COP 3502 - 11/2/23

(1) Binary Heaps / Priority Queue (data structure) (abstract data structure)

(2) Live Code? Heap Problem?

Reg Queue - first in, first out

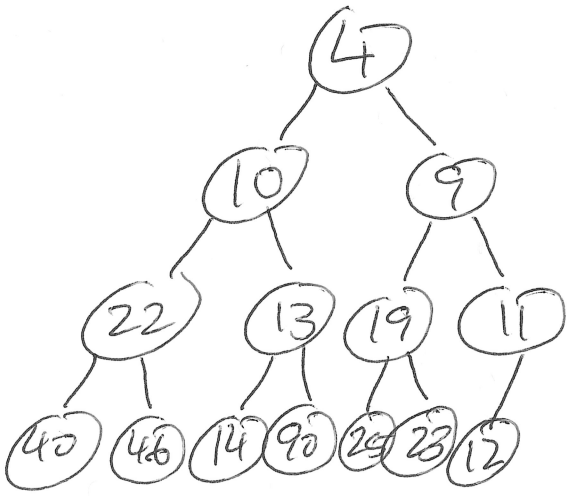
Priority Queue - whoever has "highest priority" gets removed first. → lowest numerical value. (Min Heaps)

- Insert
- DeleteMin

} O(lg n) time

Understand Coding Representations

2 Visualizations: Tree, Array



Rules for Valid Binary Heaps

(1) heap node property - all value underneath node (in its subtree) are greater than it.

(2) Structure has to be a complete binary tree. all levels but bottom completely filled in. Bottom must go L → R nothing missing

Two Subroutines which we'll use in Insert, DeleteMin

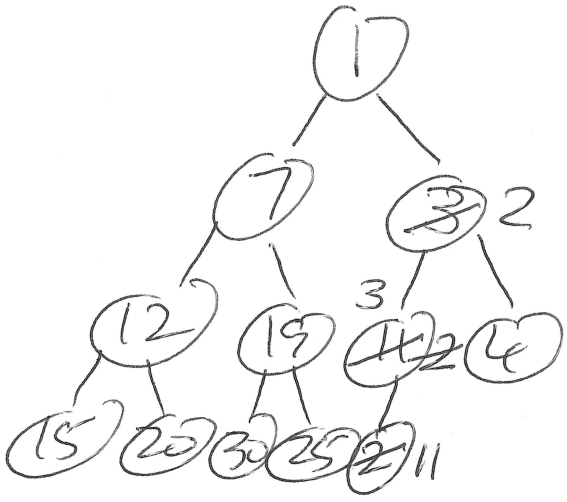
(1) PercolateUp
(2) PercolateDown

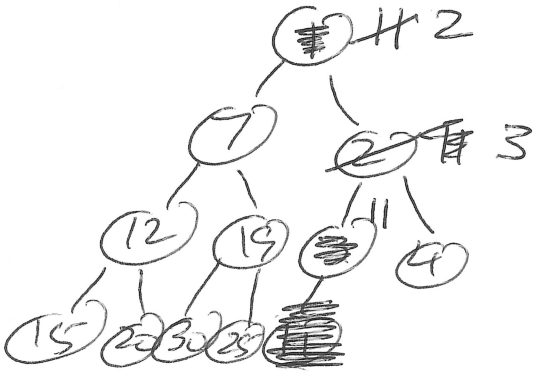
Insert

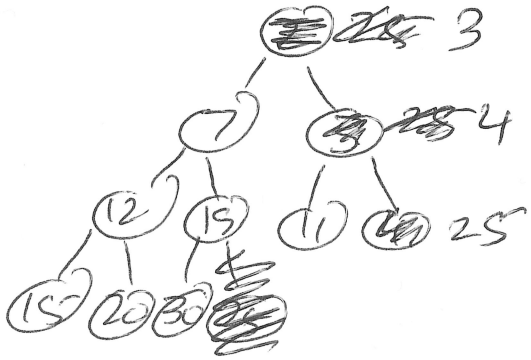
1. Place item "next" spot
2. PercolateUp

DeleteMin

1. Remove top item
2. Place "last" item in top spot.
3. PercolateDown





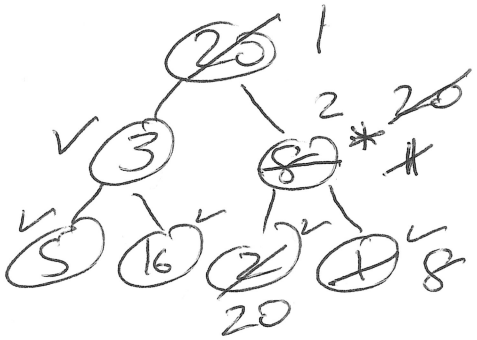


Heap Sort - Ver 1

1. Insert n items
2. for (i=0; i<n; i++)
   arr[i] = deleteMin(heap);

Ver 2

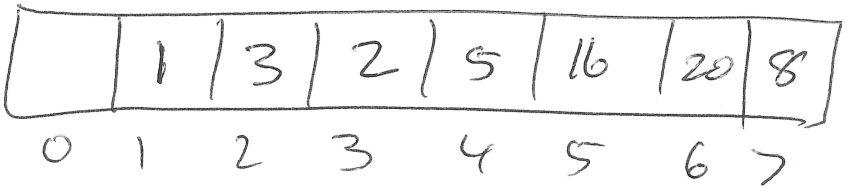
Unsorted values : 20, 3, 8, 5, 16, 2, 1



1. Copy values into heap
2. for (i=n/2; i>0; i--)
   percolateDown(heap, i);

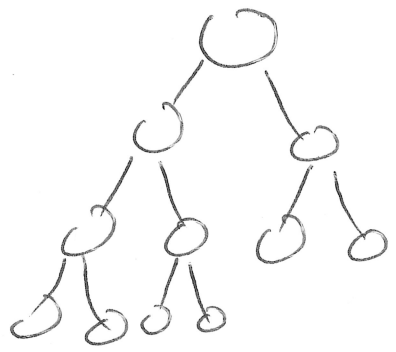
makeHeap
initializeHeap

how to store heap in an array:



root node index 1.
node i's left child index 2i
node i's right child index 2i+1
node i's parent index i/2.

Make Heap Run-Time Analysis



1
2
4
8
⋮
2^{h-1}

n nodes

$n = 2^{h+1} - 1$

2^h — no percolate down

2^{h-1} nodes — 1 swap

2^{h-2} nodes — 2 swaps

2^{h-3} nodes — 3 swaps

$S = \sum_{i=1}^{n} i \cdot 2^{h-i}$

↓

2^0 nodes — h swaps

$$S = \boxed{1 \times 2^{h-1}} + 2 \times 2^{h-2} + 3 \times 2^{h-3} + \cdots + (h-1) \times 2^1 + n \times 2^0$$

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

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

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

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

$$S = \boxed{n - 1 - h}$$

If we do n inserts, the last $\frac{n}{2}$ inserts will each potentially take $\lg n$ time. Total time (worst case) $\geq \frac{n}{2} \lg n$.