- SETs: Due Sun 6/11 11:59 PM
- PAs scripts: to be fixed.
- PA8 ext 2 more days → Fri.

6.5 Priority Queues

a Priority Queue maintains a finite set S of records each with an associated key.

$$X = (\cdot, \cdot, \cdot, \cdot, \text{key}) \in S$$

↑ $\text{key}[X] = X.\text{key}$

so

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

<u>states</u>: $\mathcal{S} = \{ \ldots, S, \ldots \}$

Two kinds of P.Q.

- max P.Q. ← we'll do this.
- min P.Q.

<u>Operations</u>:

- Insert(S, x):
- Max(S): returns record with max key
- ExtractMax(S): return and delete record with max key

IncreaseKey(S, x, k):

changes key[x] to k if k > key[x], else does nothing.

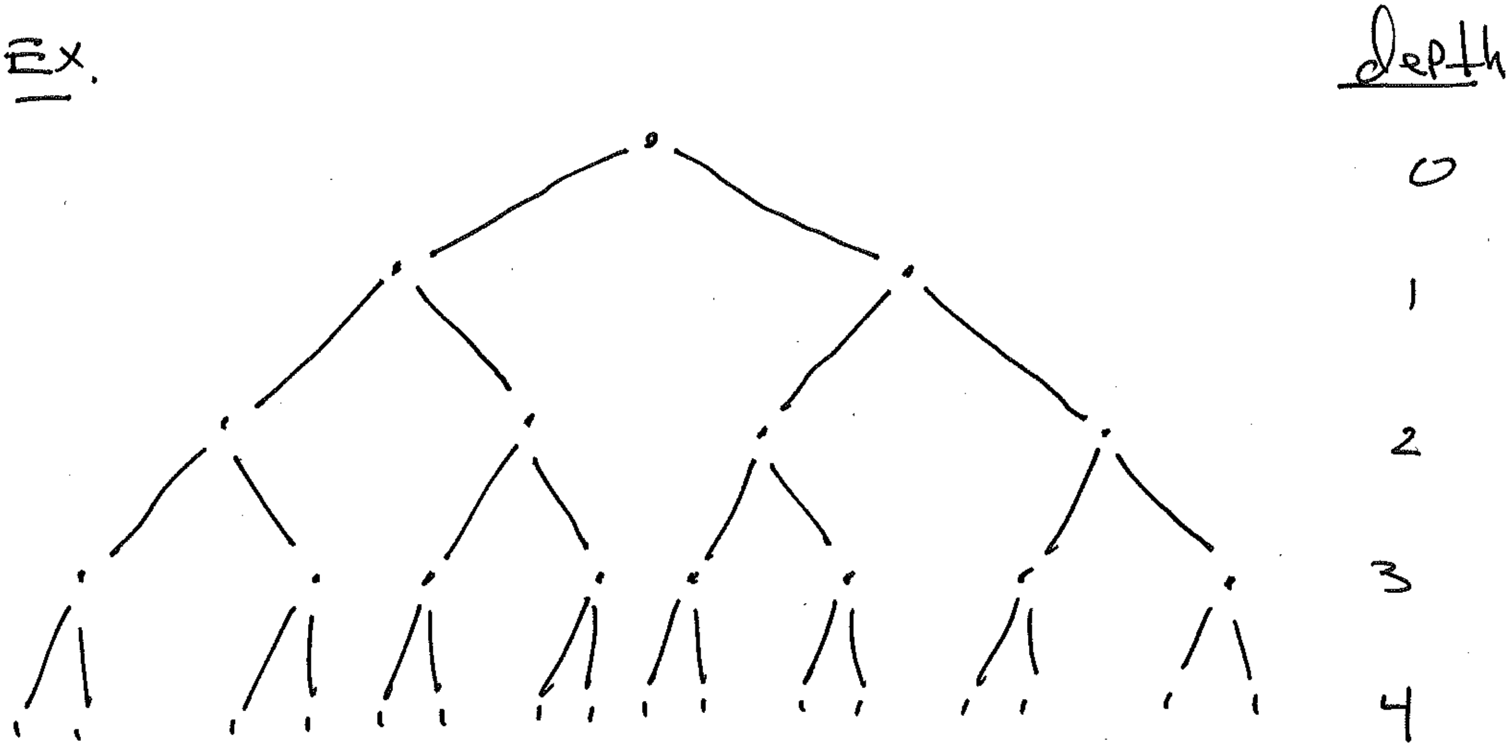
How to implement a P.Q.?

Use a (min or max) Heap.

Defn:

A <u>complete Binary Tree</u> (CBT) is a (rooted) binary tree in which all internal nodes have 2 children, and all leaves at same depth.

