Trees

CMPT 145

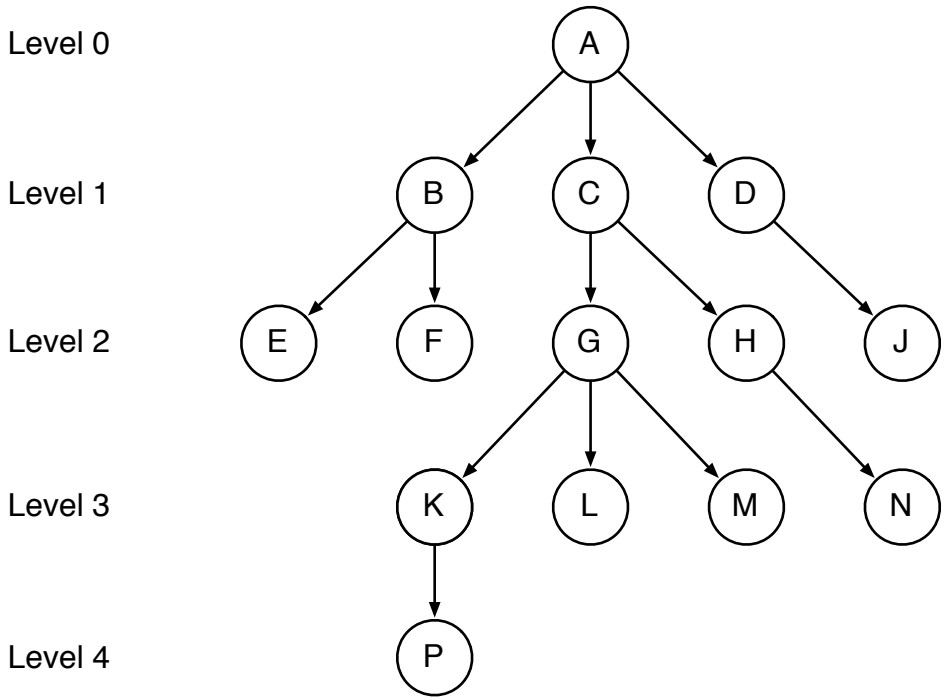
Trees organize data hierarchically

- Hierarchical organizations are common in companies, governments, etc.
- Textbooks are hierarchical: chapters, sections, paragraphs, etc.
- Computer file-systems are hierarchical: folders, sub-folders, documents.
- Computer application menus are hierarchical: menus, sub-menus.
- Hierarchies allow organization of activities!

Informal Terminology

- Data is stored in nodes.
- Nodes are connected with branches (a.k.a., edges, arcs, arrows)
- A branch connects a parent node to a child node.
- A parent may have 1 or more children; a child has exactly one parent.
- One node in a tree has no parents: the root
- Any node in the tree with no children is called a leaf node.

Example



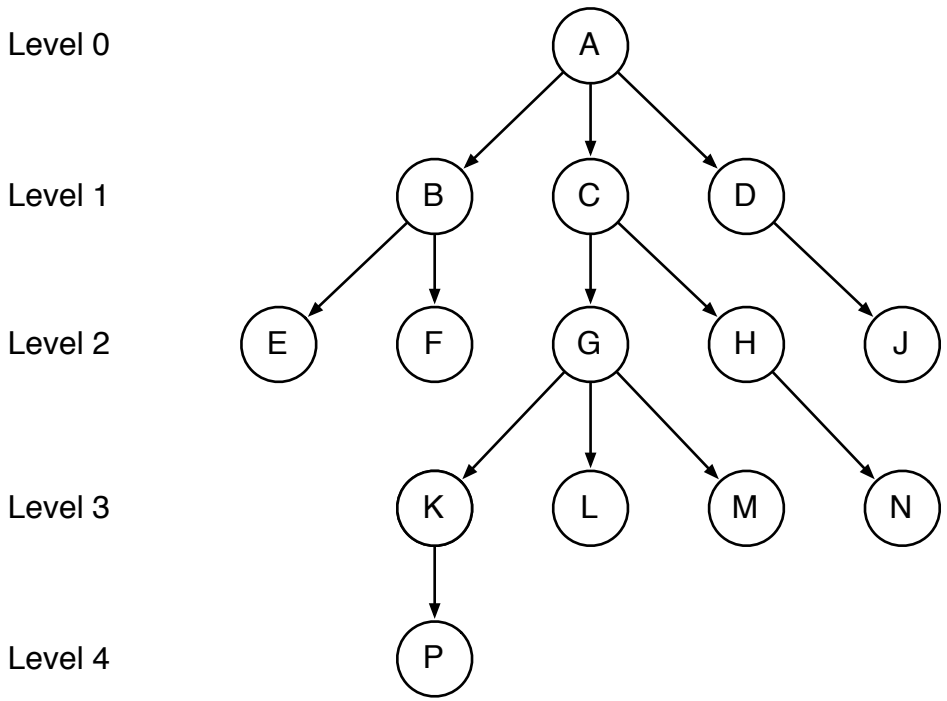
Root? Leaf? Parent? Child?

