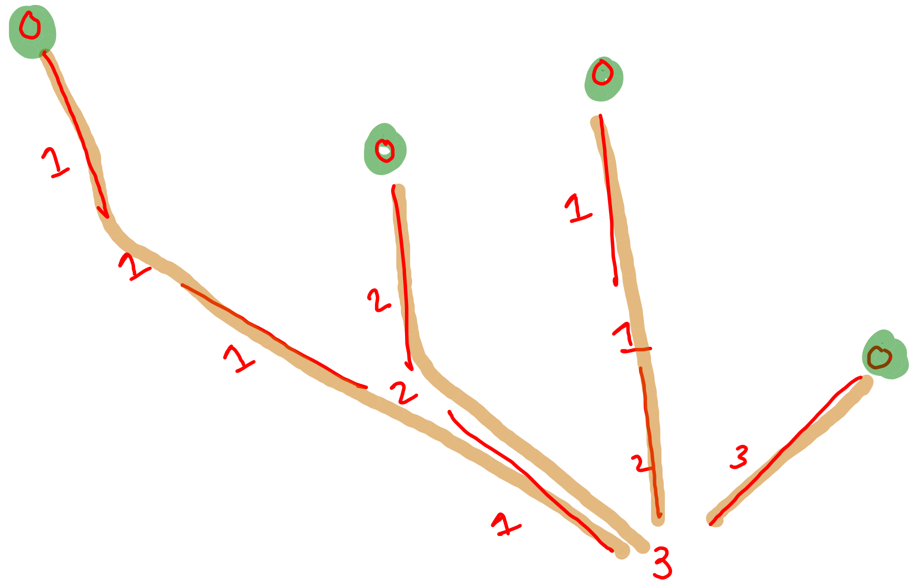
31

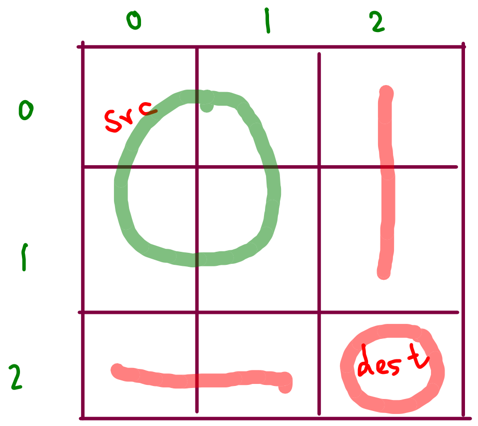


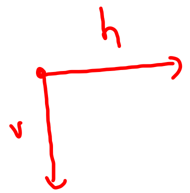

```
public static void printStairPaths(int n, String path) {
    if(n == 0) {
        System.out.println(path);
        return;
    }


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

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

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

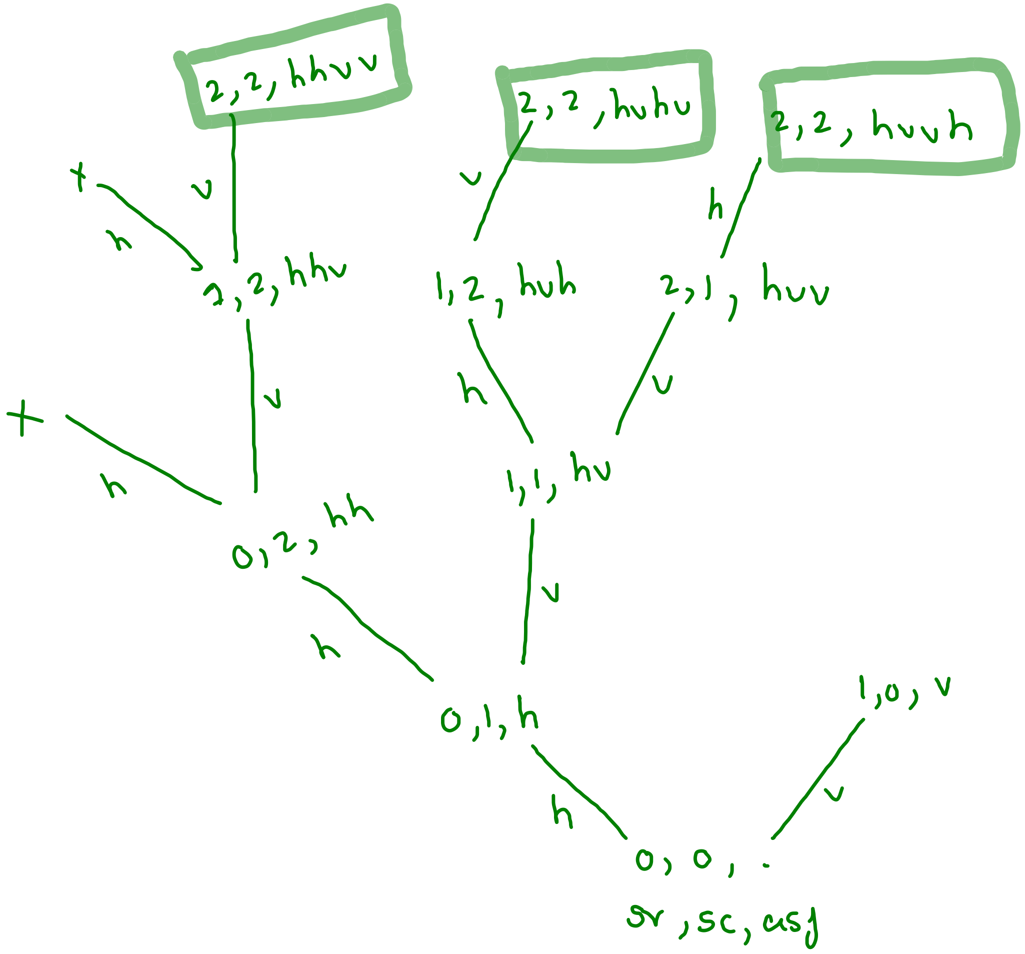