Ex.



$$(\#\text{nodes at depth } d) = 2^d$$

$$(\#\text{nodes in a CBT of height } h)$$

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

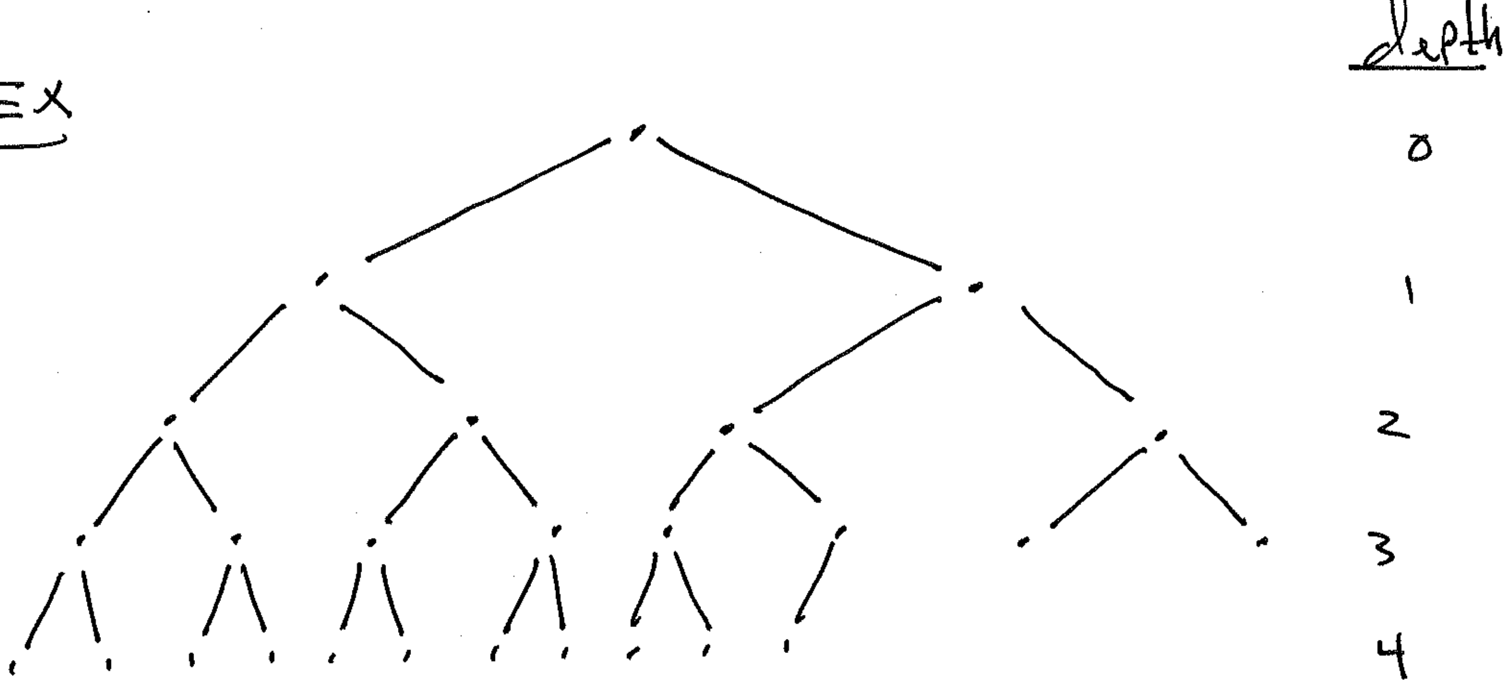
i.e. $\quad n = 2^{h+1} - 1$

Defn

An <u>Almost complete binary tree</u> (ACBT)

is a binary tree that is filled at all levels, left to right, except possibly the last

