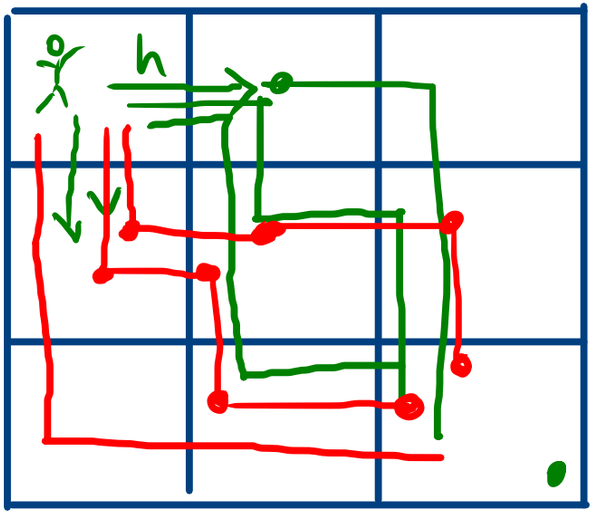


h

v

[hhvv

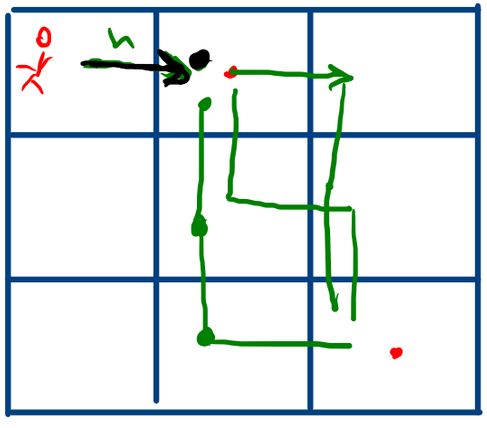
hvhv

hvvh

vhhv

vhvh

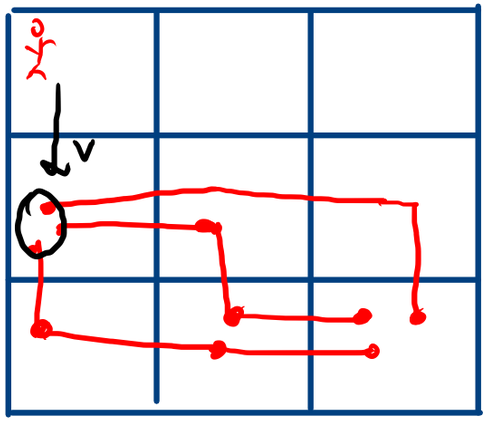
vvhh]