Formal Definition

A tree can be defined as follows:

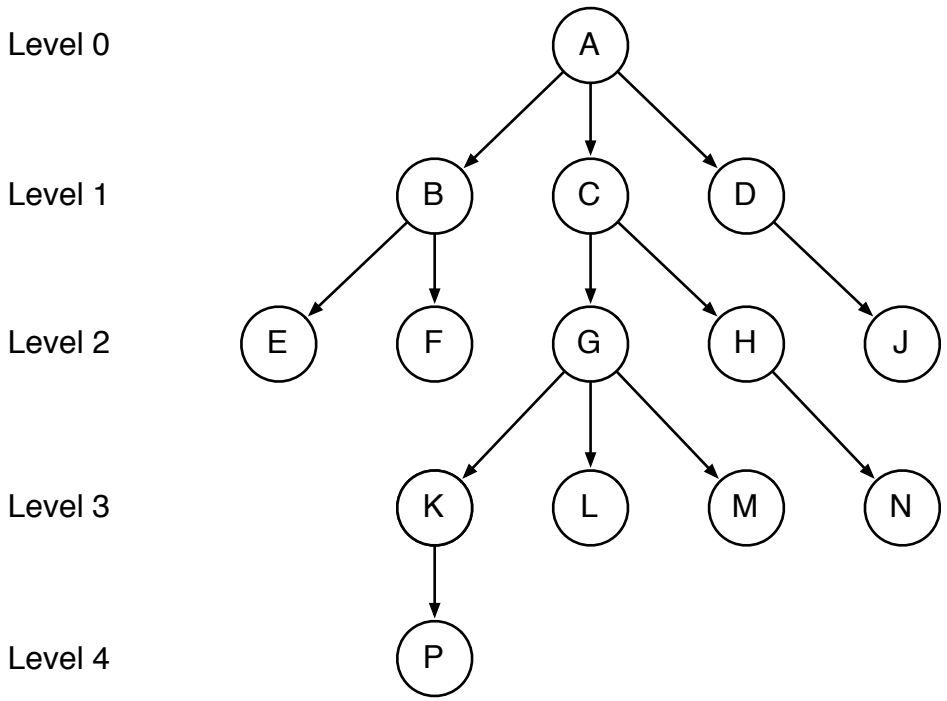
1. A structure with exactly zero nodes is an empty tree.
2. A structure with exactly one node is a tree.
3. If $t_1, \ldots, t_k$ are non-empty trees, then the structure, s, whose children are the roots of $t_1, \ldots, t_k$ is also a tree.

