CSE 101 3-12-24

- SETs : if Response Rate ≥ 85%, I will raise all overall score by .5%.
- final exam : MON. 3/18 4:00-5:30 PM
- Pas : ext. by 2 days → Fri. 3/15 10PM.

Priority Queues

a P.Q. is an ADT with a finite set of elements called records

$$x = (\underbrace{\cdot, \cdot, \cdot}_{\text{satellite data}}, key) \in S$$

so

$$S = \{\ldots, x, \ldots\}$$

is a state.

Two Kinds • min Priority Queue

• max " "

- Operations

- Insert(S, x): inserts x into S

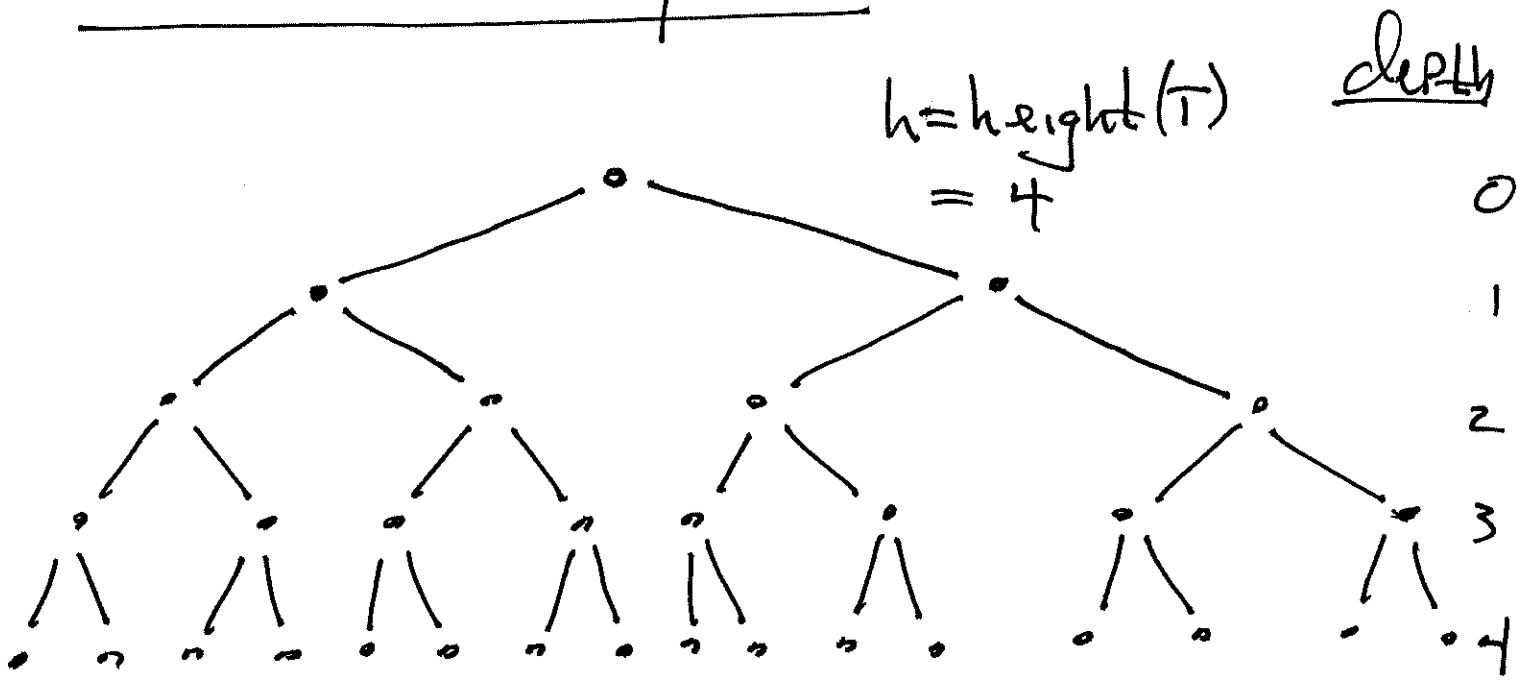
- Max(S): return record with largest key

- ExtractMax(S): delete and return record with largest key

- IncreaseKey(S, x, k): change x.key to k if k is larger than x.key, otherwise do nothing.

