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 the largest among A[i], A[left(i)], A[right(i)]

Note: All of A becomes a heap if no other violations

(Obviously A[Parent(i)] > the largest of A[i], A[left(i)], A[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 \( A = [4, 1, 3, 2, 16, 9, 10, 14, 8, 7] \)

2) Build-Heap
   Idea: Heapify bottom to top until a heap
   \( 16, 14, 10, 8, 7, 9, 3, 2, 4, 1 \)
   \( \text{Heap} \)

   Time: Careful Analysis \( O(n) \)

3) Extract Max
   Exchange with last leaf

4) Heapify Remaining elements

5) Repeat 3 + 4 until done
- The heap area grows and the sorted prefix shrinks

- After building the heap
- After 1 extract-min and heapify

- Sorted
- Sorted
- Sorted, done!

How to build a heap

- Fill in the array in arbitrary order

- Clearly not a heap...
- But the leaves are roots of heaps
- Let’s 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 \( < \frac{2^{h+1} - 2}{2} \) nodes

Total work \( \leq O(n) \cdot \sum_{h=1}^{\log_2 n} \frac{2^h}{2^{h+1}} \)

Total work \( \leq O(n) \)

Heapsort Example on Board (see p.137)

Analysis:

\[ O(n) + nO(\log n) \]

\( \Rightarrow \) \( O(n\log n) \)
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)

Increase-Key(15)

swap with parent until smaller than parent.

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

Yes, with heapify?