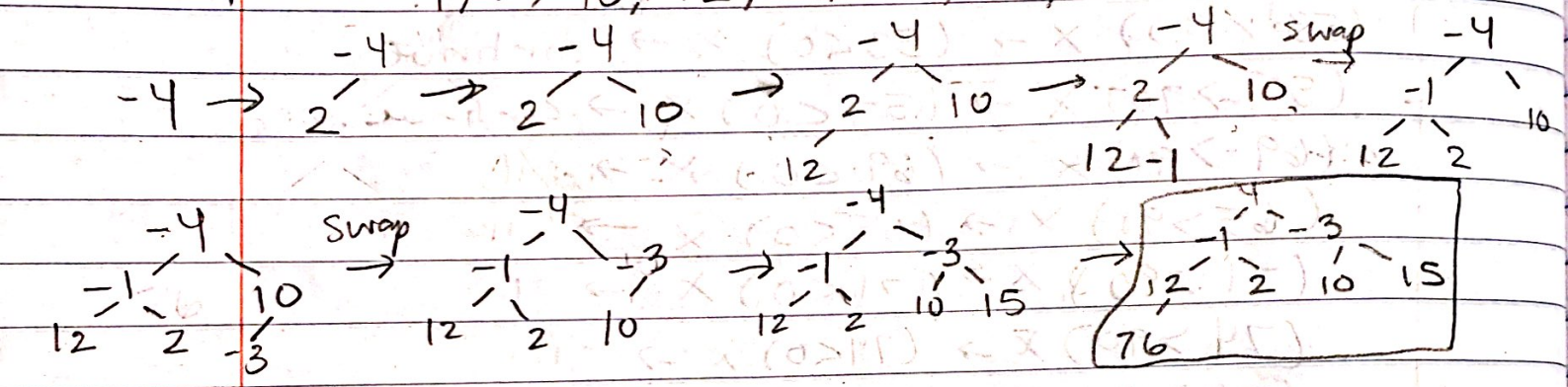
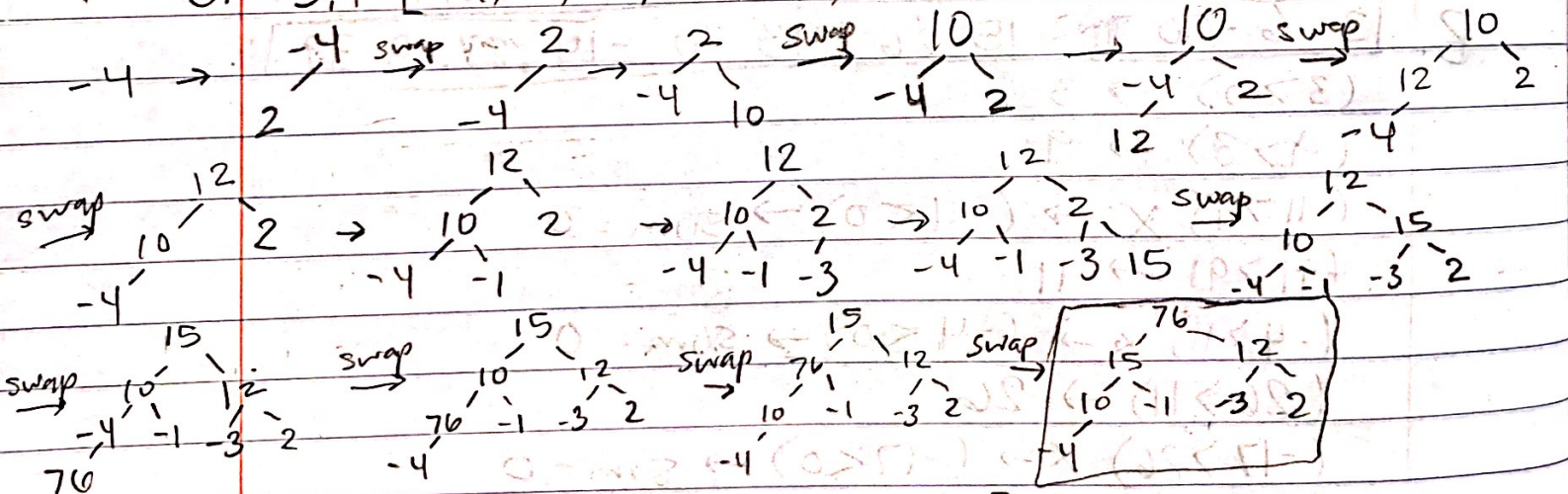
328 HW #7

1. The largest element in a min-heap must reside as one of the leaves, while the root is the smallest element and the further you go down, the higher the numbers will become, especially in this scenario because all elements are distinct & not equal.

