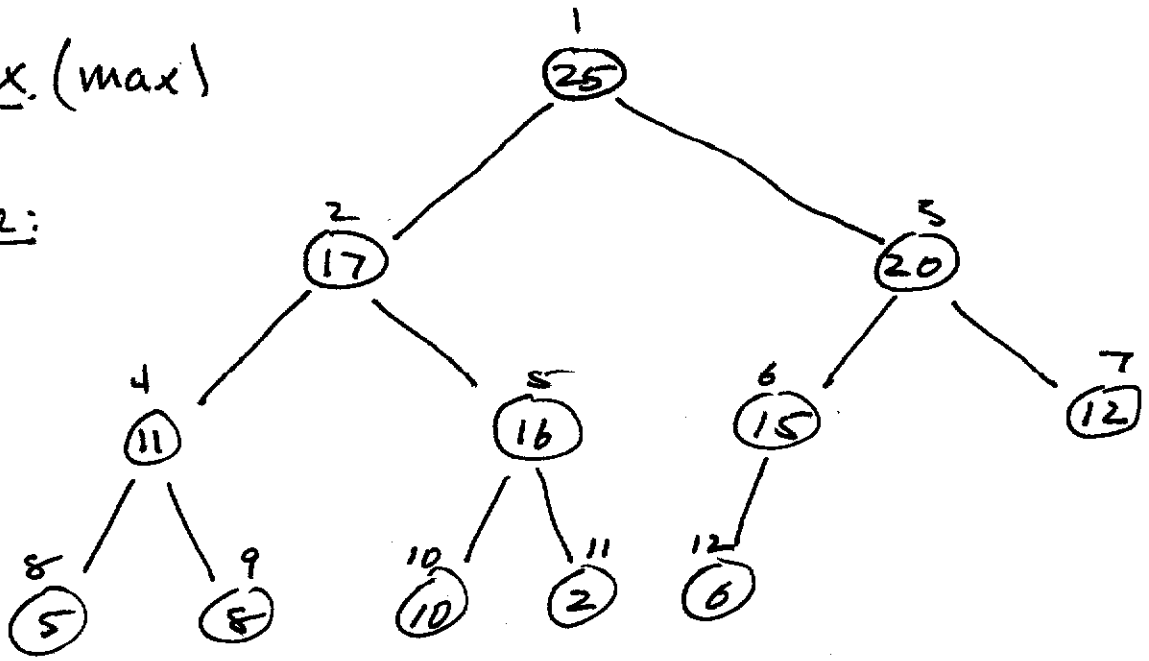
CSE 101 5-27-21

- PA6 : ext. 2 days, Sunday.
- PA7 : Posted, due Thur.
- quizs : Tuesday

6.1 Heaps

Ex. (max)

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 Attributes:

- length[A]
- heapSize[A]

Helper functions:

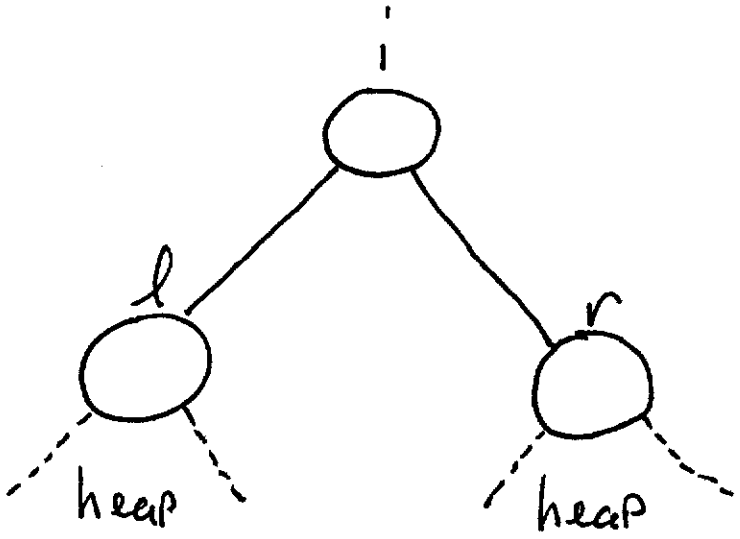
- Parent(i) = $\lfloor \frac{i}{2} \rfloor$ $\quad (i \geq 2)$
- left(i) = $2i$ $\quad 1 \leq i \leq$ heapSize[A]
- right(i) = $2i+1$ $\quad$ " " "

Two kinds of heaps: { max heap, min heap

(*)
- max heap property: $A[Parent(i)] \geq A[i]$
- min heap Property: $A[Parent(i)] \leq A[i]$

6.2 Heapify (Max-Heapify)

Heapify(A, i) has Pre-condition:



Pre: subtrees at l and r are valid (max)-heaps, i.e. satisfy max-heap Prop.

