

328 HW #7

- I. 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

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

$$-4 \cancel{+ 4x^2} - 4 \cancel{+ 4x^2} (32 - 4) \rightarrow x (-4) \xrightarrow{\text{swap}} -4$$

$$-4 \rightarrow 2' \rightarrow 2 \leftarrow 10 \rightarrow 2 \leftarrow 10 \rightarrow -2 \leftarrow 10, \quad -1 \rightarrow 10$$

$$\begin{array}{r} -4 \\ \times 12 \\ \hline -1 \quad 10 \\ \hline -3 \end{array} \xrightarrow{\text{Swap}} \begin{array}{r} -4 \\ \times 12 \\ \hline -1 \quad 2 \quad 10 \\ \hline -3 \end{array} \rightarrow \begin{array}{r} -4 \\ \times 12 \\ \hline -1 \quad 2 \quad 10 \\ \hline -3 \quad 15 \end{array} \rightarrow \boxed{\begin{array}{r} -1 \quad 2 \quad 10 \\ \hline 12 \quad 2 \quad 15 \end{array}}$$

3.

$$3. \quad 3, 1 \left[-4, 2, 10, 12, -1, -3, 15, 76 \right]$$

The diagram illustrates the execution of the bubble sort algorithm on the array [2, -4, 10, 2]. The array is shown at four stages of the sort:

- Initial State:** [2, -4, 10, 2]
- After 1st Pass:** [-4, 2, 10, 2] (swap between 2 and -4)
- After 2nd Pass:** [-4, -4, 10, 2] (swap between 2 and -4)
- After 3rd Pass:** [-4, -4, 2, 10] (swap between 10 and 2)

Annotations above the array show the swaps: "swap" between 2 and -4 for the first two passes, and "swap" between 10 and 2 for the third pass.

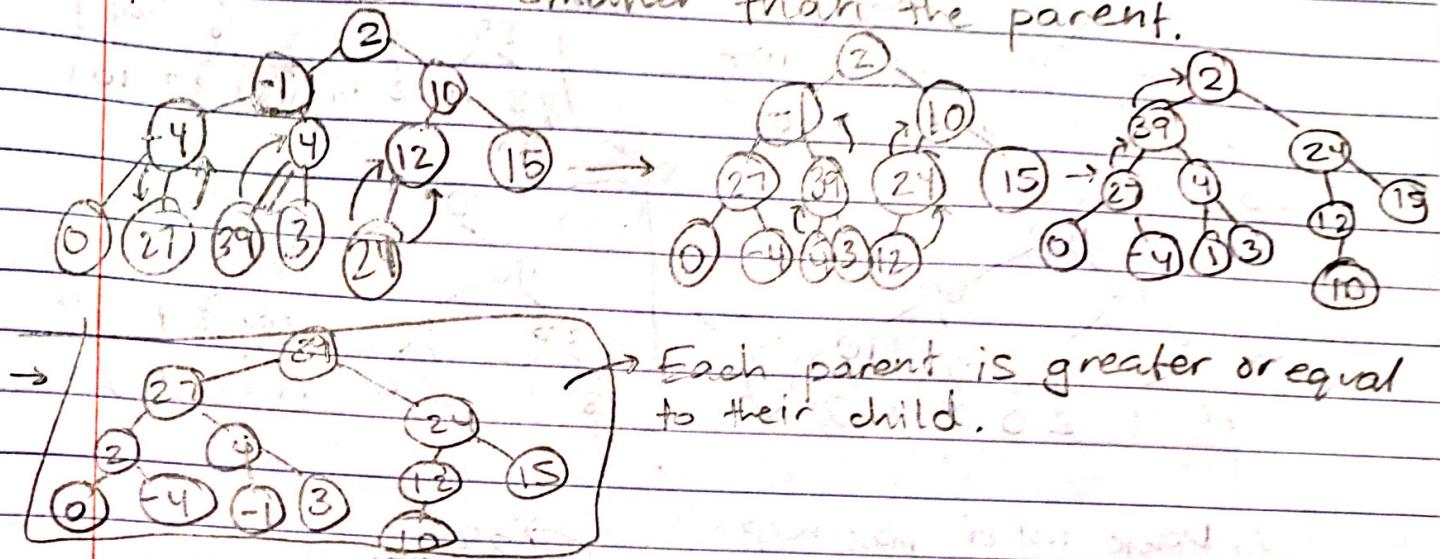
3.2 [61, 25, -12, 16, 20, 97, -1, 100]

The diagram illustrates the bubble sort algorithm on the array [97, 61, 25, 16, 20, -12, -1]. The steps show the array after each pass through the elements, with swapped elements highlighted in red.

