Homework Sheet 6

Author	**Matriculation Number**	**Tutor**
Abdullah Oğuz Topçuoğlu	7063561	Maryna Dernovaia
Ahmed Waleed Ahmed Badawy Shora	7069708	Jan-Hendrik Gindorf
Yousef Mostafa Farouk Farag	7073030	Thorben Johr

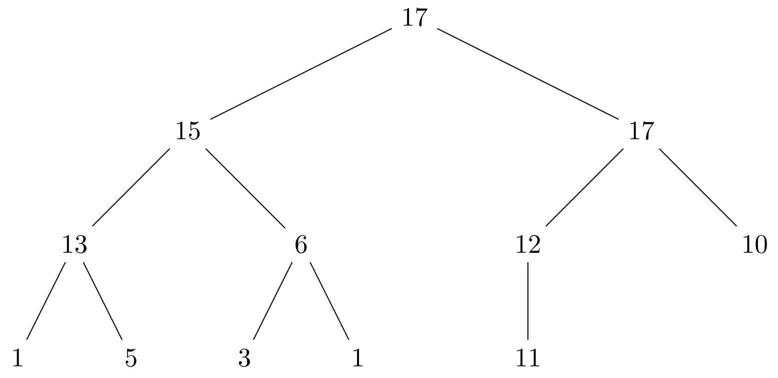
Exercise 1

(a)

We are given the array

$$A = [17, 15, 17, 13, 6, 12, 10, 1, 5, 3, 1, 11].$$

The heap tree would be



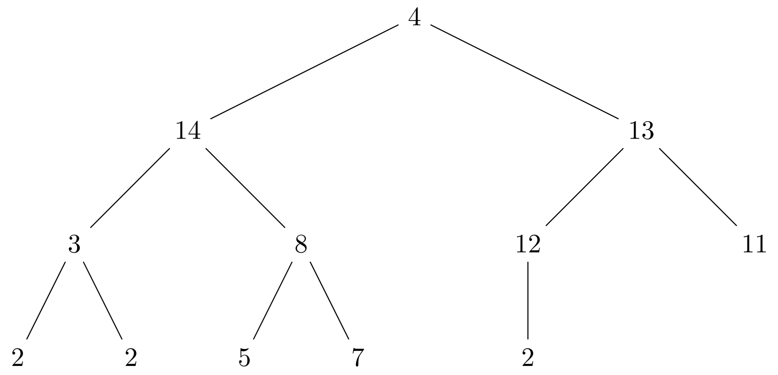
This is a heap since all nodes satisfy the max heap property.

(b)

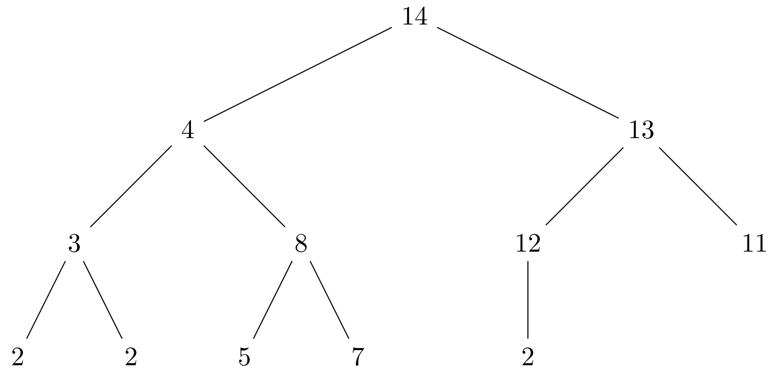
We are given the array

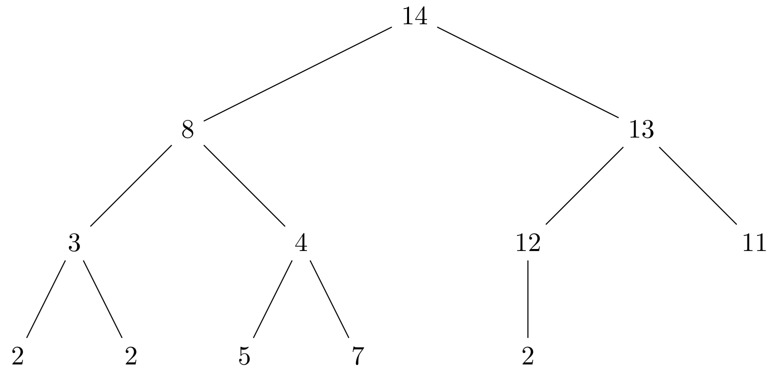
$$A = [4, 14, 13, 3, 8, 12, 11, 2, 2, 5, 7, 2].$$

