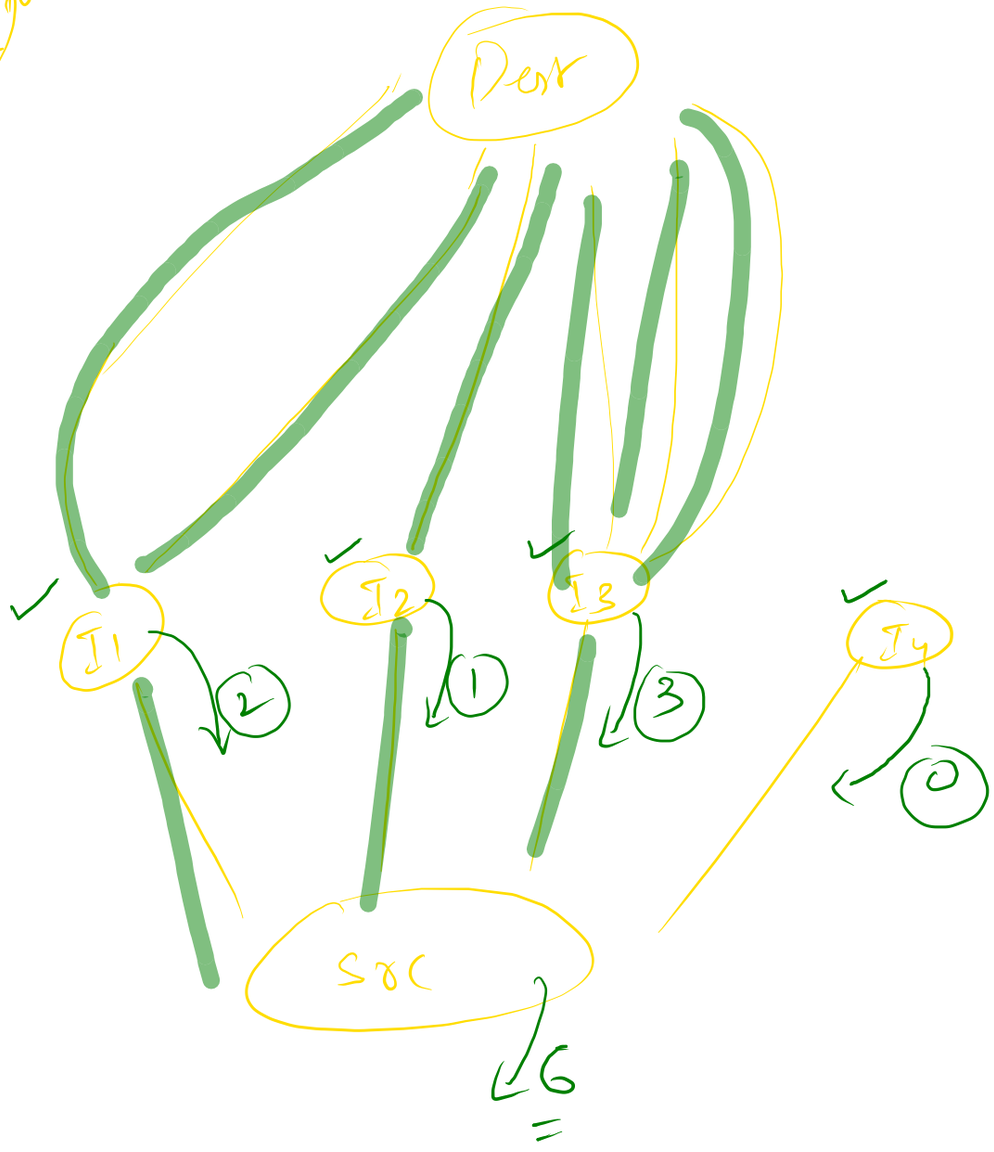
Recursion → Basics (TOH, Ex-Faith)

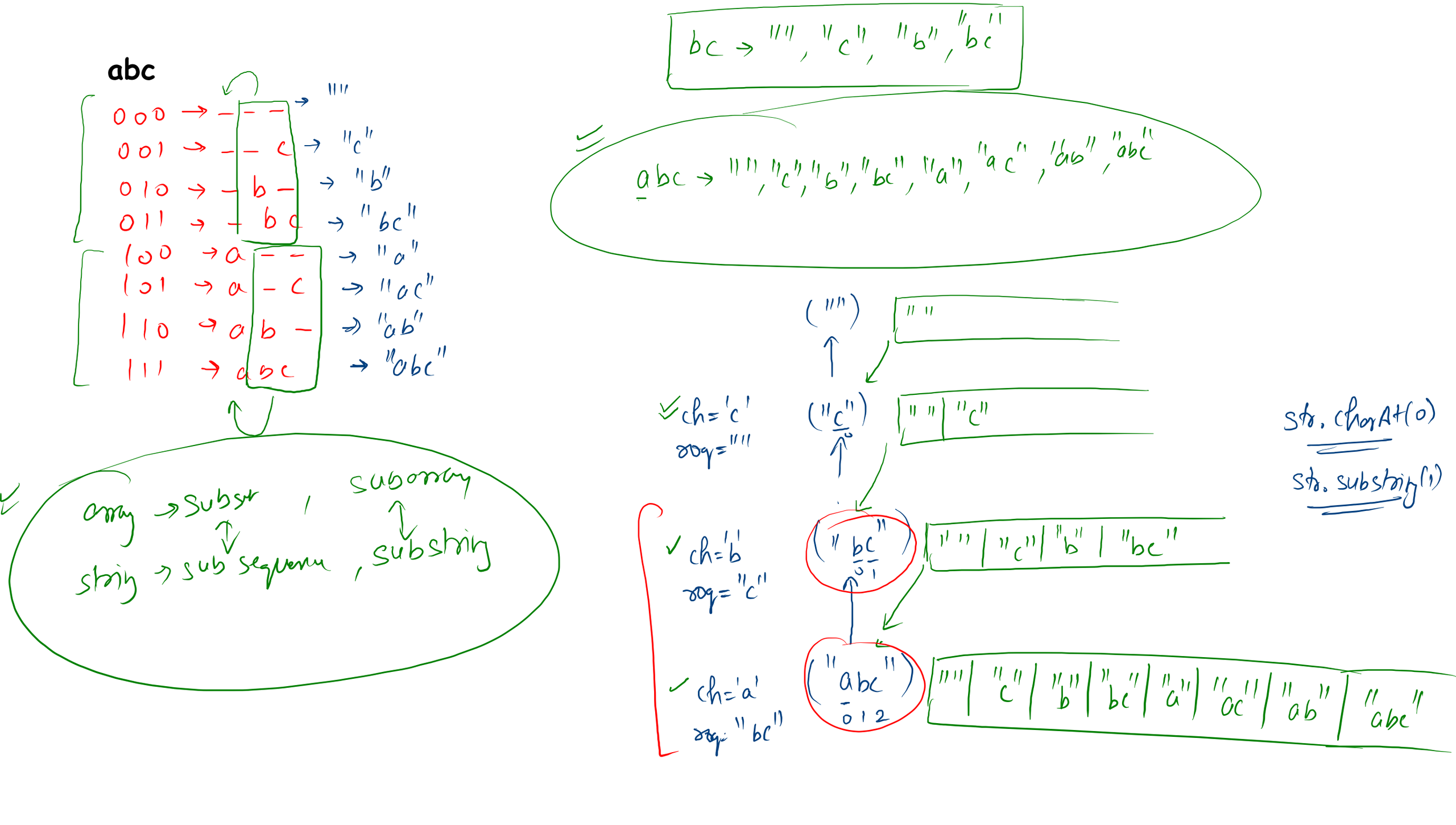
Recursion → Arrays (ido, all indices)

