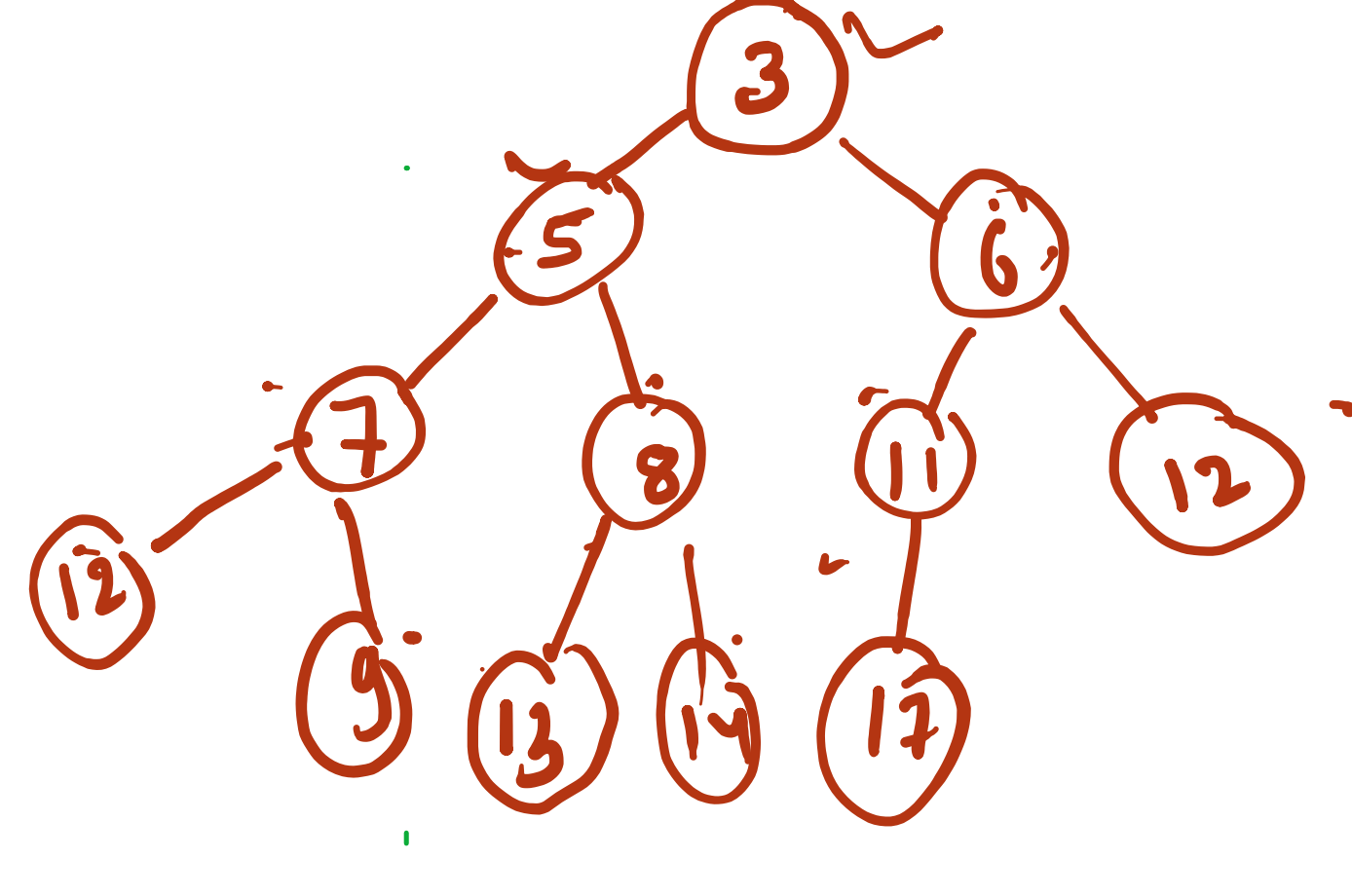
