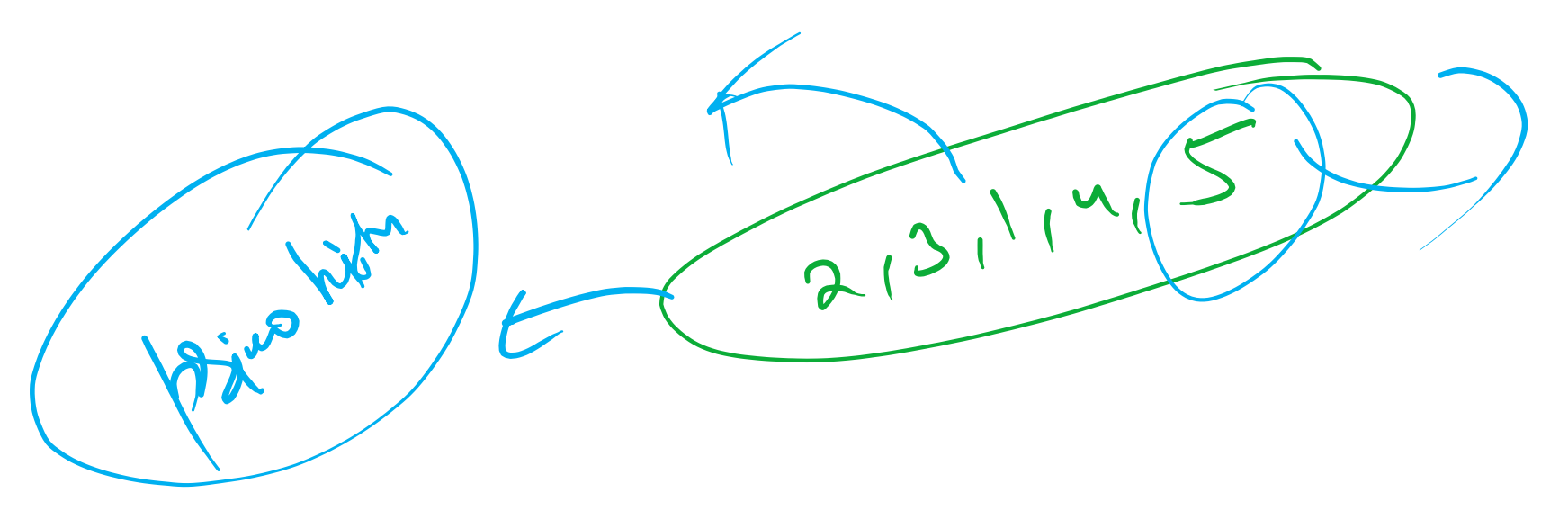
Lec-35

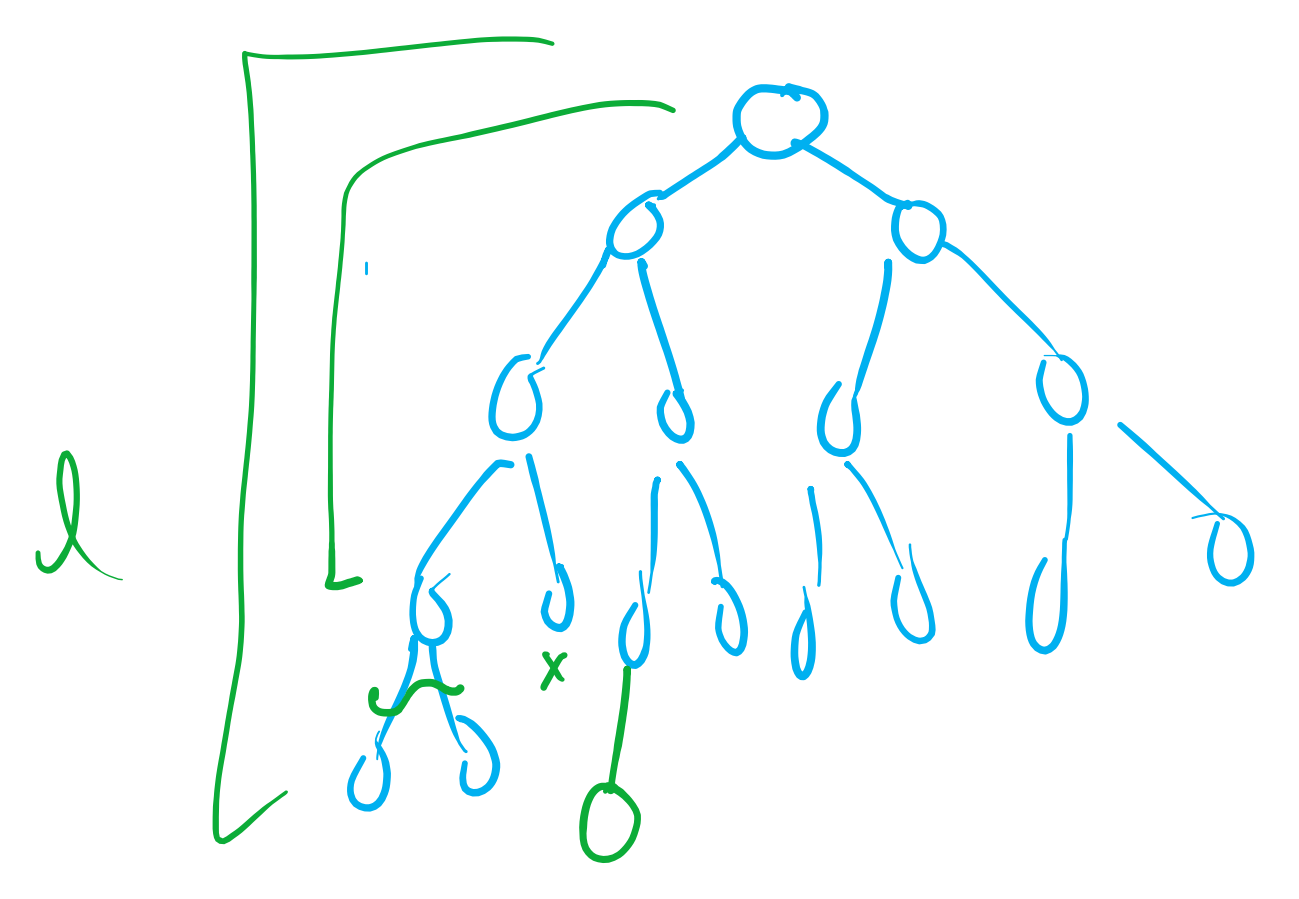
04/05/24 10:19 AM

Heap | Priority queue

