COP3502 - 11/3/23

Binary Heaps / Priority Queues

Data Structure — Abstract Data Structure

Insert items

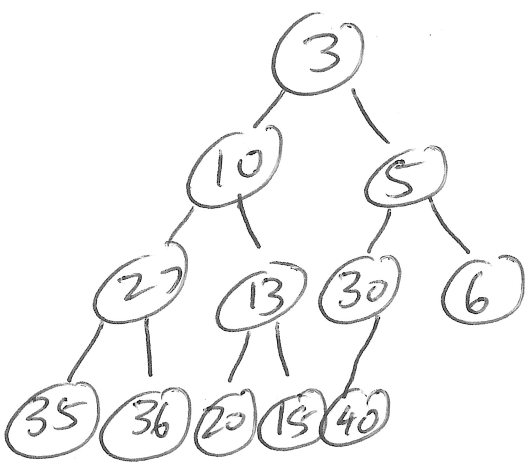
Delete Most Important - "Min"

O(lg n) time

n = # items

easy understand — easy code

2 Visuals: Tree Form, Array Form



$h = O(\lg n)$

0	1	2	3	4	5	6	7	8	9	10	11	12
	3	10	5	27	13	30	6	35	36	20	15	40

parent of node i is node i/2

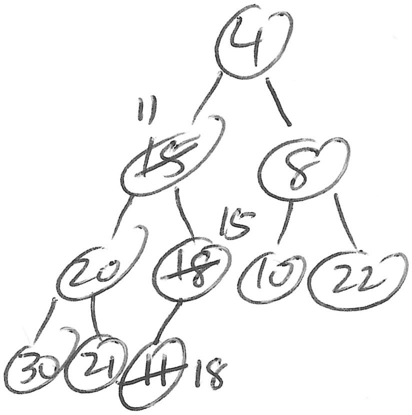
left child of node i is node 2*i.

right child of node i is node 2*i+1.

(1) all nodes in a subtree larger that root of the subtree

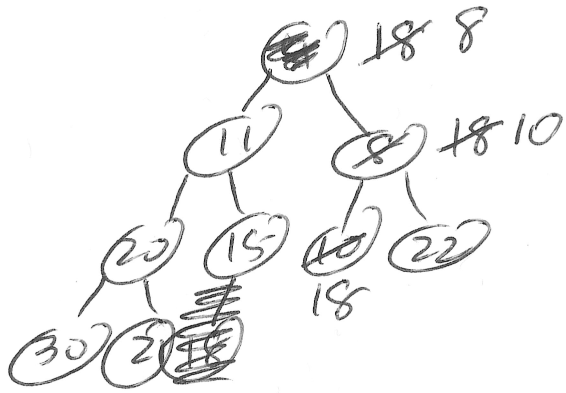
(2) Complete Binary Tree
all levels are fully filled in, except for last possible, even on last fill L→R.

Insert



1. Add item to next open pos.
2. Percolate Up

DeleteMin



1. Remove item top
2. Move item last spot to the vacated root node
3. Percolate Down

Heap Sort

1. Run Heapify $O(n)$
2. for (i=0; i<n; i++)
   arr[i] = deleteMin(heap)
   $O(n \lg n)$

$O(n \lg n)$ WORST AVG

Heapify / MakeHeap

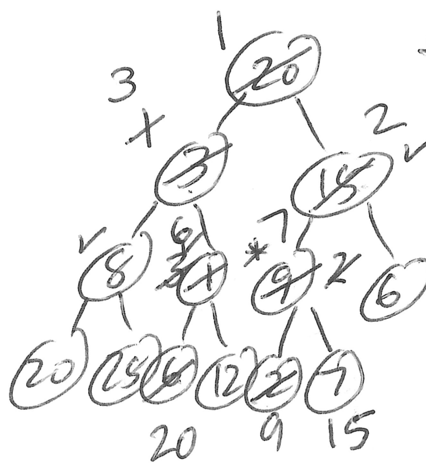
Given n values form a heap

Simple Strategy: Insert each item 1 by 1 into heap

run time ~

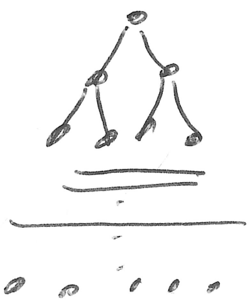
$\lg 1 + \lg 2 + \lg 3 + \dots \lg n$

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



for(i=n/2; i>0; i--)
percolateDown(i)

MakeHeap Run-Time



1
2
4
2^{h-1} item
2^h items

$n = 1+2+4+\cdots 2^h = 2^{h+1}-1$

2^{h-1} items 1 swaps
2^{h-2} items 2 swaps
2^{h-3} items 3 swaps
$\vdots$
2^{0} items h swaps

$$S = 1\times 2^{h-1} + \underline{2\times 2^{h-2}} + 3\times 2^{h-3} + \cdots \underline{\underline{h * 2^0}}$$

$$-\frac{S}{2} = \qquad 1\times 2^{h-2} + 2\times 2^{h-3} + \cdots (h-1)\times 2^0 + h\cdot\frac{1}{2}$$

$$\frac{S}{2} = 2^{h-1} + 2^{h-2} + 2^{h-3} + \cdots + 2^0 - \frac{h}{2}$$

$$S = \underbrace{2^h + 2^{h-1} + \cdots + 2} - h$$

$$S = \underbrace{2^{h+1} - 2} - h \quad , \quad n = 2^{h+1} - 1$$

$$= \boxed{n - 1 - h} \qquad O(n) \text{ time}$$