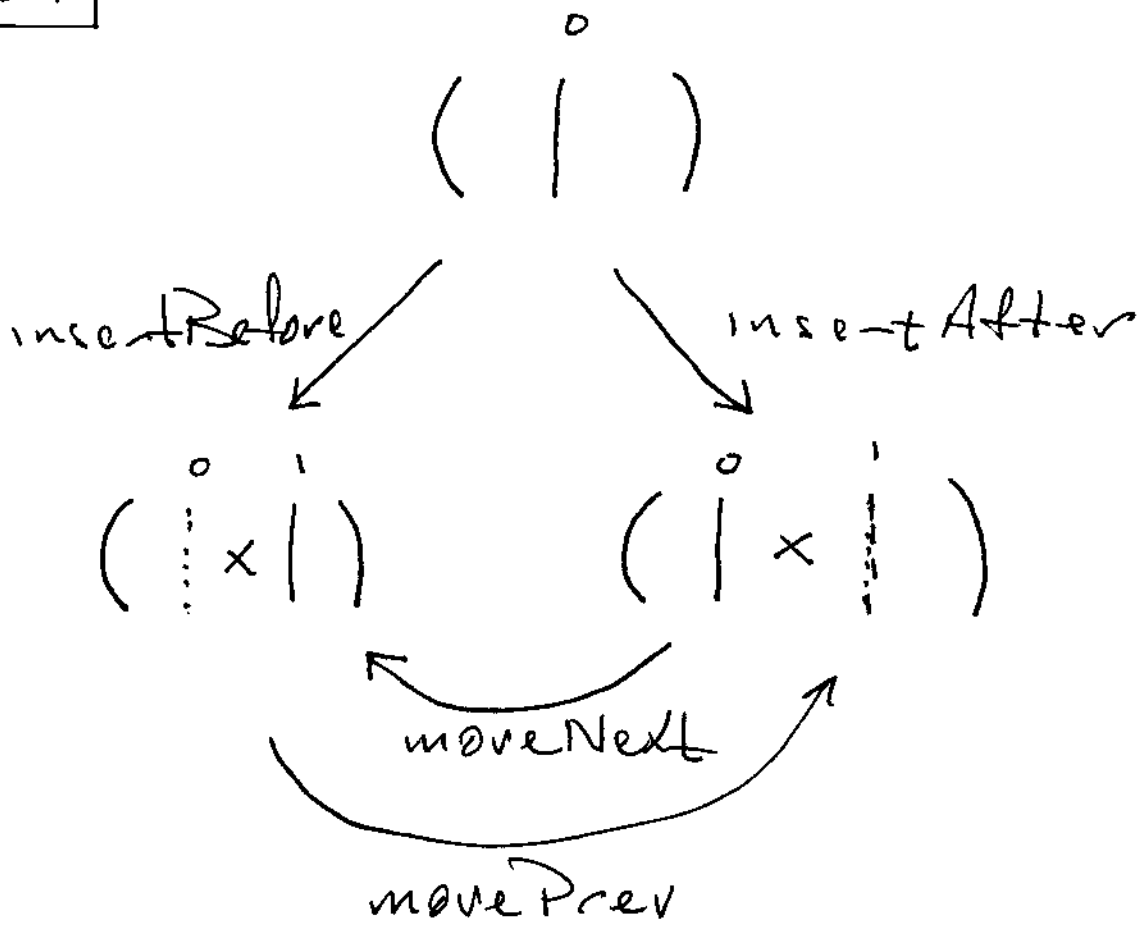
CSE 101 5-28-20

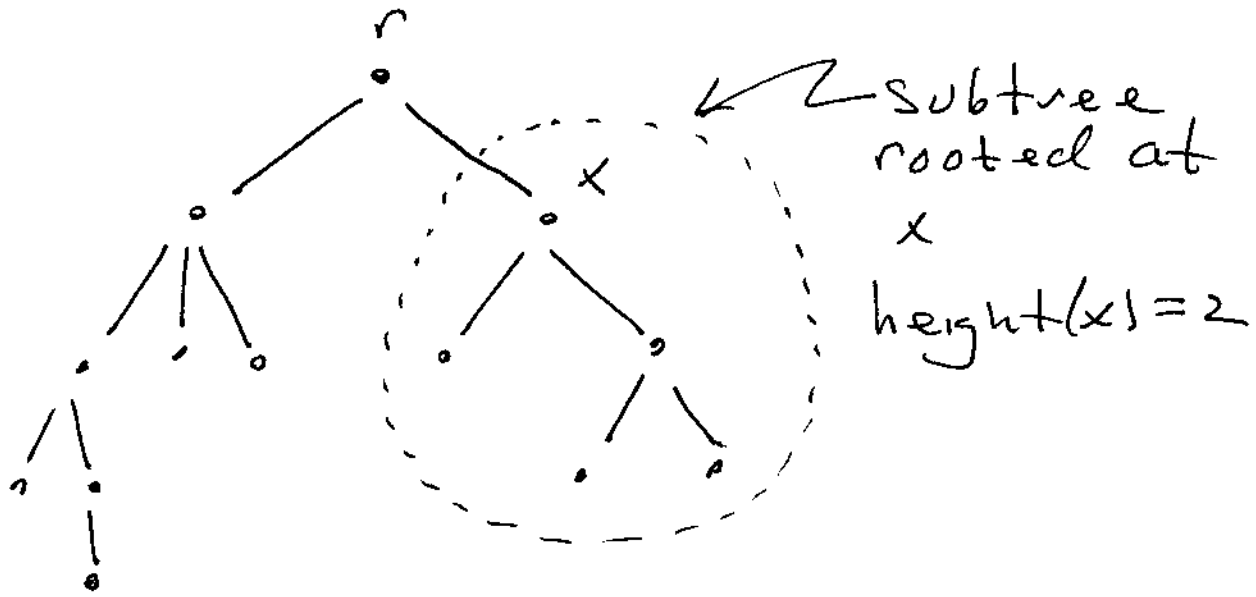
Pa7



SETS: Student Evaluations of Teaching.

chapter 6 : Heaps

Recall : given a node x in a rooted tree, the subtree rooted at x is the subtree consisting of x and all its descendants

