CMPT 225: Data Structures & Programming – Unit 12 – Trees

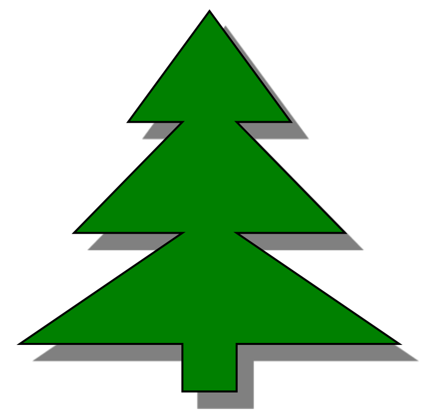
Dr. Jack Thomas

Simon Fraser University

Spring 2021

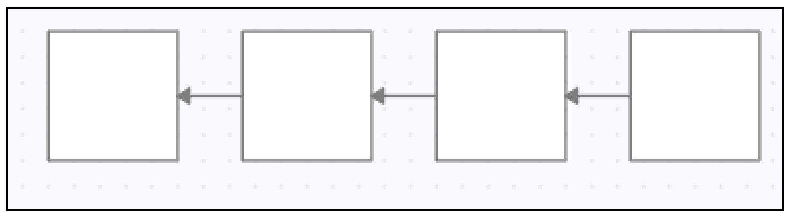
Today's Topics

- Introducing the Tree
- Non-Linear Data Storage
- Defining a Tree
- Tree Terminology
- Roots
- *Not* a part of this unit: the Tree ADT, Trees in Java, or anything on implementation. Stay tuned!

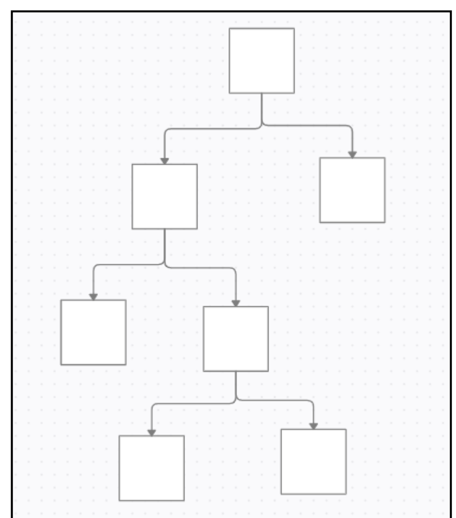


Time To Get Non-Linear

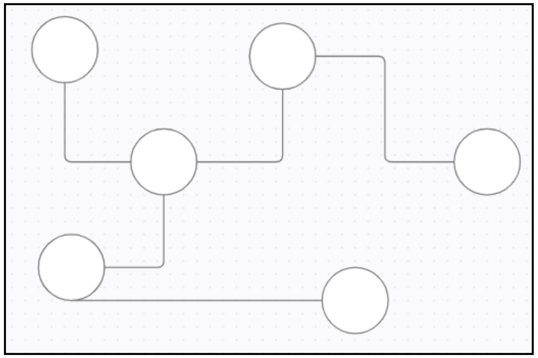
- The data structures we’ve dealt with so far have organized their data linearly – as in, like a line.

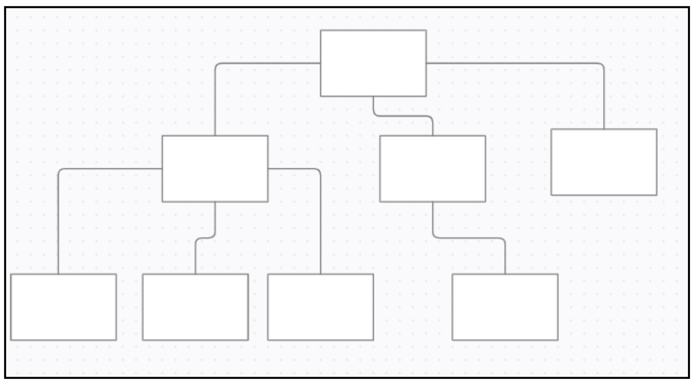


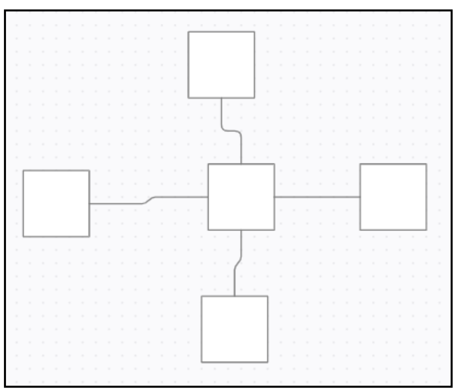
- With trees, we start to visualize how our data is stored in new ways.



