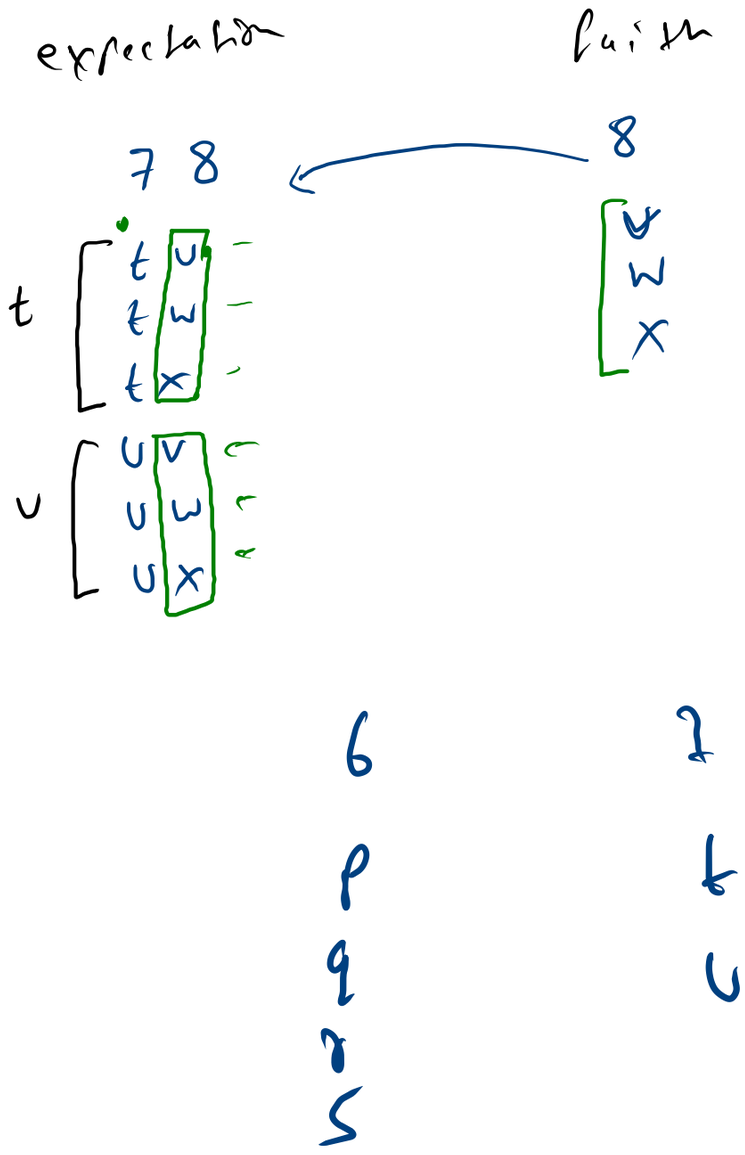
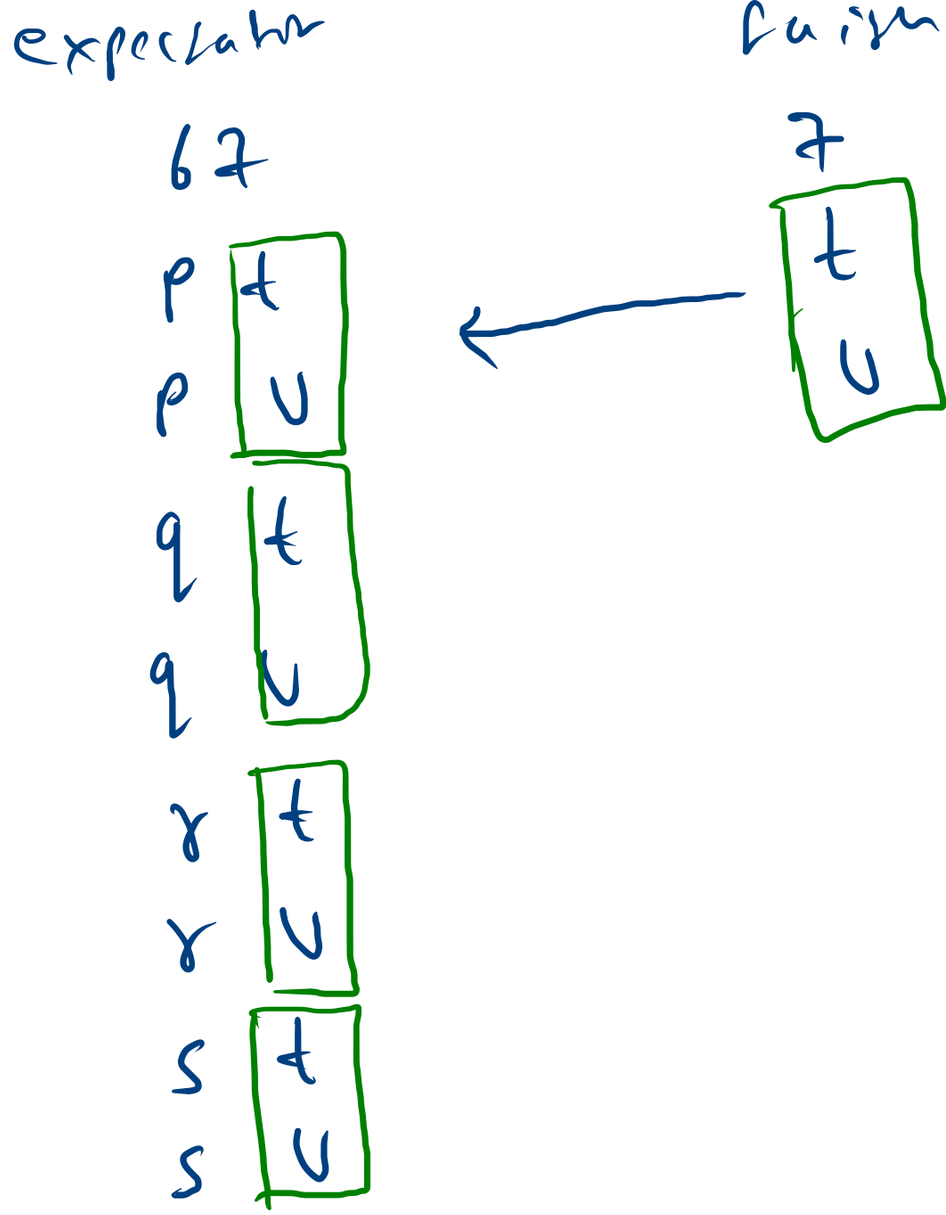
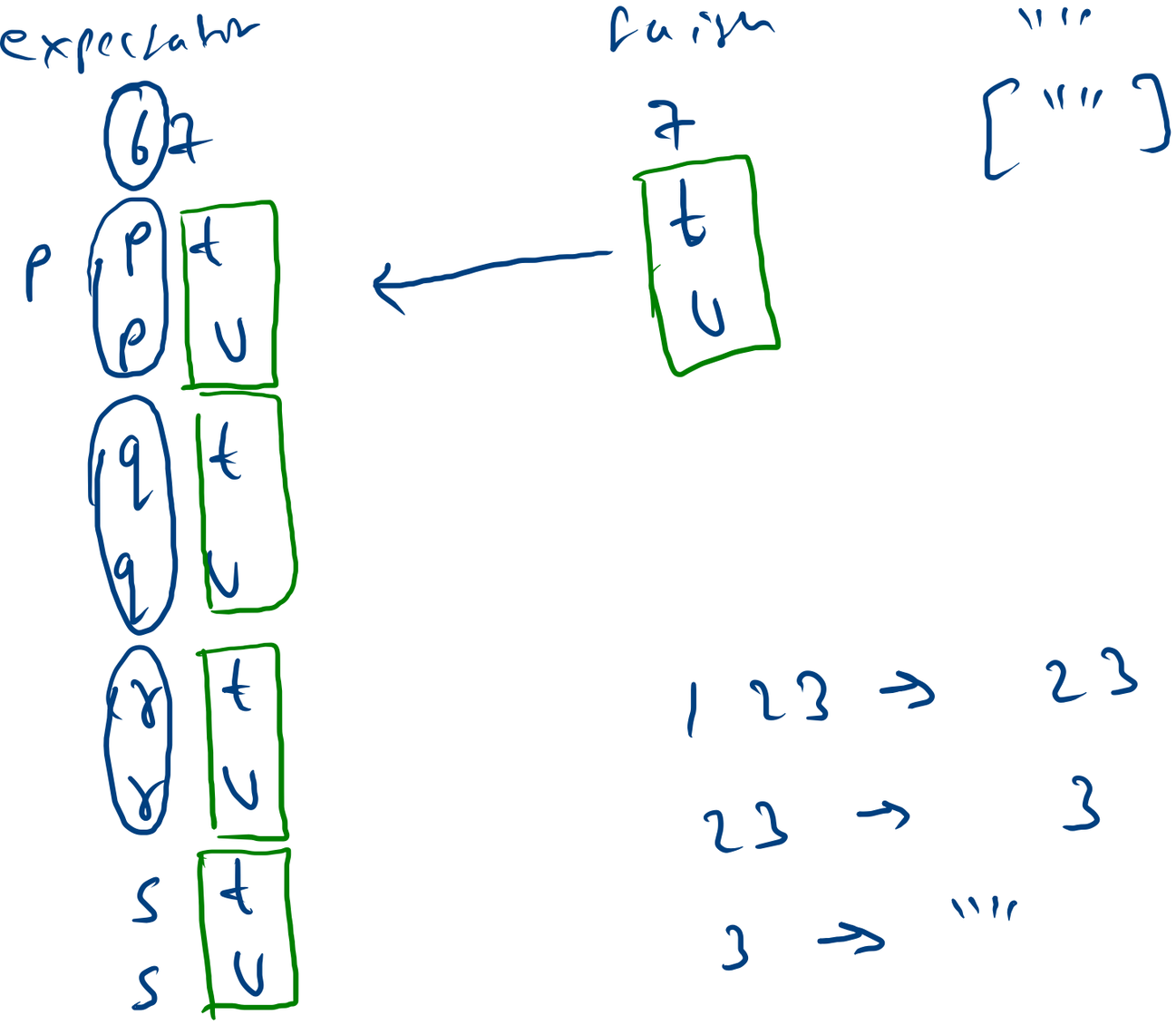
0 -> .;
1 -> abc
2 -> def
3 -> ghi
4 -> jkl
5 -> mno
6 -> pqrs
7 -> tu
8 -> vwx
9 -> yz







