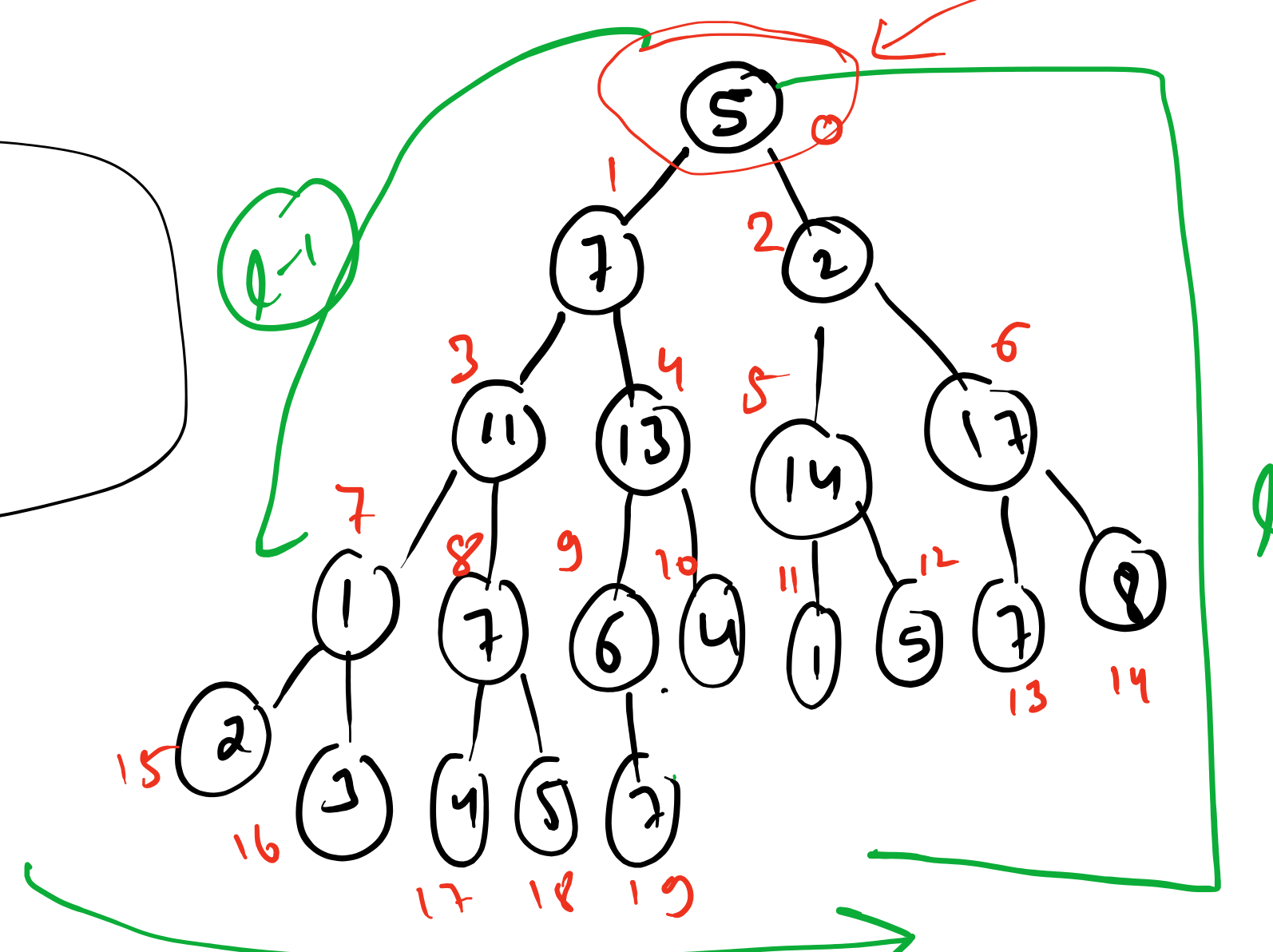
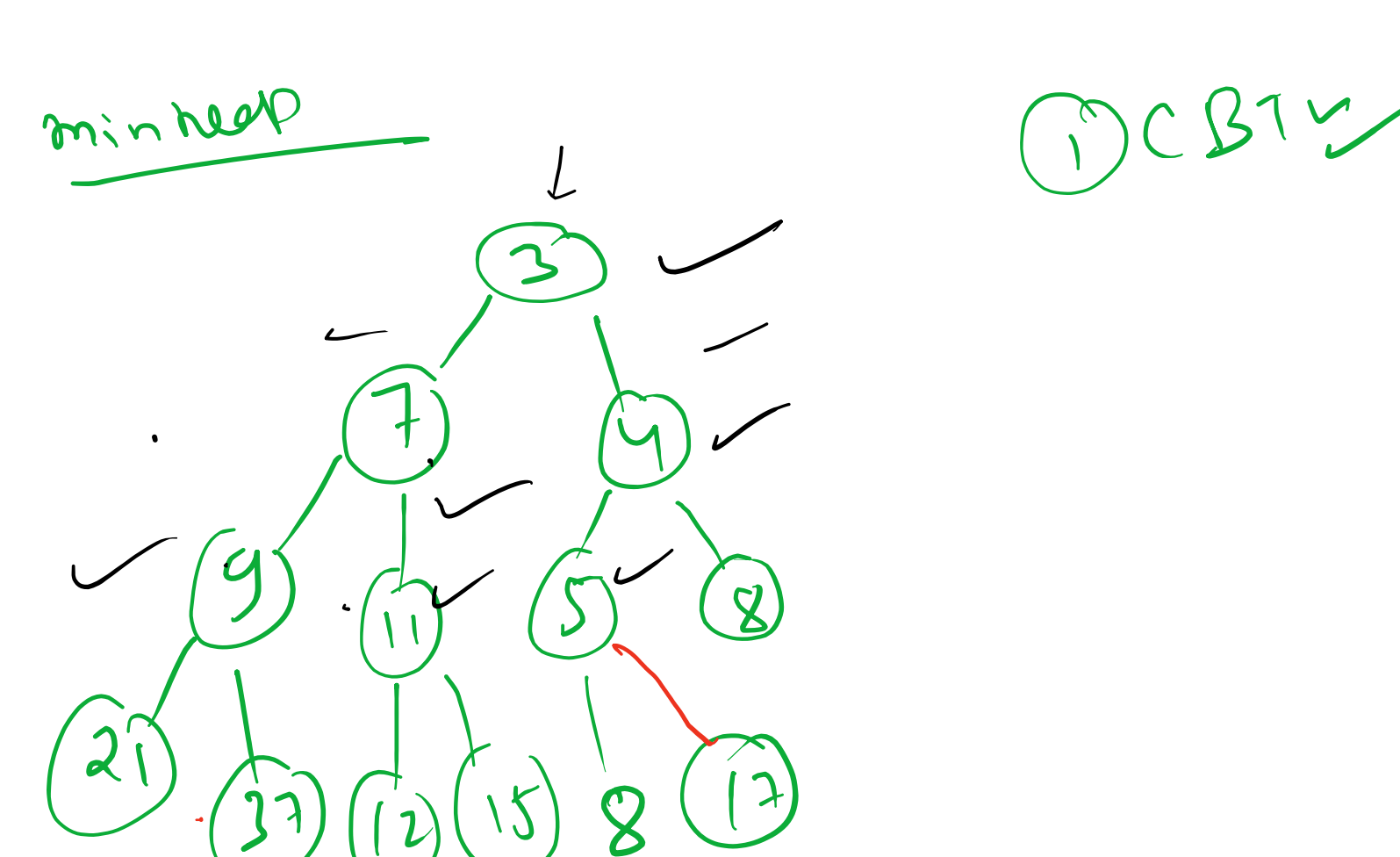


It is CBT → tree  
& complete binary



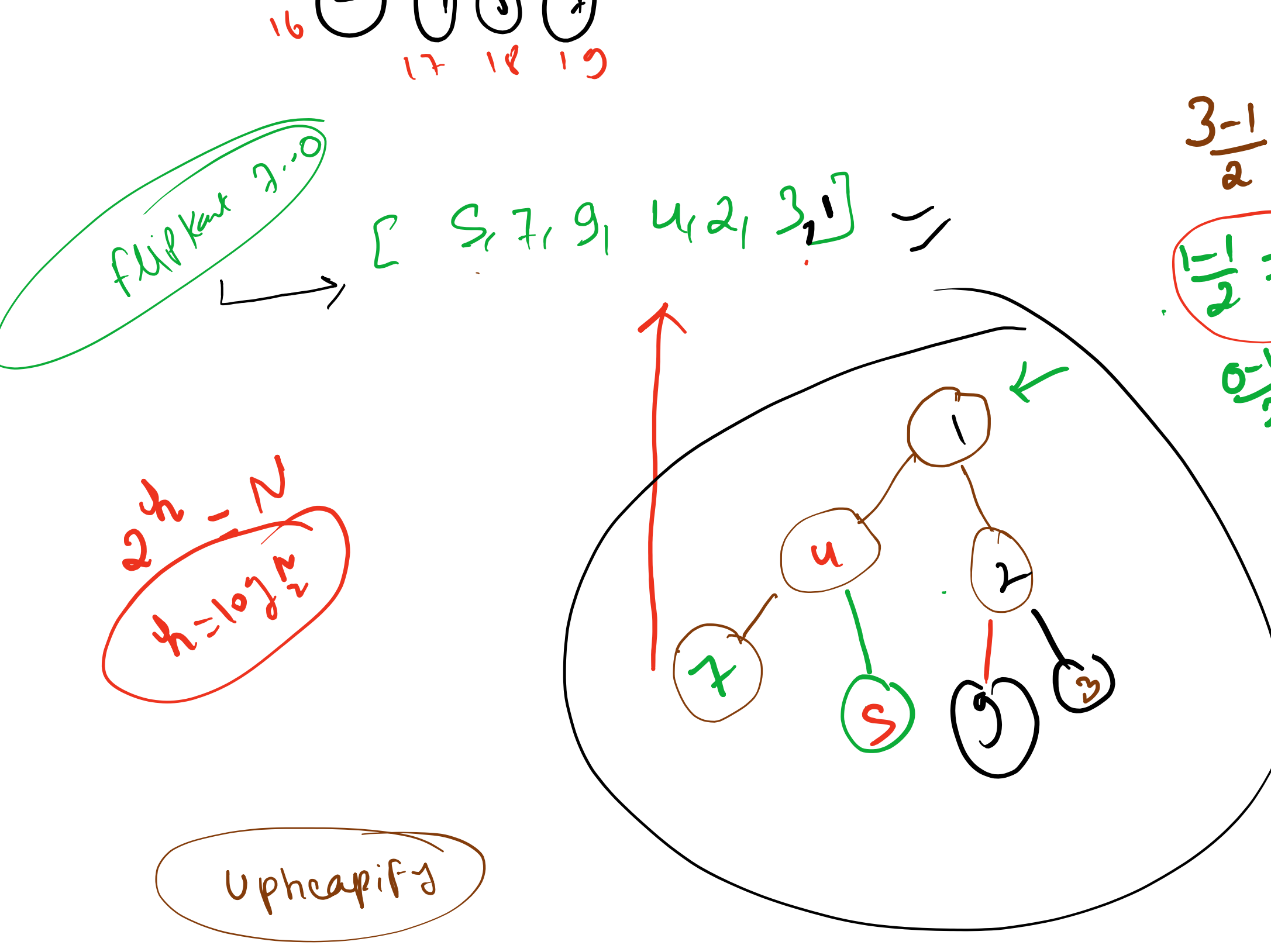
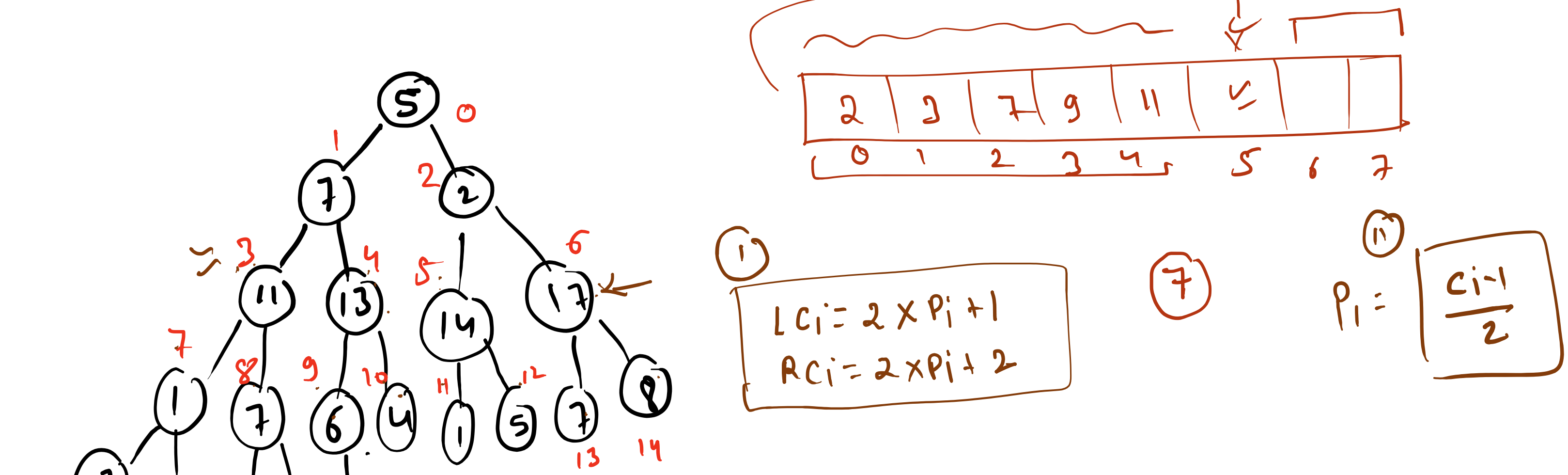
heap → CBT ⇒ Some priority  
Add

- min heap