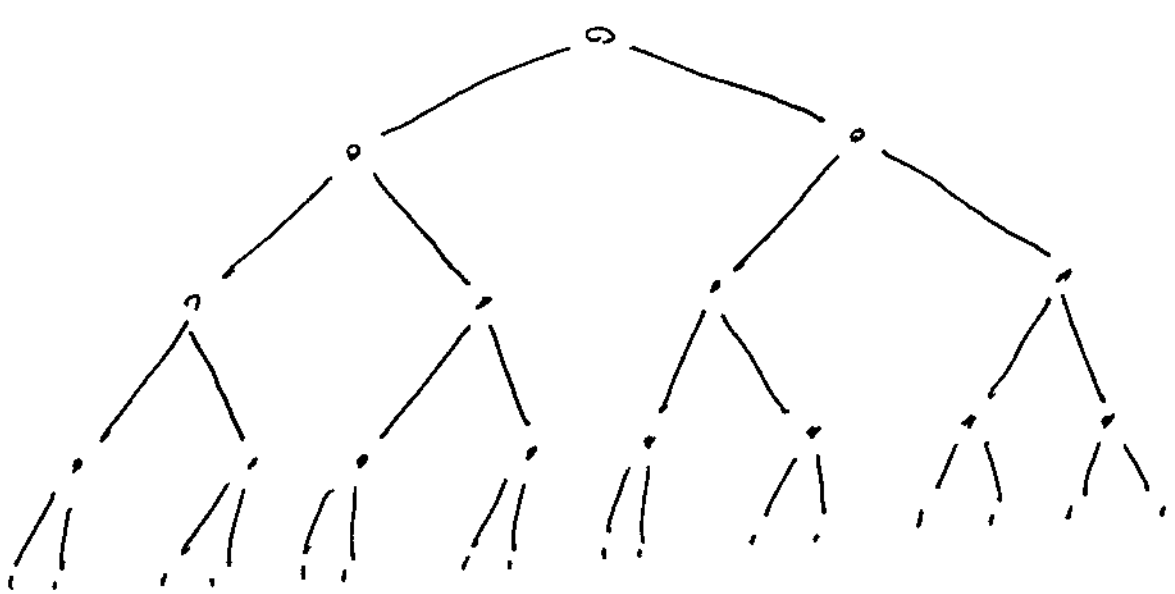


height(x) = depth of deepest leaf in subtree rooted at x

Fact: In any binary tree T with n nodes and $h = height(root[T])$,

$$h \geq \lfloor \lg n \rfloor .$$

CBT (complete binary tree)

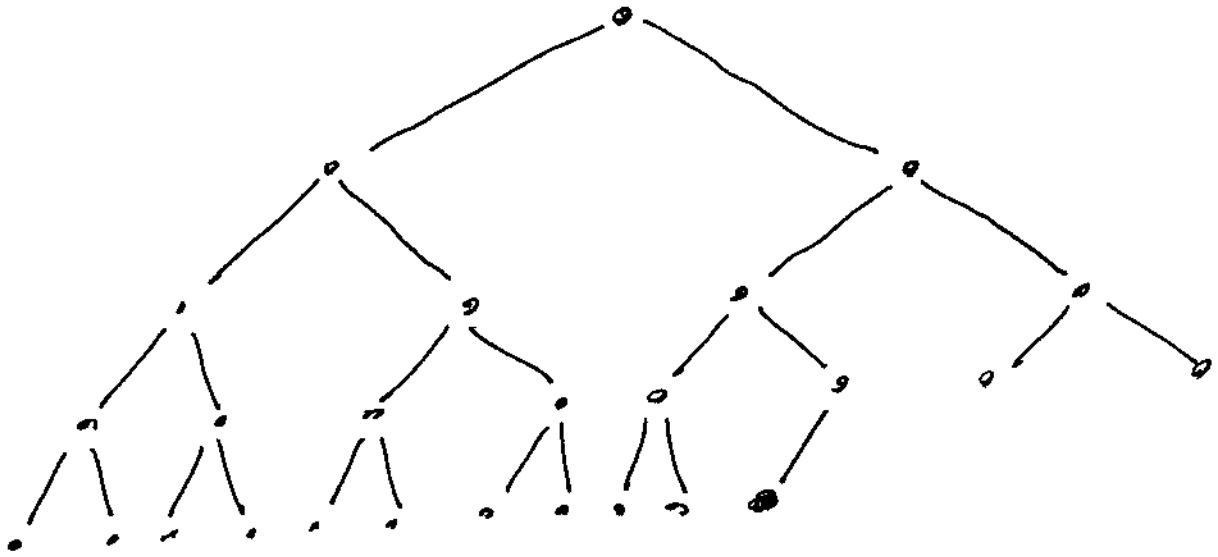


depth	# nodes
0	1
1	2
2	4
3	8
4	16

$$\Rightarrow 0$$

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

ACBT (almost complete binary tree)