str

faithans → faith(str.ss(i))

myans → [ ]

mycode => " pqrs " // other index

ch ← mycode {

for(string s : faithans) {

myans.add(ch + s)

}

p + t

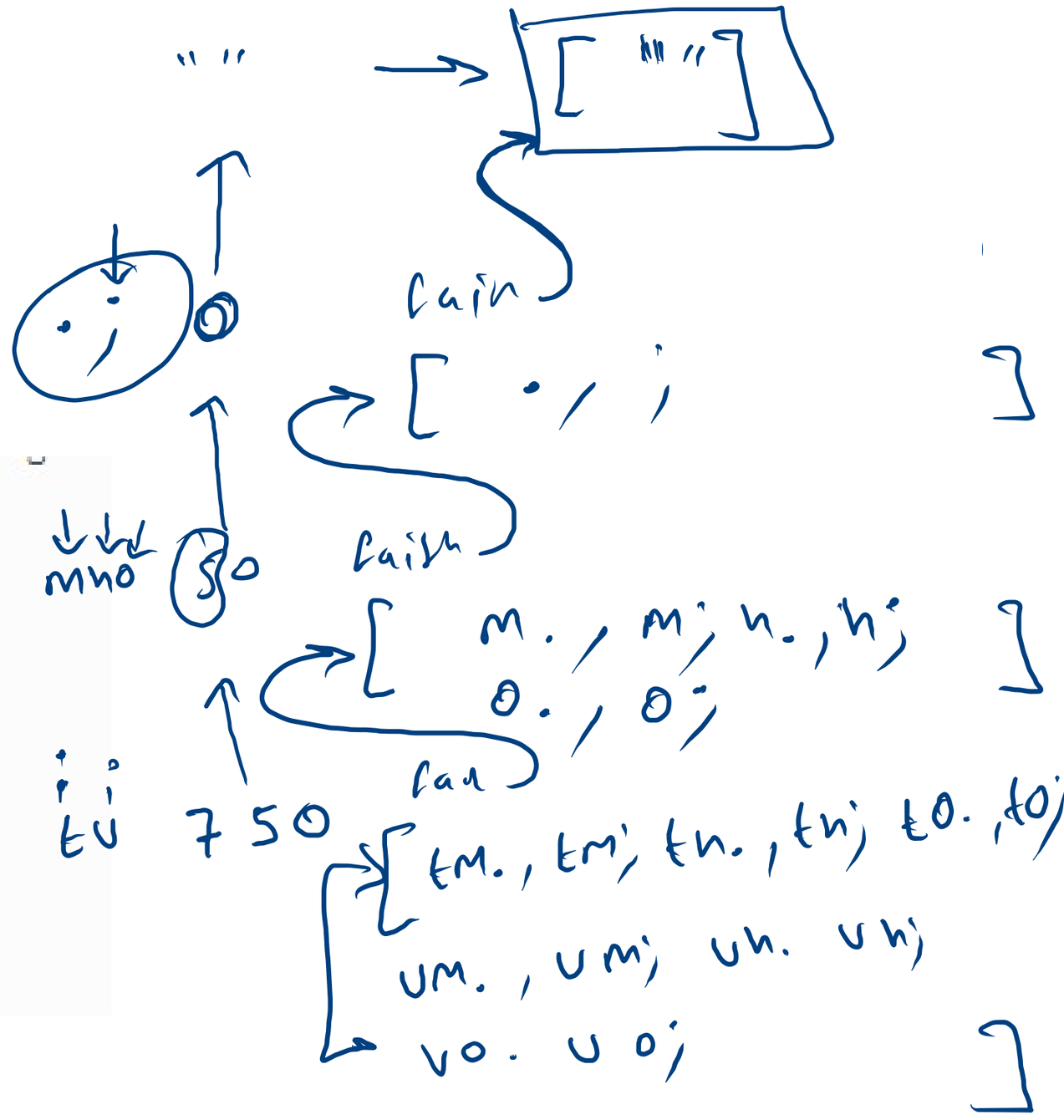
p + u

q + t

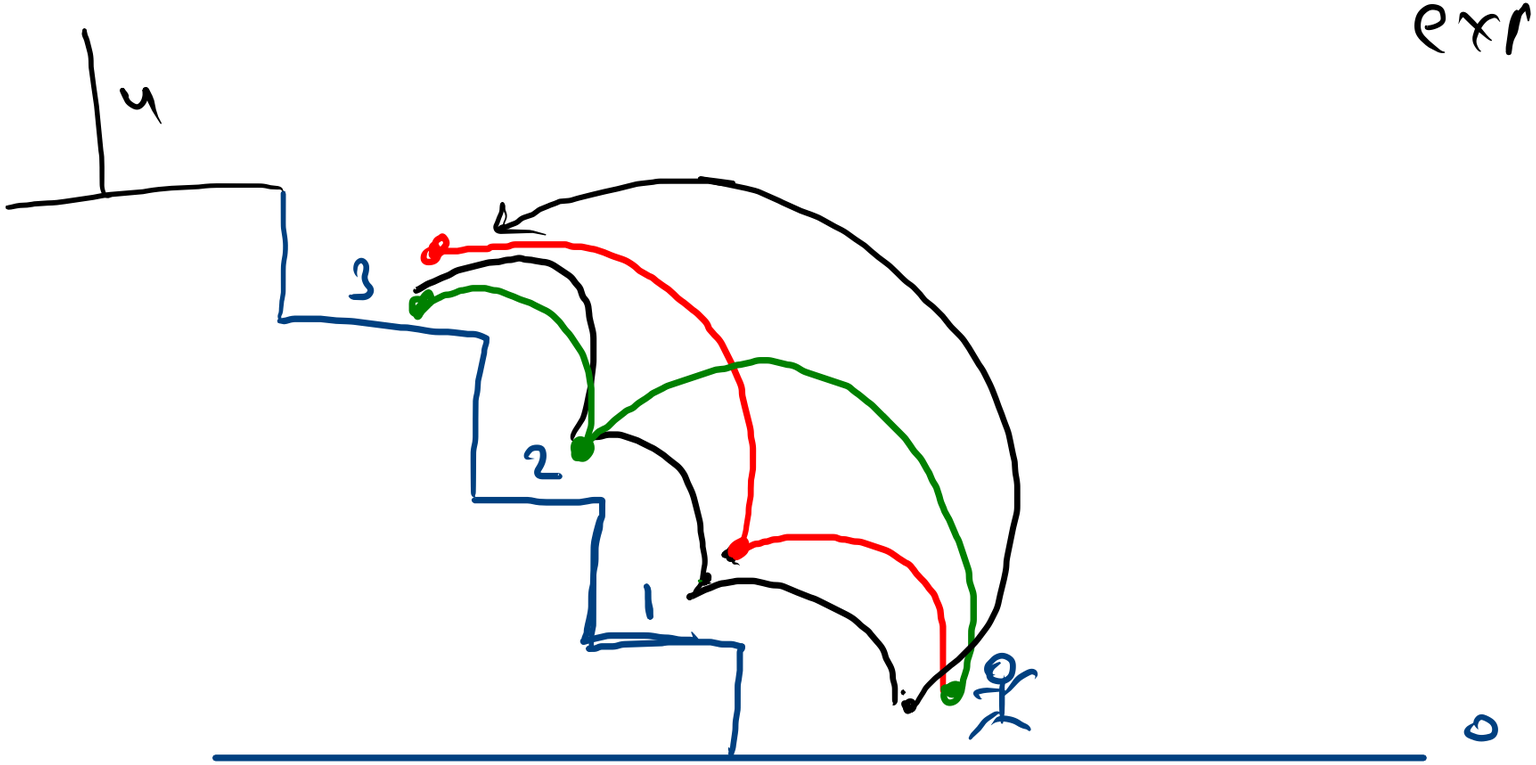
q + u

```java
public static ArrayList<String> getKPC(String str) {
    if(str.length() == 0){
        ArrayList<String> ans = new ArrayList<String>();
        ans.add("");
        return ans;
    }

    ArrayList<String> faithAns = getKPC(str.substring(1));

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

    char firstchar = str.charAt(0);
    String myCode = codes[firstchar - '0'];

    for(int i=0;i<myCode.length();i++){
        char ch = myCode.charAt(i);
        for(String s: faithAns){
            myAns.add(ch+s);
        }
    }

    return myAns;
}
```