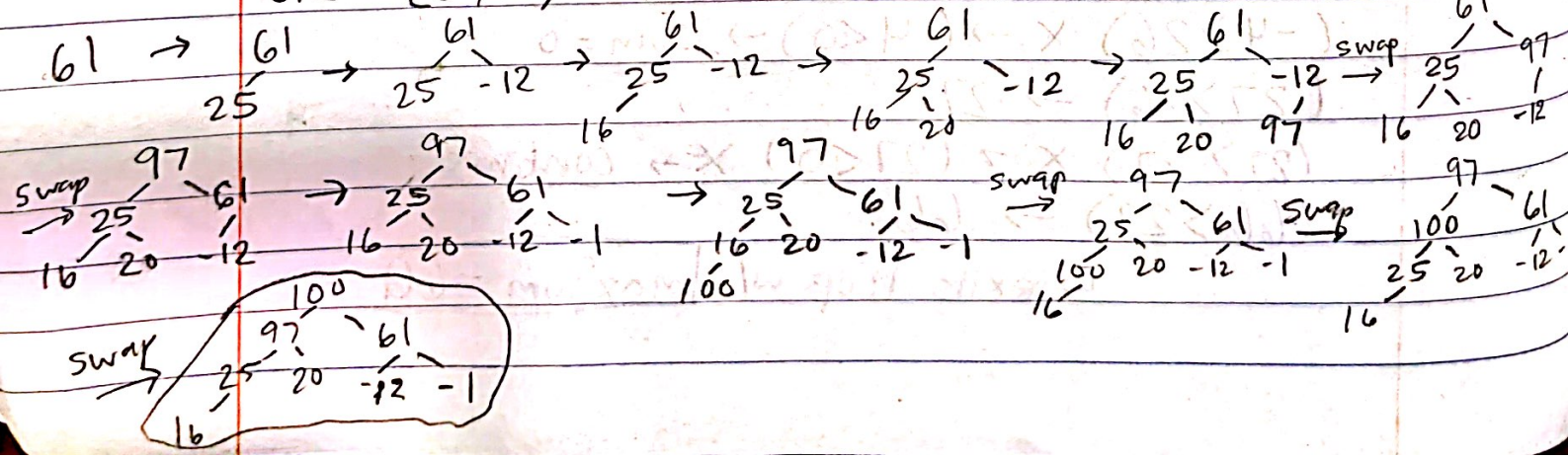
2. -4, 2, 10, 12, -1, -3, 15, 76



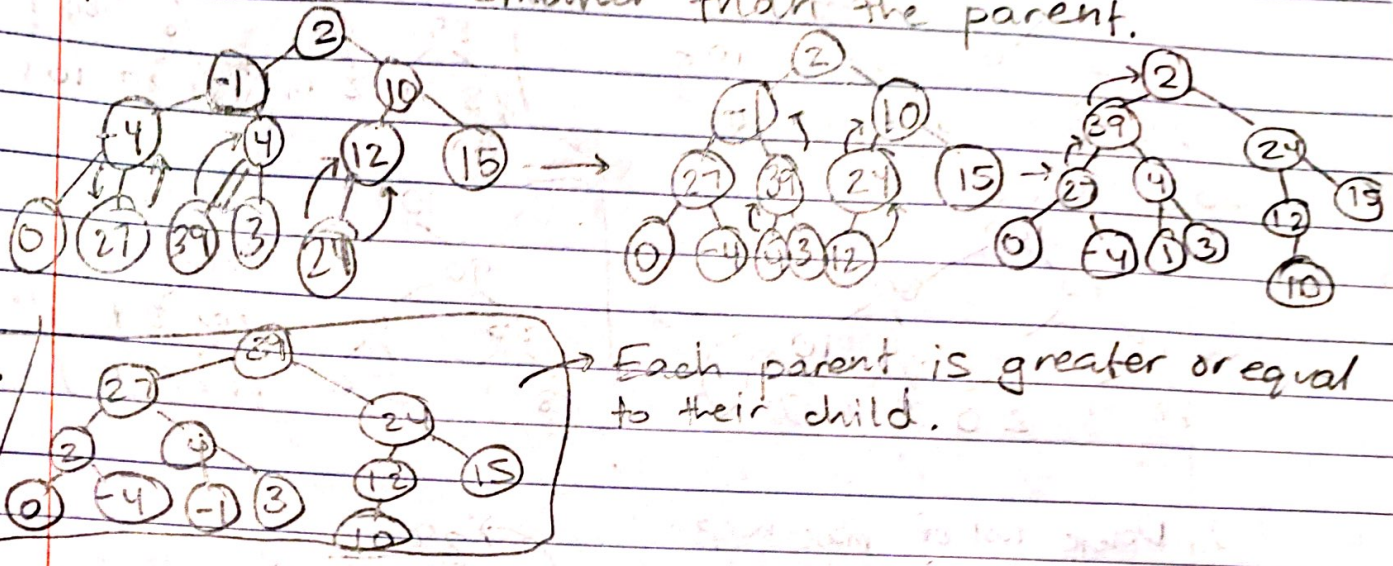
3. 3.1 [-4, 2, 10, 12, -1, -3, 15, 76]