(Recursion → Array 1d) [src → dest basic model]

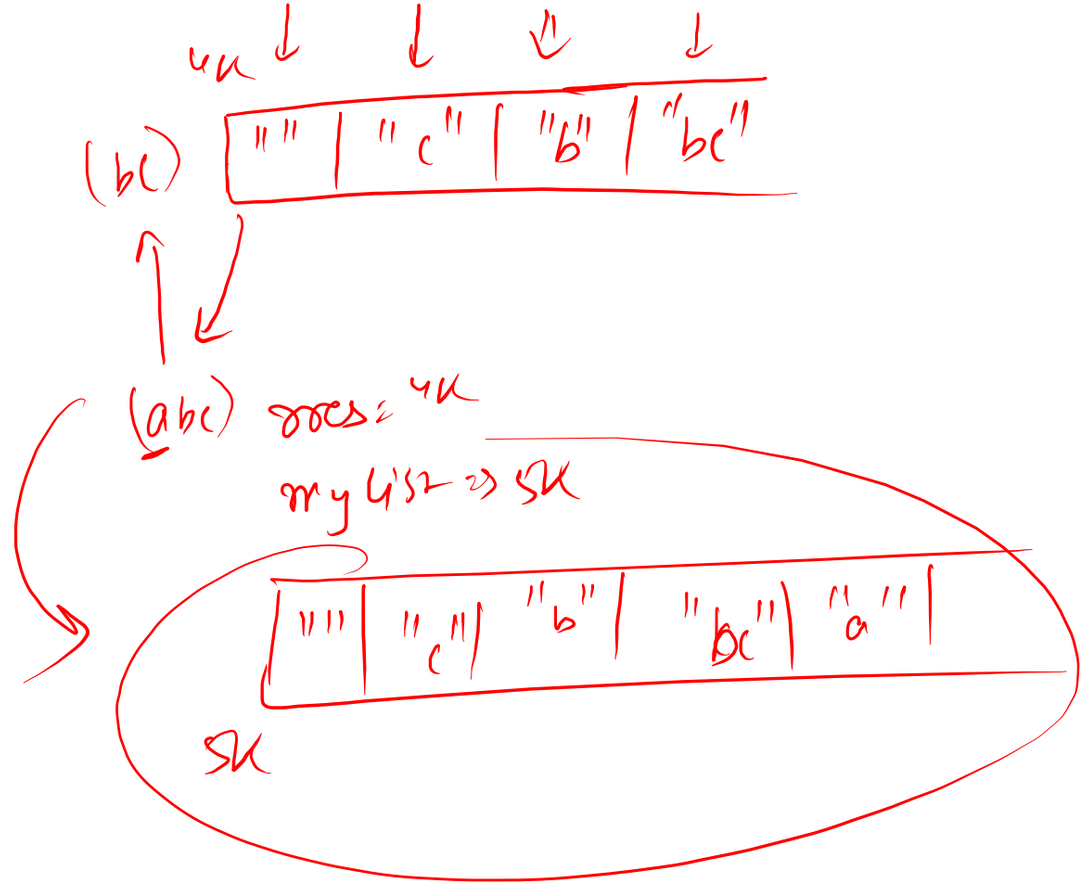
no. of ways



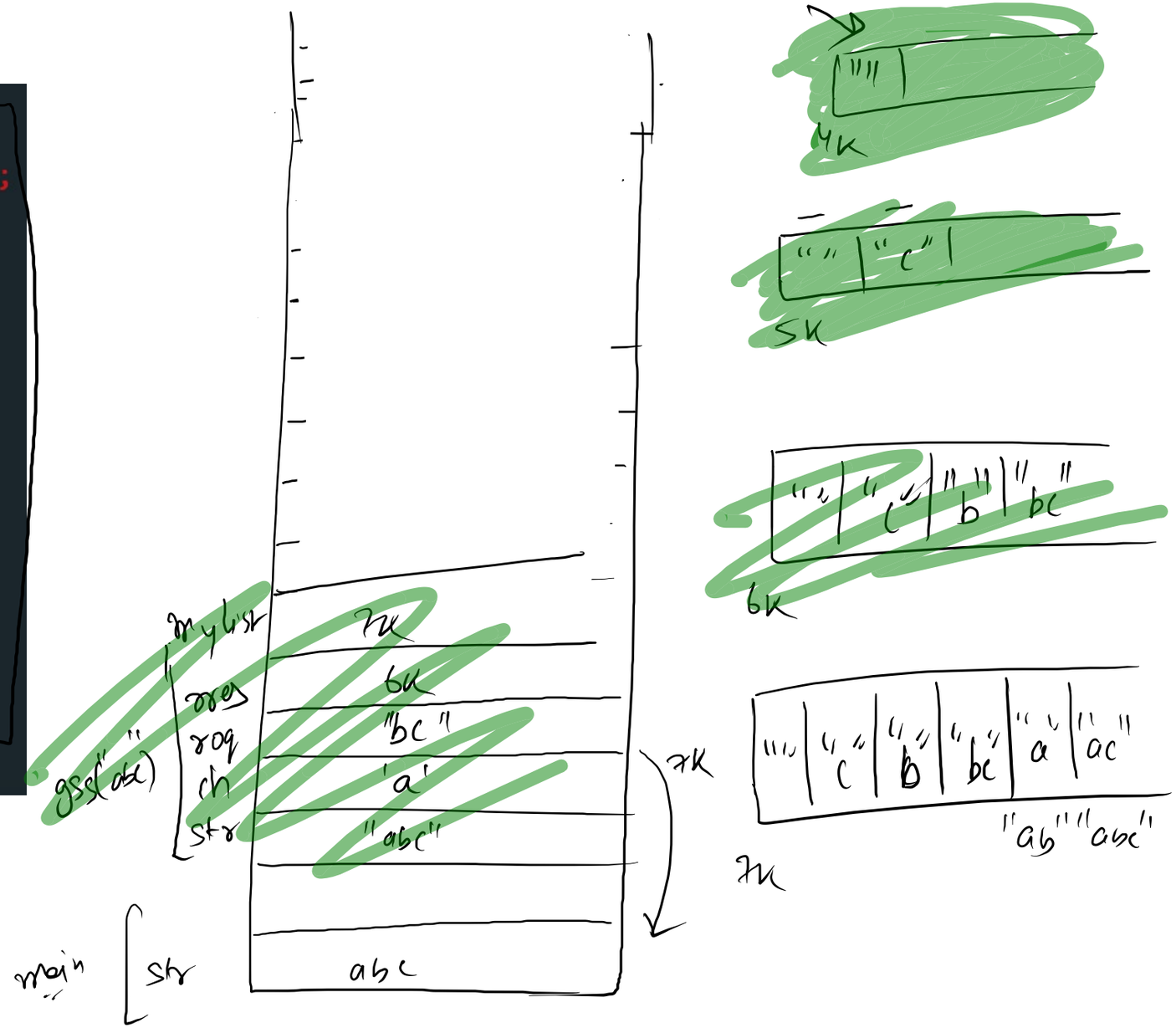
min len path


abc
000 → - - - → ""
001 → - - c → "c"
010 → - b - → "b"
011 → - b c → "bc"
100 → a - - → "a"
101 → a - c → "ac"
110 → a b - → "ab"
111 → a b c → "abc"
array → subsr
string → sub sequence
subarray
substring
bc → "", "c", "b", "bc"
abc → "", "c", "b", "bc", "a", "ac", "ab", "abc"
("")
""
ch='c'
idx=""
("c")
"" | "c"
ch='b'
idx="c"
("bc")
"" | "c" | "b" | "bc"
ch='a'
idx="bc"
("abc")
"" | "c" | "b" | "bc" | "a" | "ac" | "ab" | "abc"
str.charAt(0)
str.substring(1)

```
for(String s : rres){
    myList.add(s);
}
// inc
for(String s : rres){
    myList.add(ch+s);
}
return myList;
```



```java
public static ArrayList<String> gss(String str) {
    if(str.length() == 0){
        ArrayList<String> bList = new ArrayList<>();
        bList.add("");
        return bList;
    }
    char ch = str.charAt(0);
    String roq = str.substring(1);
    ArrayList<String> rres = gss(roq);
    ArrayList<String> myList = new ArrayList<>();
    // exc
    for(String s : rres){
        myList.add(s);
    }
    // inc
    for(String s : rres){
        myList.add(ch+s);
    }
    return myList;
}
```