EX



Let h = height and n = # nodes. Then

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

so $$2^h \le n < 2^{h+1}$$

$$\therefore \quad \lg(2^h) \le \lg(n) < \lg(2^{h+1})$$

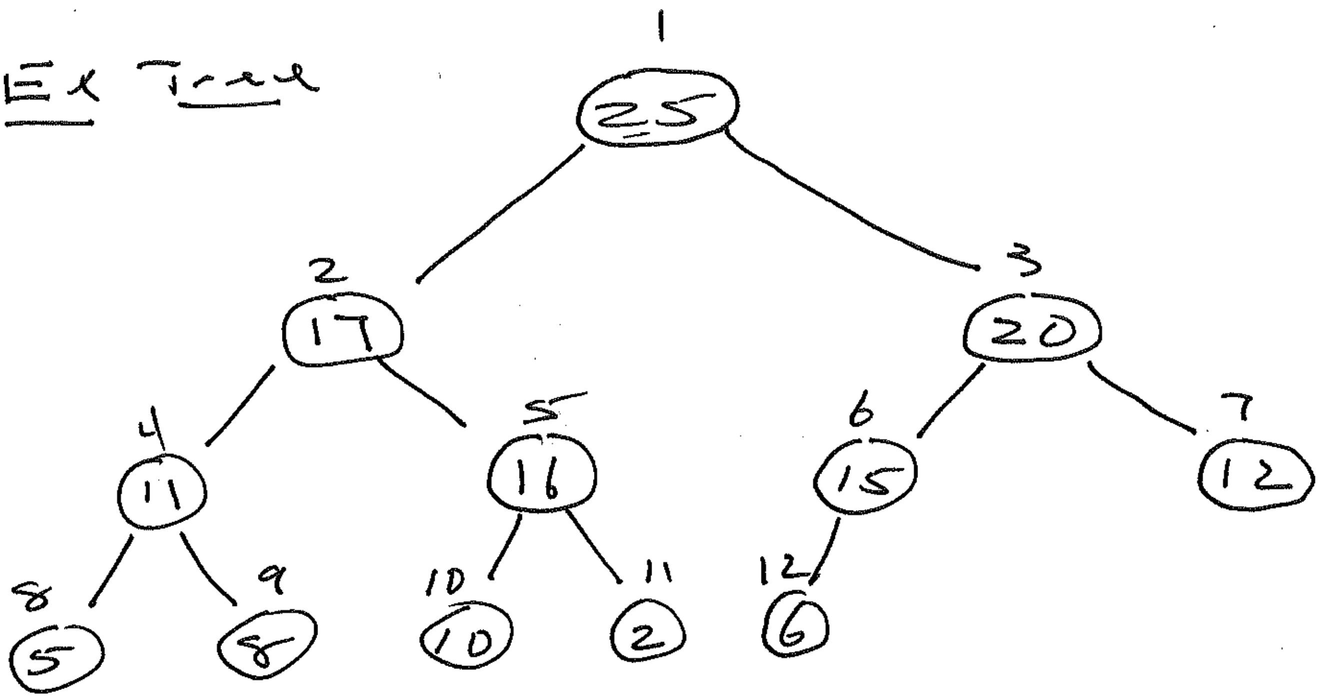
$$\therefore \quad h \le \lg(n) < h+1$$

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

6.1 Heaps

A Binary Heap is a data struct. based on an ACBT stored in an array. Each node corresponds to an array element.

Ex Tree



Array

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A	25	17	20	11	16	15	12	5	8	10	2	6	.	.	.	.

Array has 2 attributes

- length(A) constant
- heapSize(A)

note: heapSize(A) $\leq$ length(A)

helper functions:

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

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

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

Two kinds of heap: max, min

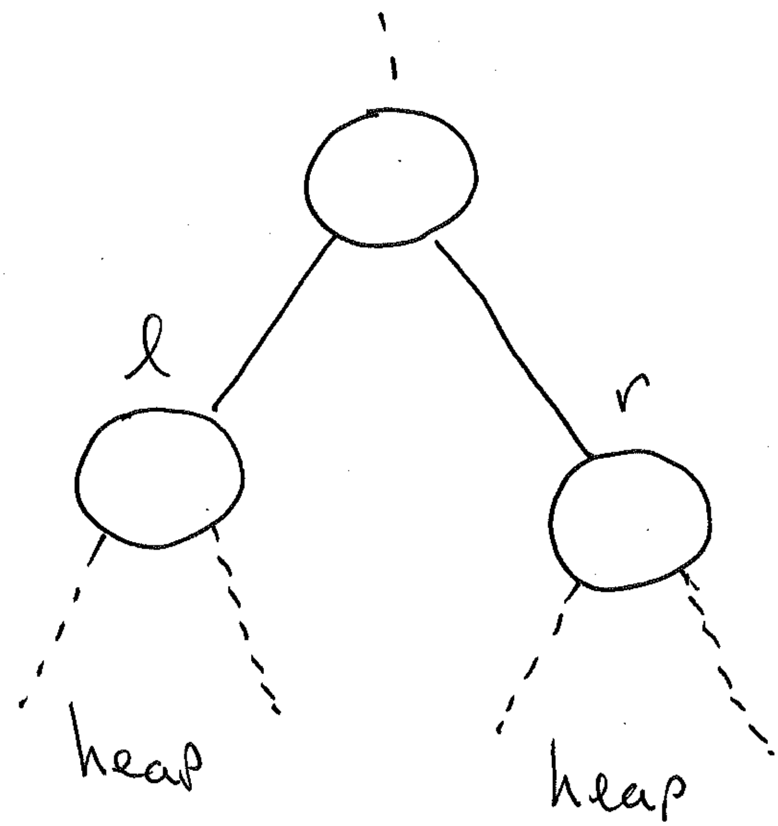
- max-heap property: $A[\text{Parent}(i)] \geq A[i]$
- min-heap property: $A[\text{Parent}(i)] \leq A[i]$.