The number of nodes n in an ACBT of height h must satisfy

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

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

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

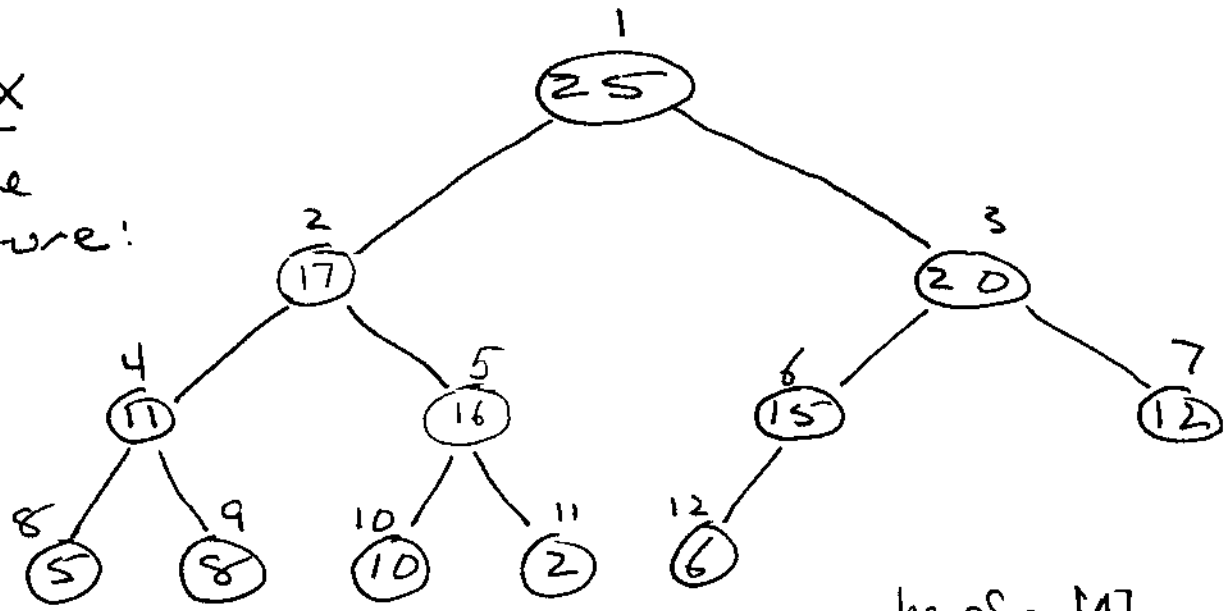
$$\therefore \quad h = \lfloor \lg(n) \rfloor$$

Sec 6.1 Heaps

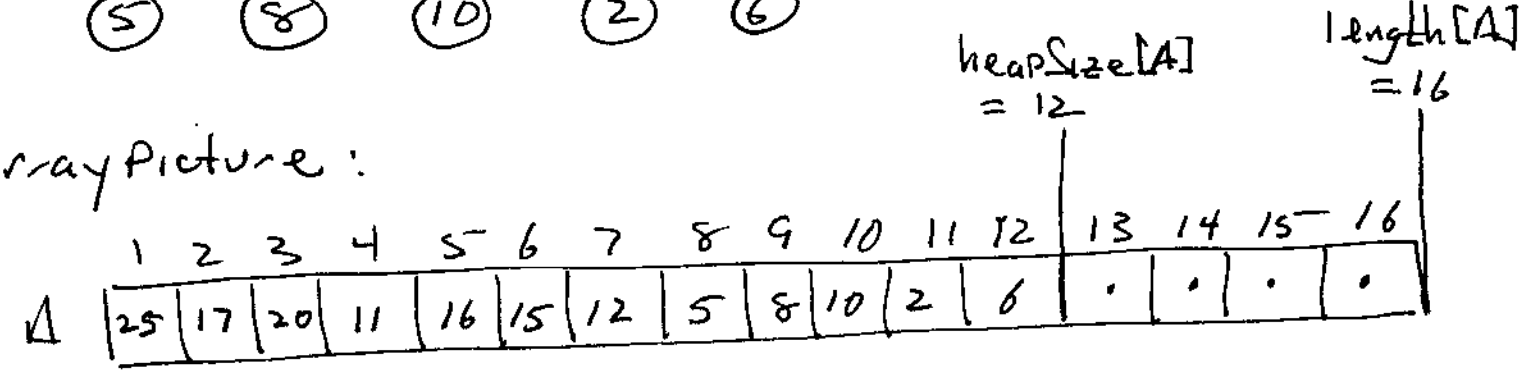
A Binary Heap data structure is an array which represents an ACBT. Each node in tree corresponds to one array element.

Ex

tree Picture:



Array Picture:



Such an array has 2 attributes

- length[A]
- heapSize[A]

Array also has 3 helper-functions

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

$$left(i) = 2i$$

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

There are 2 kinds of heaps:
max heap and min heap.