3.2 [61, 25, -12, 16, 20, 97, -1, 100]

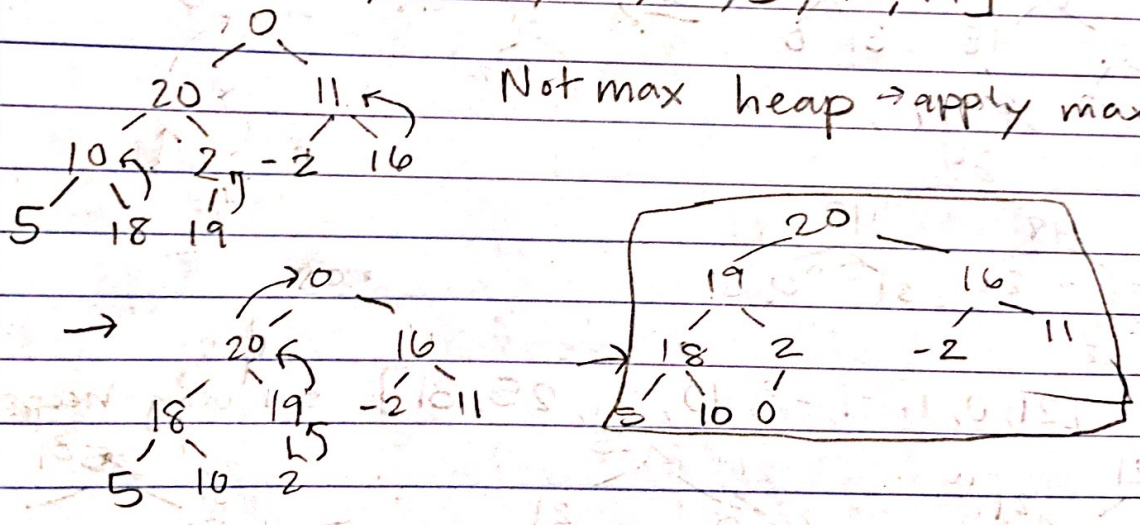


4. The BT is not a max-heap since the root is not the largest number, and the children of each parent aren't smaller than the parent.



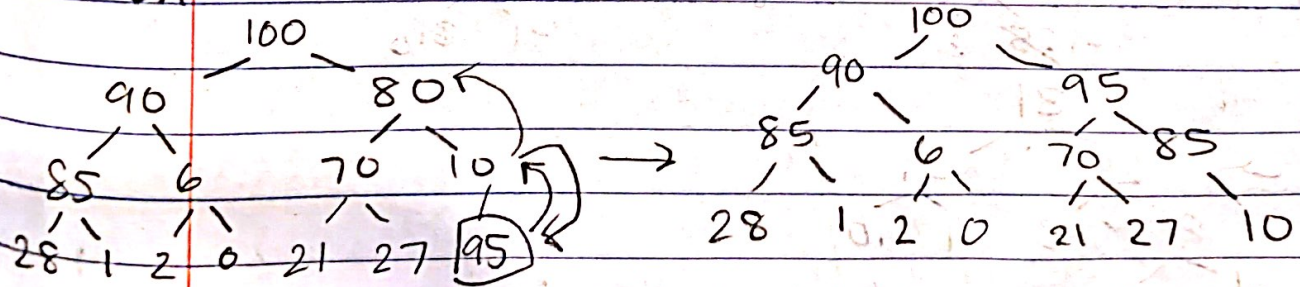
Each parent is greater or equal to their child.

