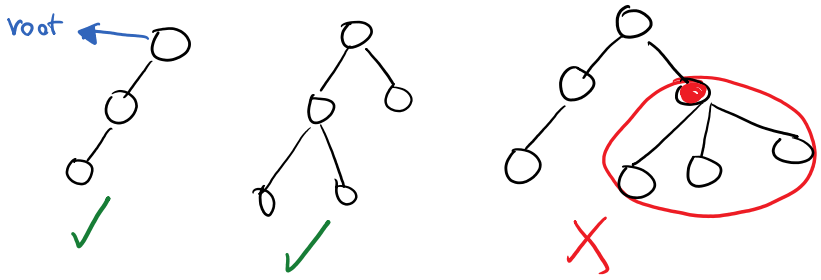
Binary Heaps

Tuesday, October 27, 2020 5:00 PM

Reminder: Lab 4 is due this Sunday.
(2 files: 1 code / 1 description of running time)

Tree: ① connected ② no cycle

Binary Tree (BT): each vertex has at most 2 children

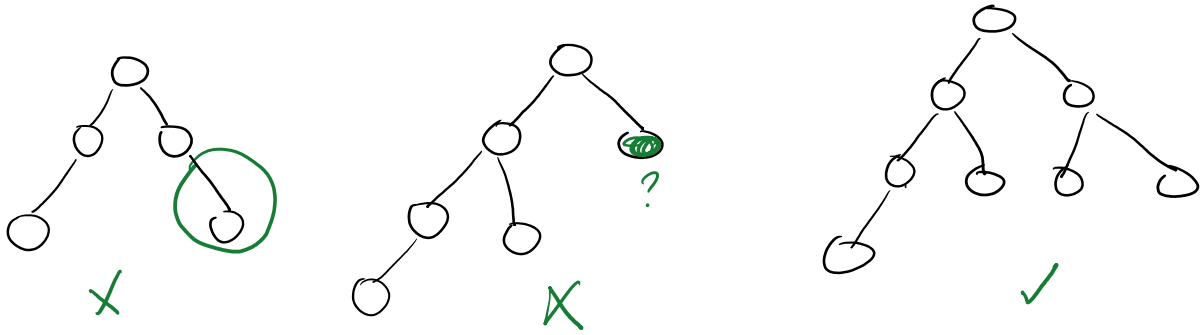


root: has no parent

internal node: has children

leaf node: has no children

Complete BT: all the levels are completely full except possibly the last one

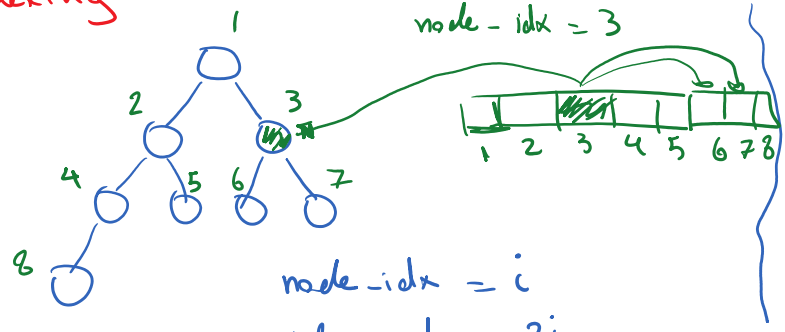


Binary Heaps: They are "complete binary trees."

* you always add the nodes from left to right

* the root has the smallest index

-indexing

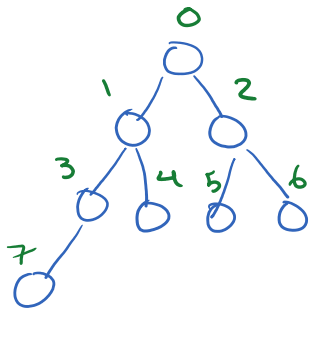


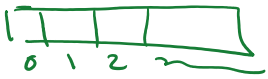
node_idx = i

childleft_idx = 2i

childright_idx = 2i+1

or



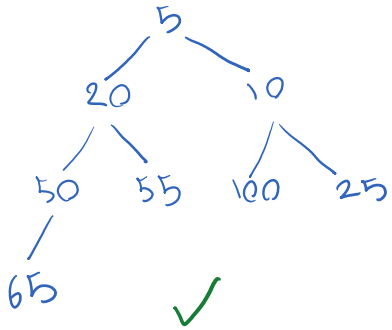


node_idx = i

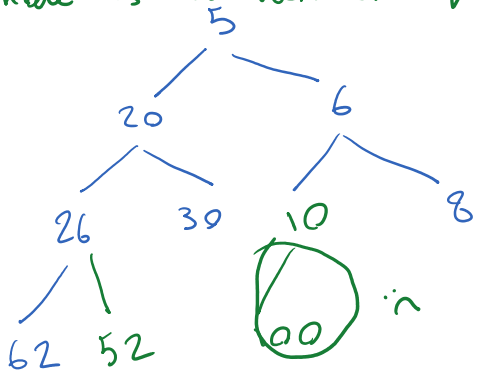
child Left_idx = 2i+1

& right_idx = 2i+2

Min heaps: → value at each node is less than or equal to the children



✓



in

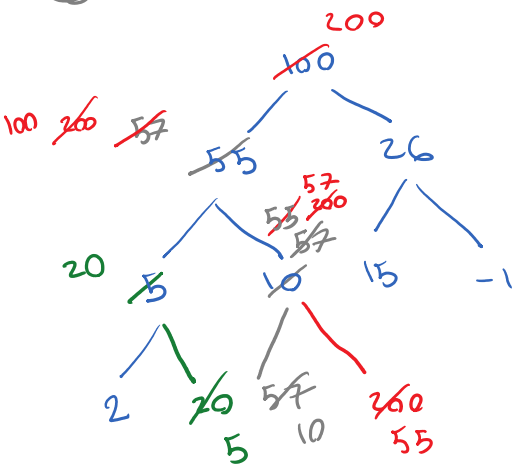
✗ This is not a complete tree

Max heaps: the value for each node is greater or equal to the children.

