$$\begin{array}{cccccccc} & & i & \xrightarrow{} & i & & & \\ a & \to & [\,2 & & 4 & 4 & 7 & 8 \quad 10\,] \end{array}$$

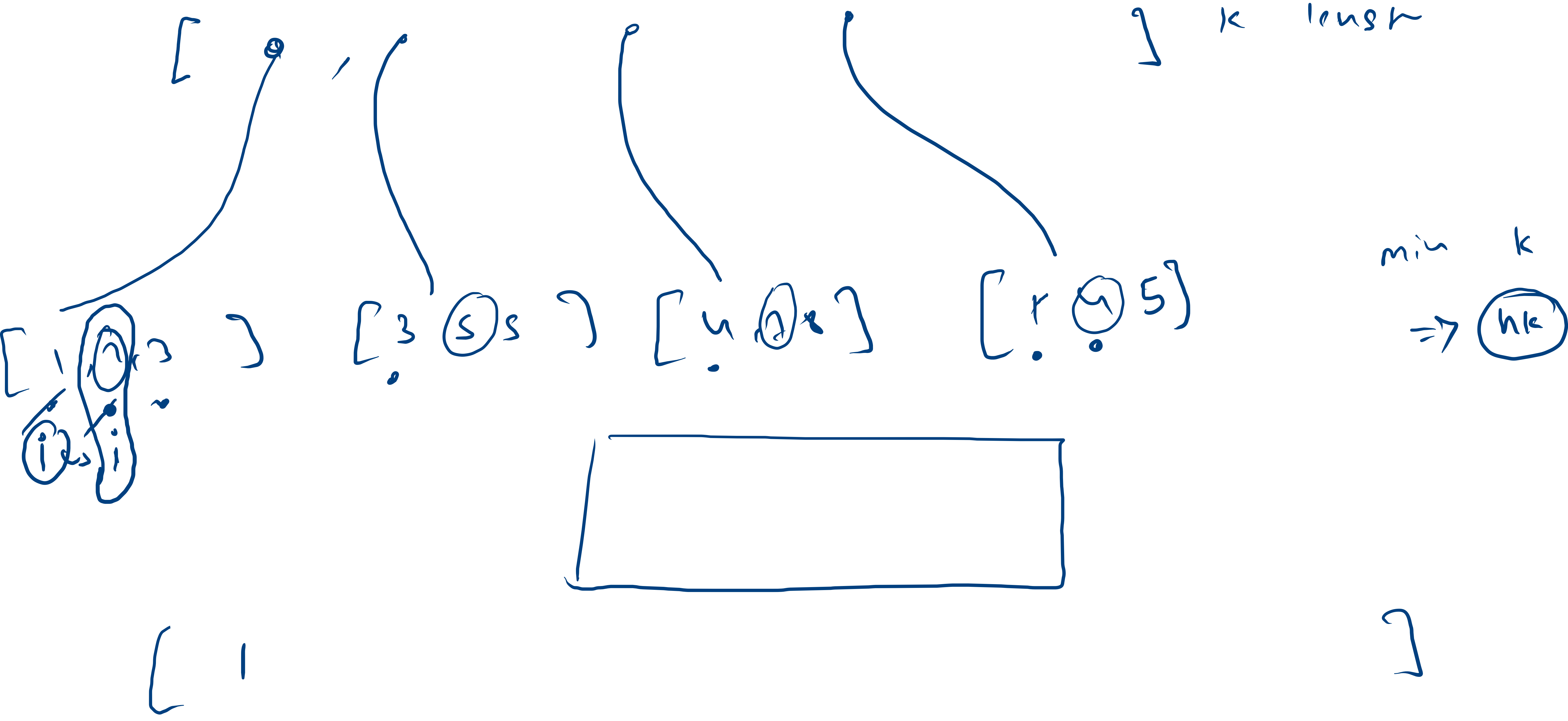
$$\begin{array}{ccccccc} b & \to & [\,1 & 3 & 4 & 6 & 7\,] \\ & & j & j & & & \end{array}$$

$$\begin{array}{cccccc} c & \to & [\,5 & 7 & 10 &] \\ & & m & & & \end{array}$$

$\boxed{k}$ ↙

z → [________]

ans → [1 2 3 4 4 4 6 7 7 8 10]


k length
min k
⇒ nk
[1 2 3]
[3 5 5]
[4 7 8]
[1 4 5]
i
j
[1]