Trees!

Complete Binary tree (CBT)

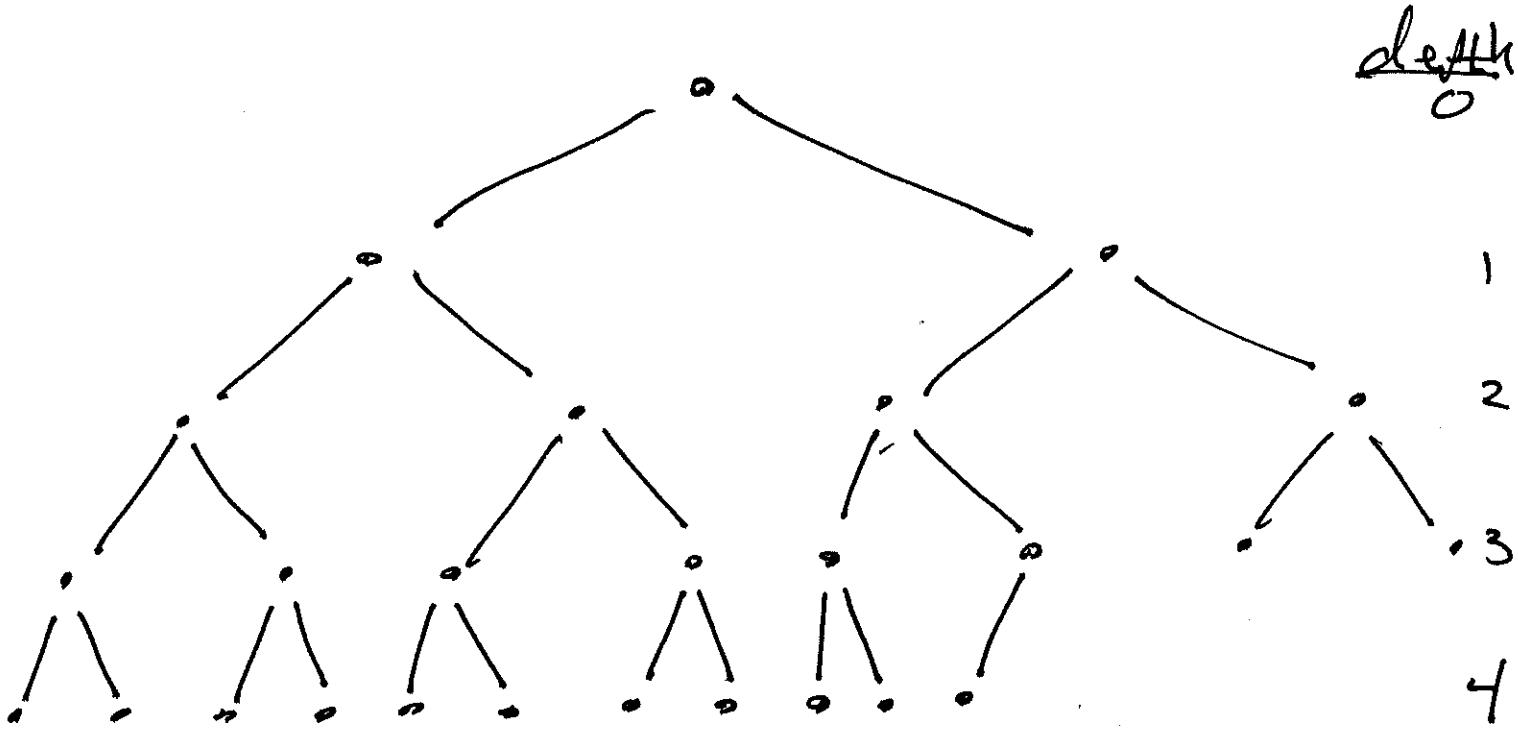


- every internal node has 2 children
- all leaves are at same depth

The # of nodes at depth d is 2^d.
So total # nodes is

$$n = \sum_{d=0}^{h} 2^d = \frac{2^{h+1}-1}{2-1} = 2^{h+1}-1$$

Almost complete Binary Tree (ACBT)



- filled at all levels, except possibly the last
- last level filled from left to right.

let an ACBT have n nodes and height h.

