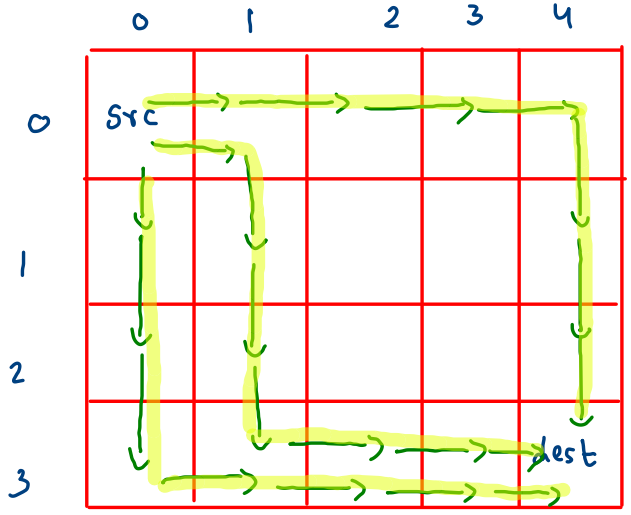
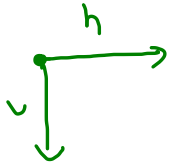
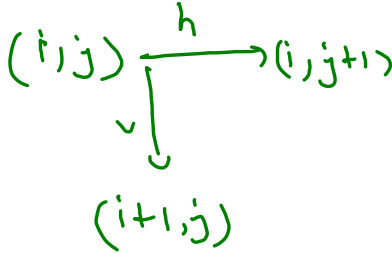
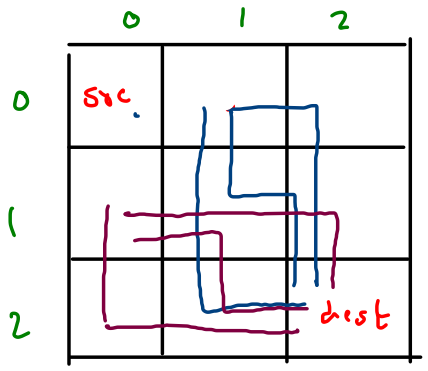
get maze paths





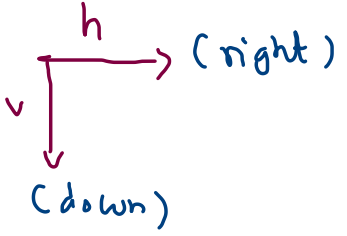
[hhhhvvv, hvvvhhh, vvvhhhh]

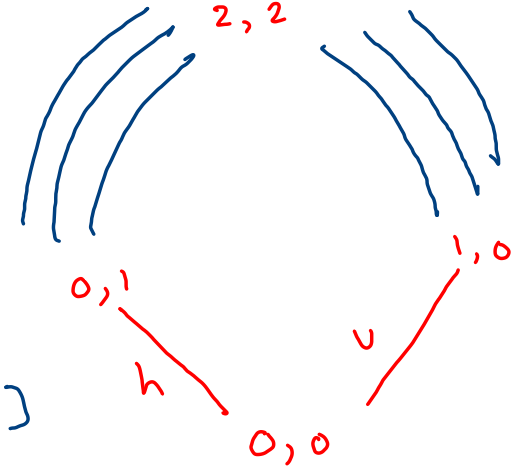




0,1 to 2,2 => [hvv, vhv, vvh]

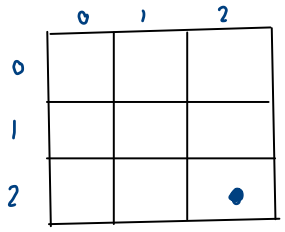
1,0 to 2,2 => [vhh, hhv, hvh]