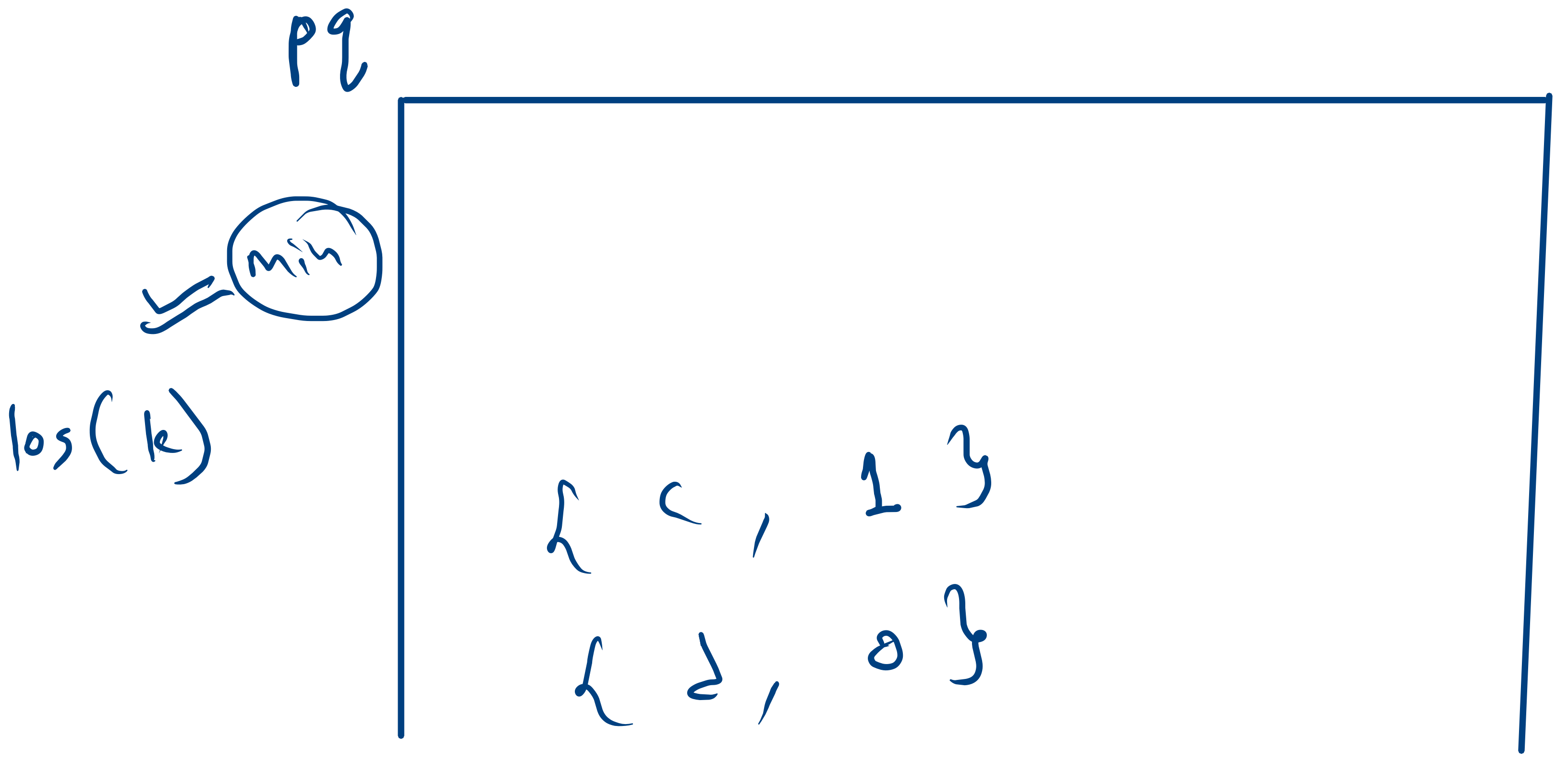
~~a [1 3 5]~~

~~b [2 3 5]~~

c [1 4 7]

d [3 7 9]

[1 , 1 , 2 , 3 , 3 3 , 4 , 5 , 5]



6k [2 7]

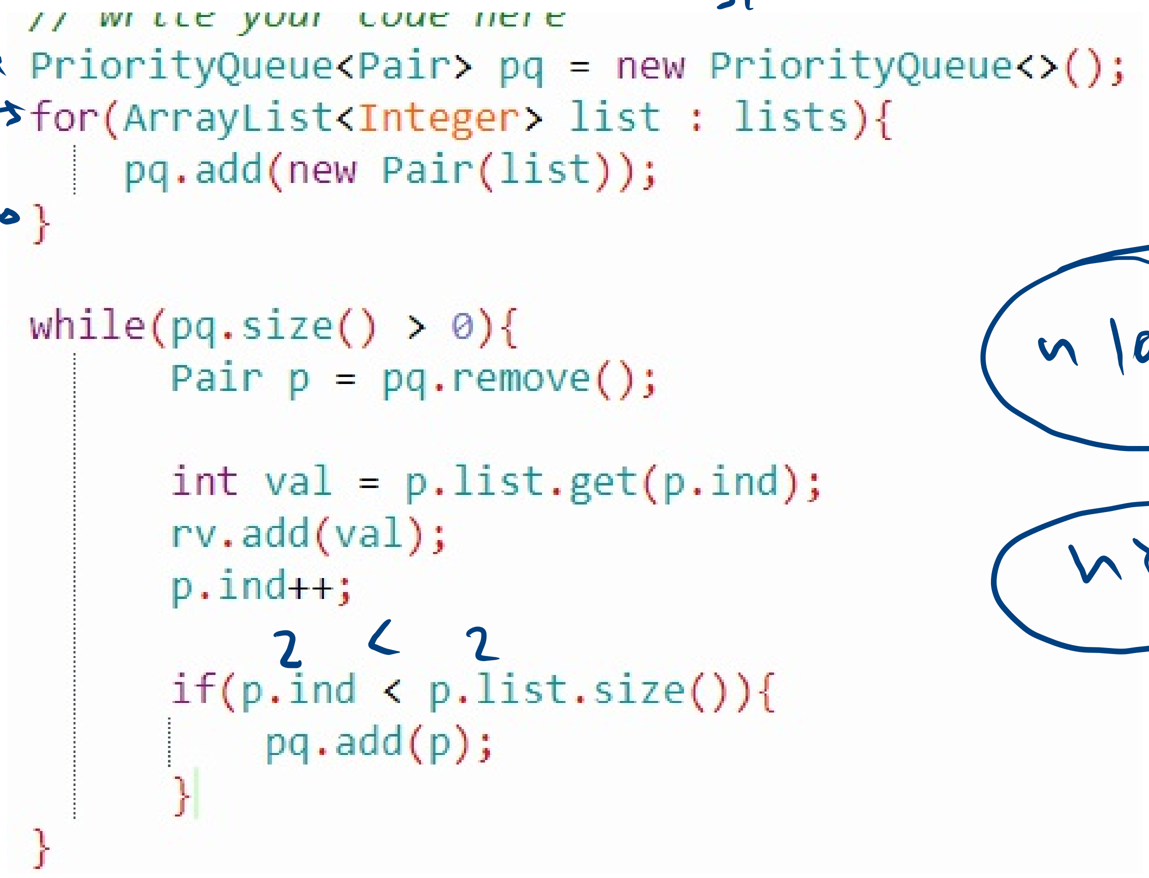
4k [1 2 3].