Priority queue → Queue FIFO



(i) Heap gives complete binary tree and priority after



~~Comp~~

CBT

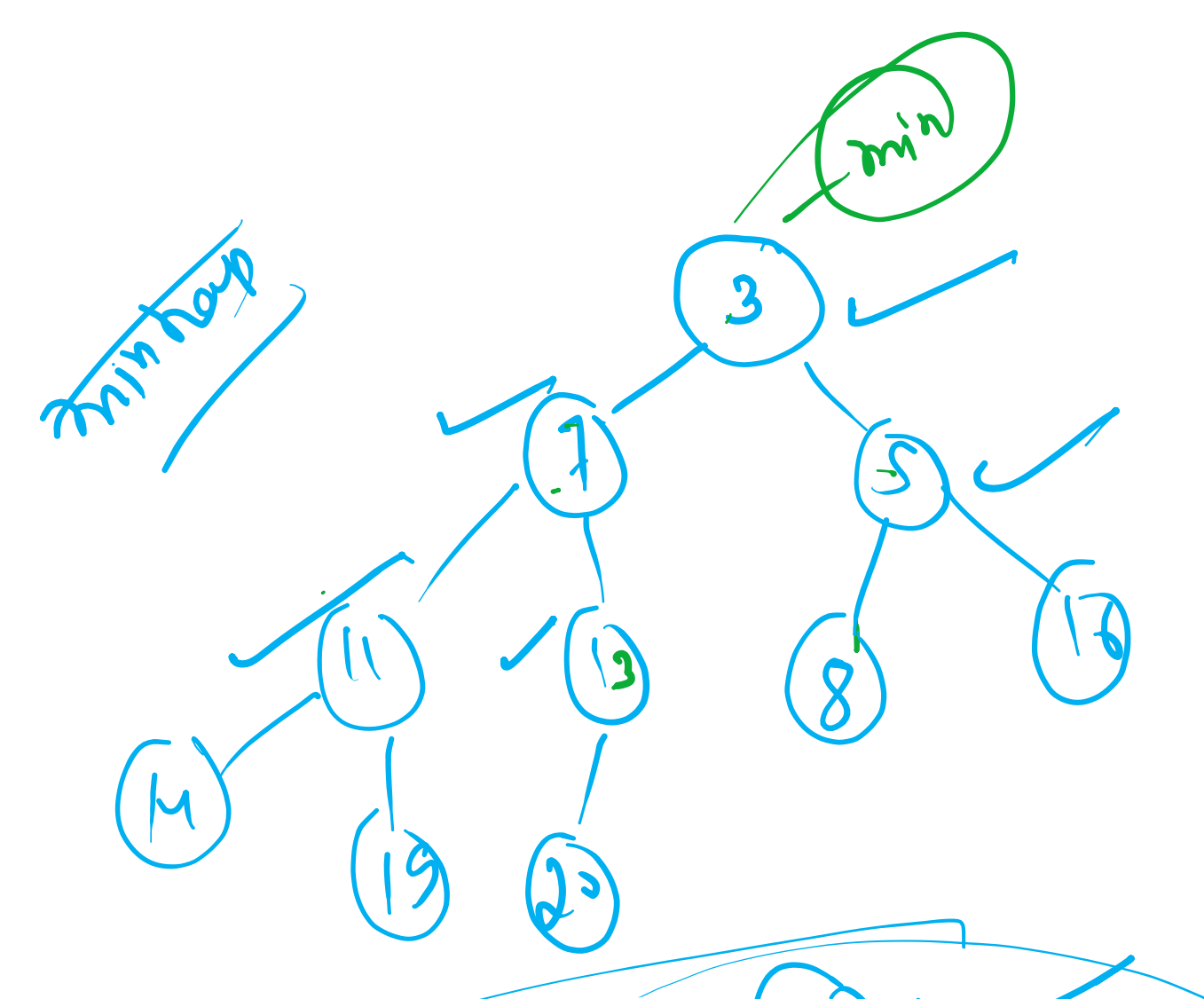
Priority

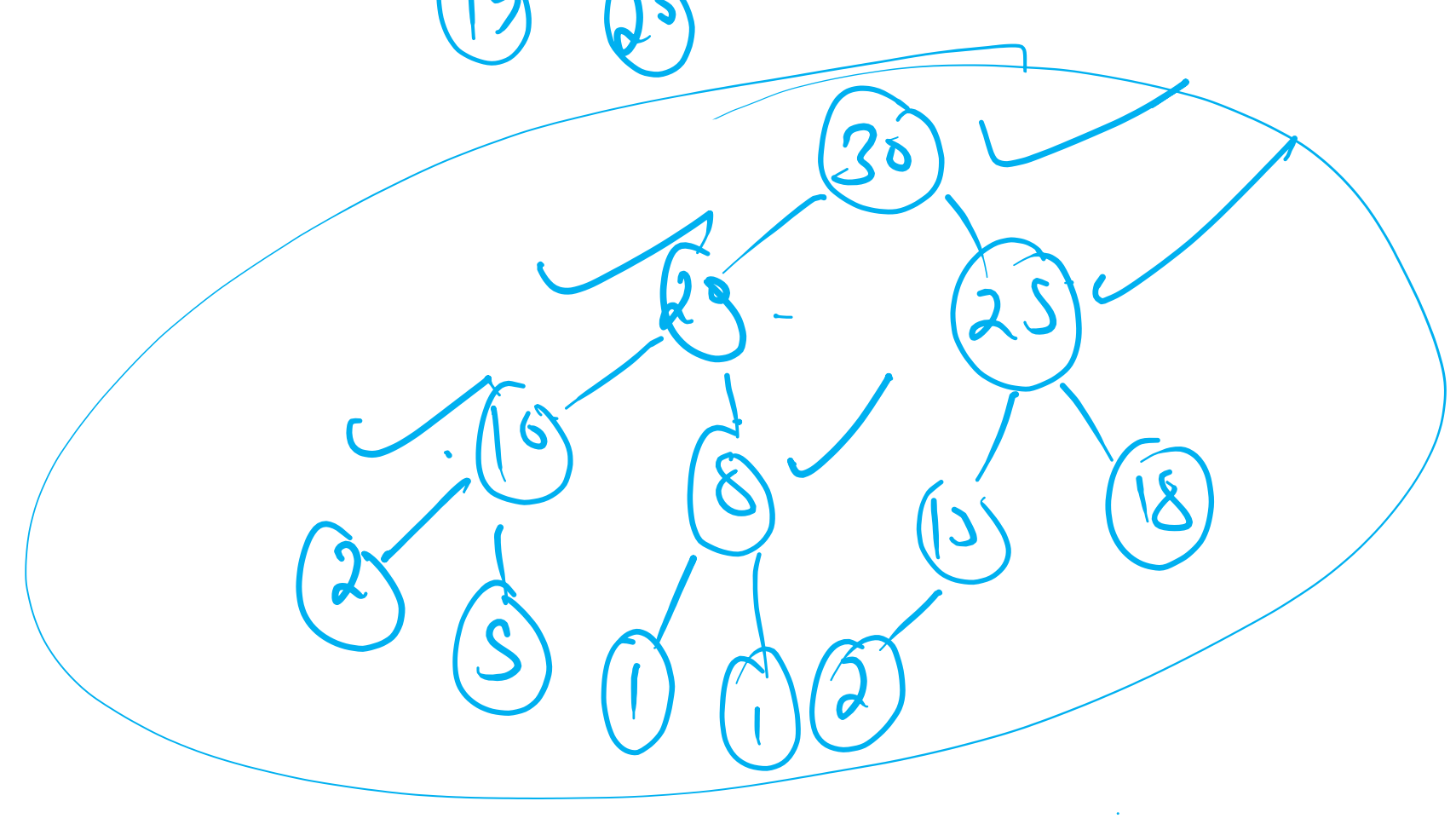
Rank

marks

max heap

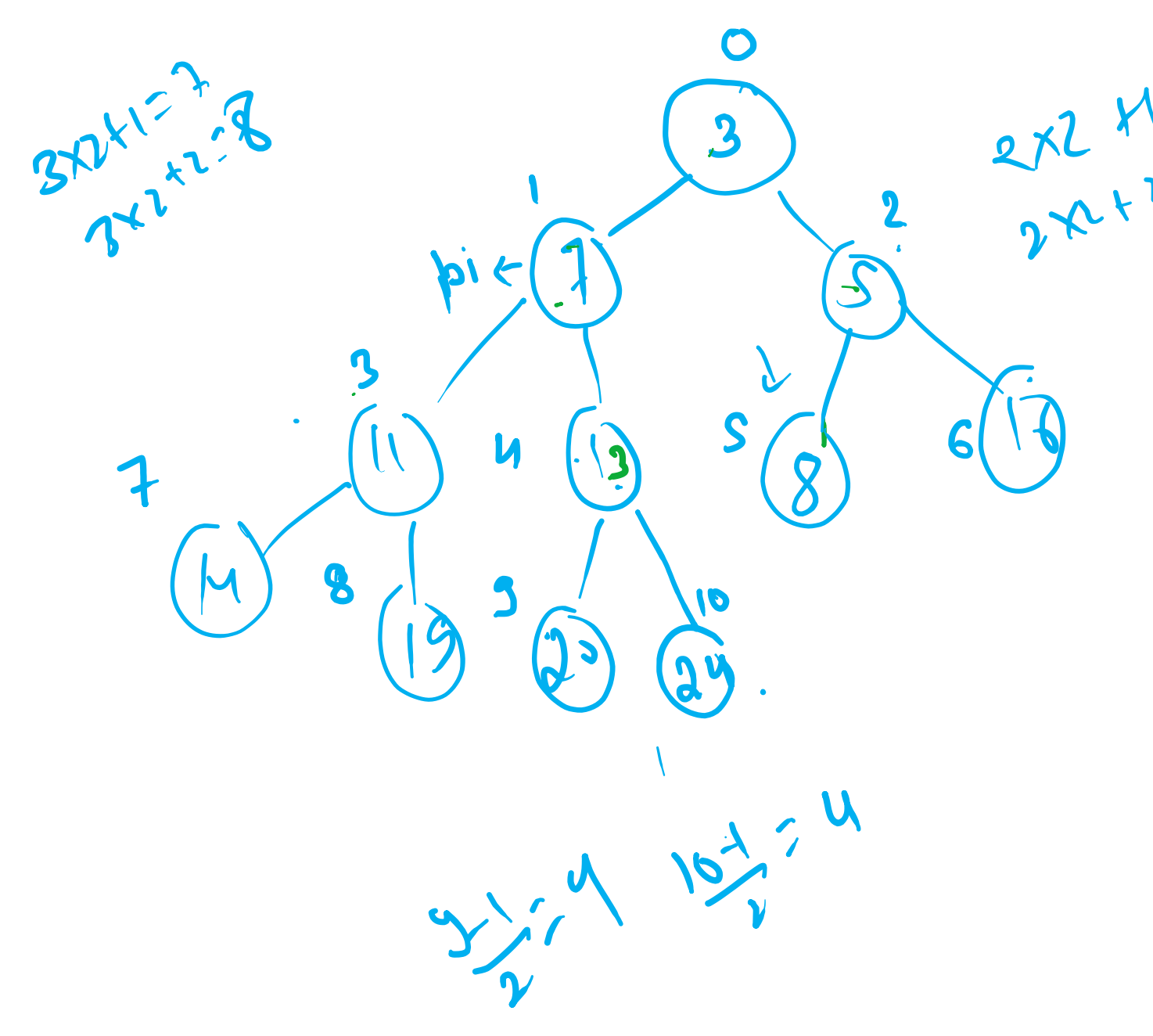
min heap





Why Heap?

	Sorted	unsorted	Heap
Add	O(n)	O(1)	log(n)
min remove	O(n)	O(n)	log(n)
get min	O(1)	O(n)	O(1)