Some Trees

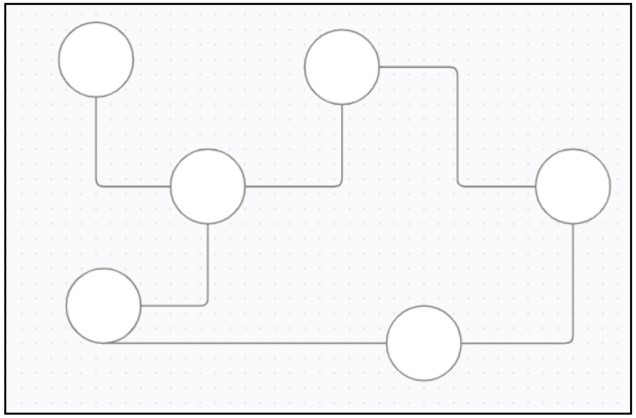


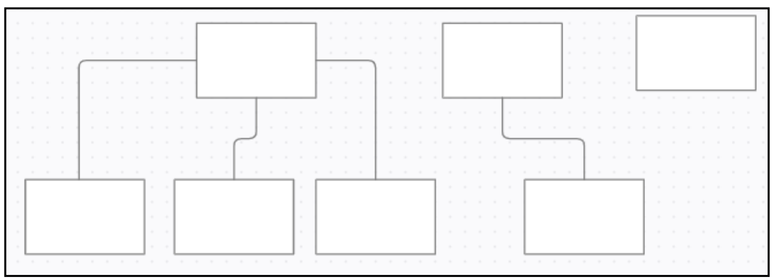


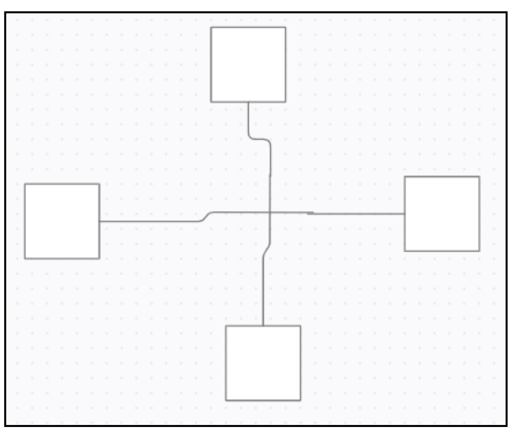


- Each node of a tree is called a **vertex**.
- Each vertex of a tree is connected to at least one other vertex via an **edge**.
- Similar in construction to a **list of nodes**, only not constrained to a strictly linear set of next/previous relationships.

