Trees

MAT230

Discrete Mathematics

Fall 2019

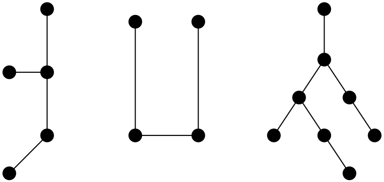
Outline

Definitions

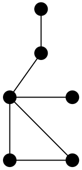
Definition

A **tree** is a connected undirected graph that has no cycles or self-loops.

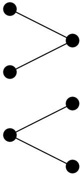
Some examples:



Trees



Not a tree: has a cycle



Not a tree: disconnected

Definition

A **forest** is an undirected graph whose components are all trees.

A Theorem About Trees

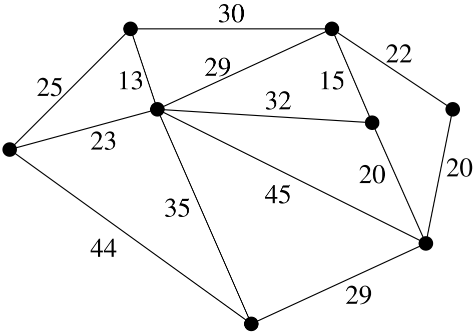
Theorem

Let $T = (V, E)$ be an undirected graph with no self-loops and $|V| = n$. Then the following statements are equivalent:

1. *T is a tree.*
2. *Any two vertices of T are connected by exactly one path.*
3. *T is connected and every edge is an* **isthmus** *(its removal disconnects T).*
4. *T contains no cycles, but the addition of any new edge creates exactly one cycle.*
5. *T is connected and has $n - 1$ edges.*

Spanning Trees and Minimum Spanning Trees

Suppose the following graph represents distance in miles between towns. The towns are to be connected by high-speed network cable. Assuming the cost of cables is proportional to their length and bandwidth is not a limiting factor, the most cost effective network will be a tree that *spans* the graph.



Definitions

Definition

Let $G = (V, E)$ be a connected undirected graph. A **spanning set** for G is a subset E' of E such that (V, E') is connected.

Definition

Let G be a connected undirected graph. The subgraph T is a **spanning tree for** G if T is a tree and every node in G is a node in T.

Definition

If G is a weighted graph, then T is a **minimal spanning tree of** G if it is a spanning tree and no other spanning tree of G has smaller total weight.

Minimal Spanning Trees (MST)

Suppose $G = (V, E, w)$ is a weighted connected undirected graph. The **minimal spanning tree problem** is to find a spanning tree $T = (V, E')$ for G such that $\sum_{e \in E'} w(e)$ is as small as possible.

Unlike the Traveling Salesman Problem, solving the MST problem is relatively easy. We consider two algorithms.

Prim's Algorithm

Let $G = (V, E, w)$ be a weighted connected undirected graph.

1. Pick any vertex $v \in V$.
2. $V' = \{v\}$
3. $V_0 = V - \{v\}$
4. $E' = \{\}$
5. While $V' \neq V$ do:
   a. Find $u \in V'$ and $v \in V_0$ such that edge $e = \{u, v\}$ has minimum weight
   b. $E' = E' \cup e$
   c. $V' = V' \cup \{v\}$
   d. $V_0 = V_0 - \{v\}$

Upon termination, $T = (V, E')$ is a minimum spanning tree for G.

Kruskal's Algorithm

Let $G = (V, E, w)$ be a weighted connected undirected graph.

1. Find edge $e \in E$ with minimum weight
2. $E' = \{e\}$
3. $E_0 = E - \{e\}$
4. $V' = \{v : v \text{ is a vertex for which } e \text{ is an incident edge}\}$
5. While $V' \neq V$ or (V', E') not connected do:
   a. Find edge $e \in E_0$ with minimum weight that will not complete a cycle in (V', E').
   b. $E_0 = E_0 - e$
   c. $E' = E' \cup e$
   d. $V' = V' \cup \{v : v \text{ is a vertex for which } e \text{ is an incident edge}\}$

Upon termination, $T = (V, E')$ is a minimum spanning tree for G.

Rooted Trees

Definition

Every nonempty tree can have a particular vertex called a **root**. In a rooted tree, the root is at **level 0**. The **level** of all other vertices is one greater than the number of edges in the walk from the root to the vertex. The **height** of a tree is the number of levels in the tree.

