CSE 101-01 6-6-23

- SETs: Due Sun 6/11/23 11:59 PM
- scripts: Pa7 & Pa8
- Pa8: ext. 2 more days → Fri.

6.5 Priority Queue

is an ADT that maintains a
a finite set S of records, each
with a key field.

$$x = (\underbrace{\cdot, \cdot, \cdot, \cdot}_{\text{sattelite data}}, \text{key})$$

defines 'Priority' of x.

key[x] = x.key

Two kinds of P.Q.s

- max P.Q. ← we do this,
- min P.Q.

<u>states</u> a state a finite set

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

so set of states

$$\mathcal{S} = \{ \cdots, S, \cdots \}$$

Operations

- Insert(S, x) : inserts a new record x into S.
- Max(S) : returns record with max key.
- ExtractMax(S) : returns and deletes record with max key.
- IncreaseKey(S, x, k) : change key[x] to k, if k > key[x], else do nothing.

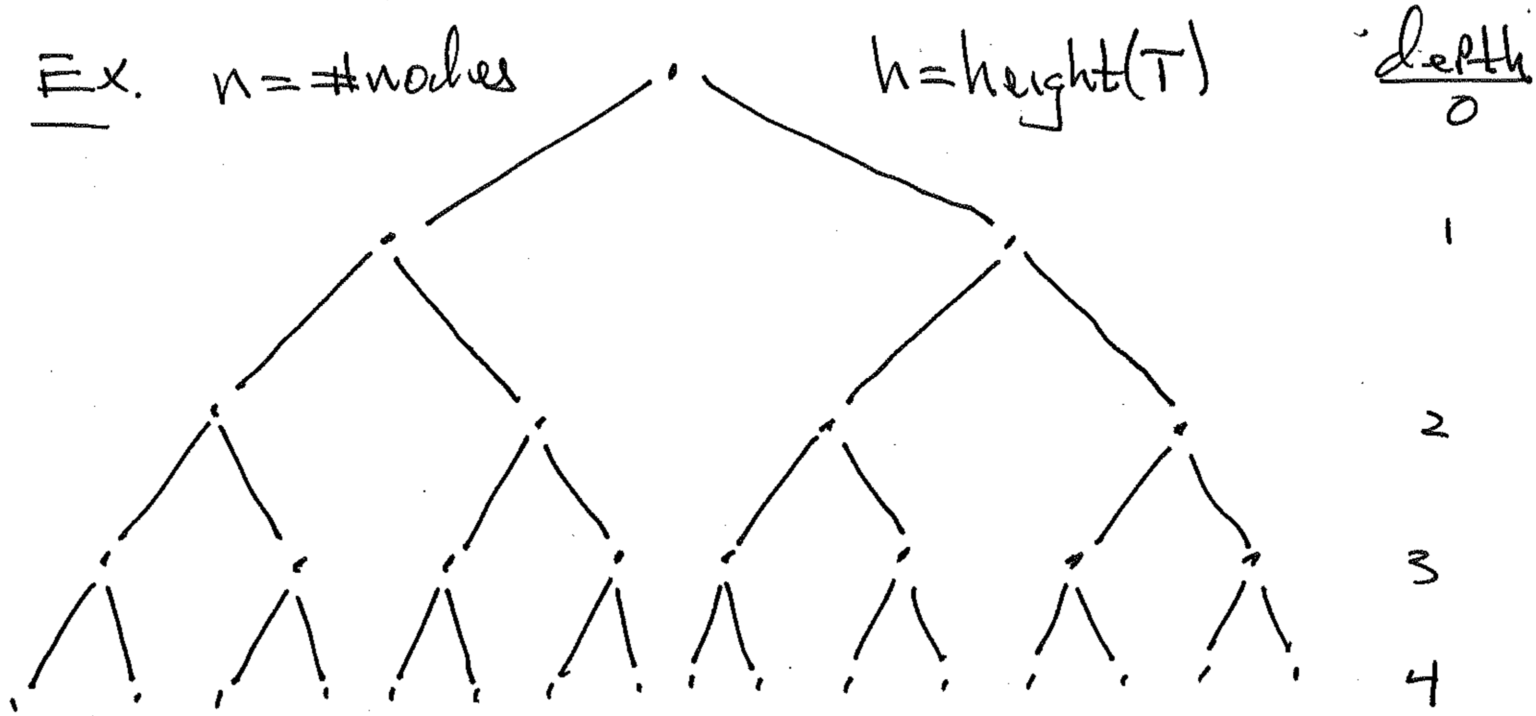
How to implement a P.Q.?

