Algorithm Assignment Feb 25

1. Demonstrate the operation of HOARE-PARTITION on the array A = <13, 19, 9, 5, 12, 8, 7, 4, 11, 2, 6, 21>. Show the values of the array after each iteration of the while loop in the lines of 4 to 11 in the code of lecture notes.

Q1.

A = [13, 19, 9, 5, 12, 8, 7, 4, 11, 2, 6, 21]

⇒ Hoare-Partition

initially ⇒ x=13, j=13, i=0

after the first iteration

⇒

A=[6, 19, 9, 5, 12, 8, 7, 4, 11, 2, 13, 21]

after the second iteration

⇒

A=[6, 2, 9, 5, 12, 8, 7, 4, 11, 19, 13, 21]

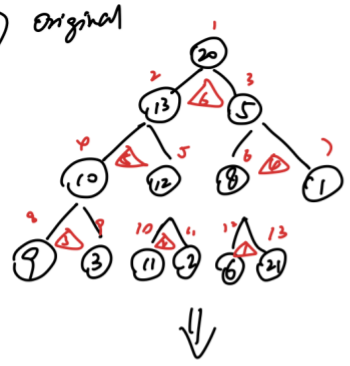
after the third iteration ⇒ at eighth iteration, loop ends

⇒

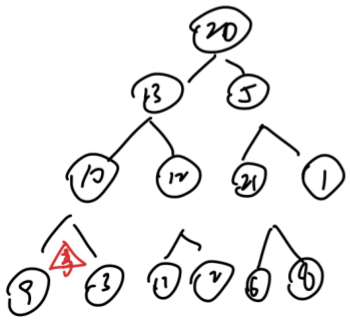
final value=A=[6, 2, 9, 5, 12, 8, 7, 4, 11, 19, 13, 21]

2. For the following array:
A = <20, 13, 5, 10, 12, 8, 1, 9, 3, 11, 2, 6, 21>
(a) Create a max heap using the algorithm BUILD-MAX-HEAP.
(b) Remove the largest item from the max heap you created in 3(a), using the HEAP-EXTRACT-MAX function. Show the array after you have removed the largest item.
(c) Using the algorithm HAX-HEAP-INSERT, insert 56 into the heap that resulted from question 3(b). Show the array after insertion.