  - max heap
- $\begin{matrix} c_1 \\ \uparrow P \\ c_2 \end{matrix}$   
min heap

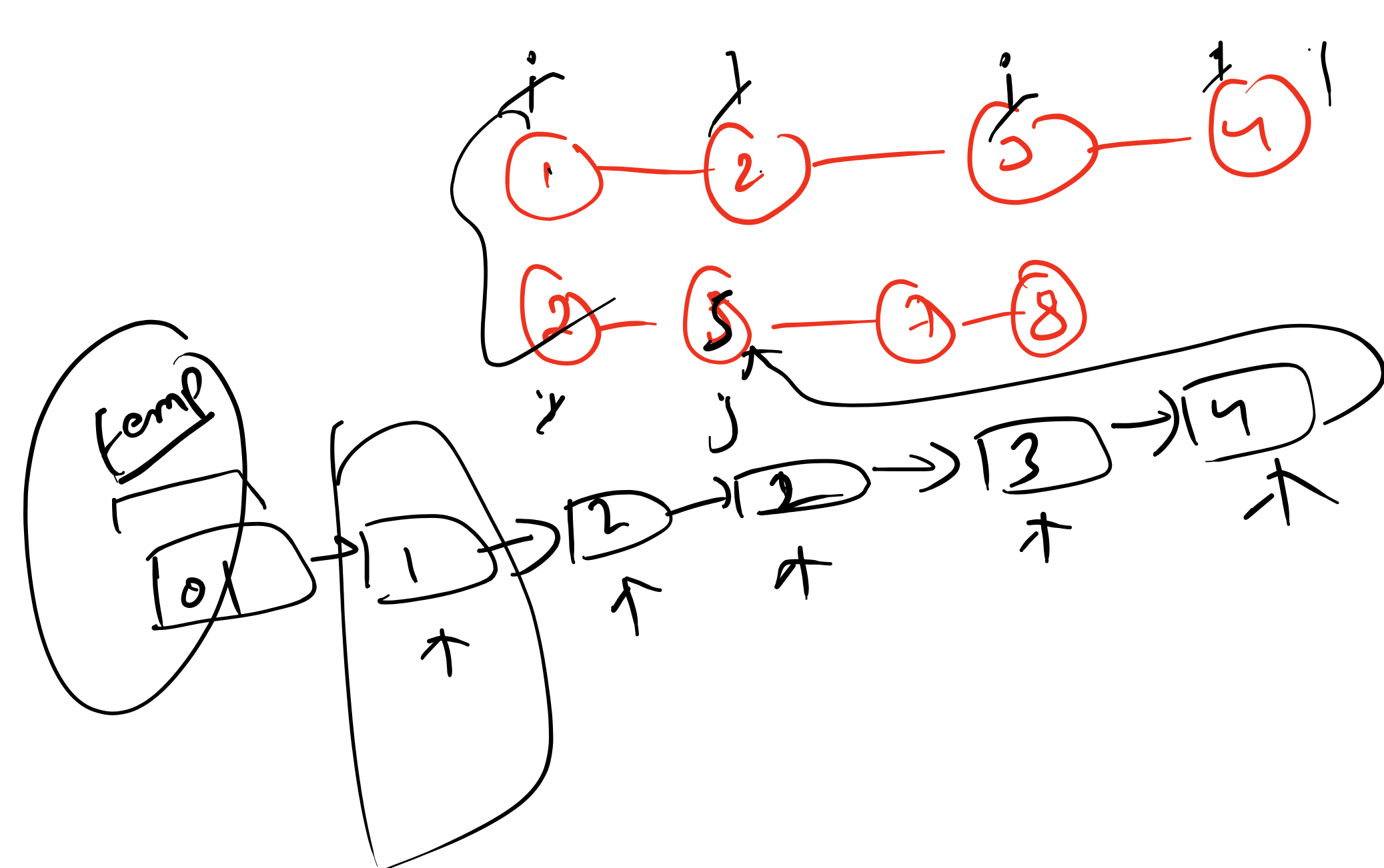
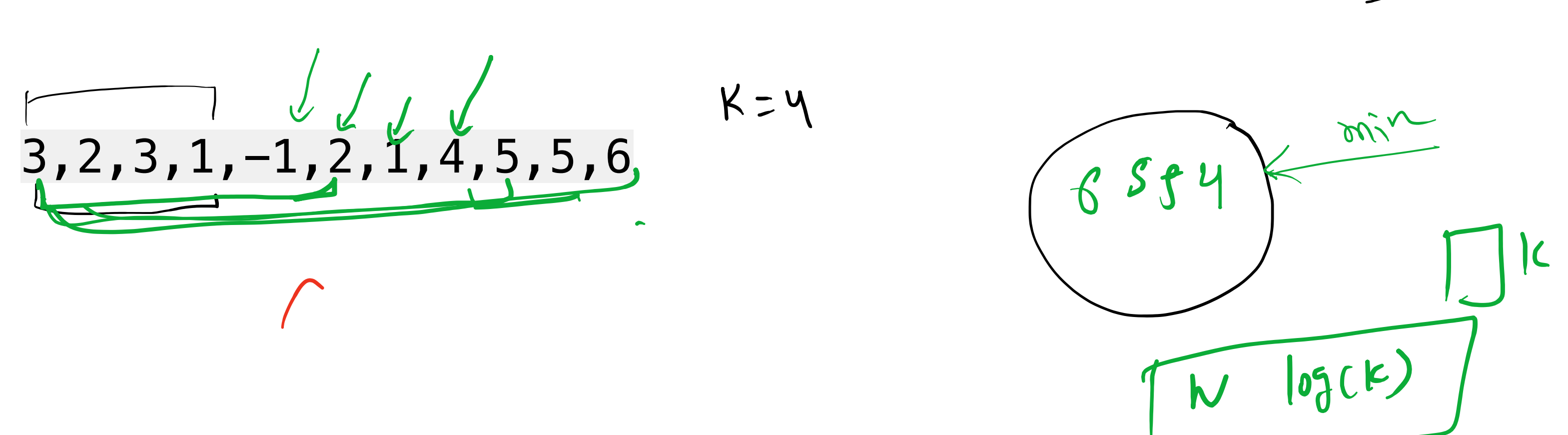
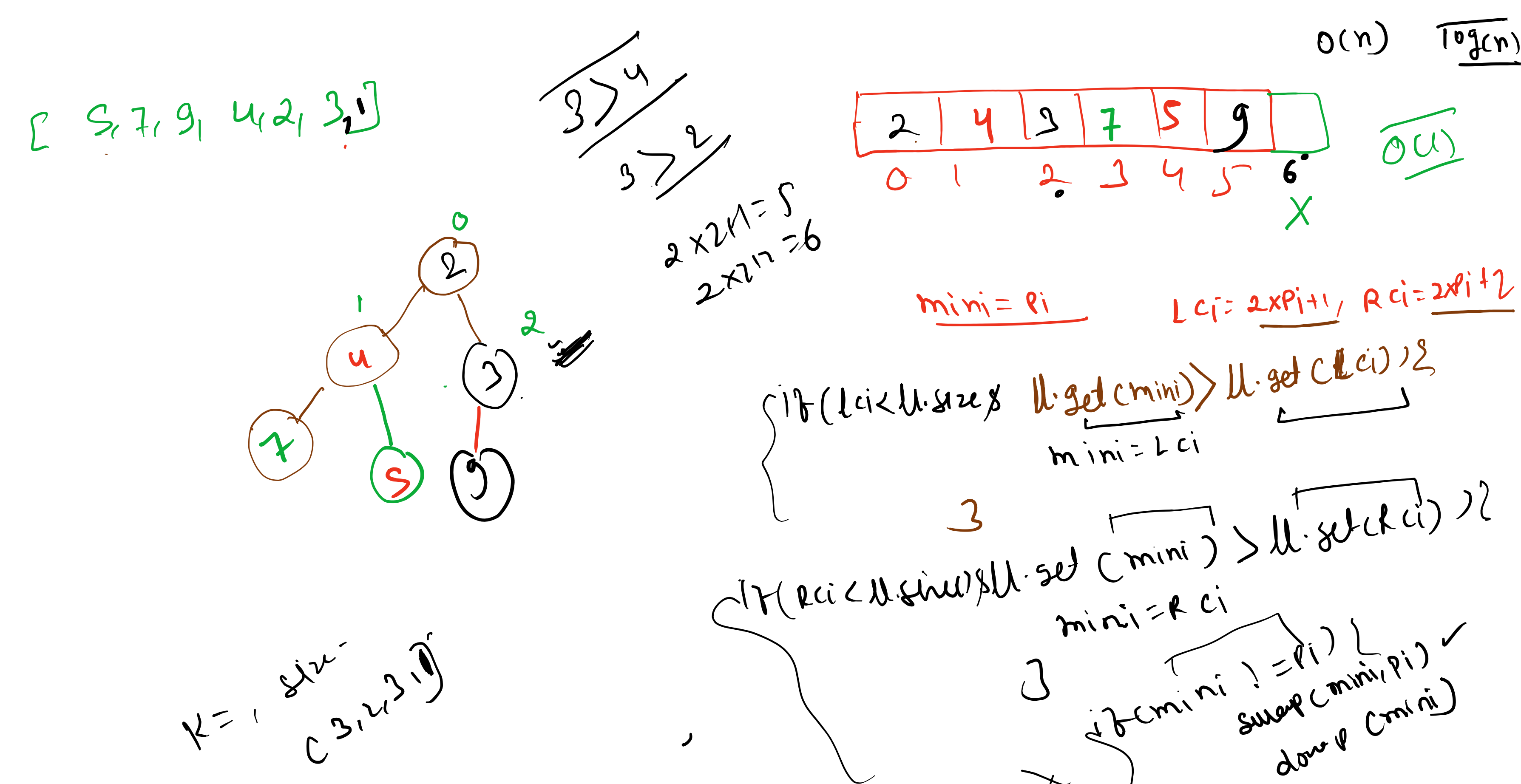


heap =

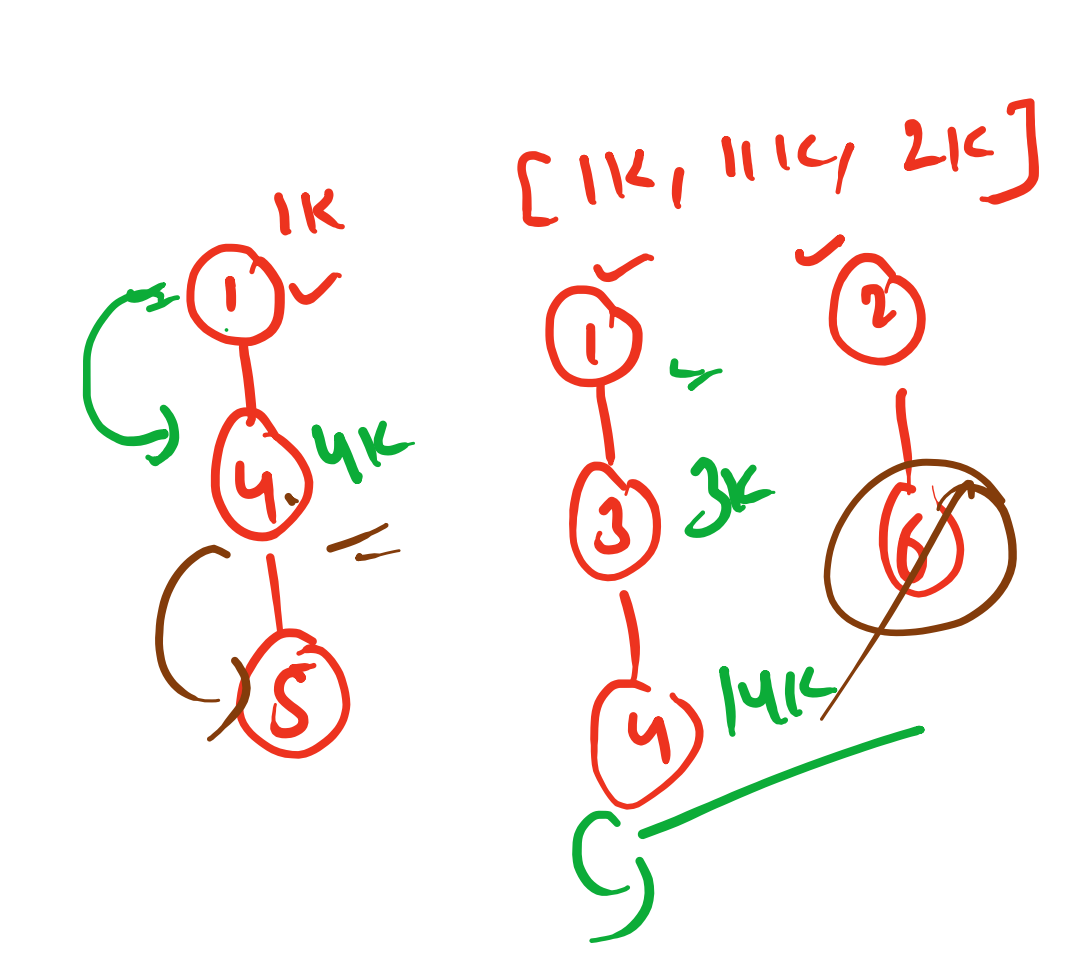