Then

$$2^h - 1 < n \leq 2^{h+1} - 1$$

$$\therefore \quad 2^h \leq n < 2^{h+1}$$

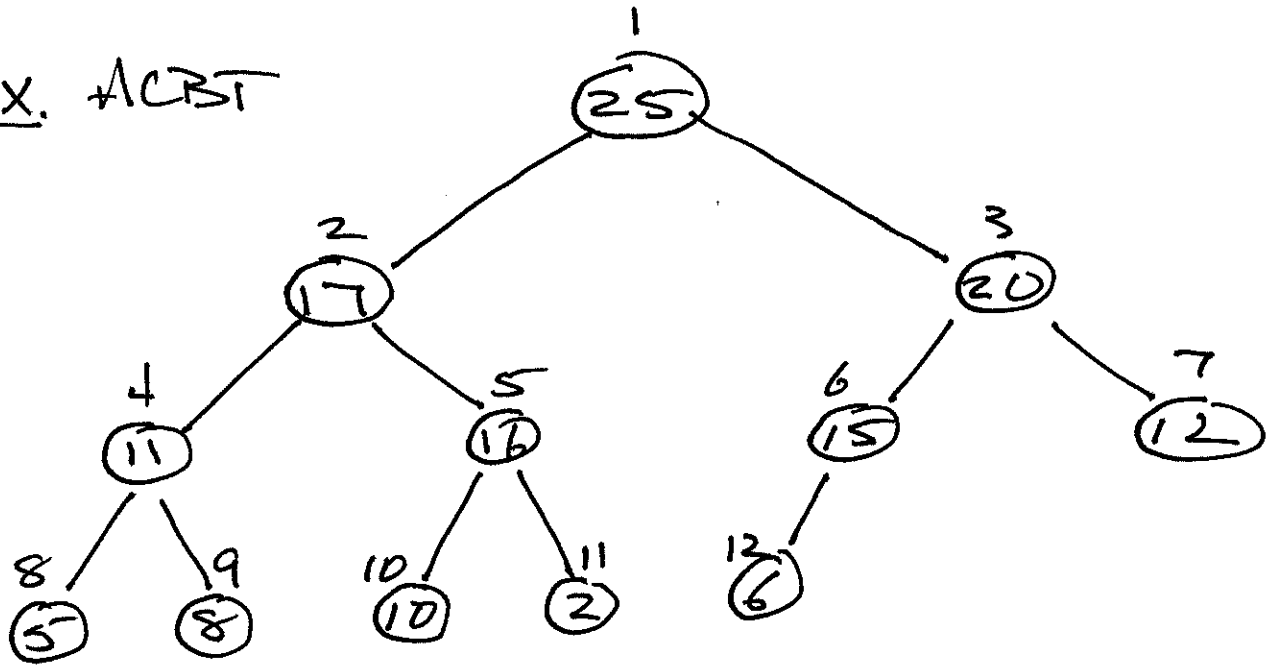
$$\therefore \quad h \leq \lg(n) < h+1$$