Goal: establish (max)-heap property at i (if it isn't already).

see: Pseudo-code

- Runtime of heapify, worst case

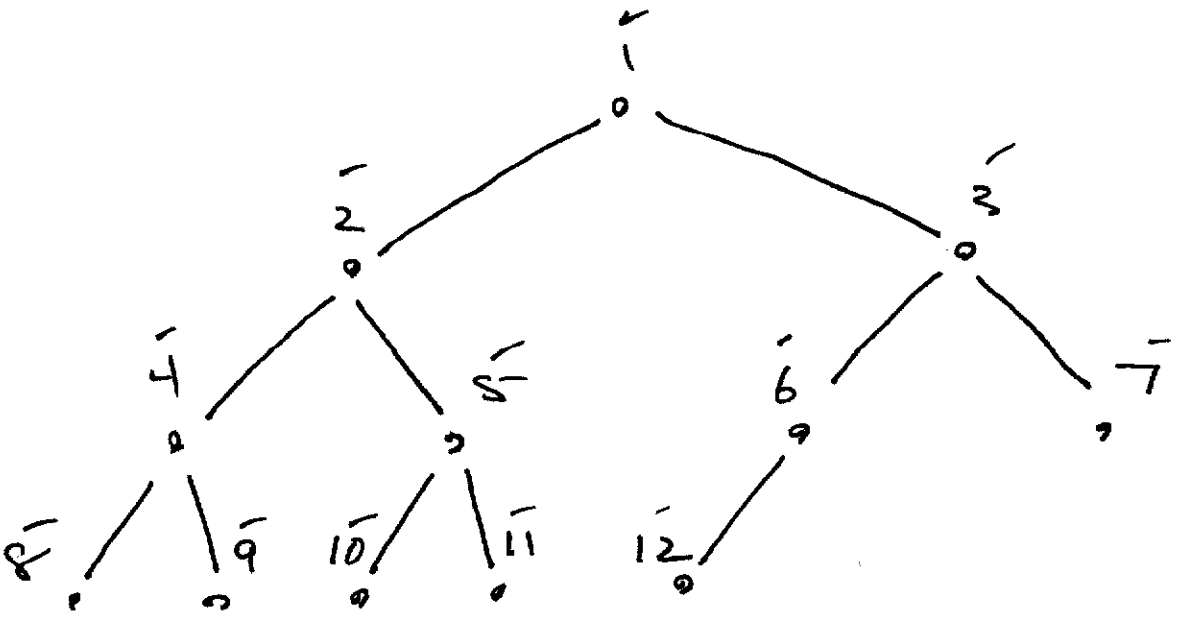
since Heapify has only const. time operations:

$$\text{runtime} = \Theta(\text{depth of recursion})$$
$$= \Theta(\text{height of tree})$$
$$= \Theta(h)$$
$$= \Theta(\lfloor \lg n \rfloor) = \Theta(\log n)$$

where $n = \#$ nodes in (sub)-tree

6.3 Build-Heap

Call heapify on a seq. of nodes, chosen so that precondition is always satisfied, ending at root (index 1).



$A_1\ A_2\ A_3\ A_4\ A_5\ \underline{A_6}\ A_7\ A_8\ A_9\ A_{10}\ A_{11}\ A_{12}$