|         | Sorted | insert | heap        |
|---------|--------|--------|-------------|
| add     | $O(n)$ | $O(1)$ | $\log(n)$ ? |
| min     | $O(n)$ | $O(n)$ | $\log(n)$   |
| remove  | $O(1)$ | $O(n)$ | $O(1)$      |
| get min |        |        |             |



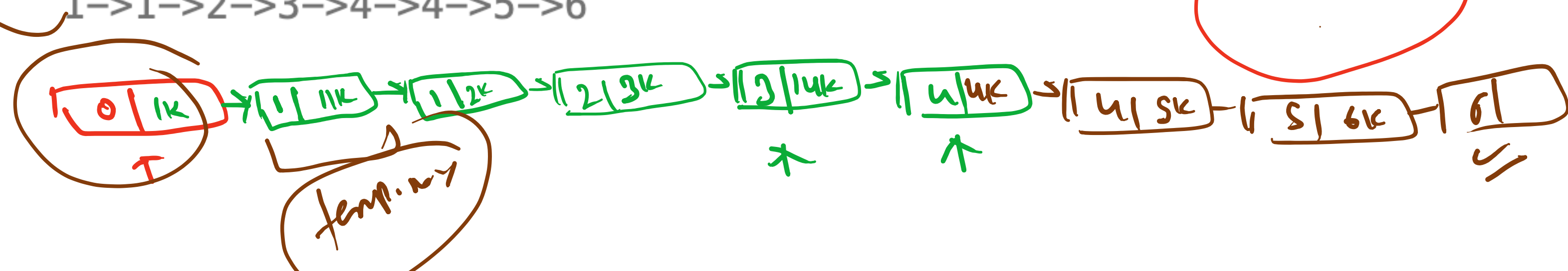
```
private ArrayList<Integer> ll = new ArrayList<>();
public void add(int item) {
    ll.add(item);
    upheapify(ll.size() - 1);
}
private void upheapify(int ci) {
