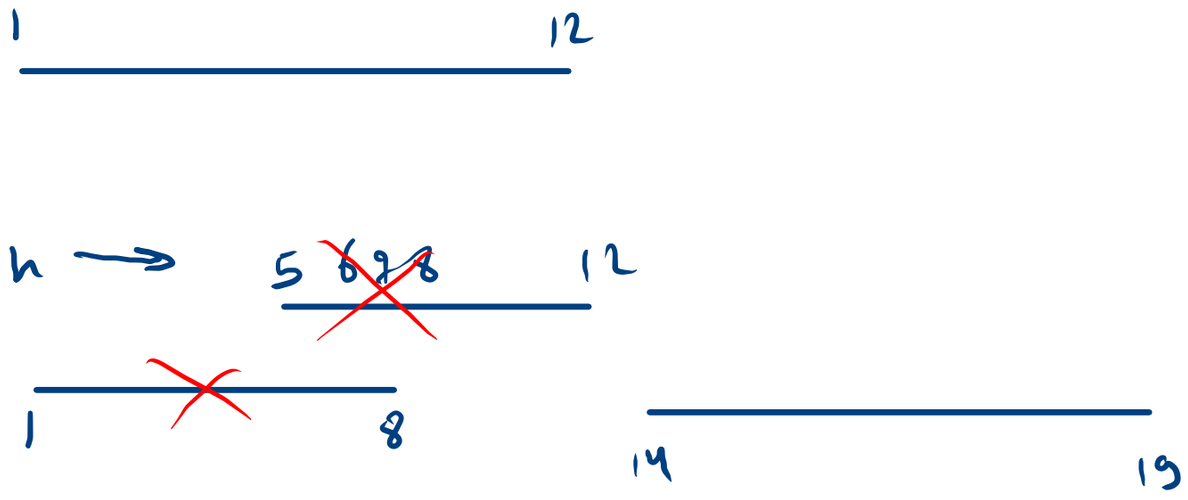
2D array

1 8 ✓
5 12 ✓
14 19 ✓
22 28 ✓
25 27 ✓
27 30 ✓



1	12
14	19
22	30

overlapping

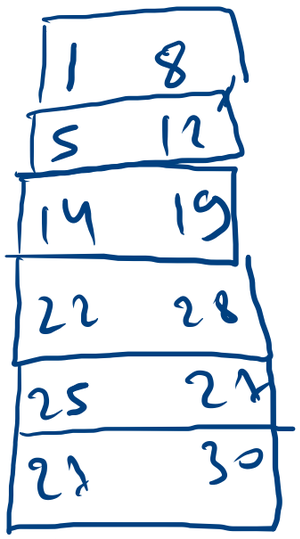
h.start ≤ p.end

new interval

start	end

min (h.start , p.start)

max (h.end, p.end)



6

22 28

1 8 ✓

25 27

14 19

27 30

5 12

overlapping

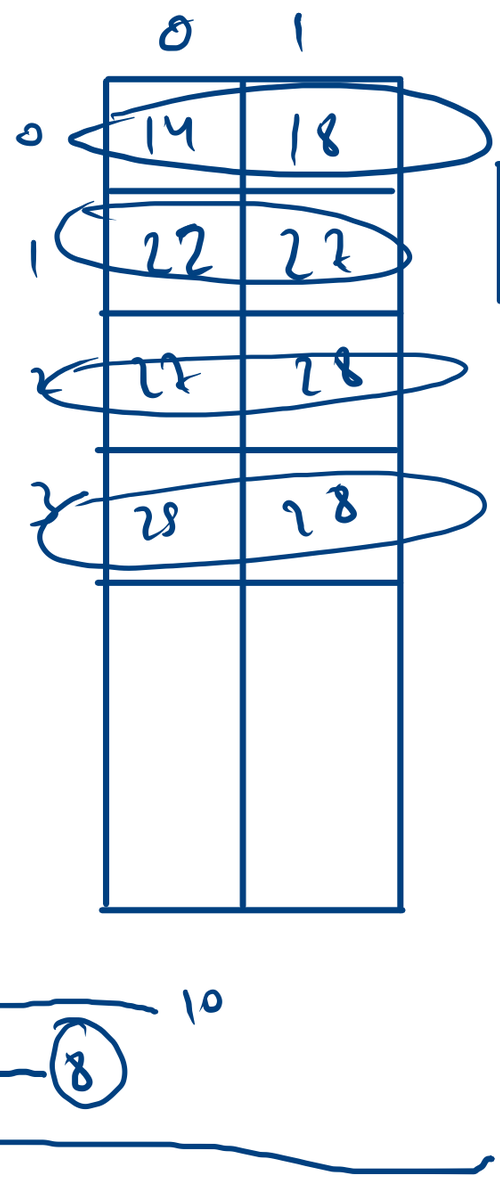
h.start ≤ p.end

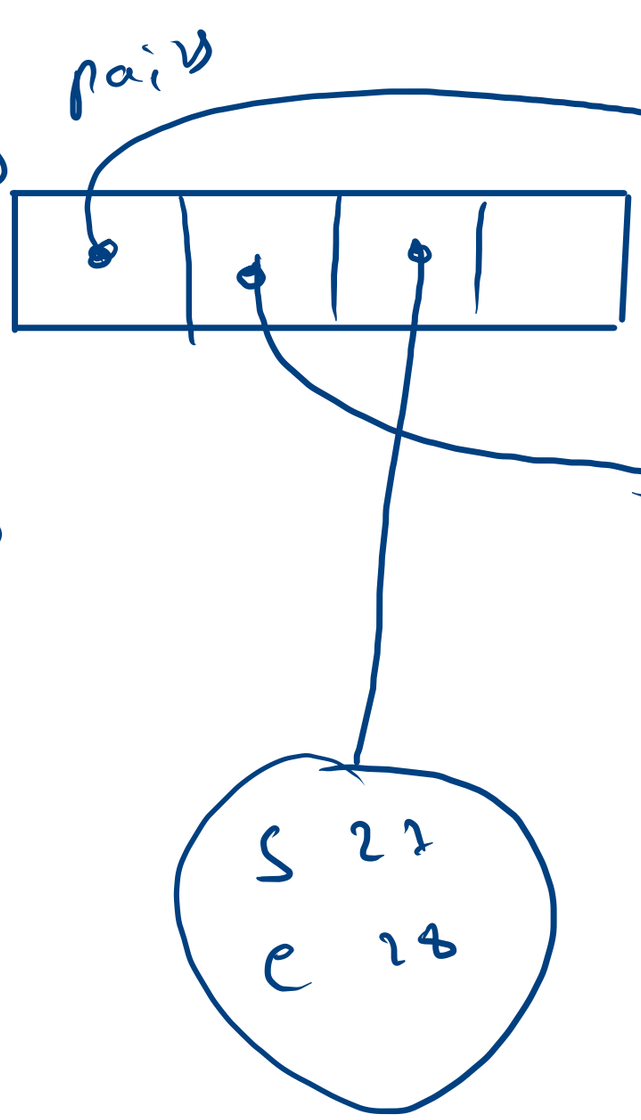
new interval

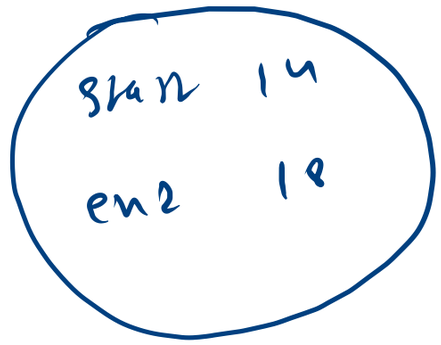
start end

min (h.start , p.start)