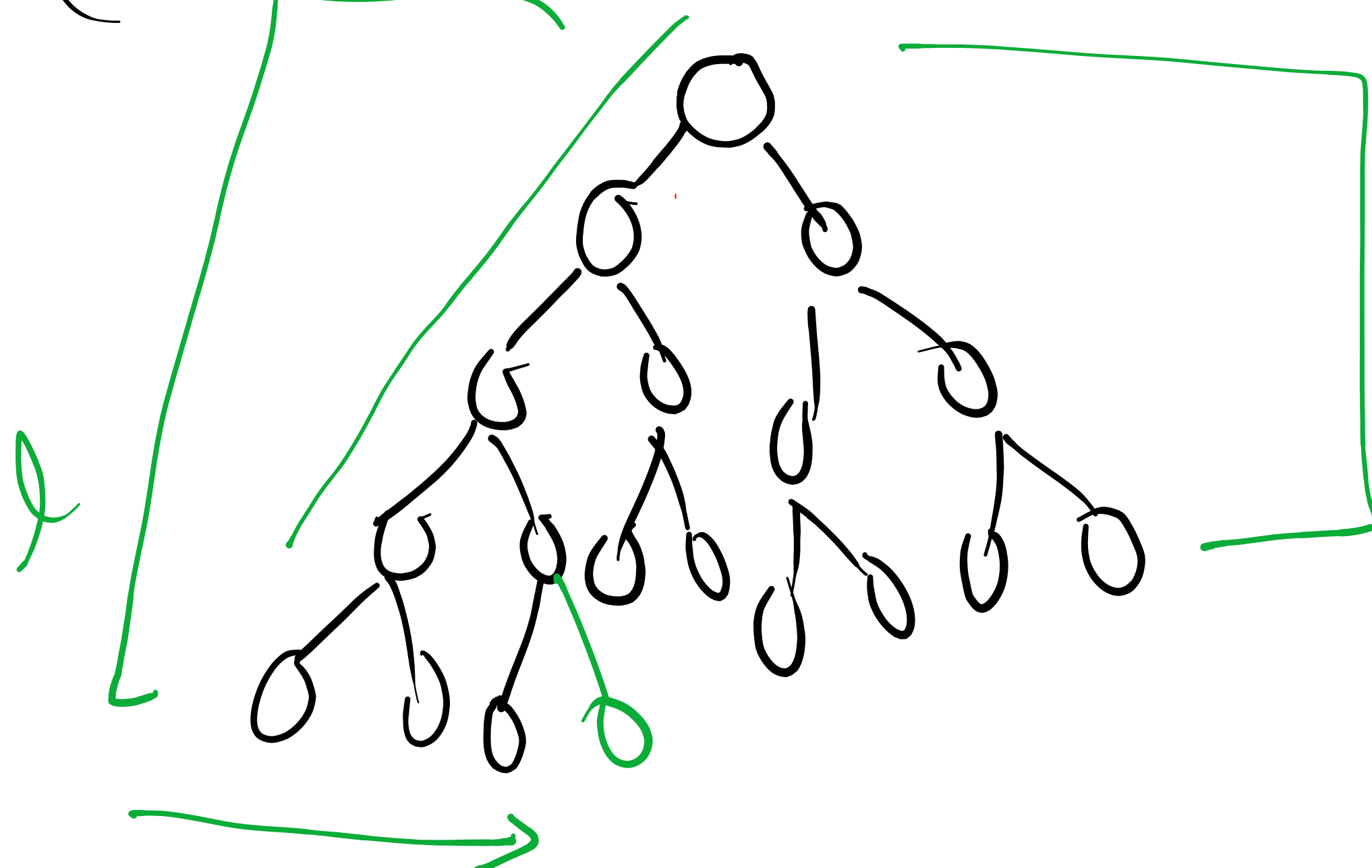


