CS / MCS 401 Homework 4 grader solutions

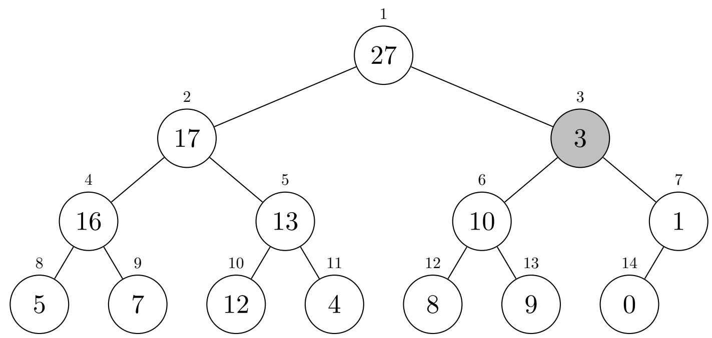
assignment due July 11, 2016
written by Jānis Lazovskis

Questions from CLRS.

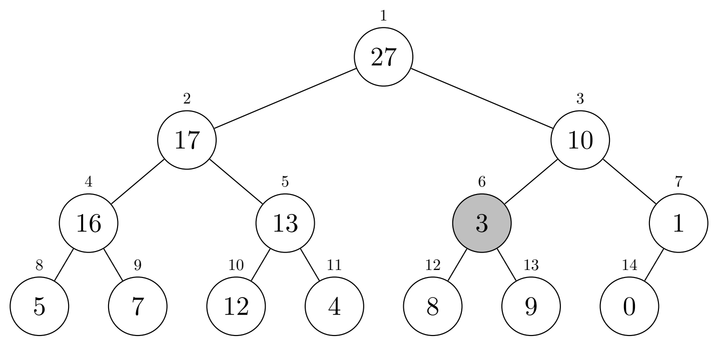
6.2-1 *(p.156)* *Using Figure 6.2 as a model, illustrate the operation of* MAX-HEAPIFY$(A, 3)$ *on the array* $A = \langle 27, 17, 3, 16, 13, 10, 1, 5, 7, 12, 4, 8, 9, 0 \rangle$.

Following the algorithm, we find that node 3 is exchanged with node 6, then node 6 is exchanged with node 13. The diagrams illustrating these operations are given below.

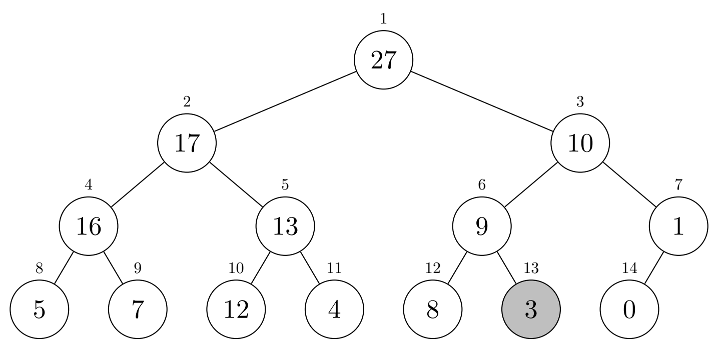
Step 0:



Step 1:



Step 2:



■

6.2-3 *(p.156)* *What is the effect of calling* MAX-HEAPIFY(A, i) *when the element* $A[i]$ *is larger than its children?*

There is no effect. All three **if** conditions fail, *largest* is set to i, and the process terminates wihtout having changed anything in the heap. ■

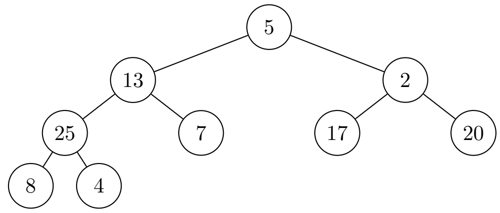
6.2-4 *(p.156)* *What is the effect of calling* MAX-HEAPIFY(A, i) *for* $i > A.heap\text{-}size/2$*?*

If $i > A.heap\text{-}size/2$, then node i has no children (and it is either at the second lowest or lowest level of the binary tree). Moreover, LEFT(i) and RIGHT(i) are larger than $A.heap\text{-}size$, meaning that lines 3 and 6 (the first two **if** conditions) of the algorithm will return errors, because the array index will be out of range. ■

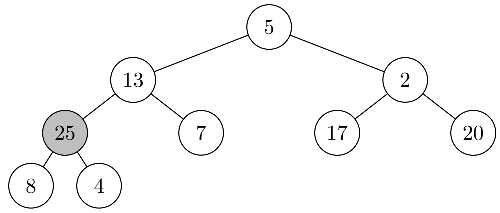
6.4-1 *(p.160)* *Using Figure 6.4 as a model, illustrate the operation of* HEAPSORT *on the array* $A = \langle 5, 13, 2, 25, 7, 17, 20, 8, 4 \rangle$.

