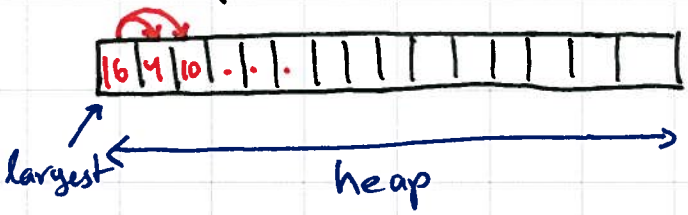
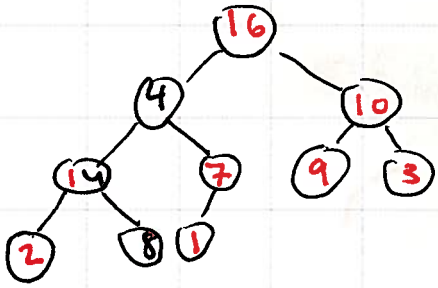
6.006 Lecture 9: Sorting II

- Heapsort, heaps, priority queues 6.6
- Decision tree lower bounds 8.1
  (not covered in the lecture on Thursday)

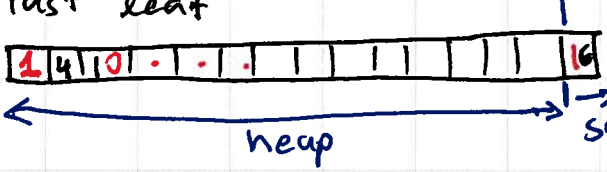
Heapsort (+ some review of heaps)

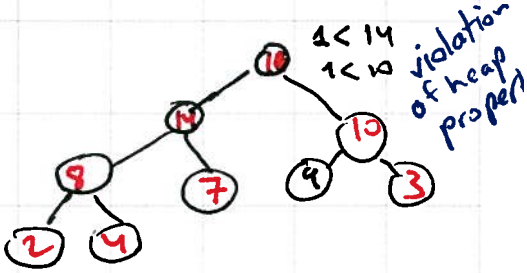
- Arrange array elements as a heap (we didn't see how yet)



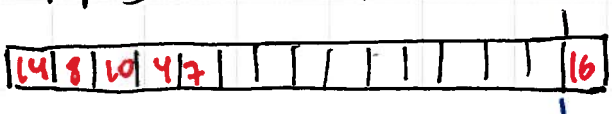


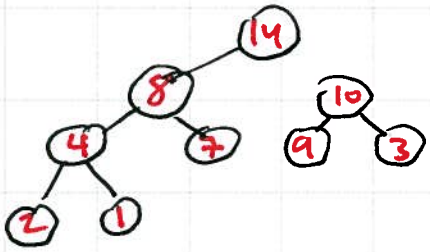
- Extract-Min and put at end; exchange max with last leaf





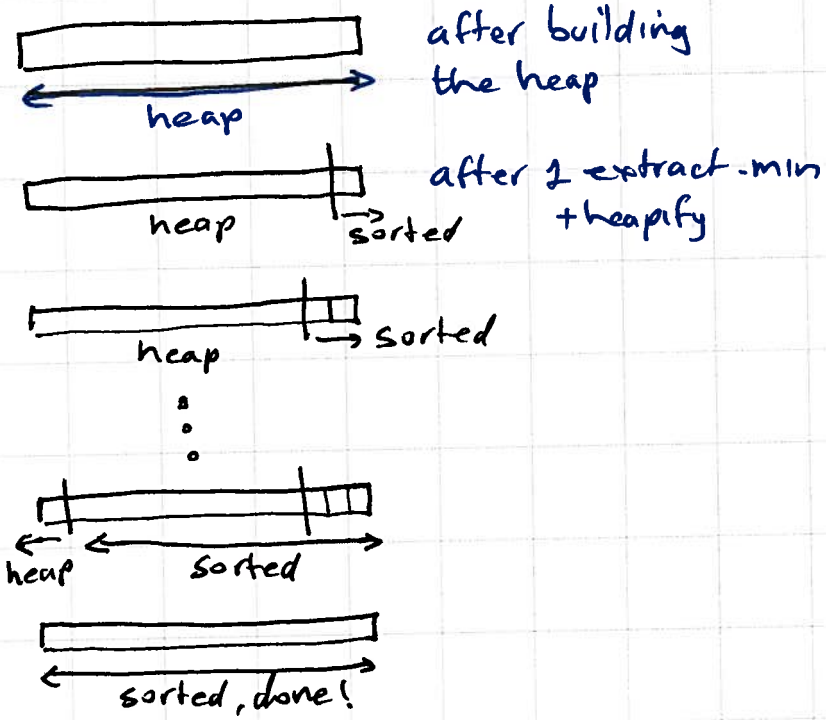
- Heapify (fix heap)