Heap

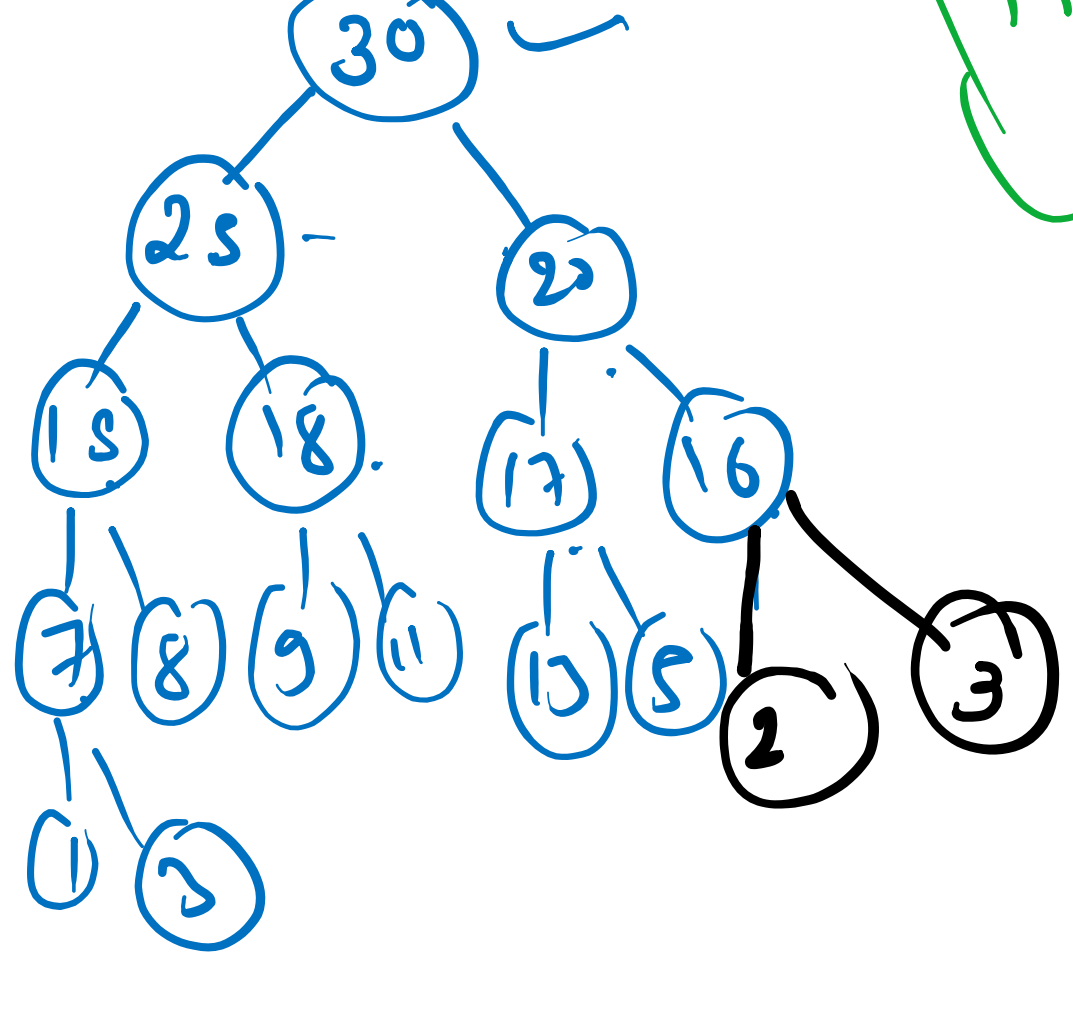