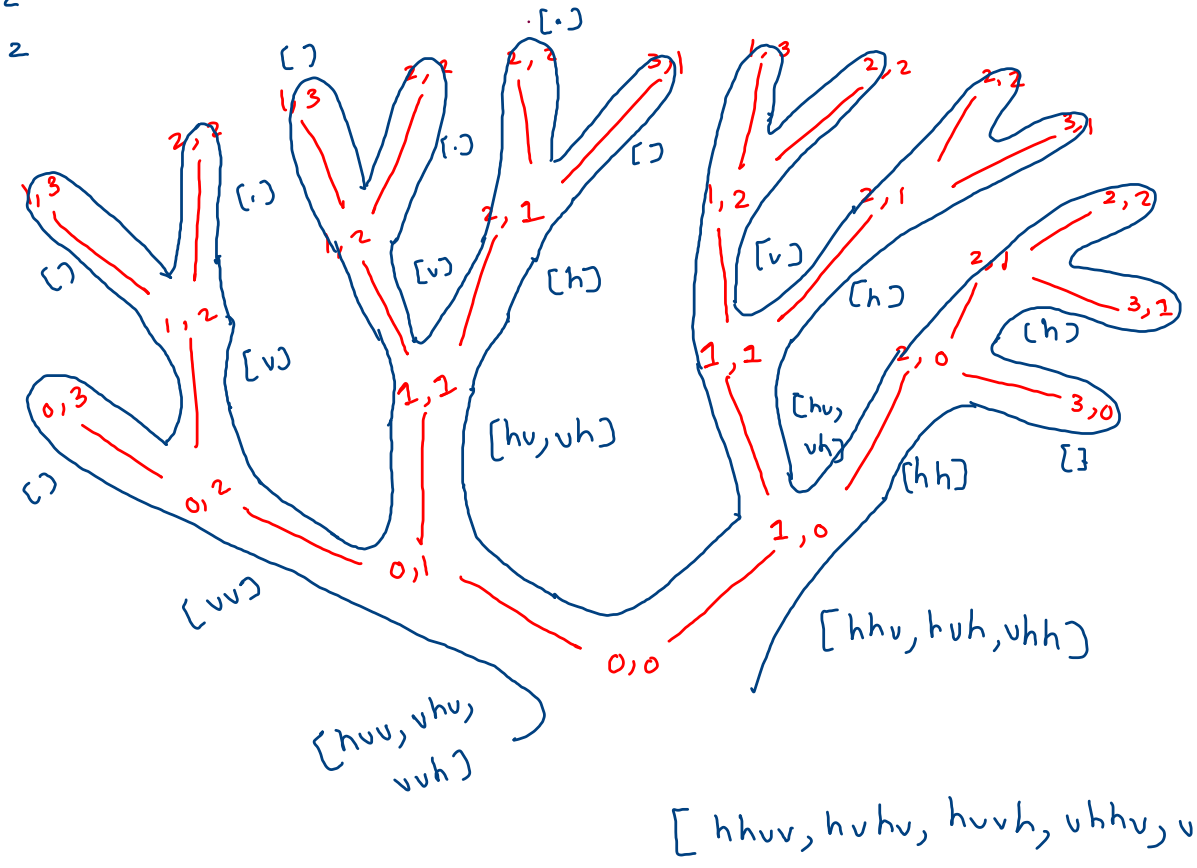
0,0 to 2,2 => [hhvv, hvhv, hvvh,

vvhh, vhhv, vhvh]



