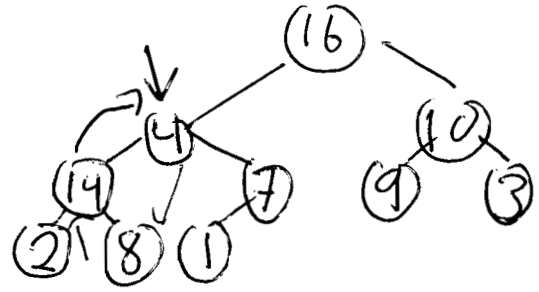
6.006 Lecture 9 Heapsort

- ☐ Heaps, Heapify (review)
- ☐ Heapsort Outline
- ☐ Build-Heap (1st step of Heapsort)
- ☐ Heapsort
- ☐ Priority Queues



Review: Heapify (A, i)

Goal: Fix tree rooted at i to become a heap

Assume: Trees rooted at left(i), right(i) already heaps

May have one violation. $A[i]$ be not be the largest among $A[i]$, $A[\text{left}(i)]$, $A[\text{right}(i)]$

Note: All of A becomes a heap if no other violations
(Of course $A[\text{Parent}(i)] \geq$ the largest of $A[i]$, $A[\text{left}(i)]$, $A[\text{right}(i)]$ is necessary)

Time $= O(h)$ $h = \text{height}(i)$
i.e. at most constant times h

Better notation $h\,O(1)$

$$h\,O(1) \leq \lg n\, O(1) = O(\lg n)$$

Heapsort Outline

1) Input array $A = \boxed{4 \mid 1 \mid 3 \mid 2 \mid 16 \mid 9 \mid 10 \mid 14 \mid 8 \mid 7}$

NOT A HEAP

2) Build-Heap

Idea: Heapify bottom to top until a heap

$\boxed{16 \mid 14 \mid 10 \mid 8 \mid 7 \mid 9 \mid 3 \mid 2 \mid 4 \mid 1}$

Heap

Time: Careful Analysis $O(n)$

3) Extract Max

Exchange with last leaf

$\boxed{1 \mid 14 \mid 10 \mid 8 \mid 7 \mid 9 \mid 3 \mid 2 \mid 4 \mid 16}$

not a heap

max here

4) Heapify Remaining elements

$\boxed{14 \mid 8 \mid 10 \mid 4 \mid 7 \mid 9 \mid 3 \mid 2 \mid 1}$ 16

Heap

5) Repeat 3 + 4 until done

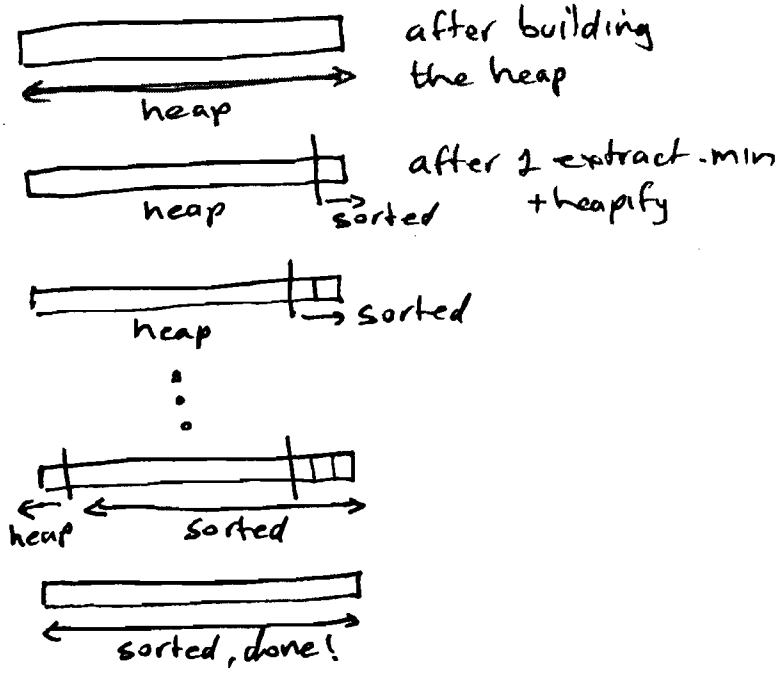
3) $\boxed{1 \quad 8 \quad 10 \quad 4 \quad 7 \quad 9 \quad 3 \quad 2}$ $\boxed{14 \mid 16}$

4) 10 8 9 4 7 1 3 2 14 16

3) 2 8 9 4 7 1 3 $\boxed{10 \mid 14 \mid 16}$

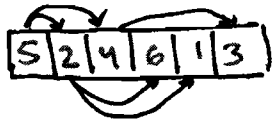
4) 9 8 3 4 7 1 2 10 14 16

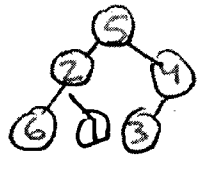
- The heap area grows and the sorted prefix shrinks



How to build a heap

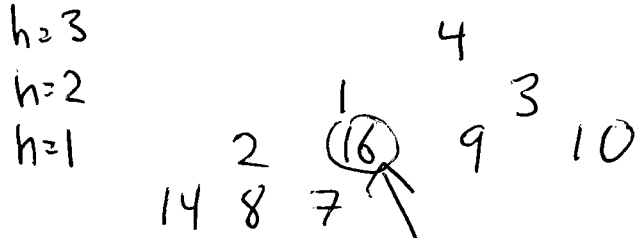
- fill in the array in arbitrary order





- clearly not a heap...
- But the leaves are roots of heaps
- Lets fix the parents of the leaves; run heapify

Build Heap Example



Start here (last parent) & heapify then move to next parent, ...

At height h work is $hO(1)$
and certainly $< 2\lceil n/2^{h+1} \rceil$ nodes

Total work $\leq O(n) \cdot \underbrace{\sum_{h=1}^{\infty} \frac{2h}{2^{h+1}}}_{1}$

Total work $\leq O(n)$

Heapsort Example on Board (see p137)

Analysis

$O(n) + n\,O(\lg n)$

$\Rightarrow O(n \lg n)$

Priority Queues

An abstract data type (many implementations) useful both in algorithms (e.g., heapsort) and directly in applications.

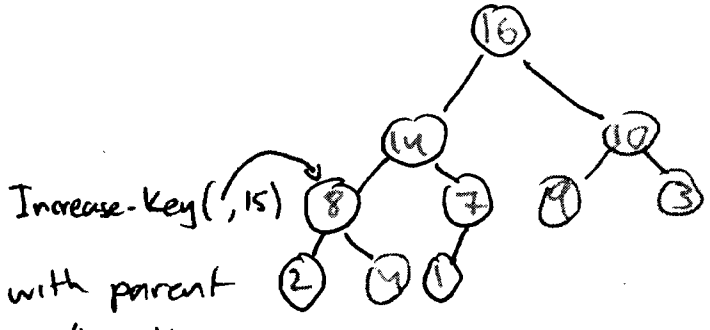
Insert (v)

Maximum()

Extract-Max()

Increase-Key(x, v) x points to an element in PQ

(can also have <u>min</u> versions of PQ)



swap with parent until smaller than parent.

Can we also support Decrease-Key in the heap implementation of this ADT?

Yes, with heapify?