```java
public static void printStairPaths(int n, String path) {
    if(n == 0) {
        System.out.println(path);
        return;
    }

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

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

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



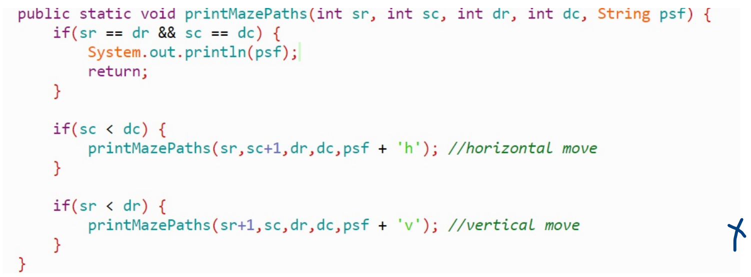

0
1
2
0
1
2
src
dest


h
v


2,2, hhvv
2,2, hvhv
2,2, hvvh
x
h
v
v
h
2,2, hhv
1,2, hvh
2,1, hvv
x
h
v
h
v
1,1, hv
0,2, hh
h
v
0,1, h
1,0, v
h
v
0, 0, .
sr, sc, asf

$d_r = 2$

$d_c = 2$



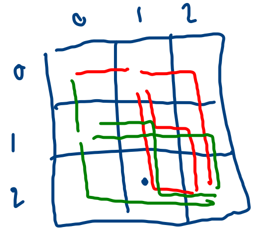
```java
