

MATH1510 - Discrete Mathematics

Trees

University of Newcastle

UoN

Outline

- ① Tree characterizations
- ② Trees with additional structure
 - Rooted trees
 - Ordered trees
 - Binary trees and Huffman codes

Trees

Definition

A **tree** is a connected graph without cycles.

The following gives four equivalent characterizations.

Theorem (Characterizations of trees)

The following statements about a graph $G = (V, E)$ with n vertices are equivalent:

- ① G is a tree.
- ② In G , there is a unique simple path between any two vertices.
- ③ G is connected and has $n - 1$ edges.
- ④ G is acyclic (i.e., does not contain any cycle) and has $n - 1$ edges.

Tree criterion (edge number)

Theorem

A tree with n vertices has $n - 1$ edges.

Proof.

We proceed by induction on n .

Basis Step: $n = 1$ works.

Inductive Step: Suppose true for all trees with k vertices: i.e., they all have $k - 1$ edges. Let T be any tree having $k + 1$ vertices. Now T must have some vertex of degree 1 (otherwise all vertices would have degree at least 2, and by a previous theorem this would imply that there is a cycle). Call that vertex x .

Remove this vertex x and its edge from T . The resulting graph, T' , is a tree with k vertices, and therefore $k - 1$ edges.

Consequently T has $k + 1$ vertices and k edges. Therefore the statement is true for T .

By induction, the statement is true for all positive integers n . □

Some corollaries

Corollary

An acyclic graph with n vertices and m connected components has $n - m$ edges.

Corollary

If an acyclic graph has n vertices and $n - 1$ edges then it is a tree

Corollary

A connected graph with n vertices and at least one cycle has $\geq n$ edges

Corollary

If a connected graph has n vertices and $n - 1$ edges then it is a tree.

Tree characterizations

These corollaries imply that a graph on n vertices is a tree if and only if it satisfies one (and thus all) of the following four equivalent conditions:

- Connected, no cycles (the definition here)
- Unique simple paths between any two vertices (Johnsonbaugh's definition)
- Connected, $n - 1$ edges
- No cycles, $n - 1$ edges

A tree must have at least two leaves

Lemma

A tree with at least 2 vertices must have at least 2 vertices of degree 1.

Proof.

Let T be a tree with $n \geq 2$ vertices.

- Handshake theorem: $\sum_{i=1}^n \delta(v_i) = 2|E| = 2(n-1) = 2n-2$
- There is no vertex of degree 0, because G is connected.
- no vertex of degree 1 \implies all degrees at least 2 $\implies \sum_{i=1}^n \delta(v_i) \geq 2n \implies$ contradiction.
- only one vertex of degree 1 $\implies n-1$ vertices of degree at least 2 $\implies \sum_{i=1}^n \delta(v_i) \geq 1 + (n-1) \cdot 2 = 2n-1 \implies$ contradiction.

□

Counting Trees

Q: How many trees are there with 3 vertices?

Q: How many trees are there with 4 vertices?

Of course, here we really mean “up to isomorphism” in that we don’t count two graphs that are isomorphic as being distinct.

How many trees with 6 vertices?

A 4

B 5

C 6

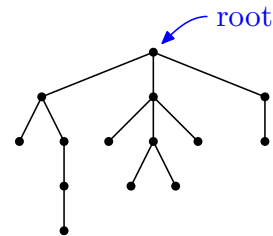
D 7

Rooted Trees

In many applications, there is one vertex which is distinguished from the others as being the “top level” vertex.

It is called the **Root** and a tree with such a vertex is called a **Rooted Tree**.

We usually draw the tree with the root at the top, and the rest of the tree downward.



The **Level** of a vertex is the length of a path from the root down that vertex. (The root is at level 0.)

The **Height** (aka **depth**) of a tree is the maximum level.

Terminology

Terms borrowed from *genealogy*: In an rooted tree T with vertex v .

parent The parent of v is the vertex adjacent to v whose level is one less than that of v .

children The children of v are the vertices adjacent to v whose level is one more than that of v .

sibling ...?

grandchild ...?

descendent ...?

Other terms

terminal A *terminal* vertex is one with no descendents (aka leaf)

internal An *internal* vertex is not a terminal vertex

Q: When does the parent exist, and why is it unique?

Q: Is every vertex of degree 1 a terminal vertex?