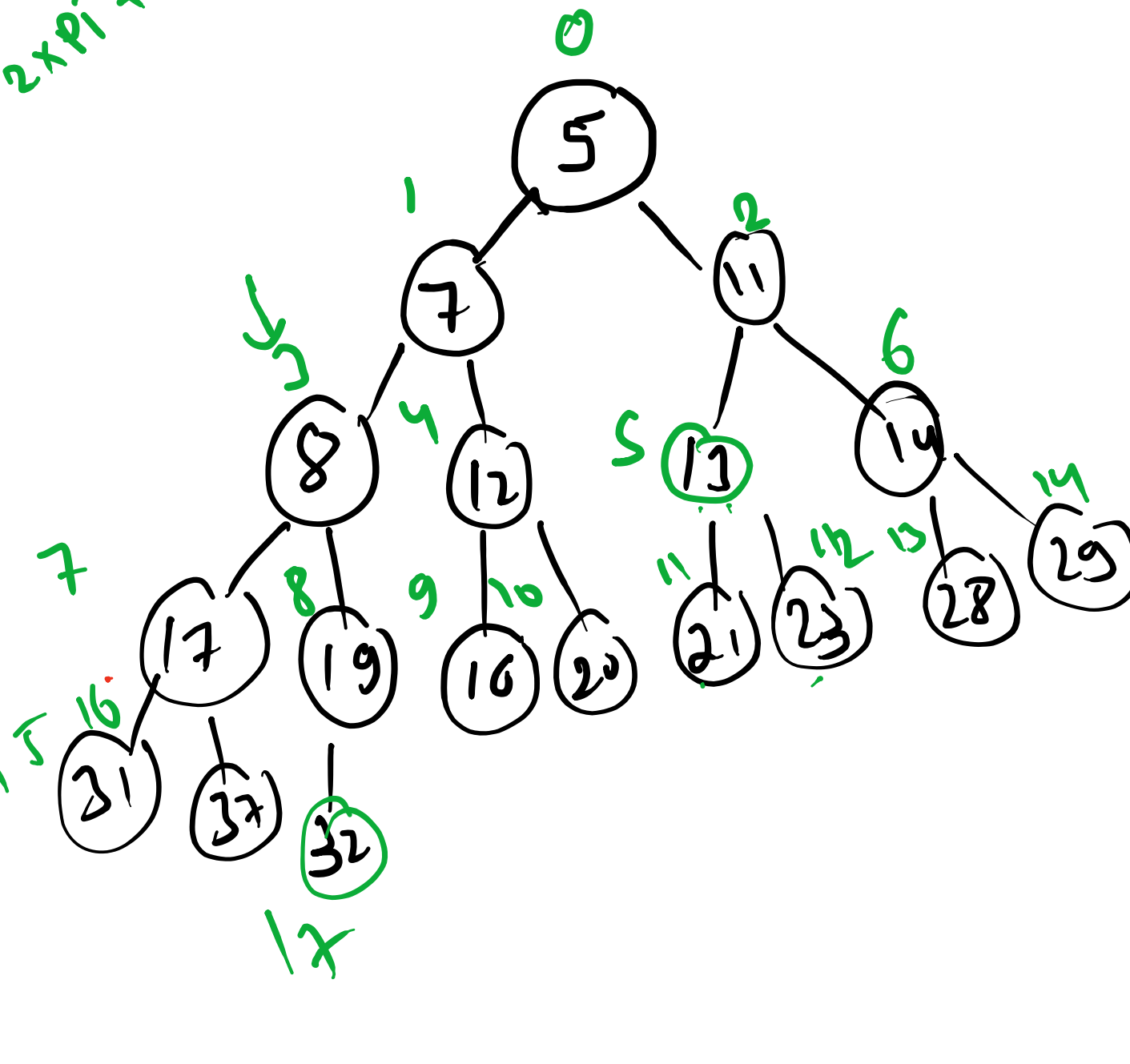
It is CBT & Binary priority



