Recursion

arr = | 15 | 11 | 40 | 4 | 4 | 9 |

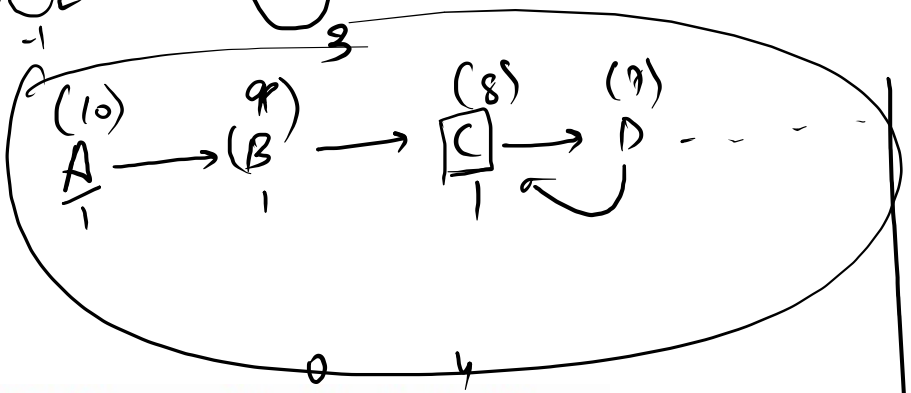
0 1 2 3 4 5

x = 4

Provide pos of first occurrence of '4' in array

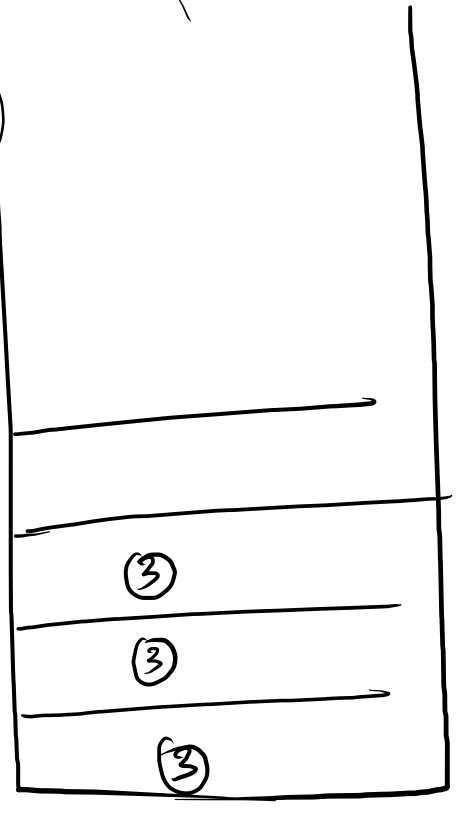
6
15
11
40
4
4
9
4

arr.length = 1

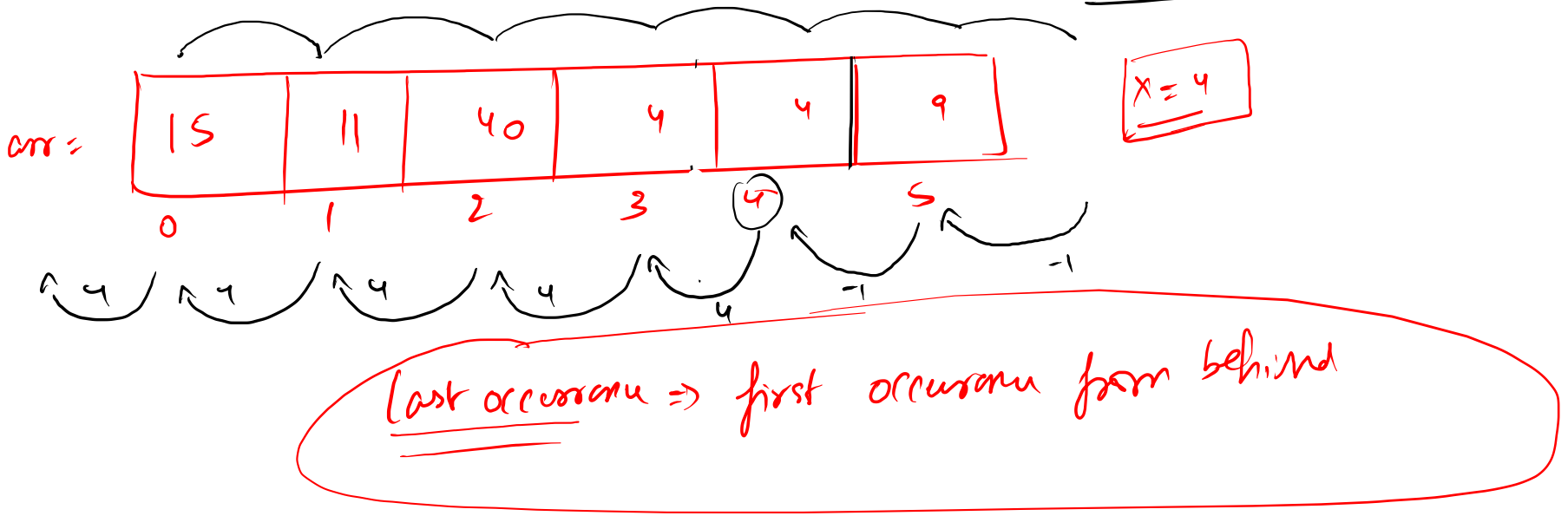



```java
public static int firstIndex(int[] arr, int idx, int x){
    if(idx == arr.length){
        return -1;      -(1)
    }

    if(arr[idx] == x){
        return idx;     -(2)
    }else{
        return firstIndex(arr,idx+1,x); -(3)
    }
}
```