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



h -> 3



-> [1, 2, 3]

explain n=3

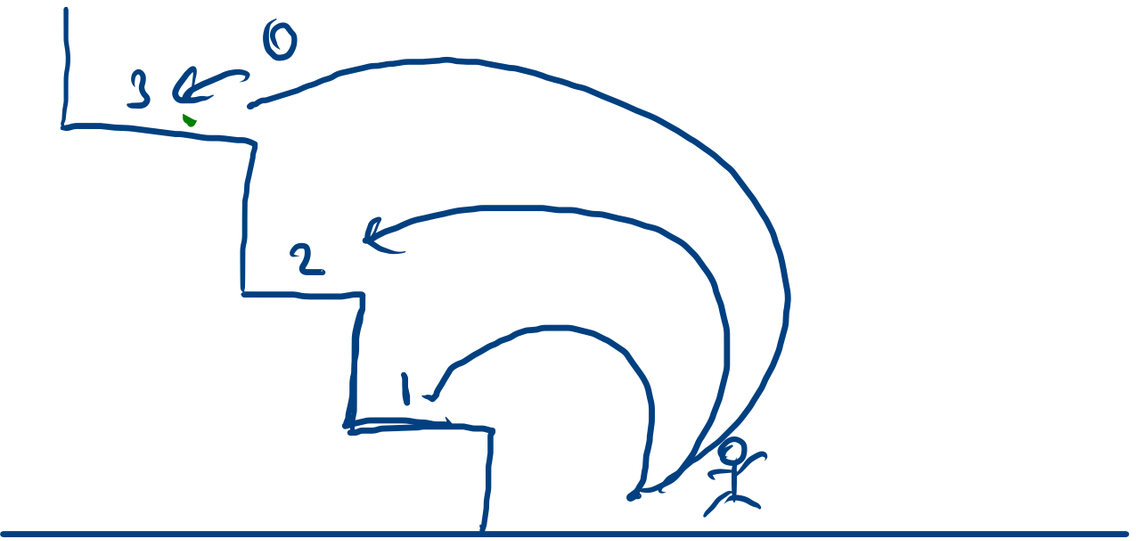
"1 1 1"

"1 2"

"2 1"

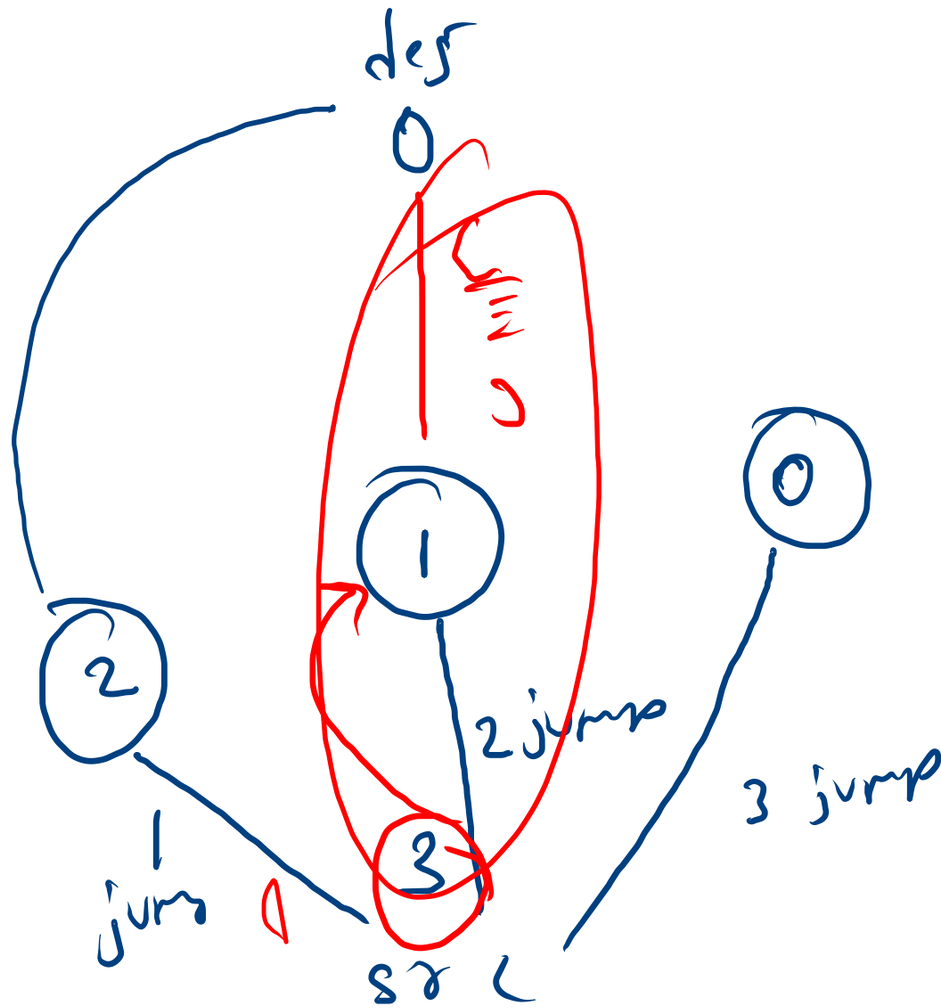
"3"