Definition: Path



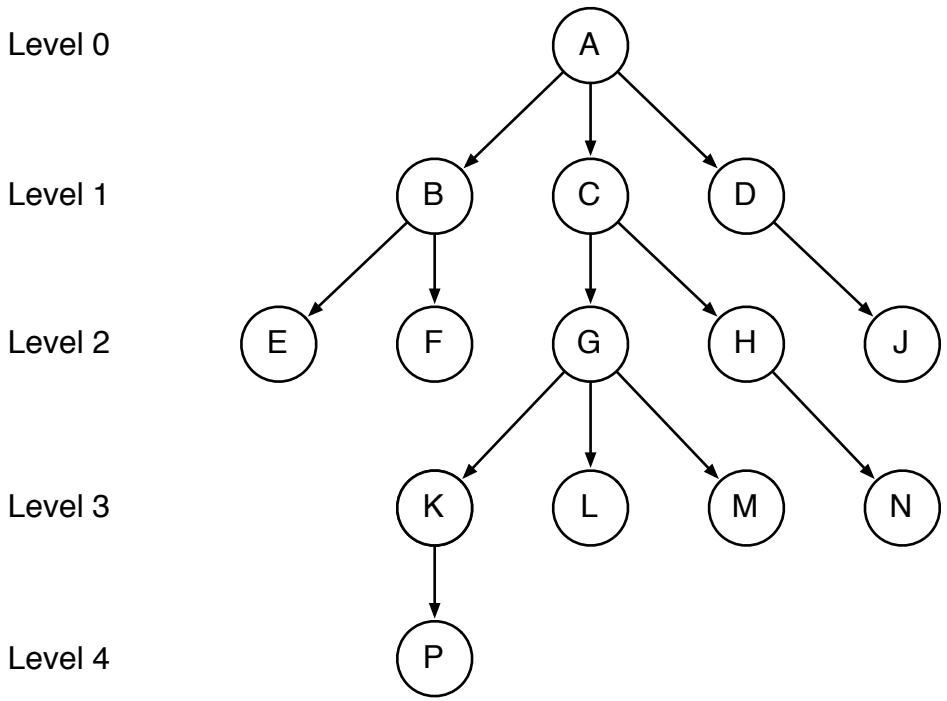
A path is a sequence of nodes connected by branches, with no repeated branches allowed.

Definition: Path length



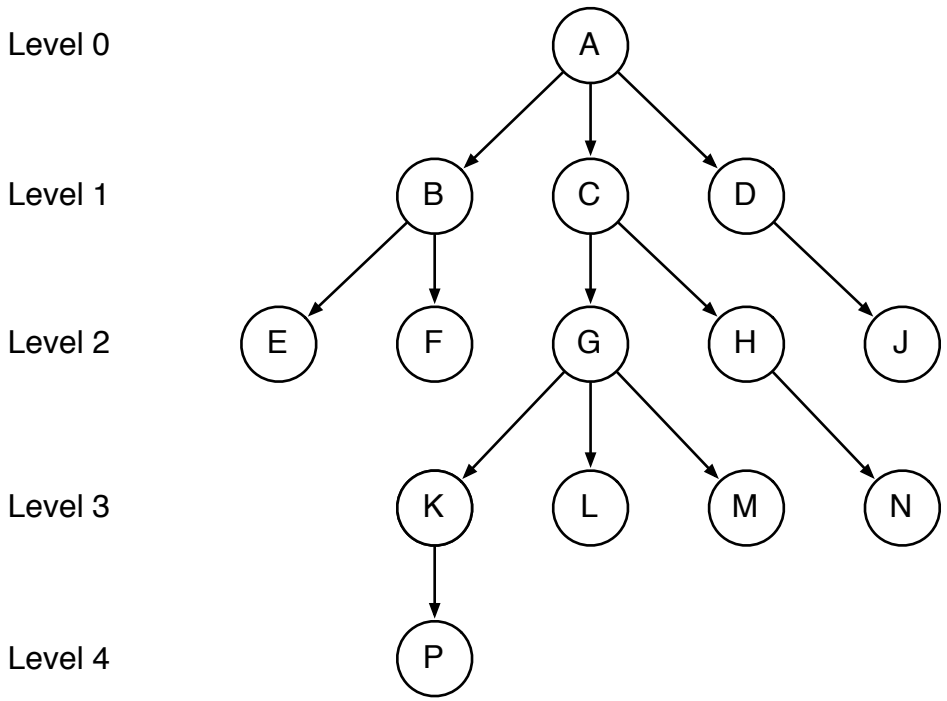
The length of the path is the number of branches in the path.

