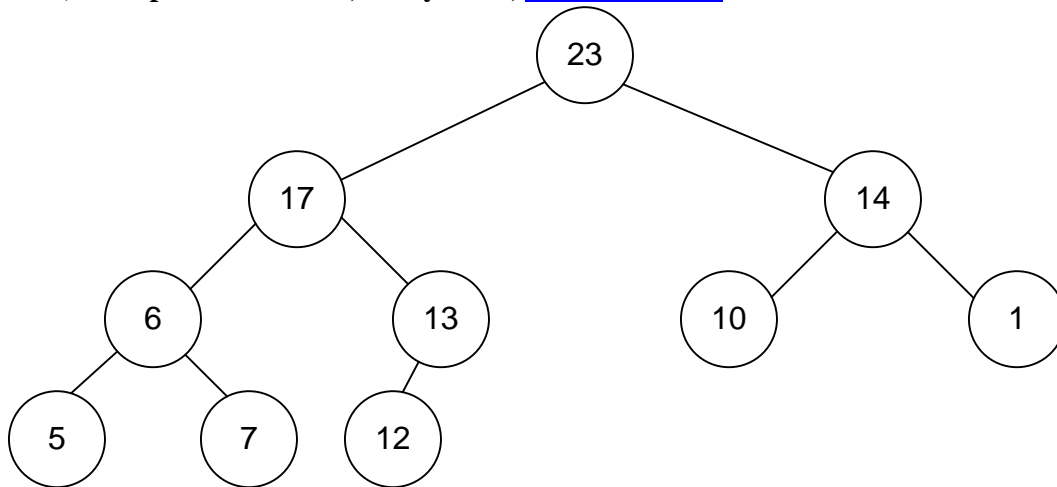


CSCI303 Fall 2006 Homework 5 Solutions

6.1-1) last updated fall 2006, Yuriy Brun, ybrun@usc.edu

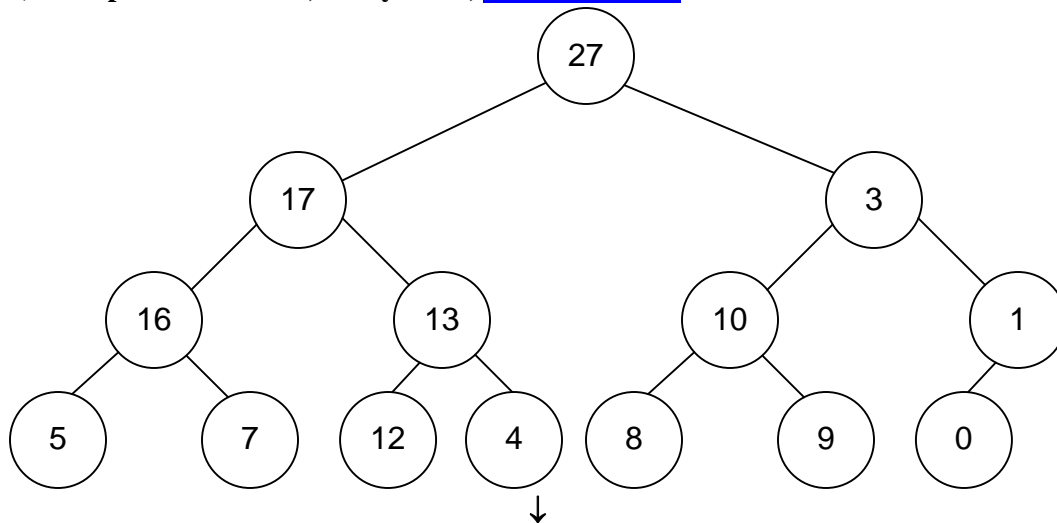
Since a heap is a semicomplete binary tree, maximum elements are when all the leaves of height h are present, so total number of elements $= 2^{h+1} - 1$. Minimum elements are when only one leaf of height h is present. So total number of elements $= (2^h - 1) + 1 = 2^h$.