Not Trees







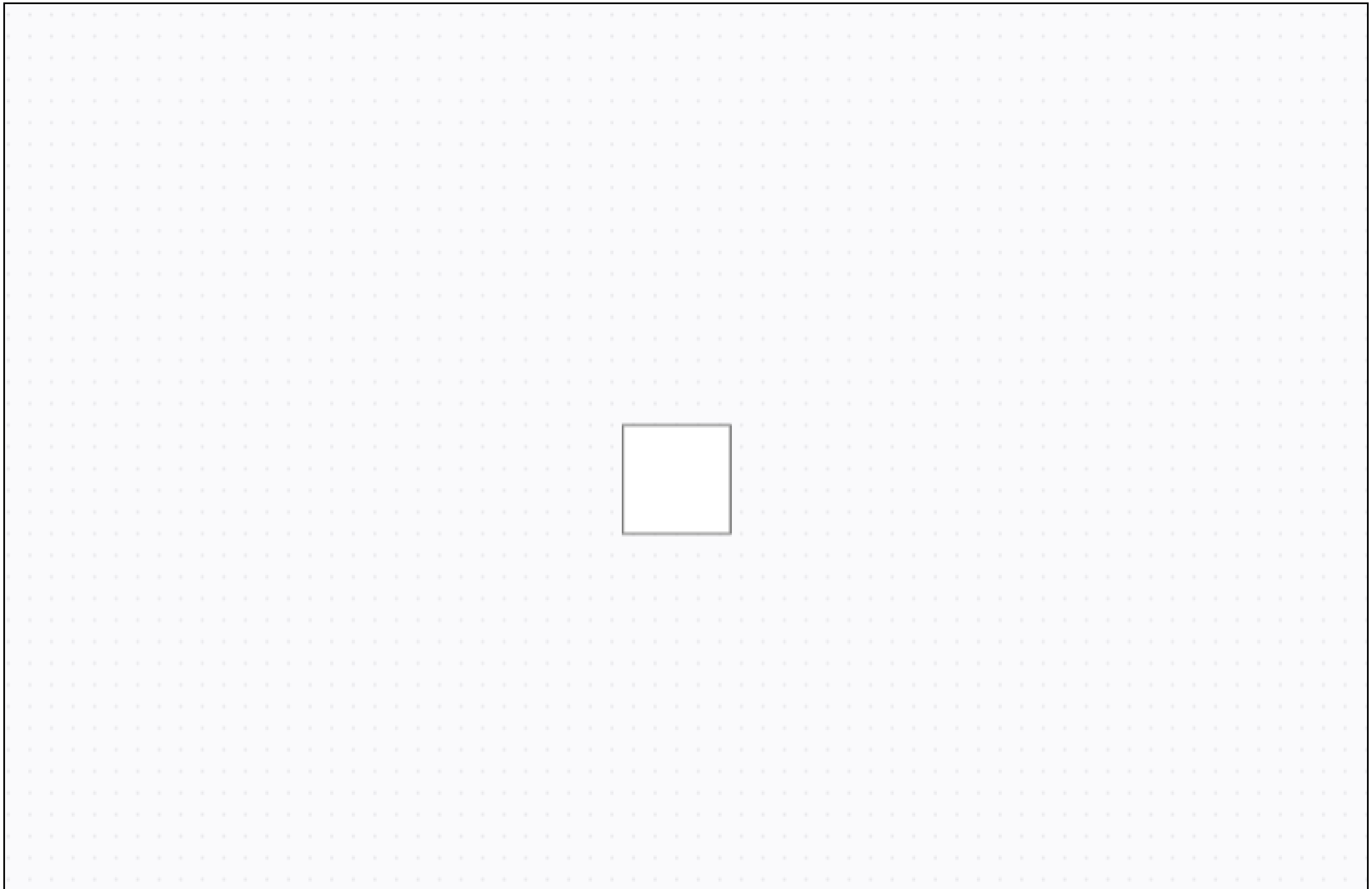
- The first example has a loop – **trees can't have cycles**.
- The second is three **disconnected trees** – a forest!
- The third has a four way connection – **tree connections are all one-to-one.**