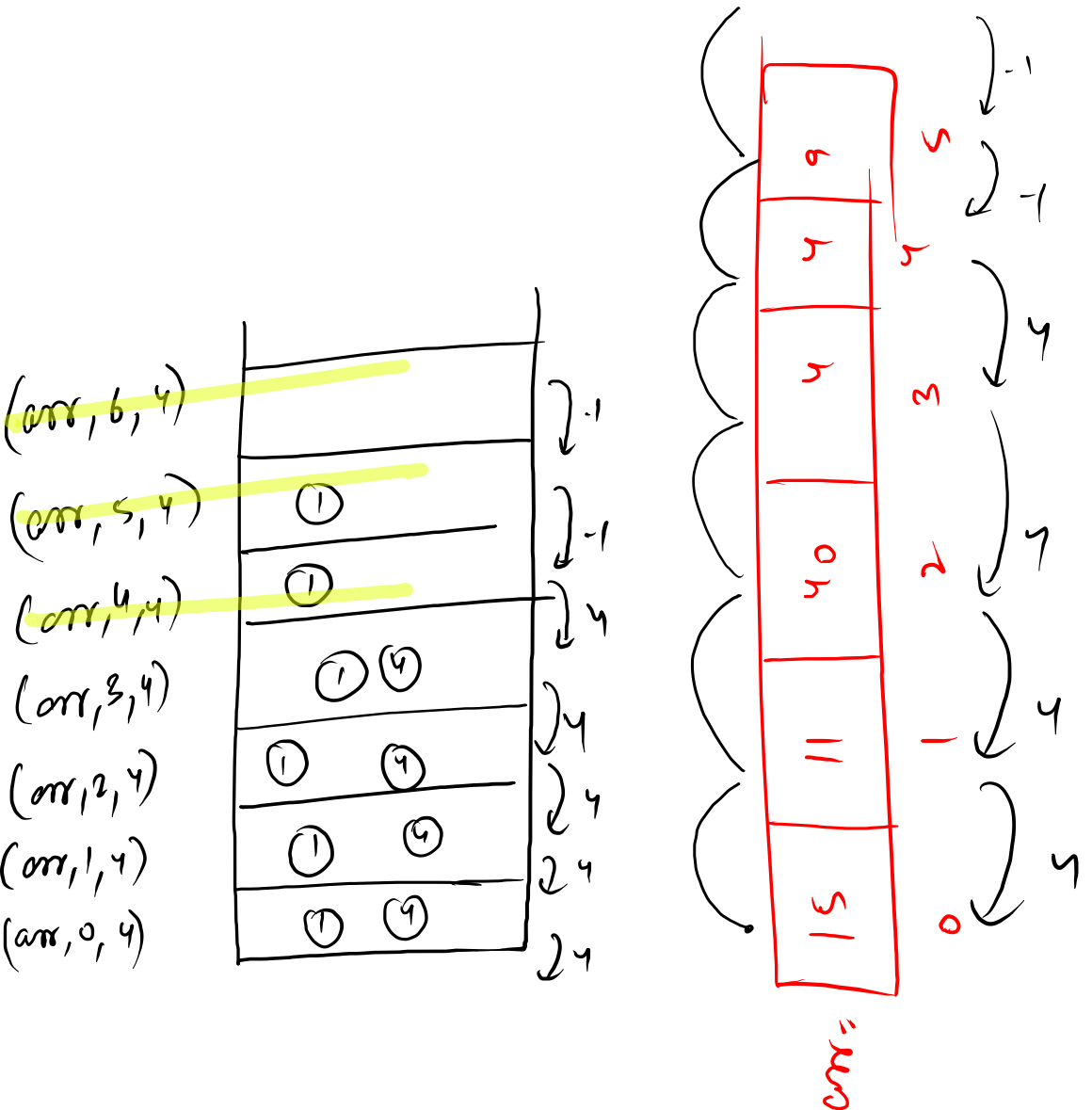
(arr, 3, 4)
(arr, 2, 4)
(arr, 1, 4)
(arr, 0, 4)
idx x