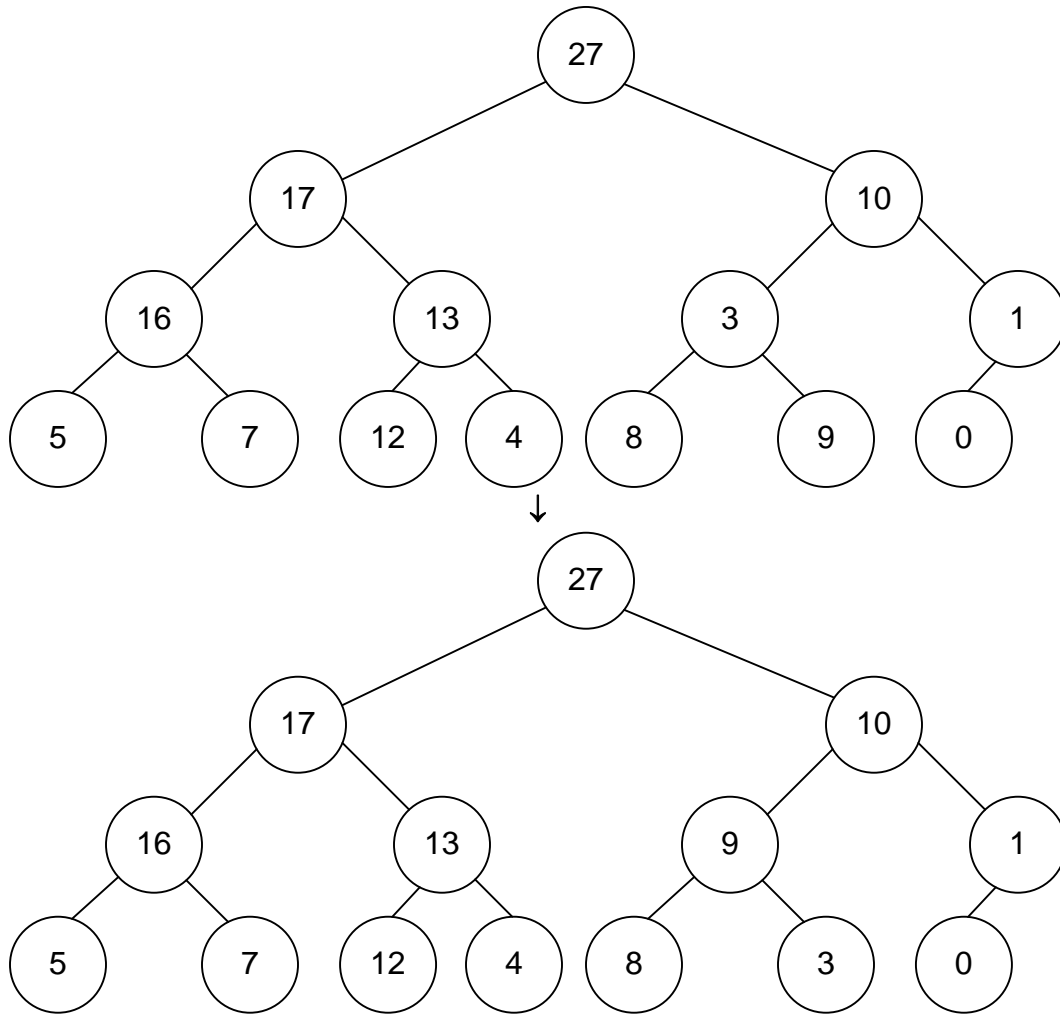
6.1-6) last updated fall 2006, Yuriy Brun, ybrun@usc.edu



This is not a max-heap since the parent of 7 is 6, and $6 < 7$.