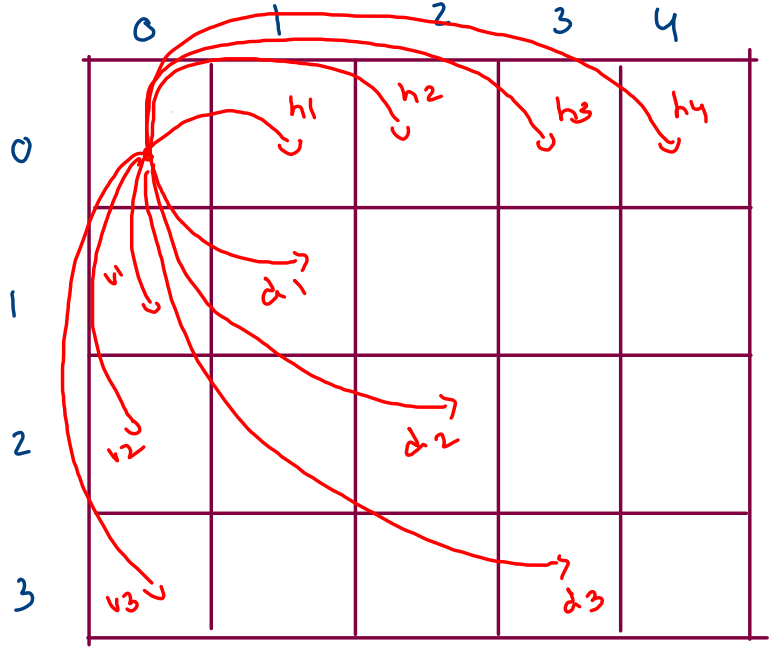
dr = 2

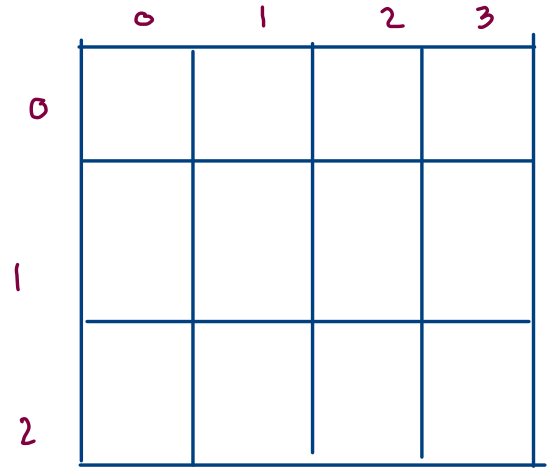
dc = 2



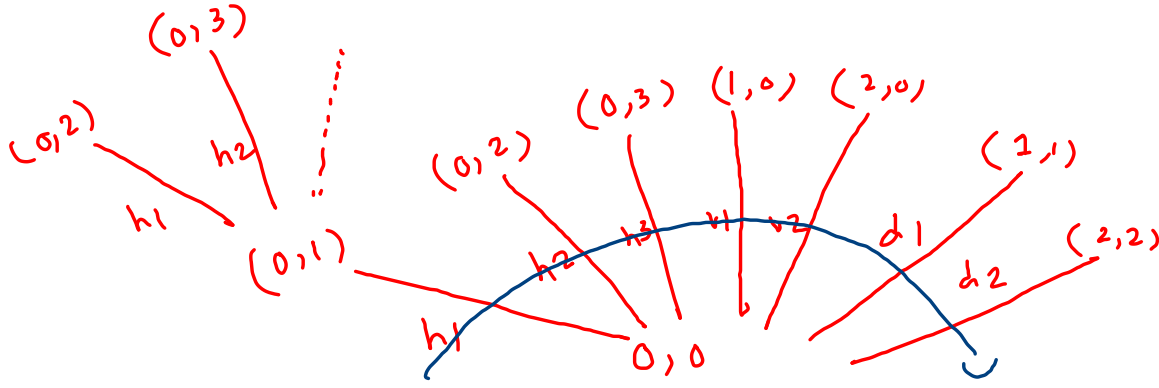

```java
public static ArrayList<String> getMazePaths(int sr, int sc, int dr, int dc) {
    if(sr == dr && sc == dc) {
        ArrayList<String>bans = new ArrayList<>();
        bans.add("");
        return bans;
    }

    if(sr > dr || sc > dc) {
        ArrayList<String>bans = new ArrayList<>();
        return bans;
    }

    ArrayList<String>hntod = getMazePaths(sr,sc+1,dr,dc); //horizontal nbr to dest
    ArrayList<String>vntod = getMazePaths(sr+1,sc,dr,dc); //vertical nbr to dest

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

    //src to dest -> 'h' + horizontal nbr to dest
    for(int i=0; i < hntod.size();i++) {
        stod.add('h' + hntod.get(i));
    }

    //src to dest -> 'v' + vertical nbr to dest
    for(int i=0; i < vntod.size();i++) {
        stod.add('v' + vntod.get(i));
    }

    return stod;
}
```