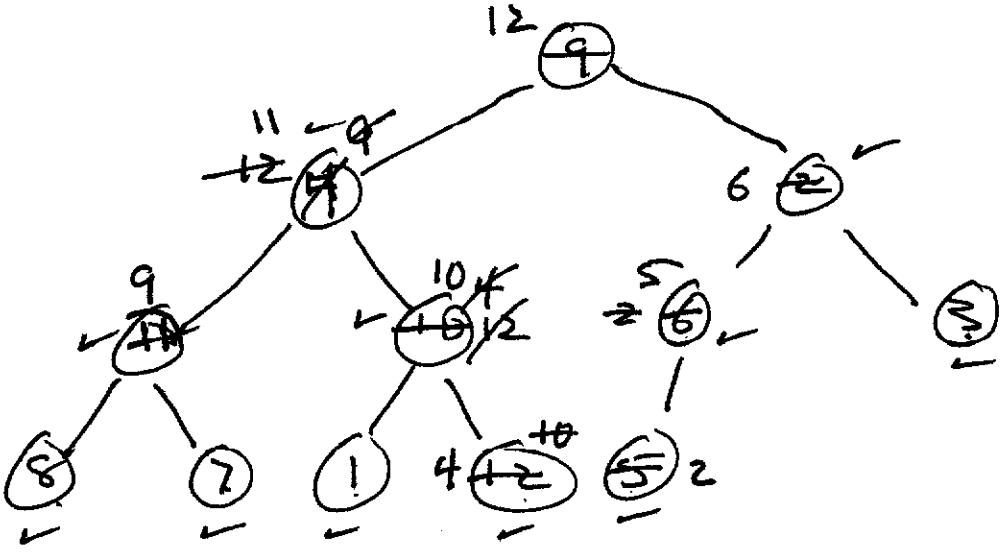
internal nodes — leaves skip these.

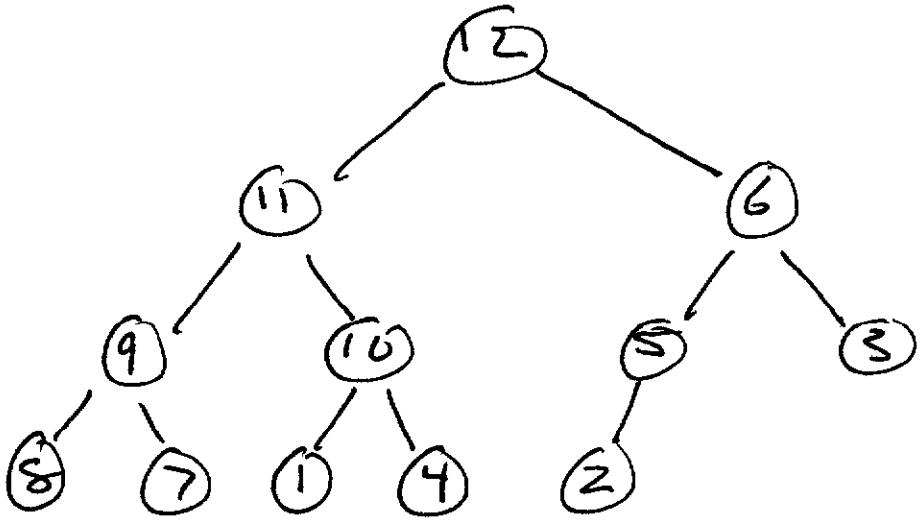
start at rightmost internal node

Ex.

A

1	2	3	4	5	6	7	8	9	10	11	12
9	4	2	11	10	6	3	8	7	1	12	5





A

1	2	3	4	5	6	7	8	9	10	11	12
12	11	6	9	10	5	3	8	7	1	4	2

- Runtime of BuildHeap:

$$T(n) = \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} \Theta(\text{height of subtree at } i)$$

$$\vdots \text{ - see text -}$$

$$= \Theta(n)$$

6.4 Heapsort

- first build a heap out of A.
- A[1] <-> A[heapSize[A]]
- heapSize[A] --
- heapify(A, 1)
- repeat

- Runtime of Heapsort(A)

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

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

note: sorts in place, i.e. doesn't use extra memory, like merge sort.

note: can't sort faster than $\Theta(n \log n)$. why? Take cse 102.

search space: $n!$

$$n!, \frac{n!}{2}, \frac{n!}{2^2}, \frac{n!}{2^3}, \ldots, 1$$

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

6.5 Priority Queues { max / min

A Priority Queue is an ADT that maintains a finite set S of records:

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

states: $\{ \ldots\ldots, S, \ldots\ldots \}$

↑ one state

Operations

- Insert(S, x) : insert new record x into S
- Maximum(S) : returns x with max key
- ExtractMax(S) : return & delete x with max key
- DeleteMax(S) : deletes x with max key
- IncreaseKey(S, x, k) : increase key[x] to k.

PA6: Questions

- fix comment block in PrintDictionary()
  do not Print values.
- Lex.c may Print sorted strings using in-order tree-walk, or Plug values (indices) into array.