Definition

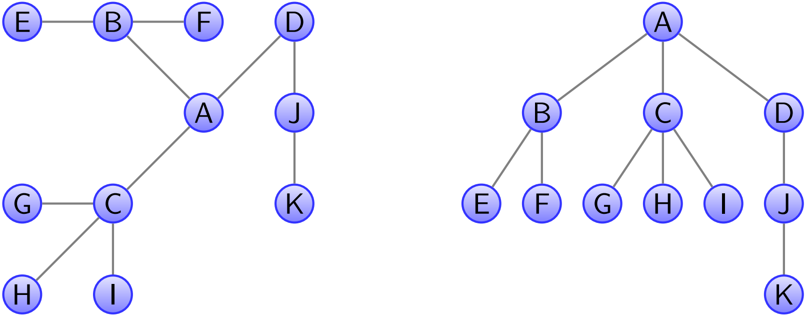
A vertex u in a rooted tree is a **parent** of a vertex v if v is adjacent to u and the level of v is one greater than the level of u. In this case v is a **child** of u. Two or more vertices are **siblings** if they have the same parent.

Definition

Nonroot vertices of degree 1 in a tree are called the **leaves** of the tree. All other vertices are called **internal vertices**.

Rooted Trees

On the left is a typical tree. On the right is the same tree redrawn with vertex A identified as the root.

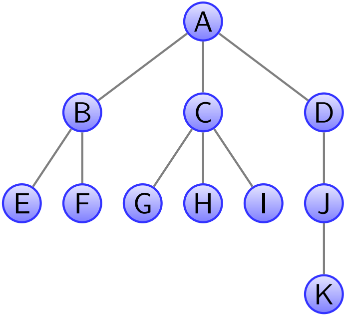


Vertex C is the parent of vertices G, H, and I. Vertices E and F are children of vertex B and so are siblings. Vertices E, F, G, H, I, and J are all at level 2. Vertices E, F, G, H, I, and K are leaves while A, B, C, D, and J are internal vertices.

Definition

A **subtree** is connected component of a rooted tree T that does not include the root of T. Every subtree is itself a rooted tree.

- This tree has subtrees rooted at B, C, and D.
- There is also a subtree rooted at J.
- Finally, each leaf is a subtree.

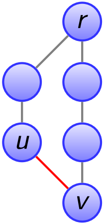


Kruskal's Algorithm (Version 2)

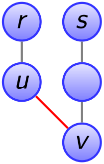
Working with rooted trees makes Kruskal's algorithm easier to describe.

Suppose there is an edge e between vertices u and v.

If u and v are both in a tree rooted at r then the edge e will create a cycle.



However, if u is in a tree rooted at r and v is in a tree rooted at s, adding edge e does not create a cycle, but merges the two trees.



In this case, we can choose either r or s to be the root of the merged tree.

Kruskal's Algorithm (Version 2)

Let $G = (V, E, w)$ be a weighted connected undirected graph and suppose we have function rootof(v) that returns the root of the tree containing vertex v.

1. Create sorted edge list E_L (sort by increasing weight)
2. Initialize a list R of rooted trees with each vertex in the tree as the root of its own tree.
3. $E' = \{\}$
4. While R contains more than one entry **and** E_L not empty do:
   a. remove edge $e = \{u, v\}$ from E_L
   b. if rootof(u) $\neq$ rootof(v) then
      - $E' = E' \cup \{e\}$
      - Merge tree rooted at rootof(u) into tree rooted at rootof(v) (or vice-versa). This reduces the number of entries in R by one.

Upon termination, $T = (V, E')$ is a minimum spanning tree for G.

Binary Trees

Definition

A **binary tree** T is a tree that has zero or more nodes in which each node has at most two **children**. Each nonleaf node has **left subtree** and a **right subtree**, either of which may be empty.

Question: How many vertices can be found on level k of a binary tree?

Binary Trees

Definition

A **binary tree** T is a tree that has zero or more nodes in which each node has at most two **children**. Each nonleaf node has **left subtree** and a **right subtree**, either of which may be empty.

Question: How many vertices can be found on level k of a binary tree?
Answer: $\boxed{2^k;}$ the number of possible vertices doubles on each level.

Binary Trees

Definition

A **binary tree** T is a tree that has zero or more nodes in which each node has at most two **children**. Each nonleaf node has **left subtree** and a **right subtree**, either of which may be empty.

Question: How many vertices can be found on level k of a binary tree?
Answer: $\boxed{2^k;}$ the number of possible vertices doubles on each level.

Question: How many vertices can be found in a binary tree of height h?

Binary Trees

Definition

A **binary tree** T is a tree that has zero or more nodes in which each node has at most two **children**. Each nonleaf node has **left subtree** and a **right subtree**, either of which may be empty.

Question: How many vertices can be found on level k of a binary tree?
Answer: $\boxed{2^k;}$ the number of possible vertices doubles on each level.

Question: How many vertices can be found in a binary tree of height h?
Answer: $\boxed{2^{k+1} - 1;}$ the sum of the powers of 2 from 2^0 up to 2^k.