5. [0, 20, 11, 10, 2, -2, 16, 5, 18, 19]

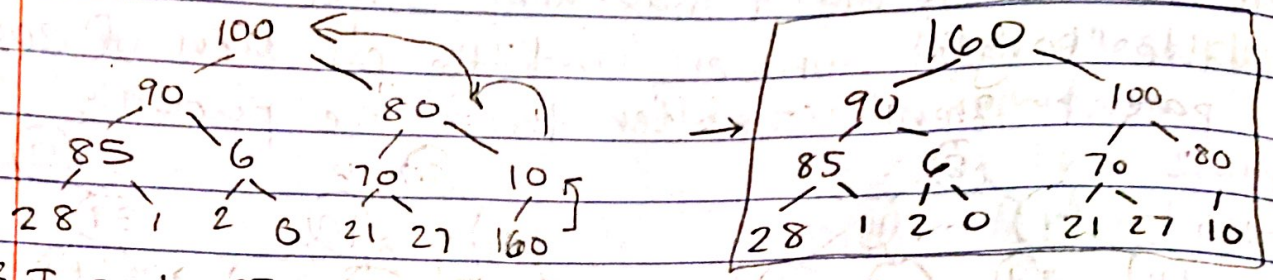


Not max heap → apply max-heapify

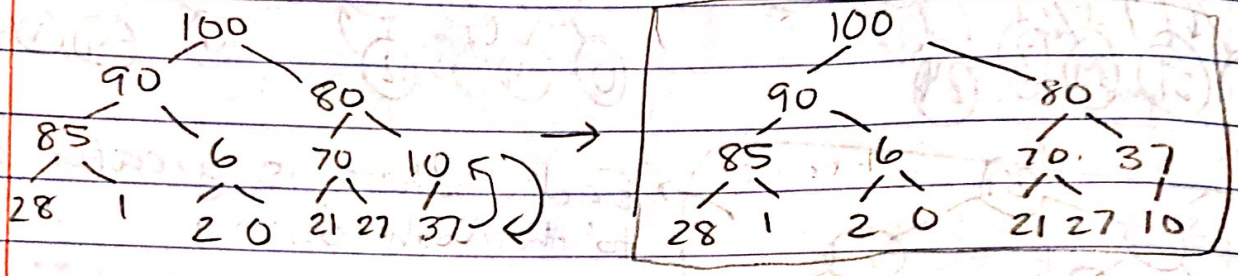
6.1 Insert 95



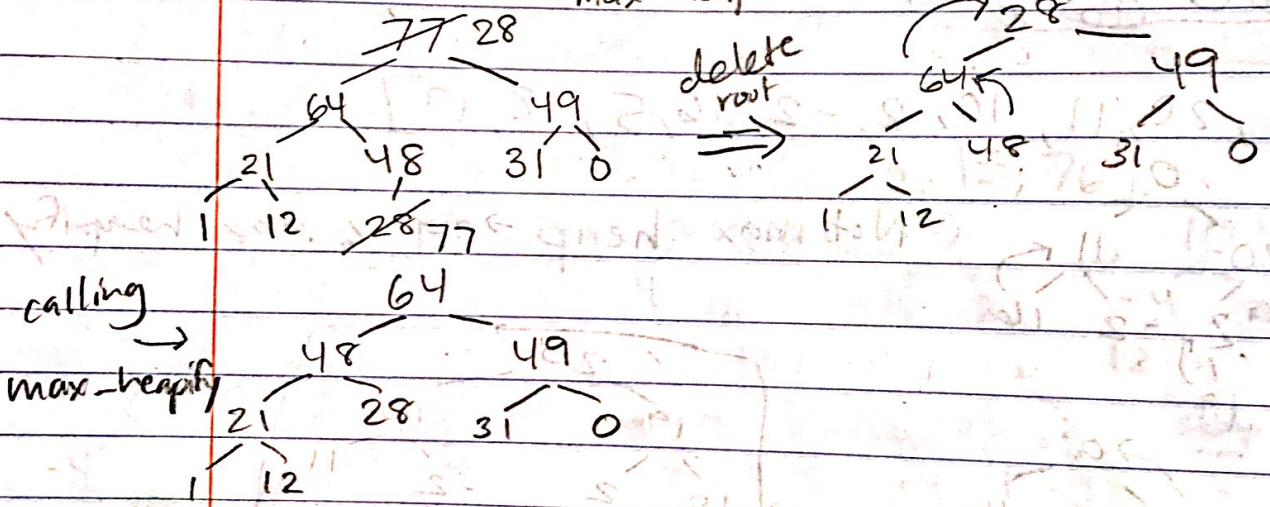
6.2 Insert 160



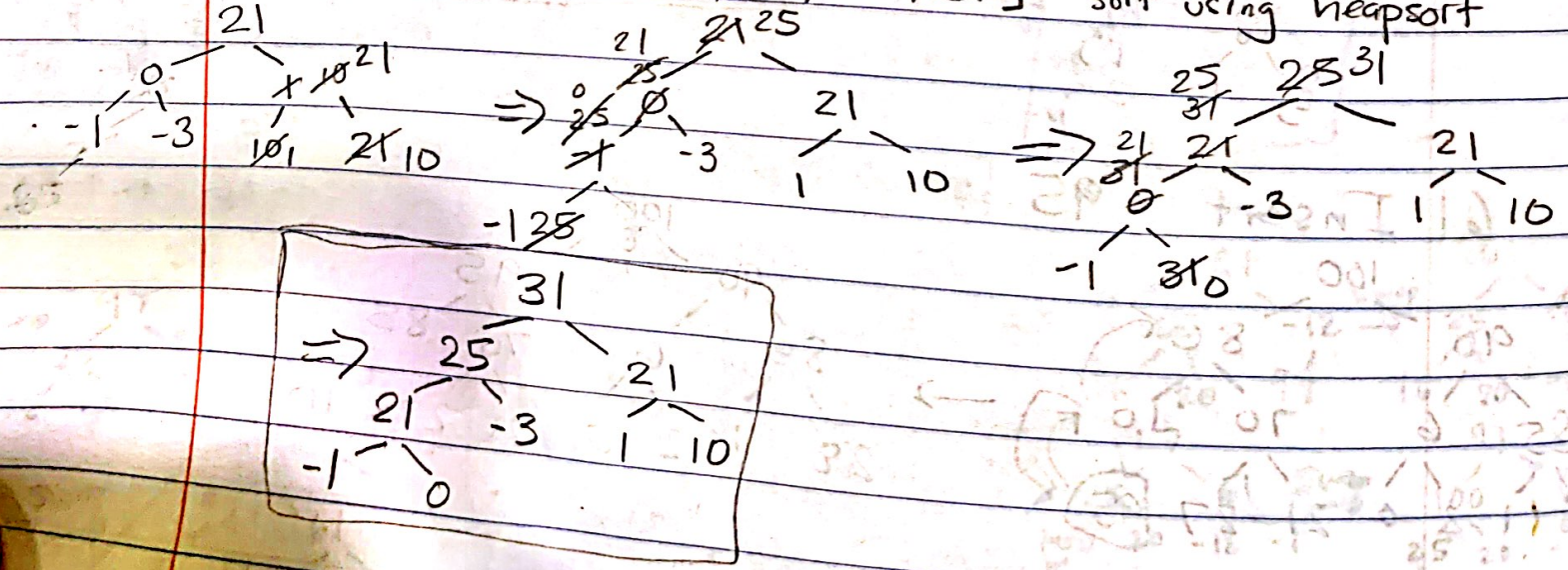
6.3 Insert 37



7. Delete root of max-heap

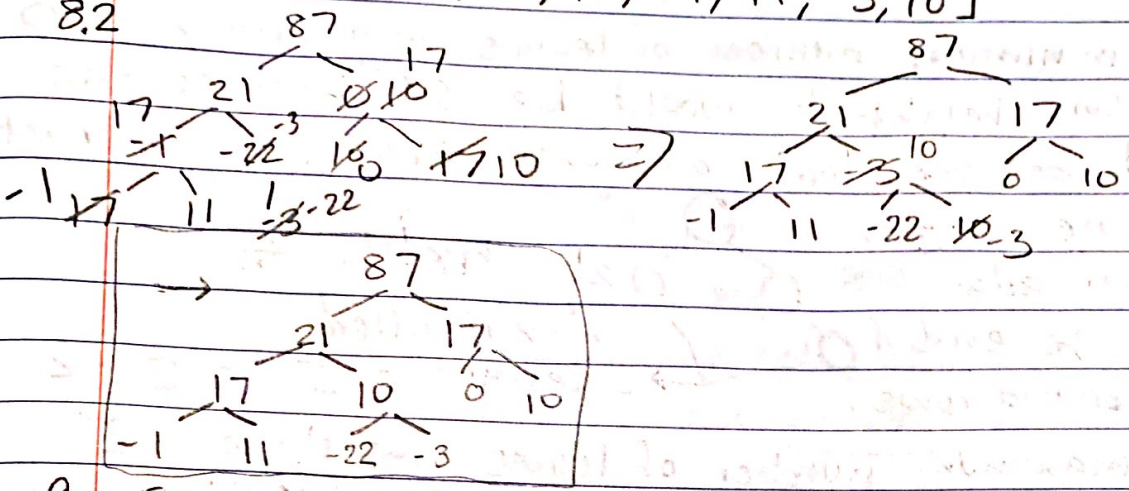


8.1 [21, 0, 1, -1, -3, 10, 21, 25, 31] sort using heapsort

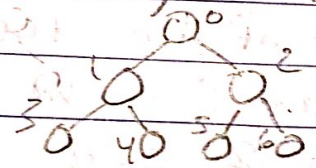


[87, 21, 0, -1, -22, 10, 17, 11, -3, 10]

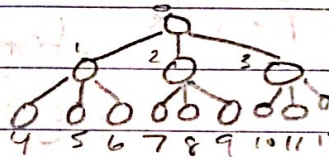
8.2



9. For binary heaps: node index = i
left index = 2i + 1
