

# University of Waterloo

## CS240, Fall 2026

### Module 2 Math Review and Intuition

Xuezhi (Jerry) Wang, CS240 ISA

#### 1 Don't Memorize

Some of you might be tempted to memorize every algorithm in this course. **Don't**. Instead, focus on understanding **why** the data structures and algorithms are designed this way.

It's honestly harder to memorize an algorithm than it is to understand it, and be able to re-derive it.

The heap's **structure property** is there to help maintain  $\lfloor \log n \rfloor$  height, so every level has to be filled. Conveniently, it can be filled left-to-right and stored in an array.

The heap's **order property** makes it so that the maximal key is at the root, which makes updating easier.

All algorithms on a heap are designed to maintain these properties/invariants of the data structure.

- **insert** inserts to the array, which is at the last level of the heap. Then it **fix-up** up the tree.
- **delete-max** deletes the root, then it **fix-down** down the tree.

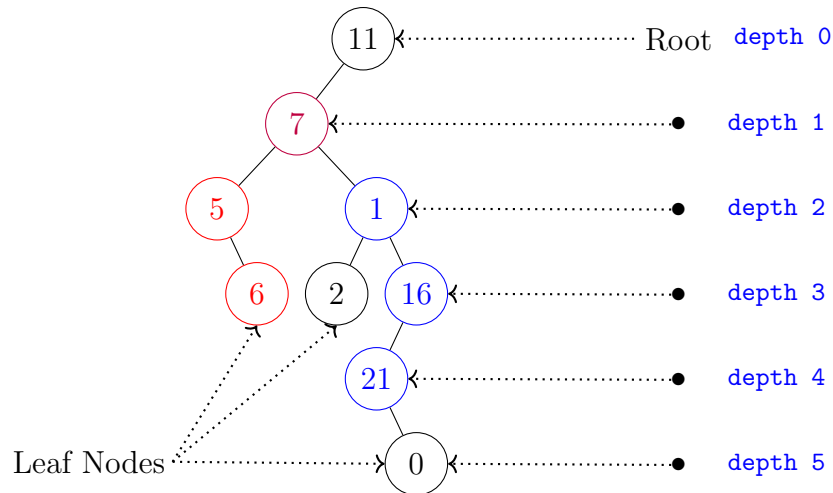
If you are unfamiliar with trees, luckily for you...

#### 2 Some useful facts about trees

Has anyone wondered why we draw trees upside down? Root at the top, leaves at the bottom, instead of real life trees with root at the bottom and leaves at the top?

I am not sure why, but it is really convenient to typeset this way.

Here is an example **binary** tree, of **height 5**. Binary means that each node has at most 2 children. Note that the root starts from height 0. The depth of a node is the number of edges from the root.



You might also notice that each child has exactly 1 parent, except for the root. If there are  $n$  nodes, then there are  $n - 1$  edges.

Also, every child has a unique simple path to the root, by recursively taking parents. Based on this, if you have any two nodes  $u, v$ , you can form a  $uv$ -walk by using  $u \rightarrow \text{root} \rightarrow v$ . By MATH2[34]9, there is therefore a **path** between two nodes.

If there are two distinct paths, you can form a cycle. A tree has **no cycles**, so this is impossible. So all pairs of nodes have a **unique path** between them! This path goes up the tree, to the **lowest common ancestor** of the two nodes, and then back down. For example, the lowest common ancestor of 6 and 21 is 7.

If you have an algorithm that starts from the root and recurses into each child or subtree, every node is visited **once**, because the path is unique. Hint: You're probably going to use this quite often!

Another important thing to remember is for a  $n$ -ary tree, layer  $k$  can contain  $n^k$  nodes. So even when  $n = 2$ , the last filled layer contains more nodes than every previous layer combined! Think about what this means for a heap, and what the average depth of a node is.