Definition: Level



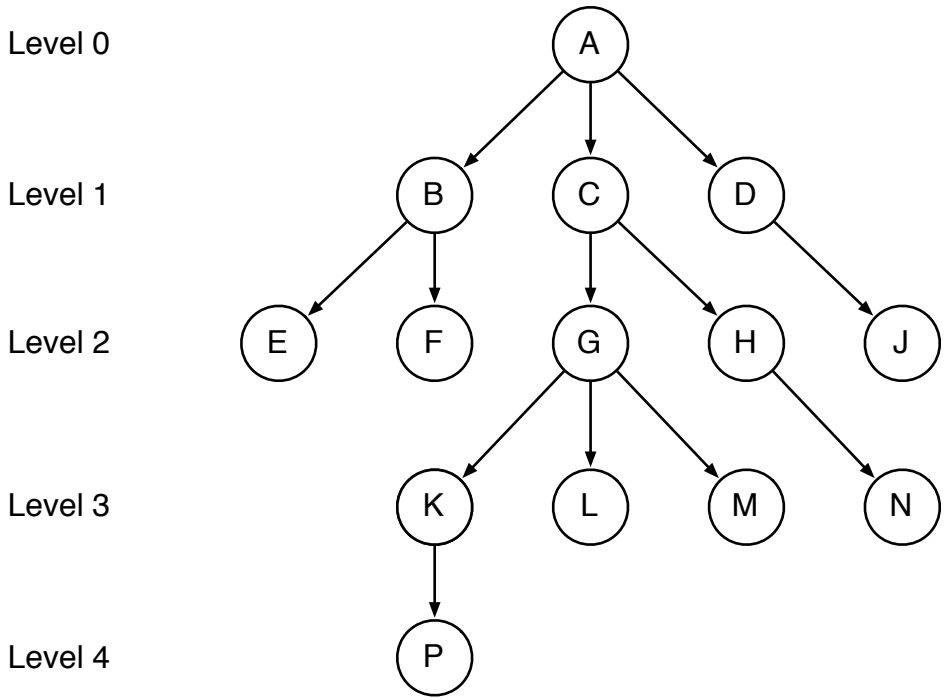
The term level describes where a node is in terms of the length of the path from the root to the node.

Definition: Height



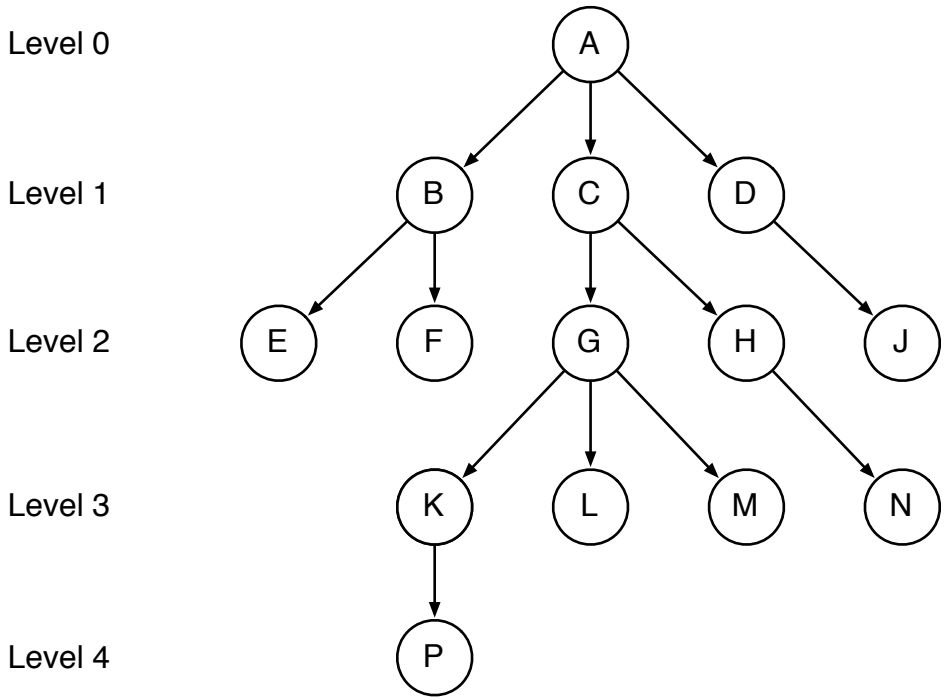
The height of a tree is always one more than the maximum level of any node in the tree.