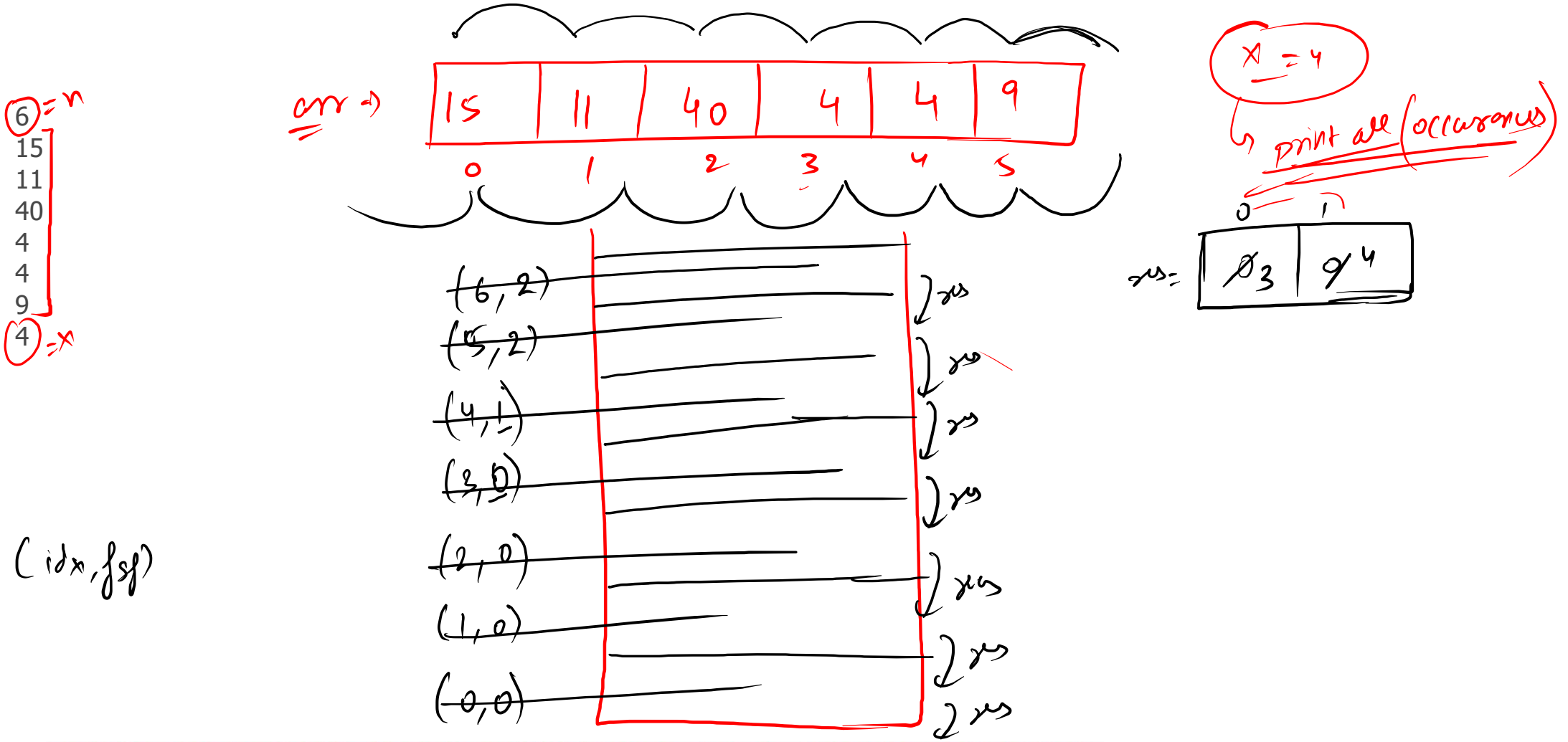
-1 : element not found



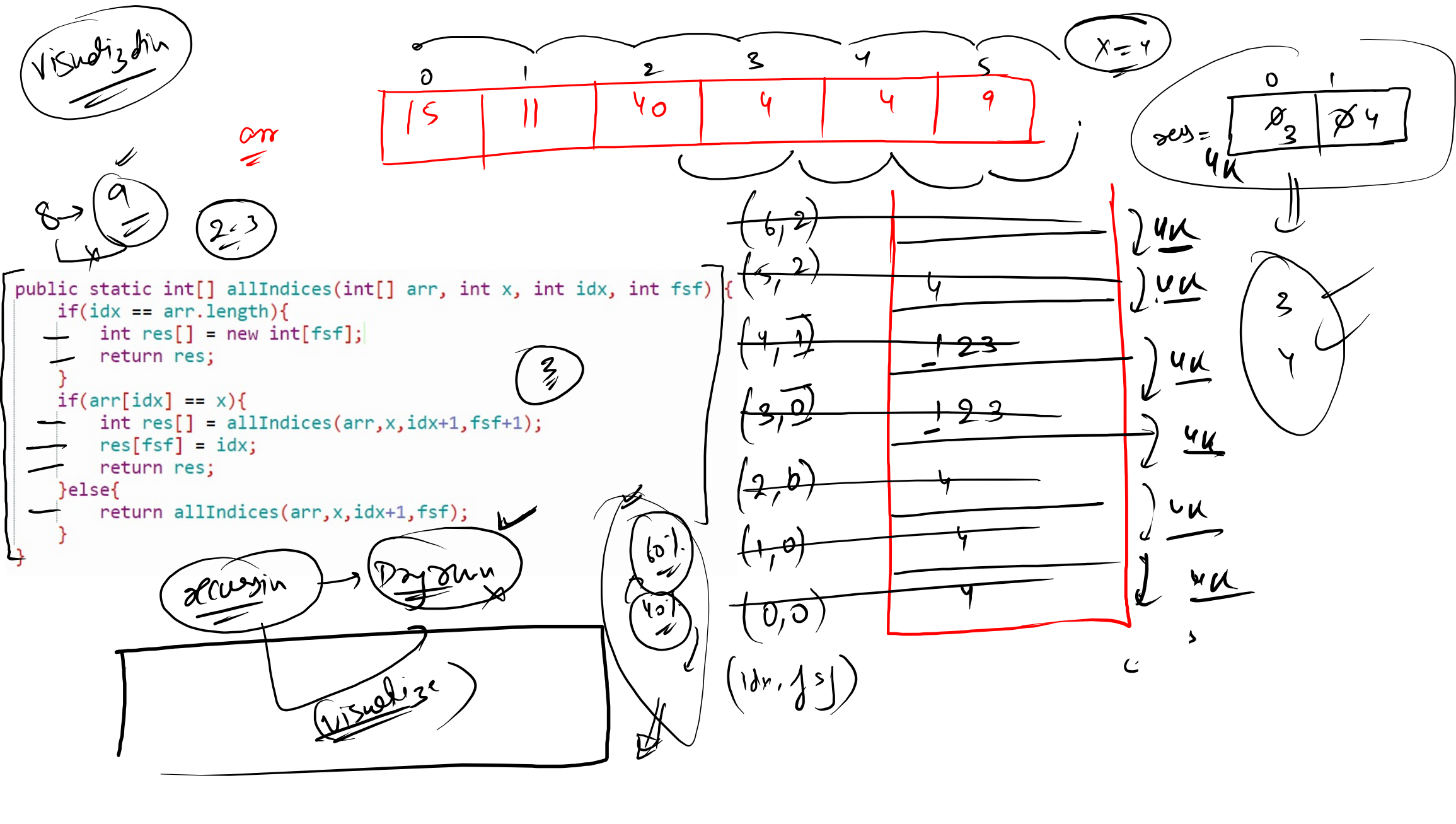
last occurrence => first occurrence from behind