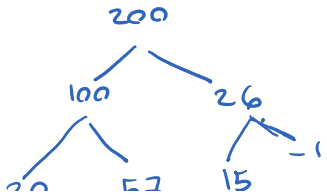
Insertion:

Example: ① Insert 20

② Insert 57



⇒

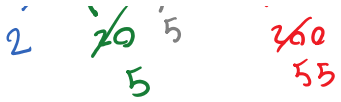


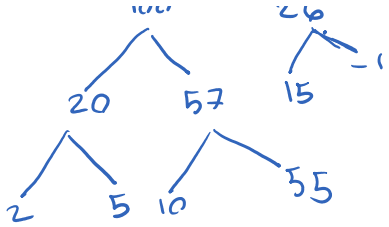
if I have parent

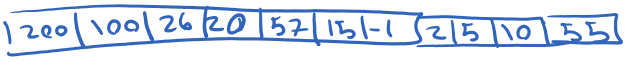
① Compare the value of the new node with its parent

② if value is smaller exit

③ if value is larger swap. and go back to step 1







Question: What is the time complexity of inserting one element in a max-heap with n elements.

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

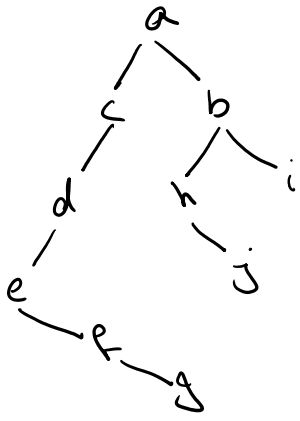
h: height

$\log n$: your hw 7

---

height: the length of the longest path from the node to one of the leaves

depth: the length from the node to the root



tree

$h(T) = 5$

$h(b) = 2$

$h(d) = 3$

$h(j) = 0$

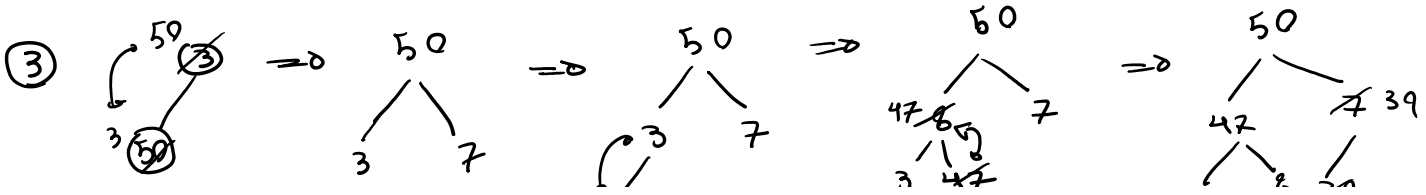
$h(f) = 1$

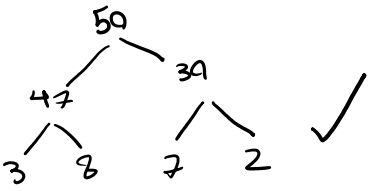
$depth(f) = d(f) = 4$

Build max_heap:

① Successive Insertions. $O(n \lg n)$

Example: ~~3~~, ~~50~~, ~~7~~, ~~8~~, ~~47~~, 39, 2





nr of nodes		height
n	→	$\log n$
7	→	$\log 7$
2	→	$\log 2$
i	→	$\log i$

To insert one element if I have n nodes $T(n) = O(\log n)$

running time to build:

$$T(n) = \log 1 + \log 2 + \log 3 + \dots + \log n$$

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

Lab 4:

$a = [2, 0, 6, 5, 100, -50]$ find the 3 closest nrs to M

→ I am not allowed to sort

$O(1)$ ① find median: 5 $(-50 \;\; 0 \;\; 2 \;\; 5 \;\; 100 \;\; 6)$

↳ call quick-select $(a, \frac{|a|}{2}, 0)$ → old partition

② find k closest nrs to M: $[0 \;\; 2 \;\; 6]$

O(?) (2.1) diff = (-45, -5, -3, 0, 95, 1)

Sol 2 → ✗ |diff| = (45, 5, 3, 0, 95, 1)

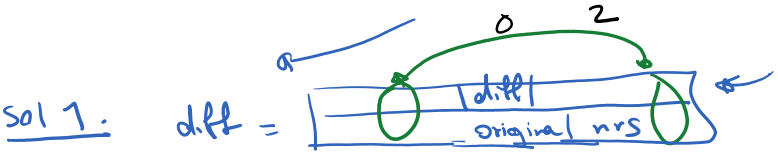
abs new partition like part B at lab 3

O(?) (2.2) call quick-select (diff, 3, 1)

O(?) (2.3) 5 3 1

Sol2. -5 -3 1 → 0 26

O(?) (2.4) shift them back (+n) → 0 ~~0~~ ~~2~~, 6



Sol 1. diff =

* Sol2 : make a abs-partition

↑ new function

while/if abs(a[left]) < abs(pivot)

left++

while/if abs(a[right]) > abs(pivot)

right--

quick-select

if flag // ==1

| new abs-partition

else

| partition

end