Definition: Ancestors



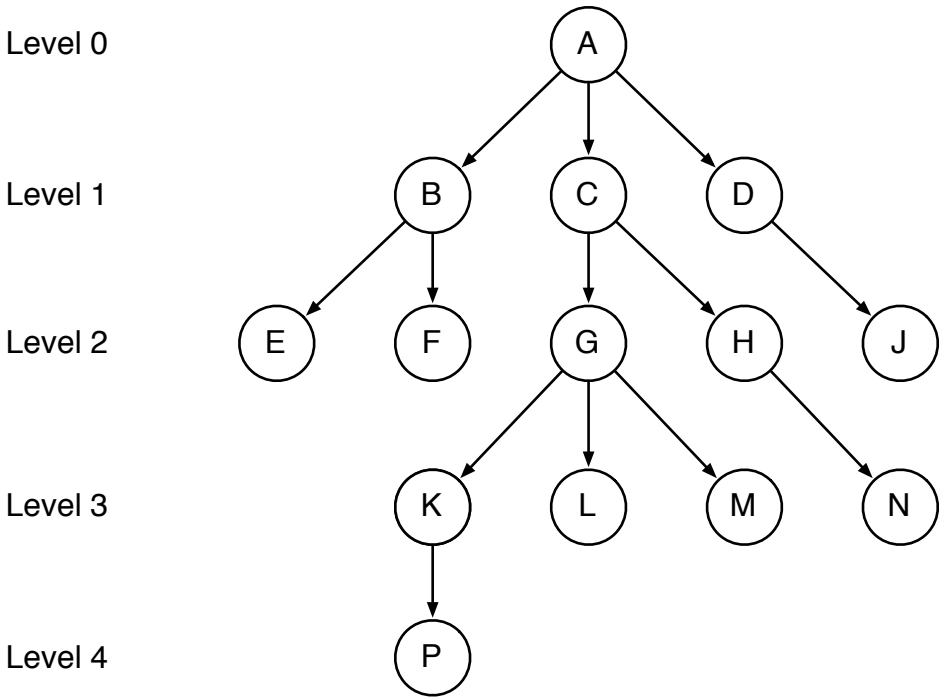
The ancestors of a node are nodes on the path from the node to the root.

Definition: Descendants



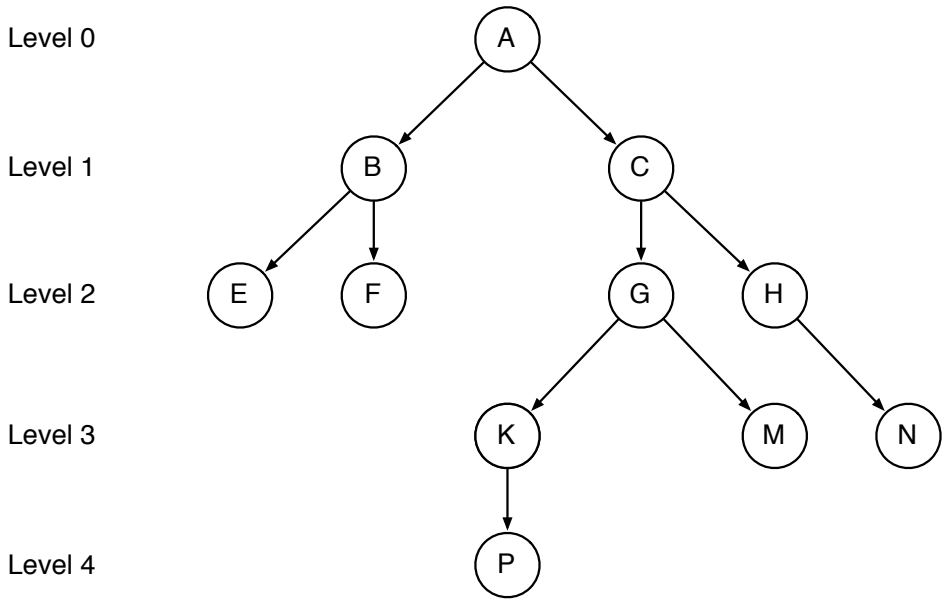
A node u is the descendant of a node v if v is an ancestor of u.