Defining a Tree

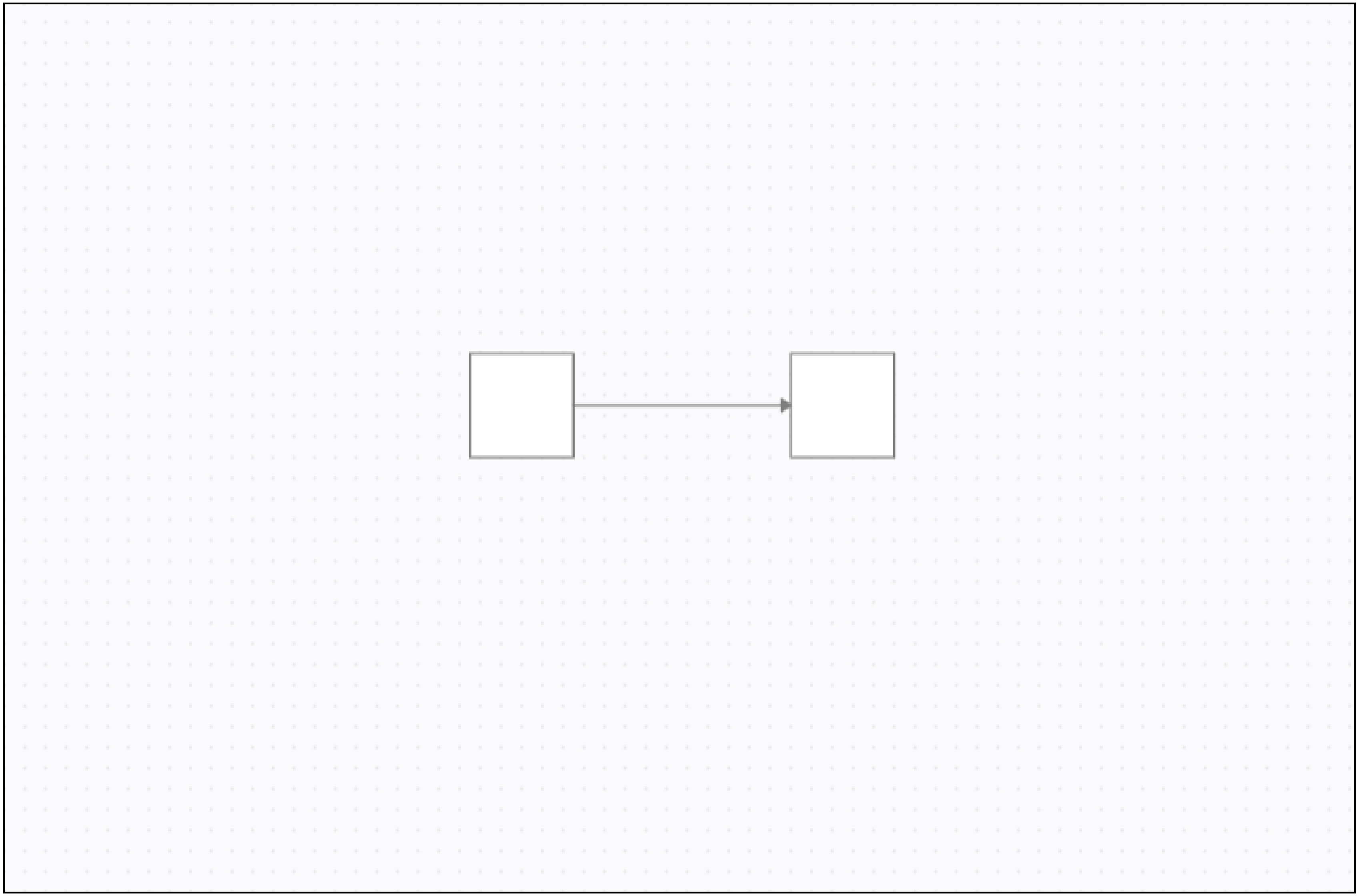
- An empty set of vertices (i.e. a blank screen) is a tree, the **empty tree**.
- A **single vertex with no edges** is also a tree (albeit a lonely one!).
- If you have a tree (T), select one vertex (u of T), then add a new vertex (u) to T, and connect them with an edge (uv), then that's a tree too.
- We're essentially using inductive reasoning here to define a tree up from its first vertex, like growing it from a seed one branch at a time!