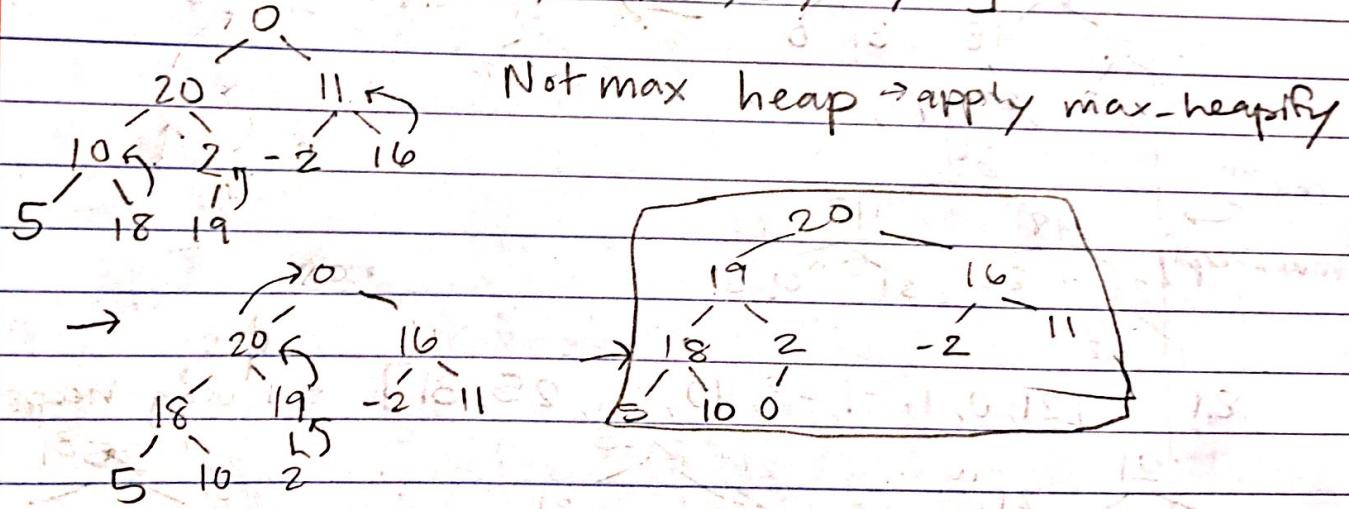
- Initial State:** [97, 61, 25, 16, 20, -12, -1]
- Pass 1:** Swaps 97 and 61. Array: [61, 97, 25, 16, 20, -12, -1]
- Pass 2:** Swaps 97 and 25. Array: [25, 61, 97, 16, 20, -12, -1]
- Pass 3:** Swaps 97 and 16. Array: [16, 25, 61, 97, 20, -12, -1]
- Pass 4:** Swaps 97 and 20. Array: [16, 20, 25, 61, 97, -12, -1]
- Pass 5:** Swaps 97 and -12. Array: [16, 20, -12, 25, 61, 97, -1]
- Pass 6:** Swaps 97 and -1. Array: [16, 20, -12, -1, 25, 61, 97]

A hand-drawn graph on lined paper. A downward-sloping curve is drawn, starting at the top left and ending at the bottom right. The word "SWAY" is written vertically along the left side of the curve. Above the curve, there are numerical values: 100 at the top, 97 near the middle, 61 at the end, -12 below 61, and -1 at the far bottom right. A vertical red line is drawn through the curve, intersecting it at approximately x = 50.