h hvv

h v hv

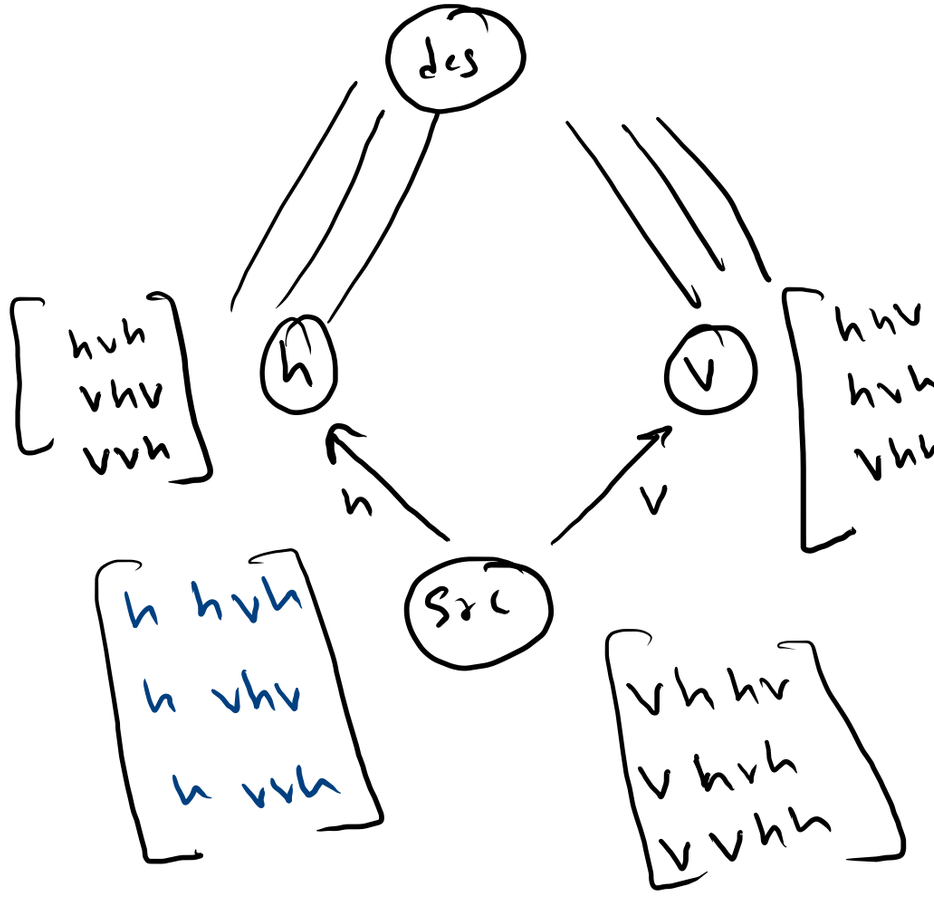
h vvh

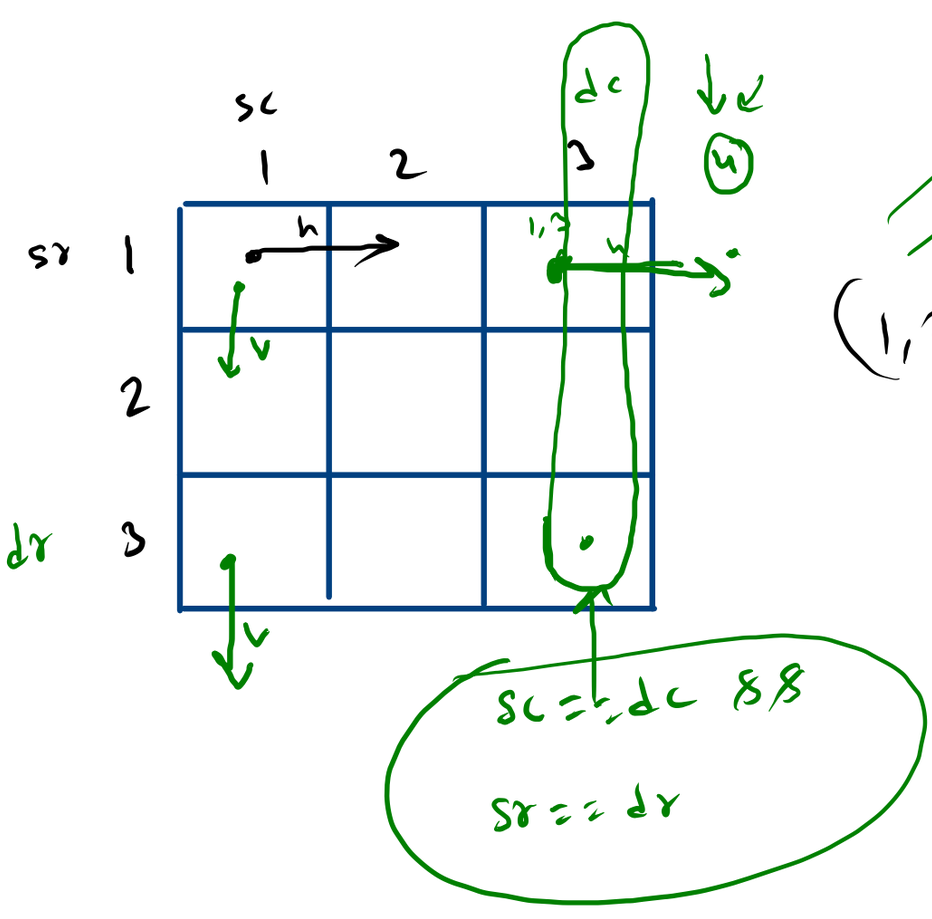


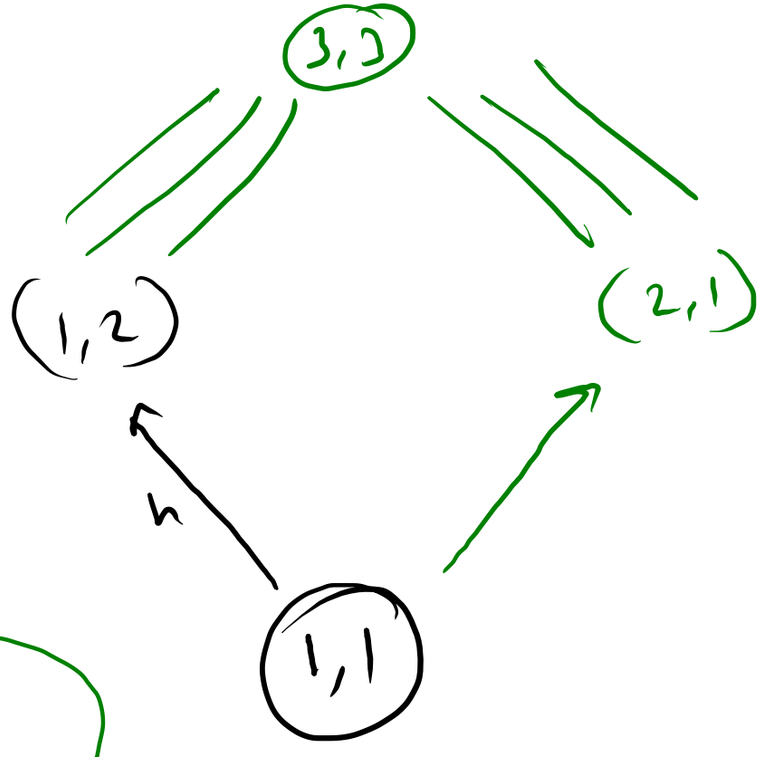
v hhv

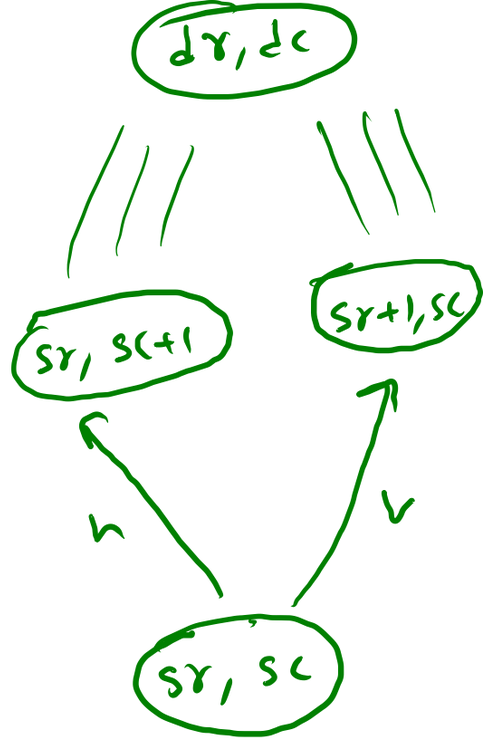
v hvh

v vhh

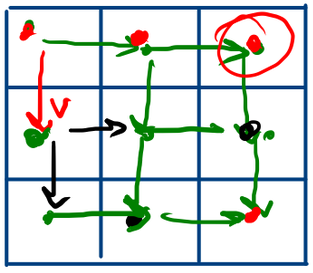


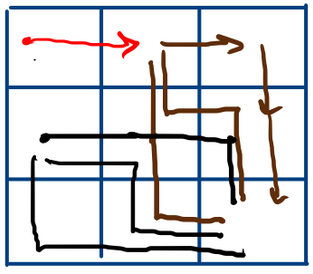




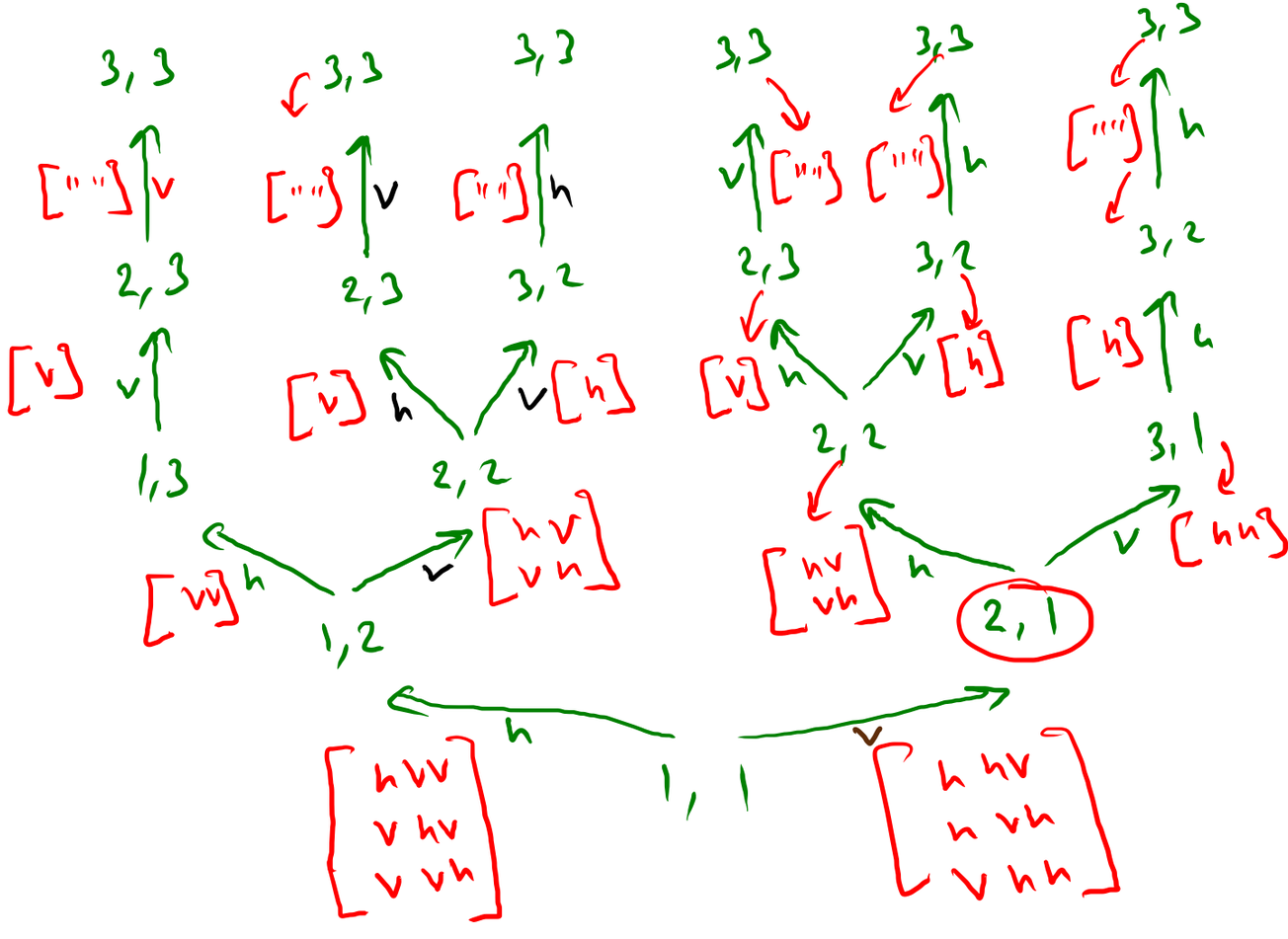



```java
public static ArrayList<String> getMazePaths(int sr, int sc, int dr, int dc) {
    return null;
}
```





```java
public static ArrayList<String> getMazePaths(int sr, int sc, int d

    if(sr == dr && sc == dc){
        ArrayList<String> paths = new ArrayList<>();
        paths.add("");
        return paths;
    }

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

    if(sc+1 <= dc){
        ArrayList<String> faith = getMazePaths(sr, sc+1, dr, dc);
        for(String s: faith){
            paths.add("h"+s);
        }
    }

    if(sr+1 <= dr){
        ArrayList<String> faith = getMazePaths(sr+1, sc, dr, dc);
        for(String s: faith){
            paths.add("v"+s);
        }
    }

    return paths;
}
```



