Mat 3770
Week 7

Spring 2014

Week 7 — Student Responsibilities

- Reading: Chapter 2.4, 3.1 (Tucker), 10.1 (Rosen)
- Homework
- Attendance **springi-ly** Encouraged

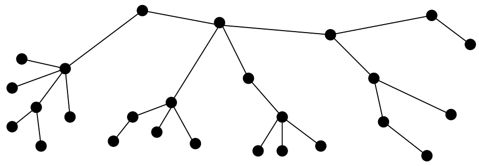
Chapter 3. Trees & Searching

Properties of Trees

- **Tree**$_1$: a connected, undirected graph with no cycles.
- **Tree**$_2$: a directed or undirected graph with a designated vertex called **root** such that there exists a unique path from the root to any other vertex in the graph.

  In an undirected graph, any vertex can be root... why is this not true in a digraph?
- **root(**T**)**: the root of tree T.

Tree

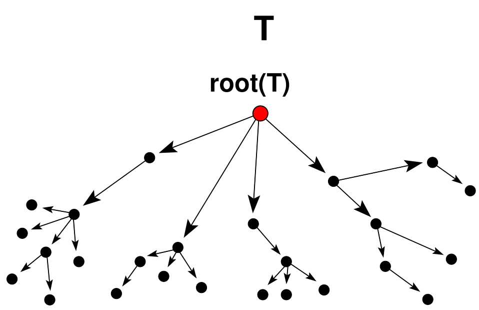


- **Rooted tree**: a directed tree (i.e., in a digraph); requires a unique root, else would have circuit.

  An unrooted tree can easily be made into a rooted tree by selecting the root and directing all edges away from it.
- **Level number**: the length of the unique path (i.e., # of edges) from the root to a particular node.

  The level number of the root is zero.
- **Leaf**: a node with no children, also known as an external node.
- **Internal node**: node which is not a leaf.

Rooted Tree



Node Relationships

- The **parent** of vertex x is the vertex y with an edge $(\vec{y}, x)$ in the rooted tree T.

  Note: root(T) has no parent.

- The **children** of a node x are all vertices z such that there exists an edge $(\vec{x}, z)$ in T.

  Note: Children have level $\#$'s one greater than their parents.

- **Siblings**: two nodes with the same parent.

Extended Node Relationships

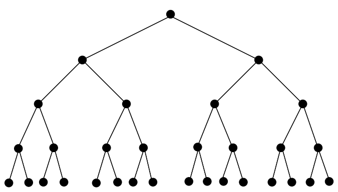
- **Ancestor** (of a node x): all nodes on the path from the root to the parent of x (including the root and parent).

- **Descendant** (of a node x): nodes on paths from x to all leaves reachable from x.

Binary Trees

- **Binary** tree: a tree in which all nodes have 0, 1, or 2 children.

- In a binary tree, we differentiate **left** child from **right** child.



Full binary tree with 31 nodes, 16 of which are leaves.

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

  An informal proof: pair each node except the root with its incoming edge. There are $n-1$ such nodes with 1 edge per node and no extra edges. Thus there are $n-1$ edges.

- **Tree traversal**: the process of **visiting** or **processing** each of the vertices in a rooted tree exactly once in a systematic manner.

- If each internal vertex of a rooted tree T has m children, T is called an m–**ary** tree.

  If m is 2, T is a binary tree.

- **Theorem 2**. Let T be an m–ary tree with n vertices, of which i vertices are internal. Then $n = mi + 1$.

  Proof: Each vertex in T, other than the root, is the child of a unique vertex (its parent). Each of the i internal nodes has m children, so there are a total of mi children. Adding the one non-child vertex, the root, we have $n = mi + 1$.