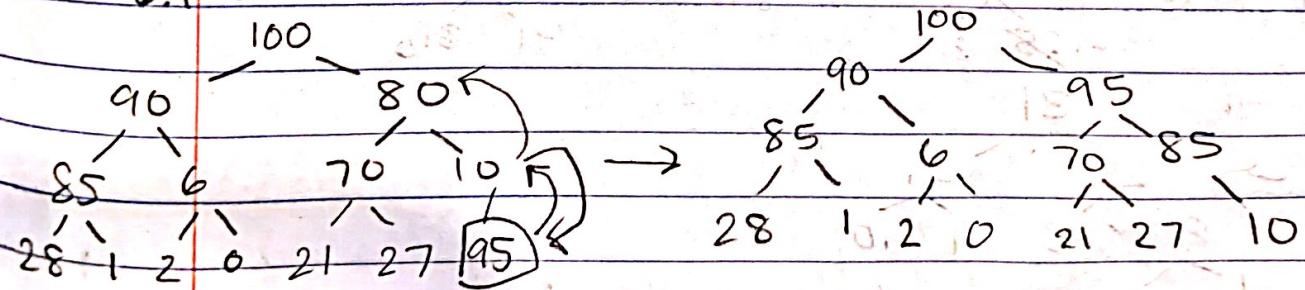
4. The BT is not a max-heap since the root is not the largest number, and the children of each parent aren't smaller than the parent.



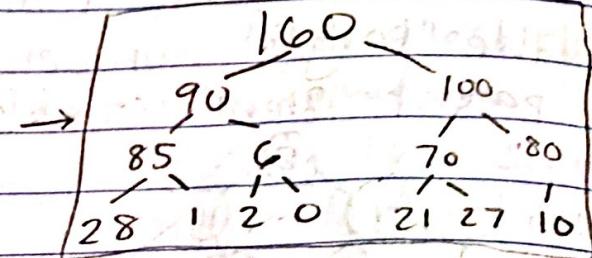
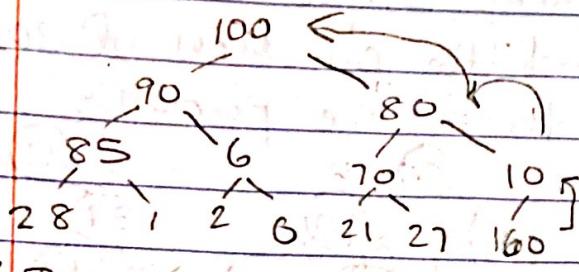
5. [0, 20, 11, 10, 2, -2, 16, 5, 18, 19]



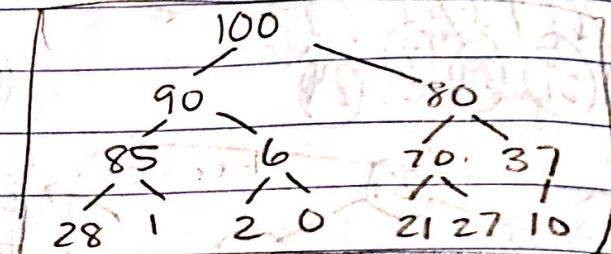
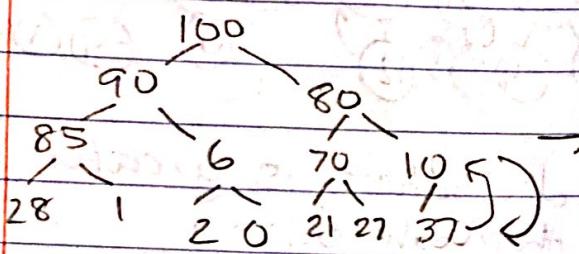
6.1 Insert 95



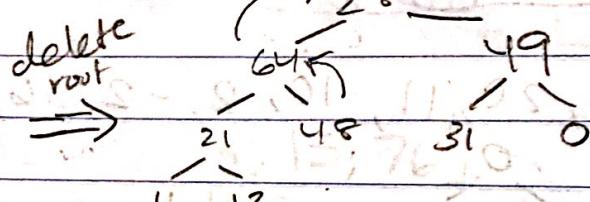
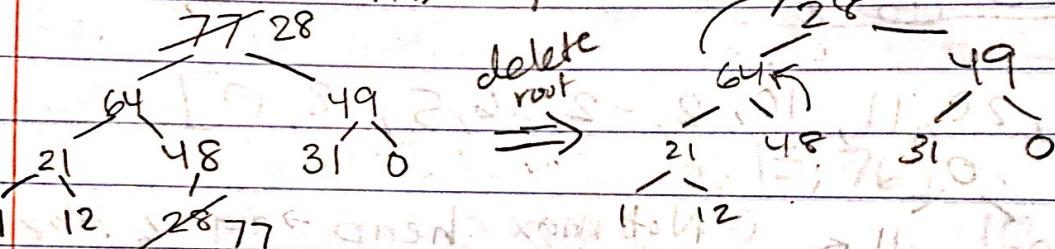
6.2 Insert 160



