

CSCI 303 Homework 4

Problem 1 (6.1-1):

What are the minimum and maximum numbers of elements in a heap of height h ?

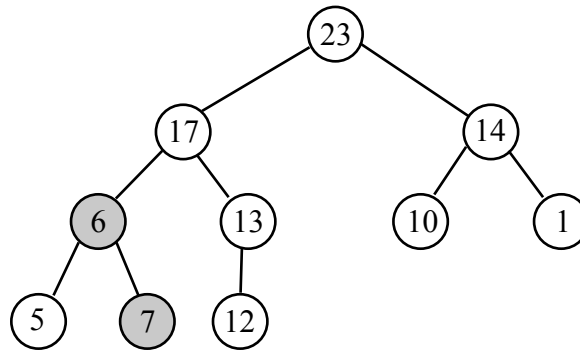
Solution 1:

A heap is a semi-complete binary tree, so the minimum number of elements in a heap of height h is 2^h , and the maximum number of elements in a heap of height h is $2^{h+1} - 1$.

Problem 2 (6.1-6):

Is the sequence $\langle 23, 17, 14, 6, 13, 10, 1, 5, 7, 12 \rangle$ a max-heap?

Solution 2:

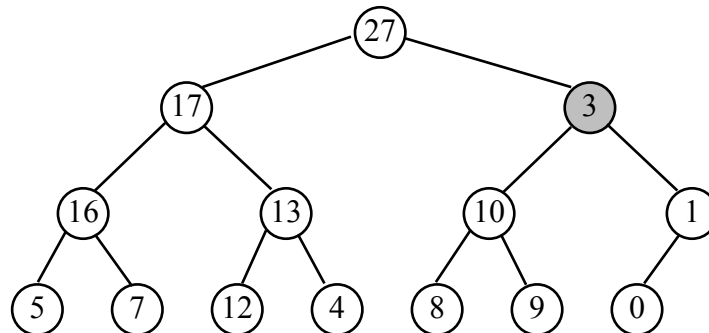


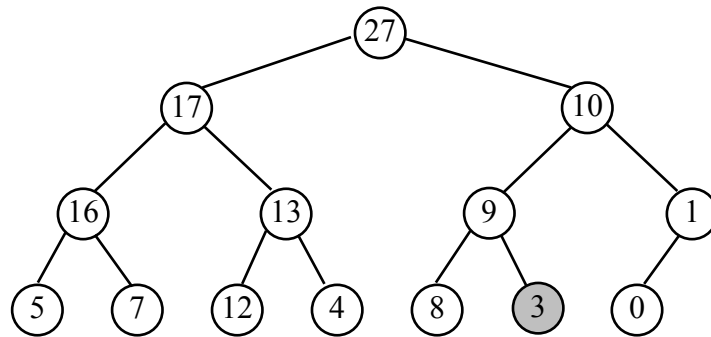
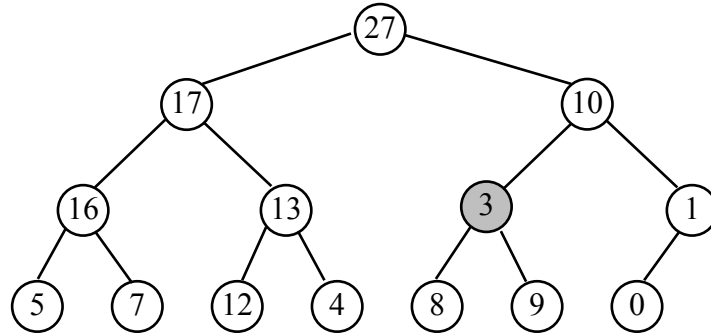
The sequence is not a max-heap, because 7 is a child of 6, but $7 > 6$.