public static void printMazePaths(int sr, int sc, int dr, int dc, String psf) {
    if(sr == dr && sc == dc) {
        System.out.println(psf);
        return;
    }

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

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

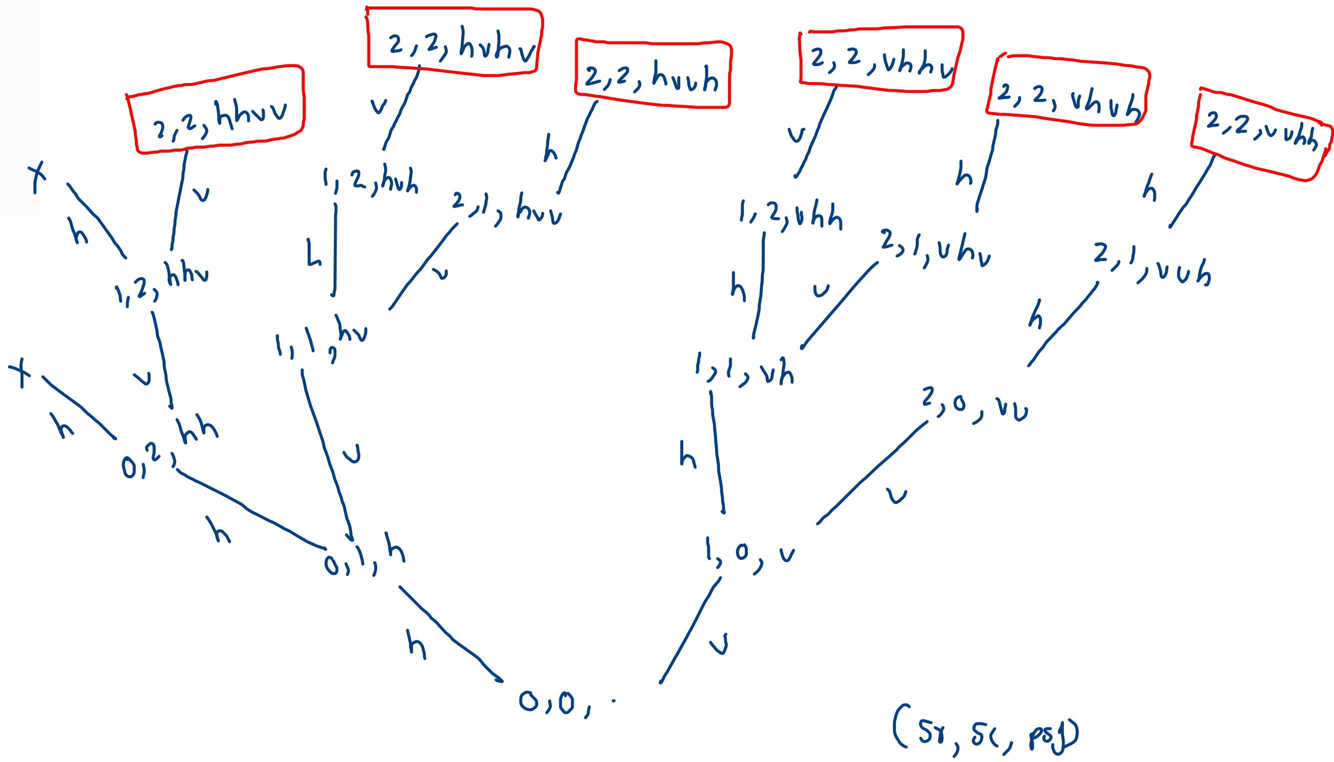
n = 3

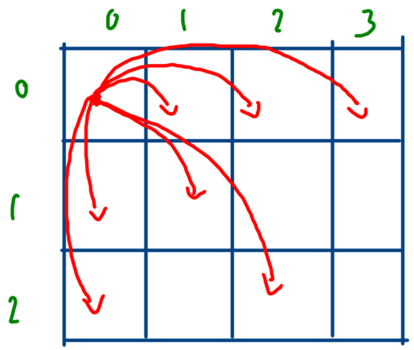
m = 3

dr = 2

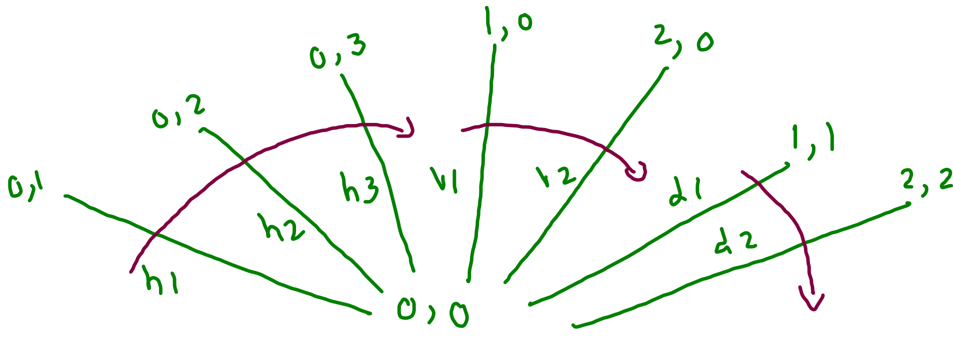
dc = 2

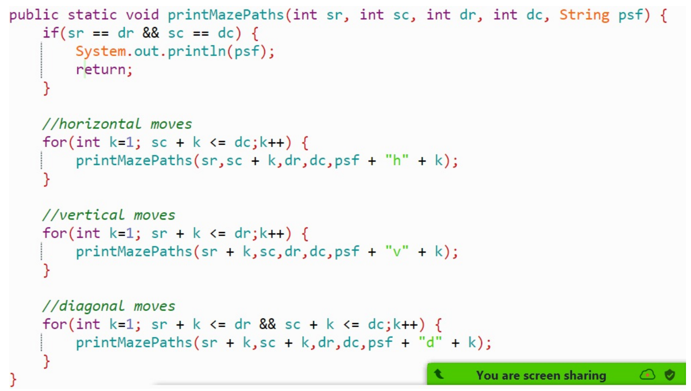






$\boxed{2, 3}$

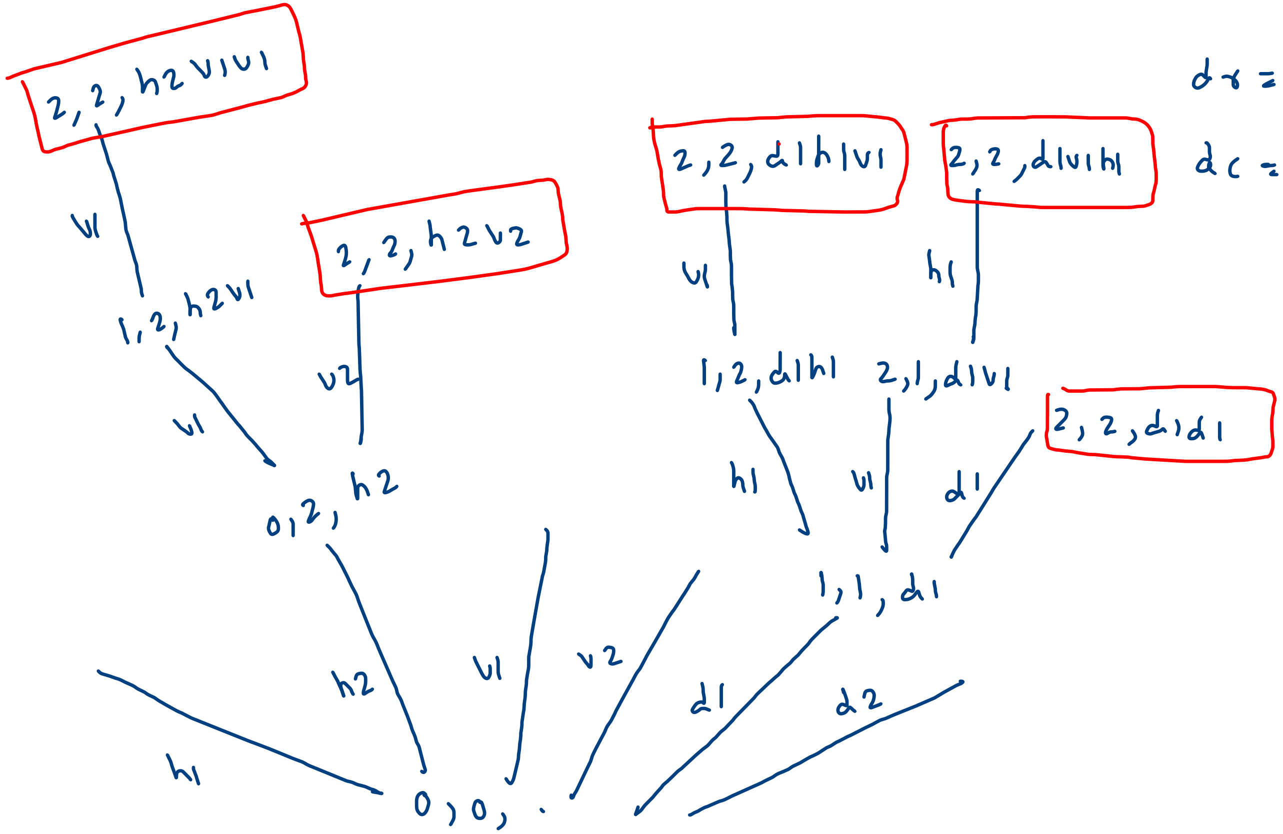


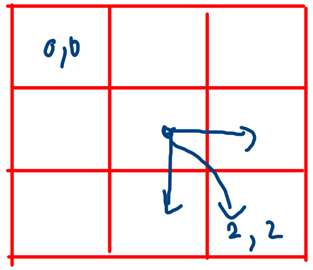