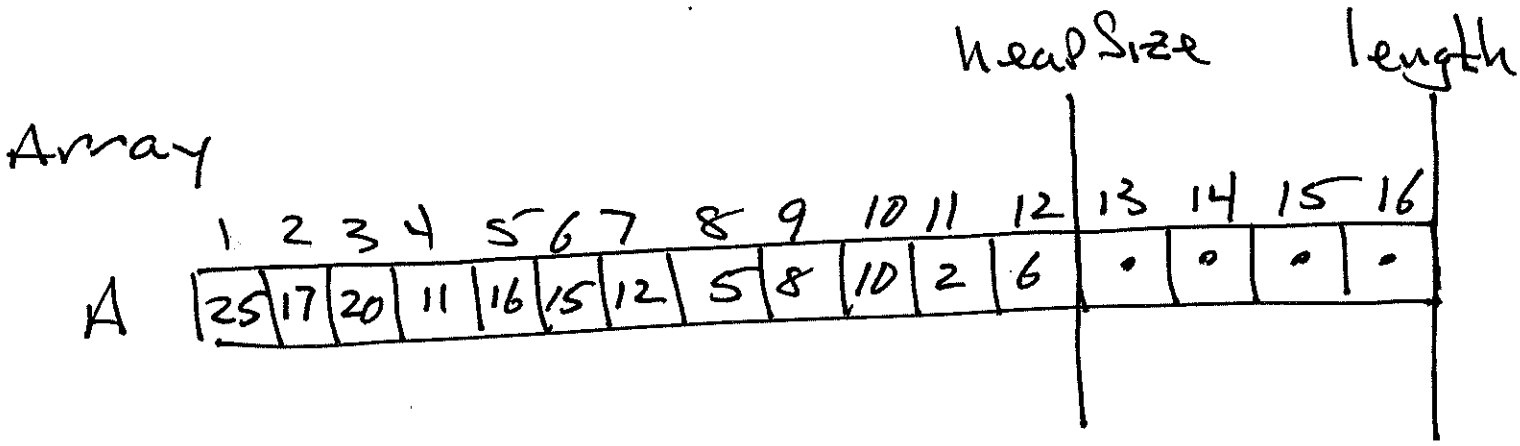
$$\therefore \quad \boxed{h = \lfloor \lg(n) \rfloor}$$

6.1 Heaps

A max (binary) heap is a data structure that represents a ACBT as an array.

Ex. ACBT





The array has 2 attributes

- length[A]
- heapSize[A]

We have 3 helper functions

$$\text{Parent}(i) = \left\lfloor \frac{i}{2} \right\rfloor \qquad (\text{for } i \geq 2)$$

$$\text{left}(i) = 2i$$

$$\text{right}(i) = 2i + 1$$

Heap Properties

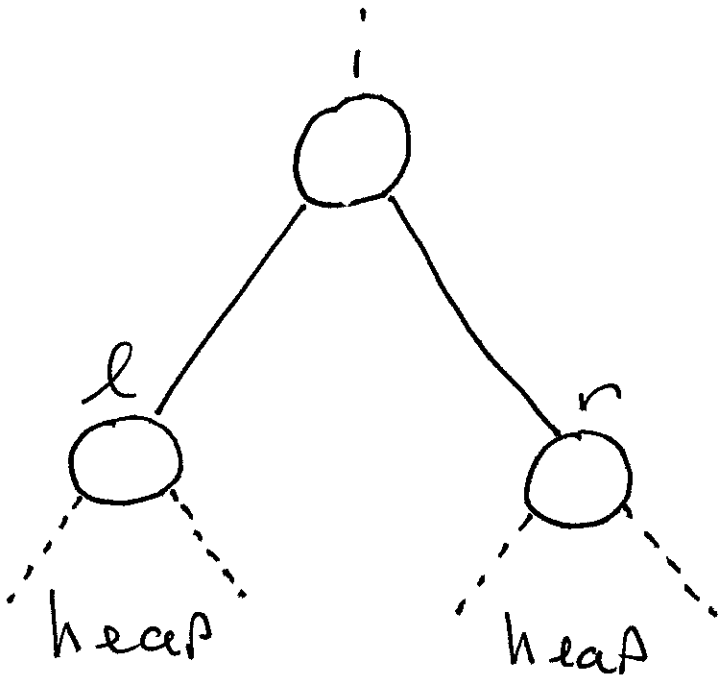
max-heap: $A[\text{Parent}(i)] \geq A[i]$ ✓

min-heap: $A[\text{Parent}(i)] \leq A[i]$

Heap operations

- Heapify
- BuildHeap
- Heapsort
- P.Q. OPS. . . . .