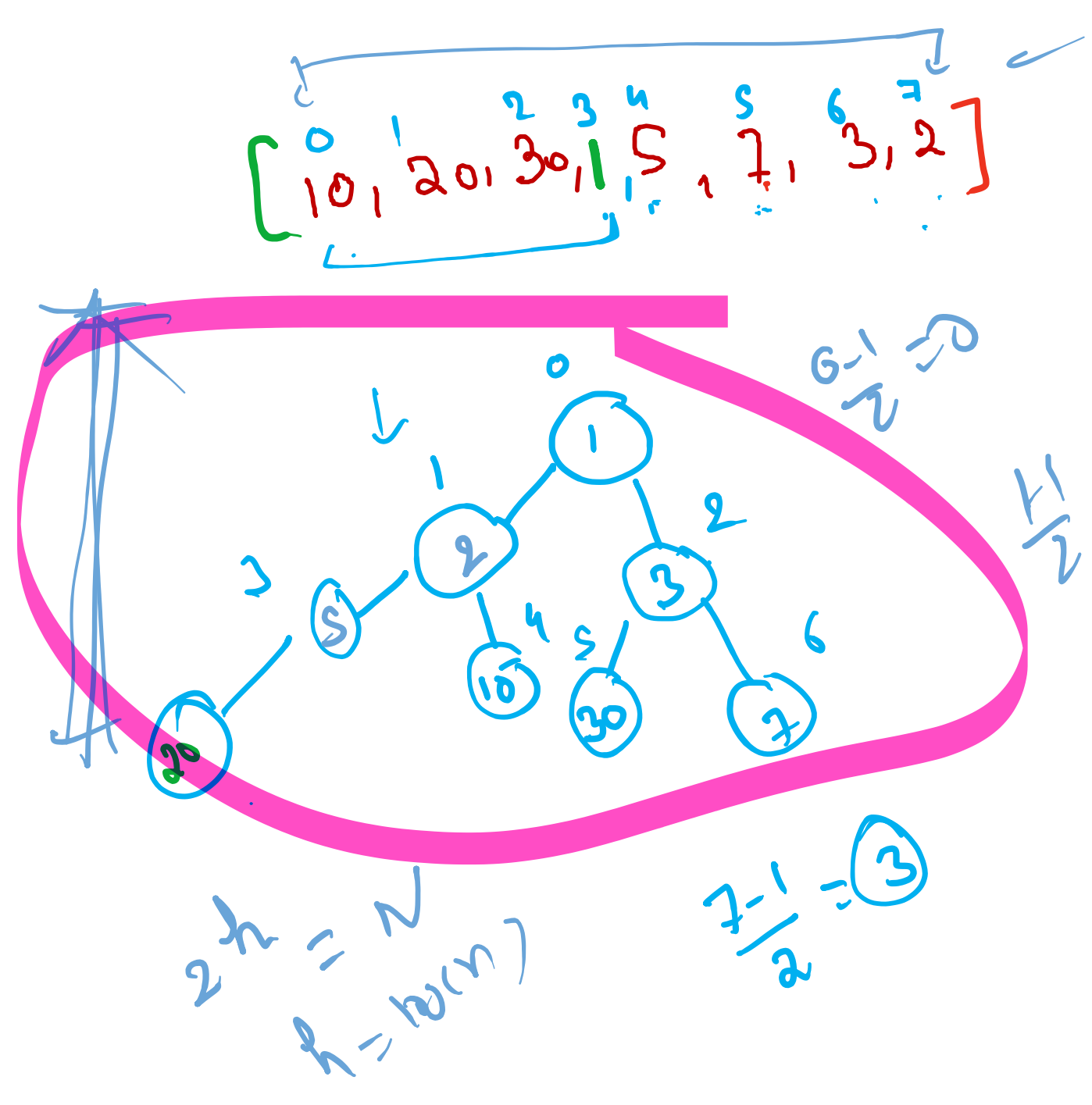


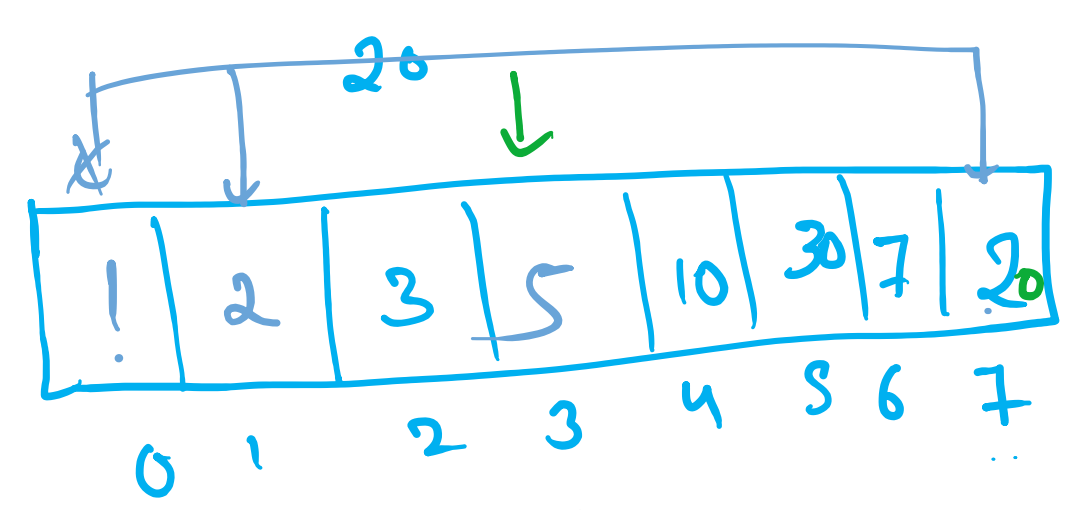
LCi = 2×pi+1 = 1×2+1=3

RCi = 2×pi+2 = 1×2+2=4

Ci=9

$pi = \left[\frac{Ci-1}{2}\right]$

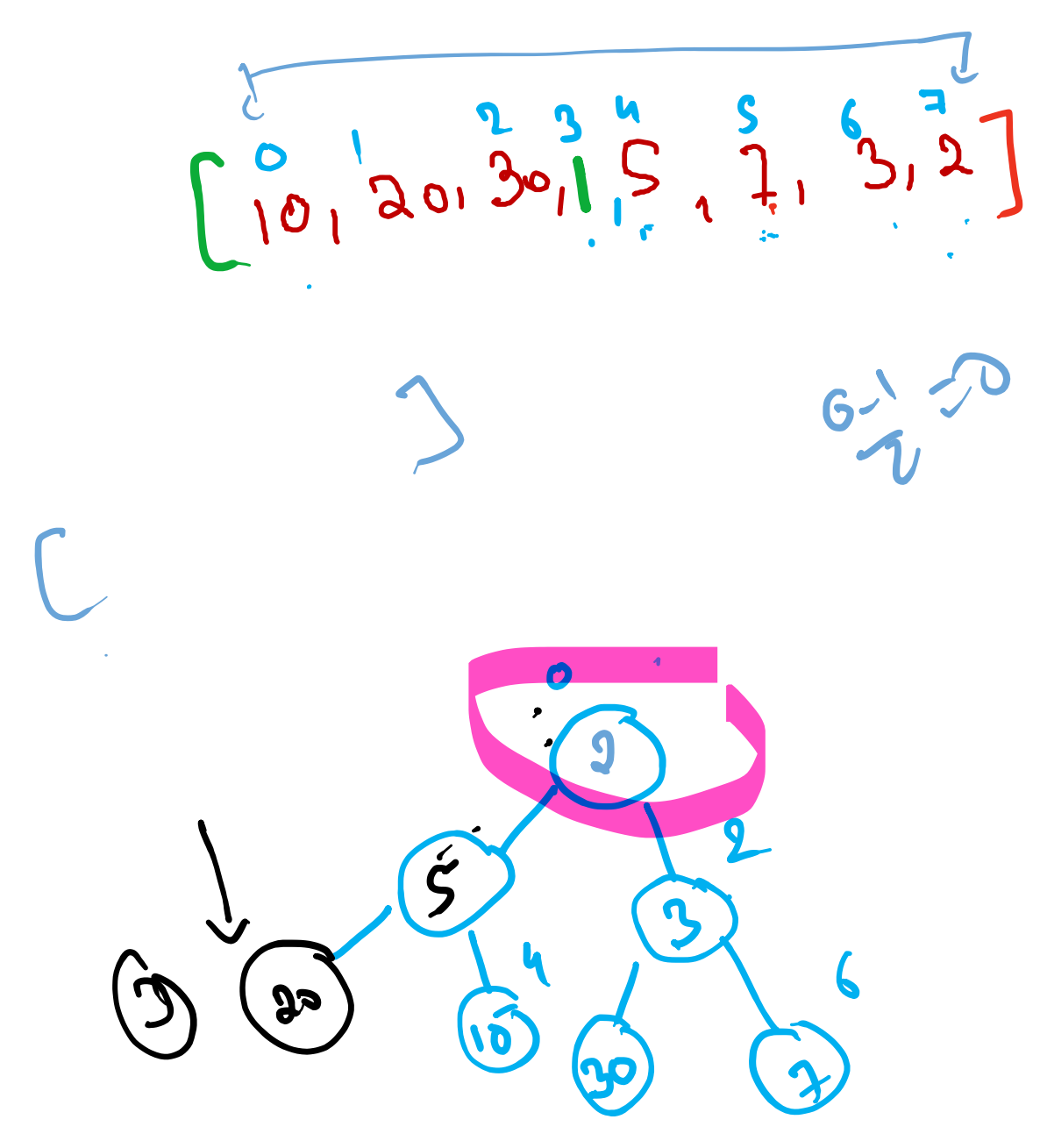


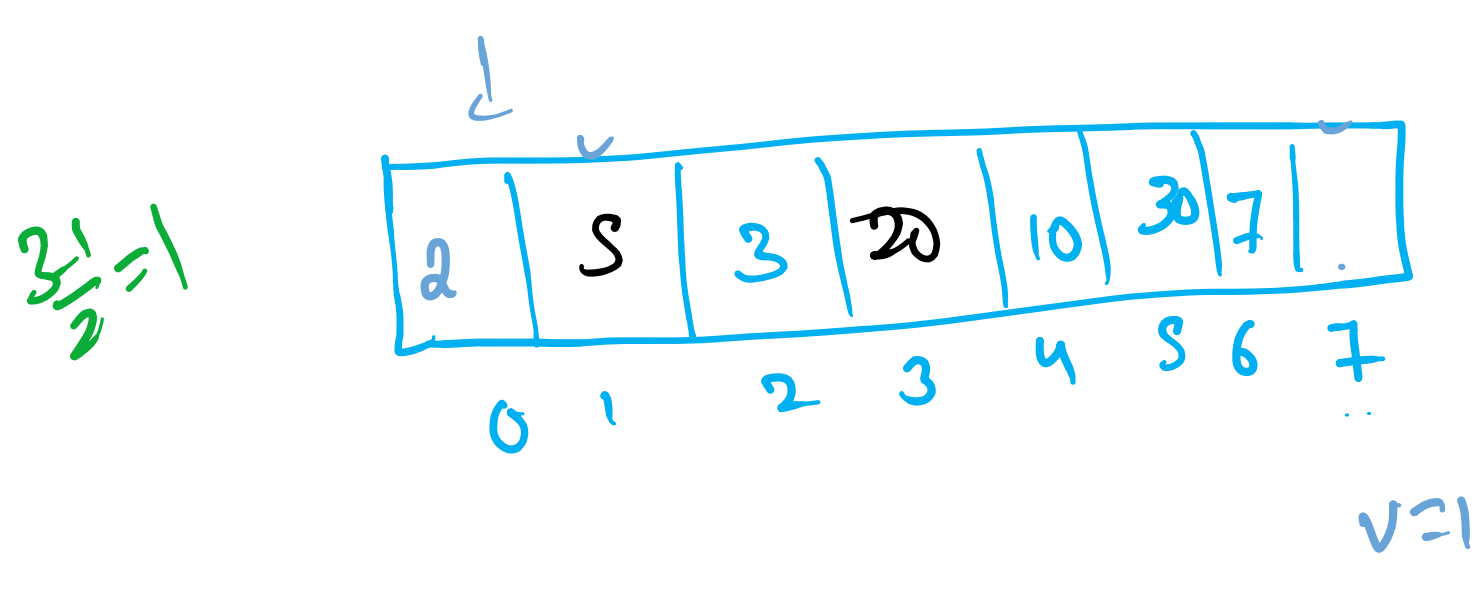


```
public void add(int item) {
        list.add(item);
        upheapify(list.siz e() - 1);

    }

    private void upheapify(int ci) {
        // TODO Auto-generated method stub
        int pi = (ci - 1) / 2;
        if (list.get(pi) > list.get(ci)) {
            swap(ci, pi);
            upheapify(pi);
        }

    }
```