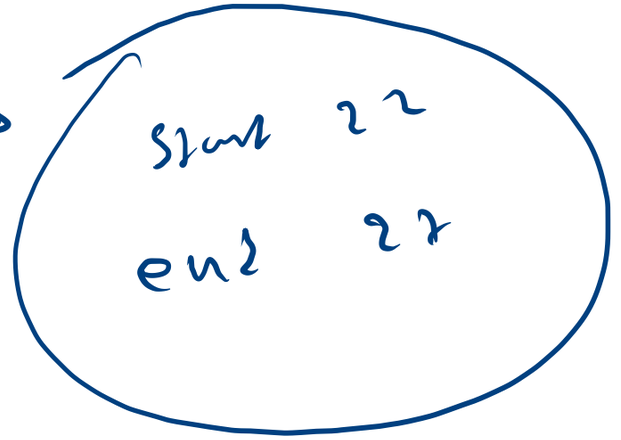
max (h.end, p.end)



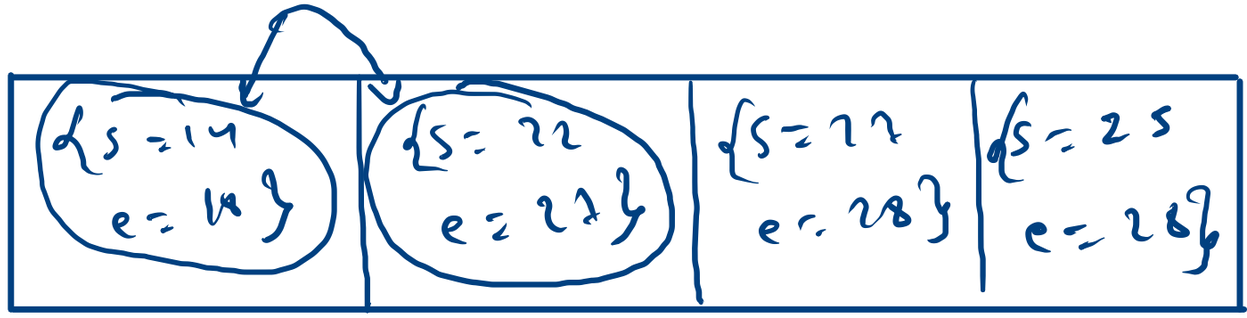




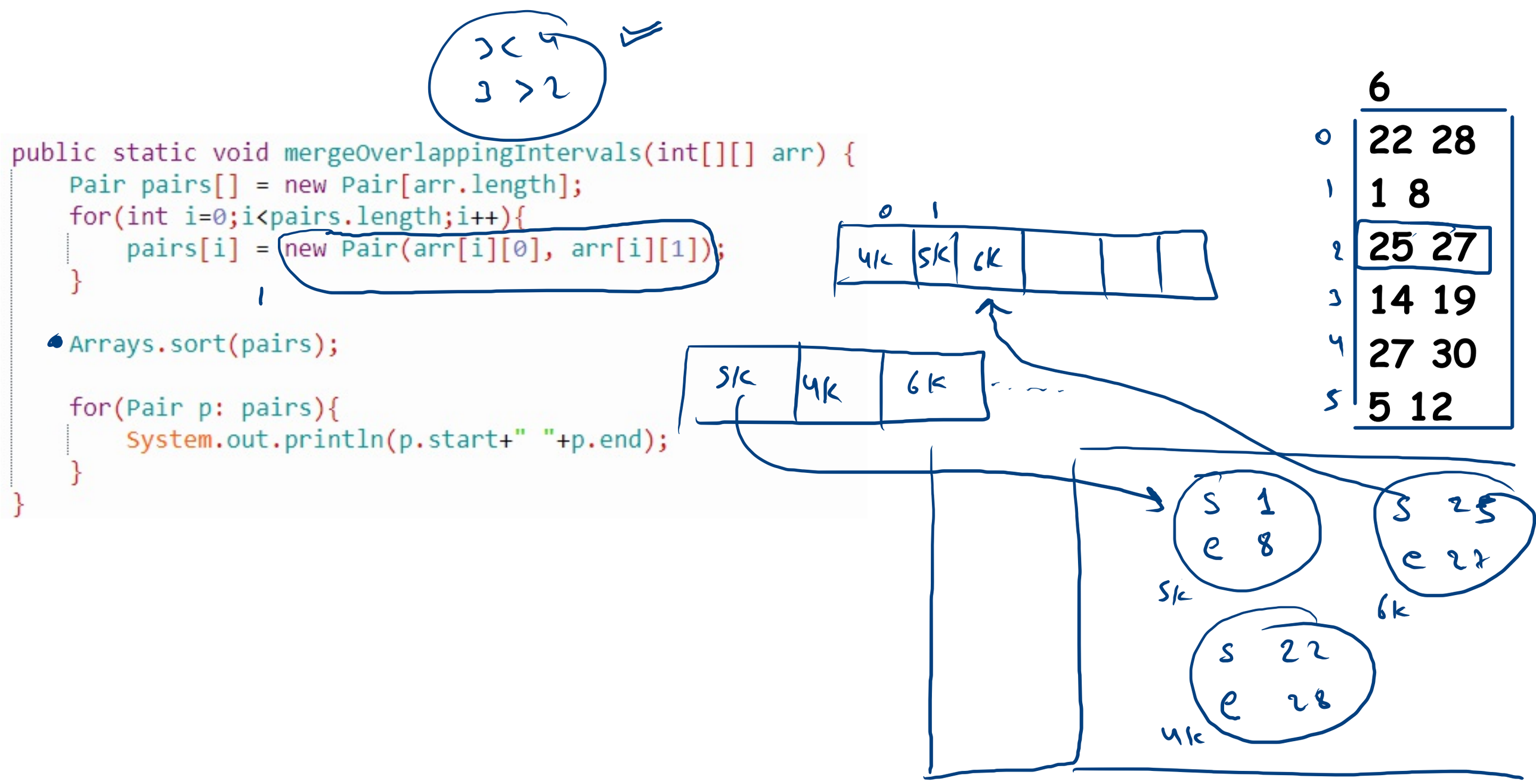
Arrays.sort



binary search



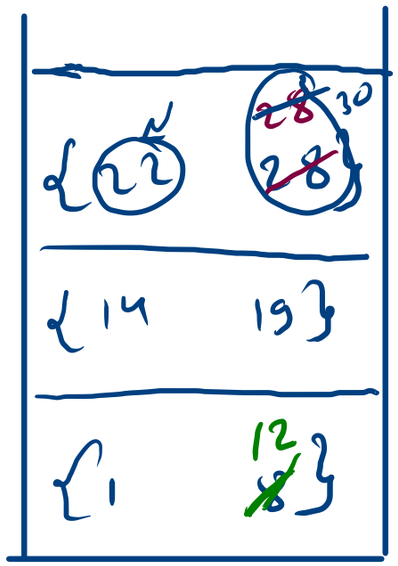
```java
public static void mergeOverlappingIntervals(int[][] arr) {
    Pair pairs[] = new Pair[arr.length];
    for(int i=0;i<pairs.length;i++){
        pairs[i] = new Pair(arr[i][0], arr[i][1]);
    }

    Arrays.sort(pairs);

    for(Pair p: pairs){
        System.out.println(p.start+" "+p.end);
    }
}
```