```java
public static void printMazePaths(int sr, int sc, int dr, int dc, String psf) {
    if(sr == dr && sc == dc) {
        System.out.println(psf);
        return;
    }

    //horizontal moves
    for(int k=1; sc + k <= dc;k++) {
        printMazePaths(sr,sc + k,dr,dc,psf + "h" + k);
    }

    //vertical moves
    for(int k=1; sr + k <= dr;k++) {
        printMazePaths(sr + k,sc,dr,dc,psf + "v" + k);
    }

    //diagonal moves
    for(int k=1; sr + k <= dr && sc + k <= dc;k++) {
        printMazePaths(sr + k,sc + k,dr,dc,psf + "d" + k);
    }
}
```

dr = 2

dc = 2



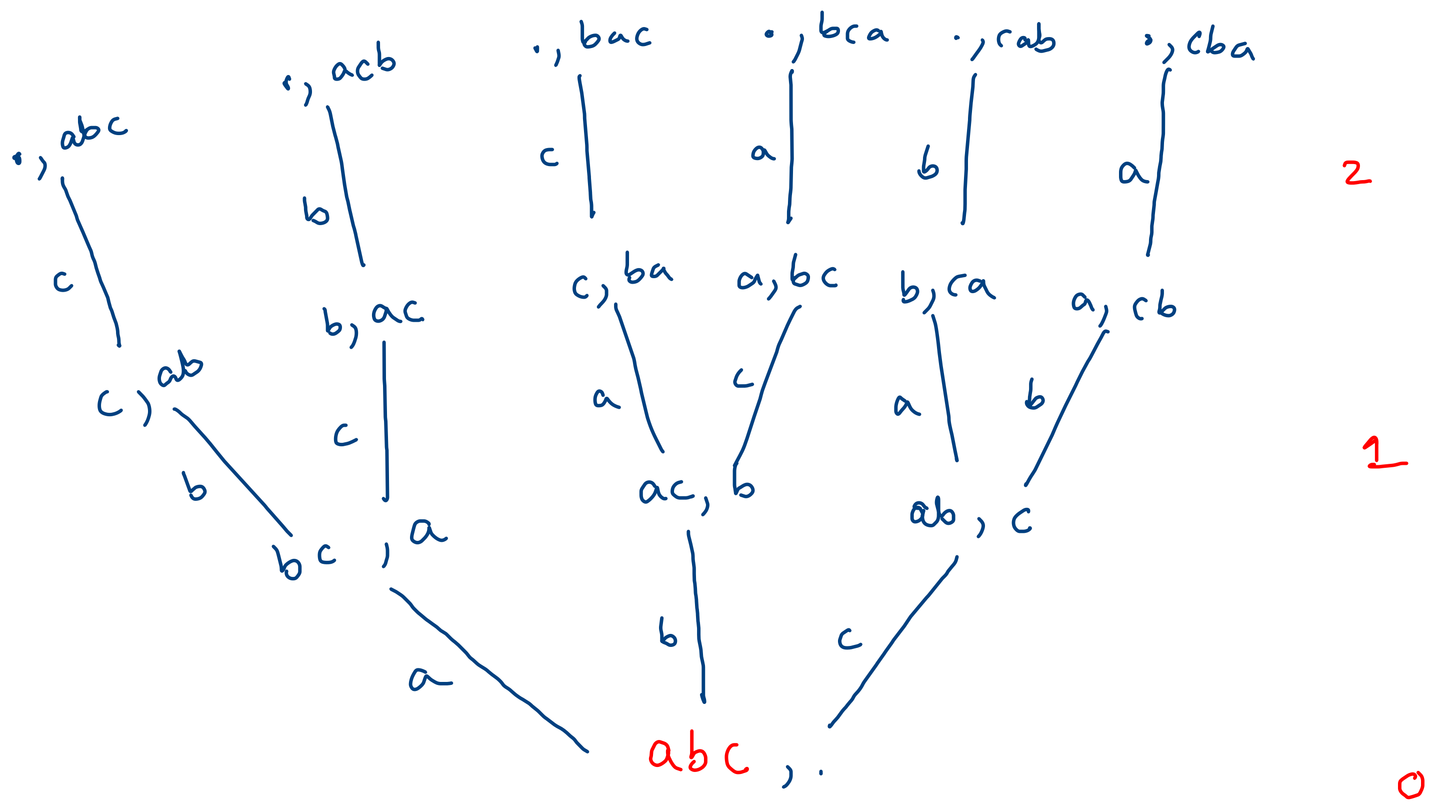


permutations :   str : abc   tp = 3! = 6

a b c
a c b