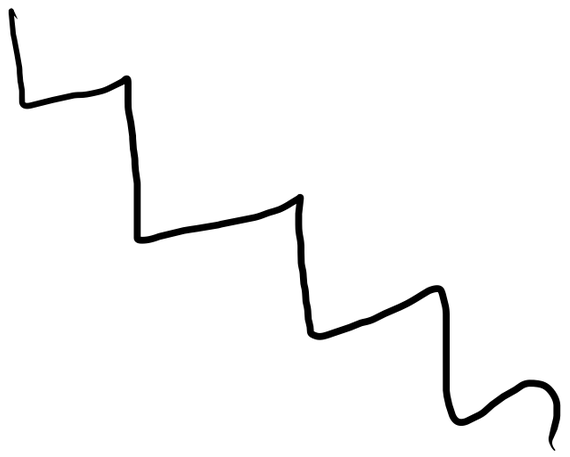
Lower level



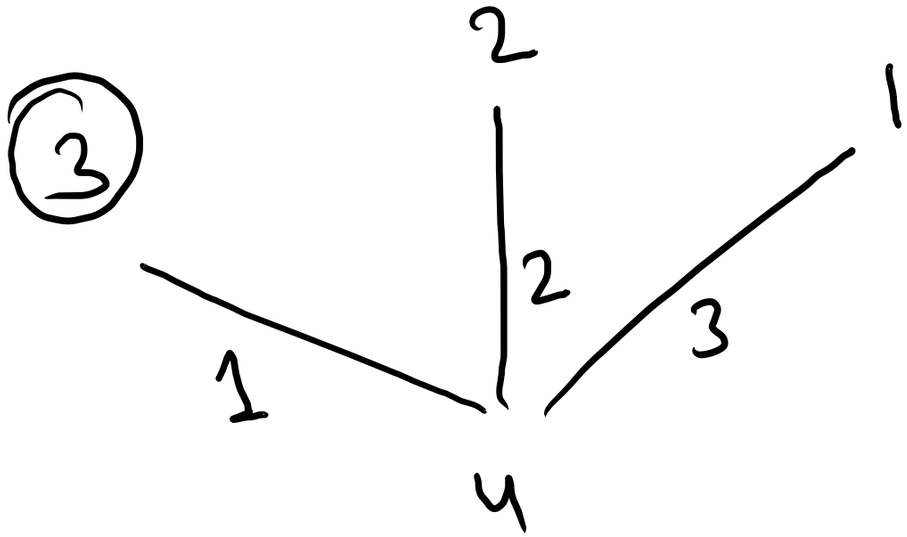
1 jmp 2 jmp

3 jmp



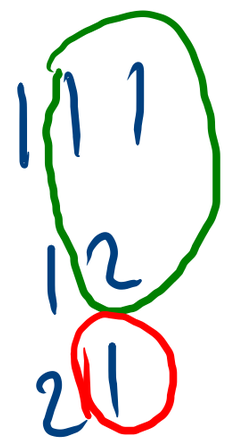


③

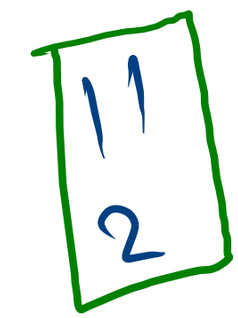


expectation

h = 3



3



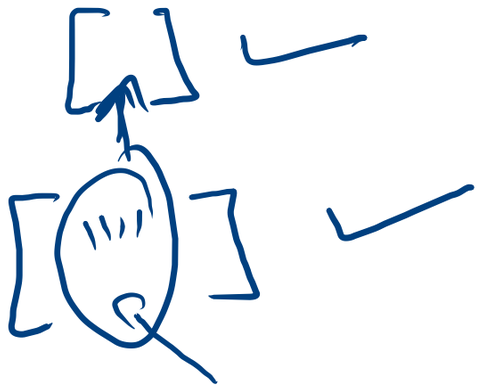
h = 2

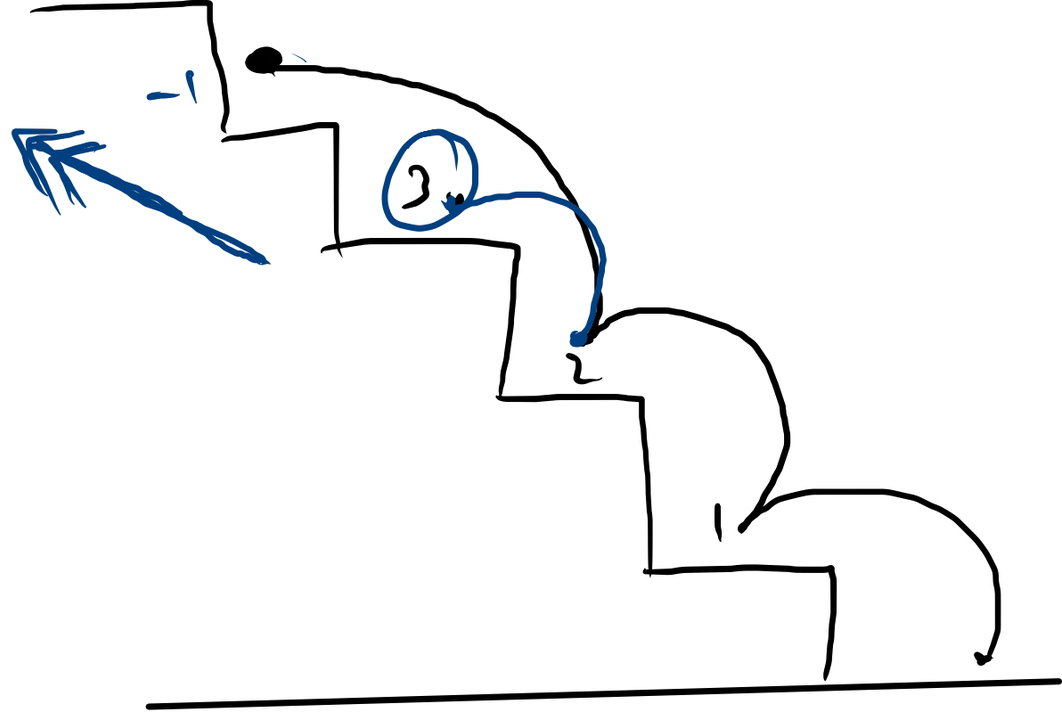
h = 1

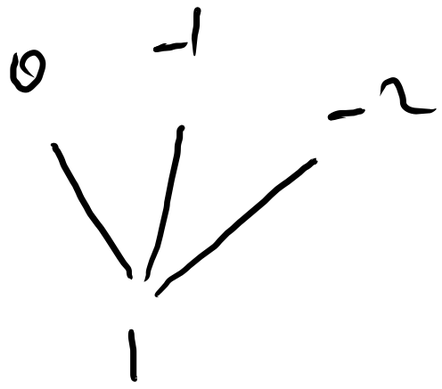
h = 0

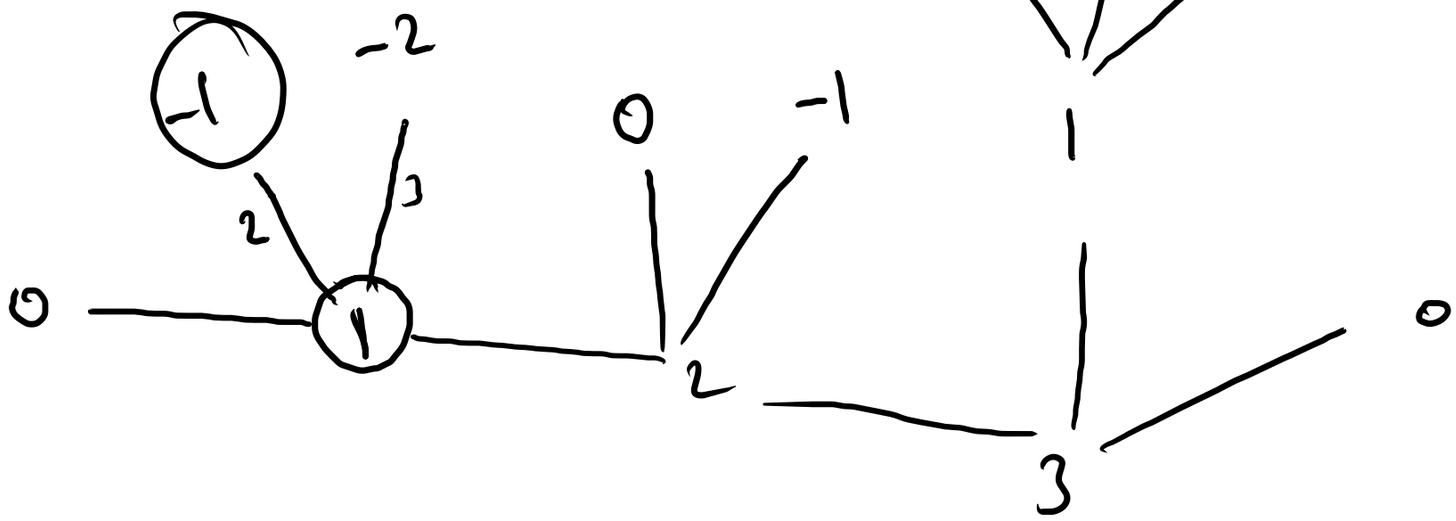
h=-1

h=-0



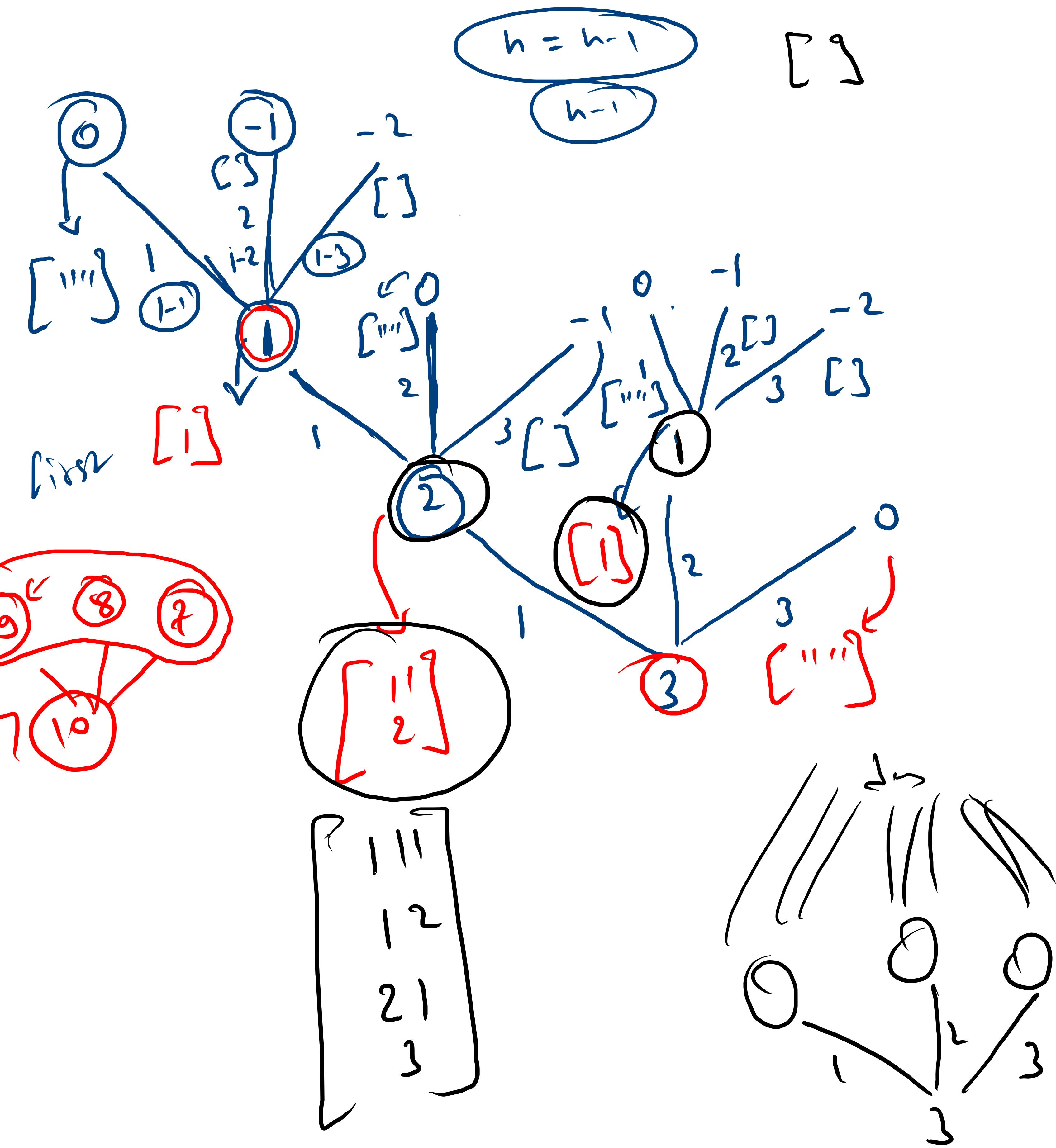






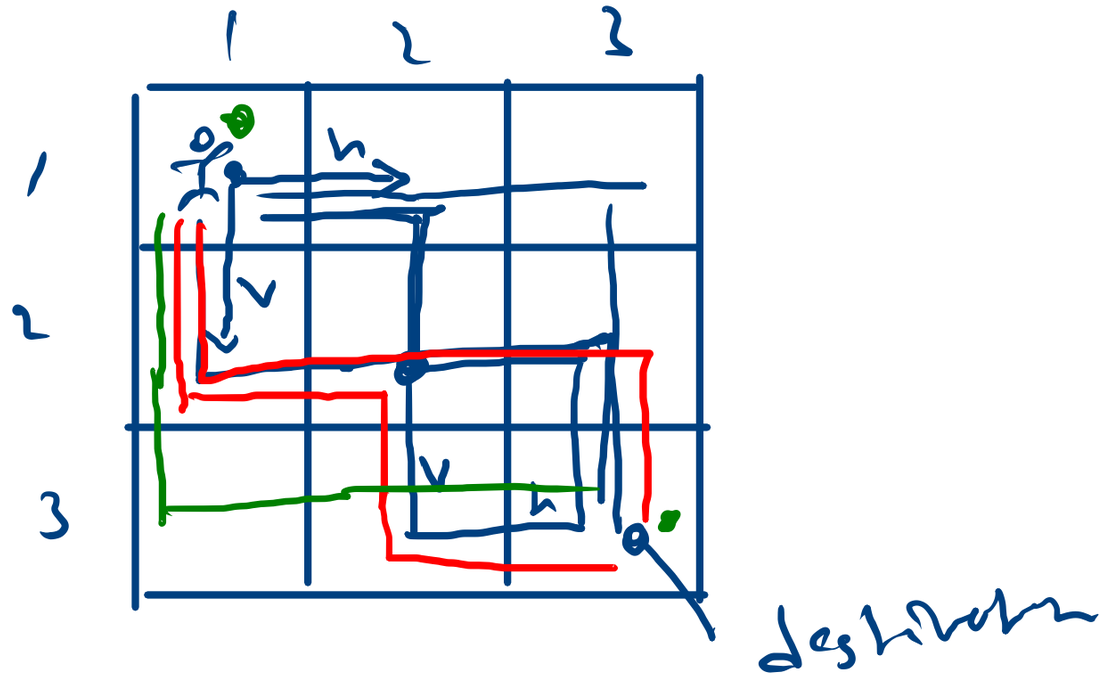
h-1
h-2
h-3

```java
public static ArrayList<String> getStairPaths(int n) {
    if(n==0){
        ArrayList<String> ans = new ArrayList<>();
        ans.add("");
        return ans;
    }
    if(n<0){
        ArrayList<String> ans = new ArrayList<>();
        return ans;
    }

    ArrayList<String> first = getStairPaths(n-1);
    ArrayList<String> second = getStairPaths(n-2);
    ArrayList<String> third = getStairPaths(n-3);

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

    for(String s: first){
        ans.add(1+s);
    }
    for(String s: second){
        ans.add(2+s);
    }
    for(String s: third){
        ans.add(3+s);
    }
    return ans;
}
```