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

for $1 \leq i \leq heapSize[A]$

we will discuss max heaps.

exercise: re-do everything for a min heap.

will also discuss:

- Heapify()
- BuildHeap()

] How

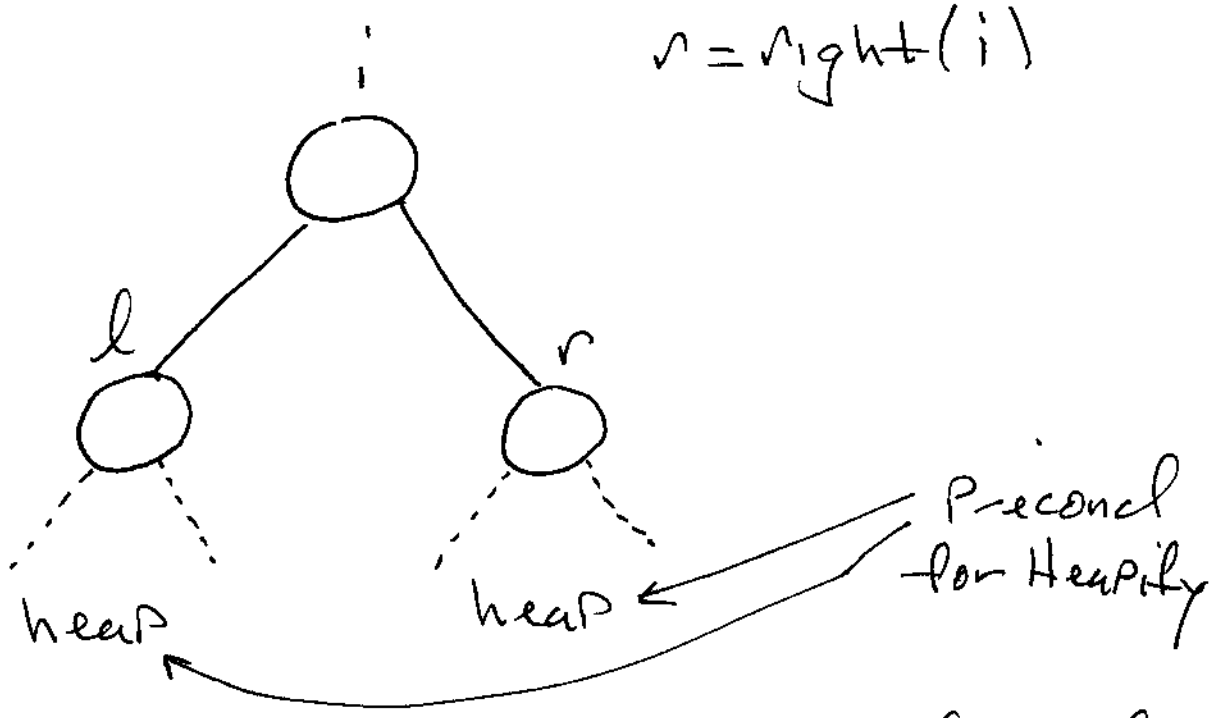
- HeapSort()
- PriorityQueue ops
  - Insert()
  - ExtractMax()
  - Max()
  - IncreaseKey()

] Why

6.2 Heapify

$l = left(i)$

$r = right(i)$



Assume subtrees rooted at l and r are valid heaps

Goal: make subtree rooted at i into a valid heap, if it isn't already

note: heapify climbs down a line of ancestry, recursively (see Pseudo-code)

