CSCA48 Winter 2015

Week 9 - Building Heaps

Anna Bretscher

March 9, 2015

Building Heaps

- So far, we have built a *heap* by *inserting* all the elements.
- There is a better way using *heapify*...

Building Heaps

Q. Given an array *A* of *elements*, how can we build a *heap* efficiently *in-place*?

A. We call *heapify* on the subtrees of height 1, then height 2, all the way to height *h* of the heap.

Consider an array:

$$[8, 9, 2, 10, 3, 7, 5, 1, 4, 6]$$

Building Heaps

[8, 9, 2, 10, 3, 7, 5, 1, 4, 6]

Let's build the heap:

Building Heaps

L = [8, 9, 2, 10, 3, 7, 5, 1, 4, 6]

Q. Which values in this *list* represent *leaf* nodes in a heap?

A. 7,5, 1, 4, 6

Q. Considering only the *list*, how do we know which values are *leaf* nodes?

A. Consider the *last leaf* or the value at index $n = len(L) - 1$.
Then it's parent is $\lfloor \frac{n-1}{2} \rfloor$.

Q. What does this tell about the positions of the leaf nodes?

A. They are in positions $(\lfloor \frac{n-1}{2} \rfloor + 1)$ to n or $\lfloor \frac{n+1}{2} \rfloor$ to n.

Building Heaps

- We call *heapify* on the index $\lfloor \frac{n-1}{2} \rfloor$.
- And again on all indices from $\lfloor \frac{n-1}{2} \rfloor$ to *0*.
- Each call to *heapify* creates a *sub-heap*.
- We could write a proof by *induction* that the last call creates a valid heap.

Complexity

Q. How *many* calls do we make to heapify?

A. From $0 \ldots \lfloor \frac{n-1}{2} \rfloor$ so roughly $\frac{n}{2}$.

Q. How many *steps* does each call make?

A. No more than $c \cdot \log n$. Why?

We do at most a multiple of $n \log n$ steps.

Can we do better?

Complexity Take 2

- We call heapify on each subtree of *height* ≥ 1.
- heapify runs in time *proportional* to the *height* of that subtree.
- We need to know how many subtrees of each height we have:

Height	Max Number of Trees
0	$\frac{n}{2}$
1	$\frac{n}{2^2}$
2	$\frac{n}{2^3}$
3	$\frac{n}{2^4}$
$\vdots$	$\vdots$
h	$\frac{n}{2^{n+1}}$

Complexity Take 2

Height	Max Number of Trees (for n nodes)
0	$\frac{n}{2}$
1	$\frac{n}{2^2}$
2	$\frac{n}{2^3}$
$\vdots$	$\vdots$
h	$\frac{n}{2^{h+1}}$

$$
\begin{aligned}
\text{Total Number of Steps} &= \sum_{h=1}^{\log n} c \cdot h \times \text{ (number of subtrees of height } h\text{)} \\
&= \sum_{h=1}^{\log n} c \cdot h \times \frac{n}{2^{h+1}} = \sum_{h=1}^{\log n} c \cdot n \times \frac{h}{2^{h+1}} \\
&= \frac{cn}{2} \sum_{h=1}^{\log n} \frac{h}{2^h} = ??
\end{aligned}
$$

SURPRISE!

$$\text{Total Steps} \quad = \quad \frac{cn}{2}\sum_{h=1}^{\log n}\frac{h}{2^h} = ??$$

Fact:

$$\sum_{h=1}^{\infty}\frac{h}{2^h} \leq 2$$

Can you prove this?

$$\begin{aligned}
\text{Total Steps} \quad &= \quad \frac{cn}{2}\sum_{h=1}^{\log n}\frac{h}{2^h} \\
&\leq \quad \frac{cn}{2}\sum_{h=1}^{\infty}\frac{h}{2^h} \\
&\leq \quad \frac{cn}{2}\cdot 2 = cn
\end{aligned}$$