6.2-1) last updated fall 2006, Yuriy Brun, ybrun@usc.edu



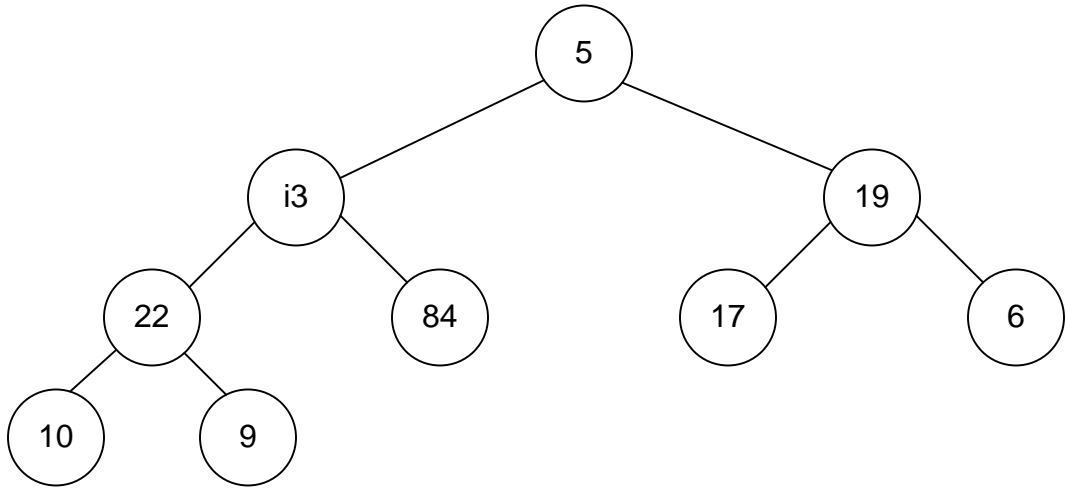
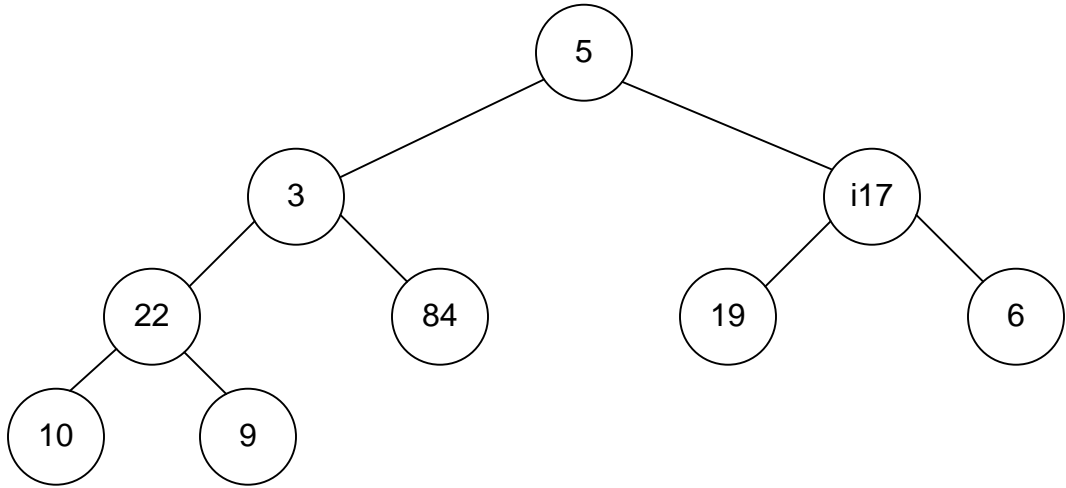
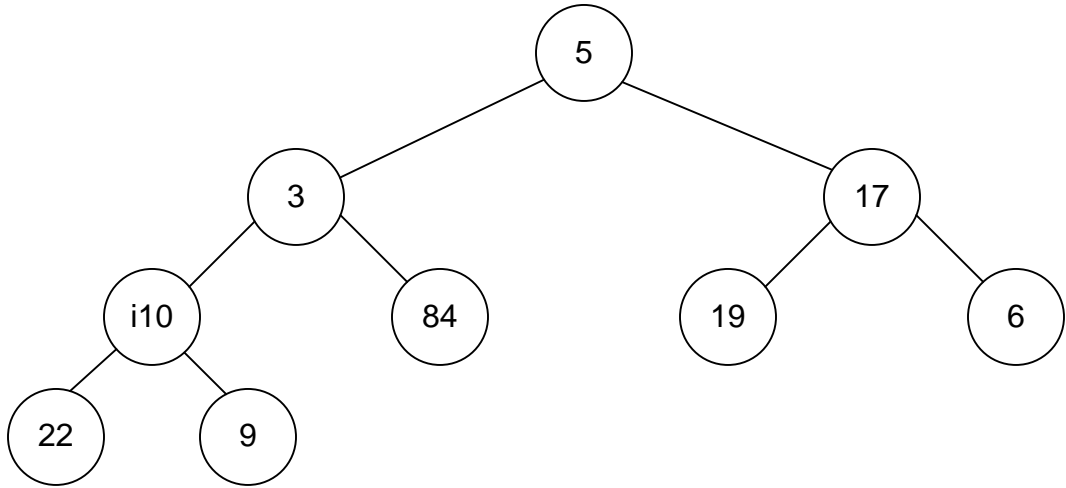


