Build Heap: Insertion vs Bottom-Up (Runtime Proof)

Casey Li

May 23, 2018

Insertion (topdown) - insert, bubble, repeat

Assuming max-heap (biggest items at top):

Best case: θ(n)
Worst case: θ(nlogn)
Average case: θ(n)

Best case proof

Items in descending order. Biggest are inserted first.
Insertion occurs n times.
Each one bubbles exactly 0 times.

Worst case proof

Items come in ascending order.
Insert occurs n times.
For each insertion, the item must bubble to the top of the tree.
At any given insertion of item i, the height of the tree is $\lfloor logn \rfloor$.
Total nlogn times.

Average proof

Upper bound, O(n)

$1 + 2 + 4 + 8 = 15$.

For a complete binary tree of 15 items (almost 15):

8/15 items bubble 0 times ($\approx 8/16 \approx 1/2$, strictly < 1)
4/15 items bubble 1 times ($\approx 4/16 \approx 1/2^2$, $< 1/2$)
2/15 items bubble 2 times ($\approx 2/16 \approx 1/2^3 < 1/4$)
1/15 items bubble 3 times ($\approx 1/16 \approx 1/2^4 < 1/8$)

P(bubble i times): $\frac{i}{2^{i+1}}$, which is strictly less than $\frac{i}{2^i}$

Expected bubbling of any given item is less than (upper bound):

$$\sum_{i=0}^{\lfloor logn \rfloor} \frac{i}{2^i}$$

This sum converges to 2 as i approaches infinity, so this particular sum is strictly less than 2. In other words, it is constant, θ(n).

We then have upper bound on the average = 2n = O(n).

Lower bound, Ω(n)

So now we just need to pick a case to prove that in the best case, the average running time is Ω(n).

Easy. We already did this. Best case, they are inserted in descending order, each one bubbles 0 times, so we still have Ω(n).

Having proved both the upper and lower bounds, we can conclude that the average case runs in $\theta(n)$.

Bottom-up: fill all, then fix subheaps

In any given heap,

1/2 of the items are leaves, no fixing
$1/2^2$ of the items are heaps of height 1 (max bubble 1 level)
$1/2^3$ of the items are heaps of height 2
$1/2^4$ of items are heaps of height 3

In general, $1/2^{h+1}$ items will bubble up at most h times.

Worst case # of bubbles in a filled binary tree is:

$$n \sum_{i=0}^{\lfloor logn \rfloor} \frac{h}{2^{h+1}} = \frac{n}{2} \sum_{i=0}^{\lfloor logn \rfloor} \frac{h}{2^h}$$

The summation converges to 2 as n goes to infinity, so this sum is strictly less than n. So worst case, this runs in θ(n) time.