AVL Tree Analysis

What is AVL tree?
It is a _____ BST, where the height of the left and right subtree can differ by at most _____ for each node **(recursively)**.

0. Time complexity of AVL tree: Let h be the height of a AVL tree.
* What is the cost of rotation?
1) Find:
2) Insert:
3) Delete:

1. Let's formally prove this.
h=f(n). Let f(n) be the tree height given number of nodes n.
We must find an _____ bound for f(n).

$n=f^{-1}(h)$. Let $f^{-1}(h)$ be the number of nodes given tree height.
We will first find a _____ bound for $f^{-1}(h)$. That is the lower bound of n given h.

N(h) be the minimum num of nodes in AVL tree given height h.
Write the recurrence equation of N(h).
What is the base case?

2. Theorem: An AVL tree with height h has at least $2^{h/2}$ nodes.
Proof by induction.

Base case: An AVL tree with height _____ has _____ nodes.

Induction hypothesis:

Proof:

3. $n > 2^{h/2}$

h < //what does this mean? Having "2" x $\log_2 n$ (compared to $\log_2 n$)

$h = O(\log_2 n)$

Binary Tree Analysis

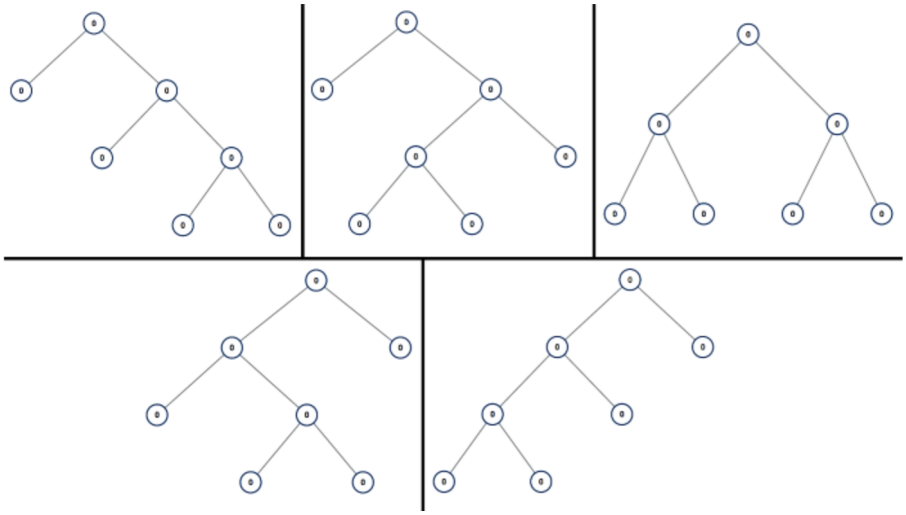
0. _____ of a node refers to the distance of the node from the root, measured in the number of e_____.

1. _____ binary tree
a binary tree in which all the levels are completely filled except possibly the lowest (deepest), which is filled from the left.
[Draw some examples]

1) How many nodes at level k (where k is not the lowest level)?
2) Min number of nodes at lowest level?
3) Max number of nodes at lowest level, where k is the height of the tree?
4) Assume, all the levels are completely filled INCLUDING the lowest level
   What is the total number of nodes in this tree with k levels?

5) Given n is the number of nodes in a complete binary tree, what is the number of levels? (Hint: the closest power of 2 that is >= n)

2. _____ binary tree
a binary tree in which every parent node/internal node has either two or no children.
[examples]



Priority Queue

1. Why do we need Priority Queue?

2. PriorityQueue ADT

1) findMin
2) deleteMin
3) insert
4) isEmpty

3. Here are some options to implement PriorityQueue.
What is the cost of insert and deleteMin? Write them down for each option.

- Unsorted Array

- Unsorted Linked List

- Sorted Circular Array List

- Sorted Linked List

- Binary Search Tree

As you can see, a new data structure is need!

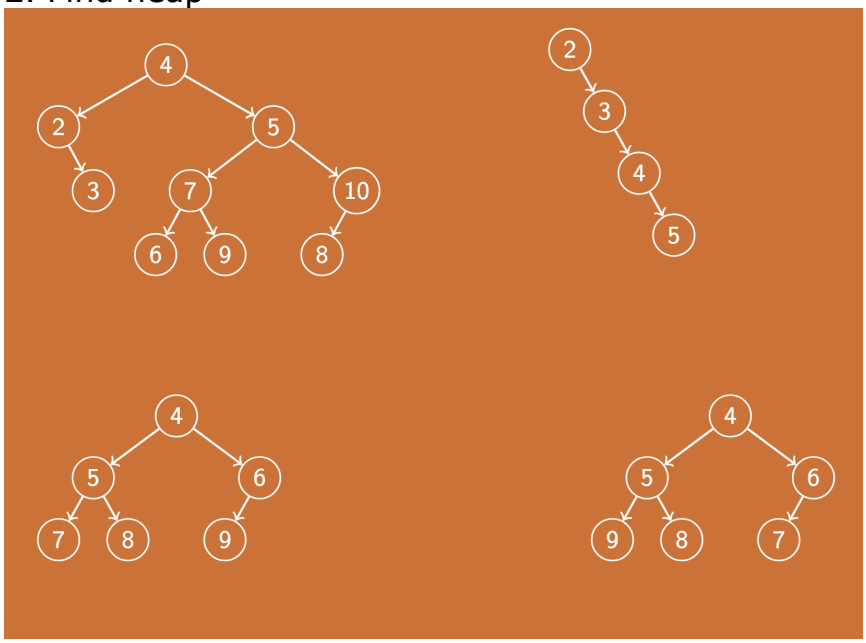
Heap

1. Heap has two properties.

1) _____ property: All c_____ are smaller than p_____.

