Making a Binary Heap from a List

CSE 30331/34331

Fall 2015 (version 1)

To initially build a binary heap from a list of \( n \) elements, we could start with an empty heap and then push each element. Equivalently, copy all the elements into the heap, in any order. Then, working top-down, reheapify-up each node. Since the reheapify-up operation takes \( O(\log n) \) time and there are \( n \) elements, this takes \( O(n \log n) \) time.

But there is a faster way, which is used by \texttt{std::priority\_queue} and \texttt{std::make\_heap}. Copy all the elements into the heap, in any order. Then, working bottom-up, reheapify-down each node. How is this any faster? It would seem that the reheapify-down operation takes \( O(\log n) \) time and there are \( n \) elements, so this takes \( O(n \log n) \) time.

A more careful analysis shows that it actually takes \( O(n) \) time. Intuitively, it’s because if we reheapify-up, the biggest levels have the longest distance to travel, whereas if we reheapify-down, the biggest levels have the shortest distance to travel.

Let \( h = \lfloor \log n \rfloor \), the height of the tree (\( h = 0 \) means just a root node).

\[
\begin{align*}
1 \text{ element at height } h &= 2 \\
2 \text{ elements at height } h - 1 \\
\leq 4 \text{ elements at height } 0
\end{align*}
\]

There is 1 element at height \( h \) (the root), 2 elements at height \( h - 1 \), and so on down to height 0 (the bottom level). In general there are \( 2^{h-k} \) elements at height \( k \) (where \( 0 \leq k \leq h \)). And an element at height \( k \) takes at most \( k \)

\[1\text{Under certain assumptions, this can be shown to be average-case linear-time, but the algorithm presented next is worst-case linear-time.}\]
operations to bubble down. So the total number of operations is at most

\[
T(n) \leq \sum_{k=0}^{h} 2^{h-k} k
\]

\[
= 2^h \sum_{k=0}^{h} \frac{k}{2^k}
\]

\[
\leq n \sum_{k=0}^{h} \frac{k}{2^k}.
\]

To evaluate the summation, we need a trick (which you are not responsible for on the exam!). Let \( x = \frac{1}{2} \). Then we have

\[
\sum_{k=0}^{h} \frac{k}{2^k} = \sum_{k=0}^{h} kx^k
\]

\[
\leq \sum_{k=0}^{\infty} kx^k
\]

\[
= x \sum_{k=0}^{\infty} kx^{k-1}
\]

\[
= x \frac{d}{dx} \sum_{k=0}^{\infty} x^k \quad \text{(the trick)}
\]

\[
= x \frac{d}{dx} \frac{1}{1-x}
\]

\[
= x \frac{1}{(1-x)^2}
\]

\[
= 2.
\]

So the total number of operations is at most 2n. So building a heap takes time \( \mathcal{O}(n) \).