Making a Binary Heap from a List

CSE 30331/34331

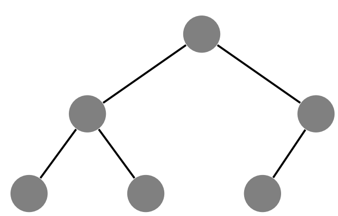
Fall 2015 (version 1)

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 $\mathcal{O}(\log n)$ time and there are n elements, this takes $\mathcal{O}(n \log n)$ time.[1]

But there is a faster way, which is used by `std::priority_queue` and `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 $\mathcal{O}(\log n)$ time and there are n elements, so this takes $\mathcal{O}(n \log n)$ time.

A more careful analysis shows that it actually takes $\mathcal{O}(n)$ 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 $h = \lfloor \lg n \rfloor$, the height of the tree ($h = 0$ means just a root node).



1 element at height $h = 2$

2 elements at height 1

≤ 4 elements at height 0

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] 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

$$\begin{aligned}
T(n) &\leq \sum_{k=0}^{h} 2^{h-k} k \\
&= 2^h \sum_{k=0}^{h} \frac{k}{2^k} \\
&\leq n \sum_{k=0}^{h} \frac{k}{2^k}.
\end{aligned}$$

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

$$\begin{aligned}
\sum_{k=0}^{h} \frac{k}{2^k} &= \sum_{k=0}^{h} k x^k \\
&\leq \sum_{k=0}^{\infty} k x^k \\
&= x \sum_{k=0}^{\infty} k x^{k-1} \\
&= x \frac{\mathrm{d}}{\mathrm{d}x} \sum_{k=0}^{\infty} x^k \qquad \text{(the trick)} \\
&= x \frac{\mathrm{d}}{\mathrm{d}x} \frac{1}{1-x} \\
&= x \frac{1}{(1-x)^2} \\
&= 2.
\end{aligned}$$

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