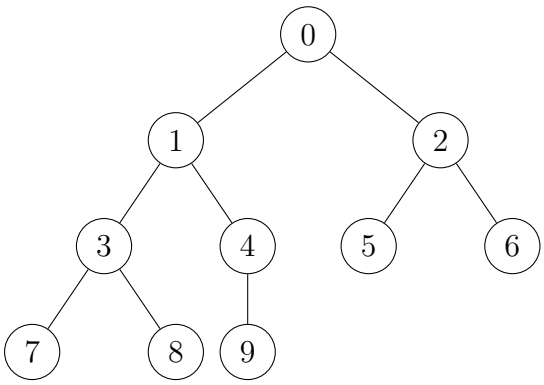
CS 1203: Data Structures **Semester:** Monsoon 2023

Take Home Quiz #1

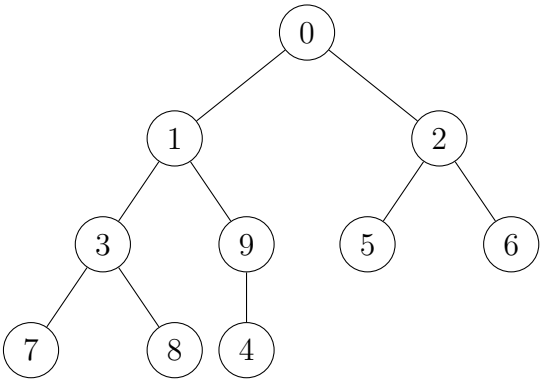
Instructor: Subhamoy Maitra
Due: September 29, 2023

Student Name: Kudakwashe Chakanyuka
kudakwashe.chakanyuka_ug25@ashoka.edu.in

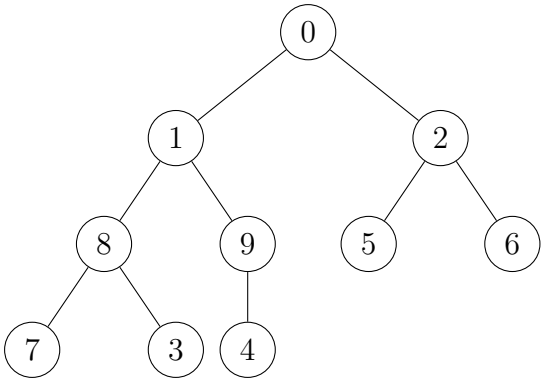
HEAPIFY:



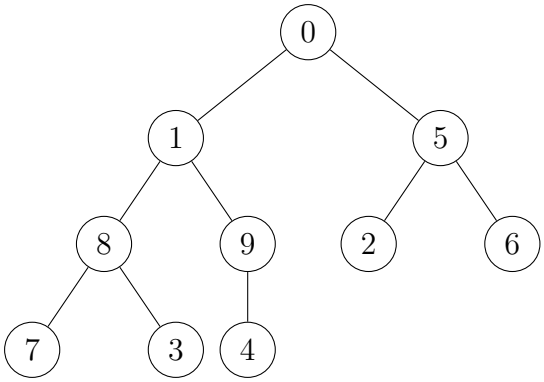
The initial comparison necessitates a position switch between node 9 and node 4.
Giving:



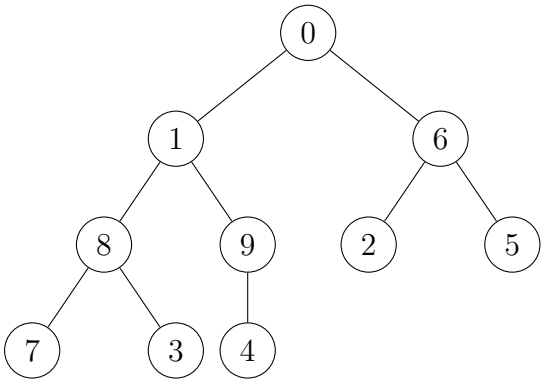
A subsequent switch takes place, transitioning from node 8 to node 3.
Giving:



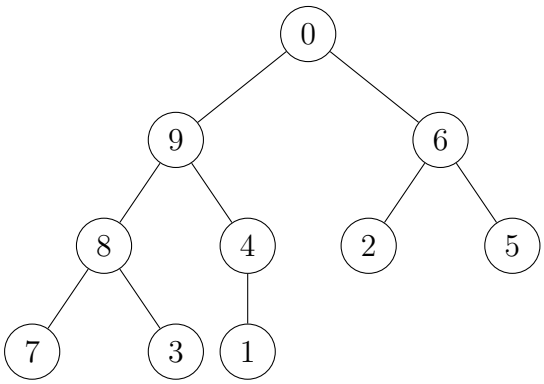
Again, a subsequent switch takes place, transitioning from node 2 to node 5.
Giving:



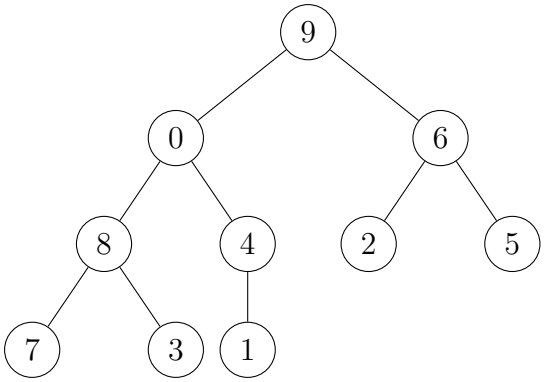
A subsequent switch takes place again, transitioning from node 6 to node 5.
Giving:



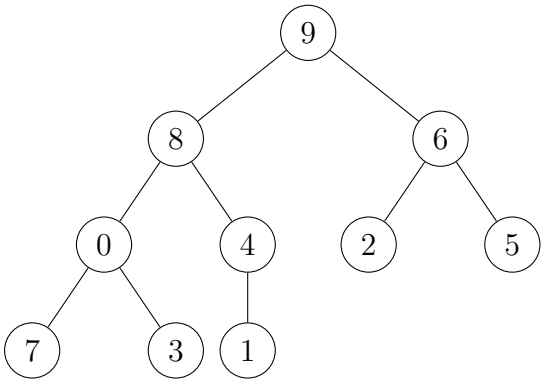
Subsequent switches had to occur sequentially, with node 9 switching to the position of node 1, followed by another switch from node 1 to the position of node 4.
Giving:



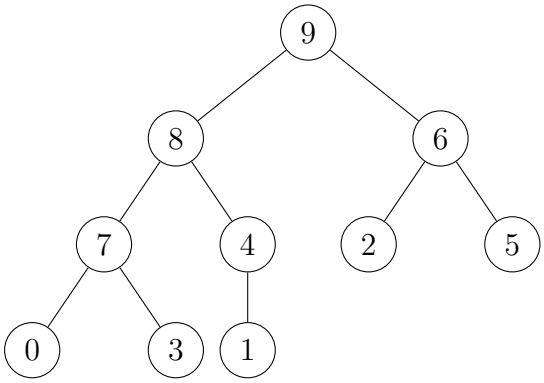
A subsequent switch takes place again, transitioning from node 9 to node 0.
Giving:



A subsequent switch takes place again, transitioning from node 8 to node 0.
Giving:



Lastly, a subsequent switch takes place again, transitioning from node 7 to that of node 0.
Giving:



- The above tree concludes the heapify (bottom-up) method for arr1 which was given as arr1[ ] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}.

- At each step, a node was compared with its children, and a swap was performed if necessary to maintain the max heap property, ensuring that the largest element remained at the root of the tree.

- The total comparisons and swaps for heapify were calculated using the formulas:

Comparisons : O(n)

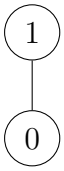
Swaps : O(n)

$\therefore$ the worst-case complexity for heapifying an array using the bottom-up approach (starting from the last non-leaf node) is O(n).