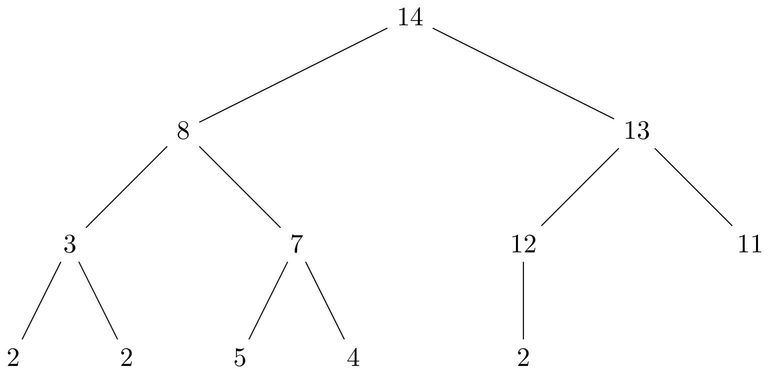
The heap tree would be



This is a near heap because only the root node (4) vioaltes the max heap property. Every other node satisfies it.
Steps of heapify operation:





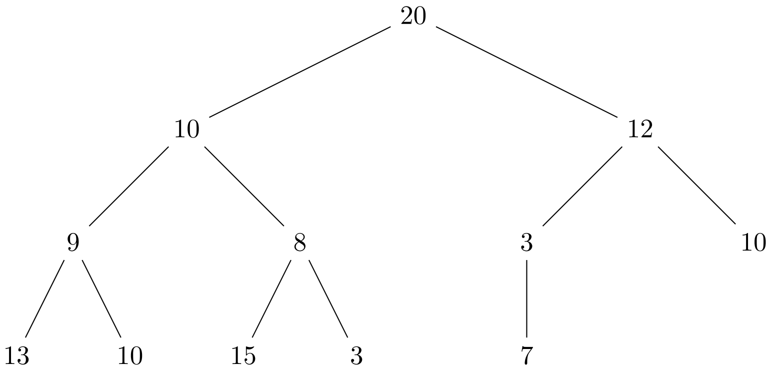


(c)

We are given the array

$$A = [20, 10, 12, 9, 8, 3, 10, 13, 10, 15, 3, 7].$$

The heap tree would be



This is not a heap also not a near heap because multiple nodes violate the max heap property. For example the node with the value 9 and the node with the value 8 both violates the max heap property and they are not descendants of each other.

Exercise 3

We are gonna use the same approach that we used in the lecture. We will build a max heap from the given array and then we will call DeleteMax() k many times.

```
int KthLargestElement(A[1..n], k)
  BuildMaxHeap(A)
  for i = 1 to k do
    result = DeleteMax(A)
  return result

void BuildMaxHeap(A[1..n]) // this is called makeheap() in the lecture slides
  for i = n/2 down to 1 do
    Heapify(A, i) // heapify function from the lecture
```

Running Time Analysis:

Building the max heap takes O(n) time as we saw in the lecture. Each call to DeleteMax() takes O(log n) time. Since we are calling DeleteMax() k many times, this part takes O(k log n) time. Therefore the total running time of the algorithm is

$$O(n) + O(k \log n) = O(n + k \log n).$$

Correctness Proof:

The BuildMaxHeap() function builds a valid max heap from the given array A. In a max heap the maximum element is always at the root node. The DeleteMax() function removes and returns the maximum element from the heap and then reestablishes the max heap property by calling Heapify(). So at the time we call DeleteMax() for the kth time we get the kth largest element from the original array A and then we return it.