Problem 3 (6.2-1):

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

Solution 3:





Problem 4 (6.2-6):

Show that the worst-case running time of MAX-HEAPIFY on a heap of size n is $\Omega(\lg n)$. (*Hint:* For a heap with n nodes, give node values that cause MAX-HEAPIFY to be called recursively at every node on a path from the root down to a leaf.)

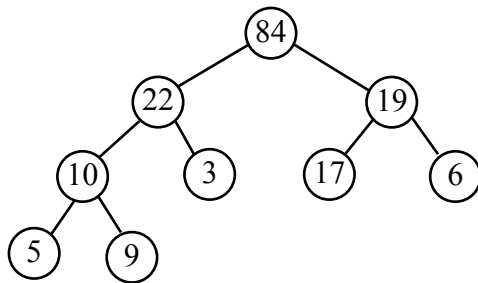
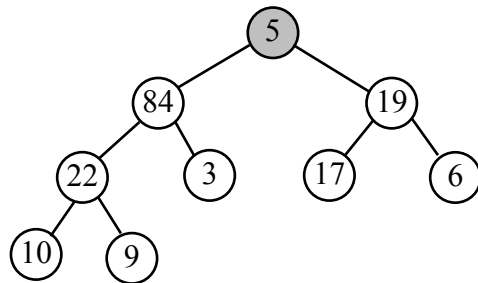
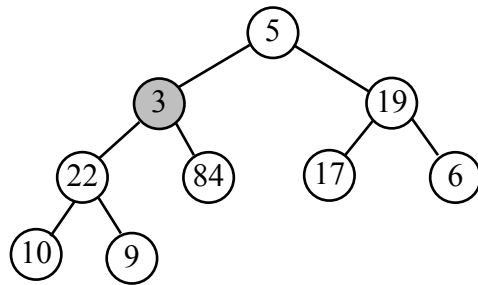
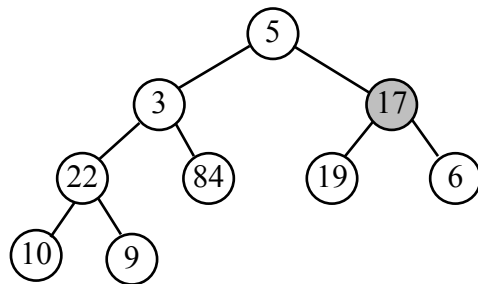
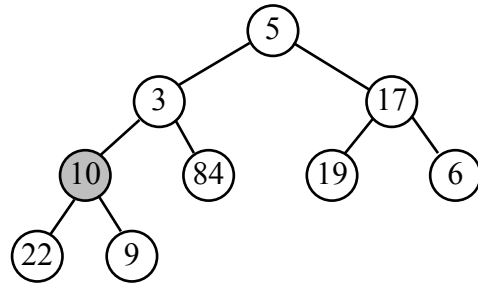
Solution 4:

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, until 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. 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 $\Omega(\lg n)$.

Problem 5 (6.3-1):

Using Figure 6.3 as a model, illustrate the operation of BUILD-MAX-HEAP on the array $A = \langle 5, 3, 17, 10, 84, 19, 6, 22, 9 \rangle$.

Solution 5:



Problem 6 (6.4-4):

Show that the worst-case running time of HEAPSORT is $\Omega(n \lg n)$.

Solution 6:

We can see that HEAPSORT is a comparison-type sort, and we have shown that all comparison-type sorts must be $\Omega(n \lg n)$.

Problem 7 (6.5-7):

The operation $\text{HEAP-DELETE}(A, i)$ deletes the item in node i from heap A . Give an implementation of HEAP-DELETE that runs in $O(\lg n)$ time for an n -element max-heap.

Solution 7:

```
HEAP-DELETE( $A, i$ )
   $A[i] \leftrightarrow A[\text{length}(A)]$ 
   $\text{length}(A) \leftarrow \text{length}(A) - 1$ 
  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 time $O(\lg n)$, so this algorithm runs in $O(\lg n)$ time.