h1 h1 v1 v1

h1 h1 v2

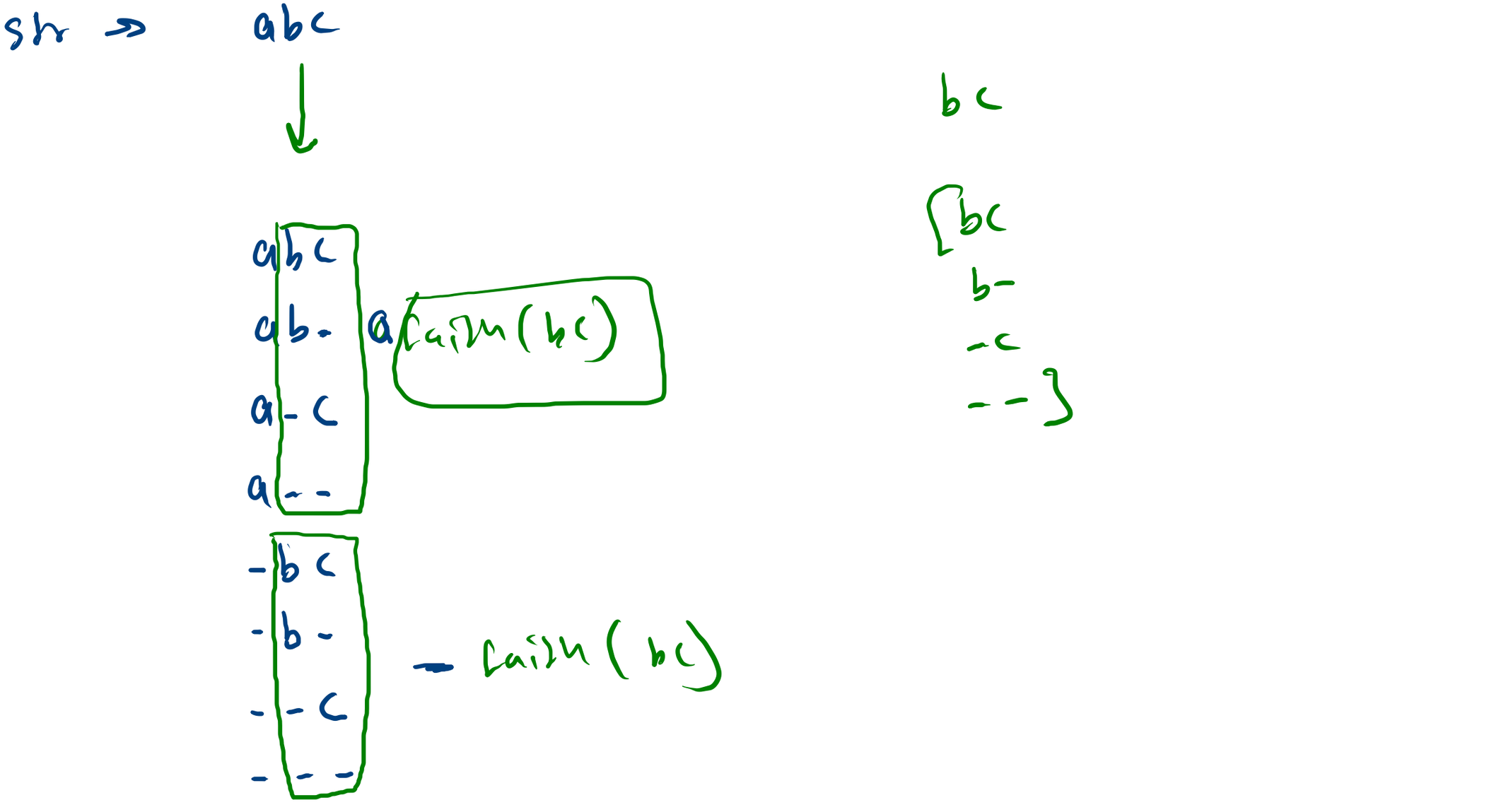
h2 v1 v1

h2 v2

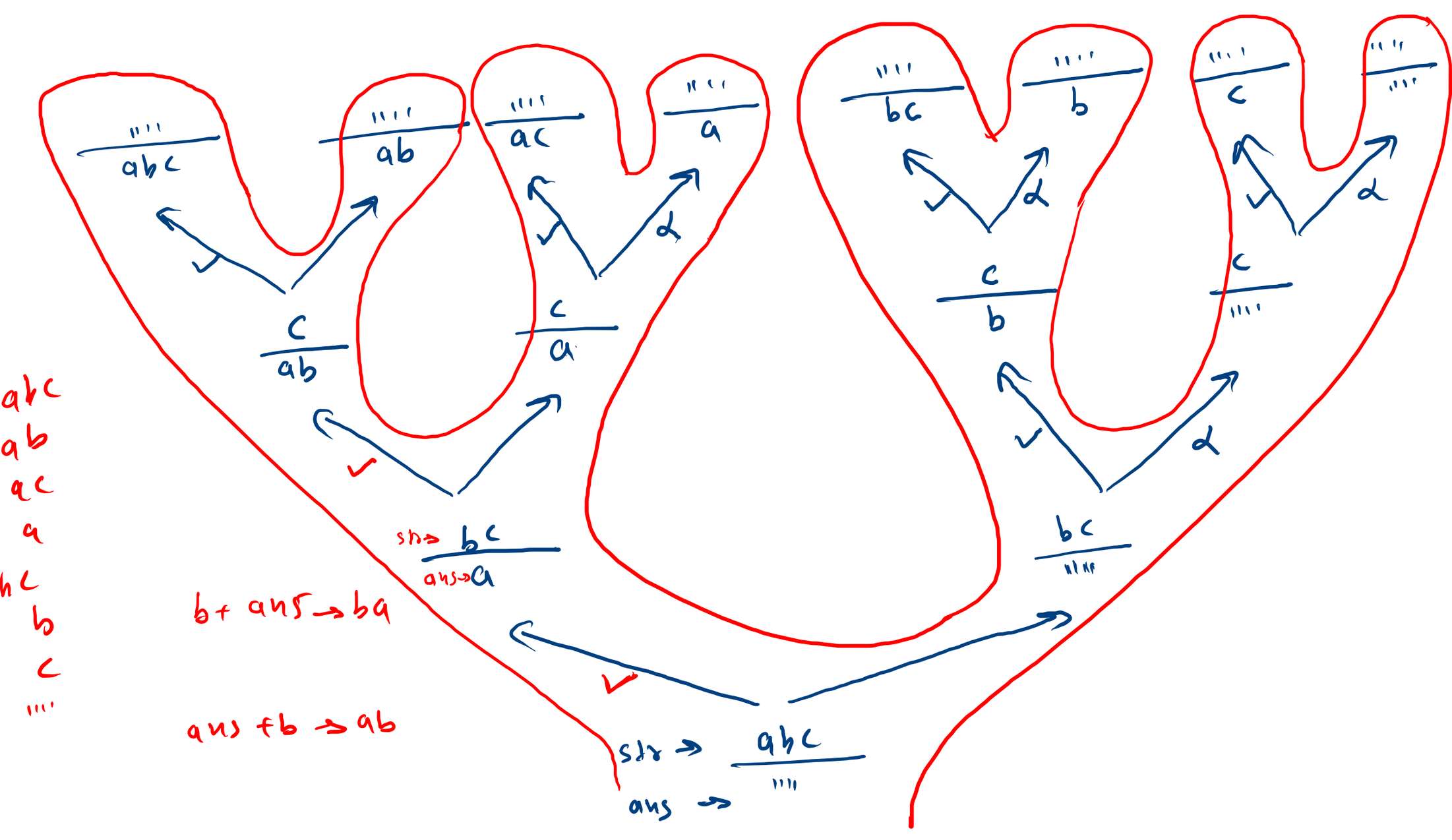
d1 h1 v1

h1 v1 d1

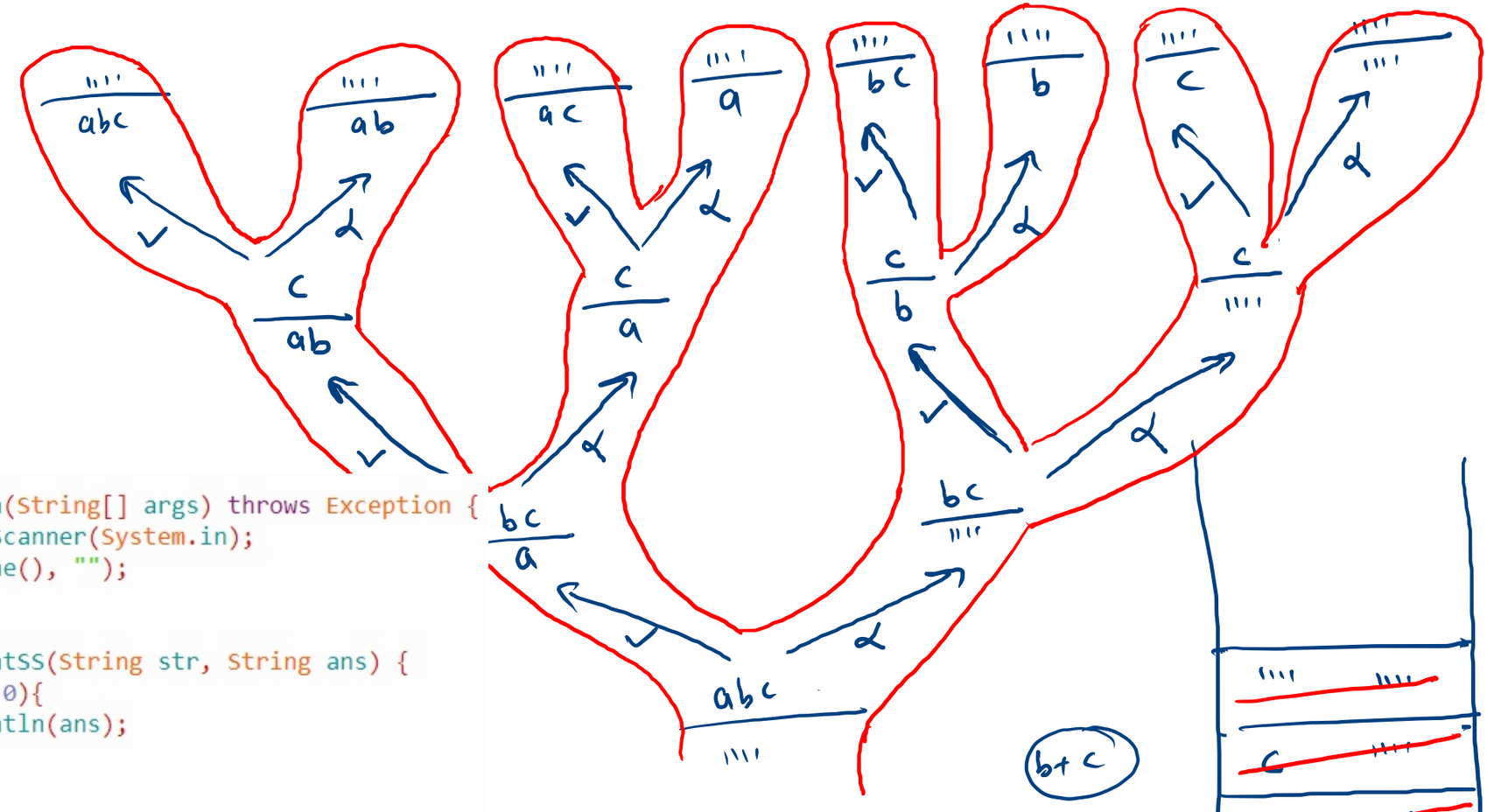
d2


str =>
abc
abc
ab-
a-c
a--
a
call(bc)
-bc
-b-
--c

-
call(bc)
bc
{bc
b-
-c
--}

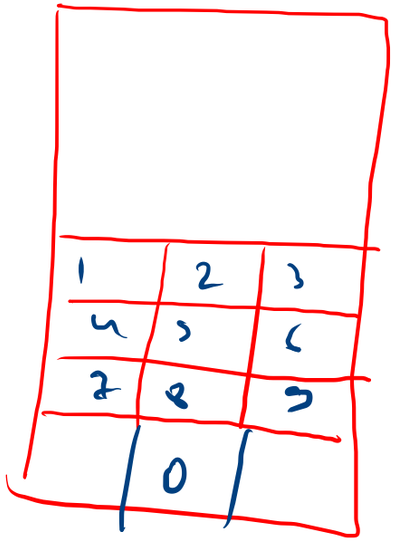

abc
ab
ac
a
bc
b
c
""
abc
""
ab
ac
a
c
ab
c
a
bc
""
b
bc
b
c
""
c
b
c
""
c
a
c
a
c
a
str→ bc
ans→a
b+ ans → ba
ans +b → ab
str → abc
""
ans →

abc ✓✓✓
ab ✓✓✗
ac ✓✗✓
a ✓✗✗
bc ✗✓✓
b ✗✓✗
c
""