Ranking min heap  
max heap

$\frac{1}{2}$  ✓

$LCI = 2 \times pi + 1$   
 $RCI = 2 \times pi + 2$



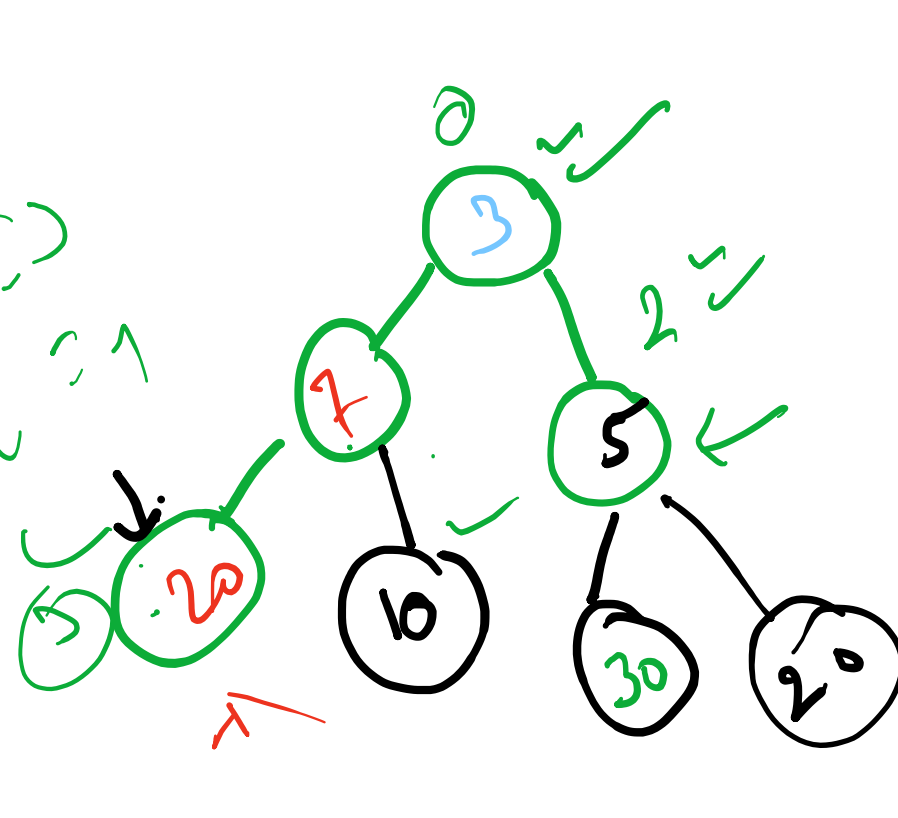
$pi = \frac{ci-1}{2}$

Sortel	unsorted	Heap
$O(n)$	$O(n)$	$\log(n)$
$O(n)$	$O(n)$	$\log(n)$
$O(1)$	$O(n)$	$O(1)$