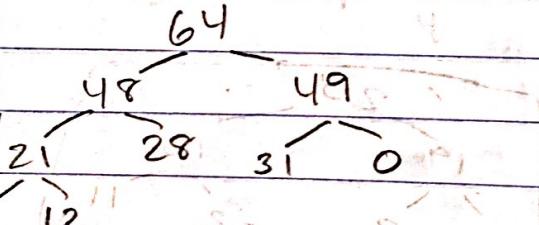
6.3 Insert 37



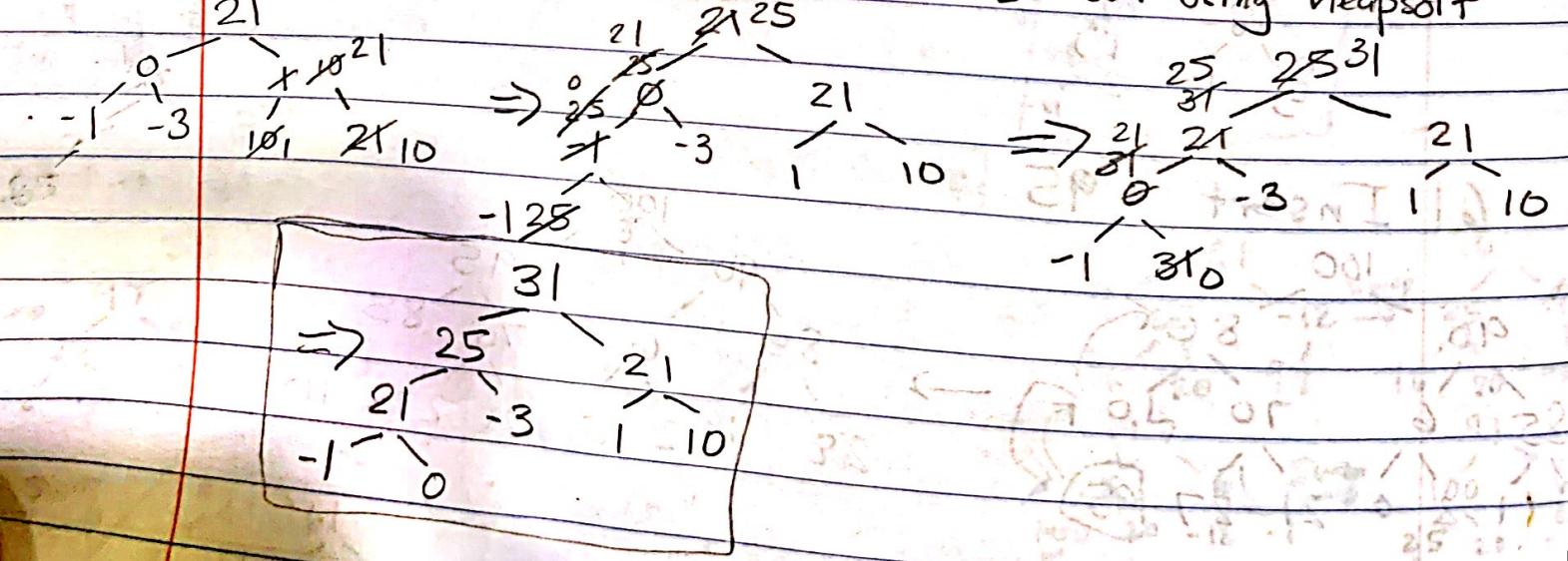
7. Delete root of max-heap



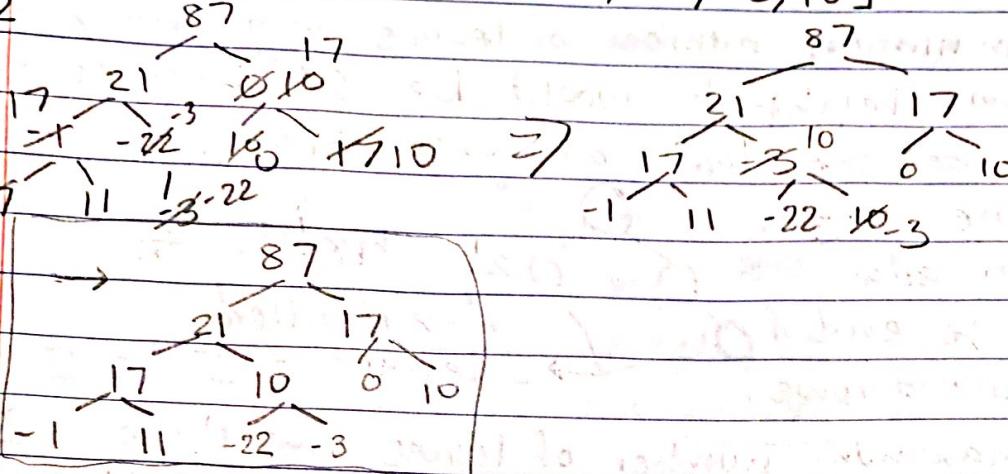
calling
max-heapify



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



8.2

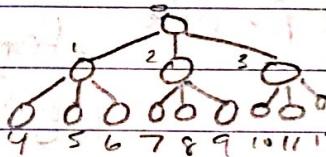
 $[87, 21, 0, -1, -22, 10, 17, 11, -3, 10]$


9. For binary heaps: node index = i

$$\text{left index} = 2i + 1$$

$$\text{right index} = 2i + 2$$

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



$$\text{right} = 3i + 3$$

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

Defining $P(n) \rightarrow$

$$\text{Basis step: } P(0) = 1 + 2 + 4 + \dots + 2^0 = 2^0 - 1 = 0$$

Induction step:

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

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

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

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

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

\therefore Through the induction step we proved that since the base step is true, and $P(k+1)$ hence is also true.

[For S-11, S1, OI SS-J-0, 18, 58]

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.

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 $\lceil \log n \rceil$

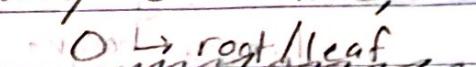
$$\sum_{i=0}^{h-1} 2^h \rightarrow 2^h + 1 \rightarrow (2^{(h-1)+1} - 1) + 1 \rightarrow n \geq 2^h$$