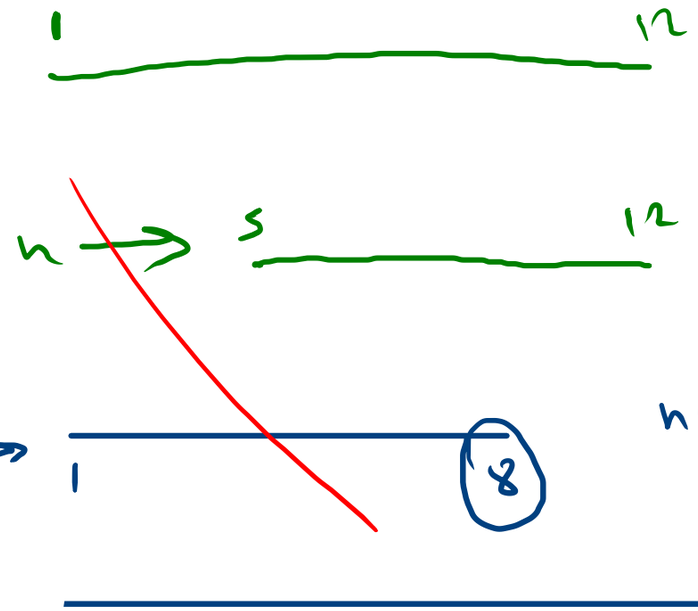


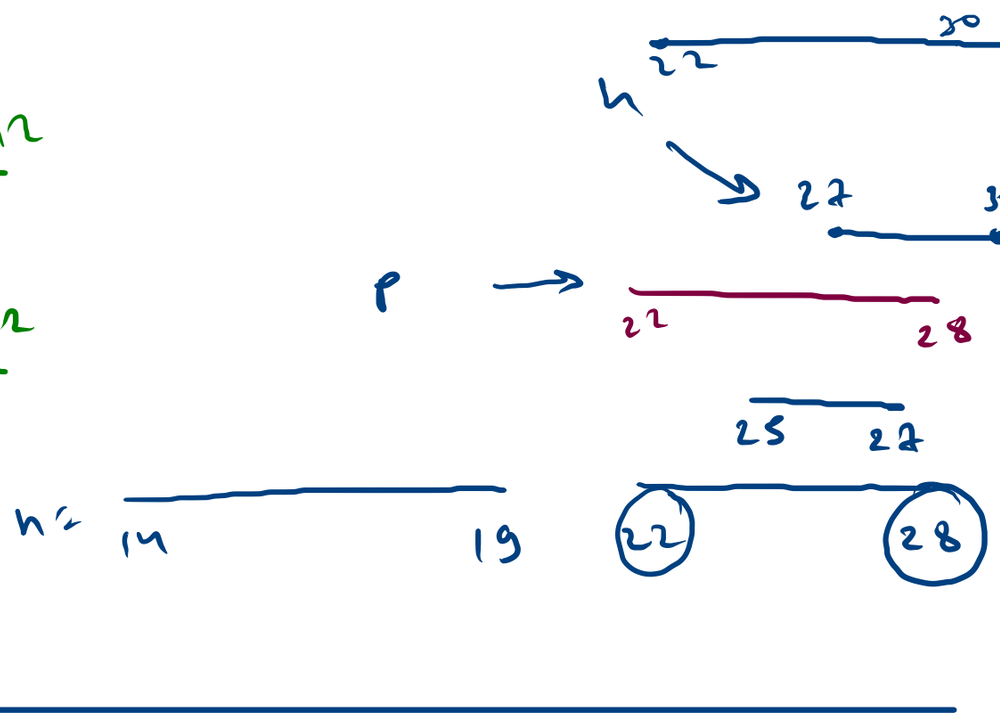
[uk sk] a[0].compareTo(a[1])

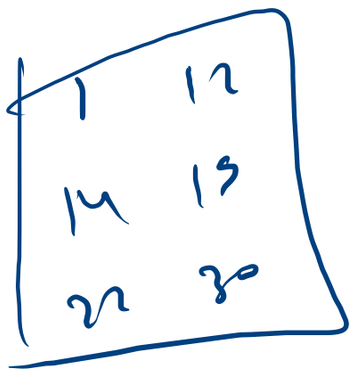
0 1

1 8 ✓
5 12 ✓
14 19 ✓
22 28 ✓
25 27 ✓
27 30 ✓









overlapping

h.start ≤ p.end

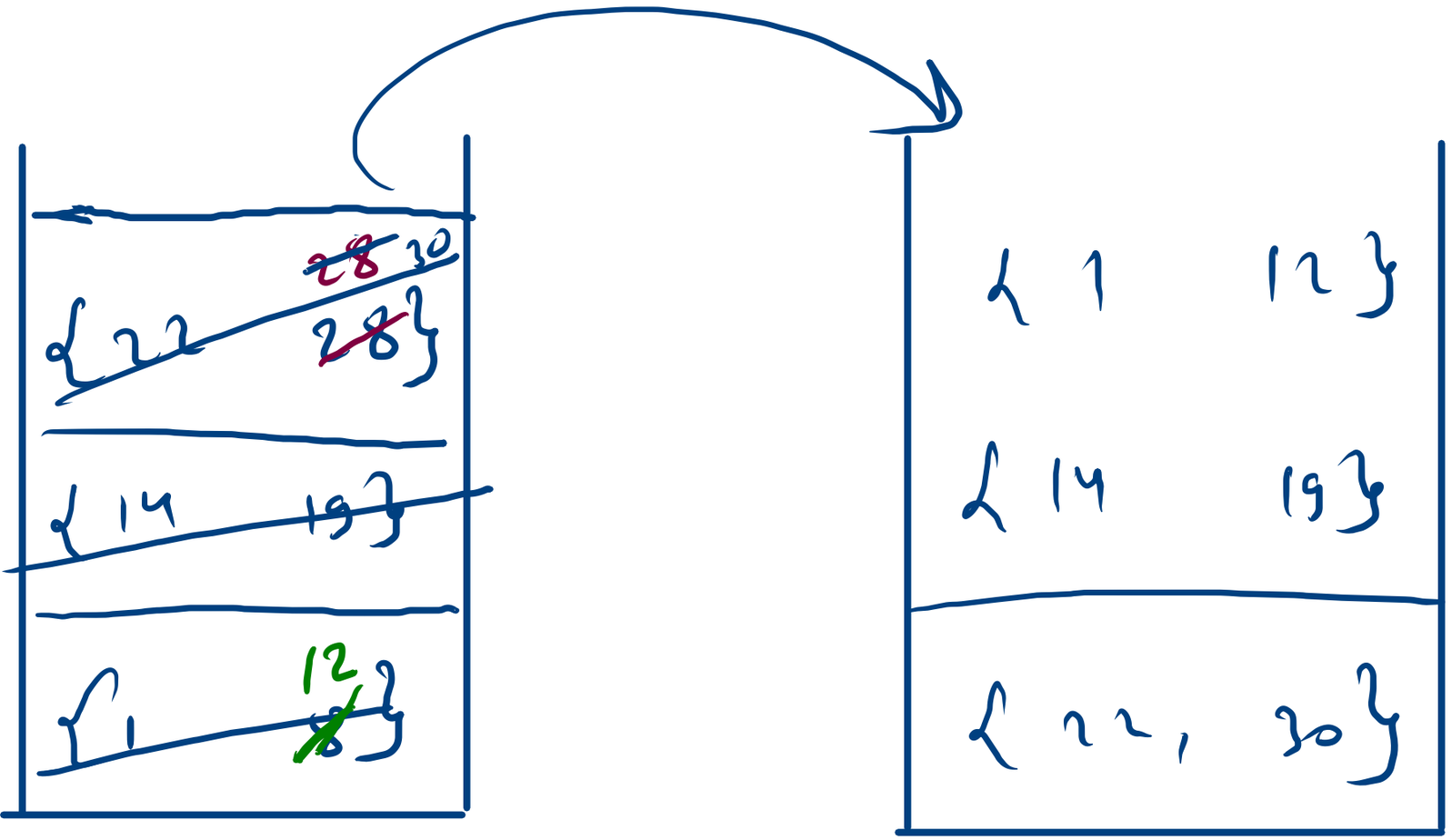
24 ≤ 28

new interval

start end

~~min (h.start, p.start)~~ p.start

max(h.end, p.end)