Definition: Sub-trees



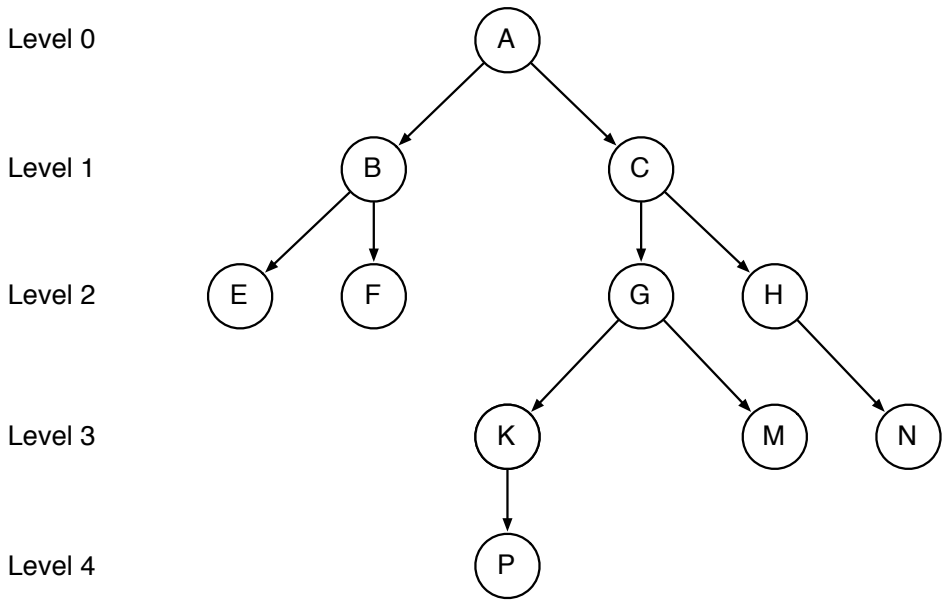
If u is a node in a tree, the sub-tree rooted at u consists of u and all of its descendants.

Definition: Binary trees



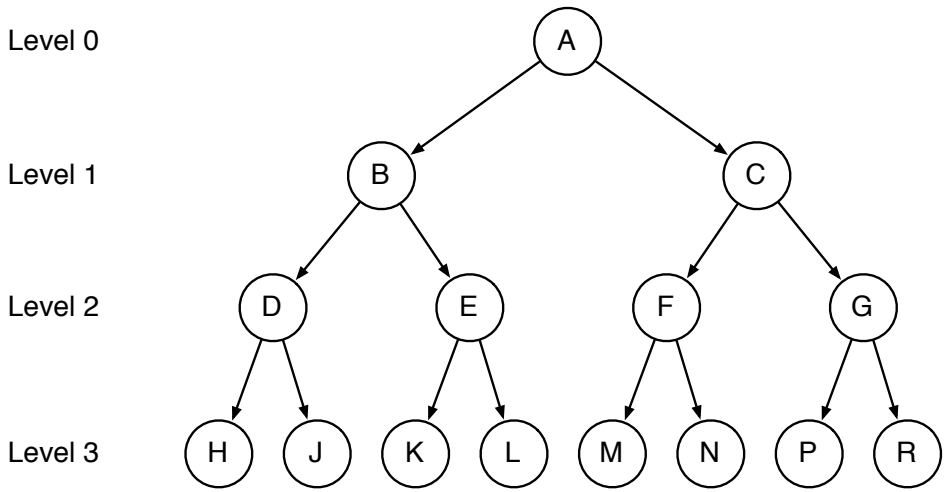
A binary tree is a special kind of tree with no node having more than 2 children.

Definition: Left and right sub-trees



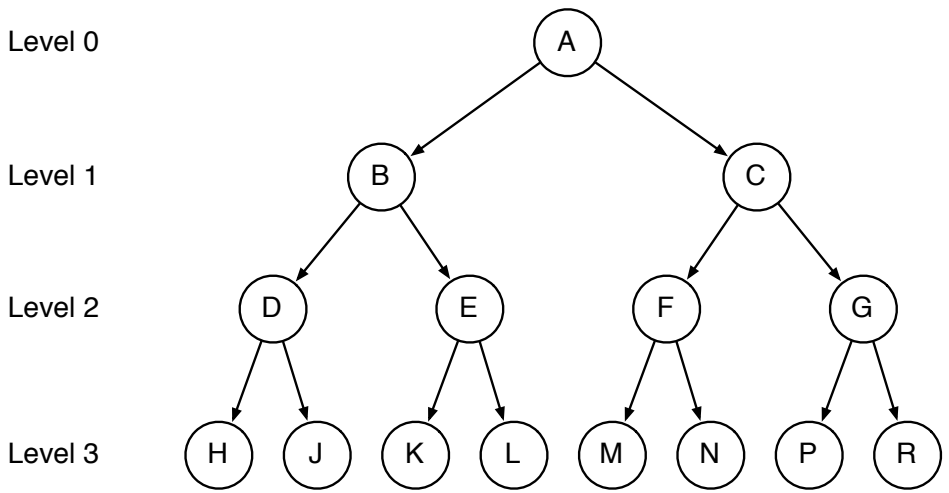
Since there are at most 2 children, we can label them left and right. Sometimes we say left sub-tree, or left node.