b a c
c a b

b c a
c b a



str = abcd

left part	+ ch	+ right part	res
""	a	bcd	abcd - 'a' = bcd
a	b	cd	abcd - 'b' = acd
ab	c	d	abcd - 'c' = abd
abc	d	""	abcd - 'd' = abc

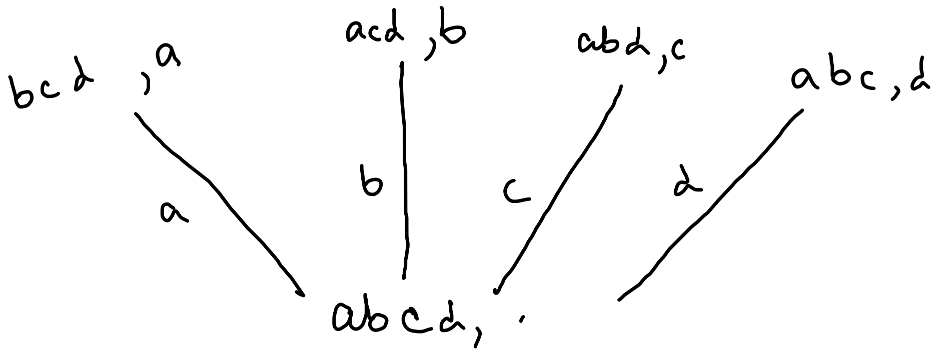
3

2

res = lp + rp

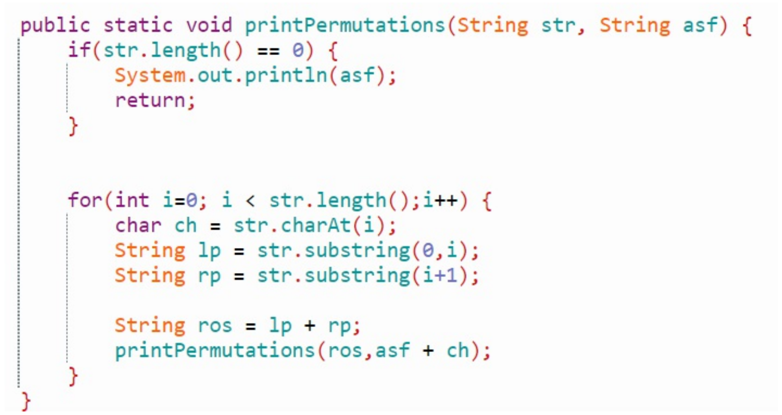
lp = str.ss(0,i);

rp = str.ss(i+1);



1

0



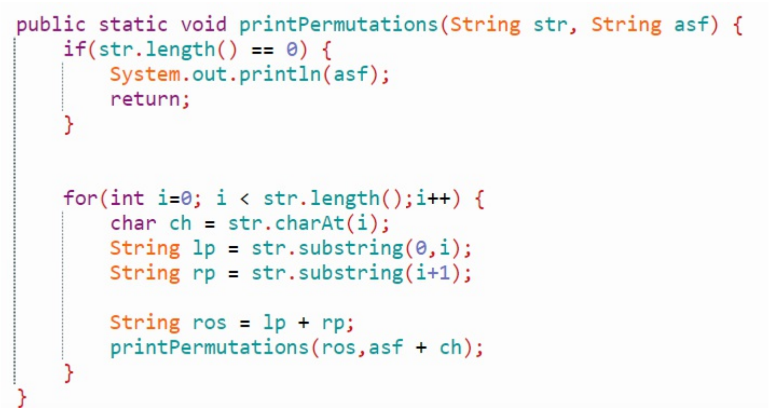
```java
public static void printPermutations(String str, String asf) {
    if(str.length() == 0) {
        System.out.println(asf);
        return;
    }


    for(int i=0; i < str.length();i++) {
        char ch = str.charAt(i);
        String lp = str.substring(0,i);
        String rp = str.substring(i+1);