{ 22 28 30 }
{ 14 19 }
{ 1 8 12 }
{ 1 12 }
{ 14 19 }
{ 22, 30 }

i d

1 2 5 4 3

3 5 8 2 1

3 — i — 5 — i — 8 — d — 2 — d — 1

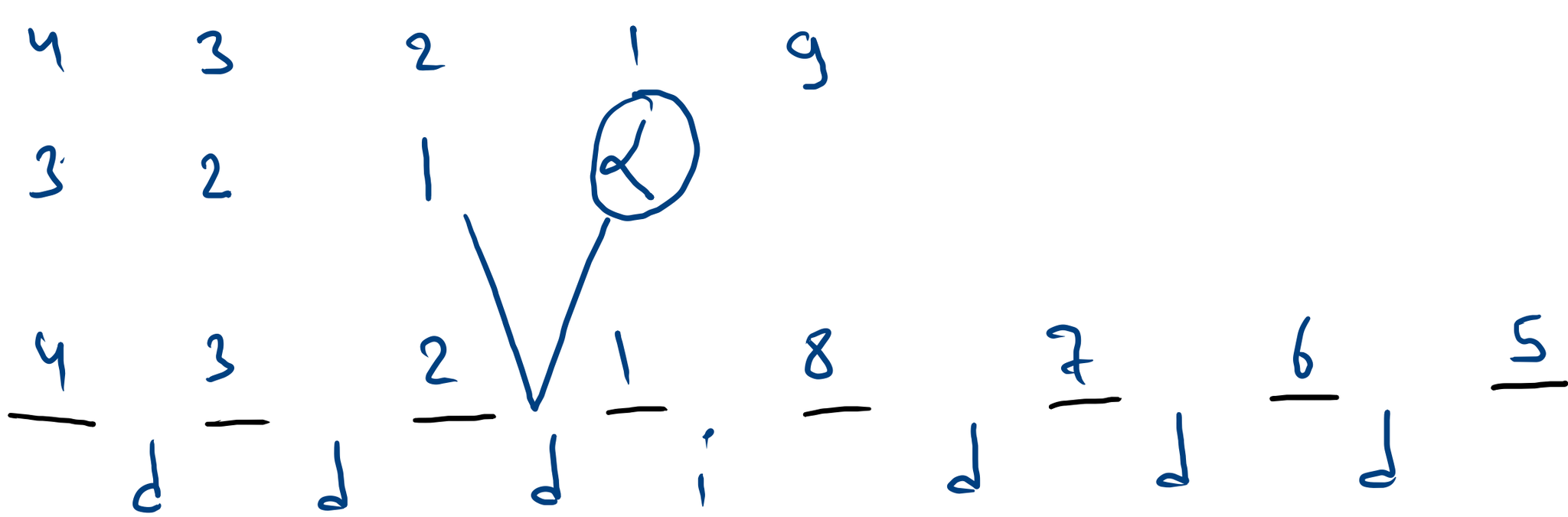
1

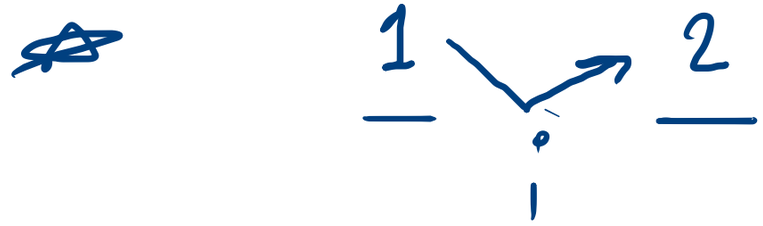
9

3 5 8 2 1

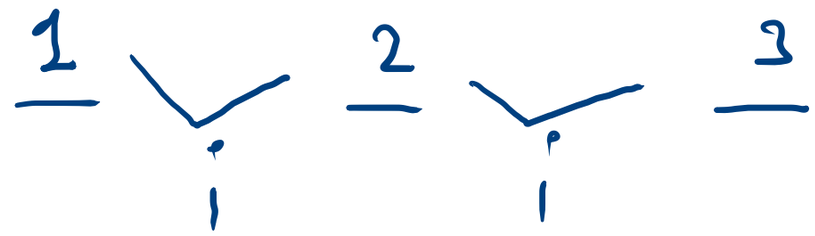
1 2 5 4 3

1
2
5
4
3

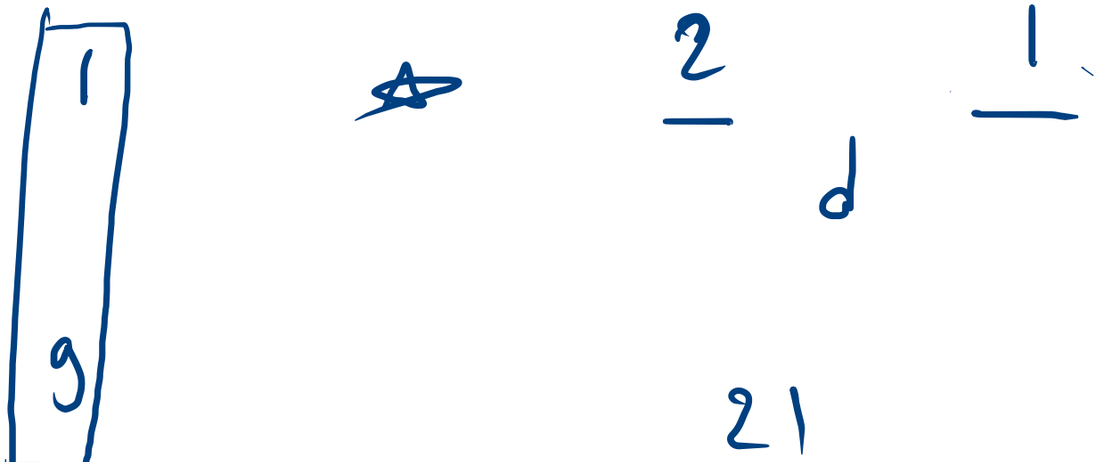




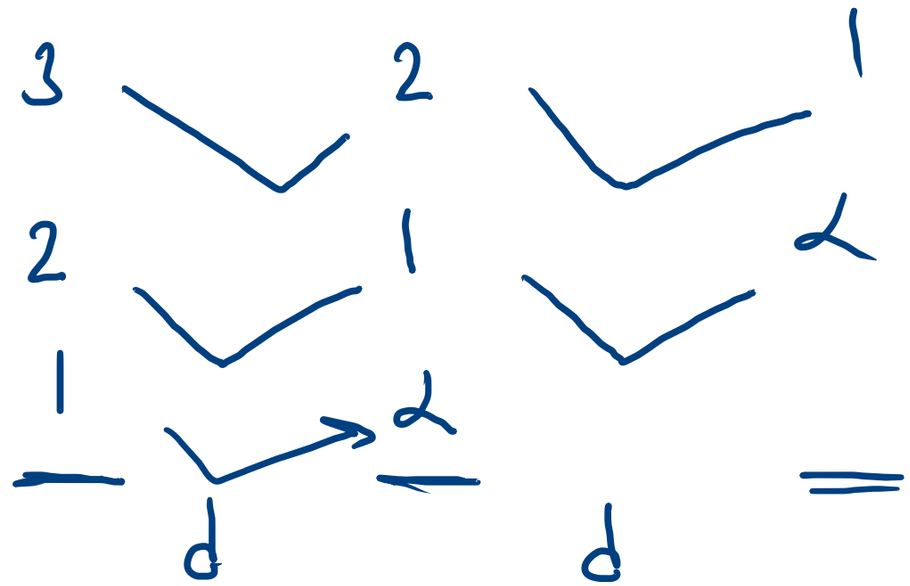
12



1 2 3

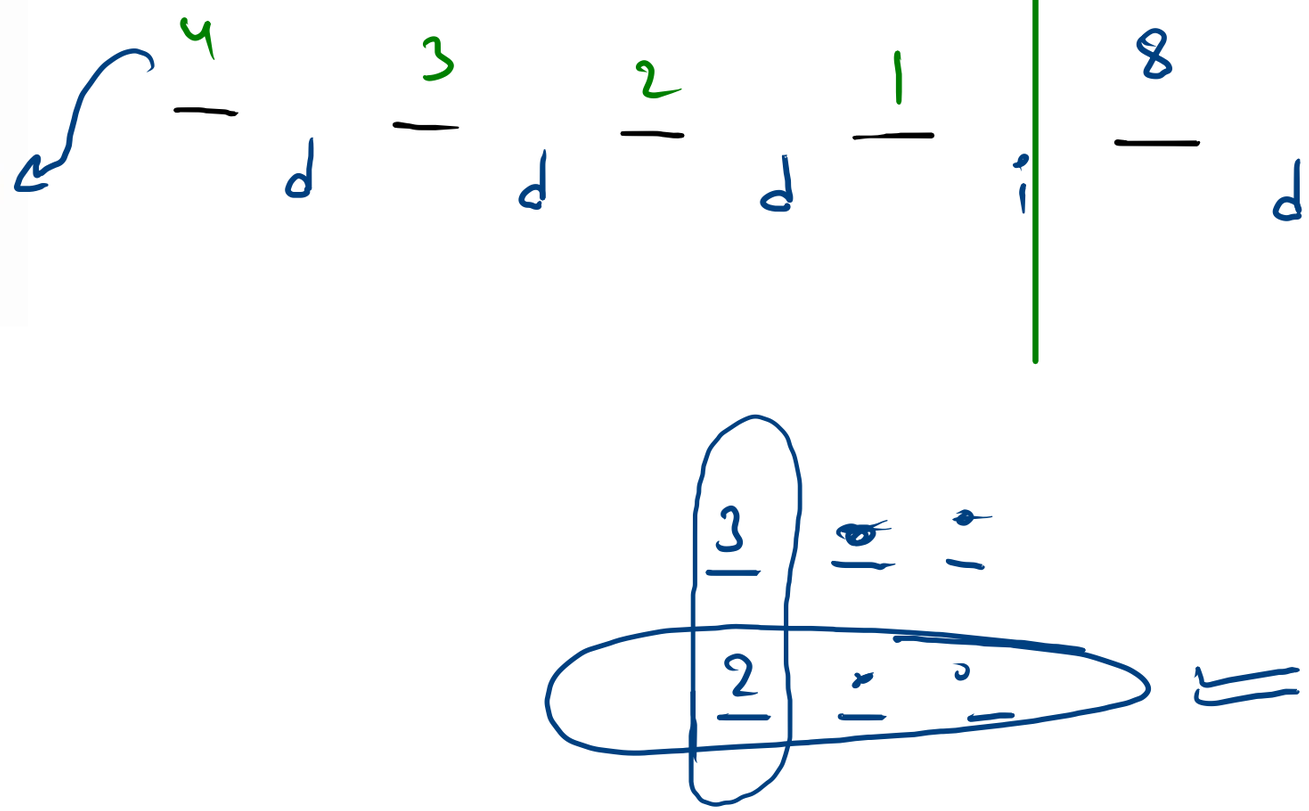