6.2 Heapify



Goal: establish heap property at index i.

see pseudo-code

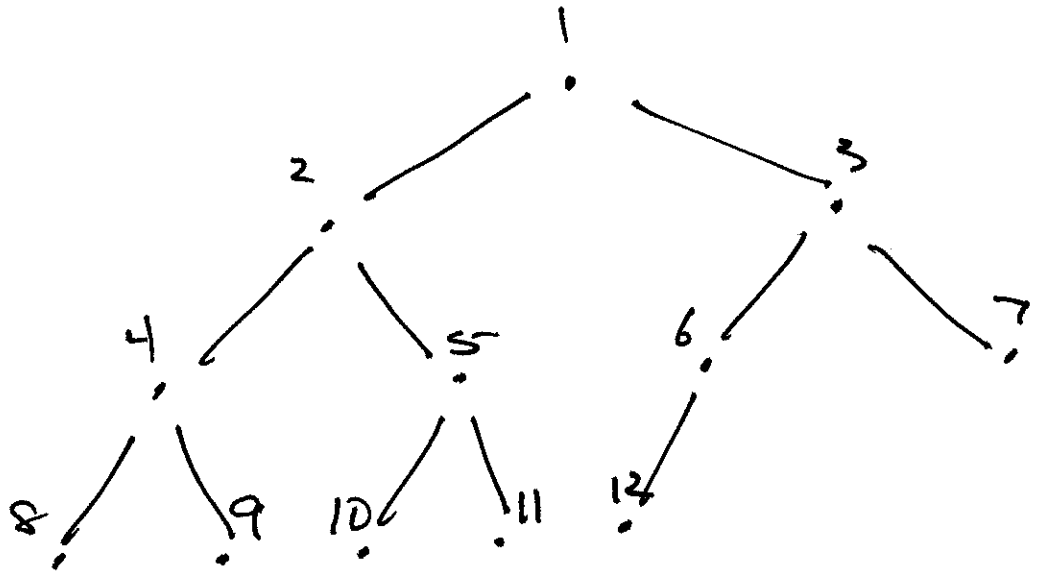
Runtime $= \Theta$(height of subtree at i)

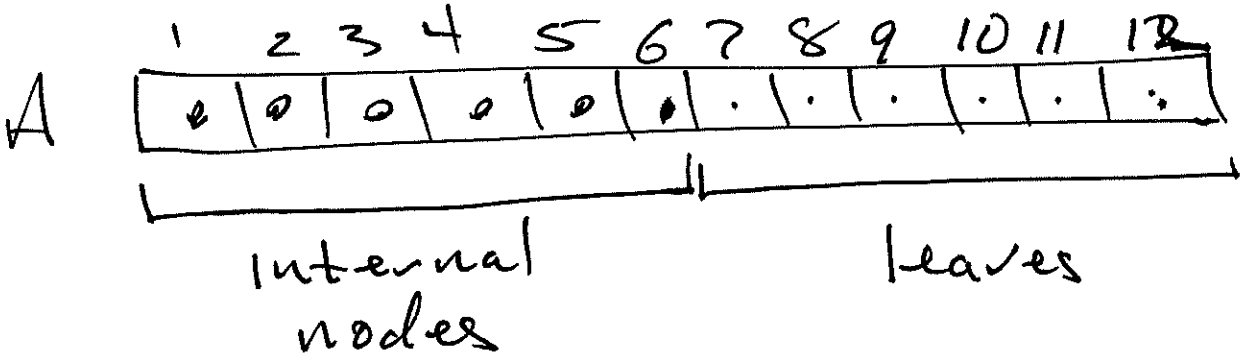
$= \Theta$(log(# nodes in subtree at i))

6.3 Build Heap

Goal: turn an unordered array into a heap

Ex.





start at Parent of rightmost leaf.

Runtime = $\Theta(n)$

6.4 Heapsort

See pseudo-code

$$\text{Runtime} = \Theta(n) + (n-1) \cdot \Theta(\log n)$$

$$= \Theta(n \log n)$$

Exercise

Run Heapsort on

$$A = (10, 8, 12, 20, 3, 15)$$

6.5 P.Q. operations

- HeapMaximum: $\Theta(1)$
- HeapDeleteMax: $\Theta(\log n)$
- HeapExtractMax: $\Theta(\log n)$
- HeapIncreaseKey: $\Theta(\log n)$
- HeapInsert: $\Theta(\log n)$