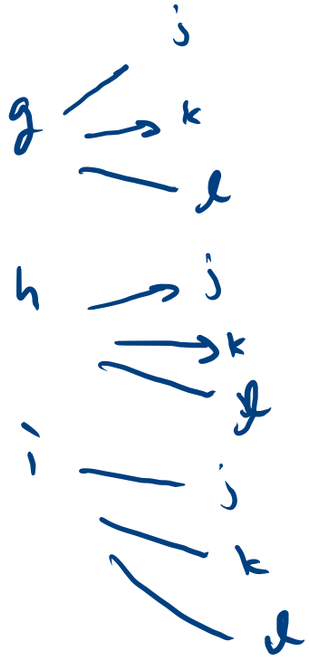


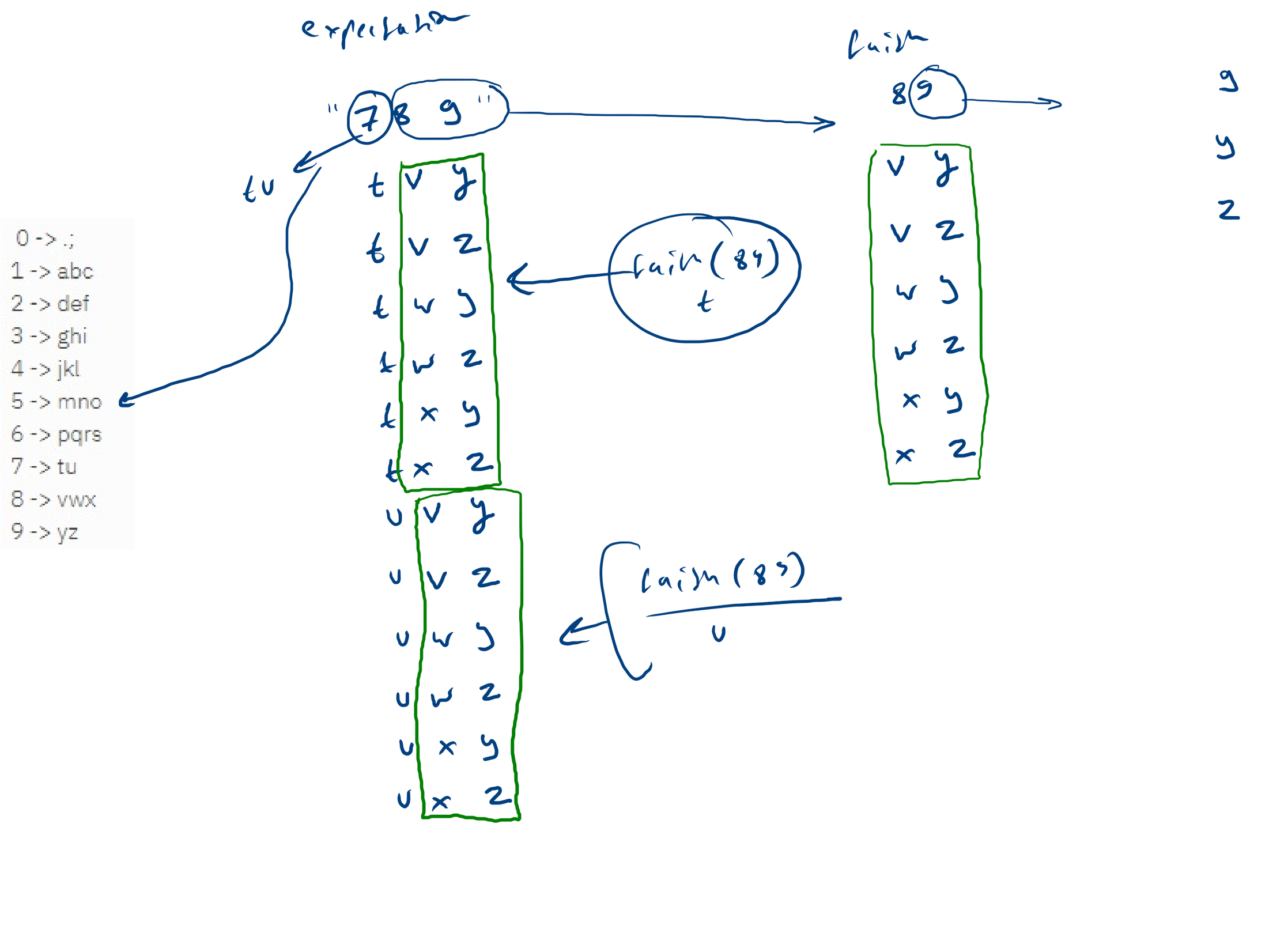

```java
public static void main(String[] args) throws Exception {
    Scanner scn = new Scanner(System.in);
    printSS(scn.nextLine(), "");
}

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

    String fstr = str.substring(1);
1   printSS(fstr, ans+str.charAt(0));
2   printSS(fstr, ans);
}
```