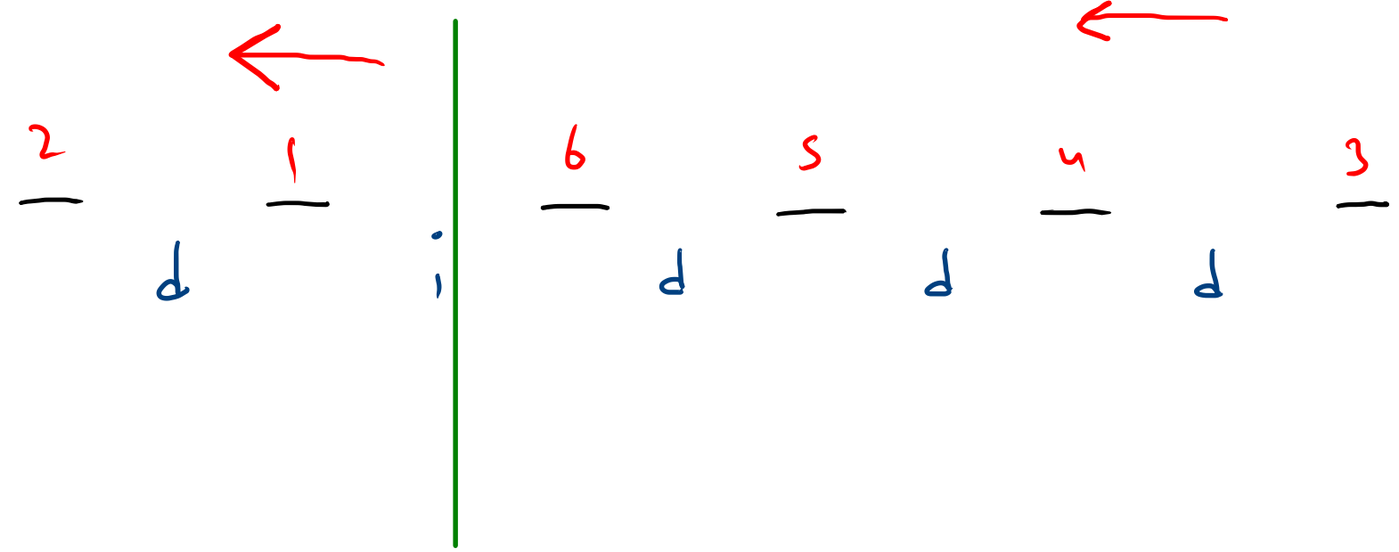
21



3 21

d -> 21
i -> 12
ddd -> 4321
iii -> 1234
•dddiddd -> 43218765
iiddd -> 126543





d -> 21

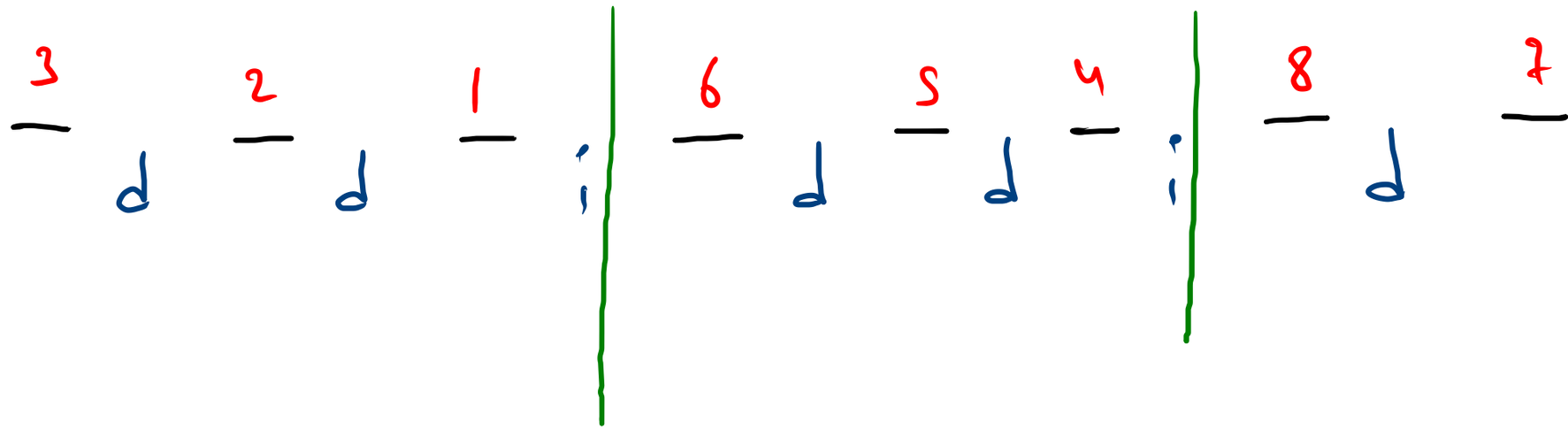
i -> 12

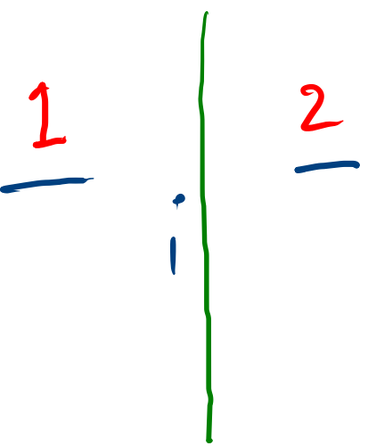
ddd -> 4321

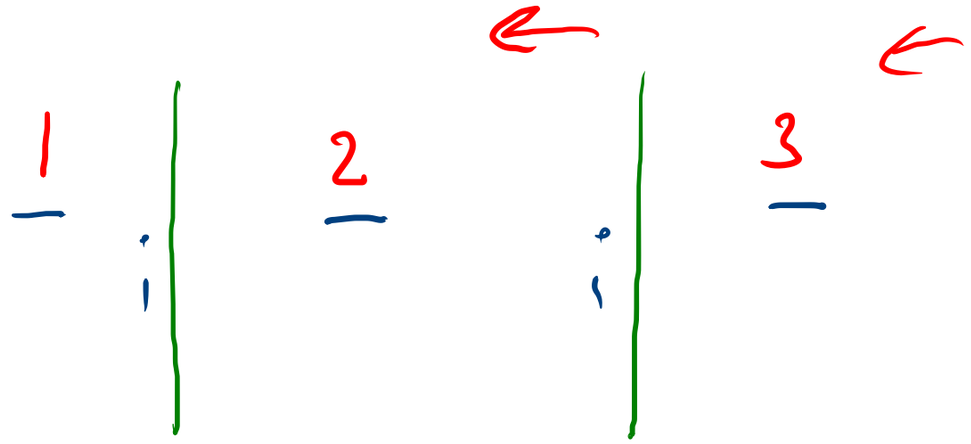
iii -> 1234

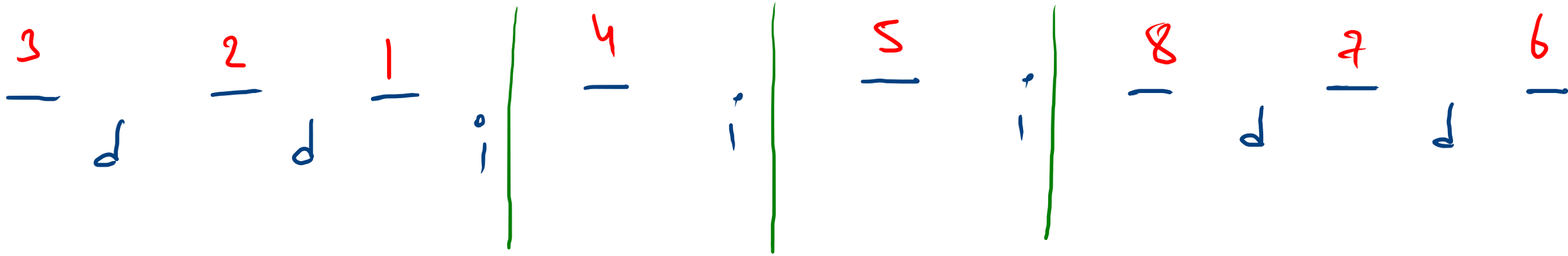
dddiddd -> 43218765

iiddd -> 126543

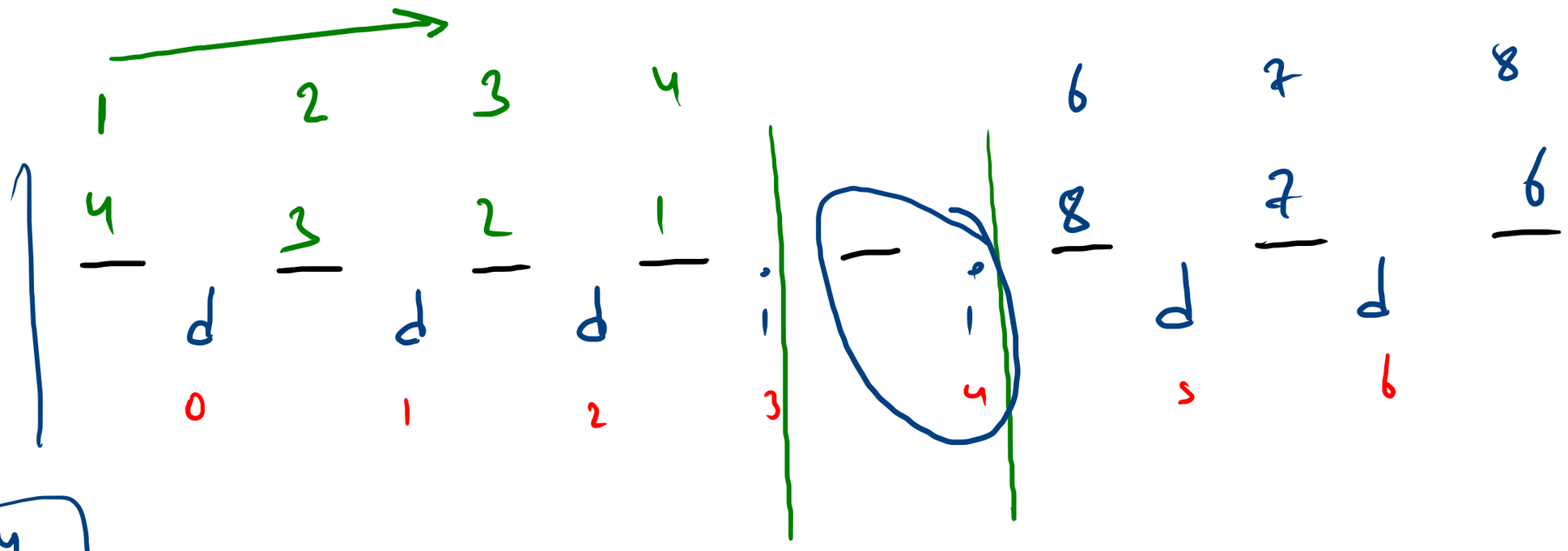