Definition: Complete Binary trees



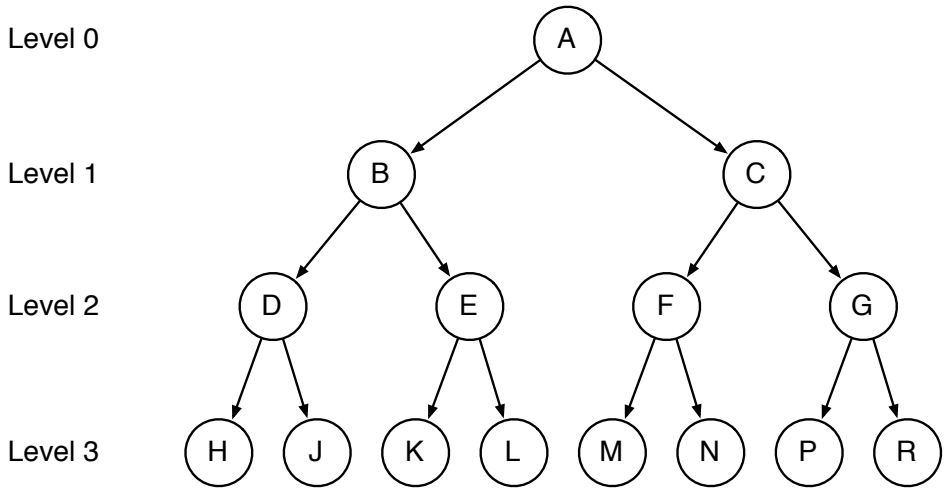
A complete binary tree is a binary tree that has exactly two children for every node, except for nodes at the maximum level, where the nodes are barren.

Property: Complete Binary tree levels



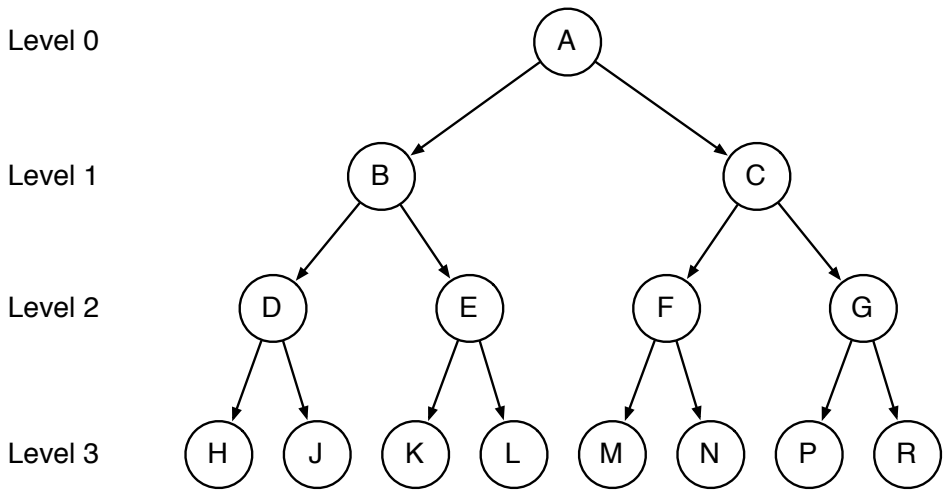
A complete binary tree has 2^l nodes at level l.

Property: Complete Binary tree height



In total, a complete binary tree whose height is h contains $2^h - 1$ nodes.