Tree 0

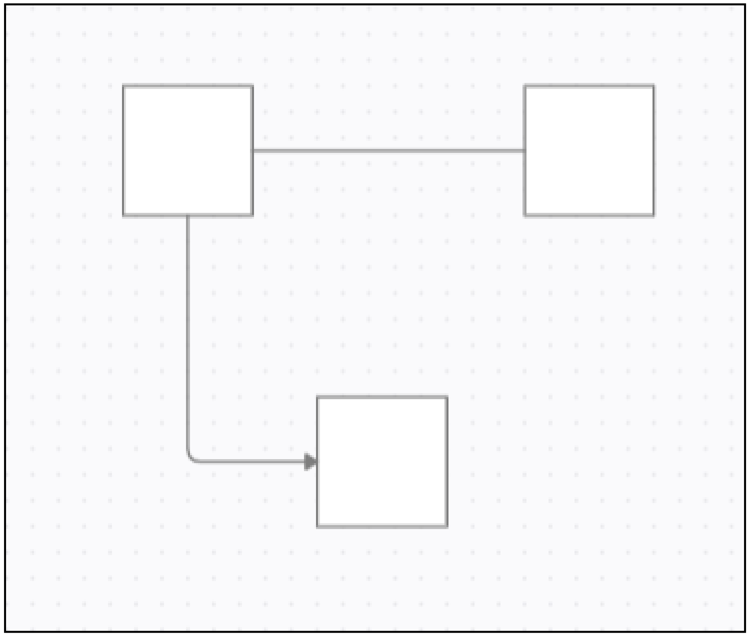
Tree 1



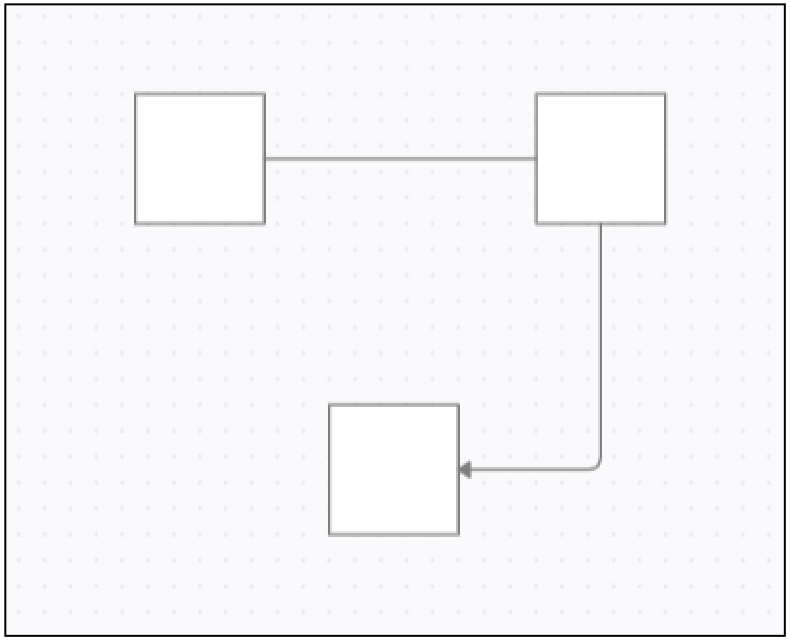
Tree 2



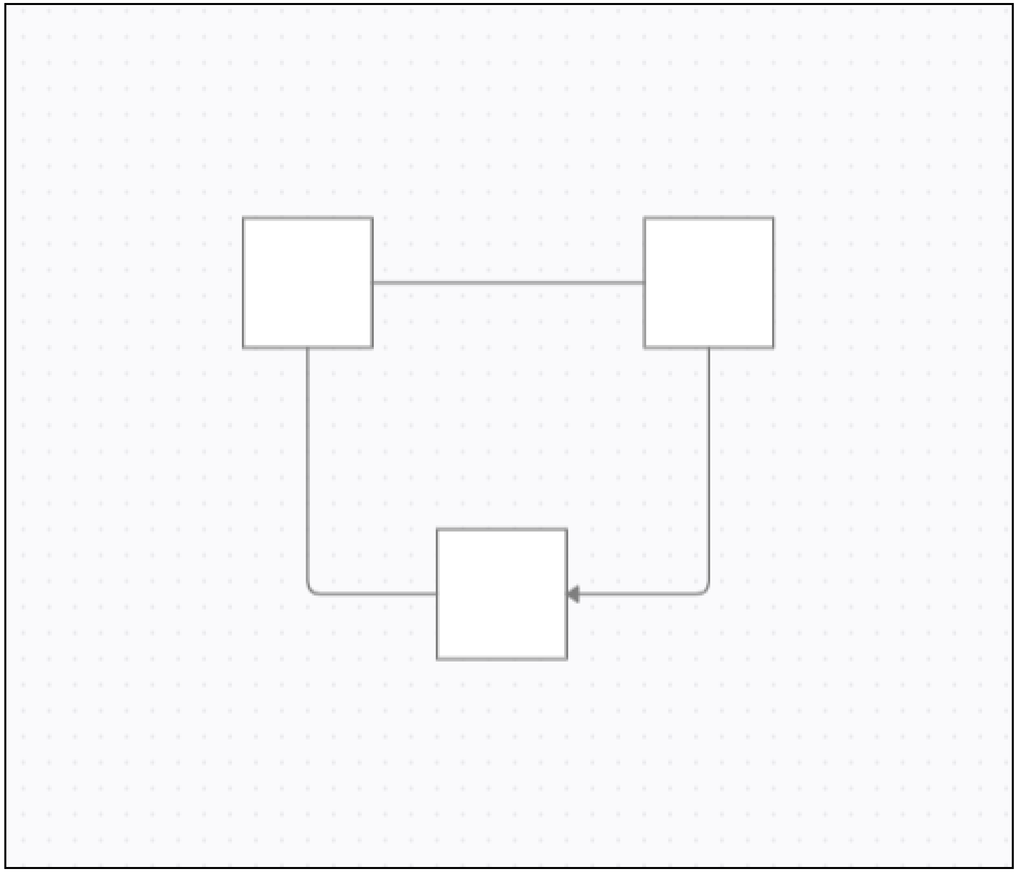
Tree(s) 3



(Or)



Not A Tree!

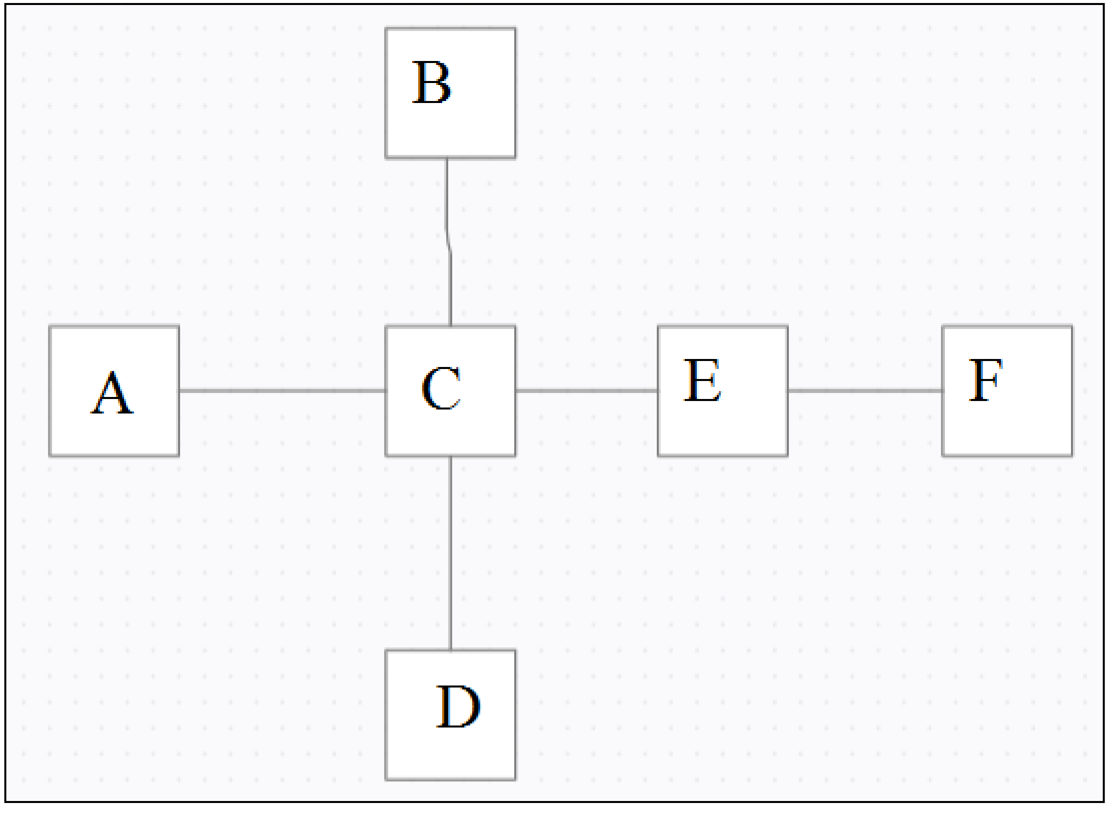


- An interesting property of this definition: there will **only be one path** from any node in a tree to any other node.

Tree Terminology

- If two vertices are connected by an edge, we say that they're **adjacent**, or **neighbours**.
- The more edges that connect to a vertex, the higher **degree** it has. A vertex connected to only one other vertex has degree one, one connected to four neighbours has degree four.
- Vertices with only one edge (so degree one) are called **leaves**.
- Vertices with more than one edge (degrees higher than one) are called **internal nodes**.
- The number of edges in the path between two nodes is their **distance**. Two neighbours are distance one, whereas a neighbours' neighbour will be distance two.