maze path
with variable jumps





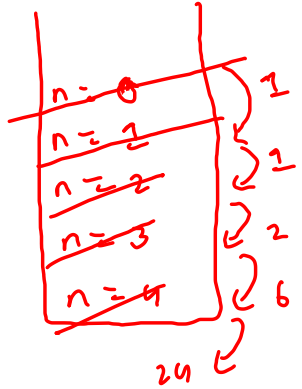
2, 3



factorial

on the way down (faith & expect.)

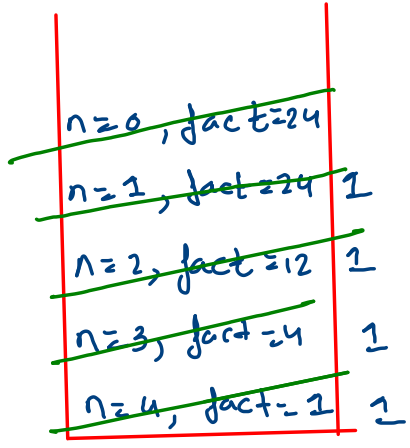
```
int factorial (int n) {
    if (n == 0) return 1;

    int fnm1 = factorial (n-1);

    int fn = n * fnm1;

    return fn;
}
```



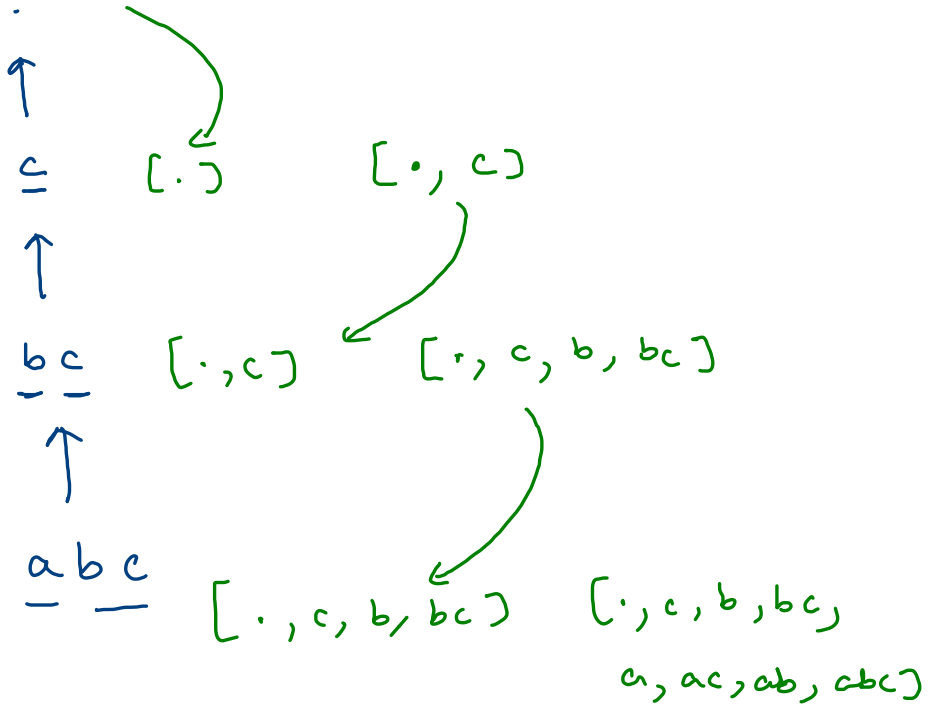
on the way up (level & options)