    // TODO Auto-generated method stub
}
```



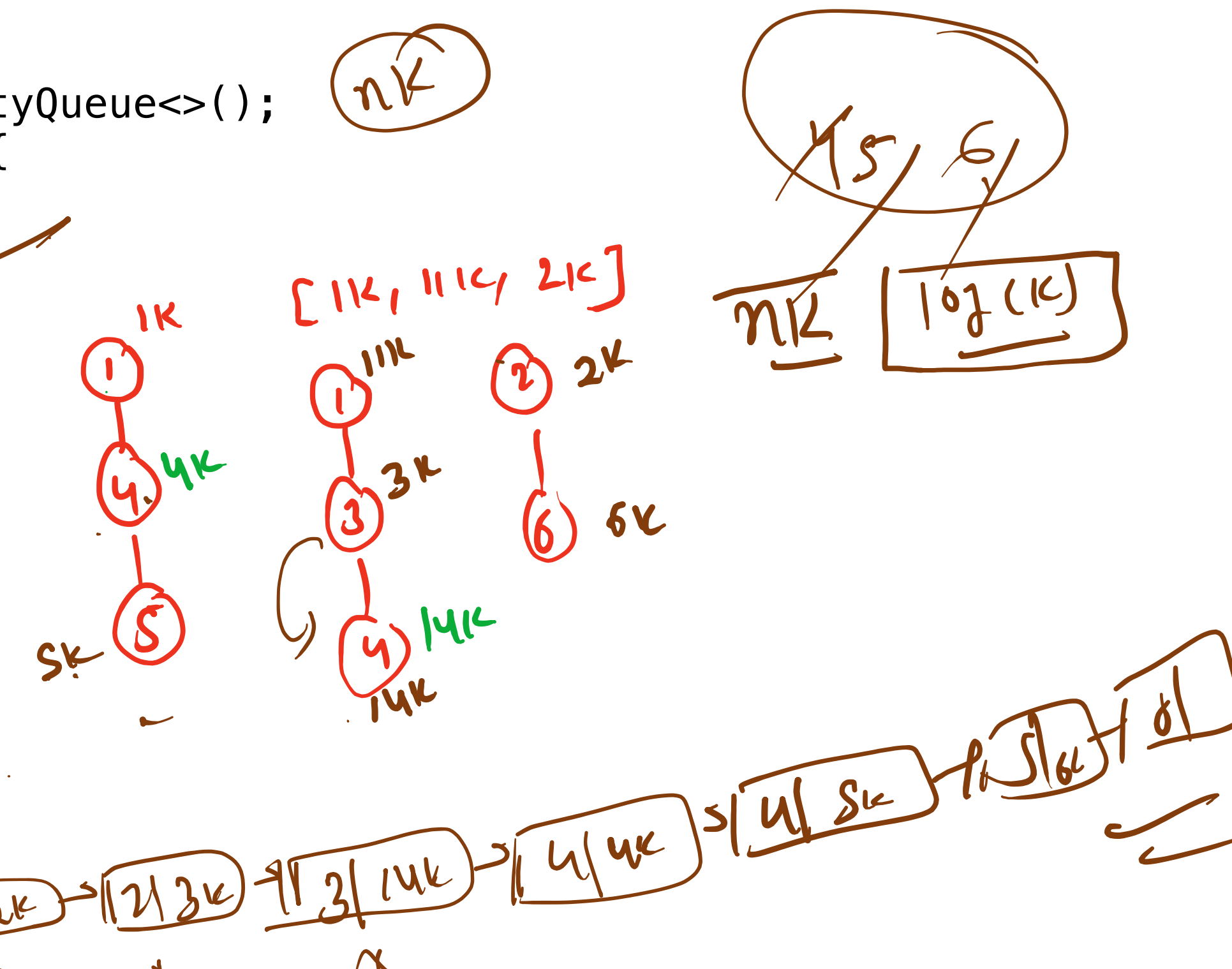
Input: lists = [[1,4,5],[1,3,4],[2,6]]  
Output: [1,1,2,3,4,4,5,6]  
Explanation: The linked-lists are:  
[  
1->4->5,  
1->3->4,  
2->6  
]