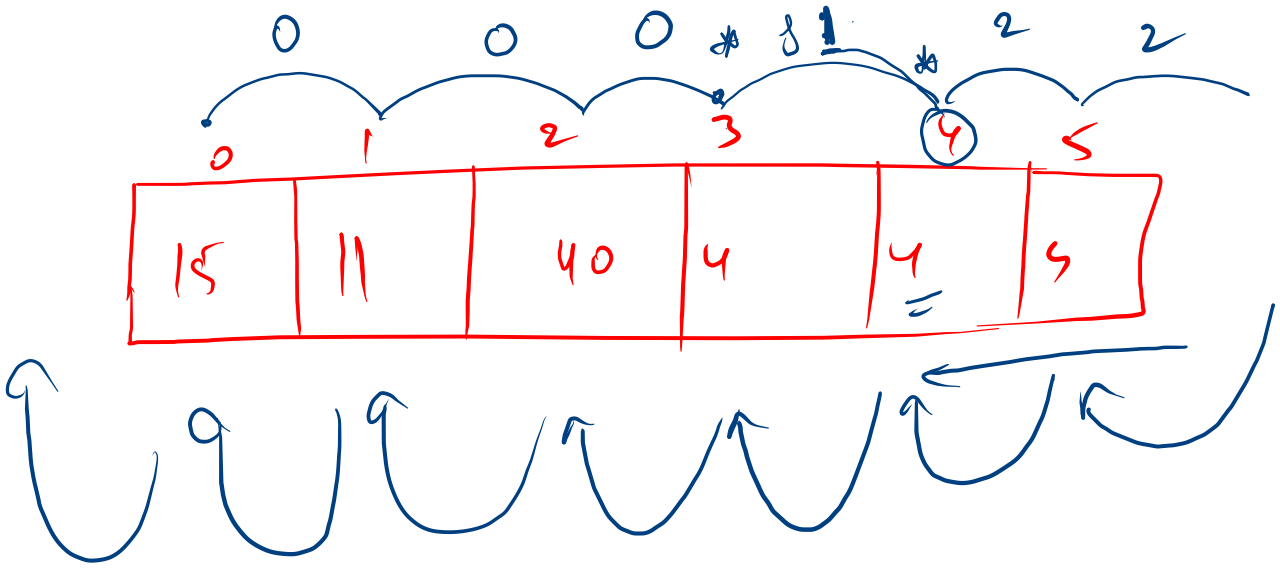



```java
public static int lastIndex(int[] arr, int idx, int x){
    if(idx == arr.length){
        return -1;
    }
    int res = lastIndex(arr,idx+1,x);  - ①

    if(res == -1){
        if(arr[idx] == x){
            return idx;     - ②
        }else{
            return -1;      - ③
        }
    }
    return res;             - ④
}
```



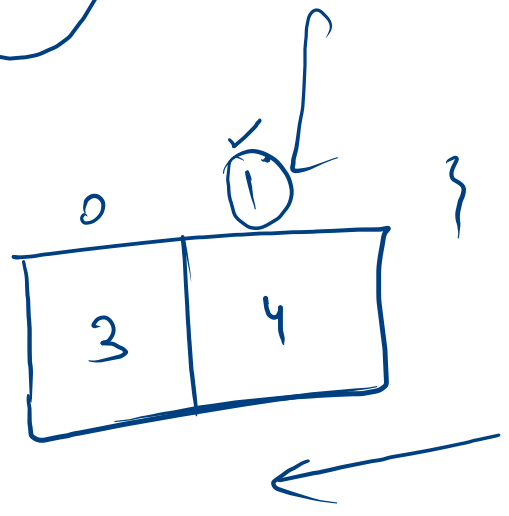

```
public static int[] allIndices(int[] arr, int x, int idx, int fsf) {
    // write ur code here
}
```


Visualization
0 1 2 3 4 5
15 11 40 4 4 9
x = 4
res =
0 1
Ø3 Ø4
4K
arr
9
2 3
public static int[] allIndices(int[] arr, int x, int idx, int fsf) {
if(idx == arr.length){
int res[] = new int[fsf];
return res;
}
if(arr[idx] == x){
int res[] = allIndices(arr,x,idx+1,fsf+1);
res[fsf] = idx;
return res;
}else{
return allIndices(arr,x,idx+1,fsf);
}
}
3
accuracy
Dry run
Visualize
(idx, fsf)
(6,2)
(5,2)
(4,1)
(3,0)
(2,0)
(1,0)
(0,0)
4K
3
4