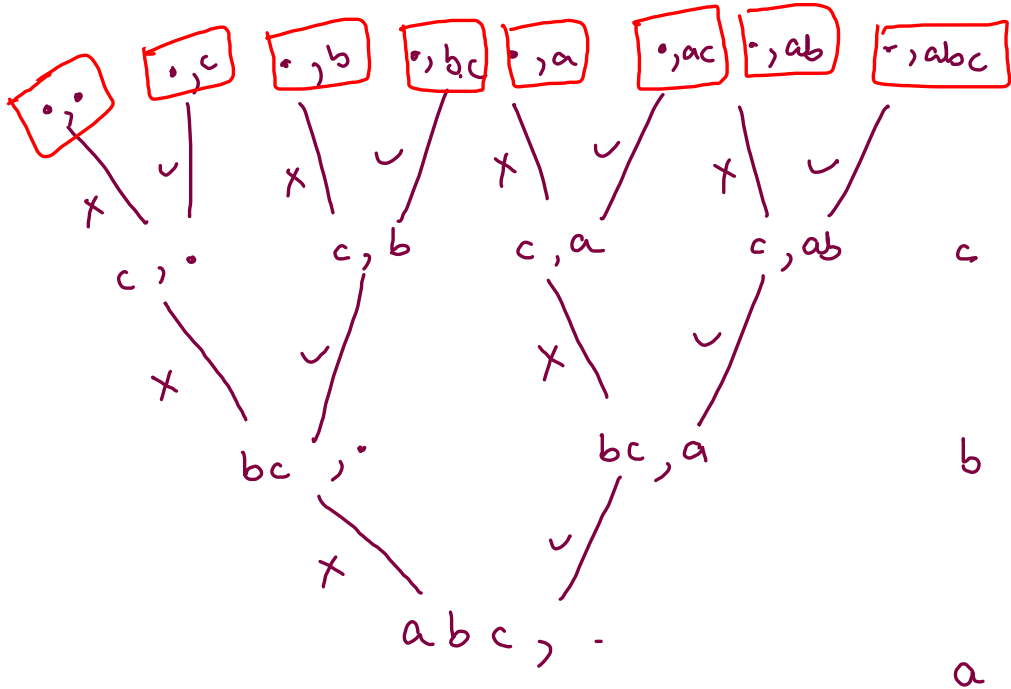
```
void factorial (int n, int fact);
    if (n == 0) {sysо (fact); return;}

    factorial (n-1, n * fact);
}
```