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

Print all possible words

String → "78"

t v
u w
x

→ tv tw tx uv uw ux

1 79 →

a → t y
b u z
c

→ aty atz auy auz

→ bty btz buy buz

→ cty ctz cuy cuz

"79" → ty | tz | uy | uz

word = "abc";

key = 1

(ch = '1') == (abc) ... char

"1" "79" "179"

aty | atz | auy | auz | bty | btz | buy | buz | cty | ctz | cuy | cuz

```java
static String keypad[] = {".;","abc","def","ghi","jkl","mno","pqrs","tu","vwx","yz"};
public static ArrayList<String> getKPC(String str) {
    char ch = str.charAt(0);
    String roq = str.substring(1);
    ArrayList<String> rres = getPKC(roq);


    ArrayList<String> myList = new ArrayList<>();
    int key = Integer.parseInt(ch+"");
    String word = keypad[key];
    for(int i = 0 ; i < word.length() ; i++){
        for(String s : rres){
            myList.add(word.charAt(i)+s);
        }
    }

    return myList;
}
```

0 1 2 3 4 5 6 7 8 9

(ch - '0')

single digits

(int) ch

encoded value

'0'

48

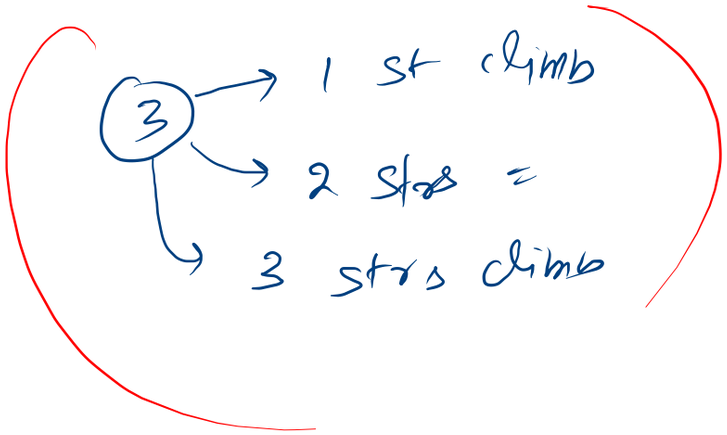
"123" → 123

String →

""

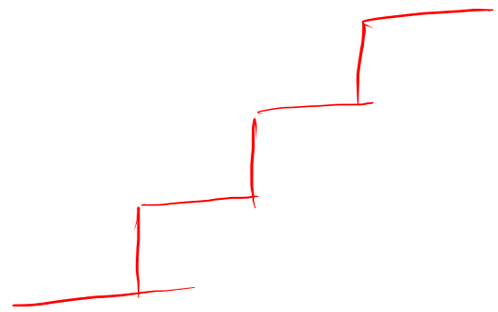
ros="yz"
key=9
ch='9'
roq="" ("9") "y" | "z"

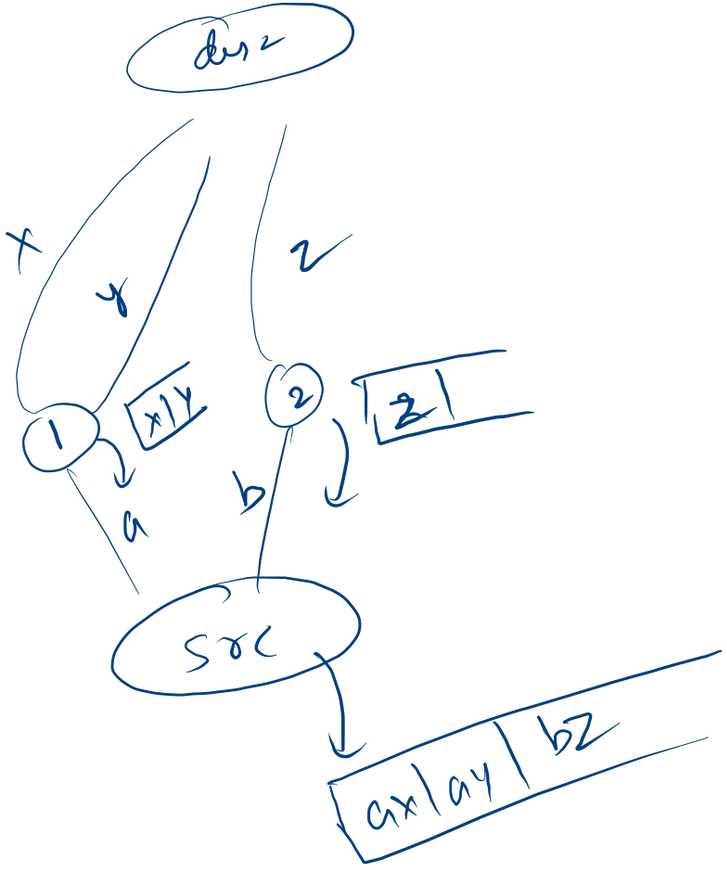
ch='7' ("79") "ty" | "tz" | "uy" | "uz"
roq="9"
key=7
ros="tu"
0 1