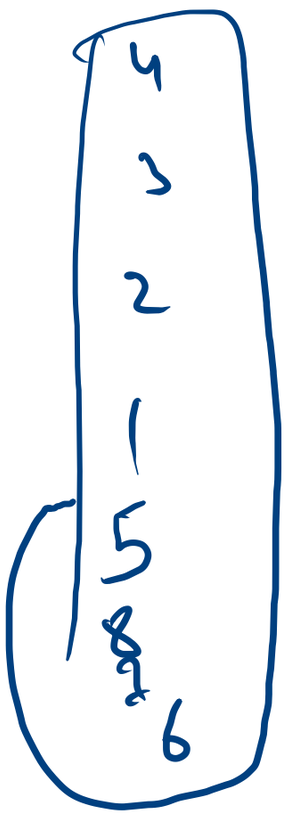

1
2
i

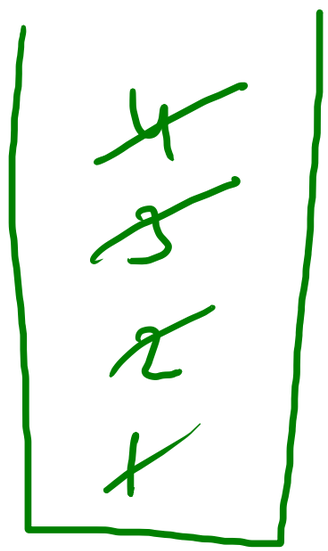

1
2
3
i
i

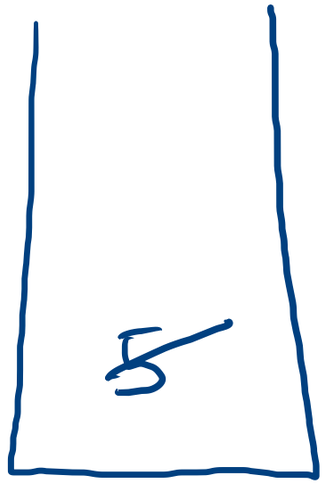

3
2
1
4
5
8
7
6
i
i
i

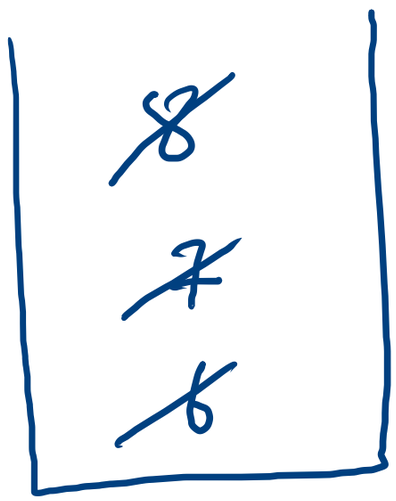
val = 1 2 3 4 5 6 7 8 9









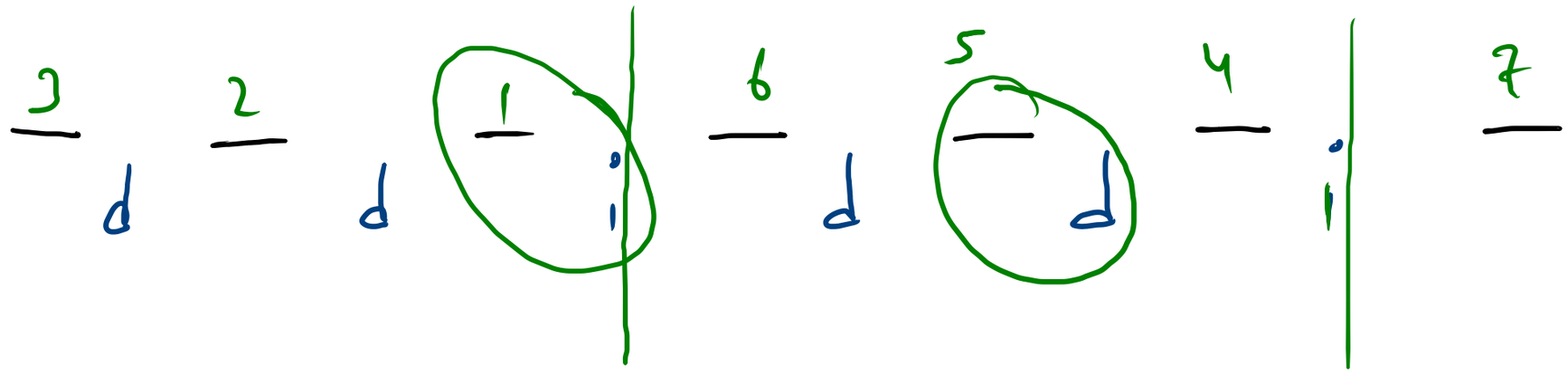


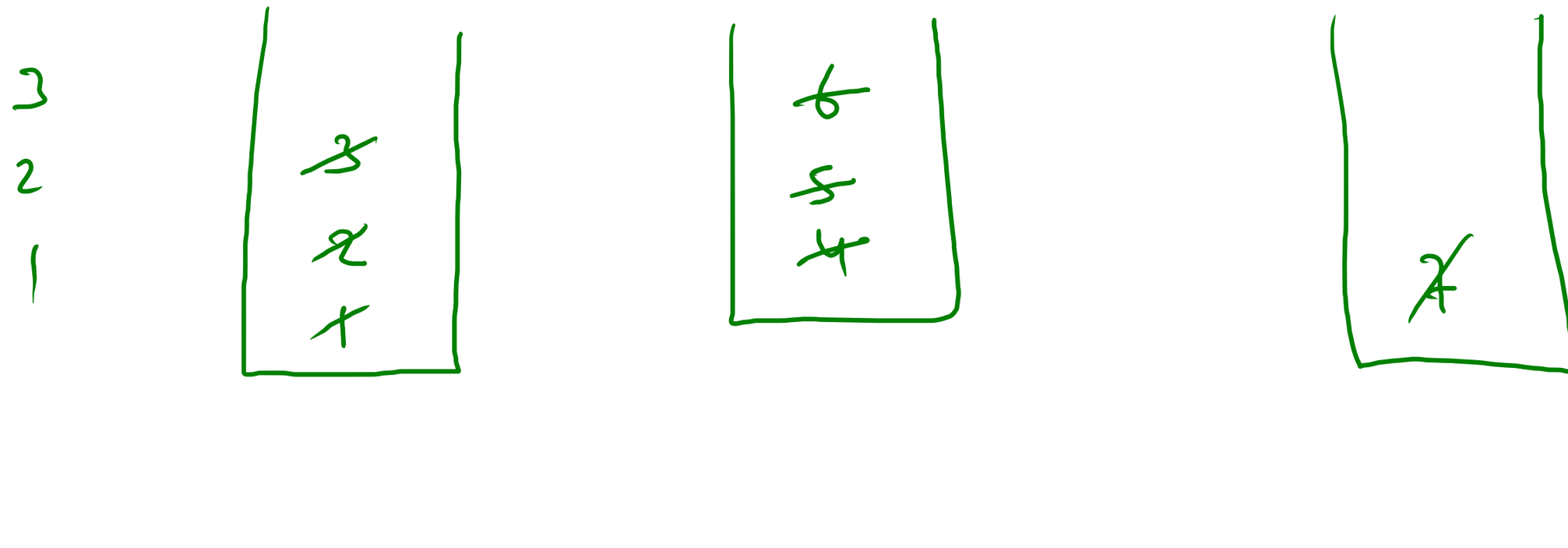
val

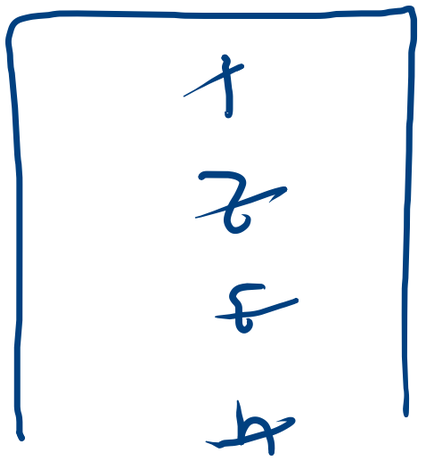
1 2 3 4 5 6 7

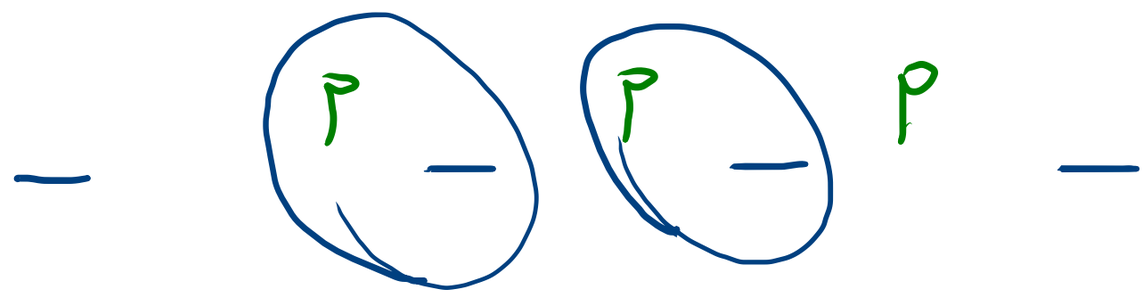
d push ++ i push ++ pop print