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



34



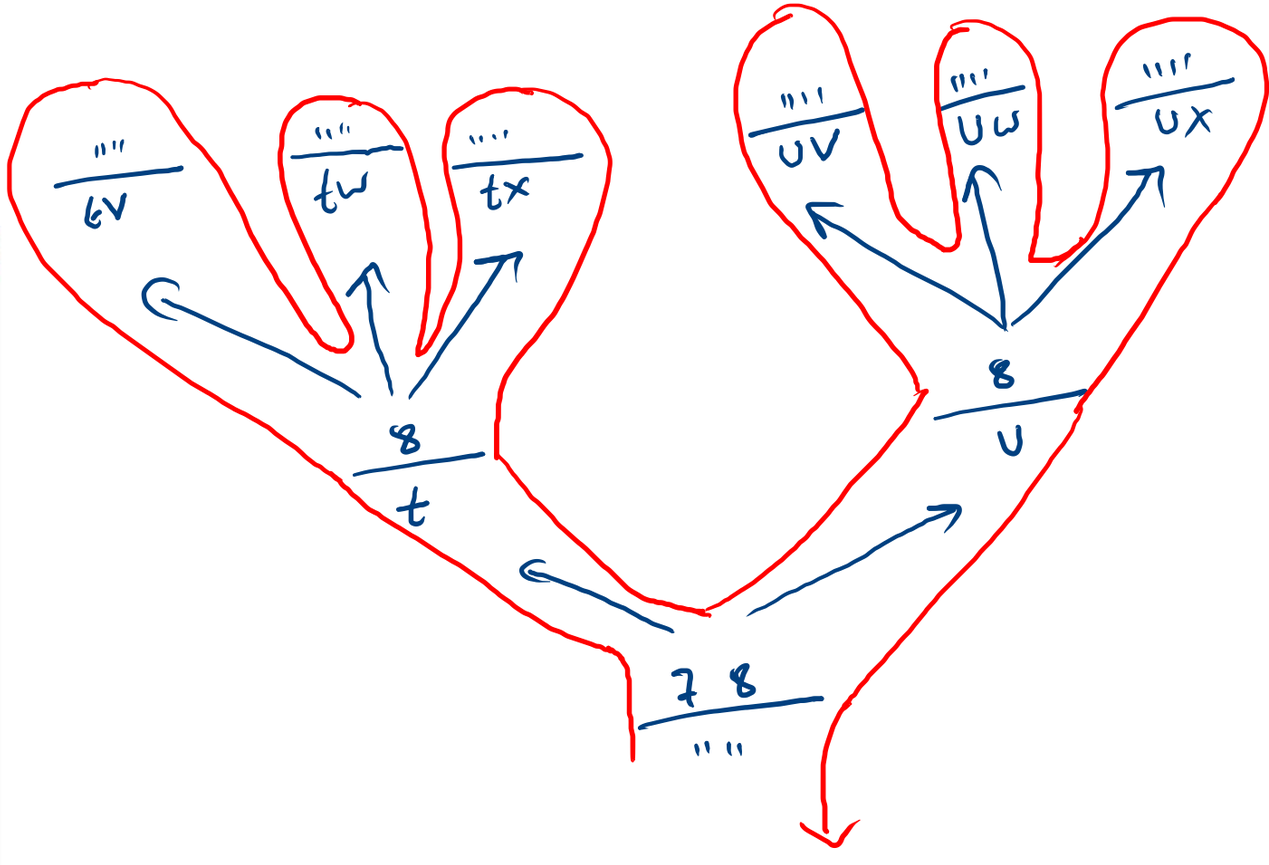

explanation
"78 9"
tu
0 -> .;
1 -> abc
2 -> def
3 -> ghi
4 -> jkl
5 -> mno
6 -> pqrs
7 -> tu
8 -> vwx
9 -> yz
t v y
t v z
t w y
t w z
t x y
t x z
main(84)
t
main
89
v y
v z
w y
w z
x y
x z
u v y
u v z
u w y
u w z
u x y
u x z
main(83)
u
9
y
z

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

tv
tw
tx
uv
uw
ux

vwx

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

    int index = str.charAt(0)-'0';
    String code = keys[index];

