6-1. ***Building a heap using insertion***

We can build a heap by repeatedly calling MAX-HEAP-INSERT to insert the elements into the heap. Consider the following variation on the BUILD-MAX-HEAP procedure:

```
BUILD-MAX-HEAP'(A)
  A.heap-size = 1
  for i = 2 to A.length
      MAX-HEAP-INSERT(A, A[i])
```

a. Do the procedures BUILD-MAX-HEAP and BUILD-MAX-HEAP′ always create the same heap when run on the same input array? Prove that they do, or provide a counterexample.

b. Show that in the worst case, BUILD-MAX-HEAP′ requires $\Theta(n \lg n)$ time to build an n-element heap.

Answer.

a. No they don't. As a counterexample, Figure 1 shows a max-heap BUILD-MAX-HEAP builds on

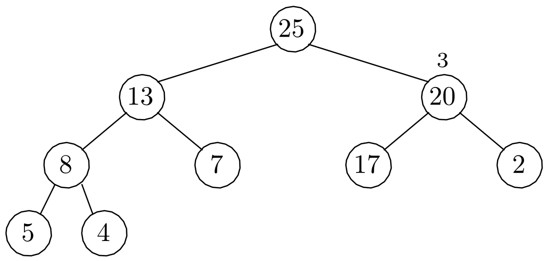

Figure 1. The max-heap BUILD-MAX-HEAP builds on array $A = \langle 5, 13, 2, 25, 7, 27, 20, 8, 4\rangle$.

array $A = \langle 5, 13, 2, 25, 7, 17, 20, 8, 4\rangle$, while BUILD-MAX-HEAP′ builds a different max-heap on the same array, shown in Figure 2. Notice that children of node of index 3 are different in two figures.

b. An upper bound of $O(n \lg n)$ time follows immediately from there being $n-1$ calls to MAX-HEAP-INSERT, each taking $O(\lg n)$ time. Form a lower bound of $\Omega(n \lg n)$, consider the case in which the input array is given in strictly increasing order. Each call to MAX-HEAP-INSERT causes HEAP-INCREASE-KEY to go all the way up to the root. Since the depth of node i is $\lfloor \lg i \rfloor$, the total time is

$$\begin{aligned}
\sum_{i=1}^{n} \Theta(\lfloor \lg i \rfloor) &\geq \sum_{i=\lceil n/2 \rceil}^{n} \Theta(\lfloor \lg \lceil n/2 \rceil \rfloor) \\
&\geq \sum_{i=\lceil n/2 \rceil}^{n} \Theta(\lfloor \lg (n/2) \rfloor) \\
&= \sum_{i=\lceil n/2 \rceil}^{n} \Theta(\lfloor \lg n - 1 \rfloor) \\
&\geq n/2 \cdot \Theta(\lg n) \\
&= \Omega(n \lg n)
\end{aligned}$$

Email address: informlarry@gmail.com

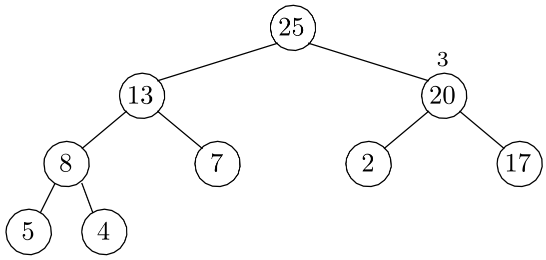

Figure 2. The max-heap BUILD-MAX-HEAP′ builds on array $A = \langle 5, 13, 2, 25, 7, 27, 20, 8, 4 \rangle$.

In the worst case, therefore, BUILD-MAX-HEAP′ requires $\Theta(n \lg n)$ time to build an n-element heap.