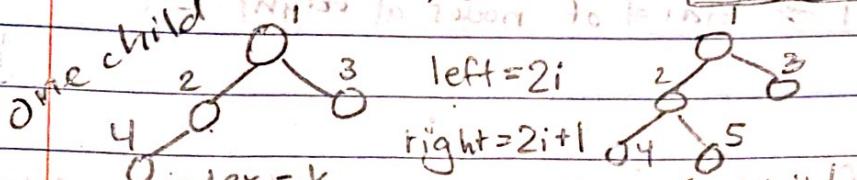
$\therefore h = \log_2 n$

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 have one leaf, \rightarrow 

a. $\hookrightarrow P(k) = \lceil k/2 \rceil \Rightarrow \boxed{n=k+1}$ b. The left nodes would be even, while right would be odd.


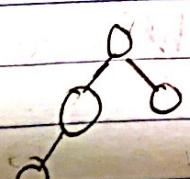
c. After adding one child to the left (even) we see that the index would equal to k , or the number of nodes. Meanwhile, when we add another child next to it, we get $k+1$ nodes.

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

$\lceil \frac{k+2}{2} \rceil = \lceil \frac{k+1}{2} \rceil$ 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.  $\lceil \frac{3}{2} \rceil = 2$ leaves
Diff for interval vals.

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

Inductive step:

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

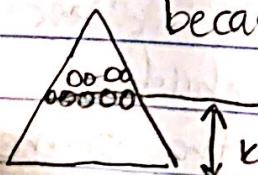
(II) Proof: $h = k+1 \Rightarrow$ total # of nodes at height $k+1$ is $\left\lceil \frac{n}{2^{k+2}} \right\rceil$

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

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

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

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 $\left\lceil \frac{n}{2^{k+2}} \right\rceil$ because we go a layer above the leaves.



Used these drawings from lecture to help w answer.