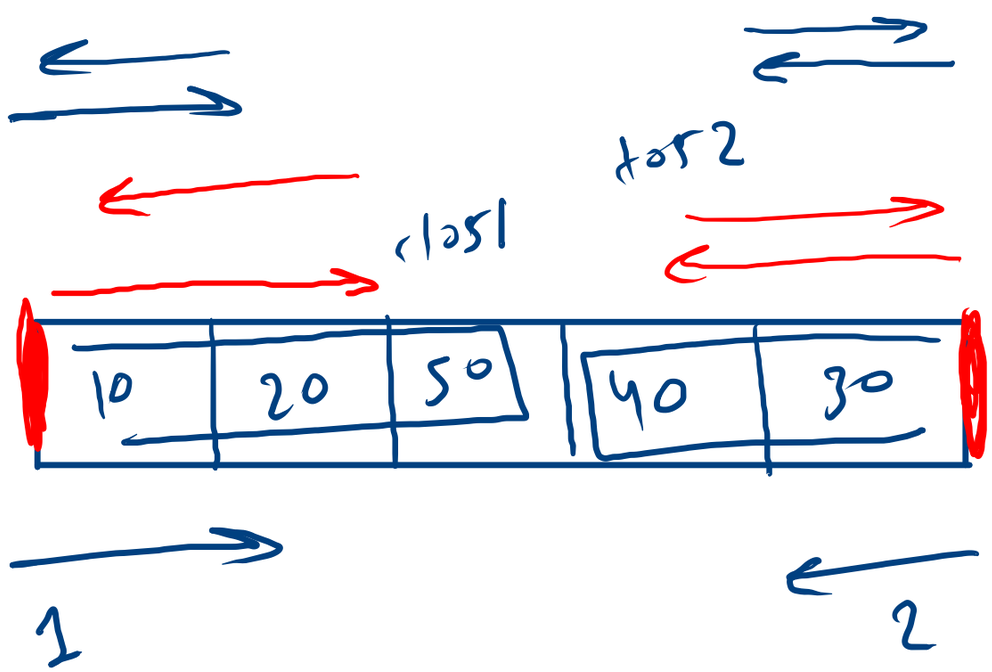
push print

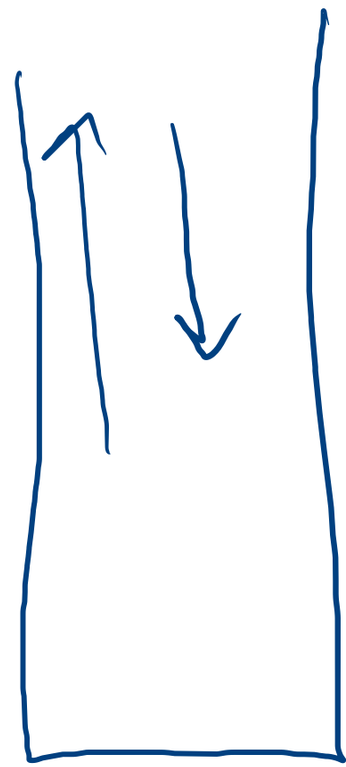












first

push1 10

push 20

push2 30

push 2 40

push 1 50

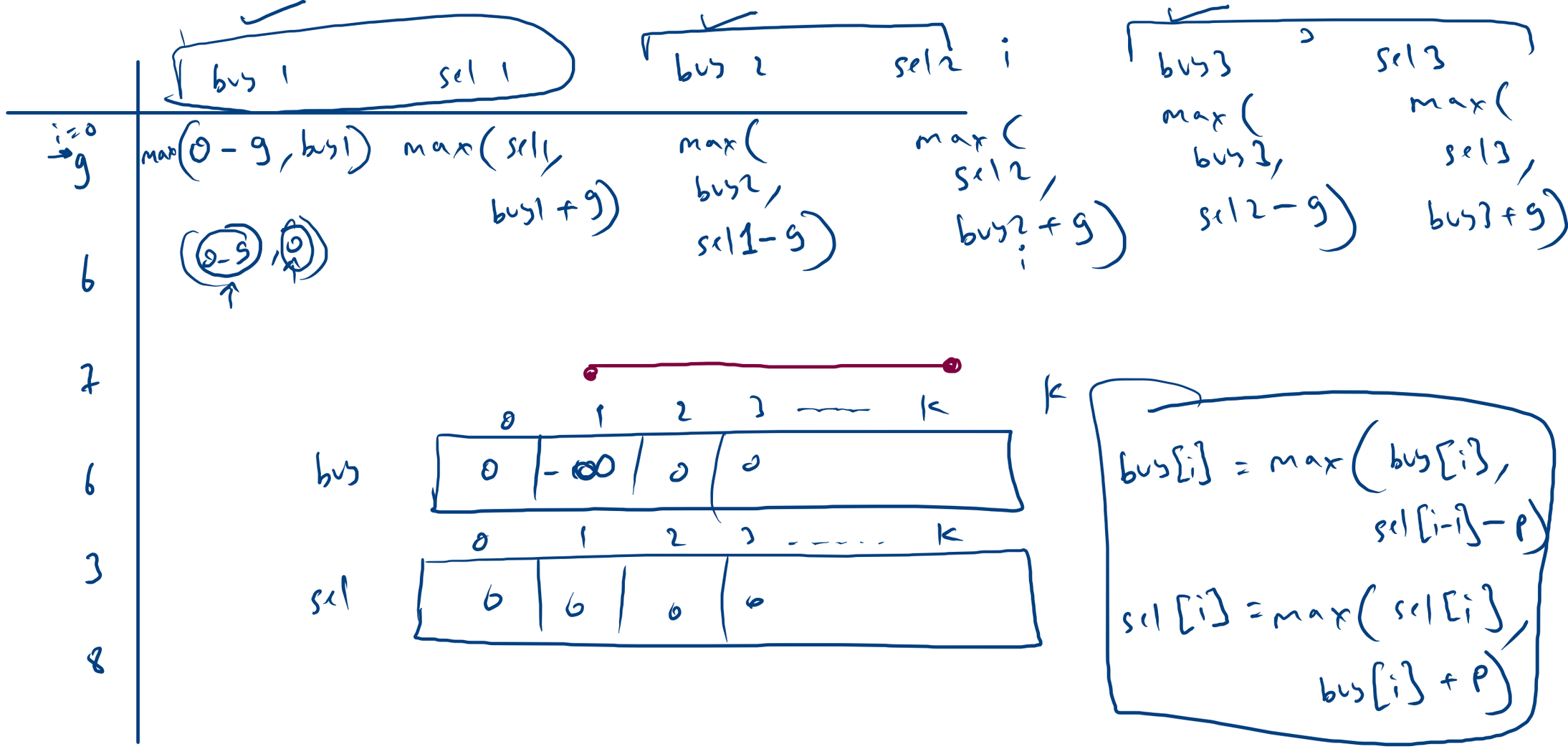
push 1 ✗

push 2 ✗ error

6

9 6 7 6 3 8

1


buy 1
sel 1
buy 2
sel 2
i
buy 3
sel 3
i=0
g
max(0-g, buy1)
max(sel1, buy1+g)
max(buy2, sel1-g)
max(sel2, buy2+g)
max(buy3, sel2-g)
max(sel3, buy3+g)
0-g
0
6
7
6
3
8
buy
sel
0 1 2 3 ---- k
0 -∞ 0 0
0 1 2 3 ---- k
6 6 0 6
k
buy[i] = max(buy[i], sel[i-1]-p)
sel[i] = max(sel[i], buy[i]+p)

6

3 5 1 4 1 2 4 7 7 8 1 8 2 5 1 5 6

buy

sel

```java
int buy[] = new int[k+1];
int sel[] = new int[k+1];
Arrays.fill(buy, -(int)1e9);

for(int i=1;i<n;i++){
    int price = cost[i];

    for(int c=1;c<=k;c++){
        buy[c] = Math.max(buy[c], sel[c-1]-price);
        sel[c] = Math.max(sel[c], buy[c]+price);
    }
}
```