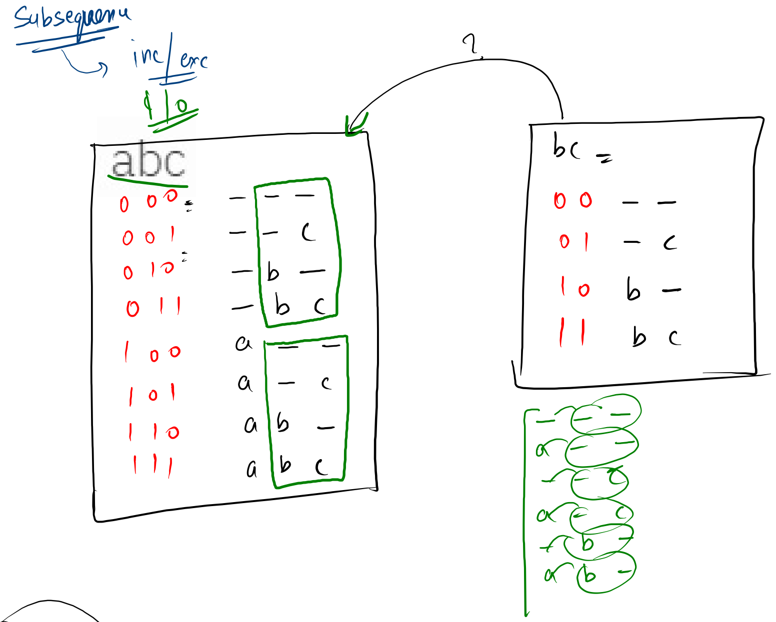


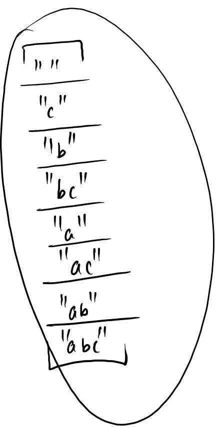
if (arr[idx] == x]) {

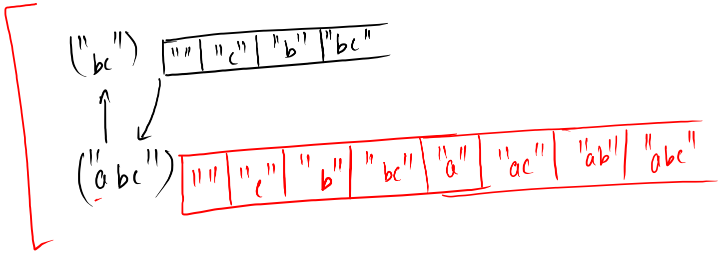
} else {

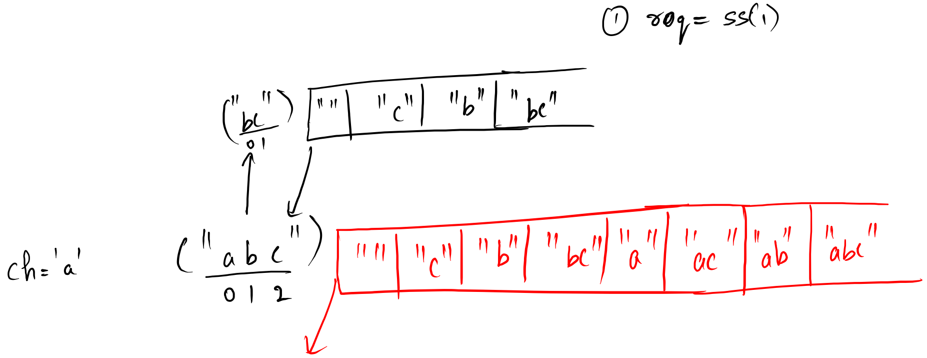


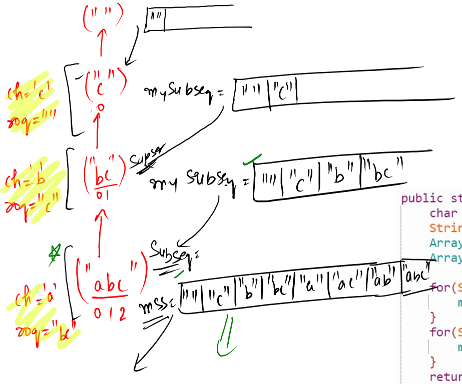
}



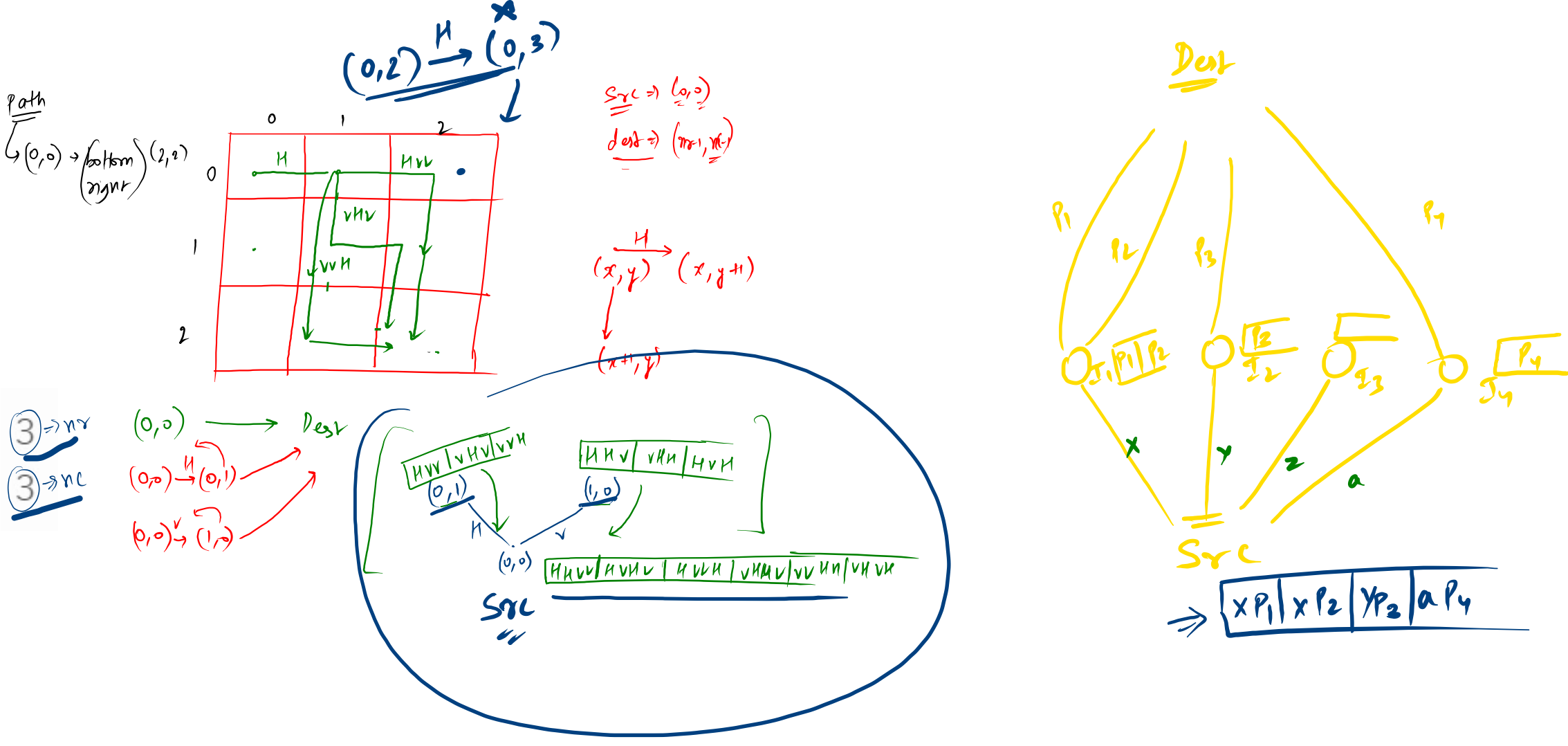


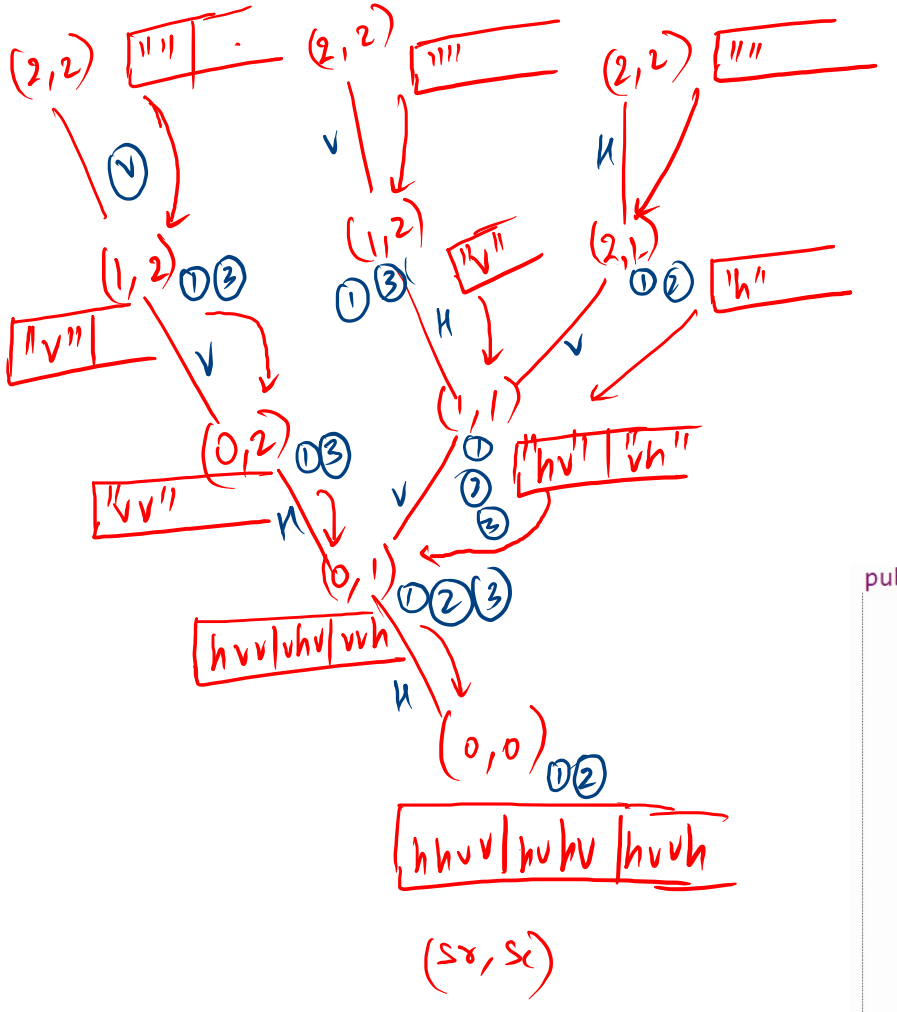






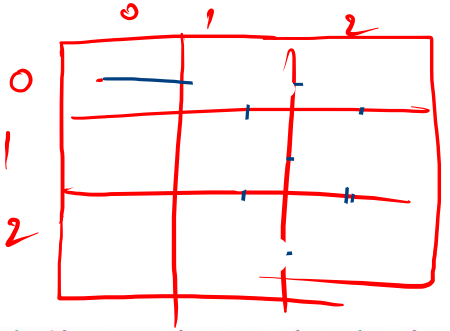
```
public static ArrayList<String> gss(String str) {
    char ch = str.charAt(0);
    String roq = str.substring(1);
    ArrayList<String> subSeq = gss(roq);
    ArrayList<String> mySubSeq = new ArrayList<>();

    for(String s : subSeq){
        mySubSeq.add(s);
    }
    for(String s : subSeq){
        mySubSeq.add(ch+s);
    }
    return mySubSeq;
}
```


Path
(0,0) → (bottom right) (2,2)
(0,2) H → (0,3)
Src → (0,0)
dest → (m-1, n-1)
H
HVV
VHV
VVH
(x,y) H → (x,y+1)
(x+1,y)
3 → nr
3 → nc
(0,0) → Dest
(0,0) H → (0,1)
(0,0) V → (1,0)
HVV | VHV | VVH
(0,1)
HHV | VHH | HVH
(1,0)
H
V
(0,0)
HHVV | HVHV | HVVH | VHHV | VVHH | VHVH
Src
Dest
P1
P2
P3
P4
J1
J2
J3
J4
x
y
z
a
Src
x P1 | x P2 | y P2 | a P4