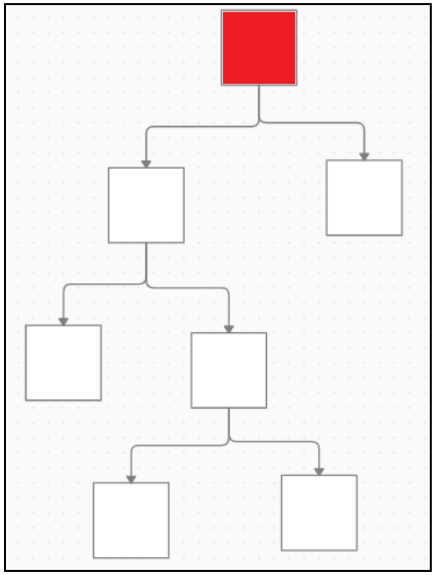
Label This Tree



- Give each vertex's neighbours, their degree, their distance to the other vertices, and whether they're leaves or internal nodes.

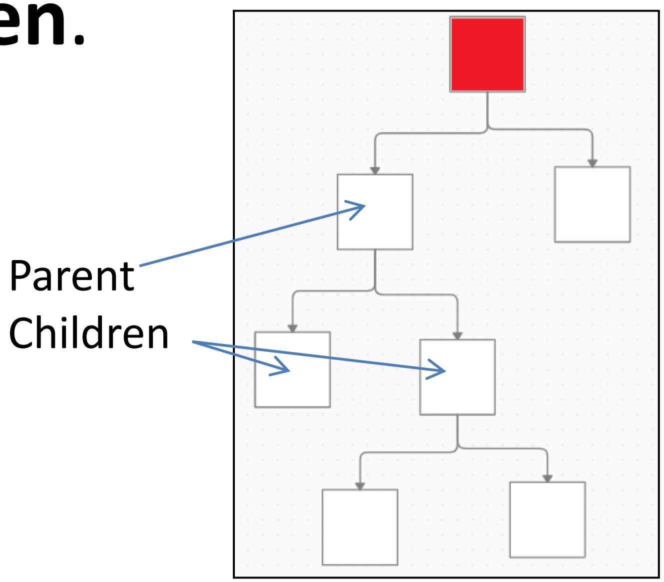
Trees and Roots

- Trees are often (though not always) actually rooted trees, a type of tree where a special vertex is called the root. This root vertex is often drawn at the top of a diagram, with adjacent vertices flowing down from it.



Rooted Tree Terminology

- Think of a rooted tree like a family tree – a node that is "above" (i.e. a neighbour a shorter distance to the root) another node is called that node's **parent**. The neighbours that are "below" (i.e. a further distance from the root) are that node's **children**.