5k [2 4].

list [4k, 5k, 6k]

rv = [1, 2, 2, 2, 3, 4, 7]

p = { 6k, 2 }



```
// write your code here
PriorityQueue<Pair> pq = new PriorityQueue<>();
for(ArrayList<Integer> list : lists){
    pq.add(new Pair(list));
}

while(pq.size() > 0){
      Pair p = pq.remove();

      int val = p.list.get(p.ind);
      rv.add(val);
      p.ind++;
        2  <  2
      if(p.ind < p.list.size()){
          pq.add(p);
      }
}
```

n log(k)

n×k

AL   int

{ 4k, 1 }
{ 5k, 2 }

```
class person {
    String name,
    int age
}
```

{ 2 7 3 5 1 }

int

{A,10}
{C,15}
{B,20}

age

{"A", 10} ←
{"B", 20}
{"C", 15}

	return
a < b	-ve
a = b	0
a > b	+ve

smaller

$\cancel{2}$ 7 3

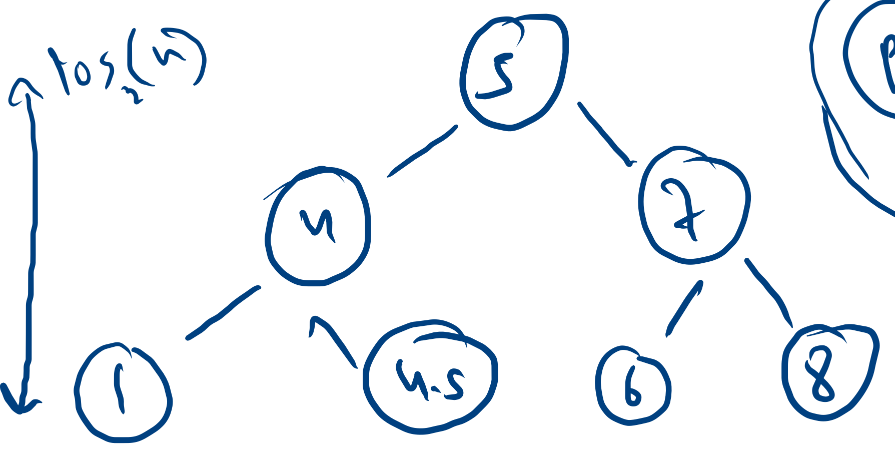
5 $\cancel{1}$

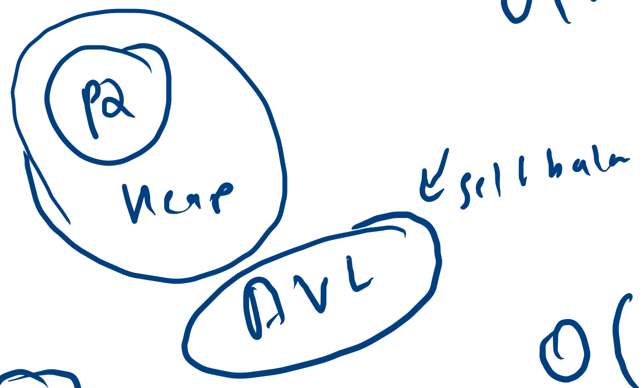
pq

	I	II	III	heap
☆ add	O(1)	O(n)	log(n)	log(n)
remove	O(n)	O(1)	log(n)	log(n)
peek	O(n)	O(1)	log(n)	O(1)
size				

[~~1~~, 2, 3, 4, 5, 7]