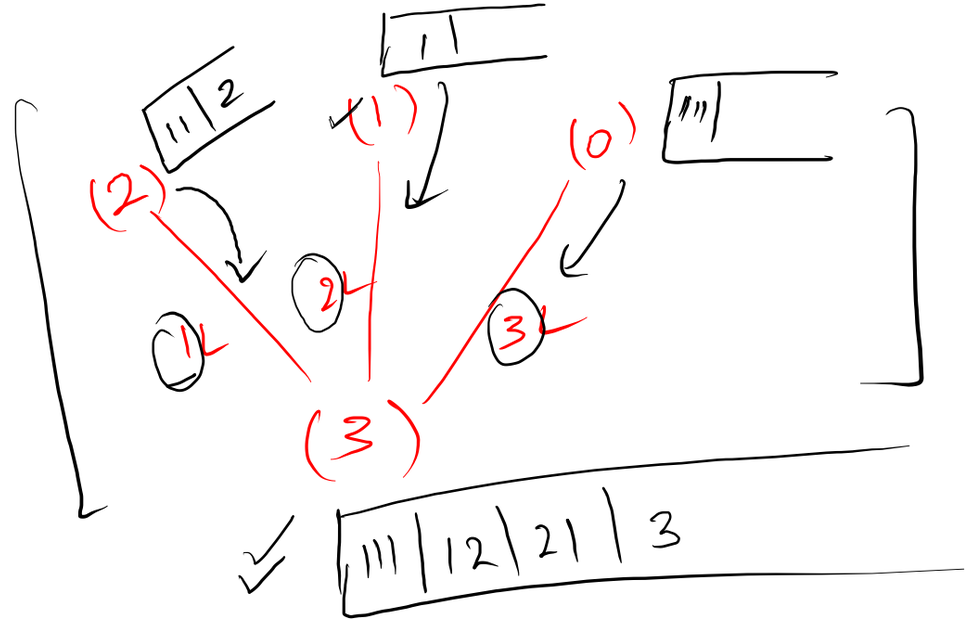


n steps

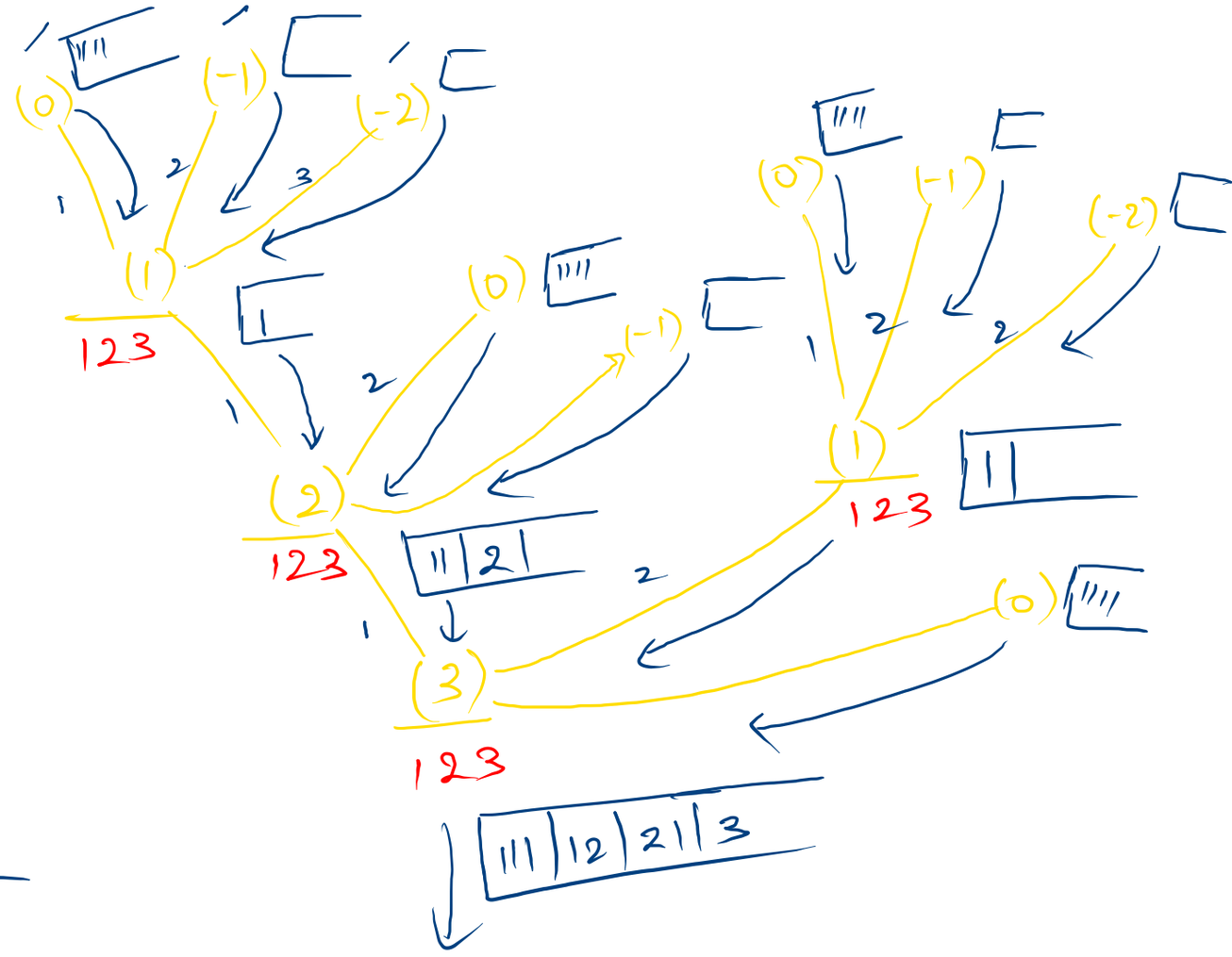
111, 12, 21, 3







```java
public static ArrayList<String> getStairPaths(int n) {

    ArrayList<String> oneLenPath = getStairPaths(n-1);
    ArrayList<String> twoLenPath = getStairPaths(n-2);
    ArrayList<String> threeLenPath = getStairPaths(n-3);

    ArrayList<String> myList = new ArrayList<>();
    for(String path : oneLenPath){
        myList.add(1+path);
    }
    for(String path : twoLenPath){
        myList.add(2+path);
    }
    for(String path : threeLenPath){
        myList.add(3+path);
    }

    return myList;
}
```