Ans. Binary (max or min) Heap.

<u>Defn</u>

A complete binary tree (CBT) is a rooted binary tree in which all leaves have same depth and all internal nodes have 2 children.

Ex. $n = \#\text{nodes}$ $\quad h = \text{height}(T)$



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

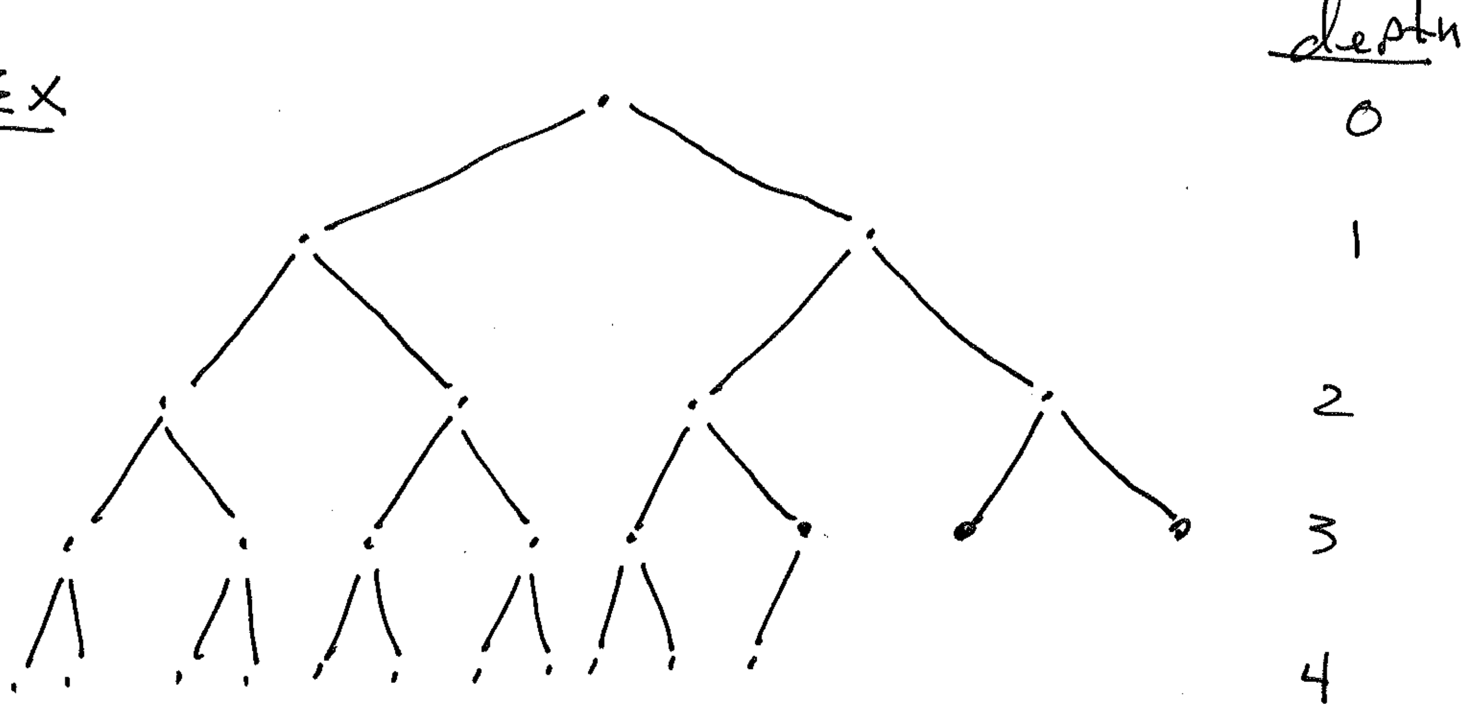
$$(\#\text{nodes}) = \sum_{d=0}^{h} 2^d = \frac{2^{h+1}-1}{2-1}$$