        String ros = lp + rp;
        printPermutations(ros,asf + ch);
    }
}
```

i

str = abcd

0 1 2 3

	lp	ch	rp		ros
i = 0	(0,0) = .	a	(1)	= bcd	bcd
i = 1	(0,1) = a	b	(2)	= cd	acd
i = 2	(0,2) = ab	c	(3)	= d	abd
i = 3	(0,3) = abc	d	(4)	= .	abc



```
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 lp = str.substring(0,i);
        String rp = str.substring(i+1);

        String ros = lp + rp;
        printPermutations(ros,asf + ch);
    }
}
```

abc

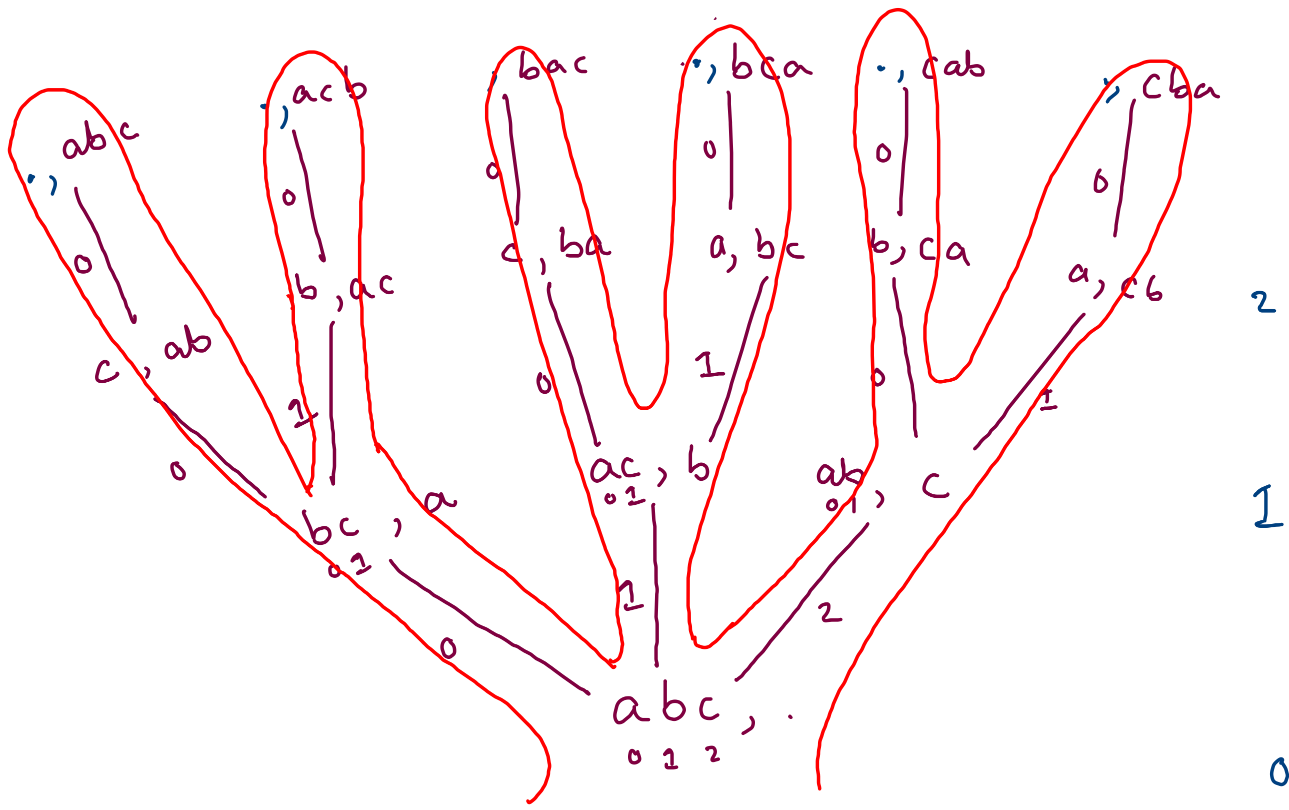
acb

bac

bca

cab

cba



Encodings

1	→ a	11	→ k	21	– u
2	→ b	12	→ l	22	– v
3	→ c	13	→ m	23	– w
4	→ d	14	→ n	24	– x
5	→ e	15	– o	25	– y
6	→ f	16	– p	26	– z
7	→ g	17	– q		
8	→ h	18	– r		
9	→ i	19	– s		
10	→ j	20	– t		

1123 → 1 | 1 | 2 | 3 aabc

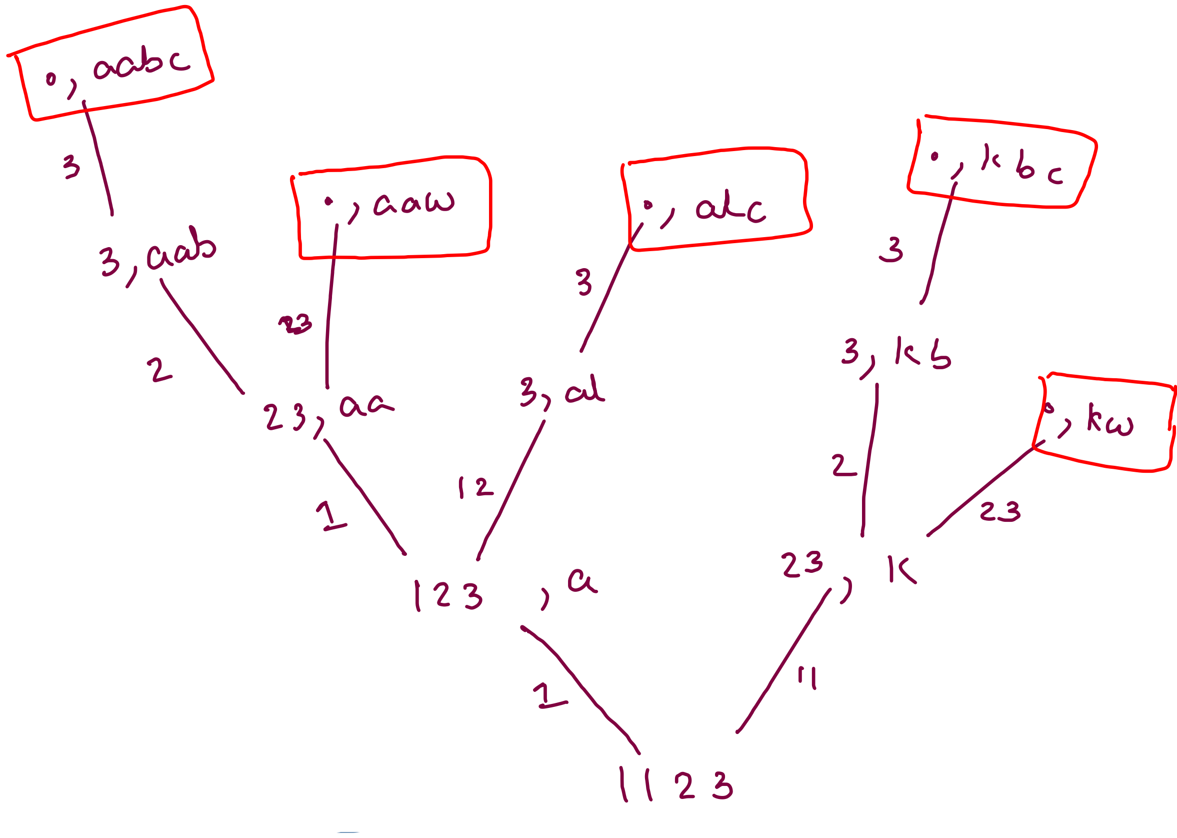