n

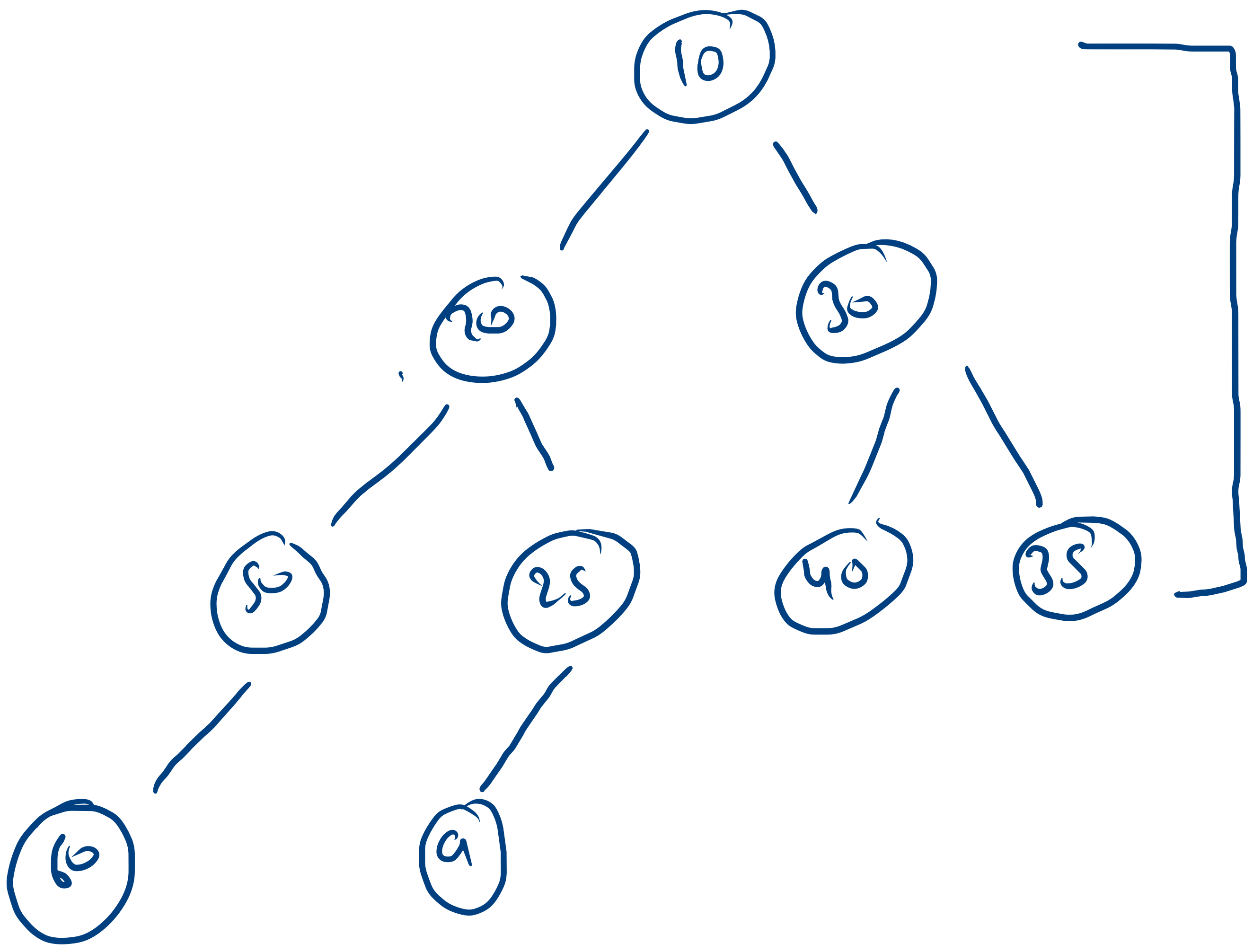




II	I
$O(n^2)$	$O(n)$
$O(n)$	$O(n^2)$

```
for( val : arr){
    pq.add(val)
}
while( pq.size > 0){
  2n  n  print (pq.peek)
      n  pq.remove
}
```

Heap



- Hop

p → ch: high → less

- CBT left → right

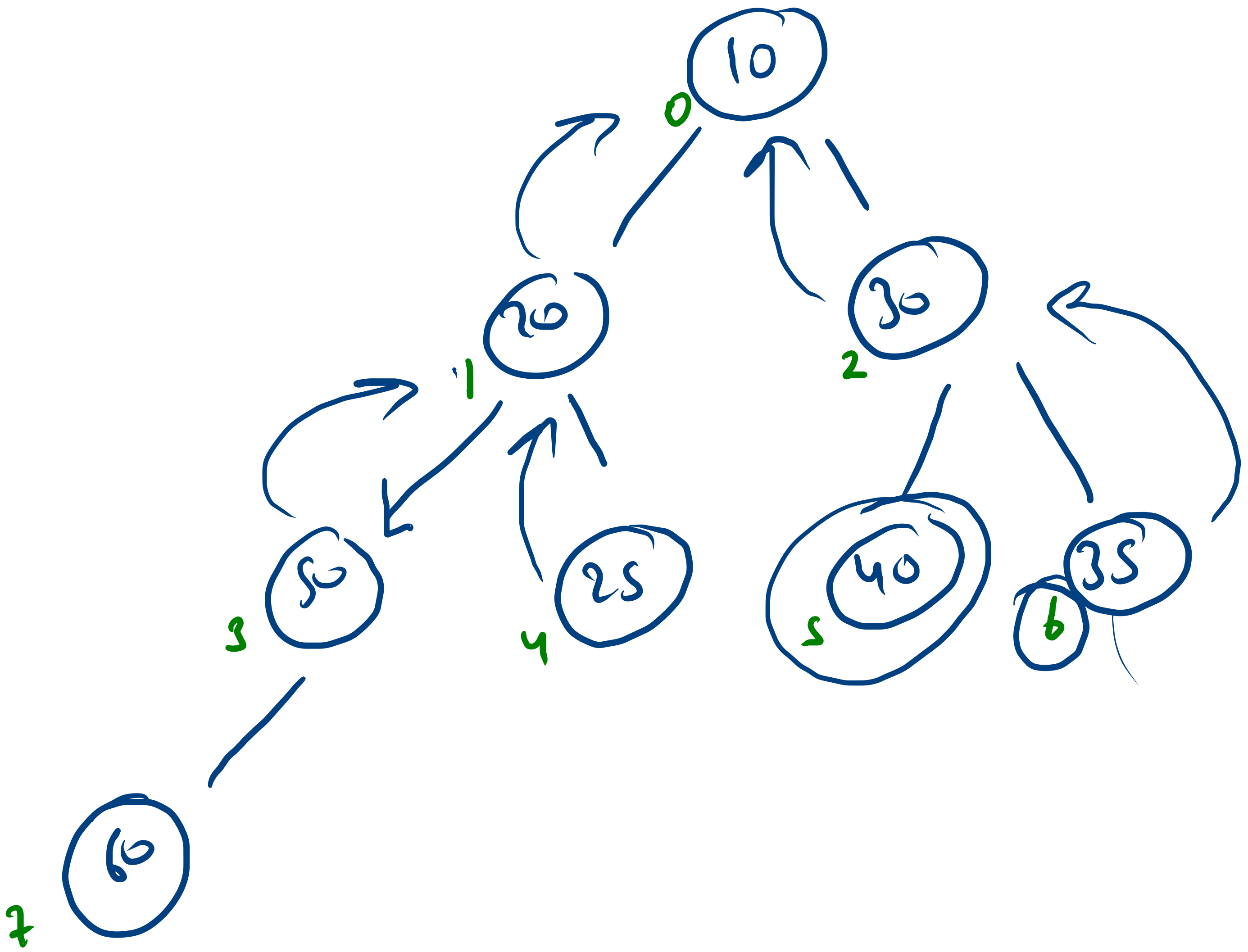
min

$$\begin{array}{lcccccccc} \text{AL} & [\ 10 & 20 & 30 & 50 & 25 & 40 & 35 & 60\ ] \\ & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array}$$

$\frac{2}{2} = 1$

$\frac{3}{2} = 1$

$\frac{0}{2}\ \frac{1}{2} = 0$



left = pi x2 + 1

right = pi x2 + 2

pi = (i-1)/2

$\frac{5-1}{2} = \boxed{2}$

$\frac{6-1}{2} = \boxed{2}$

Add

AL [10 15 30 20 25 40 35 60 50 14]