How many rooted trees with 3 vertices

A 1

B 2



C 3

D 4

Isomorphisms and Invariants

Q: How many rooted trees are there with n vertices?

This question illustrates an unexpected complication. When counting “different” graphs or trees, we used isomorphisms and invariants of the graphs to count two graphs as being the same or different.

However, with rooted trees, we want to count  and  as being different, even though they are isomorphic graphs.

Isomorphisms of rooted trees

- This situation is resolved by only allowing isomorphisms that *preserve the root of the tree*.
- In the one-to-one correspondence between the two vertex sets, the root of one tree is matched to the root of the other.
- Such isomorphisms are called **isomorphisms of rooted trees**, and we say that two rooted trees with such an isomorphism are **isomorphic as rooted trees**.
- With this, some new **invariants** come into play, e.g., two isomorphic rooted trees have the same number of level-1 vertices, the same number of level-2 vertices, etc.

Ordered trees



For some applications of rooted trees, we need to assign a left-to-right order to the children of each vertex. This produces an **Ordered Tree**.

Examples:

- Genealogy — ordered by birth
- File systems — files and subfolders ordered alphabetically by name

Ordering of all vertices

The order on the children can be **inherited by descendants**. By tracing up to a common ancestor of two vertices u and v , it is possible to determine whether u is to the left or the right of v . This induces a **partial order** on the set of all vertices.

When enumerating ordered trees, the ordering of siblings is significant, so that  and  are counted separately.

This is because they are **non-isomorphic as ordered trees**, even though they are **isomorphic as rooted trees** and *isomorphic as graphs*. The one-to-one correspondence between the two vertex sets does not preserve the left-to-right order of the siblings.

An **isomorphism between ordered trees** is an isomorphism between the trees that is an isomorphism of rooted trees that *also* preserves the ordering of siblings.

What is the number of ordered trees on 5 vertices with depth 2, having 2 level-1 vertices and 2 level 2-vertices?

A 2

B 3

C 4

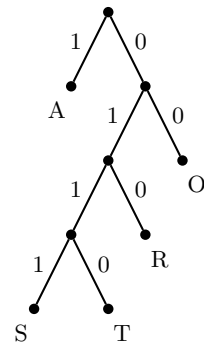
D 5

Binary trees

- A **Binary Tree** is an *ordered rooted tree* in which any vertex has at most 2 children.
- Since a binary tree is ordered, any child is either a **rightchild** or a **leftchild**.
- These are often labelled with binary digits (bits) 0 and 1 so that every vertex can be uniquely specified by a sequence of 0s and 1s.
- Due to this, binary trees are used extensively in algorithms and data structures, where the bit strings provide an efficient way to encode and/or access the vertices.

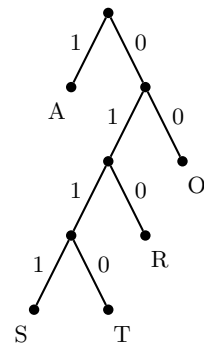
Huffman Codes

Each character (letter, digit, punctuation symbol, etc.) is represented by a binary string. In the *ASCII* code, these are all 7 bits long. In a **Huffman code**, characters are represented by strings of various lengths — commonly used characters are given shorter strings; this can save a considerable amount of storage space. The code word for each character is found by tracing down some branch of a certain rooted tree.



"STAR" is encoded by 0111 0110 1010.

What is the encoding of "SORT"?



- A** 0100001110110
- B** 01000101110110
- C** 10111011000010
- D** 0111000100110

Optimal Huffman Code

When constructing the tree for a Huffman code, we want the more frequent letters to appear on the shortest branches, so that they are represented by the shorter bitstrings (so the overall coded message occupies the least possible space).

Q: How do we achieve the minimum possible?

Algorithm

- 1 Compile a frequency table for the message.
- 2 Merge the two lowest-frequency entries in the table
- 3 Repeat step 2 until there is only one entry
- 4 Construct a binary tree from the pattern of mergers.

Summary

Tree Connected, acyclic graph (or equivalently via unique paths)

Properties Simple, planar, $e = v - 1$

Root Rooted trees have a root vertex, and then each vertex has a level

Order Ordered trees have an order between siblings

Binary Binary trees have at most two children to any parent

Huffman Efficient coding algorithm using binary trees