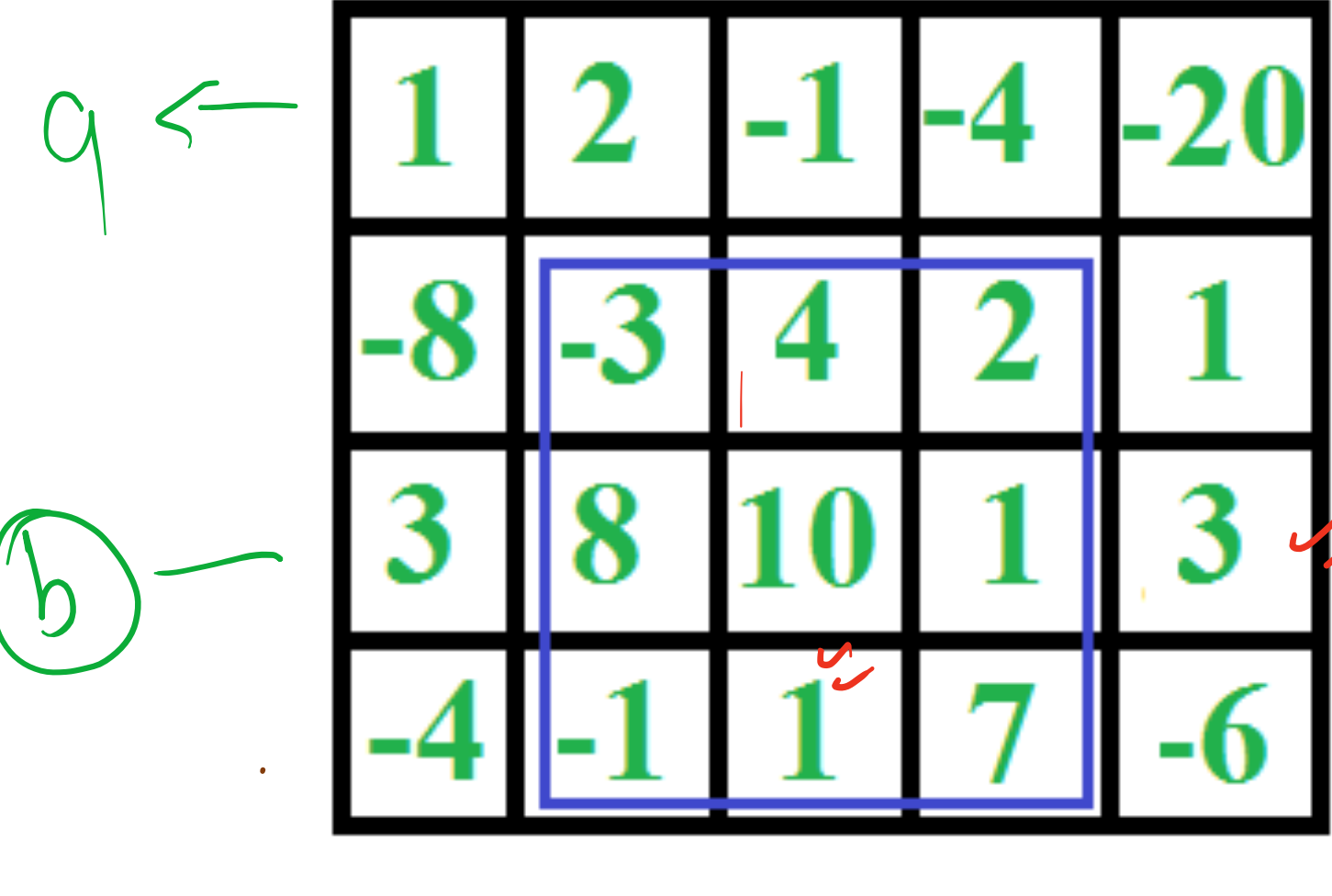
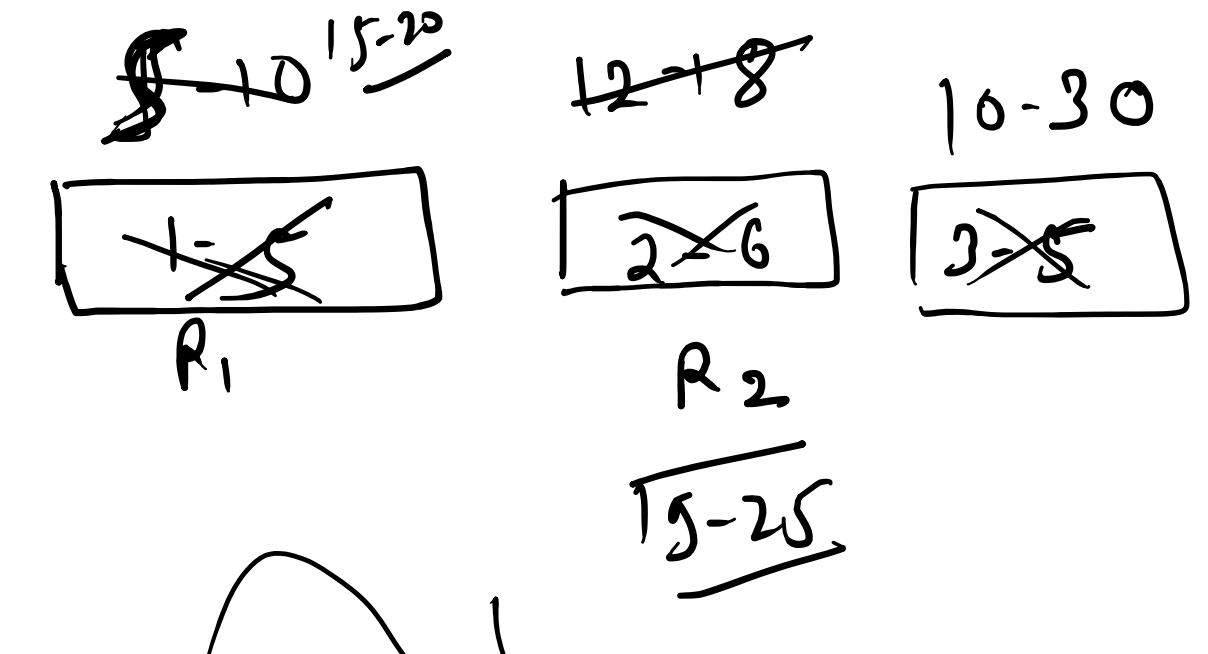
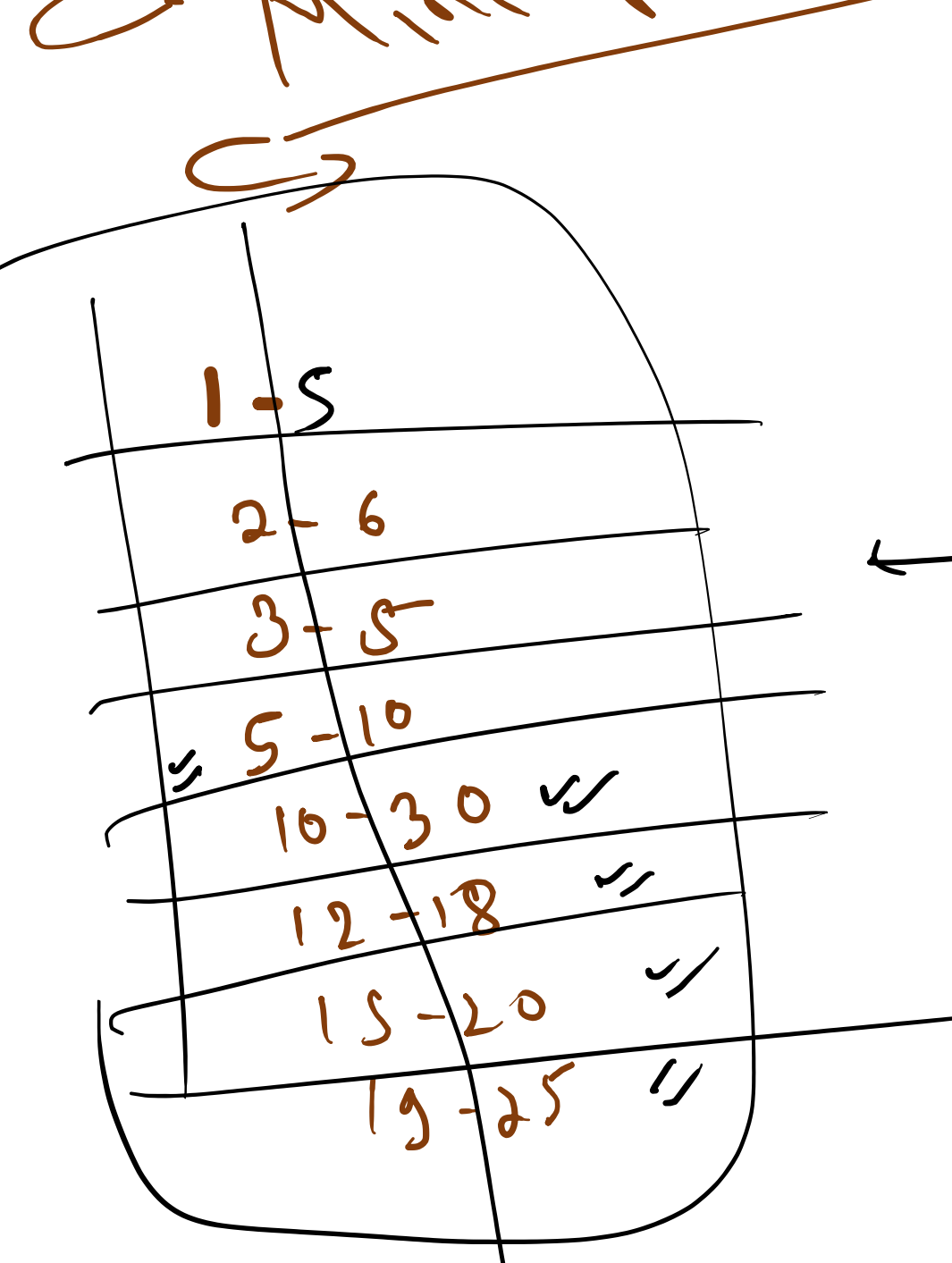
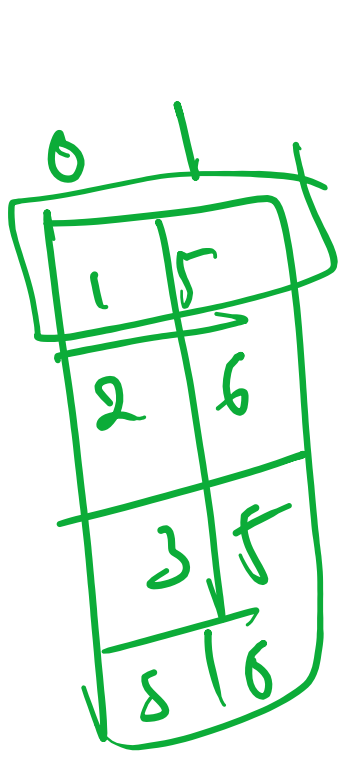
merging them into one sorted list:  
1->1->2->3->4->4->5->6



```