2) _____ property: A binary tree with no "_____".
What do you call this type of binary tree?

2. Find heap



3. Heap features.

- Where is the minimum item in a heap?
- What is the height h of a heap with n items?

findMin is easy. Let's consider deleteMin and insert.

4. deleteMin in 3 steps.

1) Step 1:

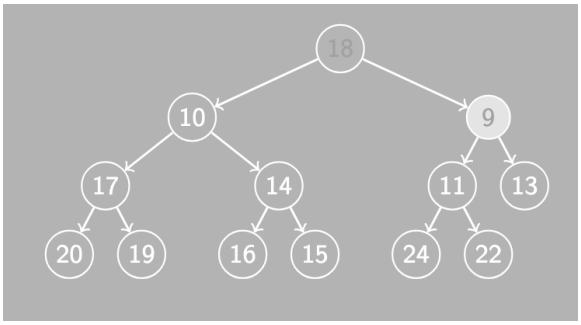
2) Step 2: How to fill the gap?

3) Step 3: "Percolate _____"

5. Pseudo-code of percolate down.

```
percolateDown(node) {
    while(node.data is _________ than ______ child) {
           swap data with _________ child
    }
}
```

6. *PercolateDown example*



Time complexity of percolate down?

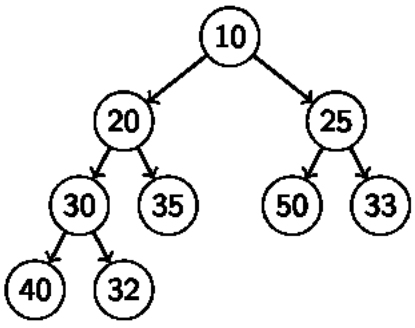
4. insert in 2 steps.
1) Step 1: Where to insert to preserve structure property?

2) Step 2: "Percolate _____" to fix heap property.

5. Percolate up

```
percolateUp(node) {
    while(node.data is ________ than ________) {
        swap data with parent data
    }
}
```

6. Insert example: Insert "2"



7. How to implement heap? Use _____.
Draw the Heap array of 6.

8. Average case analysis of insert. (Count the number of comparisons.)

1) How many nodes are in the last level? _____.
2) How many nodes are in the second to the last level? _____.

Insertion is like finding a "seat" in a heap.
What are the chances that this seat will be at the last level? (That is, we end up inserting at the last level).
How many "comparisons" needed in this case?

What are the chances that this seat will be at the second to the last level? (That is, we end up inserting at the second to the last level).
How many "comparisons" needed in this case?

As an average case, we pick a random value x to insert.

Then the average number of comparisons =

$$\sum_{k=1}^{\infty} Prob\ (Settling\ down\ at\ kth\ to\ the\ last\ level) * k\ comparisons$$

½ * 1 + ¼ * 2 + 1/8 * 3 + …. = $\sum_{k=1}^{\infty} \frac{k}{2^k}$ = Z (Let this be Z).

Z-Z/2 = Z/2 = S =

S – S/2 = S/2 =

Thus S = 1.
Z = 2.

This is O( ). Constant time average case of insert!

9. Other possible operations supported (expand ADT) for PriorityQueue

1) increasePriority

2) decreasePriority

3) buildHeap

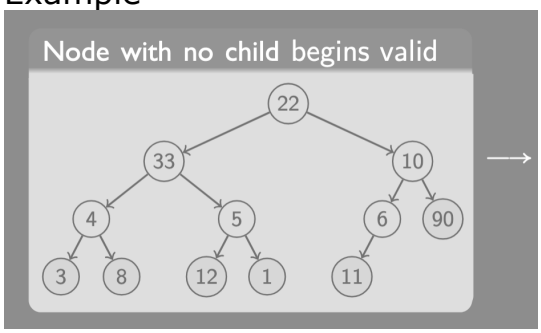
10. Can we build heap more efficiently?
Floyd's algorithm.

Rather than insert one node and fix (percolate up), and then insert another node and fix (percolate up) how about insert all items first and try to fix?

Given array itself is the heap that we need to fix.

```
void buildHeap(int[] input) { //input itself is a heap that we need to fix
    for(i = size/2 -1; i>=0 ; i--) { //starting from the last node with children
        percolateDown(i);
    }
}
```

Example



11. Time complexity of Floyd's algorithm.
Tighter analysis. Let n be the number of nodes in thie heap.

- How many elements in the second to the lowest level?_____. At most how many times do we percolate down? _____
- How many elements in the third to the lowest level? _____. At most how many times do we percolate down? _____.

Putting it altogether, write down the equation.

What is the time complexity of Floyd's algorithm?