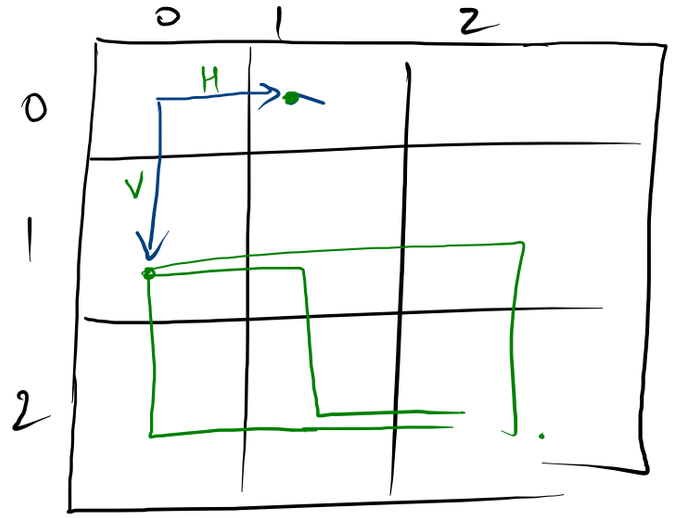
valid path/ patterns → 1111

invalid path/ →

3 → nr
3 → nc

Sample Output

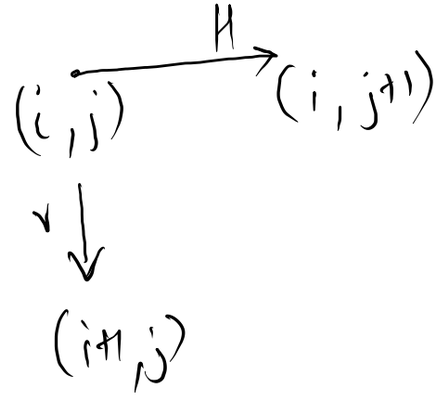
[hhvv, hvhv, hvvh, vhhv, vhvh, vvhh]



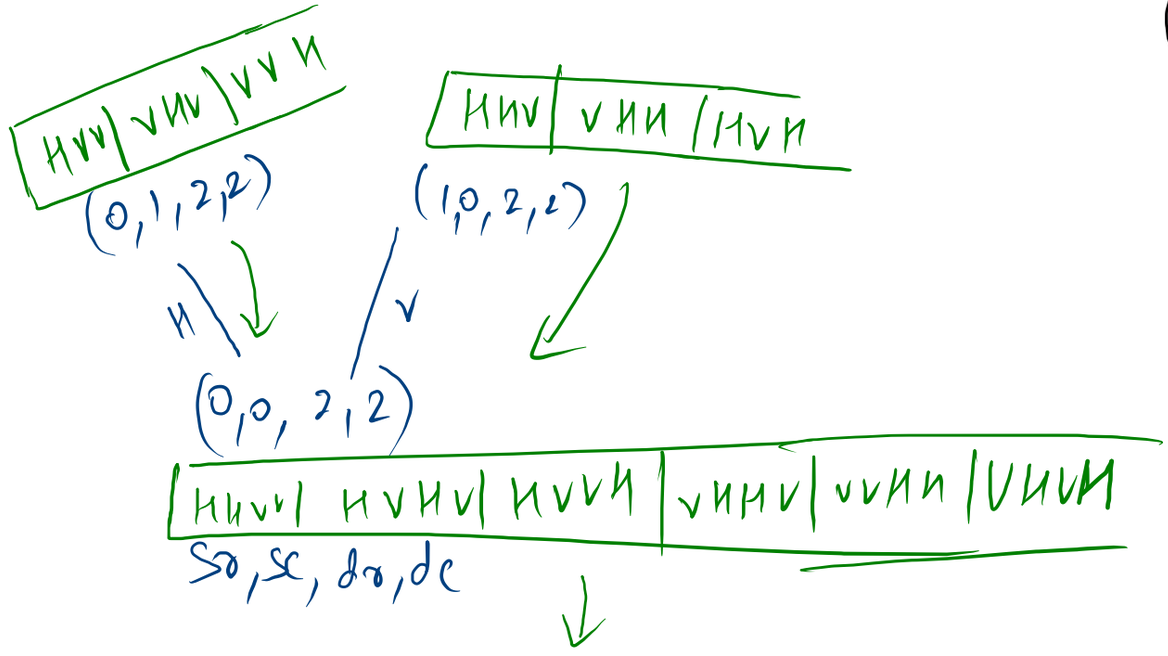
Starting point → (0,0)

dest → $(\frac{nr-1}{2}, \frac{nc-1}{2})$

movement allowed → h, v



Point
src → dest
Path



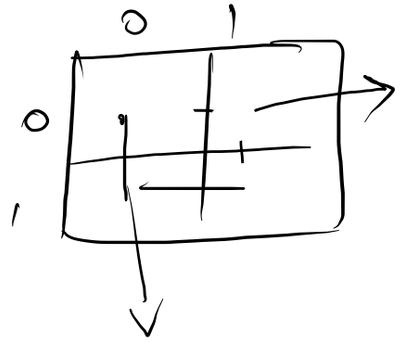
```java
// sr - source row
// sc - source column
// dr - destination row
// dc - destination column
public static ArrayList<String> getMazePaths(int sr, int sc, int dr, int dc) {
    ArrayList<String> HPath = getMazePaths(sr,sc+1,dr,dc);
    ArrayList<String> VPath = getMazePaths(sr+1,sc,dr,dc);

    ArrayList<String> myPath = new ArrayList<>();
    for(String path : HPath){
        myPath.add('h'+path);
    }
    for(String path : VPath){
        myPath.add('v'+path);
    }
    return myPath;
}
```