7

→ 7 7-1/2 ⇒ 3





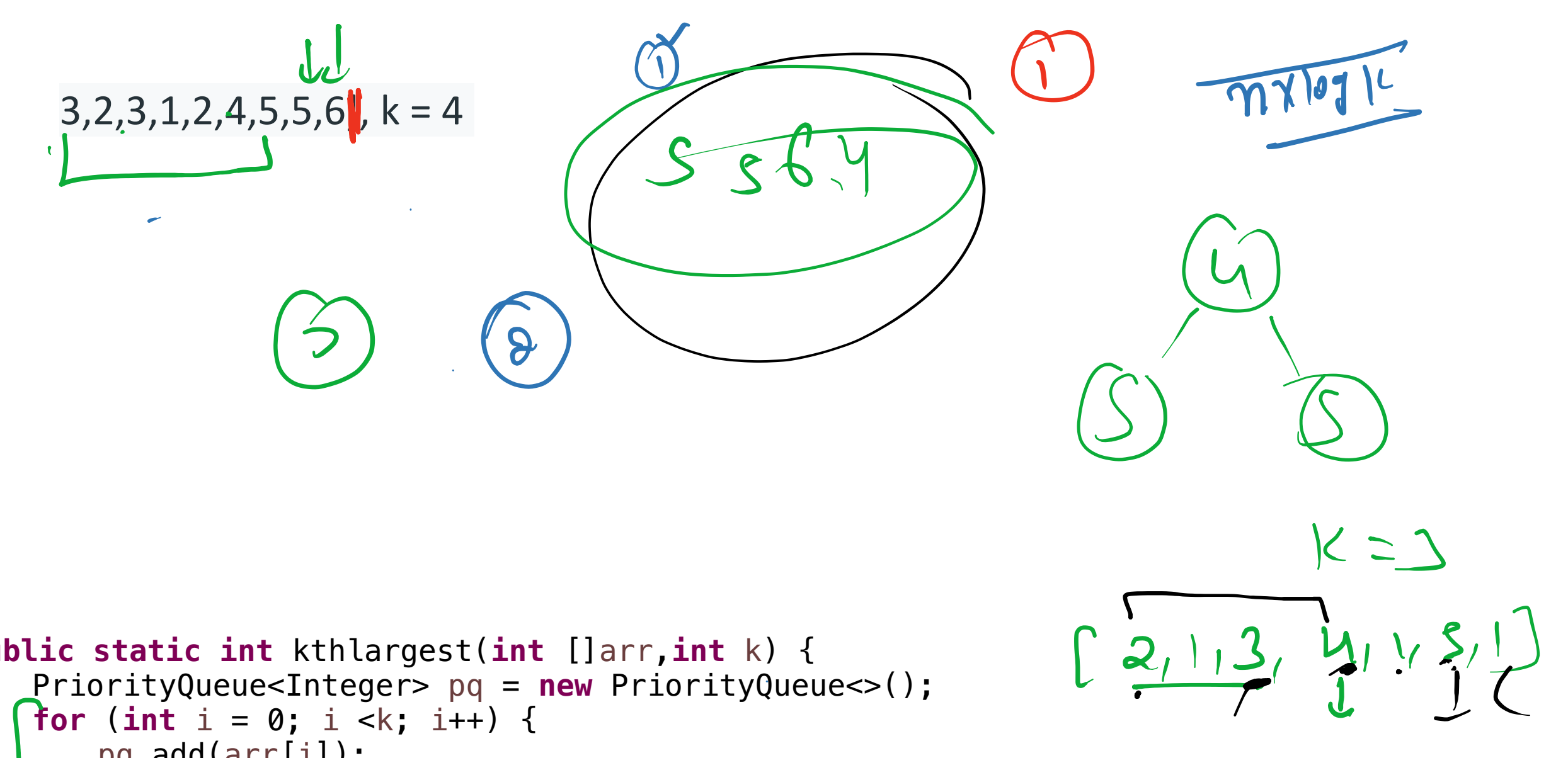
```
private void dowmheapify(int pi) {
        // TODO Auto-generated method stub
        int lci = 2 * pi + 1;
        int rci = 2 * pi + 2;
        int mini =pi;
        if(list.get(lci)<list.get(mini)) {
            mini=lci;
        }
        if(list.get(rci)<list.get(mini)) {
            mini=rci;
        }
        if(mini!=pi) {
            swap(mini, pi);
            dowmheapify(mini);
        }

    }
```

Given an integer array `nums` and an integer `k`, return *the* `k`th *largest element in the array*.