Corollary

Let T be an m–ary tree with n vertices consisting of i internal vertices and L leaves. If we know one of n, i, or L, then the other two parameters are given by the following formulas based on:

$$n = mi + 1 \quad \text{and} \quad n = i + L$$

a) Given i, then $L = (m-1)i + 1$ *and,*

$n = mi + 1$

b) Given L, then $i = \frac{L-1}{m-1}$ *and,*

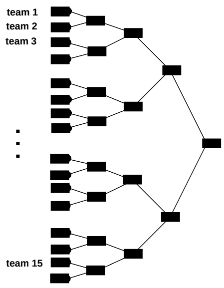
$n = \frac{mL-1}{m-1}$

a) Given n, then $i = \frac{n-1}{m}$ *and,*

$L = \frac{(m-1)n+1}{m}$

Example

If 15 teams sign up for an intramural flag football tournament, how many matches will be played? I.e., how may internal nodes will a binary tree with 15 leaves have?

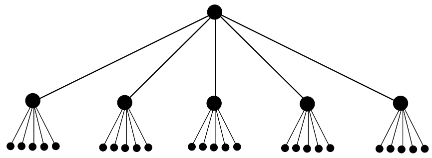


Since the tree is binary, $m = 2$.
So:

$$\begin{aligned} i &= \frac{L-1}{m-1} \\ &= \frac{15-1}{2-1} \\ &= 14 \end{aligned}$$

Spam

Suppose a chain e–mail requires the receiver to send it on to five other people (whom we'll assume have not already received a copy).



At level 4 in the tree, how many people are sending out emails?

How many emails are they sending?

What is the total number of emails sent by level 5?

And the Answers Are...

Level 0:	1 emailer
Level 1:	5 emailers
Level 2:	25 emailers
Level 3:	125 emailers
Level 4:	__________ emailers
Level 5:	__________ emailers

Balanced Trees

- The **Height** of a rooted tree is the length of the longest path from the root to a leaf.
- Alternately, the height can be defined as the largest level number of any vertex.
- A rooted tree of height h is **balanced** if all leaves are at levels h and $h - 1$.
- Balancing a tree minimizes its height.

Theorem 3. Let T be an m–ary tree of height h with L leaves. Then:

(a) $L \leq m^h$, and if all leaves are at height h, $L = m^h$

(b) $h \geq \lceil \log_m L \rceil$, and if the tree is balanced, $h = \lceil \log_m L \rceil$.

Theorem 3 — Proof — Part (a)

$L \leq m^h$, and if all leaves are at height h, $L = m^h$
(in a tree of height h with L leaves)

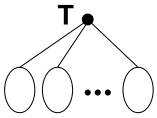
Proof by induction on the height h:

BC Let $h = 1$
An m-ary tree of height 1 has m leaves, the children of the root.
And $L \leq m^h = m^1$ $\surd$

IH Assume m-ary trees of height k, $1 \leq k < h$, have $\leq m^k$ leaves (*and if all leaves are at height* k, $L = m^k$).

IS Show m-ary trees of height h have $\leq m^h$ leaves.

An m-ary tree of height h can be broken into m subtrees rooted at the m children of the root.



These m trees have at most height $h-1$.

By the IH, each has at most m^{h-1} leaves (*and if all leaves are at height* $h-1$ *in these subtrees, then each has exactly* m^{h-1} *leaves*).

The m subtrees combined have at most $m \times m^{h-1} = m^h$ leaves (*and if all leaves are at height* $h-1$ *then there are exactly* m^h *leaves*) which are exactly the leaves of T.

Theorem 3 — Proof — Part (b)

$h \geq \lceil \log_m L \rceil$, **and if the tree is balanced,** $h = \lceil \log_m L \rceil$
(in a tree of height h with L leaves)

By Part (a):

$$L \leq m^h$$
$$\log_m(L) \leq \log_m(m^h) \quad \textit{take } \log_m \textit{ both sides}$$
$$\log_m L \leq h$$
$$\lceil \log_m L \rceil \leq h \quad \textit{since } h \textit{ is an integer}$$

If the tree is balanced, the largest value for L is m^h (if all leaves are at height h). The smallest value for L is $m^{h-1}+1$, (one leaf at height h, and the rest at height $h-1$).

