

### 6.3 Build-Heap

WE CAN CREATE A HEAP FROM AN UNORDERED ARRAY BY CALLING HEAPIFY ON THE INTERNAL NODES STARTING AT THE BOTTOM AND WORKING OUR WAY UP.

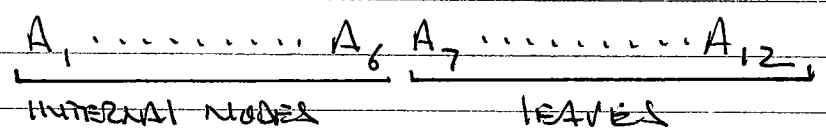
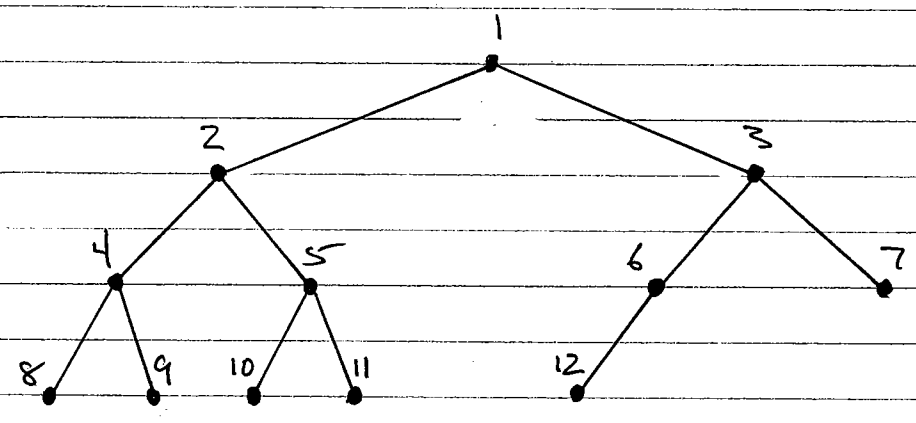
LET  $n = \text{heap-size}[A] = \text{length}[A]$  AND OBSERVE THAT  $\lfloor \frac{n}{2} \rfloor$  IS THE INDEX OF THE LAST (i.e. RIGHTMOST) INTERNAL NODE, SINCE IT IS THE PARENT OF THE LAST LEAF.

THUS THE ELEMENTS OF  $A[\lfloor \frac{n}{2} \rfloor + 1, \dots, n]$  ARE ALL LEAVES AND HENCE EACH IS ALREADY A HEAP. WE PROCESS THE INTERNAL NODES  $A[1, \dots, \lfloor \frac{n}{2} \rfloor]$  IN AN ORDER WHICH GUARANTEES THAT THE SUBTREE ROOTED AT EACH CHILD IS ALREADY A HEAP.

#### Build-Heap(A)

- 1.)  $n \leftarrow \text{heap-size}[A] \leftarrow \text{length}[A]$
- 2.) for  $i \leftarrow \lfloor \frac{n}{2} \rfloor$  DOWN TO 1
- 3.)     Heapify(A, i)

EX



THE RUN TIME OF BUILD-HEAP IS  $O(n \lg n)$  SINCE EACH CALL TO HEAPIFY COSTS  $O(\lg n)$  AND THERE ARE  $\lfloor n/2 \rfloor = \Theta(n)$  SUCH CALLS.

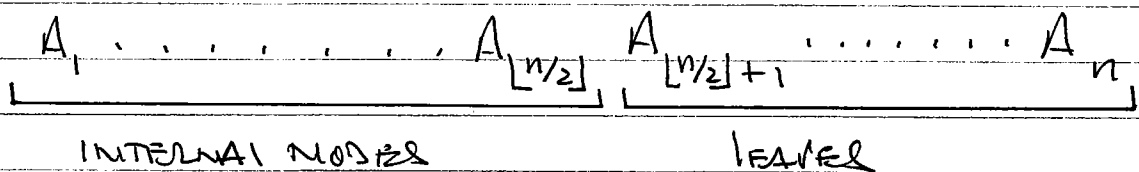
THIS BOUND IS NOT TIGHT HOWEVER. OBSERVE THAT THE COST OF CALLING HEAPIFY ON A NODE OF HEIGHT  $h$  IS  $\Theta(h)$ , AND THE HEIGHTS OF MOST NODES ARE SMALL.

LEMMA (PROBLEM 6.3-3, P. 135)

AN ACBT ON  $n$  NODES HAS AT MOST  $\lfloor n/2^{h+1} \rfloor$  NODES AT HEIGHT  $h$ .

PROOF

RECALL THE LAST (I.E. RIGHTMOST) INTERNAL NODE IS THE PARENT OF THE LAST LEAF, AND HENCE HAS INDEX  $\lfloor n/2 \rfloor$ .



THEREFORE THE NUMBER OF LEAFS (NODES AT HEIGHT 0) IS

$$n - \lfloor n/2 \rfloor$$

IF WE DELETE THESE LEAFS WE OBTAIN AN ACBT ON  $\lfloor n/2 \rfloor$  NODES WHICH THEREFORE