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 node = pq.poll();
        Dummy.next = node;
        Dummy = Dummy.next;
        if (node.next != null) {
            pq.add(node.next);
        }
    }
    return temp.next;
}
```

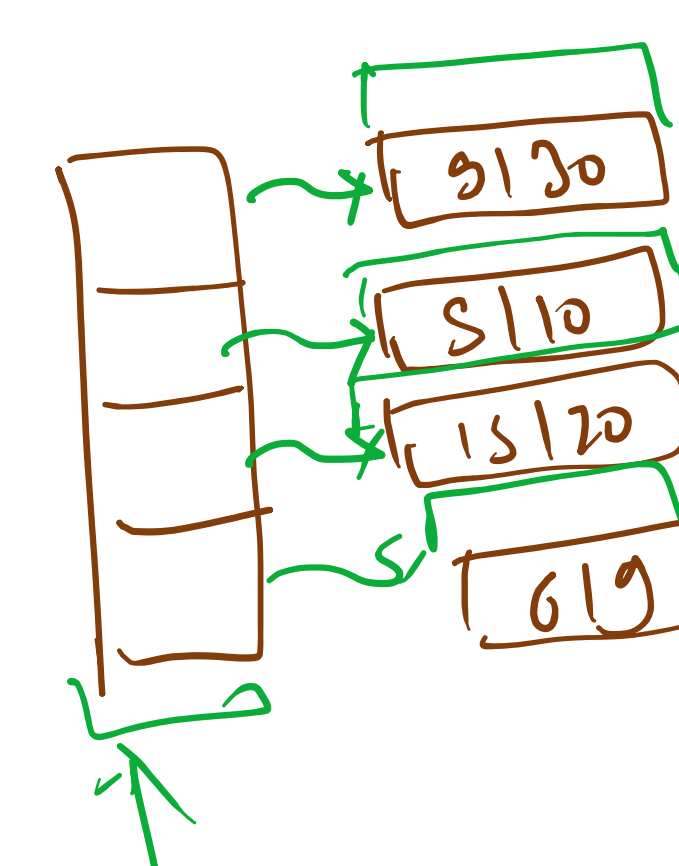


- $[10, 30]$
- $[5, 10]$
- $[15, 20]$
- $[2, 6]$
- $[12, 18]$
- $[19, 25]$
- $[1, 5]$
- $[3, 5]$



$q \leftarrow$

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



`int[][] intervals = { { 9, 30 }, { 5, 10 }, { 15, 20 }, { 6, 9 } };`

