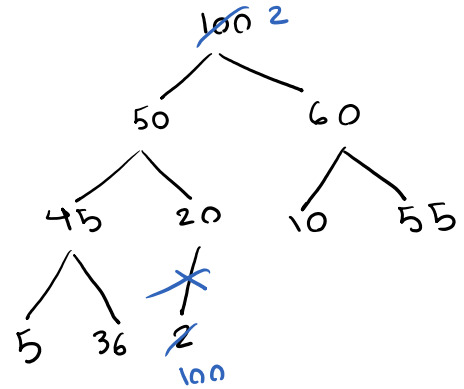
Binary Heaps

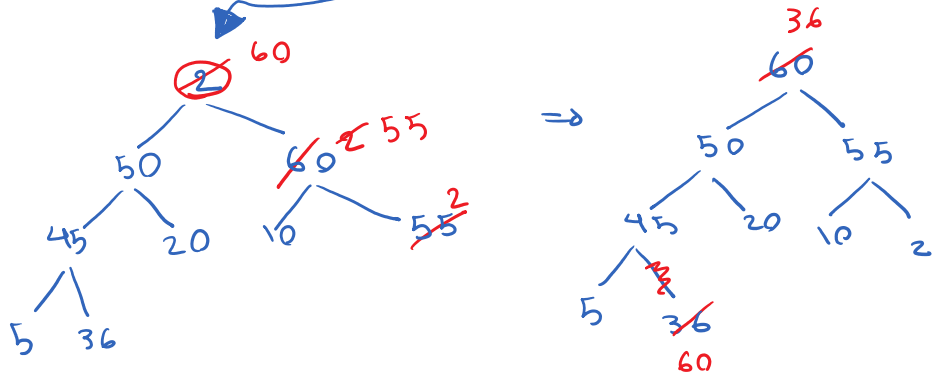
Thursday, October 29, 2020 5:00 PM

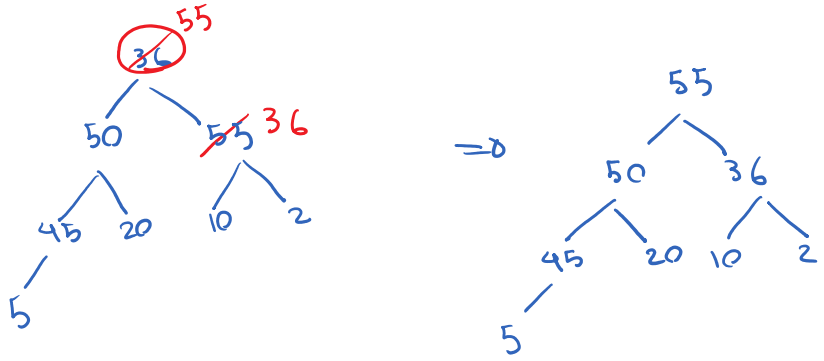
Reminder: * Lab 4 is due this Sunday.
* Exam 2 is on Nov 12th

Deleting a root:

1. swap the root with the least element → $\Theta(1)$
2. delete the root → $\Theta(1)$ (Skip this when coding)
3. call max_heapify → $\Theta(h)$ for the new root (index=0)





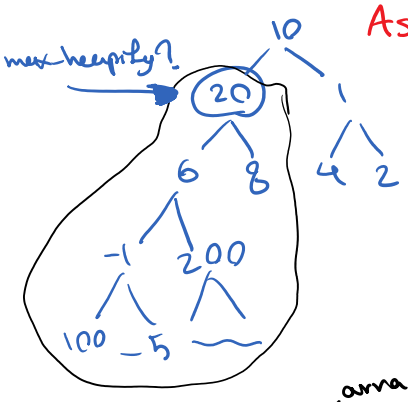


Question: What is the running time to delete the root only once if you n elements in a max_heap?

$T(n) = O(h) = O(\log n)$

↳ hw 7

10 Assumption of max_heapify:

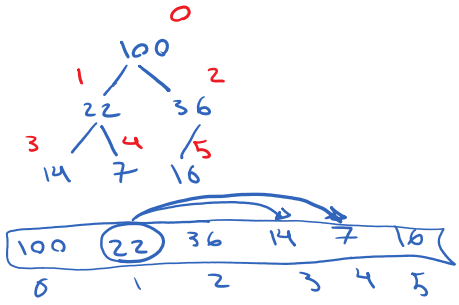


Assumption of max_heapify:

⇒ the left sub-tree & the right ~ should both be max-heaps.

max_heapify (a, i) — a: array; i: index at the vertex I want to apply max_heapify

```
mx = i;
left = 2i+1;
right = 2i+2;
if a[mx] < a[left]   → error
if a[mx] > a[right]  → error
```