right index = 2i + 2



For ternary heaps: left = 3i + 1
middle = 3i + 2
right = 3i + 3



10. $1 + 2 + 4 + \ldots + 2^n = 2^{n+1} - 1$

Defining P(n) →

Basis step: $p(0) = 1 + 2 + 4 + \ldots + 2^0 = 2^0 - 1$

Induction step:

$k = 2^{n+1} - 1 = 2 - 1 = 1$ for $n = k + 1$

so n would be $1 + 2 + 4 + \ldots + 2^k + 2^{k+1}$

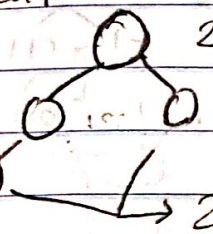
$\rightarrow 2^{k+1} - 1 + 2^{k+1}$

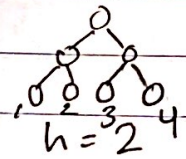
$\hookrightarrow 2 \cdot 2^{k+1} - 1 \rightarrow 2^{k+2} - 1$

This is the same as $p(k+1) = 2^{k+1+1} - 1 = 2^{k+2} - 1$

∴ Through the induction step we proved that since the base step is true, and p(k+1) hence is also true.

11. The minimum number of leaves in a binary heap that has height h would be 2^{h-1} because this would be one leaf at lowest height. By subtracting 1 from the height, we can add one leaf to the end of our last two rows.

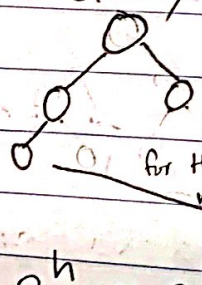




$2^2 = 4$ ↑ for max

The maximum number of leaves would be 2^h since we do all the previous leaves (2^{h-1}) + from the minimum and add the rest of the leaves that were lost from "-1." Other than that, we can mentally figure out that it should be 2^h since each parent gains 2 children during a max binary completed tree