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

Defn

an Almost Complete Binary Tree (ACBT) is a (rooted) binary tree that is filled at all levels, except possibly the last, which is filled left to right

Ex



observe that in an ACBT

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

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

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

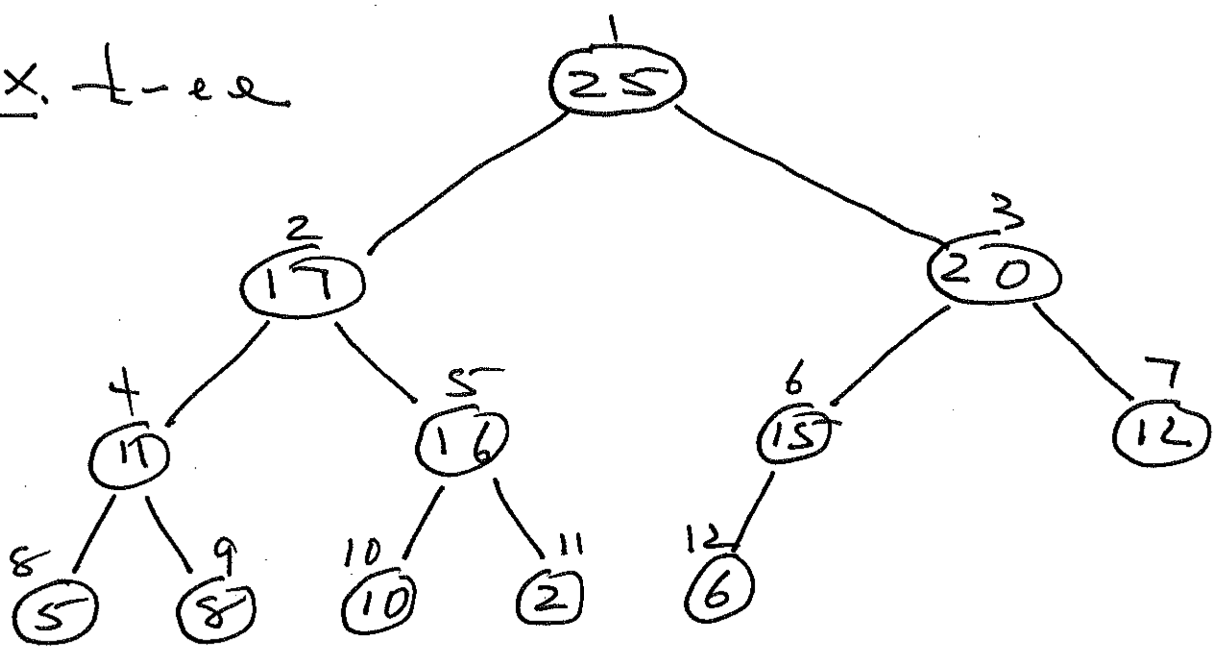
$$\therefore \quad h \le \lg(n) < h+1$$

$$\therefore \quad h = \lfloor \lg n \rfloor$$

6.1 Heaps

A **Binary Heap** is an ACBT represented as an array + some properties.