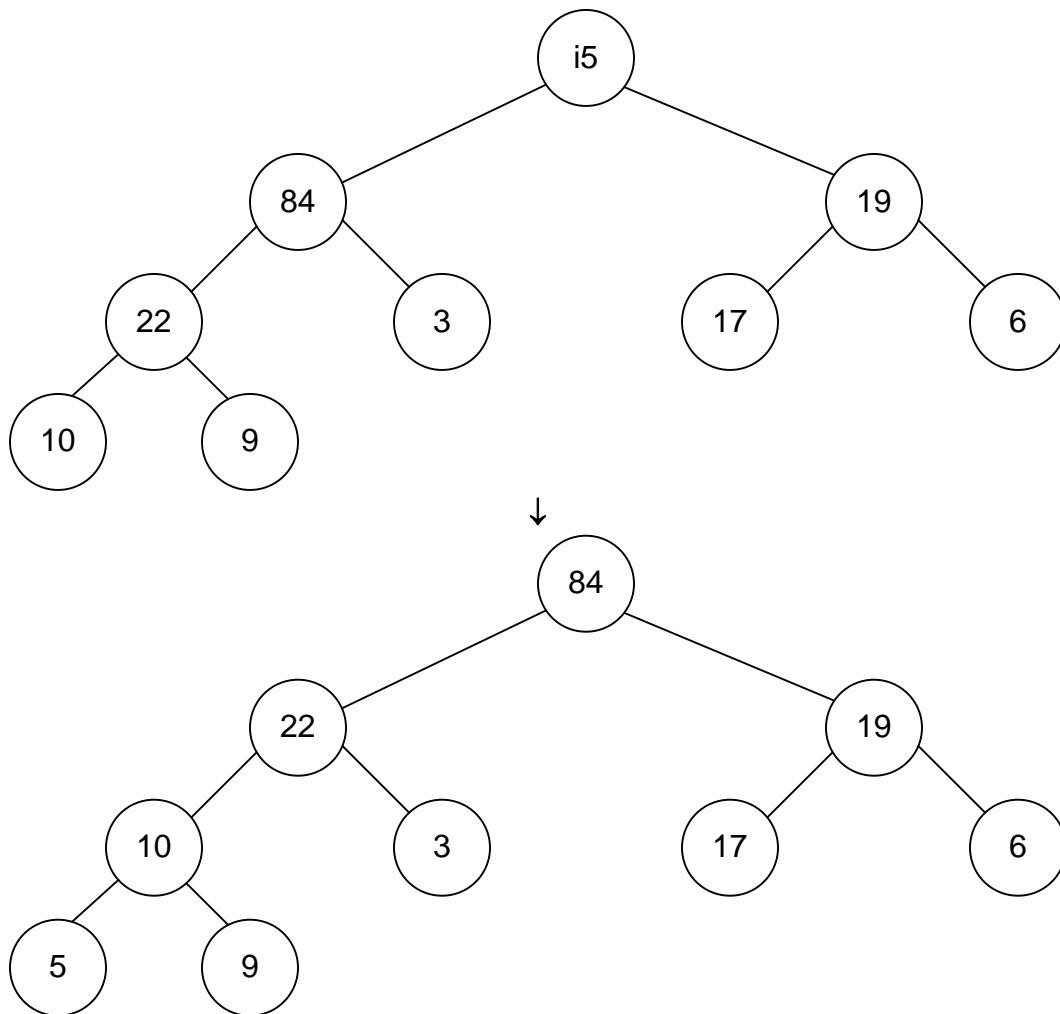
6.2-6) last updated fall 2006, Yuriy Brun, ybrun@usc.edu

Consider a heap of n nodes where the root node has been changed to contain the smallest value of all the nodes. Now when we call max-heapify on the root, the value will have to be exchanged down with its child at every level, till it reaches the lowest level, this is because, after every swapping, the value will still be smaller than both its children (since it is the minimum), until it reaches the lowest level where it has no more children. So in such a heap, the number of exchanges to max-heapify the root will be equal to the height of the tree, which is $(\lg n)$. So the worst case running time is $\Omega(\lg n)$.

6.3-1) last updated fall 2006, Yuriy Brun, ybrun@usc.edu

$A=[5,3,17,10,84,19,6,22,9]$





6.4-4) last updated fall 2006, Yuriy Brun, ybrun@usc.edu

Heapsort executes Build-Max-Heap which takes $\Theta(n)$ time. Assume that in the worst case, the heap is represented by the array $[n, n-1, n-2, \dots, 1]$. Then there are n calls to Max-Heapify, and each will take exactly the height of the tree time because the element at the top of the heap will always be the smallest element in the set.

$$T_{HS}^{WC}(n) = \Theta(n) + \sum_{i=1}^n \lceil \lg i \rceil \geq \Theta(n) + \int_1^n (\lg i) di = \Theta(n) + n \lg n - n = \Omega(n \lg n).$$

Alternatively, we have already shown that all comparison sorts must be $\Omega(n \lg n)$.

6.5-7) last updated fall 2006, Yuriy Brun, ybrun@usc.edu

```
Heap-Delete(A,i) {  
  exchange (A[i] ↔ A[length(A)])  
  heap-size[A] = heap-size[A]-1  
  Heapify(A,i)  
}
```

The algorithm deletes the element at node i , and replaces it with the last element. Then the algorithm runs Heapify from the node i . The first two lines execute in constant time and Heapify runs in $O(\lg n)$ time, so this algorithm runs in $O(\lg n)$ time.