Binary Tree Traversal

A **traversal** of a tree visit each vertex of the tree in some order determined by the connectivity within the tree. There are three common traversals of binary trees, each described by a recursive algorithm.

Preorder Traversal:

1. Visit the root of the tree
2. Preorder traverse the left subtree
3. Preorder traverse the right subtree

Inorder Traversal:

1. Inorder traverse the left subtree
2. Visit the root of the tree
3. Inorder traverse the right subtree

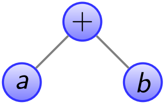
Postorder Traversal:

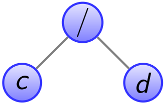
1. Postorder traverse the left subtree
2. Postorder traverse the right subtree
3. Visit the root of the tree

Expression Trees

The arithmetic operations $+$, $-$, $\times$, and $/$ are called *binary operators* since they each take two operands.

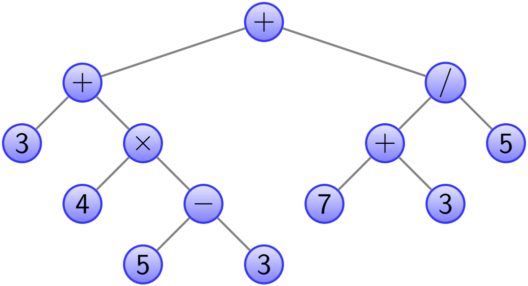
We can diagram them with trees. For example, trees for $a + b$ and c/d are





Expression Trees

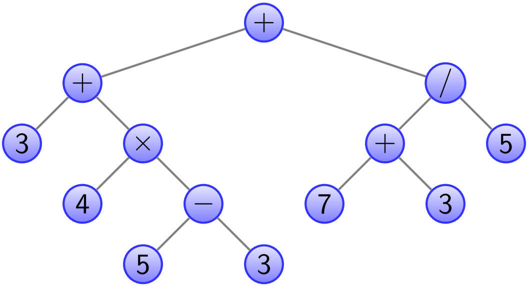
Consider the arithmetic expression $3 + 4(5 - 3) + (7 + 3)/5$. A tree for this expression is



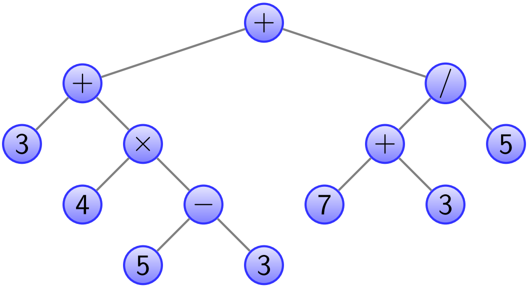
This tree is not unique, other binary trees could be used to represent the same expression.

Construct a preorder and postorder traversal of this tree.

Expression Trees

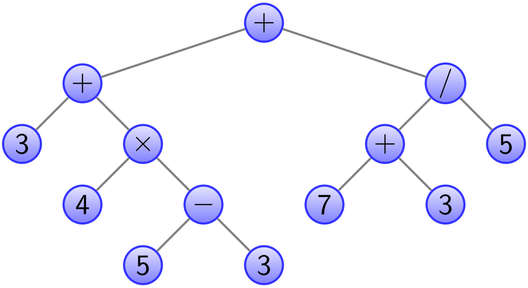


Expression Trees



Preorder traversal: $+ \quad + \quad 3 \quad \times \quad 4 \quad - \quad 5 \quad 3 \quad / \quad + \quad 7 \quad 3 \quad 5$

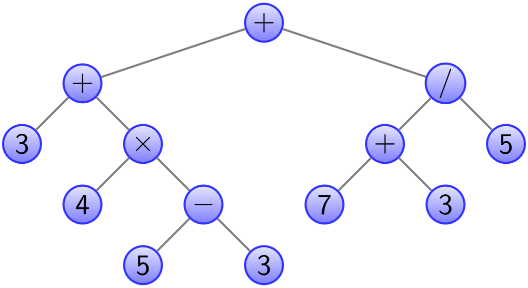
Expression Trees



Preorder traversal: $+\ +\ 3\ \times\ 4\ -\ 5\ 3\ /\ +\ 7\ 3\ 5$

Postorder traversal: $3\ 4\ 5\ 3\ -\ \times\ +\ 7\ 3\ +\ 5\ /\ +$

Expression Trees



Preorder traversal: $+ \quad + \quad 3 \quad \times \quad 4 \quad - \quad 5 \quad 3 \quad / \quad + \quad 7 \quad 3 \quad 5$

Postorder traversal: $3 \quad 4 \quad 5 \quad 3 \quad - \quad \times \quad + \quad 7 \quad 3 \quad + \quad 5 \quad / \quad +$

Both of these can be evaluated unambiguously without the need for parentheses.