→ 1 | 12 | 3 alc

→ 1 | 1 | 23 aaw

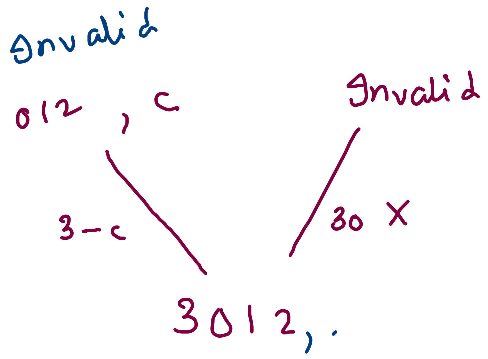
→ 11 | 23 kw

→ 11 | 2 | 3 kbc

1 →	a	11 →	k	21 -	u
2 →	b	12 →	l	22 -	v
3 →	c	13 →	m	23 -	w
4 →	d	14 →	n	24 -	x
5 →	e	15 -	o	25 -	y
6 →	f	16 -	p	26 -	z
7 →	g	17 -	q		
8 →	h	18 -	r		
9 →	i	19 -	s		
10 →	j	20 -	t		



1 →	a	11 →	k	21 -	u
2 →	b	12 →	l	22 -	v
3 →	c	13 →	m	23 -	w
4 →	d	14 →	n	24 -	x
5 →	e	15 -	o	25 -	y
6 →	f	16 -	p	26 -	z
7 →	g	17 -	q		
8 →	h	18 -	r		
9 →	i	19 -	s		
10 →	j	20 -	t		



(i) if any sub-part has > 26

(ii) if any sub-part has starting cha as 0.

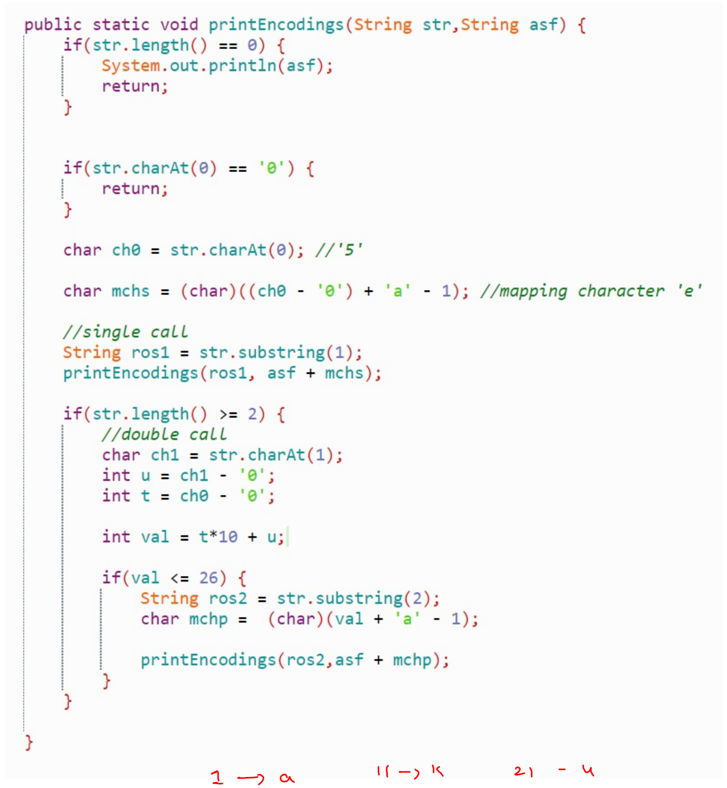
1 → a	11 → k	21 - u
2 → b	12 → l	22 - v
3 → c	13 → m	23 - w
4 → d	14 → n	24 - x
5 → e	15 - o	25 - y
6 → f	16 - p	26 - z
7 → g	17 - q	
8 → h	18 - r	
9 → i	19 - s	
10 → j	20 - t	

single char