10	15	30	20	25	40	35	60	50	14
0	1	2	3	4	5	6	7	8	9

add at last

parent compare

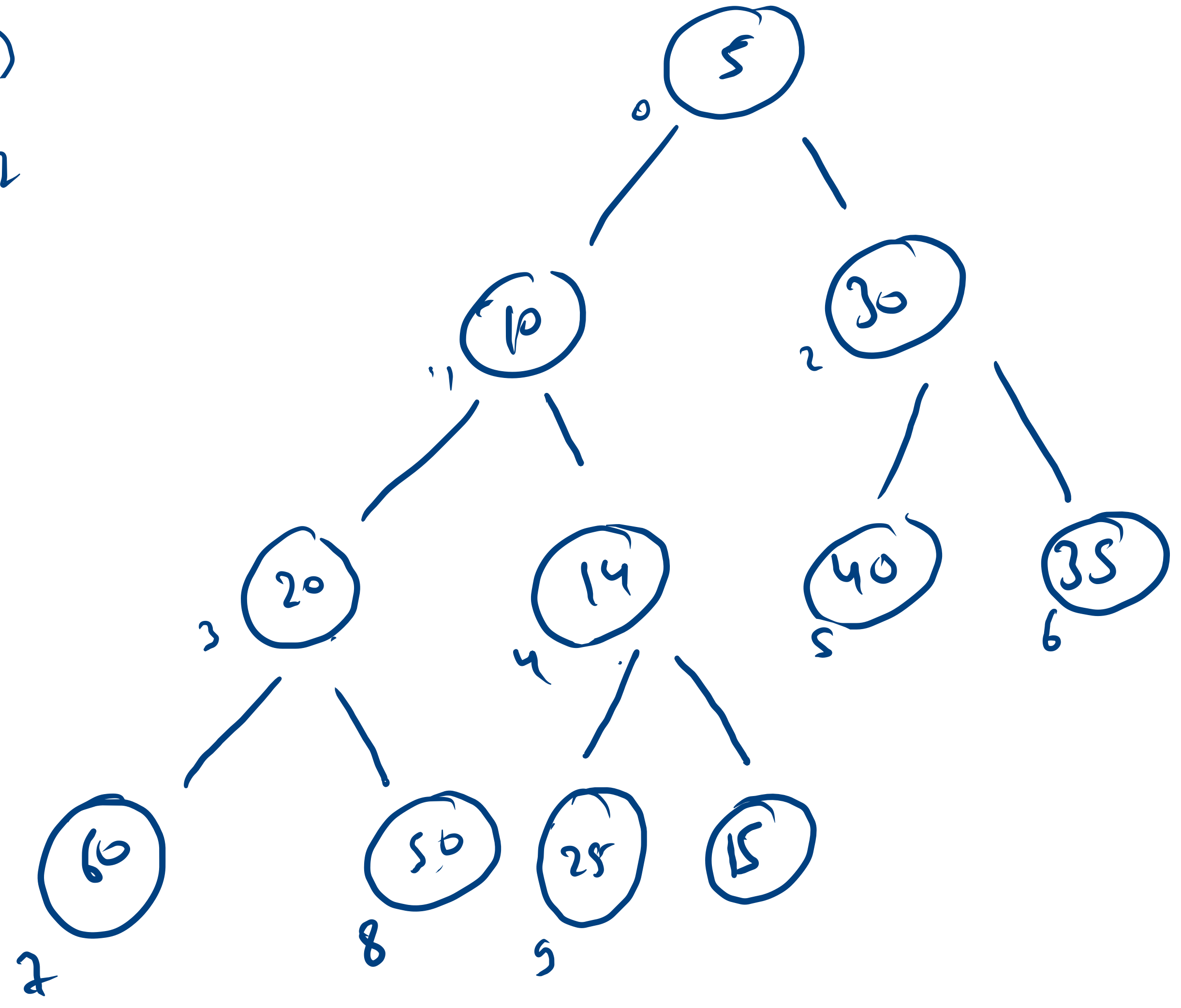
add last

uphipify

CBT

HOP

PiO → min



left = $Pi \times 2 + 1$

right = $Pi \times 2 + 2$

$Pi = (i-1)/2$

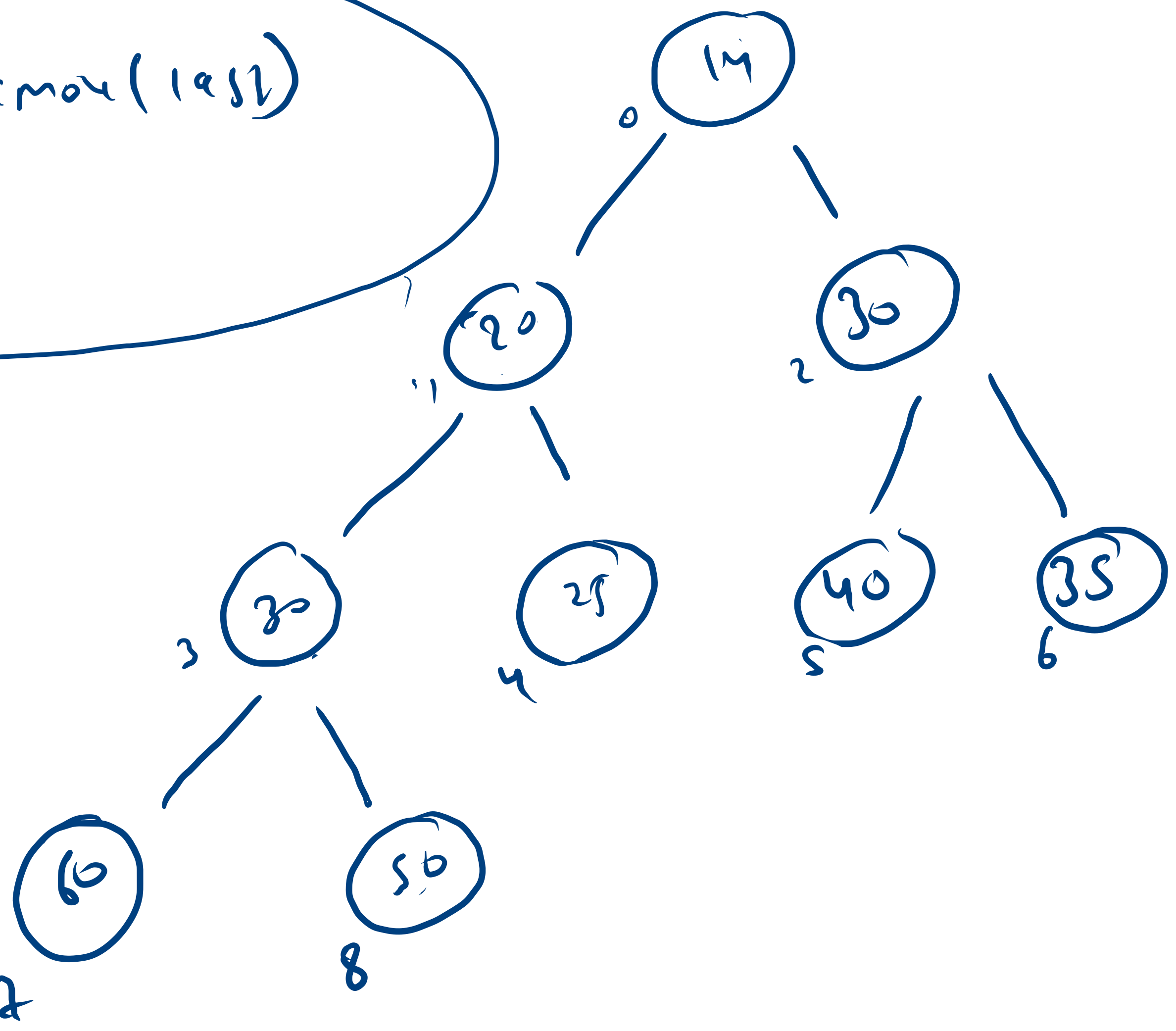
Remove ⑤
peek O(1)