Rooted Tree Terminology

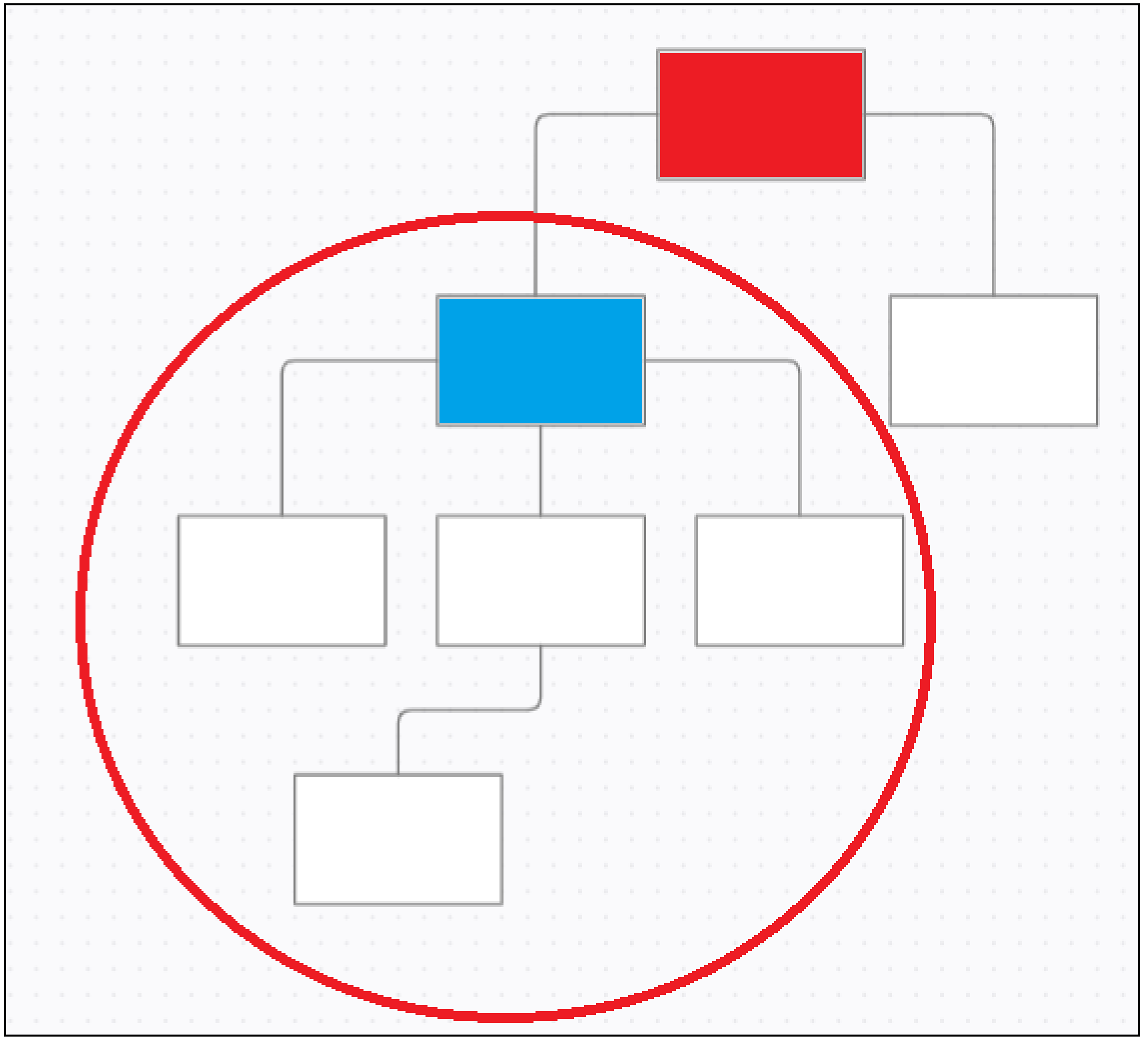
- A **grandparent**, naturally, is a parent's parent, while a **grandchild** is a child's child. **Siblings** have the same parent.
- Hopefully you didn't need me to tell you that.
- **Ancestors** of a node are all the nodes on the path from that node back to the root, while that node's **descendants** are all nodes whose path back to the root will pass through them.

Rooted Tree Terminology

- The **subtree** of a node is that node and all of its descendants (both the vertices and the edges, to be clear). Essentially a new tree with the chosen node as the root.
- The **depth** of a node is how many edges it takes to connect them to the root – so zero for the root, one for the root’s children, two for the root’s grandchildren, etc.
- The **height** of the tree is the maximum depth of any node across the whole tree.

Diagramming A Rooted Tree

- If the **red** node is the **root**, the **blue** node is the **root of the subtree** in the red circle.
- The other nodes in the circle are the blue nodes' **descendants**, while the root is its **parent** and **ancestor**.
- The **depth** of the red root is 0, the **depth** of the blue node is 1 (or 0 within its own subtree), the **depth** of the blue node's children is 2 (or 1 within blue's subtree)
- The **height** of the tree is 3, while the **height** of the subtree is 2.



Recap – Seeing The Forest

- **Trees** are a kind of **non-linear data structure** for storing a set of nodes.
- The definition of a tree **ensures all nodes are connected by edges** and there is **only one path connecting any two nodes**.
- **Rooted trees** are a common variant where one vertex (node) of the tree is named the root, with all other vertices "descending" from it.
- There's a boatload of **tree-related terminology** to learn, including parents, children, depth, height, neighbours, subtrees, descendants, ancestors, degrees, leaves, internal nodes, siblings, grandchildren, and grandparents.