VC

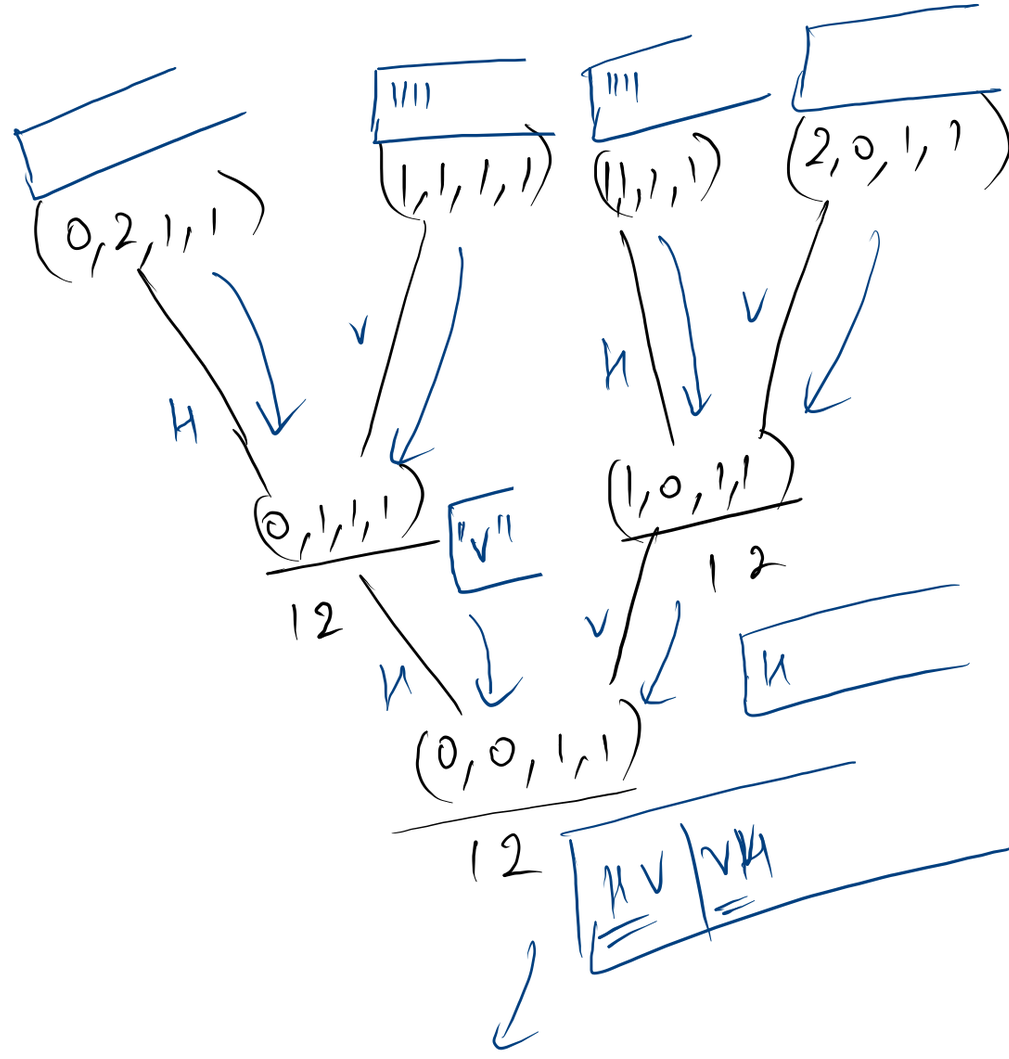
sr == dr && sc == dc

Base

sr > dr || sc > dc

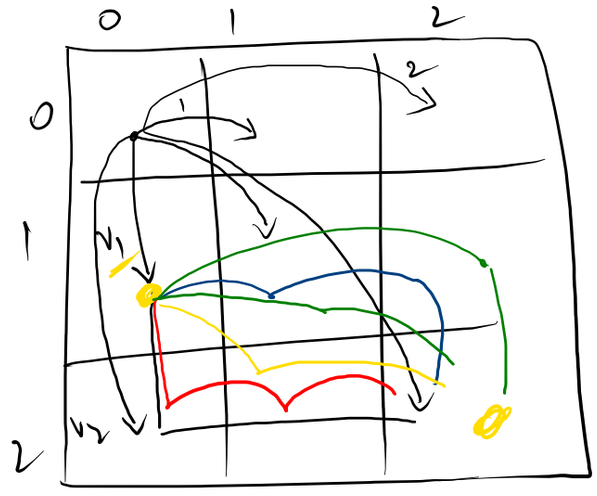


nr = 2, nc = 2



nr = 3

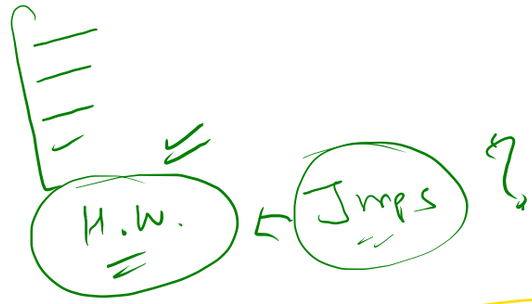
nc = 3

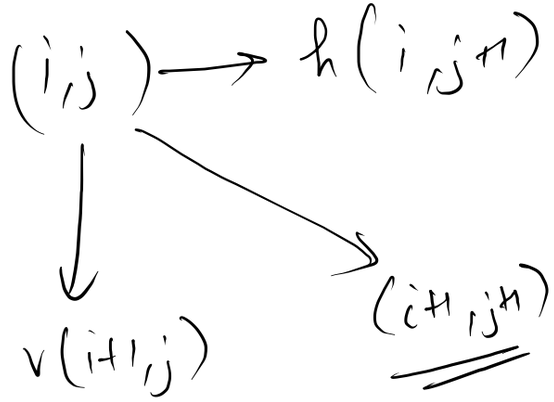