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

        printKPC(fstr, asf+ch);
    }

}
```



n = 3

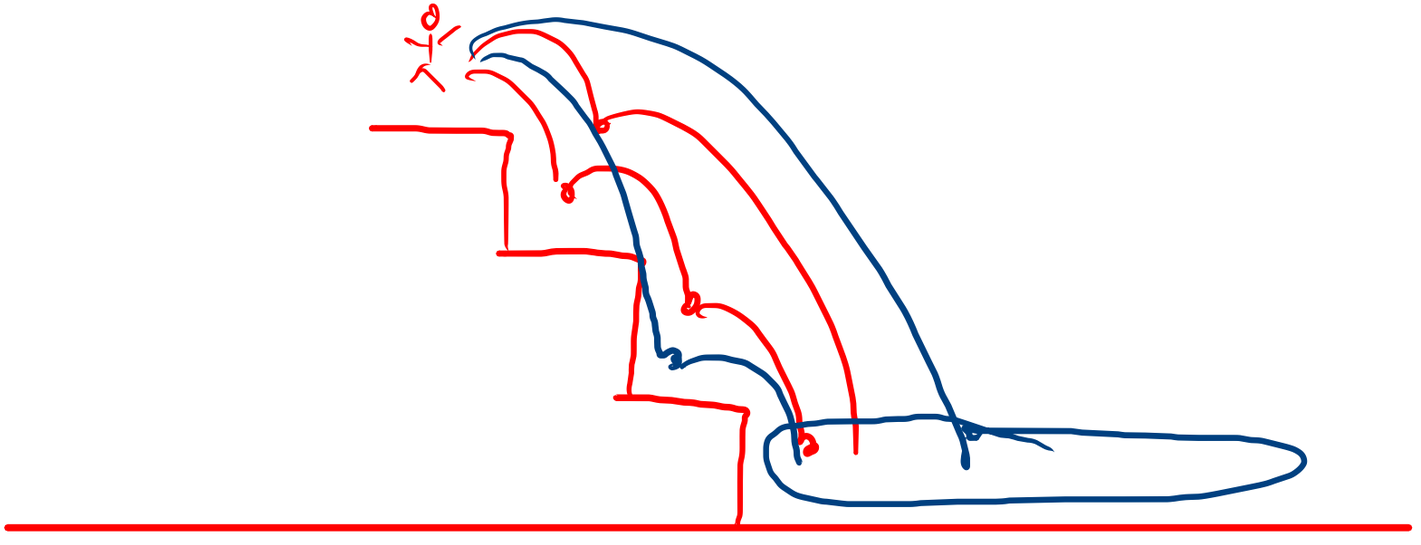
[1, 2, 3]

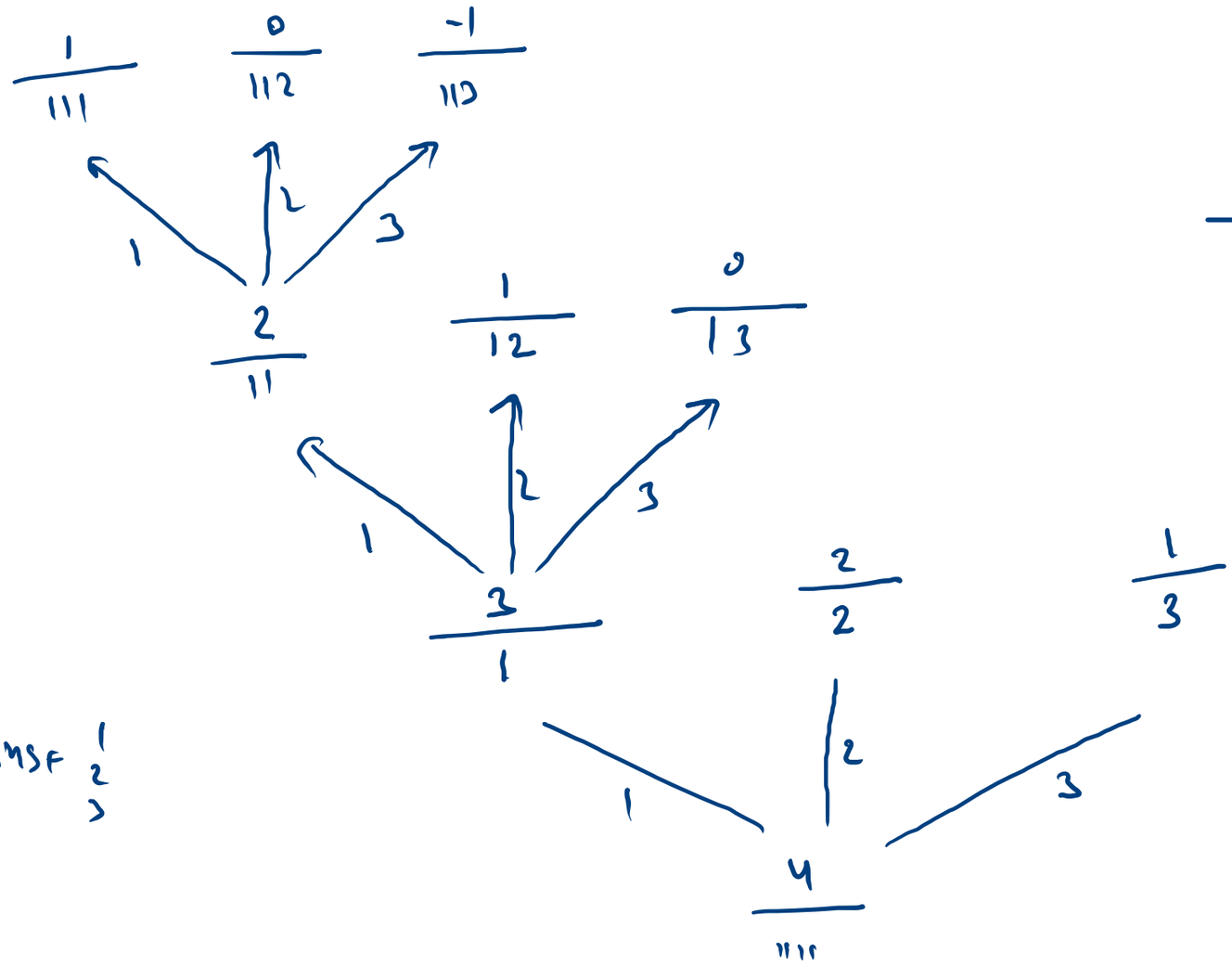
111

12

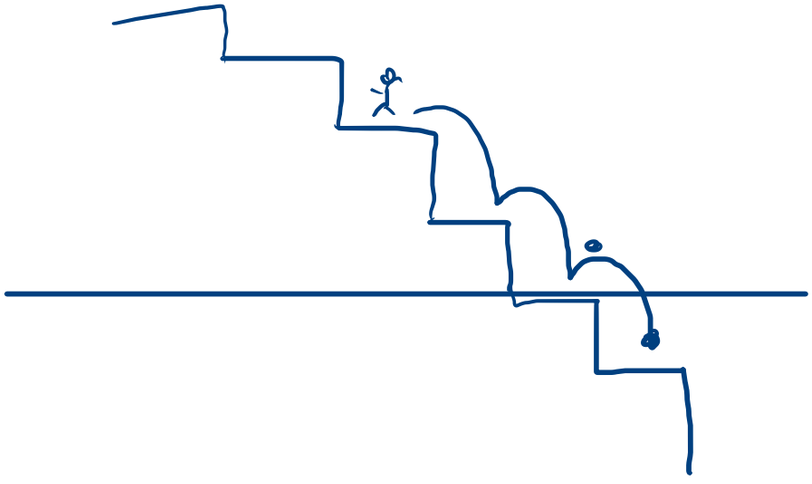
21

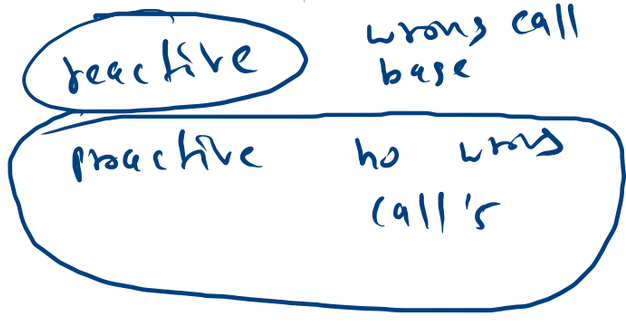
3



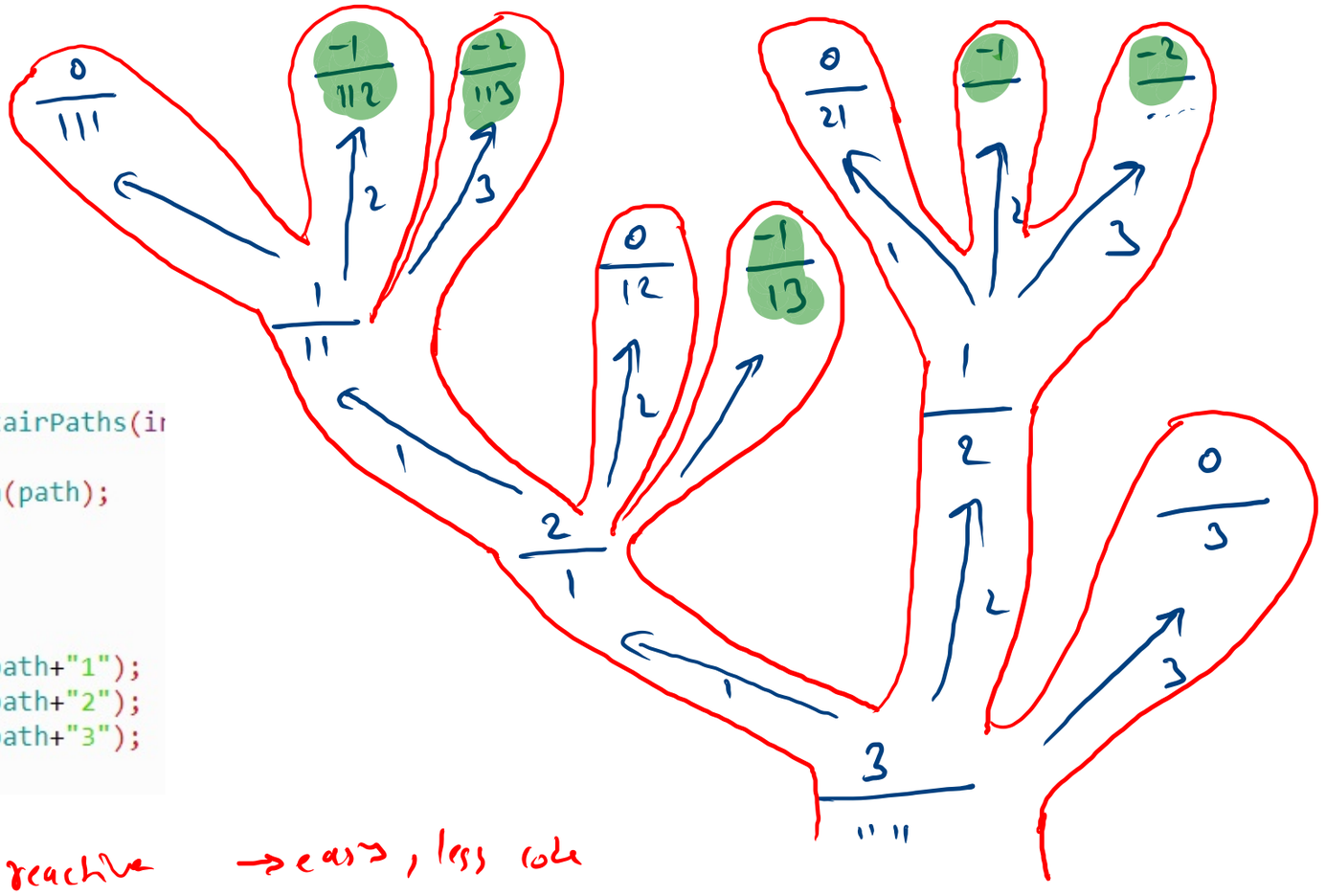

1
111
0
112
-1
113
1
2
3
2
11
1
12
0
13
1
2
3
3
1
2
2
1
3
2
1
3
4
111

ans: 1
2
3




reactive
wrong call base
proactive
no wrong call's

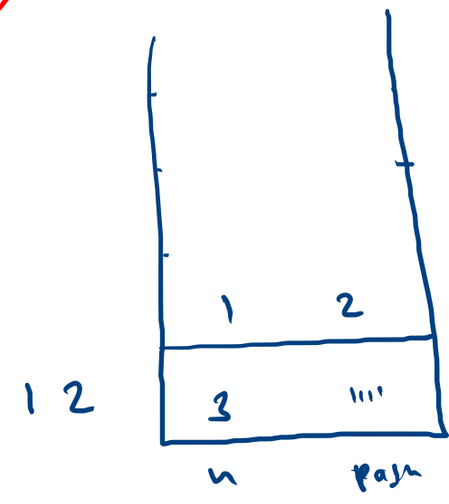
111
12
21
3



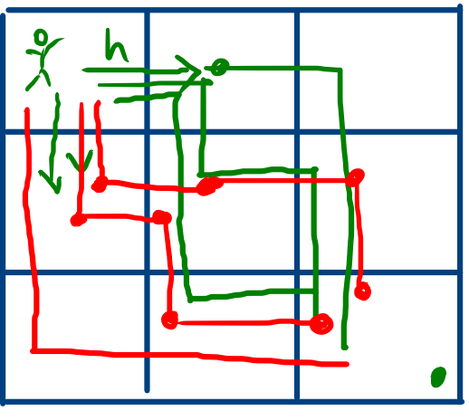