AL [10 15 30 20 25 40 35 60 50 30]
0 1 2 3 4 5 6 7 8 9

val ⇒ get(0)
set(0) = remove(last)
downheapify

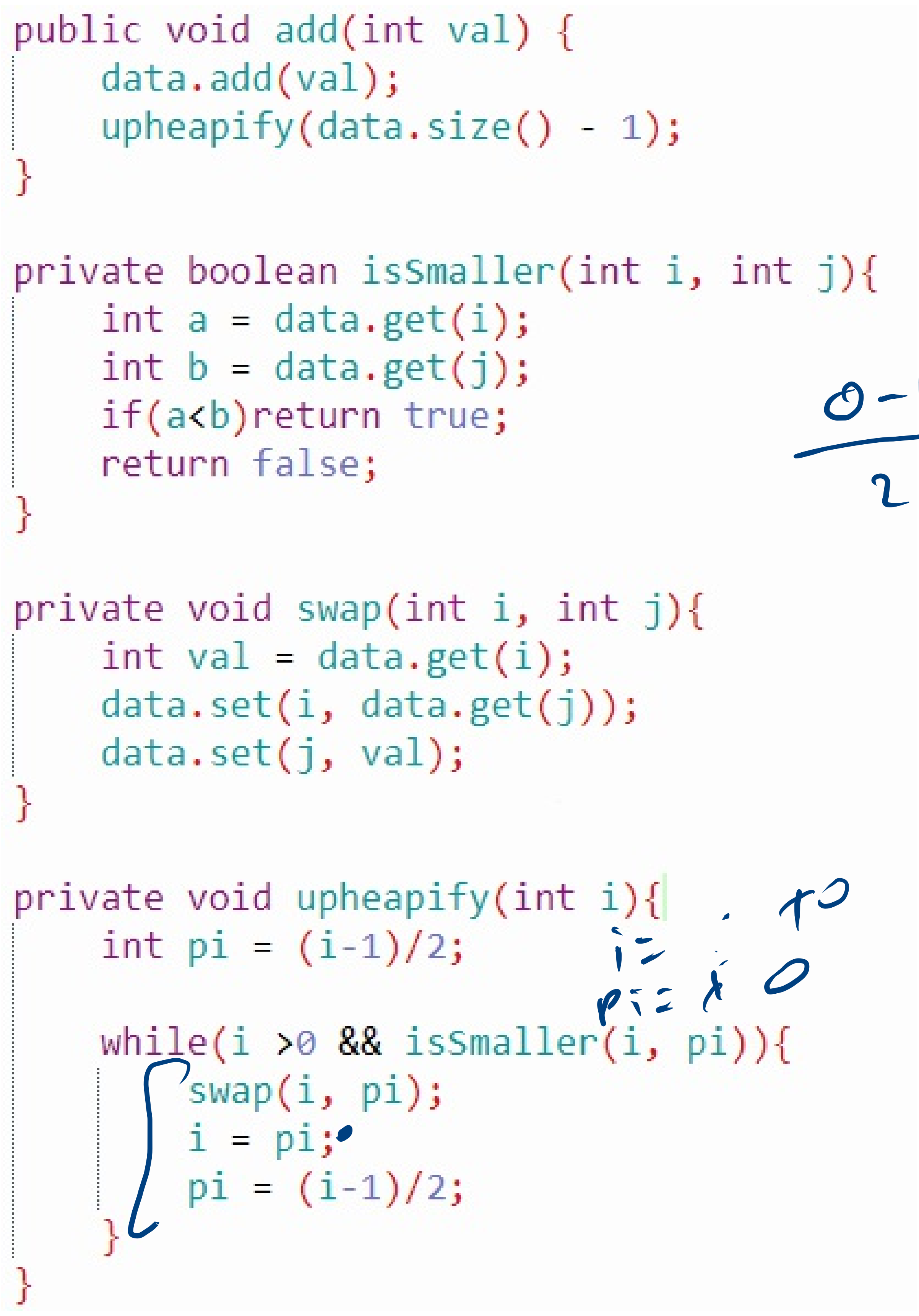
HOP
CBT

min



left = Pi x2 + 1
right = Pi x2 + 2

Pi = (i-1)/2



```
public void add(int val) {
    data.add(val);
    upheapify(data.size() - 1);
}

private boolean isSmaller(int i, int j){
    int a = data.get(i);
    int b = data.get(j);
    if(a<b)return true;
    return false;
}

private void swap(int i, int j){
    int val = data.get(i);
    data.set(i, data.get(j));
    data.set(j, val);
}

private void upheapify(int i){
    int pi = (i-1)/2;

    while(i >0 && isSmaller(i, pi)){
        swap(i, pi);
        i = pi;
        pi = (i-1)/2;
    }
}
```