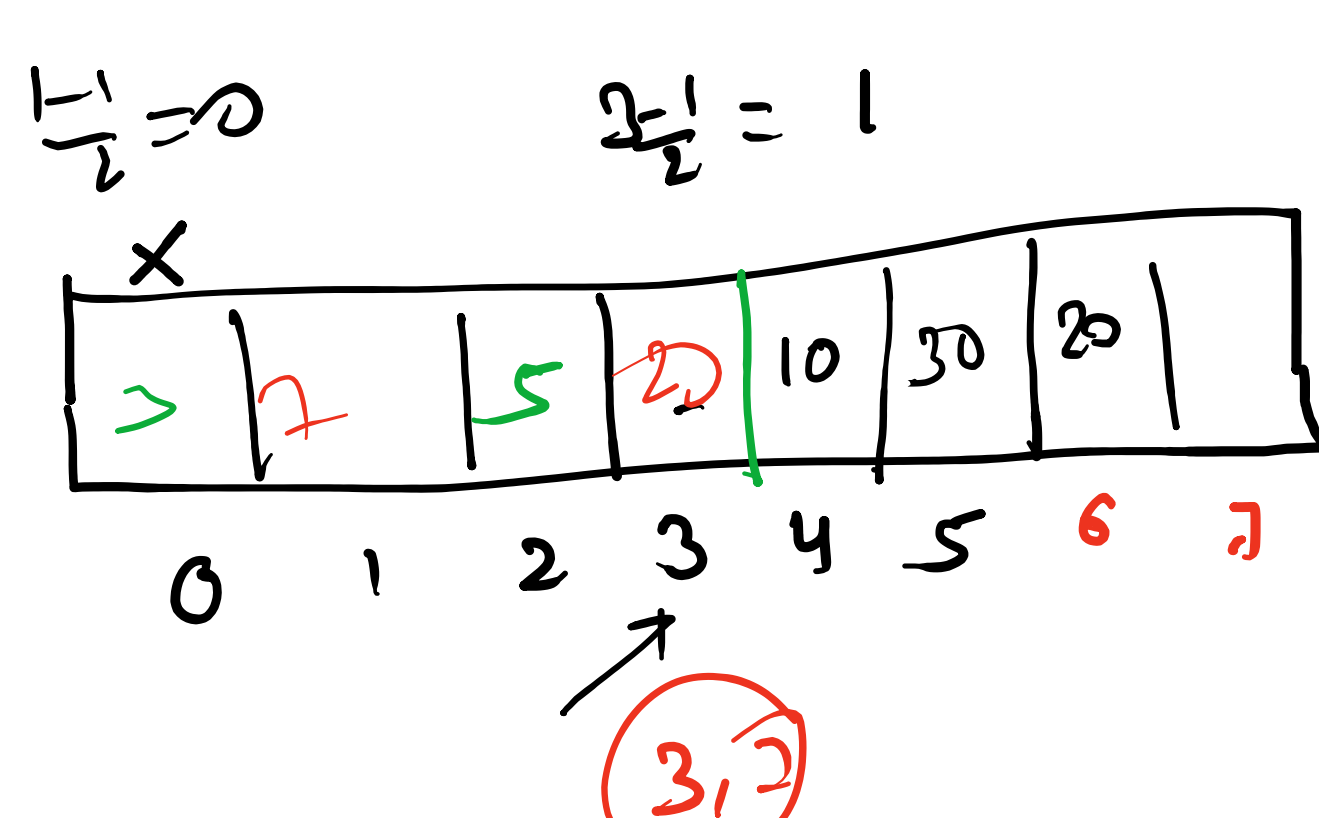
add  
min remove  
del min

$\{10, 20, 30, 5, 7, 3, 20, 2\}$

$3 \times 2 + 1 = 7$   
 $3 \times 2 + 2 = 8$   
 $1 \times 2 + 1 = 3$   
 $1 \times 2 + 2 = 4$



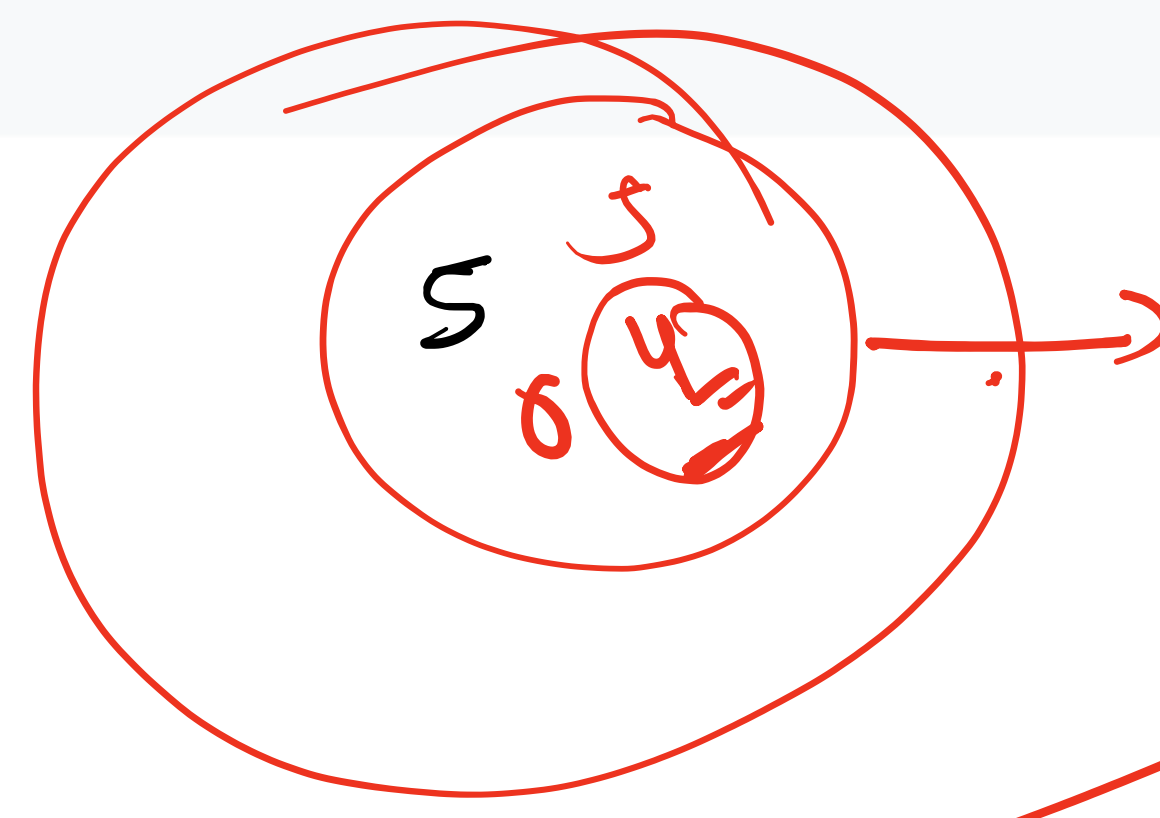
$h = n$   
 $h = \log(n)$



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

```
private void upheapify(int ci) {  
    int pi = (ci - 1) / 2;  
    if (ll.get(pi) > ll.get(ci)) {  
        swap(pi, ci);  
        upheapify(pi);  
    }  
}
```