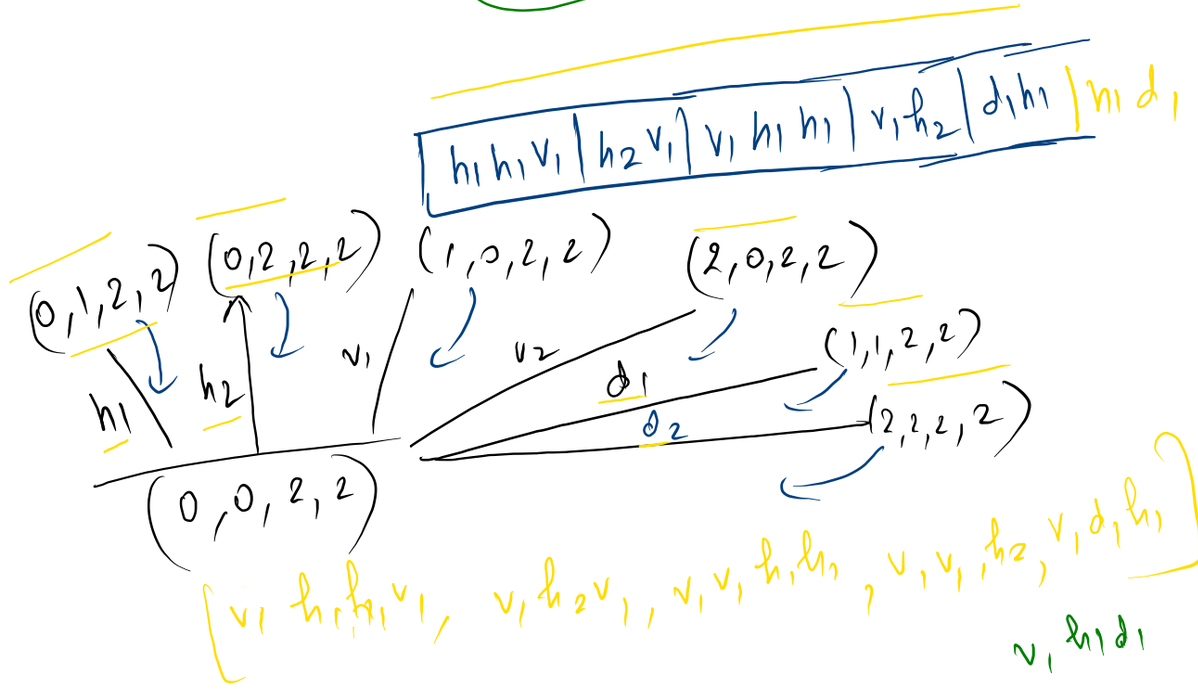
Src → 0,0

dest → 2,2

Jumps allowed





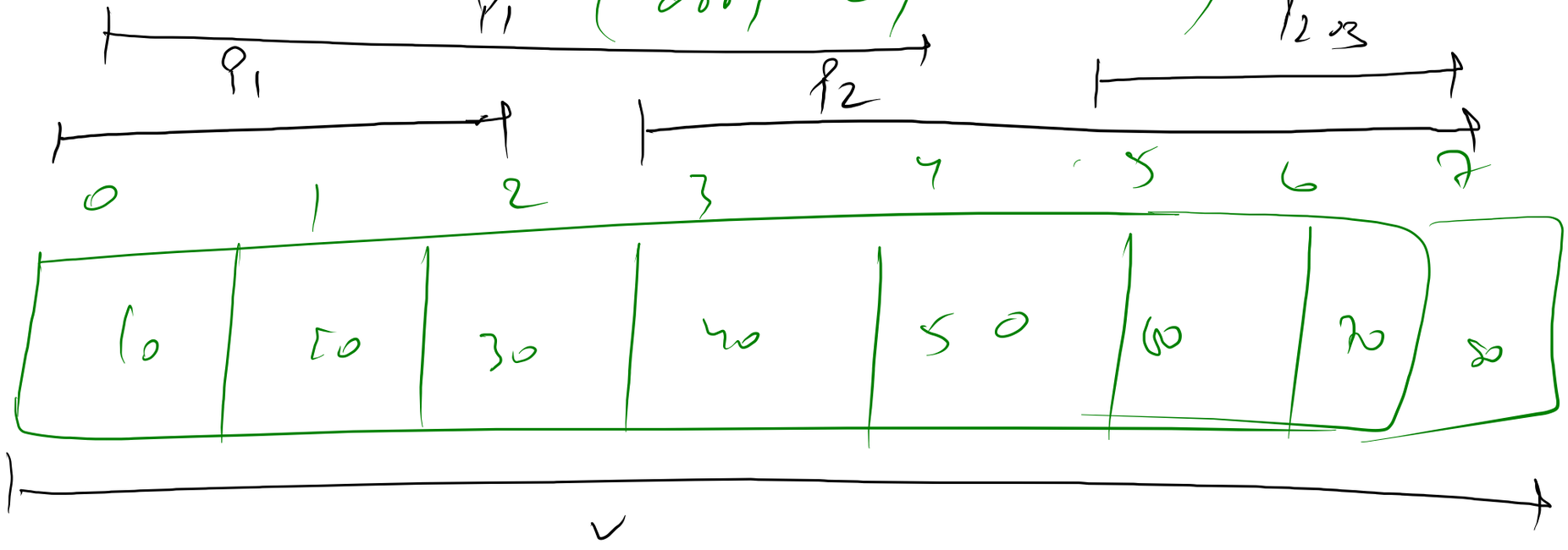


d = 2

1. reverse (arr, 0, d)

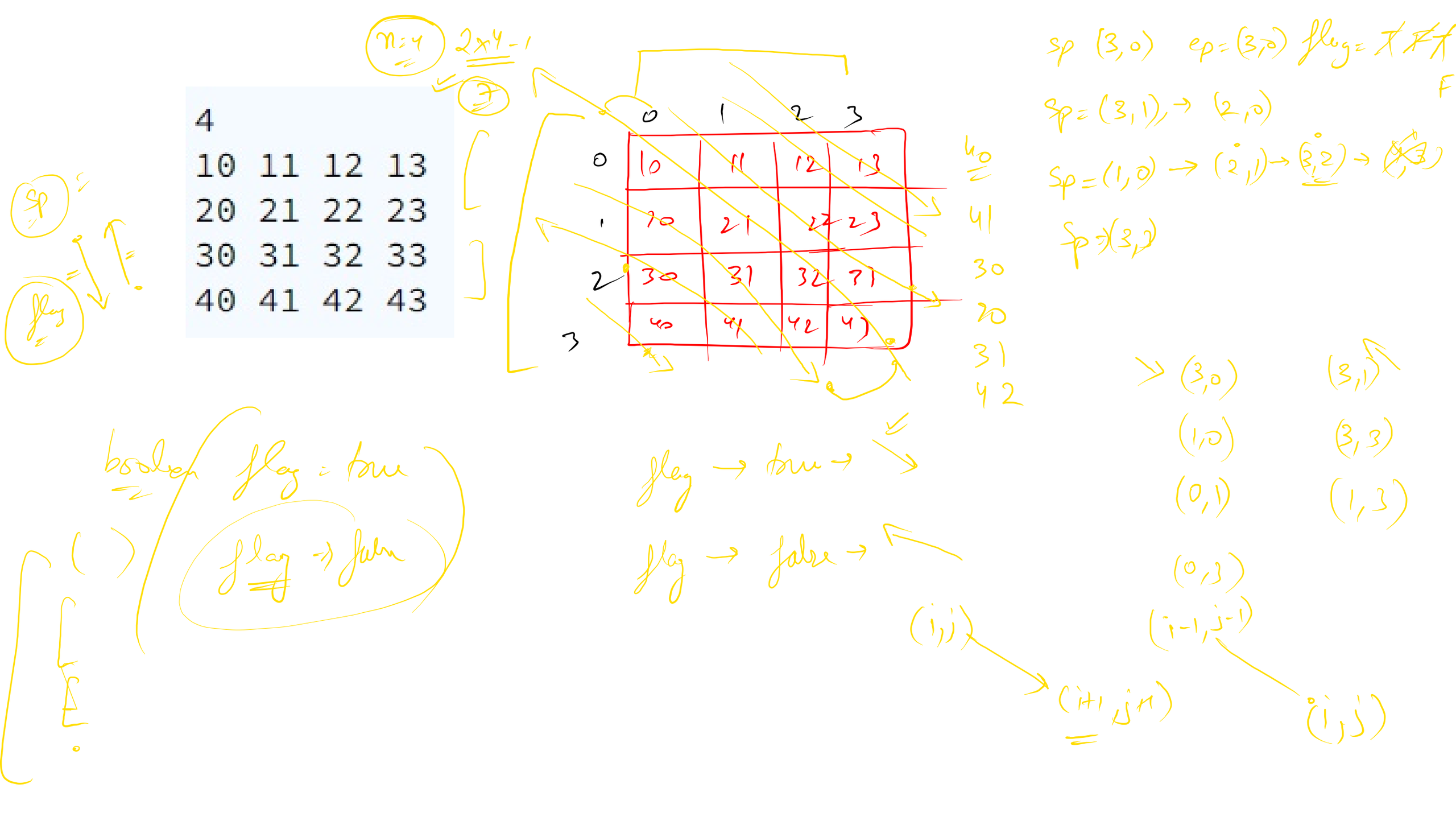
2. reverse (arr, d+1, len-1)

3 reverse (arr, 0, len-1)