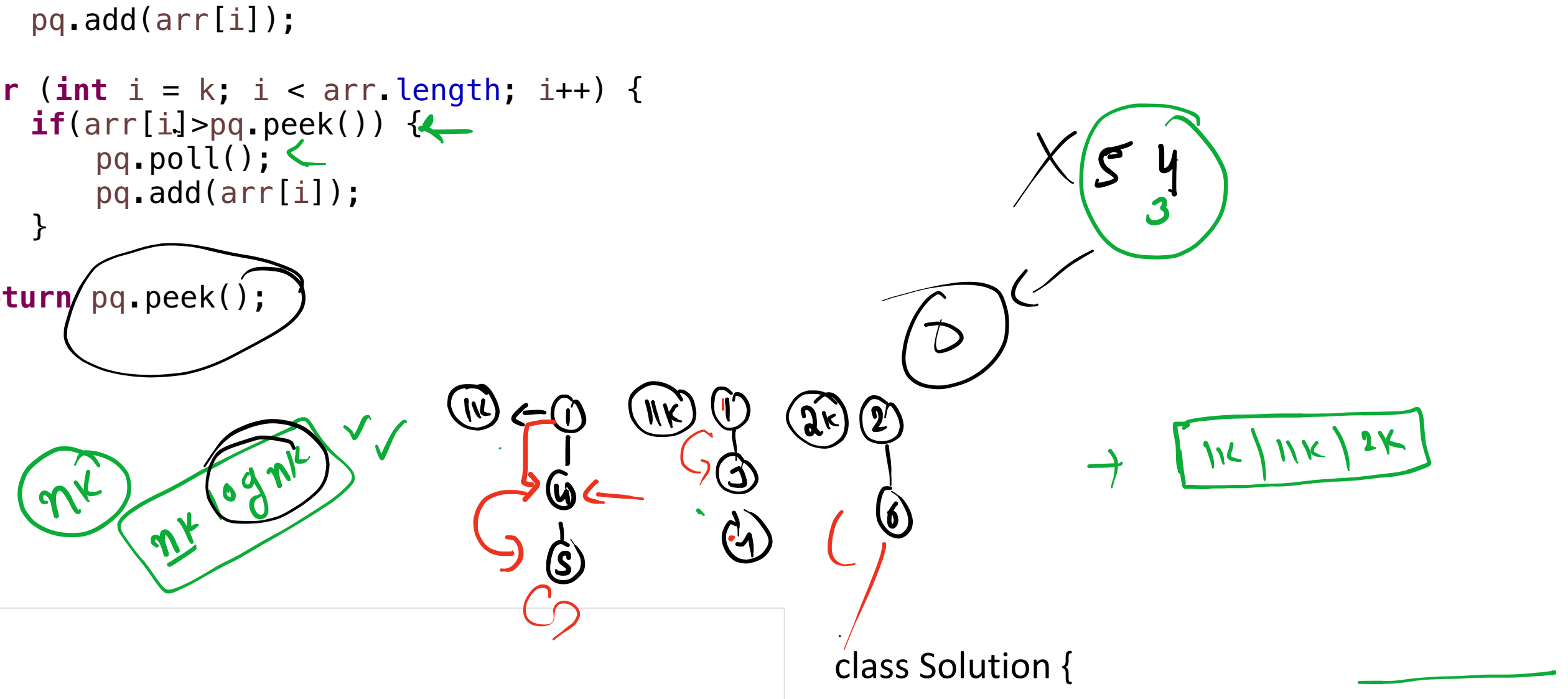
Note that it is the `k`th largest element in the sorted order, not the `k`th distinct element.

Can you solve it without sorting?

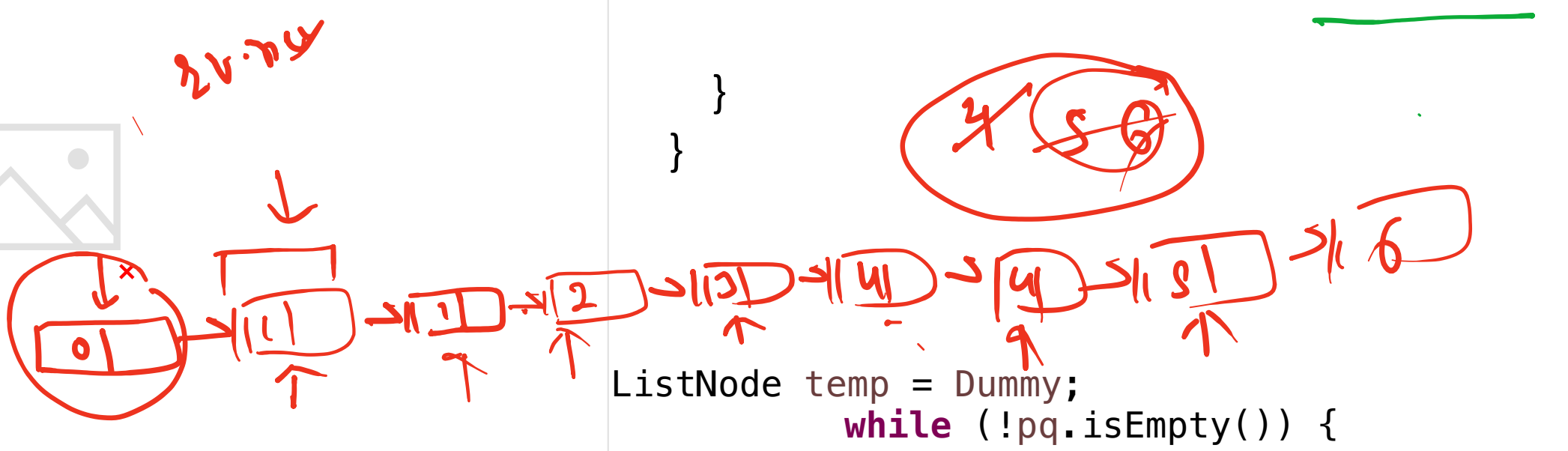
3,2,3,1,2,4,5,5,6, k = 4



```
public static int kthlargest(int []arr,int k) {
    PriorityQueue<Integer> pq = new PriorityQueue<>();
    for (int i = 0; i <k; i++) {
        pq.add(arr[i]);
    }
    for (int i = k; i < arr.length; i++) {
        if(arr[i]>pq.peek()) {
            pq.poll();
            pq.add(arr[i]);
        }
    }
    return pq.peek();
}
```



```
class Solution {
    public ListNode mergeKLists(ListNode[] lists) {

    }
}
```



```
ListNode temp = Dummy;
        while (!pq.isEmpty()) {
            ListNode rv = pq.remove();// poll
            Dummy.next = rv;
            Dummy = Dummy.next;
            if (rv.next != null) {
                pq.add(rv.next);
            }

        }
```