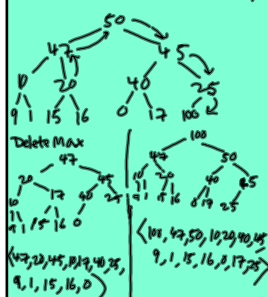
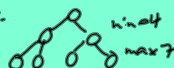


$A = \langle 50, 42, 45, 10, 20, 40, 25, 9, 15, 16, 0, 17 \rangle$



k levels
min 2^{k-1}
max $2^k - 1$

$k=3$:



$$2^3 = 8$$

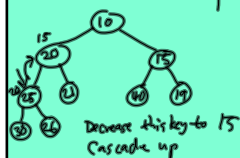
Min heap operations

Insert

Delete Min

Empty — true iff $H.size == 0$
 $O(1)$

DecreaseKey (Heap, item, value)



item.key := newkey
// let i be index of the item
while $i > 1$ and
 $Heap[i].key < Heap[parent(i)].key$
isparent(i) Swap $Heap[i] \leftrightarrow Heap[parent(i)]$
endwhile
// assumes newkey $\leq$ old key

Heap sort & BuildMaxHeap

Heapsort($A[1..n]$)
// A is an array of numbers
// (arbitrary order)
// output: A in sorted ascending order

Approach:

1. Turn A into a max heap
2. While heap is not empty, extract maximum and place it in the only vacant spot.

BuildMaxHeap (1.)

(2.) Use HeapsortMax

$O(n)$ - BuildMaxHeap(A, n)

For $i = n$ downto 1
Let x be the max element of the heap
Delete x from the heap

$A[i] := x$ — $O(1)$
// worst case
// time to cascade
= $O(\text{height})$
= $O(\lg i)$

$$\text{Total time} = O\left(\sum_{i=1}^n \lg i\right)$$

$$= O(n \lg n)$$

$$\sum_{i=1}^n \lg i \leq \sum_{i=1}^n \lg n = n \lg n$$

$$\begin{aligned} \sum_{i=1}^n \lg i &\geq \sum_{i=\frac{n}{2}}^n \lg i \\ &\geq \sum_{i=\frac{n}{2}}^n \lg \frac{n}{2} \\ &\geq \frac{n}{2} (\lg n - 1) = \Omega(n \lg n) \end{aligned}$$

Build Max Heap

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

One approach

Let H be empty max heap
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 :

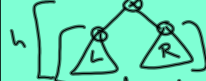
$$\sum_{i=1}^n \lg i = \Theta(n \lg n)$$

Better approach:

Build Heap from bottom-up:



At each step: some node subtree in the array



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

For $i = \lfloor \frac{n}{2} \rfloor$ down to 1

// precondition: subtree rooted at i is a heap
// subtrees rooted at $\text{left}(i)$ and $\text{right}(i)$
// are both in max heap order

$\rightarrow$ 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 $T(n)$

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

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

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

$$= n \sum_{h=0}^{\lg n} h 2^{-h} = n \sum_{h=0}^{\lg n} h \left(\frac{1}{2}\right)^h$$

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

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

lower bound = $\Omega(n)$

because we do something

$\frac{n}{2}$ times

(Recall: For $i = \frac{n}{2}$ dumb)

$\therefore$ Conclusion:

BuildMaxHeap is $\Theta(n)$