Q3 (1) Original

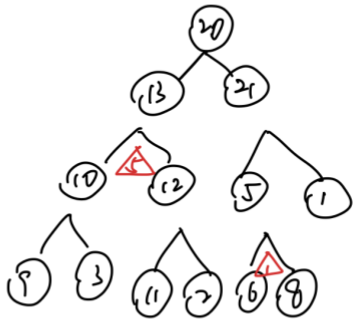


① swap 21 with 8



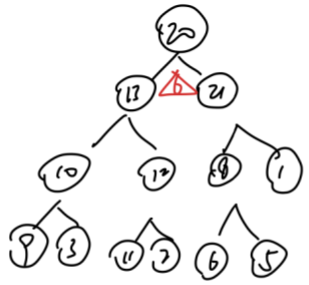
② ⇒ do nothing

③ ⇒ 10 > 9 > 3, do nothing. ④ ⇒ swap 21 with 5

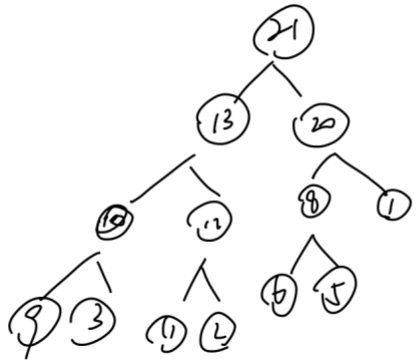


⇒ for the ⑤, do nothing

⇒ re - do △1 ⇒



△6
⇒ swap 20 with 21

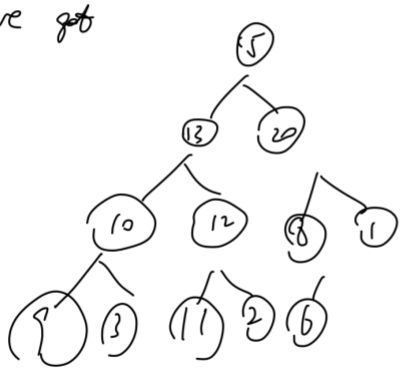


⇒ ⋱ max - heap

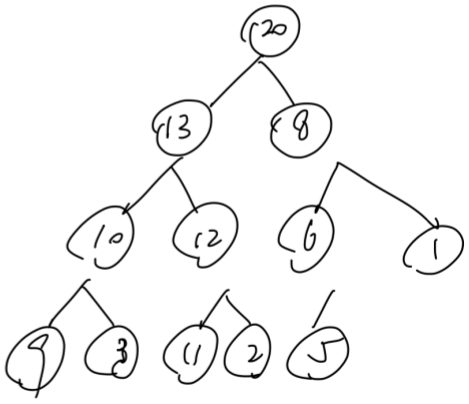
② remove 21 ⇒

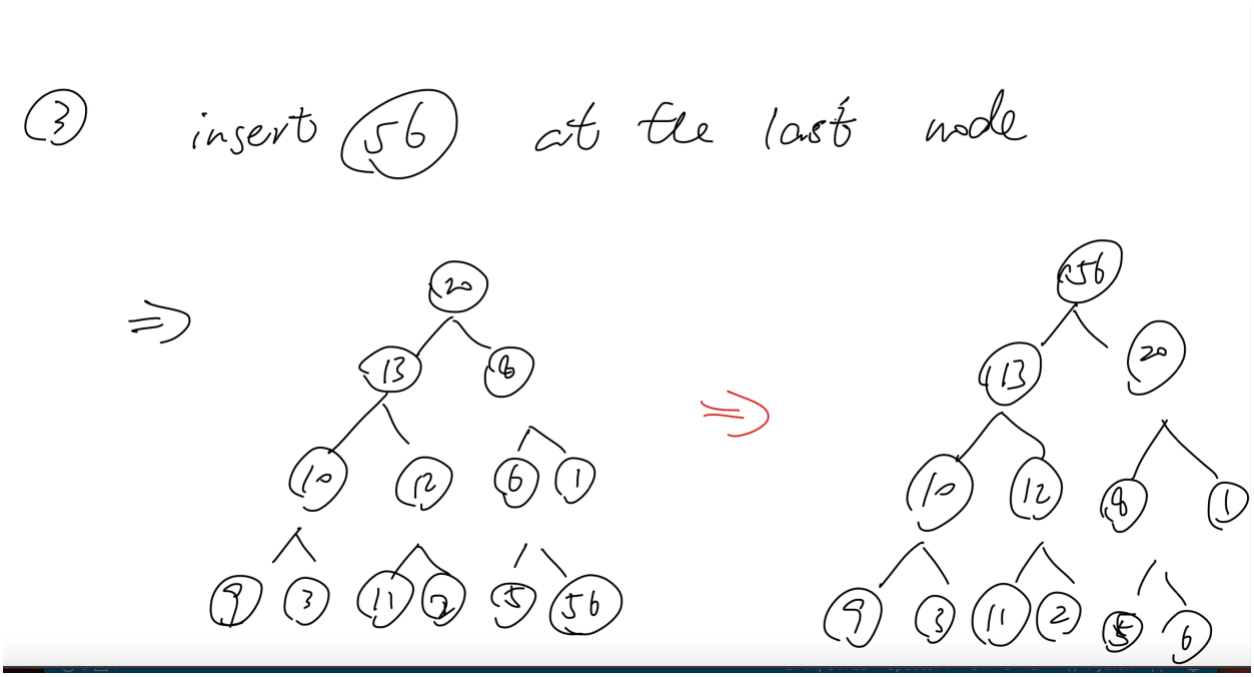
move last node to the top

we get



⇒





3. Design an algorithm to merge k sorted arrays, and return it as a new array. The new array should be made by splicing together the nodes of the k arrays. Additionally, the total number of elements of all arrays is kn. (Notice that the number of elements of each array is not necessary the same). Ideally, your algorithm should run in $O(kn \log k)$ time, lower algorithm will lose some points. Please give the procedure of your algorithm and analyze the running time. (Description is enough, you do not need to provide any pseudocode)
For example:
Input: A: <1, 5, 18>, B: <3, 6, 7, 11>, C: <2, 4, 9, 12, 13>
Output: <1, 2, 3, 4, 5, 6, 7, 9, 11, 12, 13, 18>

Answers:

First, we need to initialize a min-heap
Then we need to insert the first element of A, B, C into the heap.

This process will take O(k) time.

Then repeat the following steps till the algorithm go through to the end:

First step is to get the minimum value from the heap and
Create an array to store it

Second step is to replace heap root with the next element from the array. If it is the final element, no next element, replace root then call the heapify procedure that convert a binary tree into a heap data structure.

This heapify process will take O(logk)

Therefore, the repeated process will take knO(logk)

The overall algorithm will take O(k) + knO(logk) , which is O(knlogk)

4. Let T be a min heap storing n keys. Given an efficient algorithm for reporting all the keys in T
that are smaller than or equal to a given query key x (which is not necessarily in T). Note that the keys do not need to be reported in sorted order. Ideally, your algorithm should run in $O(k)$
time, where k is the number of keys reported that is smaller than x.

Answer:

Our algorithm will take the array, A[], the key, the index, i

```
        ## First, we define the base case,
if the I > len(A), stop the function,

If the A[i] <= key:
        We return the A[i]


        ##Then we define the two index for the recursion,
left = I * 2,
right = i*2 + 1

        ##Then we start to recursively calling the function by

Function(A, key, left)
Function(A, key, right)
```

***

If the A is a treeNOde, then pusedocode will be like

```
If (node, && node.val < key):

        Return key
Else:
        Function(A, key, node.left)
        Function(A, key, node.right)
```