Property: Complete Binary tree height



A complete binary tree with n nodes has height $\log_2(n+1)$

Treenode Data Structure

A `treenode` is a simple record, implemented in Python as a dictionary with three fields:

data A data value
left A `treenode` (or the value `None`)
right A `treenode` (or the value `None`)

Treenode ADT

The primary operations provided for the `treenode` ADTs are as follows:

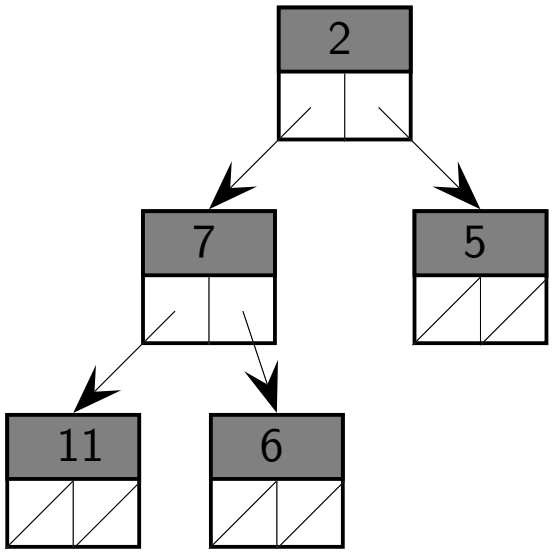
- `create(data, left=None, right=None)` creates a `treenode` to contain the data value and the given left and right values. If left and right are not given, the vlaue `None` is used by default.
- `get_data(treenode)` returns the contents of the data field of the given `treenode`
- `get_left(treenode)` returns the contents of the left field of the given `treenode`
- `get_right(treenode)` returns the contents of the right field of the given `treenode`

Treenode ADT

The primary operations provided for the `treenode` ADTs are as follows:

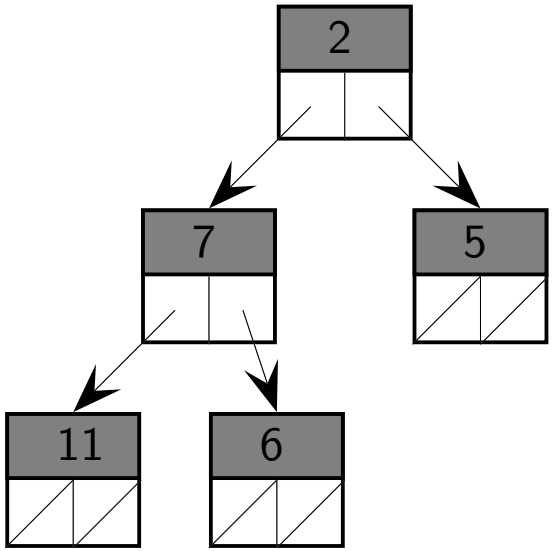
- `set_data(treenode, v)` given a `treenode`, sets the contents of the data field to `v`
- `set_left(treenode, n)` given a `treenode`, sets the contents of the left field to `n`
- `set_right(treenode, n)` given a `treenode`, sets the contents of the right field to `n`

Exercise



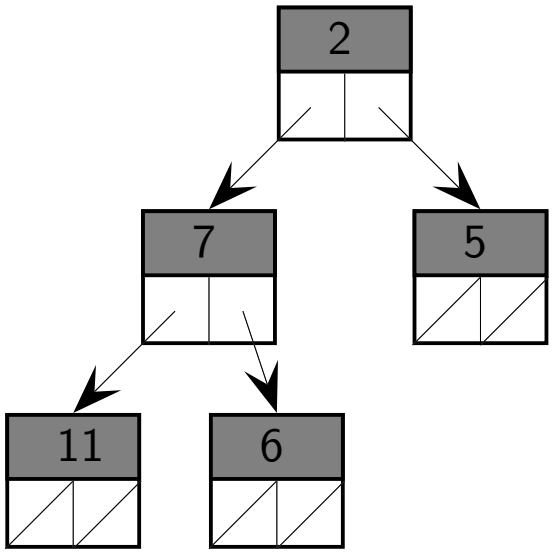
Use the treenode ADT to construct the given binary tree.

Exercise



Add a new left child to node 5.

Exercise



Add a new right child to node 6.