24

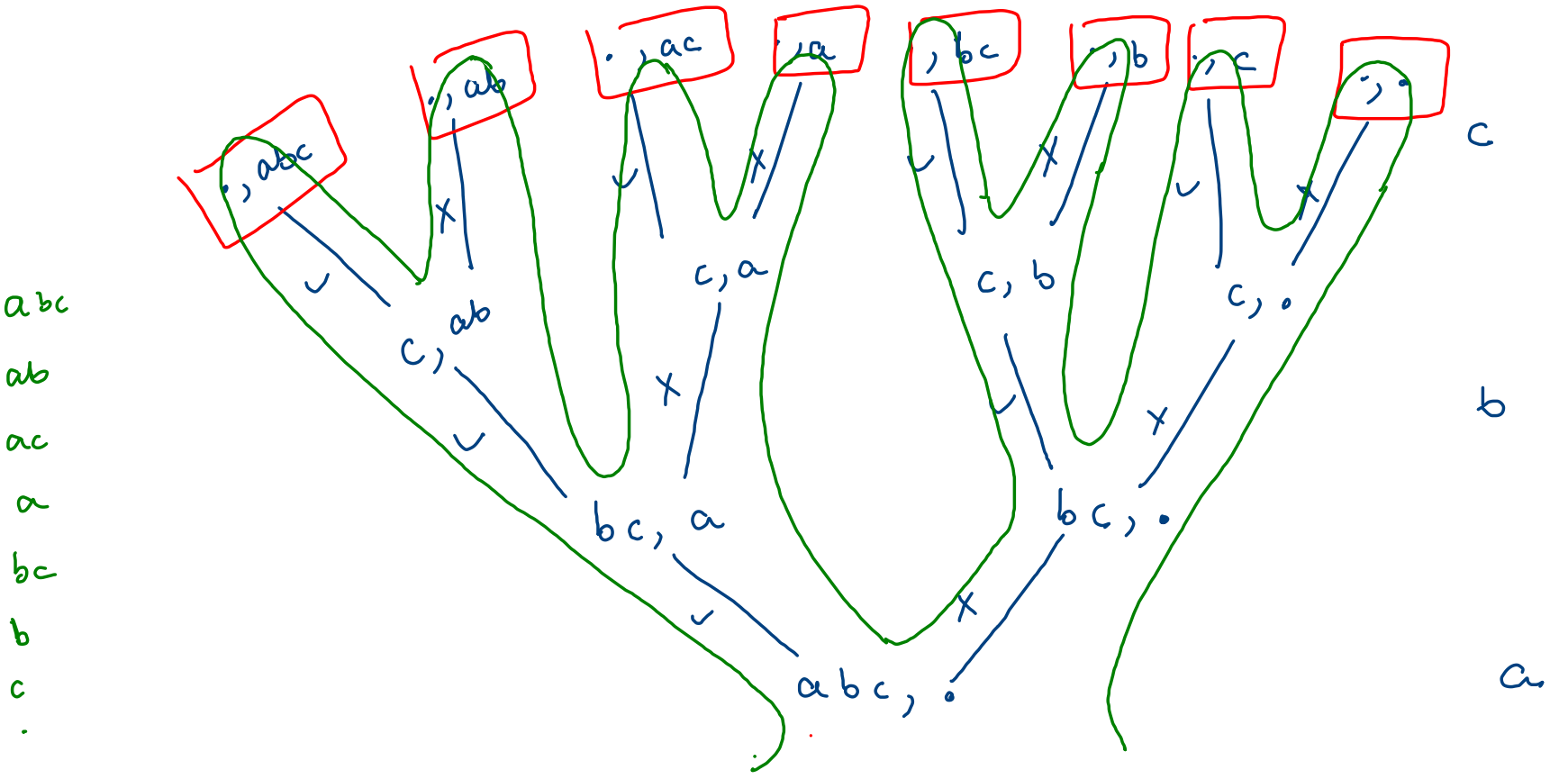


Str = abc

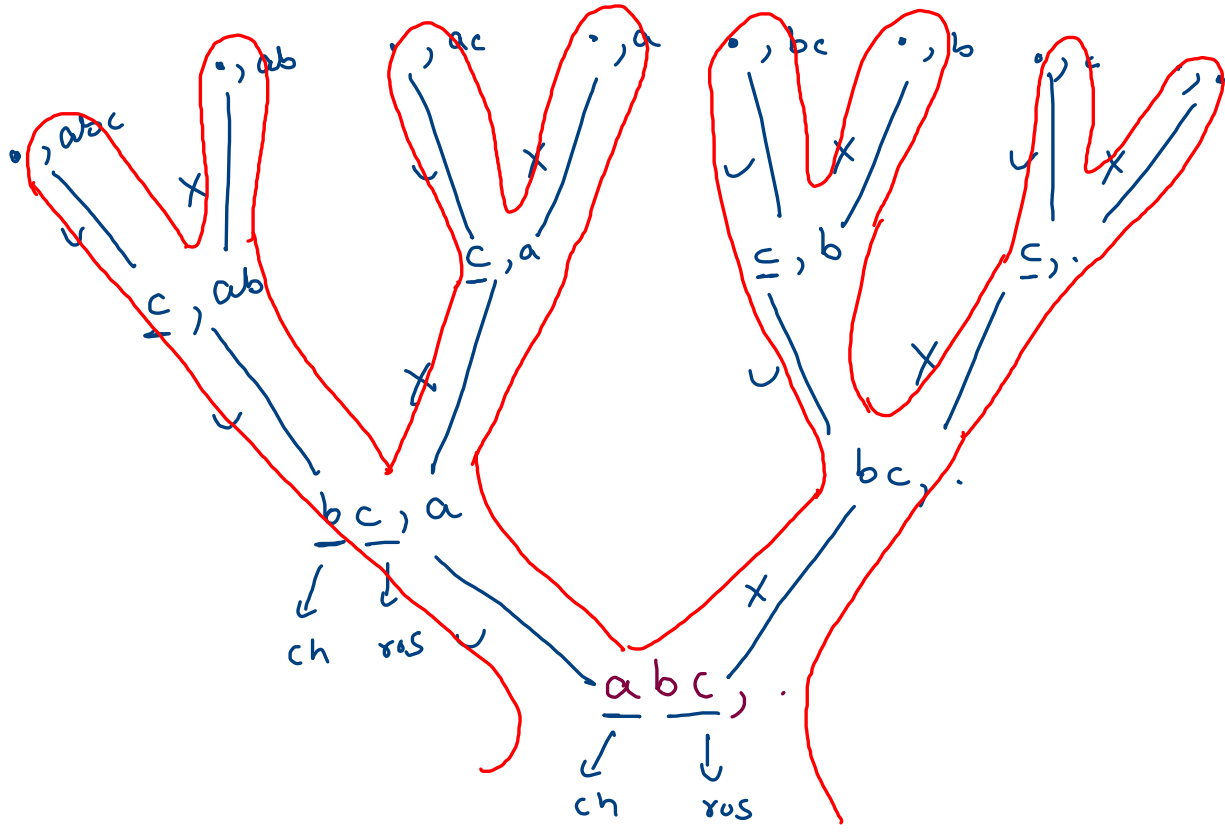

c
bc
abc
[.]
[., c]
[., c]
[., c, b, bc]
[., c, b, bc]
[., c, b, bc,
a, ac, ab, abc]


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

Str = a b c

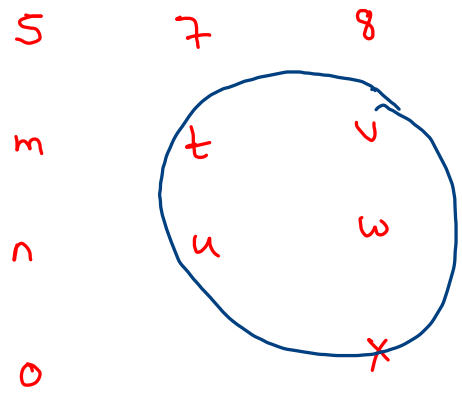


```java
public static void printSS(String str, String ans) {
    if(str.length() == 0) {
        System.out.println(ans);
        return;
    }

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

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

    //ch -> no choice
    printSS(ros, ans);
}
```



str: 5 7 8

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



m [tv, tw, tx, uv, uw, ux]

n [tv, tw, tx, uv, uw, ux]

o [tv, tw, tx, uv, uw, ux]

578
578, .
578, m
578, n
578, o
8, mt
8, mu
8, nt
8, nu
8, ot
8, ou
mtv
mtw
mtx
muv
muw
mux
ntv
ntw
ntx
nuv
nuw
nux
otv
otw
otx
ouv
ouw
oux
8
7
5

```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);

    String mycode = code[ch-'0'];

    for(int i=0; i < mycode.length();i++) {
        char mch = mycode.charAt(i);

        printKPC(ros,asf + mch);
    }
}
```

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