```
str = "241"
int val = '2' - '0' → 2

char ch = char(val + 'a' - 1)
```

pair up

```
int u = str[1] - '0';      4
int t = str[0] - '0';      2
int val = t * 10 + u;

char ch = char(val + 'a' - 1)
```

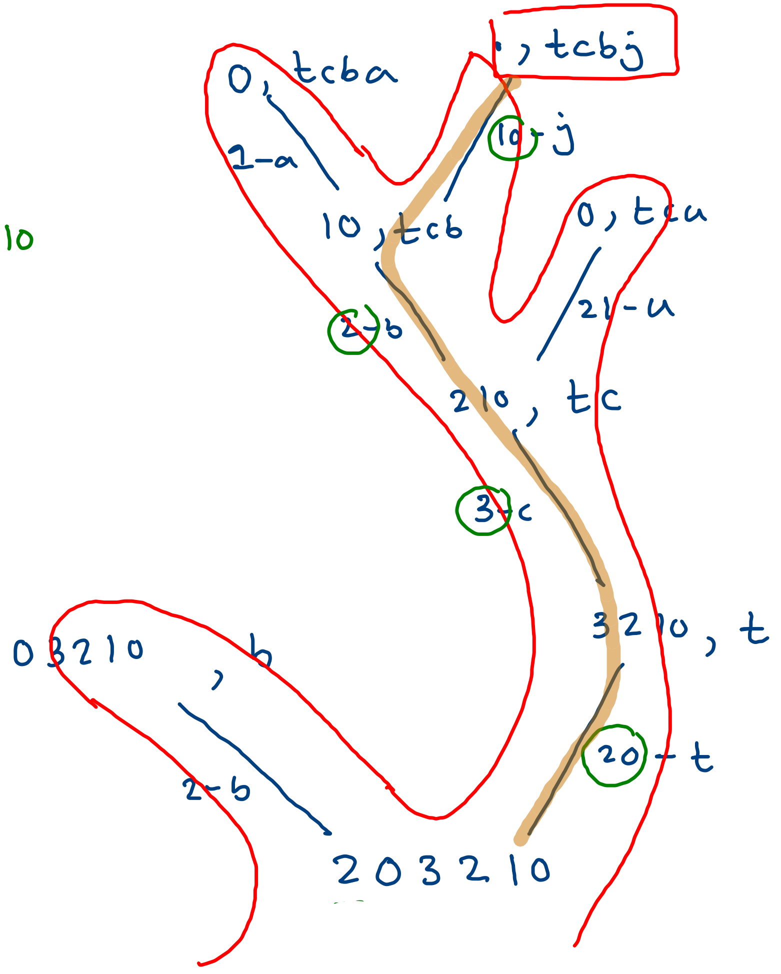


```java
public static void printEncodings(String str,String asf) {
    if(str.length() == 0) {
        System.out.println(asf);
        return;
    }


    if(str.charAt(0) == '0') {
        return;
    }

    char ch0 = str.charAt(0); //'5'

    char mchs = (char)((ch0 - '0') + 'a' - 1); //mapping character 'e'

    //single call
    String ros1 = str.substring(1);
    printEncodings(ros1, asf + mchs);

    if(str.length() >= 2) {
        //double call
        char ch1 = str.charAt(1);
        int u = ch1 - '0';
        int t = ch0 - '0';

        int val = t*10 + u;

        if(val <= 26) {
            String ros2 = str.substring(2);
            char mchp =  (char)(val + 'a' - 1);

            printEncodings(ros2,asf + mchp);
        }
    }

}
```

1 → a	11 → k	21 - u
2 → b	12 → l	22 - v
3 → c	13 → m	23 - w
4 → d	14 → n	24 - x
5 → e	15 - o	25 - y
6 → f	16 - p	26 - z
7 → g	17 - q	
8 → h	18 - r	
9 → i	19 - s	
10 → j	20 - t	

20 | 3 | 2 | 10

t c b j



, tcbj
0, tcba
1-a
10-j
10 , tcb
0, tcu
21-u
2-b
210 , tc
3-c
3210 , t
03210 , b
20-t
2-b
203210