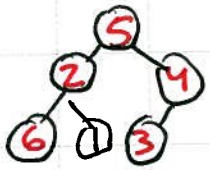
- Repeat

- The heap area grows and the sorted prefix shrinks

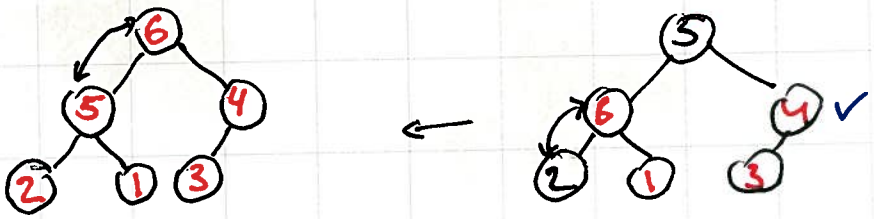


How to build a heap

- 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



- How long does it take to build a heap?

A naive answer: $O(\lg n)$ per Heapify $\Rightarrow O(n \lg n)$

A more detailed analysis:

$\approx \frac{n}{4}$ parents of leaves, 2 comparisons each $\Rightarrow \frac{n}{2}$

$\frac{n}{8}$ grandparents, 3 comparisons or less $\Rightarrow \frac{3}{8}n$

$\frac{4}{16}n = \frac{n}{4}$

$\vdots$

$\frac{\lg n}{2^{\lg n}}n$

$\sum_{h=0}^{\infty} \frac{h}{2^h} = 2 \Rightarrow$ Build-Heap costs $O(n)$

Priority Queues

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

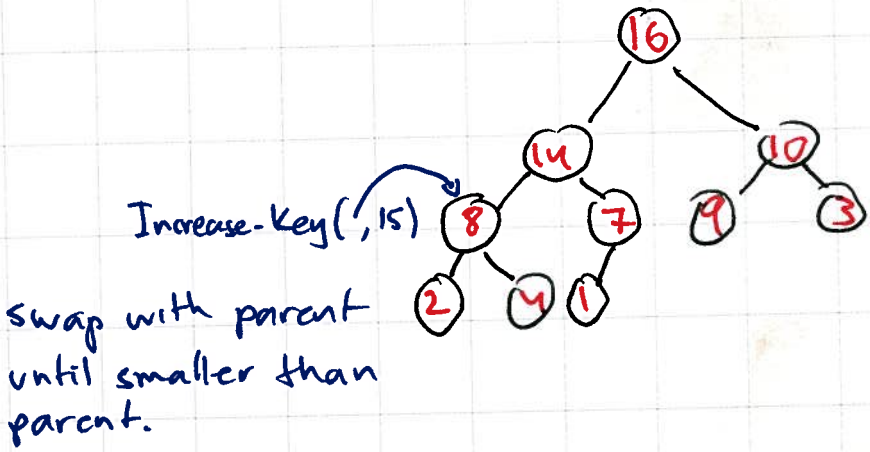
Insert (x)

Maximum()

Extract-Max()

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

(can also have min versions of PQ)



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

Yes, with heapify?