Dorothy :)

n = a.length

```
max_heapify (a, i)
  mx = i;
  left = 2i+1;
  right = 2i+2;
  if (left < a.length && a[left] > a[mx])
    mx = left;
  if (right < a.length && a[right] > a[mx])
    mx = right;
  if (mx != i)
    swap (a[i], a[mx]);
    max_heapify (a, mx);
```

Example: Sort a = [25, 11, 9, 69, 420, 27]

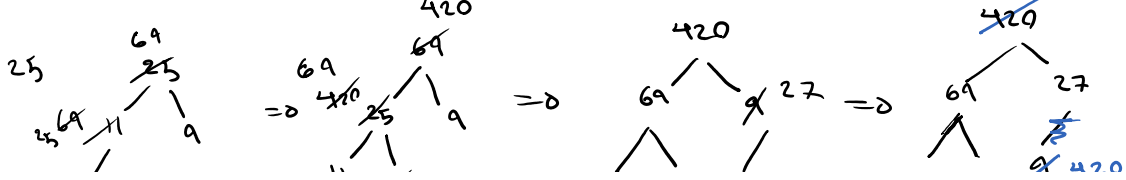
Heap Sort: O(nlogn)

⇒ ① Build max heap → better way to build: using successive insertion: O(nlogn)

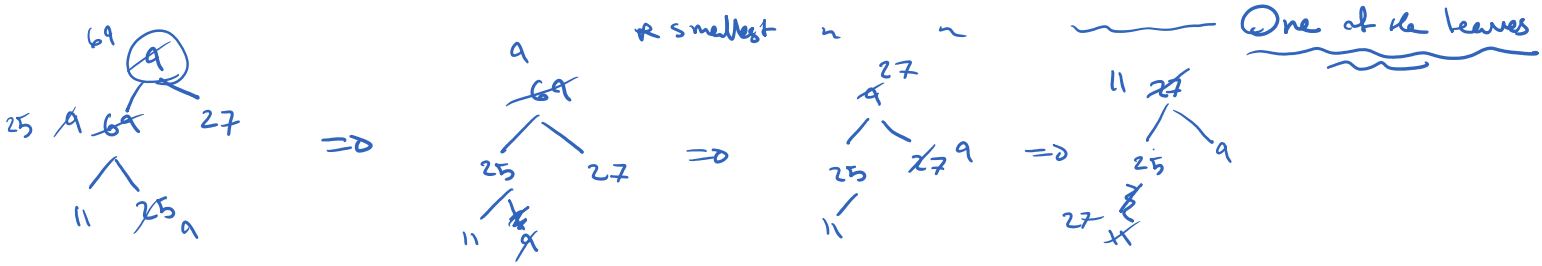
② Keep removing the roots until the heap becomes empty → O(nlogn)

1	9	11	25	27	69	420

sorted ✓



* largest element in a max-heap is the root
* smallest " " " " One of the leaves



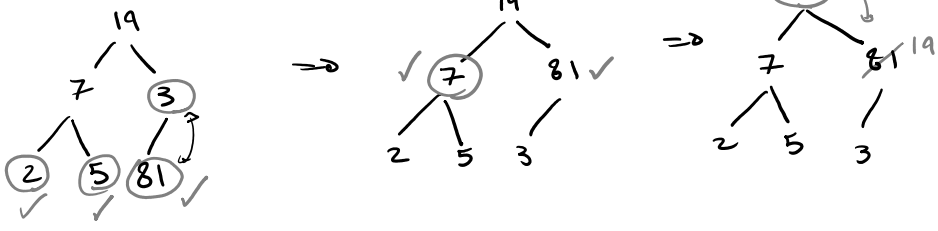
Step2: running time

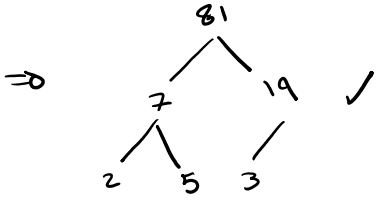
$$T(n) = \log n + \log(n-1) + \log(n-2) + \cdots + \log 1$$

$$= \sum_{i=1}^{n} \log i \le O\left(\int_1^n \log x\,dx\right) = \boxed{O(n \log n)}$$

Better version to build max-heap:

$a = [19, 7, 3, 2, 5, 81]$





① Place all the nos into a Binary Tree (Skip for Lab)

② Start calling max-heapify from the last node to the root!

last internal node (your hw 7 to find its index)

Build-max-heap (a)

Andrew

```
Build_max_heap(a)
int n= a.length
int start= (n/2)-1
for start: 0
  max_heap(a, i)
```

for start: -1 : 1