Ex. tree



array:

												h.s.				len
	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 attributes:

$$0 \le heapSize(A) \le length(A)$$

- helper functions:

$$Parent(i) = \lfloor \tfrac{i}{2} \rfloor \quad (\text{for } i \ge 2)$$

$$left(i) = 2i$$

$$right(i) = 2i+1$$

- heap properties:

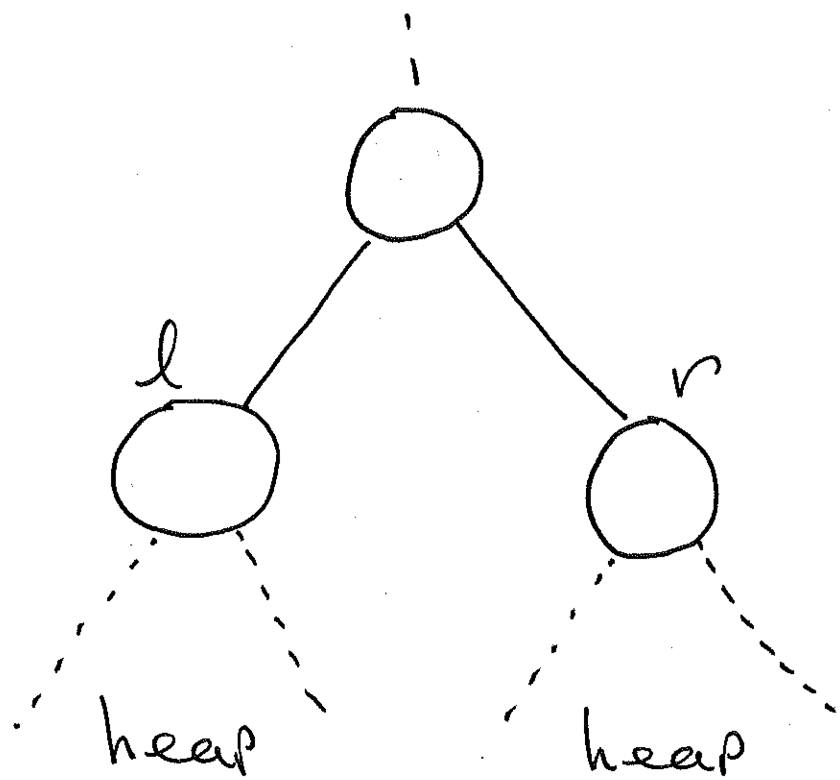
* max-heap: $A[Parent(i)] \ge A[i]$

min-heap: $A[Parent(i)] \le A[i]$

Heap ops

- Heapify()
- BuildHeap()
- HeapSort()
- P.Q. ops: Insert, Max, ExtractMax, IncreaseKey.

6.2 Heapify



Assume subtrees at l, r are valid heaps (max).

Goal of Heapify(A, i): establish heap property at i.

Runtime:

worst case $= \Theta(h)$

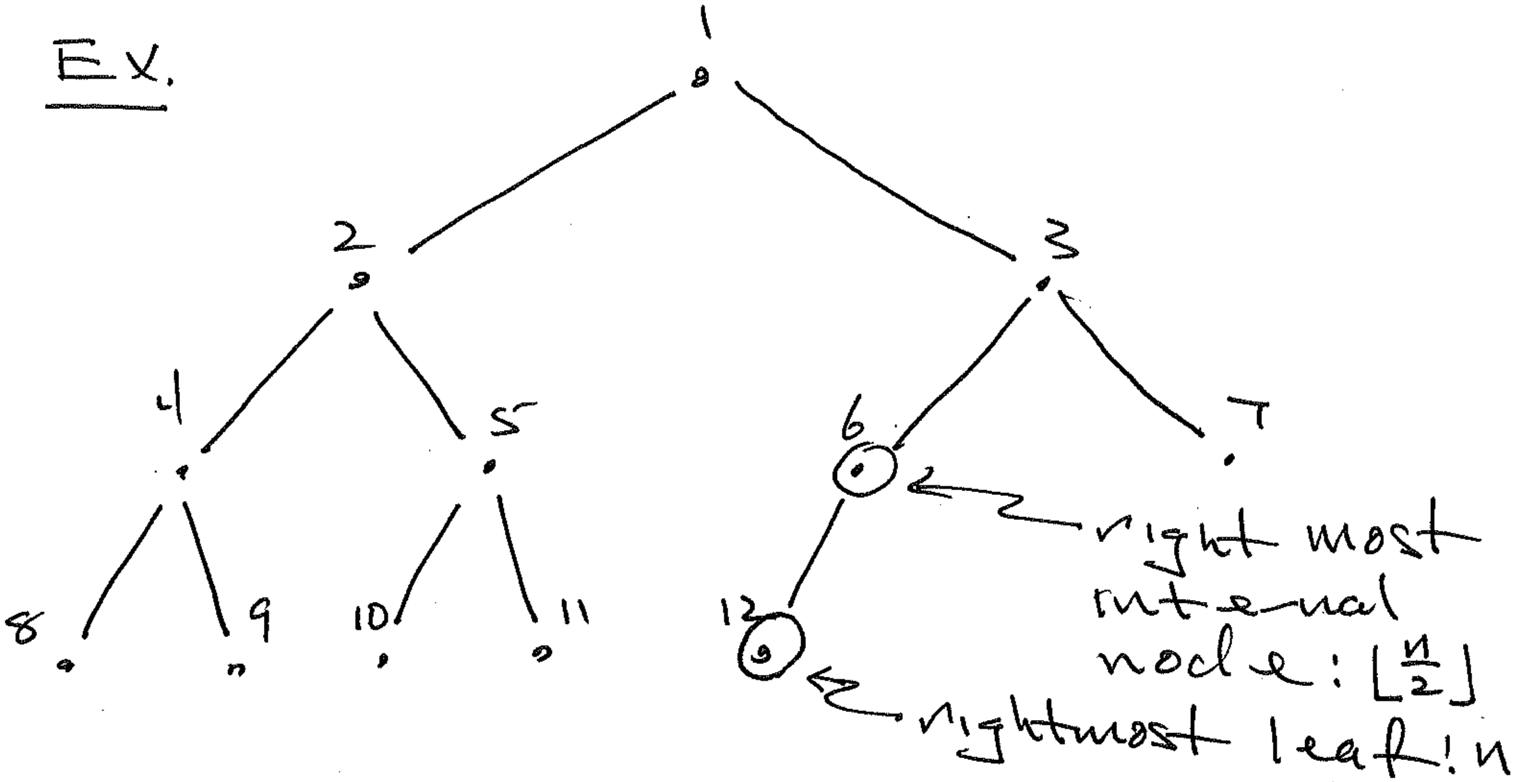
$= \Theta(\lfloor \lg n \rfloor)$

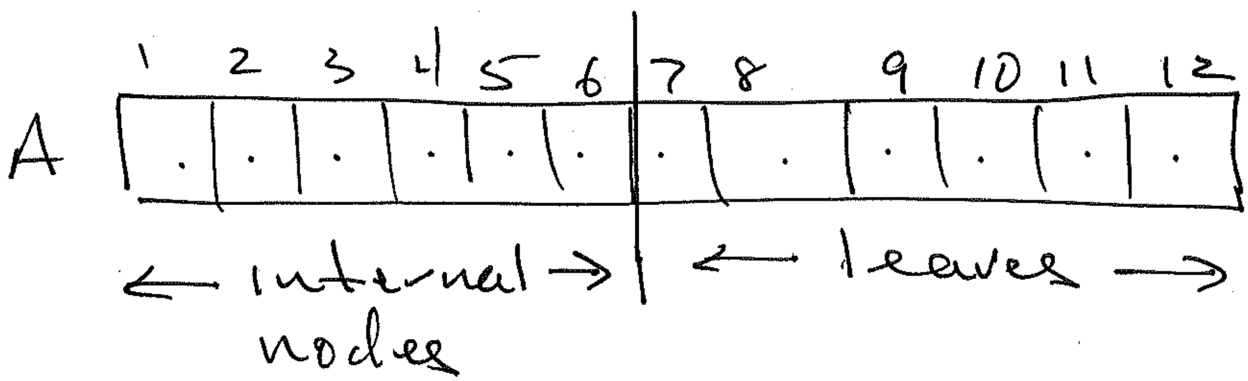
$= \Theta(\log(n))$

6.3 BuildHeap

Goal: turn an unordered array into a valid heap.

Ex.





Runtime: $\Theta(n)$

6.4 Heapsort

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