h → 3

r → 3



h h v v

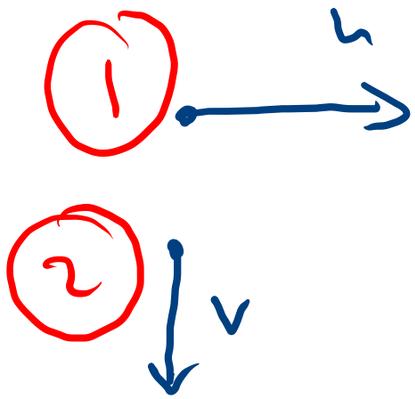
h v h v

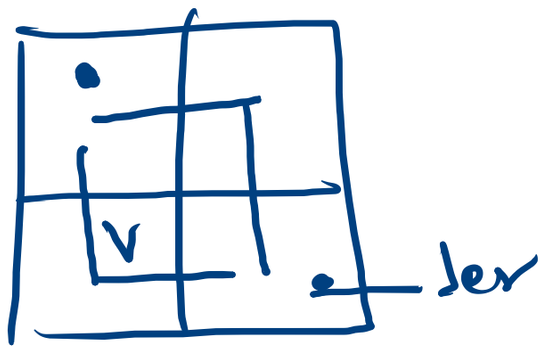
h v v h

v h h v

v h v h

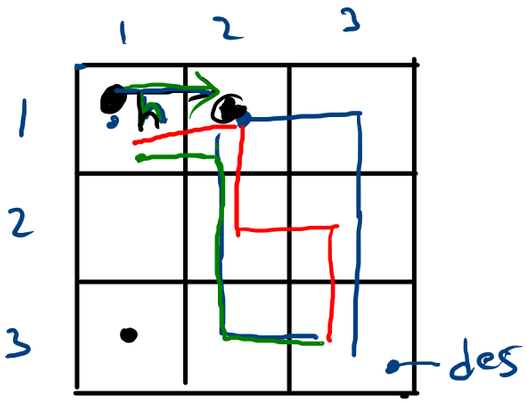
v v h h





h v

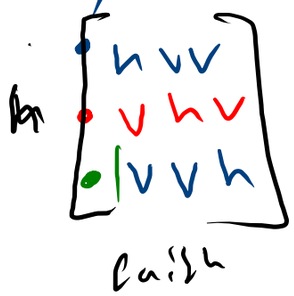
v h



1,1 → 1,2

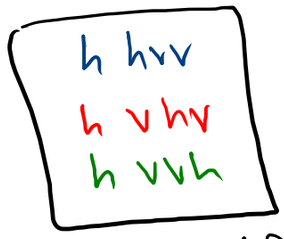
h

1,2 → 3,3

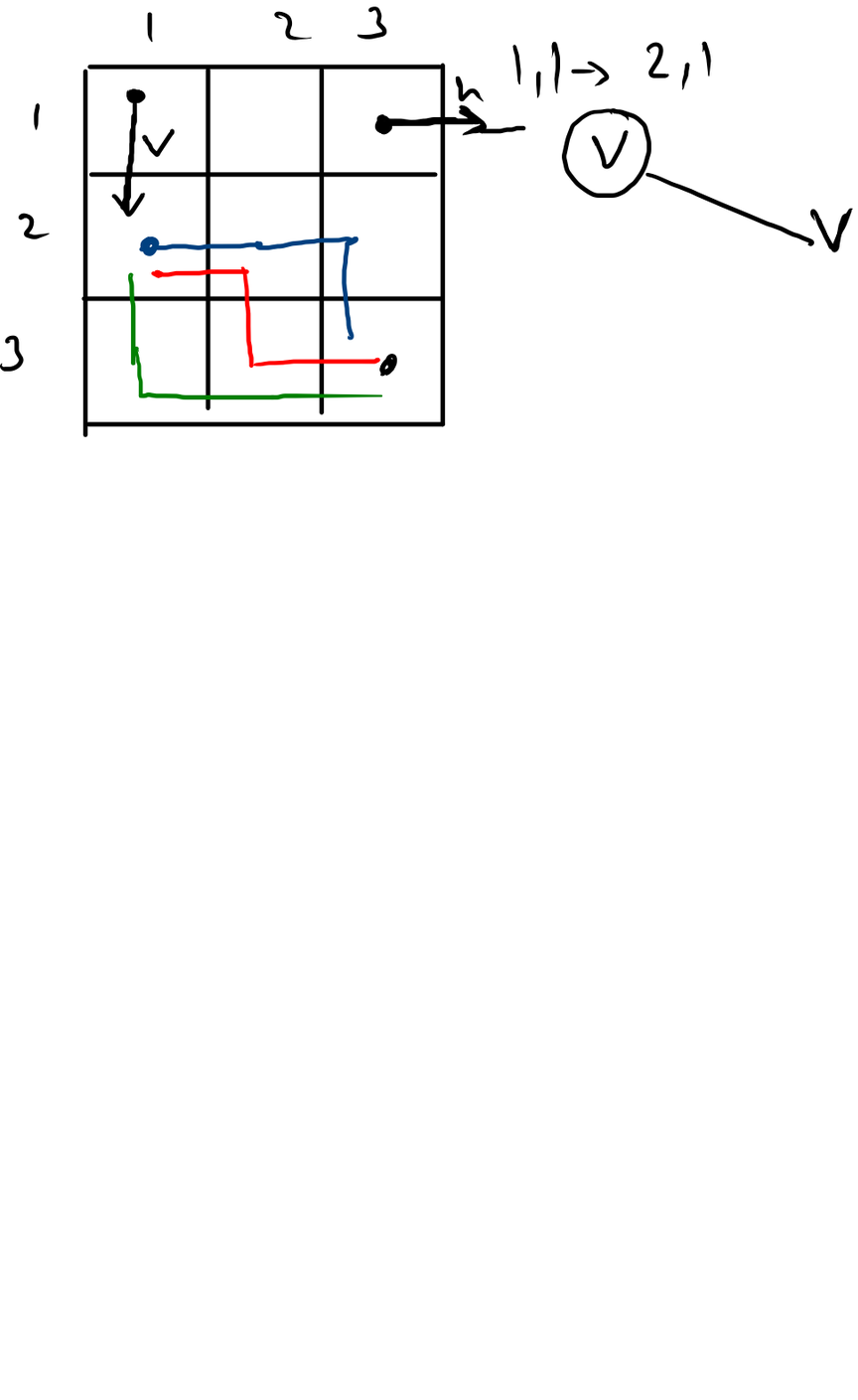


faith

1,1 → 3,3



expectation



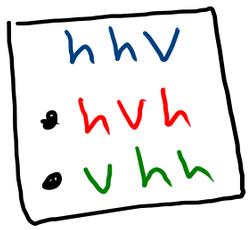
1,1 → 2,1

V

V

faith

2,1 → 3,3

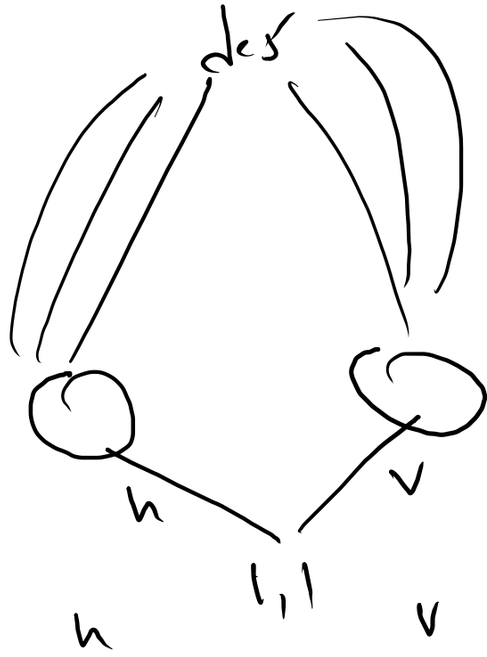


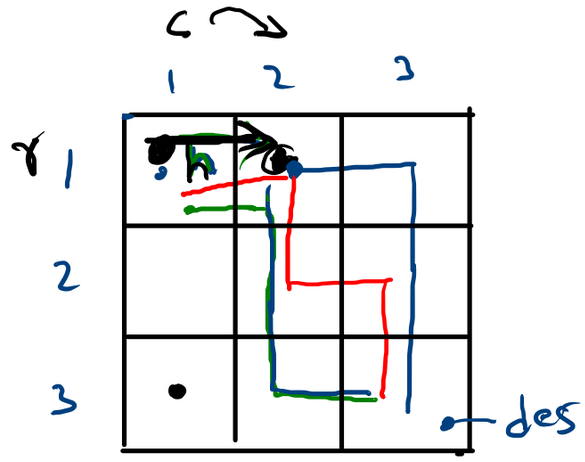
1,1 → 3,3

v hhv

v hvh

v vhh





1,1 → 1,2

h

1,2 → 3,3

h hvv

v hv

v v h

faith

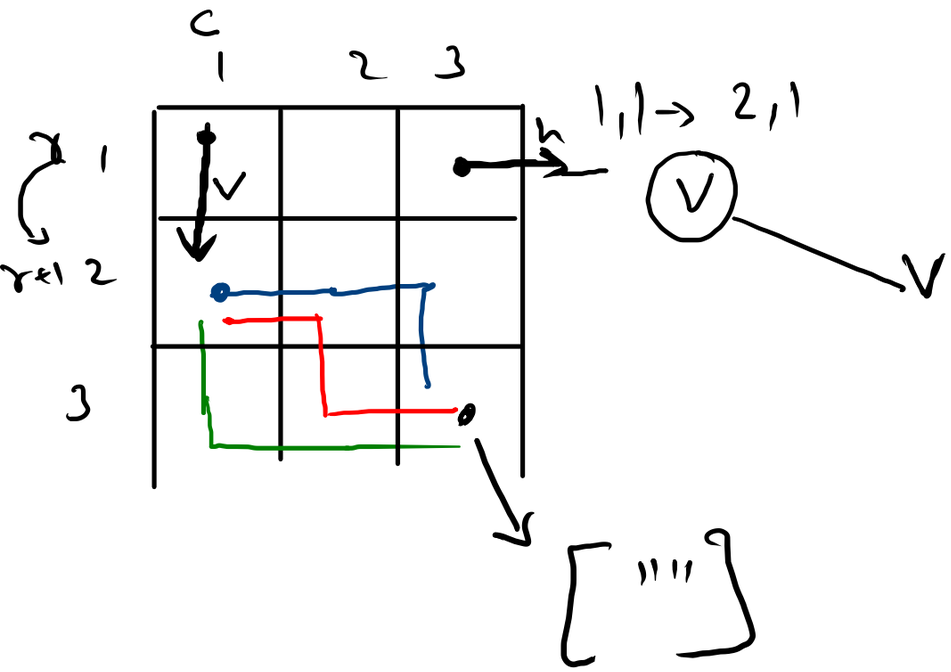