So, using these upper and lower bounds:

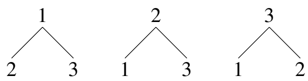
$$\begin{array}{rcccl} m^{h-1}+1 & \leq & L & \leq & m^h \\ m^{h-1} & < & L & \leq & m^h \\ \log_m(m^{h-1}) & < & \log_m(L) & \leq & \log_m(m^h) \\ h-1 & < & \log_m(L) & \leq & h \end{array}$$

Or, equivalently, $h = \lceil \log_m L \rceil$

Read over examples 3 & 4, pages 97 & 98 in Tucker.

Theorem 4. There are n^{n-2} different undirected trees on n items.

Example The number of different undirected trees on 3 distinct items, say 1…3, where order of sibling leaves is not important, is



The number of different sequences of length $n-2$ over the n items is:

$$\underbrace{n * n * n \ldots *n}_{n-2 \text{ times}} = n^{n-2}$$

Prufer Sequences

We wish to construct a **mapping**, a 1–1 correspondence, between trees on n items and $(n-2)$–length sequences of the n items.

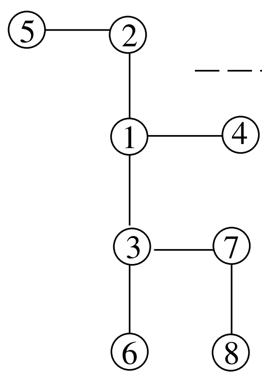
For any tree on n numbers, we can form a **Prufer Sequence** $(s_1, s_2, \ldots, s_{n-2})$ of length $n-2$ as follows:

Repeat until only two vertices remain
{

- Let L_1 be the leaf in the tree with the smallest number, and let s_1 be the number of the one vertex adjacent to it.
- Delete vertex L_1 from the graph

}

Prufer Sequence Example



— — — — — —

Prufer Sequences Define Unique Trees

Next, we must show any such $(n-2)$–length sequence of n items defines a unique n–item tree by reversing the above process.

Notes:

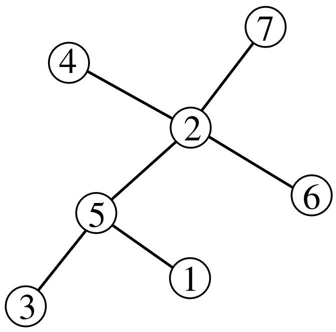
1. Leaves, vertices of degree 1, will never appear in the sequence.
2. The first number in the sequence is the neighbor of the smallest numbered leaf.
3. The smallest numbered leaf has the smallest number which doesn't appear in the sequence.

Reversing the Prufer Sequence

(1, 2, 1, 3, 3, 7) — 1 2 3 4 5 6 7 8

- The number of nodes in the tree: ________
- The smallest number not appearing: ________
- Set this leaf aside as the smallest and consider the first item (1) as a node that will be adjacent to some item in the remaining list.
- Repeat the process of identifying the smallest leaf in the remaining $(n-1)$–item tree specified by the remaining $(n-3)$–item sequence.
- For (2, 1, 3, 3, 7), item ________ is the smallest of the remaining numbers not in the sequence, so it is a leaf adjacent to item 2.

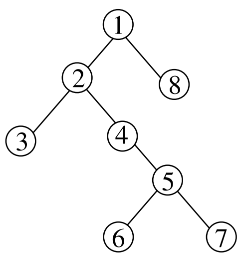
Find the Prufer Sequence



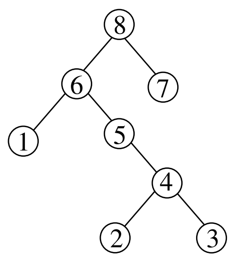
— — — — —

From the Sequence Back to the Tree

Find Prufer Sequences



— — — — — —



— — — — — —

From the Sequences Back to Trees