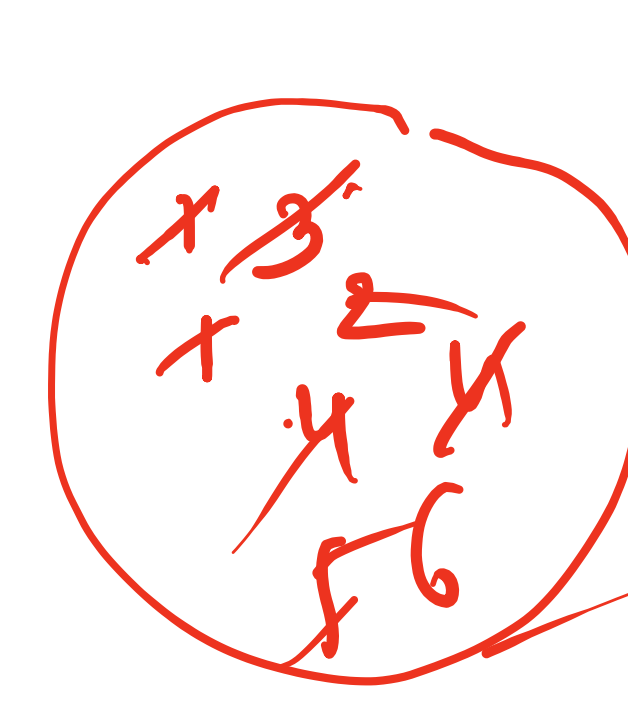
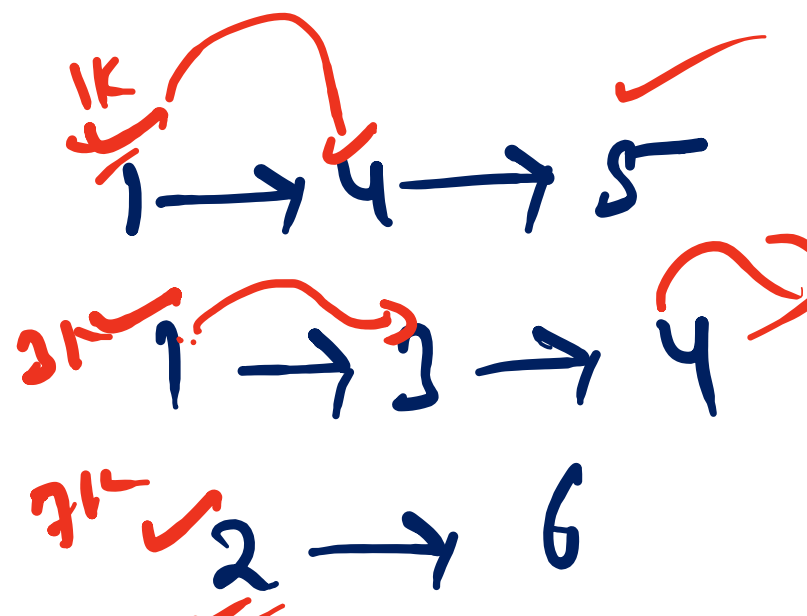
Input: nums = [3,2,3,1,2,4,5,5,6,2] k = 4  
Output: 4



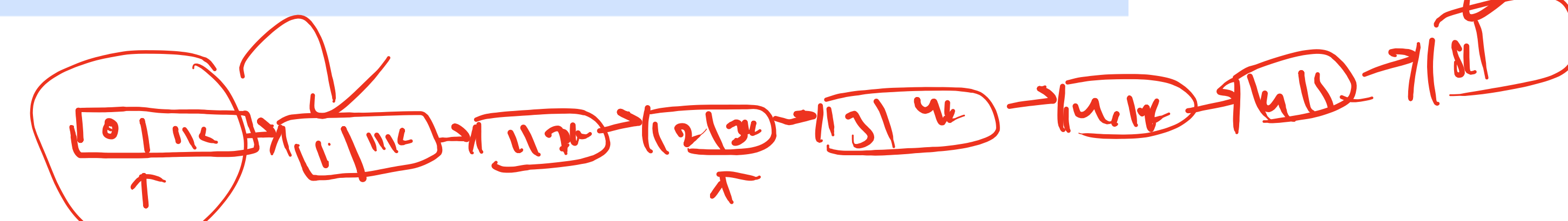
$n \rightarrow$   
 $O(n)$   
 $O(k \times n)$