so in worst case

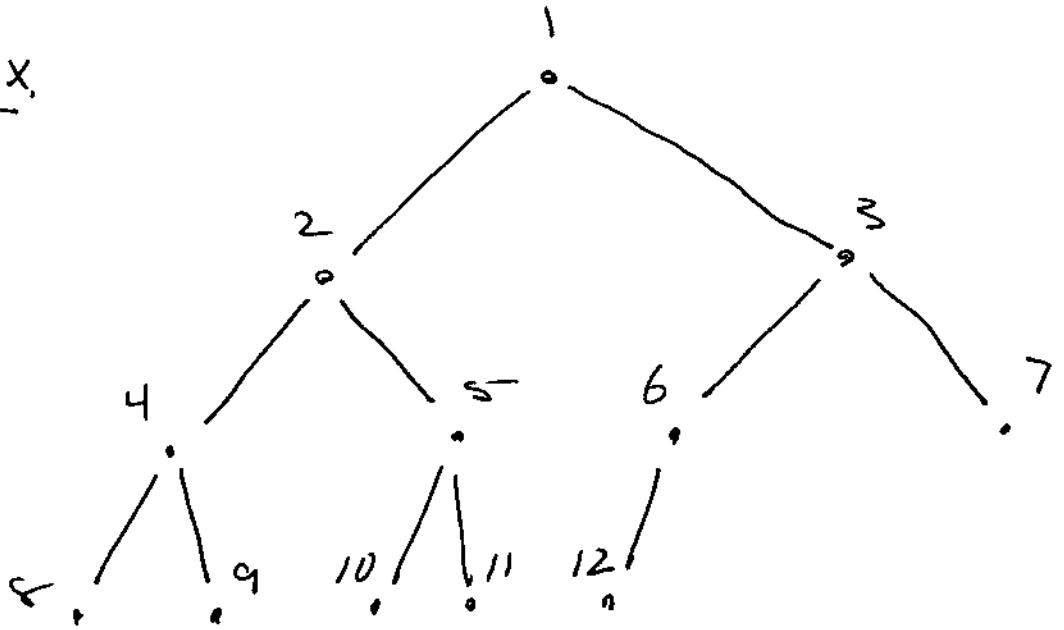
$$\text{run time} = \Theta(\log n)$$

$$= \Theta(h)$$

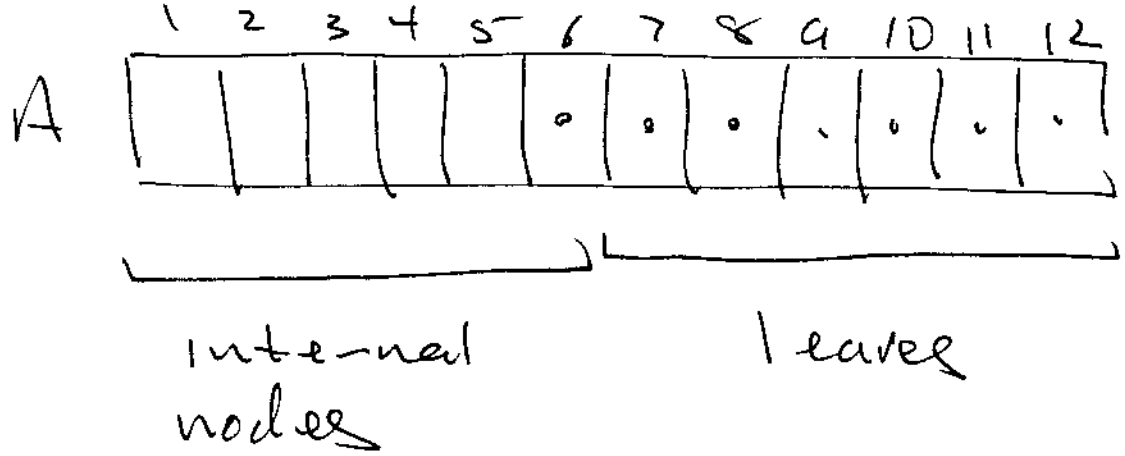
↑ height of subtree rooted at i

6.3 Build Heap

Ex.



Array Picture



start at rightmost internal node

if $n = length[A]$, then rightmost internal node has index $\lfloor \frac{n}{2} \rfloor$

Runtime: $\Theta(n)$

(better than $\Theta(n \log n)$.)

6.4 Heapsort

- first build a max heap out of array A.

- swap $A[1] \leftrightarrow A[heapSize[A]]$
- decrement heapSize[A]
- Heapify(A, 1).

repeat

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

$= \Theta(n \log n)$

Pa 7