Heap Operations:

- Heapify
- BuildHeap
- HeapSort
- P.Q. Operations
  - Insert
  - ExtractMax
  - Max
  - IncreaseKey

6.2 Heapify

Suppose we have at index i



Goal: establish (max) heap property at index i.

Heapify(A, i)

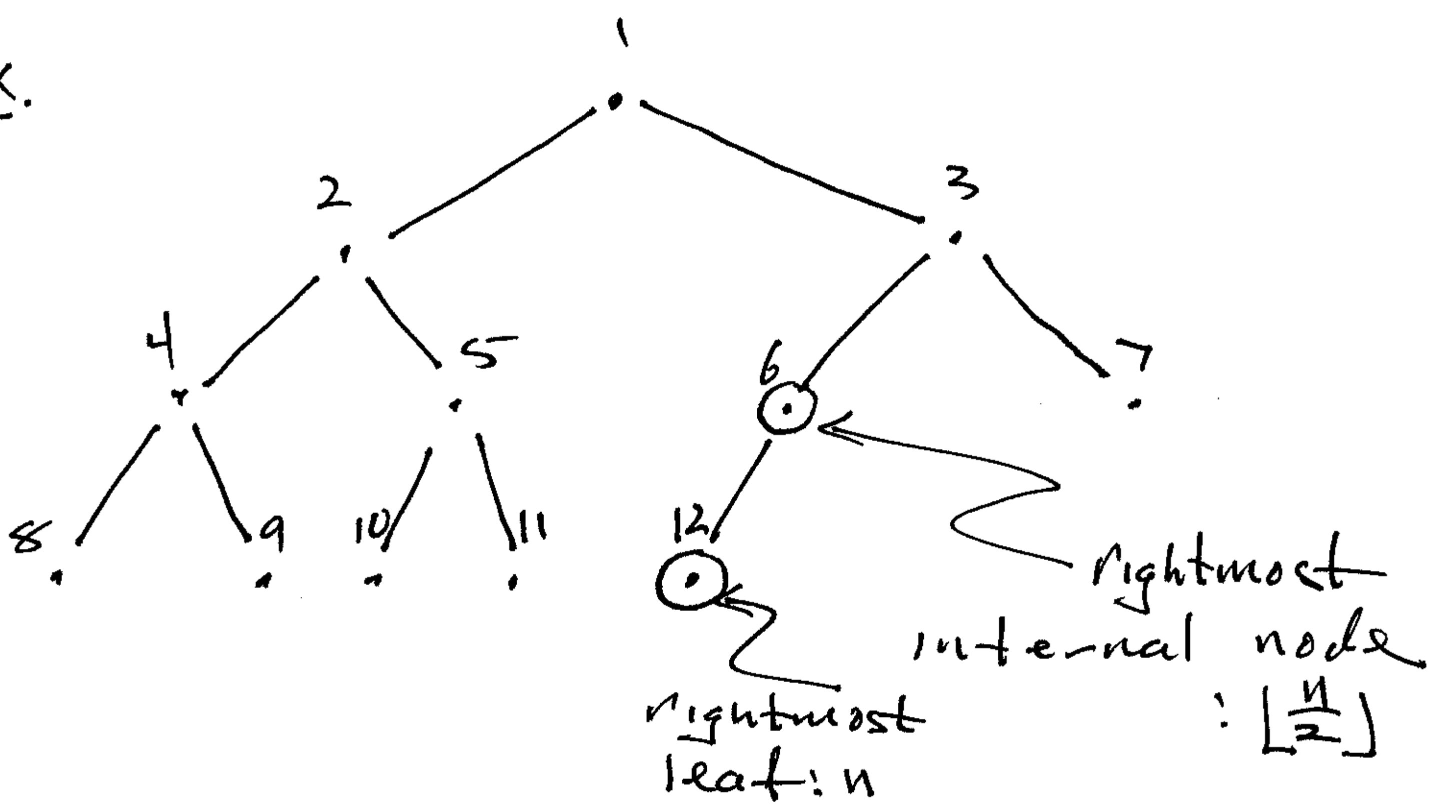
Runtime:

$$\Theta(h) = \Theta(\lfloor \lg(n) \rfloor) = \Theta(\log n)$$

height(T)

6.3 Build Heap

Ex.



Runtime: $\Theta(n)$

6.4 Heapsort

see pseudo code:

Runtime: $\Theta(n \log n)$