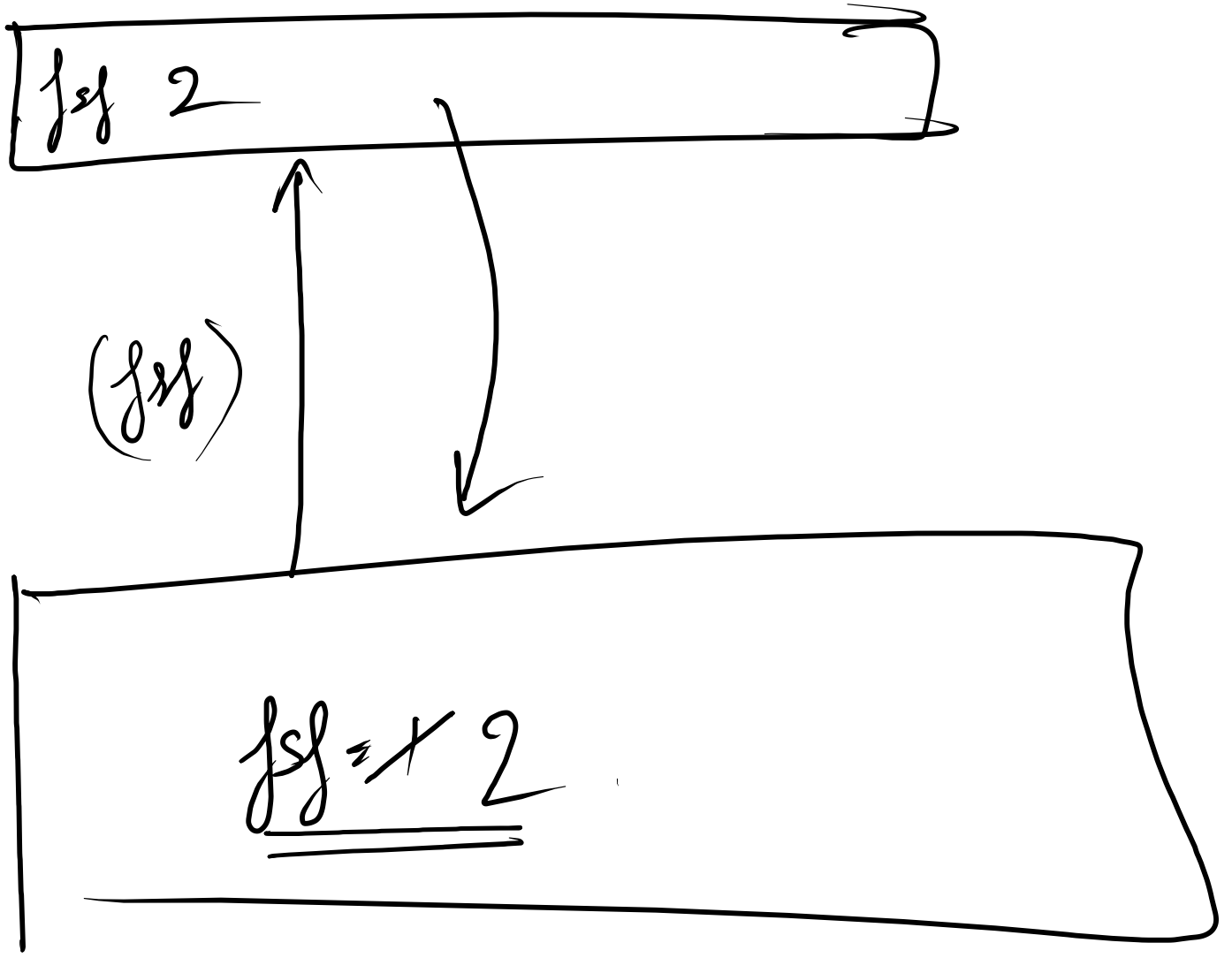
$dr = 2$

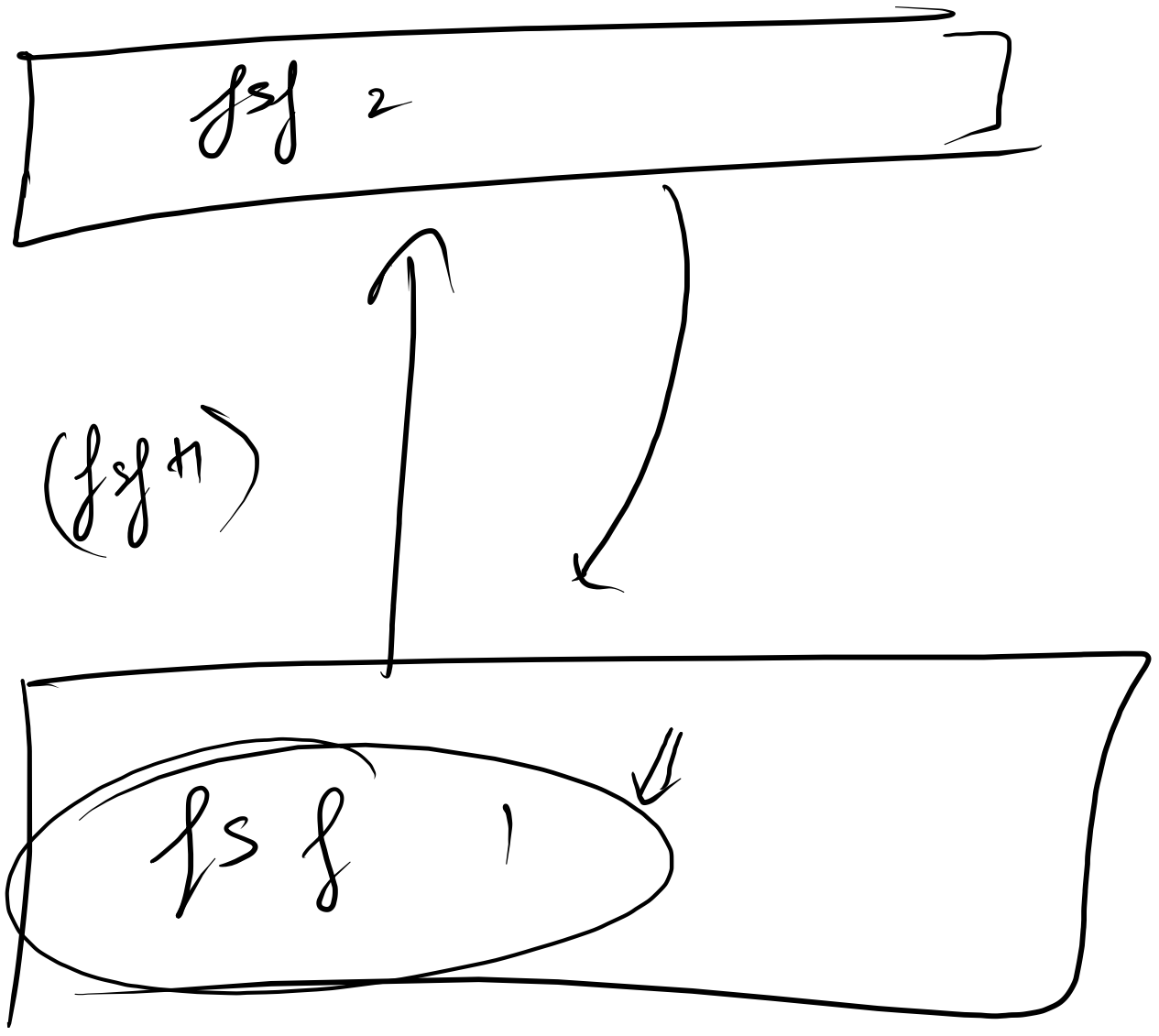
$dc = 2$




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

    ArrayList<String> myPath = new ArrayList<>();
    if(sc+1 <= dc){
        ArrayList<String> hPath = getMazePaths(sr,sc+1,dr,dc);
        for(String path : hpath){
            myPath.add("h"+path);
        }
    }
    if(sr+1 <= dr){
        ArrayList<String> vpath = getMazePaths(sr+1,sc,dr,dc);
        for(String path : vpath){
            myPath.add("v"+path);
        }
    }
    return myPath;
}
```

```java
int nr = scn.nextInt();
int nc = scn.nextInt();

ArrayList<String> path = getMazePaths(0,0,nr-1,nc-1);
System.out.println(path);
```


fsf 2
(fsf)
fsf = x 2


ʃsʃ 2
(ʃsʃ n)
ʃs ʃ 1