1,1 → 3,3

h hvv

h vhv

h vvh

expectation



faith

2,1 → 3,3

h h v

h v h

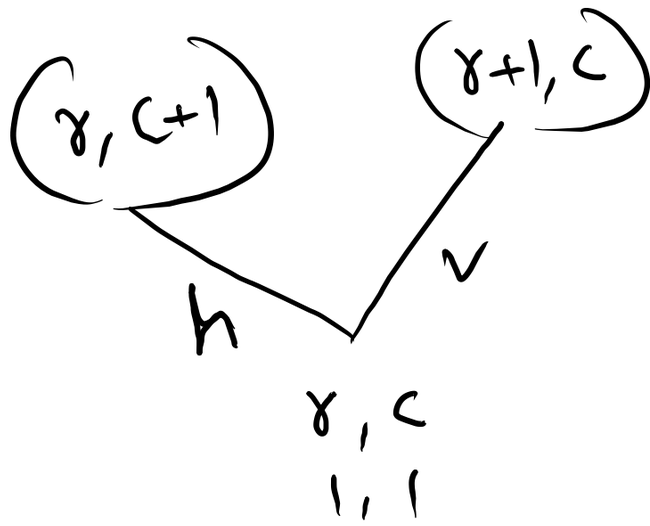
v h h

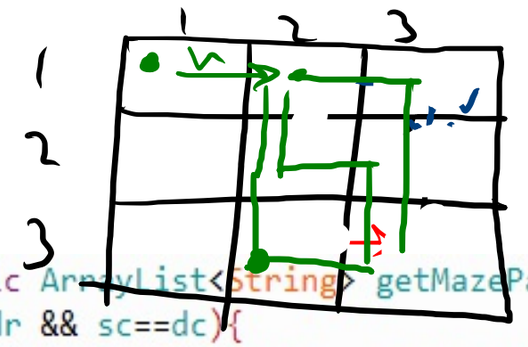
1,1 → 3,3

v h h v

v h v h

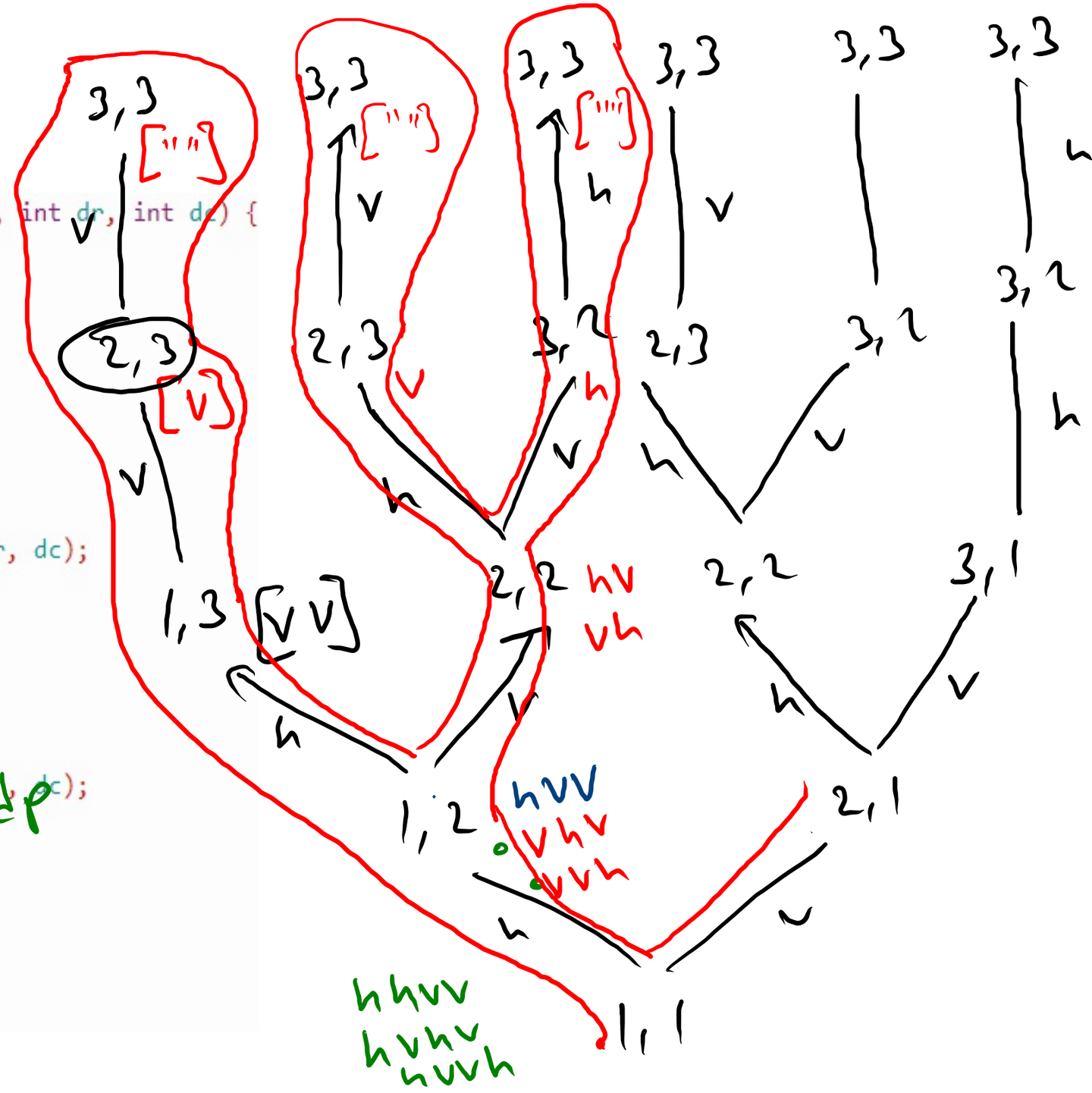
v v h h

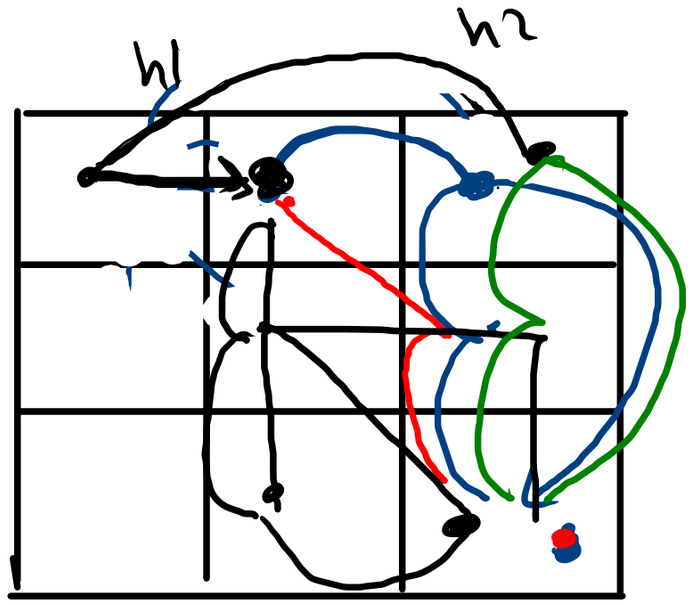





```java
public static ArrayList<String> getMazePaths(int sr, int sc, int dr, int dc) {
    if(sr==dr && sc==dc){
        ArrayList<String> ans = new ArrayList<>();
        ans.add("");
        return ans;
    }

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

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

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

    return ans;
}
```