$k \times n \times \log(k \times n)$   
 $k \times n \times \log(k)$

```
Input: lists = [[1,4,5],[1,3,4],[2,6]]  
Output: [1,1,2,3,4,4,5,6]  
Explanation: The linked-lists are:  
[  
  1->4->5,  
  1->3->4,  
  2->6  
]
```

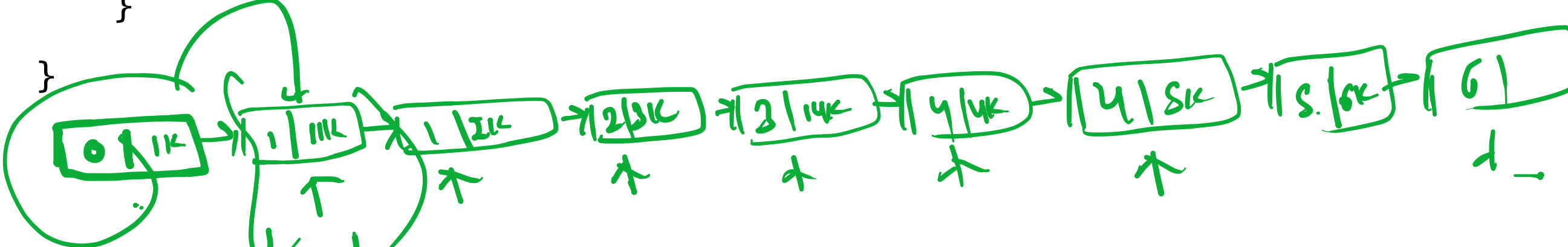
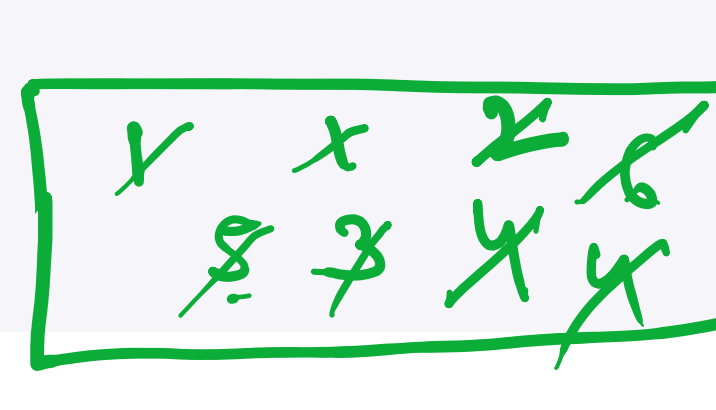


```
public ListNode mergeKLists(ListNode[] lists) {  
    // ...  
}
```



```
public ListNode mergeKLists(ListNode[] lists) {  
    PriorityQueue<ListNode> pq = new PriorityQueue<>();  
    for (int i = 0; i < lists.length; i++) {  
        if (lists[i] != null) {  
            pq.add(lists[i]);  
        }  
    }  
    ListNode Dummy = new ListNode();  
    ListNode temp = Dummy;  
    while (!pq.isEmpty()) {  
        ListNode rv = pq.poll();  
        Dummy.next = rv;  
        Dummy = Dummy.next;  
        if (rv.next != null) {  
            pq.add(rv.next);  
        }  
    }  
    return Dummy.next;  
}
```

```
Input: lists = [[1,4,5],[1,3,4],[2,6]]  
Output: [1,1,2,3,4,4,5,6]  
Explanation: The linked-lists are:  
[  
  1->4->5,  
  1->3->4,  
  2->6  
]
```



6  
7 9  
0 10  
4 5  
8 9  
4 10  
5 7

3  
2

0 10  
4 6  
4 5  
5 7  
7 9  
8 9

0-10

0-10  
4-6  
4-8  
5-7  
7-9  
8-9

1	2	-1	-4	-20
-8	-3	4	2	1
3	8	10	1	3
-4	-1	1	7	-6

-8 -3 4 2  
-4 -1 1 7 -6  
1 2 -1 -4 -20  
2 8