Since $A.length = 9$, the command MAX-HEAPIFY(A, i) is called for $i = 4, 3, 2, 1$. The action of BUILD-MAX-HEAP is as follows (these first few diagrams are not required for a correct answer), with the nodes exchanged at each step shaded:

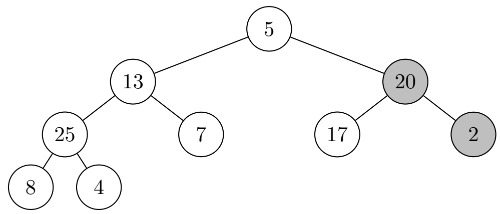
BUILD-MAX-HEAP(A):



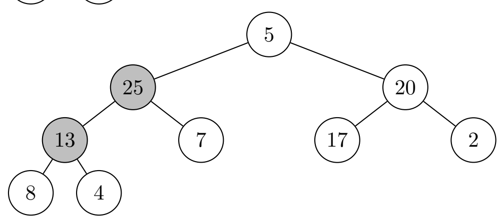
MAX-HEAPIFY$(A, 4)$:



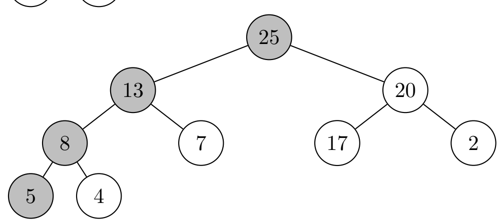
MAX-HEAPIFY$(A, 3)$:



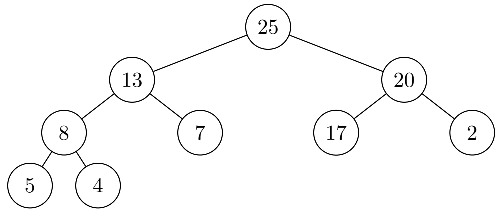
MAX-HEAPIFY$(A, 2)$:



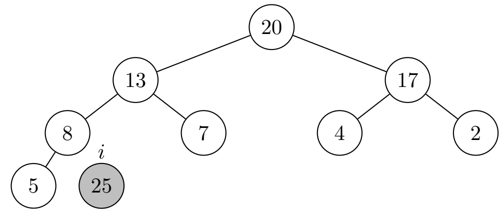
MAX-HEAPIFY$(A, 1)$:



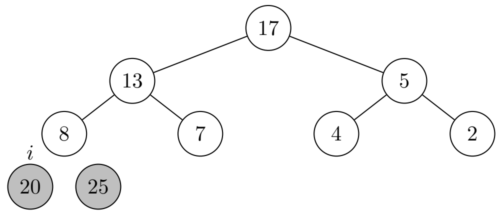
Now we follow Figure 6.4 (these diagrams are required for a correct answer):



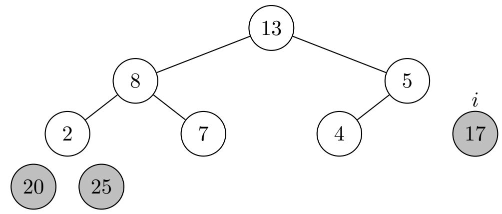
Step 0



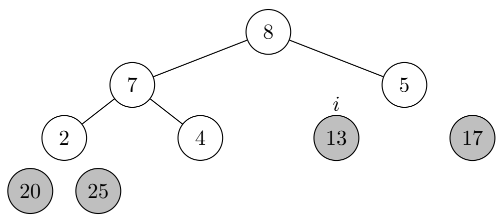
Step 1



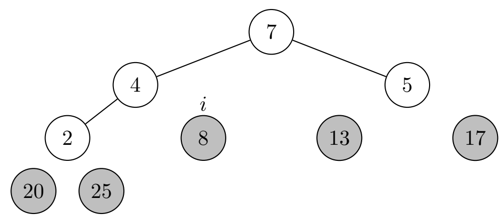
Step 2



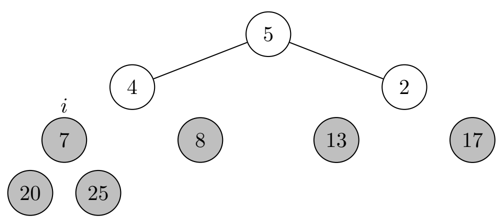
Step 3



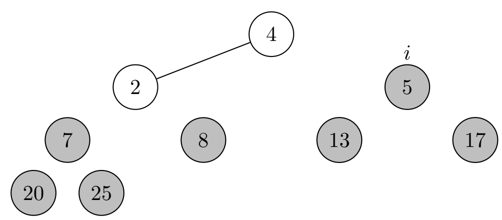
Step 4



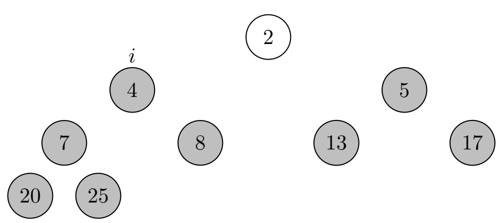
Step 5



Step 6



Step 7



Step 8

This gives a final sorted array $A = \langle 2, 4, 5, 7, 8, 13, 17, 20, 25 \rangle$. ∎