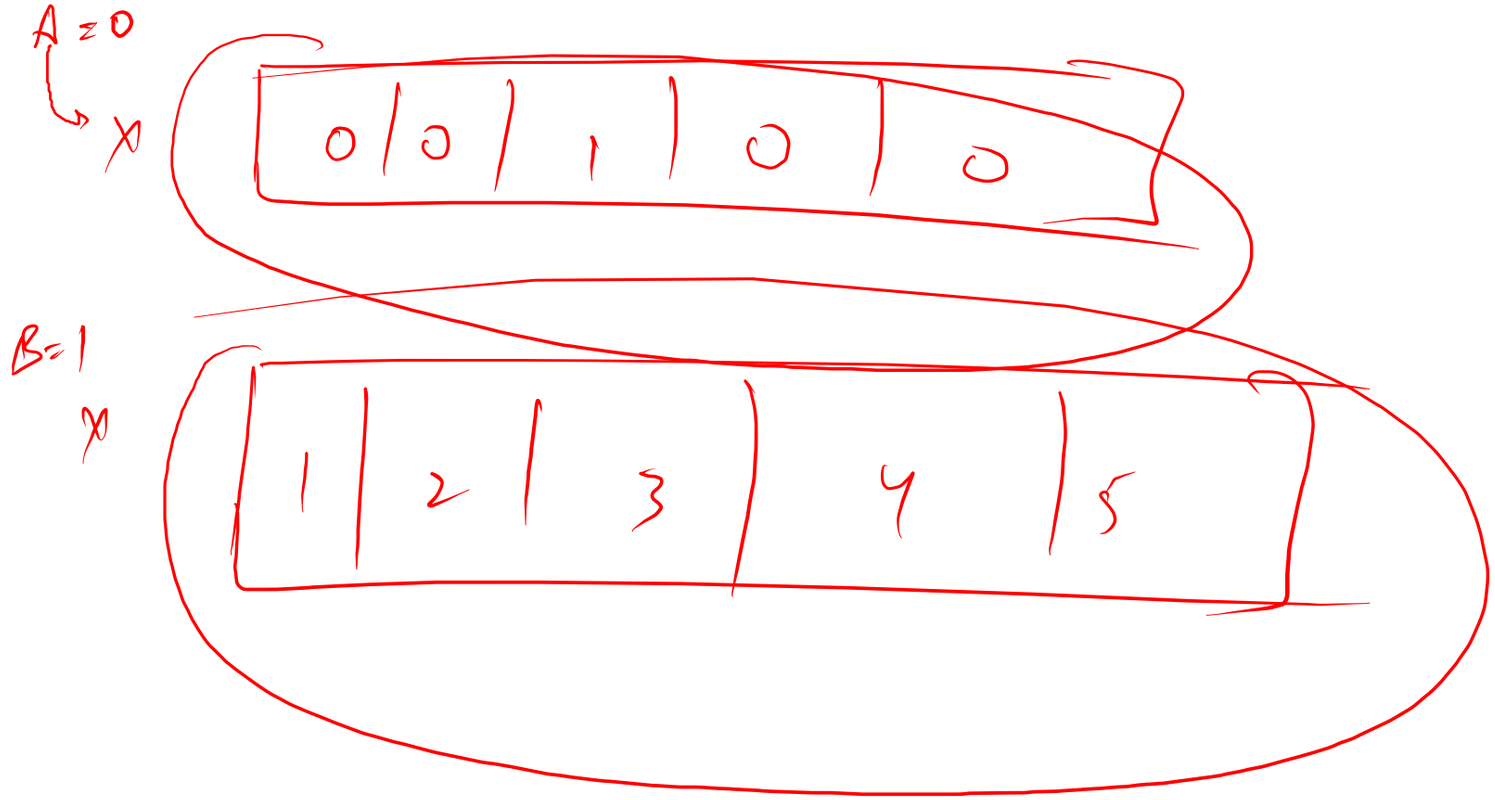


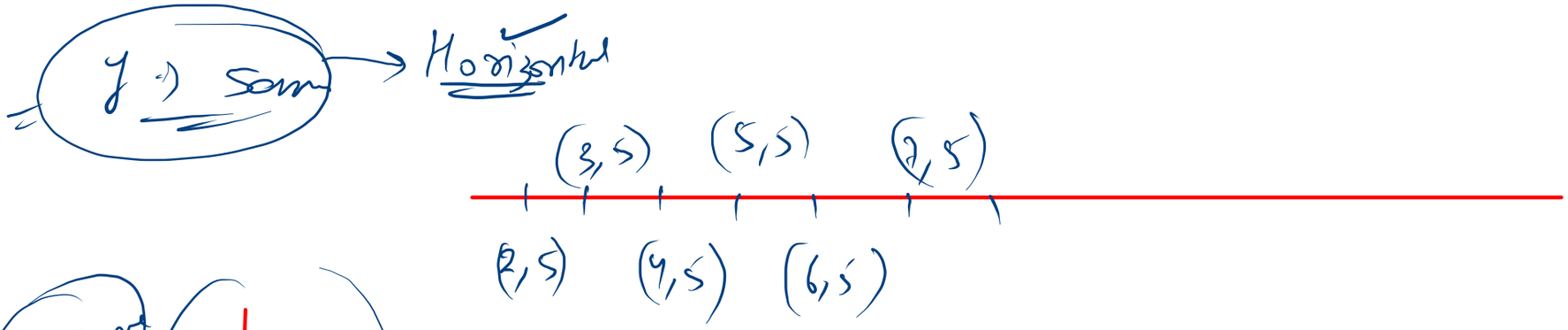
k = 2

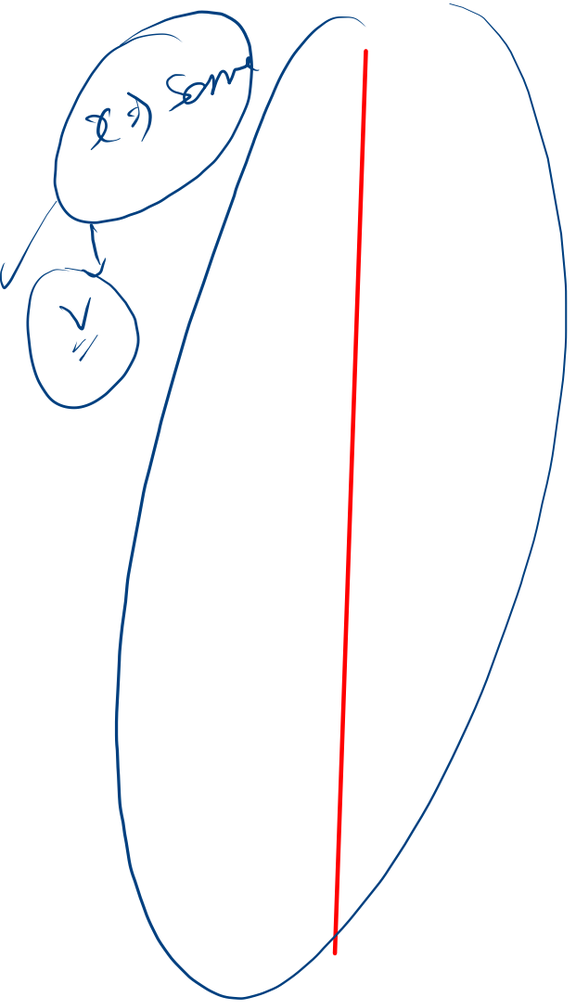
```
4
10 11 12 13
20 21 22 23
30 31 32 33
40 41 42 43
```



```
5
0 1
0 2
1 3
0 4
0 5
```




y = 5
Horizontal
(3,5)
(5,5)
(7,5)
(2,5)
(4,5)
(6,5)


x = 5
V