✓ time } dp
• space }





1,1 → 3,3

h1

h1 v1 v1

h1 v2

d1 v1

v1 h1 v1

v1 d1

v1v1 h1

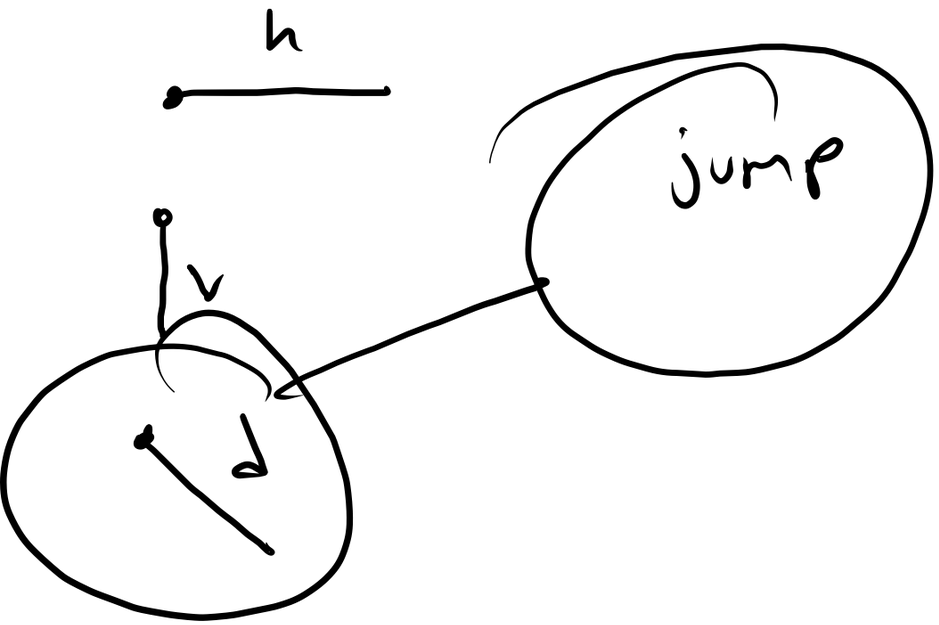
v2 h1

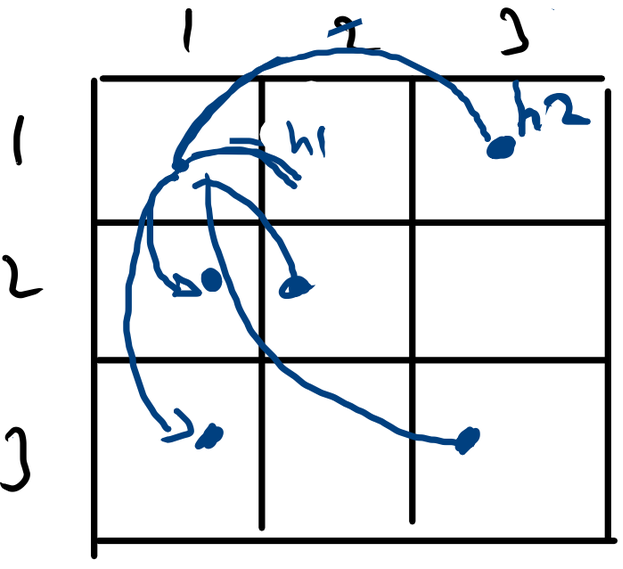
=

h2

v1v1

v2





f(1,1)

jump

1 h1 f(1,2)

2 h2 { f(1,3)

jump

d1 { f(2,2)

d2 { f(3,3)

jump

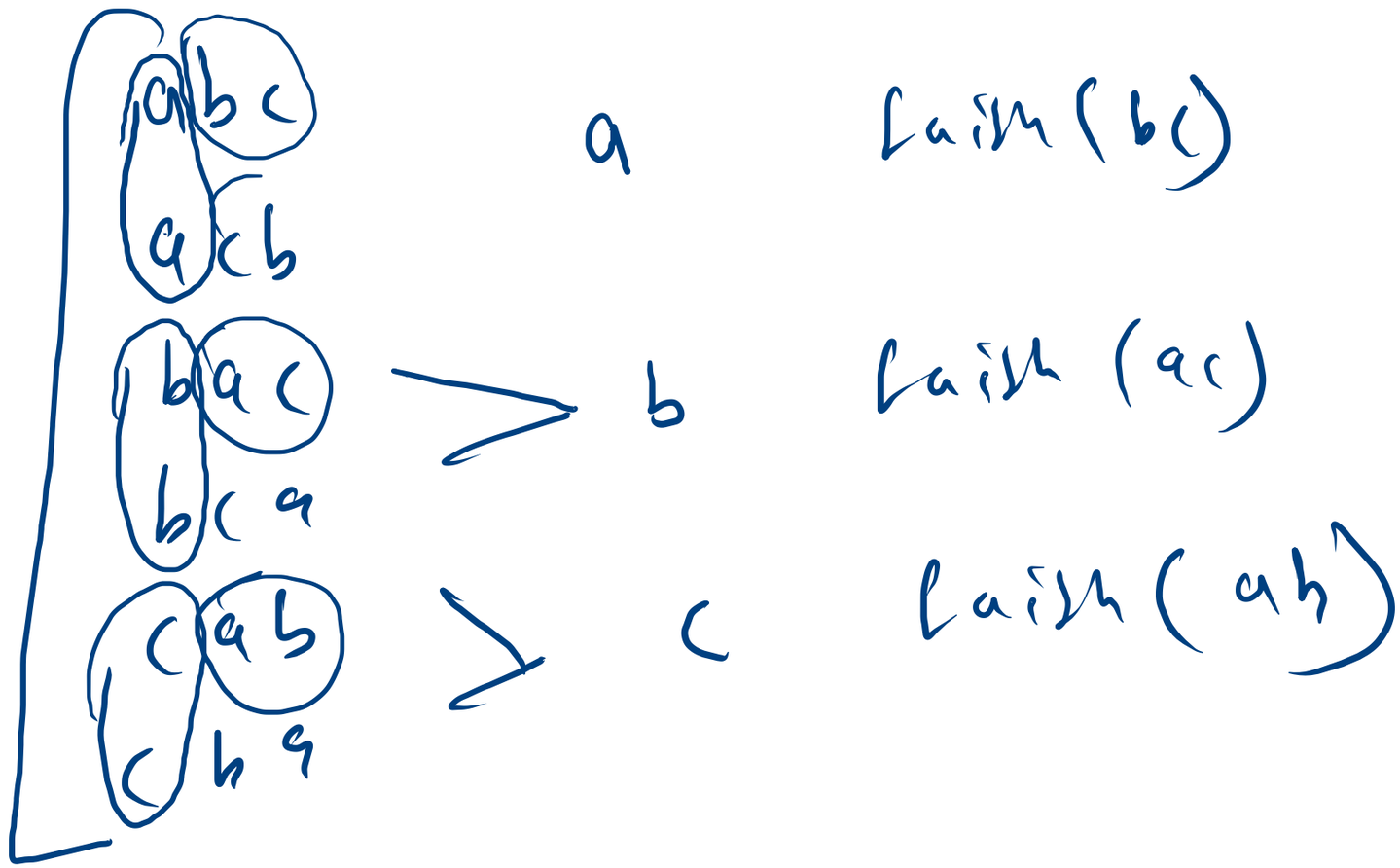
1 v1 f(2,1)

2 v2 f(3,1)

abc
0 1 2

permutations

ArrayList L



2^{18}

2×9

2^{m^m}

3 6 4

2^2 2^2

$2^{\min}$ 2^2 2^{-2} gcd

8 2^2 gcd

14 2^1

why

$2^{\min(\ )}$

2×7

$2 \rightarrow$ $f1^a$ $f2^b$ $f3^c$ $2^{\min}$

$4 \rightarrow$ $f1^d$ $f2^n$ $f3^0$

$\min(b, n)$

$gcd \rightarrow f1^{(a,d)} \times f2$

5 7 8 1 ∞

$\min$ 8 1 ∞