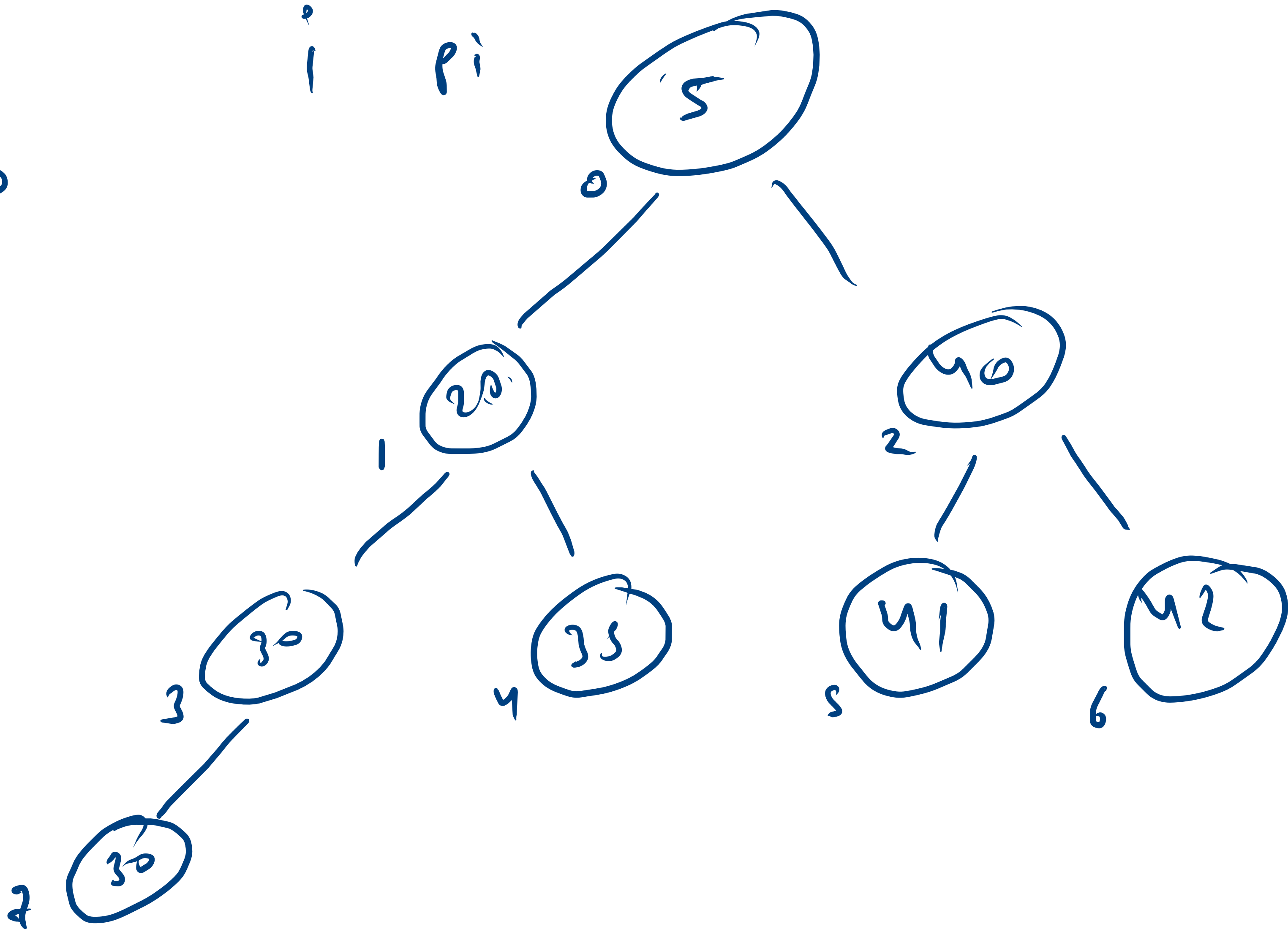
$\boxed{5}$

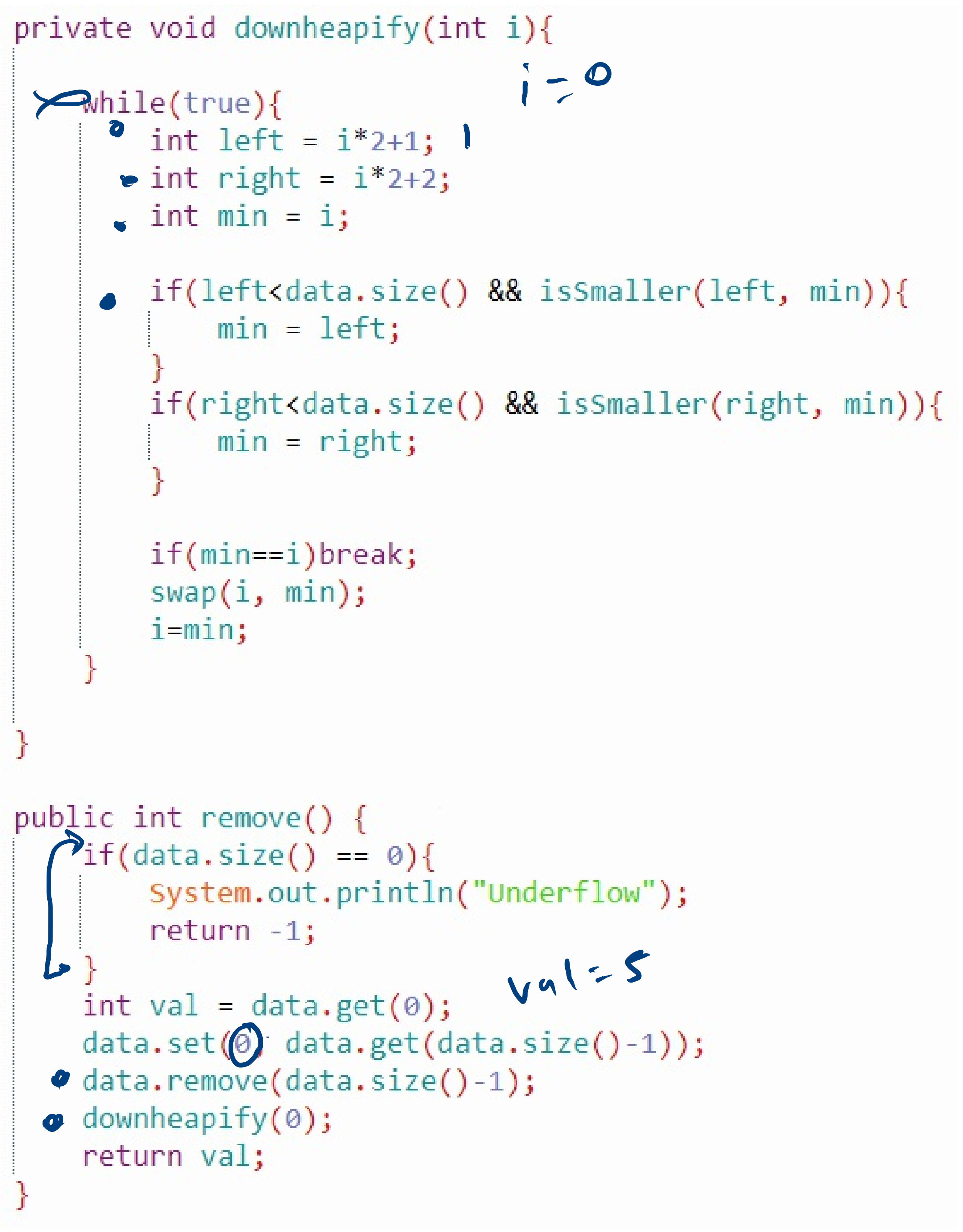
$\frac{0-1}{2}$ -0

i = ~~1~~ 0

pi = ~~1~~ 0

i pi





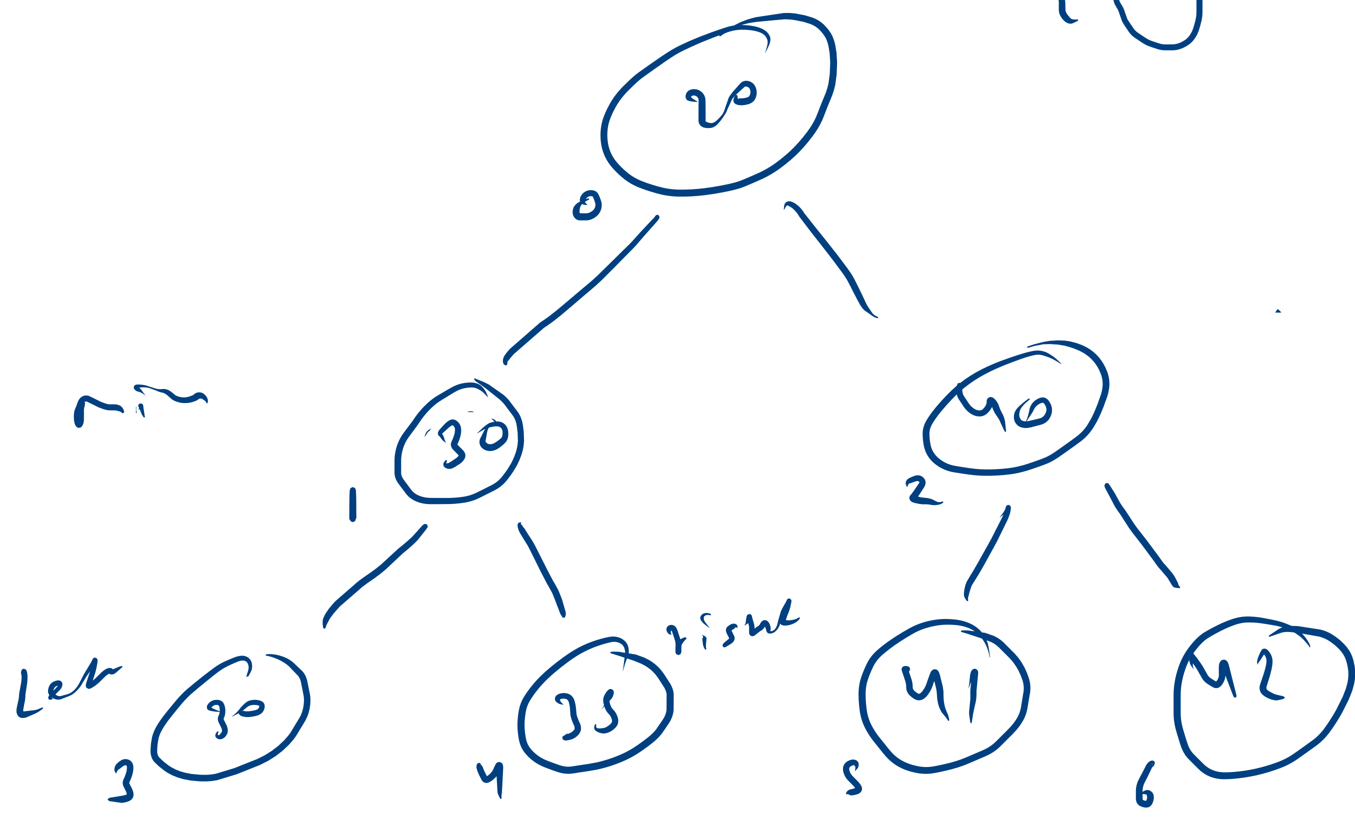
```java
private void downheapify(int i){

    while(true){
        int left = i*2+1;
        int right = i*2+2;
        int min = i;

        if(left<data.size() && isSmaller(left, min)){
            min = left;
        }
        if(right<data.size() && isSmaller(right, min)){
            min = right;
        }

        if(min==i)break;
        swap(i, min);
        i=min;
    }

}

public int remove() {
    if(data.size() == 0){
        System.out.println("Underflow");
        return -1;
    }
    int val = data.get(0);
    data.set(0, data.get(data.size()-1));
    data.remove(data.size()-1);
    downheapify(0);
    return val;
}
```

i=0

val=5

i

min



20
0
30
1
40
2
Left
30
3
35
4
right
41
5
42
6