12. Prove binary heap w/ n elements has height $\lfloor \log n \rfloor$



$$\sum_{i=0}^{h-1} 2^h \rightarrow 2^h + 1 \rightarrow (2^{(h-1)+1} - 1) + 1 \rightarrow n = 2^h$$

$$\therefore h = \log_2 n$$

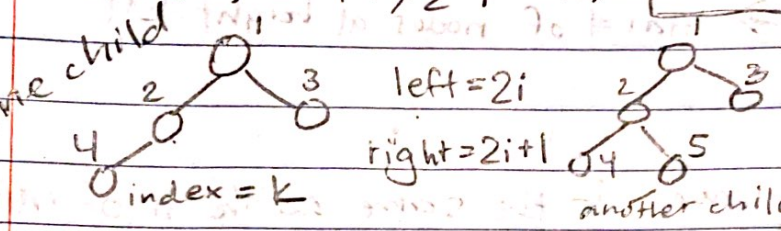
Since we're looking at the floor of log n, we can look at the minimum nodes summation equation. Rearranging and solving the summation we find that with the log property, that h is log n for n elements.

13. Binary heap with n nodes has exactly $\lceil n/2 \rceil$ leaves.

Induction:

P(n) would be the amount of leaves in bin. heap and given that our base case is $n=1=\lceil 1/2 \rceil$, we know that a bin. heap with only one node, or root would also have one leaf, → 0 ↳ root/leaf

a. ↳ $P(k) = \lceil k/2 \rceil \Rightarrow \boxed{n=k+1}$



b. The left nodes would be even, while right would be odd.

Parent's index = k-1

Index = k → last child

c. After adding one child to the left (even) we see that the index would equal to k, or the number of nodes. Meanwhile, when we add another child next to it, we get k+1 nodes.

d. Based on $\lceil \frac{k}{2} \rceil$ leaves → $\lceil \frac{k}{2} \rceil + 1 \rightarrow \lceil \frac{k+2}{2} \rceil$

$(k+1) = k$

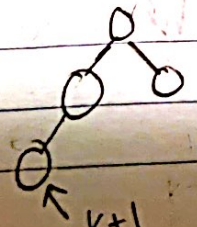
$\lceil \frac{k+2}{2} \rceil = \lceil \frac{k+1}{2} \rceil$ because when you add another node, k would turn into "k+1."

Therefore, in this equation k is equal to k+1 allowing it to be equal to $\lceil k+1+1/2 \rceil$. This is similar to proving induction with base case then the next case which would be P(k+1).

When k is even, the # of leaves will increase by 1. However, when k is odd, or right child, we see that the number of leaves stays the same.

e. $\lceil k/2 \rceil \rightarrow \lceil 4/2 \rceil = 2$ leaves

Diff for internal nodes.



→ Although we add another node and the number of nodes is increased by 1, our # of leaves remains the same in this case (2).

14. $\left\lceil \frac{n}{2^{h+1}} \right\rceil$ nodes of height h in any n-element heap.

Basis step: $\left\lceil \frac{n}{2^{0+1}} \right\rceil = \left\lceil \frac{n}{2} \right\rceil$ (doing $p(0)$)

Inductive step:

(I) Assuming the height is equal to # of nodes (k) having height $k = \left\lceil \frac{n}{2^{k+1}} \right\rceil$

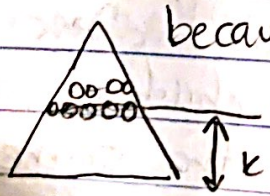
(II) Proof: $h = k+1 \Rightarrow$ total # of nodes at height $k+1$ is $\left\lceil \frac{n}{2^{k+2}} \right\rceil$

Since the # of leaves is the same as the # of internal nodes, we can also establish that

$$\#\text{ of leaves/internal} = \left\lfloor \frac{n}{2^{k+1}} \right\rfloor$$

If we look at base case, we see that when removing leaves, the # of leaves and internal nodes would be $\lceil n/2 \rceil$ & $\lfloor n/2 \rfloor$ but after removing all the leaves, we get $\lceil \frac{n}{2^2} \rceil$ & $\lfloor \frac{n}{2^2} \rfloor$ since the total nodes would change from $n \to \lfloor n/2 \rfloor$.

Internal nodes would be $\left\lceil \frac{n}{2^{k+2}} \right\rceil$ bc k, or the height until the leaves is $k+1$ and when we go one layer above that we get the "2^{k+2}". So basically, we proved that at height $k+1$, the total nodes is $\lceil n/2^{k+2} \rceil$ because we go a layer above the leaves.



Used these drawings from lecture to help w answer.