```java
public static void printStairPaths(i
    if(n == 0){
        System.out.println(path);
        return;
    }
    if(n<0)return;

    printStairPaths(n-1, path+"1");
    printStairPaths(n-2, path+"2");
    printStairPaths(n-3, path+"3");
}
```

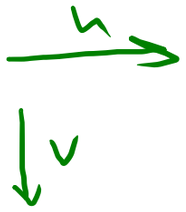
reachable → easy, less code

possible → more code, less time

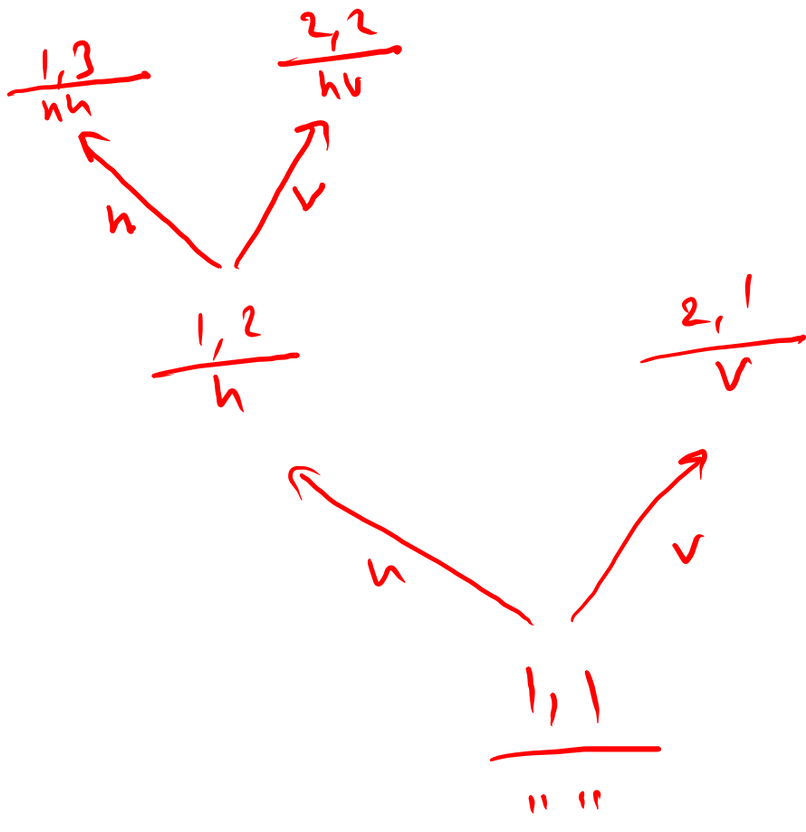


1 2



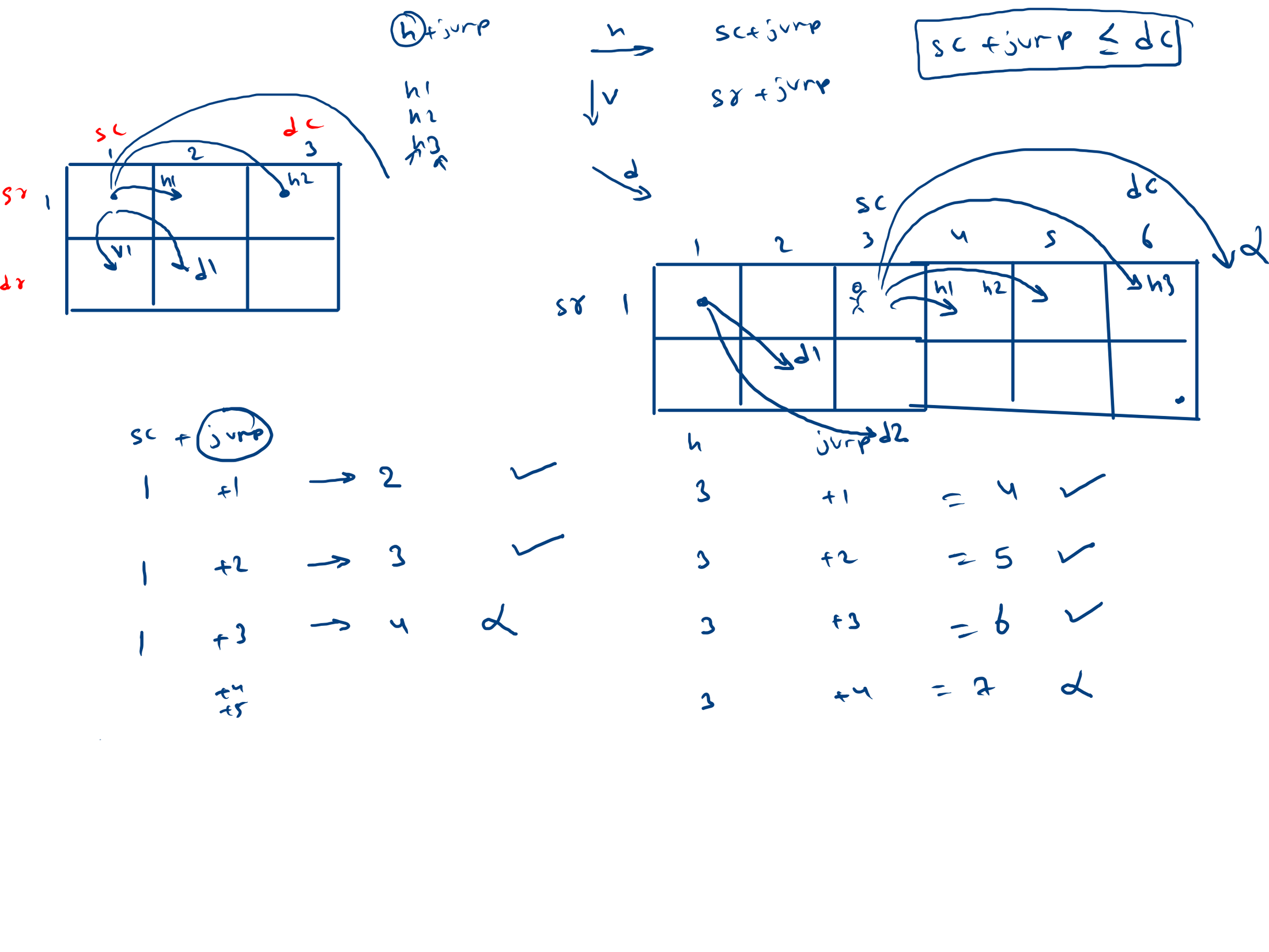


[hhvv
hvhv
hvvh
vhhv
vhvh
vvhh]



if (sc+1 ≤ dc)

if (sr+1 ≤ dr)



sc + (jump)

1	+1	→ 2	✓	
1	+2	→ 3	✓	
1	+3	→ 4	✓	α
	+4			
	+5			

3	+1	= 4	✓	
3	+2	= 5	✓	
3	+3	= 6	✓	
3	+4	= 7	✓	α

sc	sr	Jump	sc ≤ dc && sr ≤ dr
2	1	+1	3, 2
		+2	4, 3