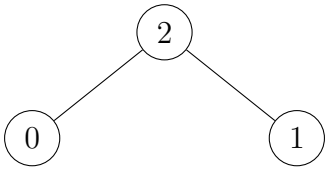
HEAP-INSERT

Each tree begins with a single node and in our case, it can be represented as:

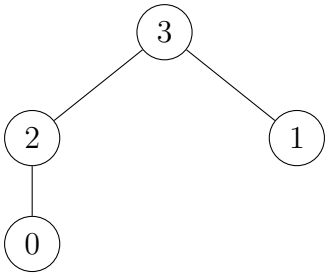
Adding node 1 and performing a comparison, can be illustrated as follows:



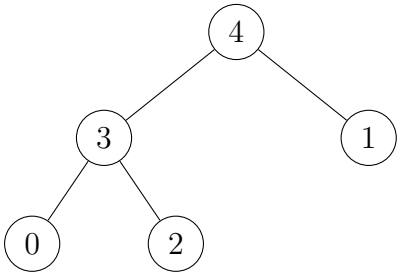
Adding node 2 and performing a comparison can be illustrated as follows:



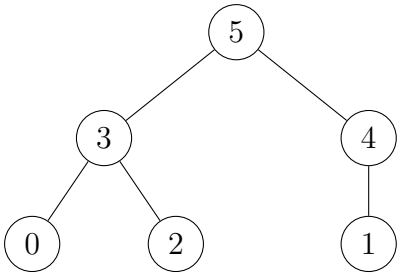
Adding node 3 to the tree and performing a comparison can be illustrated as follows:



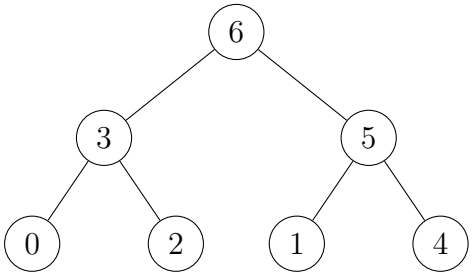
Adding node 4 to the tree and performing a comparison can be illustrated as follows:



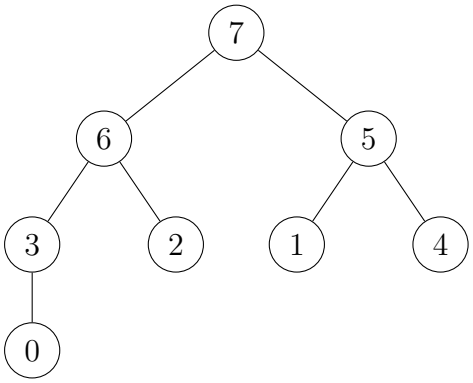
Adding node 5 to the tree and performing a comparison can be illustrated as follows:



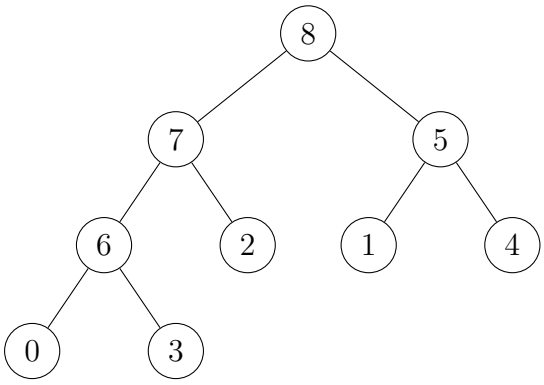
Adding node 6 to the tree and performing a comparison can be illustrated as follows:



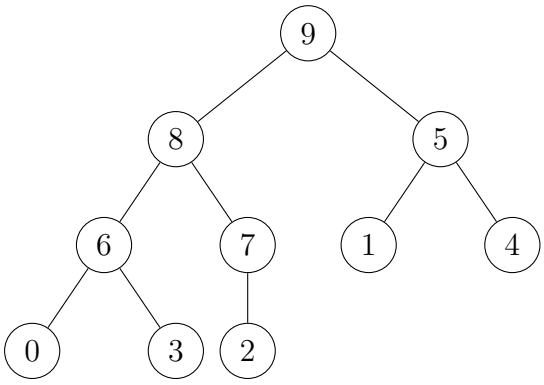
Adding node 7 to the tree and performing a comparison can be illustrated as follows:



Adding node 8 to the tree and performing a comparison can be illustrated as follows:



Adding node 9 to the tree and performing a comparison can be illustrated as follows:



- The above tree concludes the heapinsert (top-down) method for arr2 which was given as arr2[ ] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}.

- Moving each element into the heap and moving it up the tree as needed.

- In each insertion, there was a compare and swap that was conducted.

- The total comparisons and swaps for heapinsert were calculated using the formulas:

Comparisons : O(n log(n))

Swaps : O(n log(n))

$\therefore$ the worst-case complexity for inserting elements one by one into an empty heap, the worst-case time complexity is O(n log n).