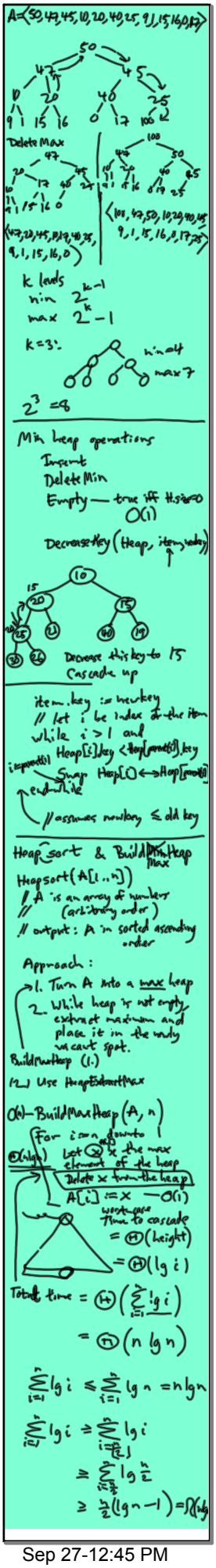


Sep 27-12:45 PM

BuildMaxHeap

// input A[1..n] arbitrary order

// output A[1..n] in max heap order

One approach

Let H be empty maxheap

Insert each element 1 by 1 into H.

Inserting (i+1)st element into a heap of size i

Time to insert = $\Theta(\lg(\text{size}))$ (depth, worst case)

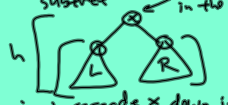
Total time to get H:

$$\sum_{i=1}^{n} \lg i = \Theta(n \lg n)$$

Better approach:

Build Heap from bottom up:

At each step: subtree (same node in the array)



just cascade x down into L or R (according to which has the bigger root)

Takes time h.

For $i := \lfloor \frac{n}{2} \rfloor$ downto 1 (highest index with a child)

// precondition

// subtrees rooted at

// left(i) and right(i)

// are both in max heap order

→ cascade A[i] down

// postcondition: subtree rooted

// at i is in max heap order

Running time (worst-case)

Time to cascade x down is $\Theta(\text{height of } x)$

2^i many nodes on level i, all with height lg n − i

Total time T(n)

$$= \sum_{i=0}^{\lg n} \underbrace{2^i}_{\text{(level i)}} \underbrace{(\lg n - i)}_{\text{height}} \qquad h = \lg n - i$$

$$= \sum_{h=0}^{\lg n} 2^{\lg n - h}(h)$$

$$= \sum_{h=0}^{\lg n} h\left(2^{\lg n} 2^{-h}\right)$$

$$= n \sum_{h=0}^{\lg n} h 2^{-h} = n \underbrace{\sum_{h=0}^{\lg n} h\left(\frac{1}{2}\right)^h}_{\sum i r^i}$$

$$\leq n \underbrace{\sum_{h=0}^{\infty} h\left(\frac{1}{2}\right)^h}_{\text{finite}}$$

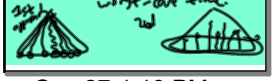
$$= O(n) \quad \text{upper bound}$$

lower bound $= \Omega(n)$ because we do something $\frac{n}{2}$ times

(Recall: For $i := \frac{n}{2}$ downto 1 ...)

∴ Conclusion:

BuildMaxHeap is $\Theta(n)$ worst-case time.



Sep 27-1:18 PM