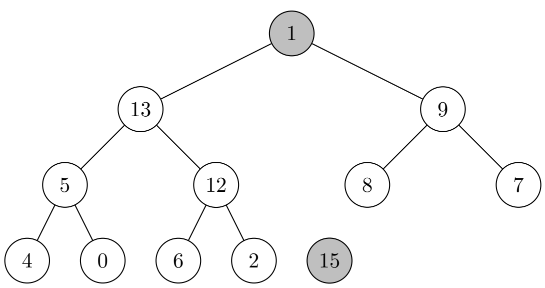
6.4-3 *(p.160)* *What is the running time of* HEAPSORT *on an array A of length n that is already sorted in increasing order? What about decreasing order?*

If A is sorted in increasing order, BUILD-MAX-HEAP will attain the maximum running time of $\Theta(n)$, since it tries to order the array in a decreasing order. The $n-1$ calls MAX-HEAPIFY$(A, 1)$ will take at most $O(\log_2(n))$ time (there are no particular time saves from max-heapifying an ordered array), hence the running time of HEAPSORT will be $O(n \log_2(n))$.

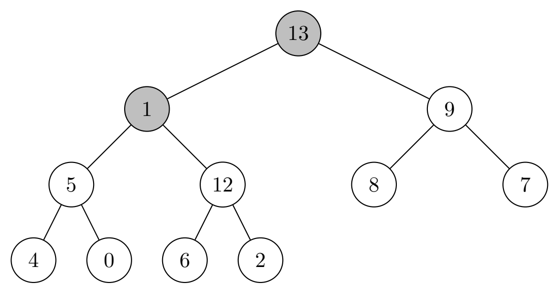
If A is sorted in decreasing order, MAX-HEAPIFY(A, i) has running time $O(1)$ for any i (since it never calls itself recursively, as $largest = i$ for all i. However, this makes no difference in the running time of BUILD-MAX-HEAP, as we still get $\Theta(n)$ running time due to the $\lfloor n/2 \rfloor$ calls to MAX-HEAPIFY. Here as well the $n-1$ calls MAX-HEAPIFY$(A, 1)$ will take at most $O(\log_2(n))$ time (completely reversing the order of an array certainly does not save time). Hence the running time of HEAPSORT will be $O(n \log_2(n))$. ∎

6.5-1 *(p.164)* *Illustrate the operation of* HEAP-EXTRACT-MAX *on the heap* $A = \langle 15, 13, 9, 5, 12, 8, 7, 4, 0, 6, 2, 1 \rangle$.

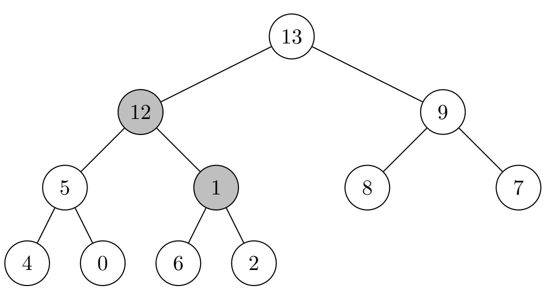
First we set $max = 15$ (which will be the returned value), then set $A[1] = 1$ and shorten the array (Step 0), and then do MAX-HEAPIFY$(A, 1)$ on the remaining array (Steps 1-3). The nodes which are exchanged in each step are darkened.



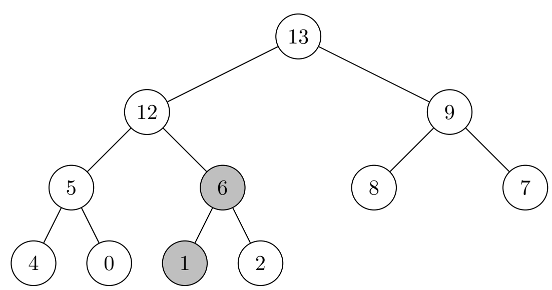
Step 0



Step 1



Step 2



Step 3

Now $A = \langle 13, 12, 9, 5, 6, 8, 7, 4, 0, 1, 2 \rangle$ and HEAP-EXTRACT-MAX returns the value 15. ∎