

ECEN 227 - Introduction to Finite Automata and Discrete Mathematics

Dr. Mahmoud Nabil
mnmahmoud@ncat.edu

North Carolina A & T State University

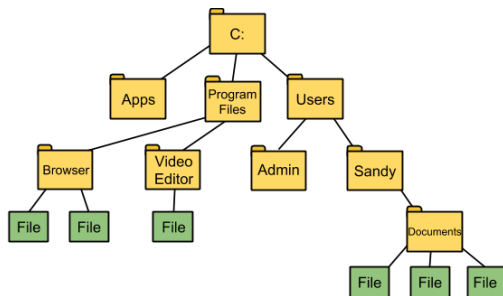
December 2, 2019

Talk Overview

- 1 Introduction to trees
- 2 Tree application examples
 - Game Tree
 - Prefix codes
- 3 Properties of trees
- 4 Tree Traversal

Tree Example

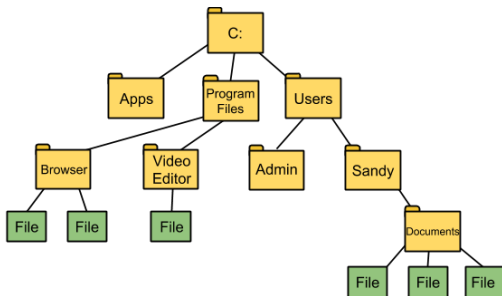
- The file system can be seen as a graph in which each folder or file is a vertex.
- There is an edge between two folders if one folder is a subfolder of the other.
- There is an edge between a file and a folder if the file resides in that folder.



Tree Defination

Tree

A tree is an **undirected graph** that is connected and has **no cycles**.



Free Tree vs Rooted Tree

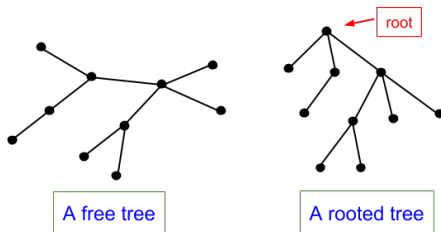
Free Tree

The tree on the left is called a **free tree** because there is no particular organization of the vertices and edges.

Rooted Tree

The tree on the right is called a **rooted tree**. The vertex at the top of the drawing is designated as the root of the tree.

Ex.



Terminologies I

Tree root

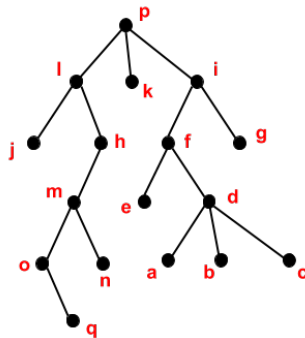
The vertex at the **top** of the drawing is designated as the root of the tree.

Vertex Level

The level of a vertex is its **distance** (i.e., **number of edges**) from to the **root**.

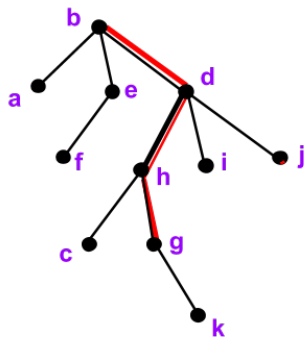
Tree Height

The height of a tree is the **deepest level** of any vertex.



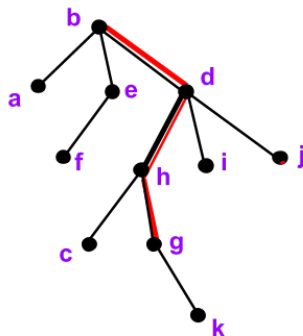
Terminologies II

- The **parent** of vertex v is the first vertex after v encountered along the path from v to the root.
 - (Ex: The parent of vertex g is h .)
- Every vertex along the path from v to the root (except for the vertex v itself) is an **ancestor** of vertex v .
 - (Ex: The ancestors of vertex g are h , d , and b .)
- If v is the **parent** of vertex u , then u is a **child** of vertex v .
 - (Ex: Vertices c and g are the children of vertex h .)



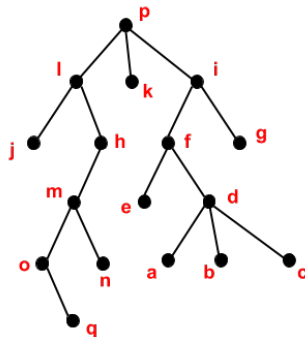
Terminologies III

- **Descendants** of v are children or children of children and so on.
 - (Ex: The descendants of vertex h are c , g , and k .)
- A **leaf** is a vertex which has no children.
 - (Ex: The leaves are a , f , c , k , i , and j .)
- Two vertices are **siblings** if they have the same parent.
 - (Ex: Vertices h , i , and j are siblings because they have the same parent, which is vertex d .)
- A **subtree** rooted at vertex v is the tree consisting of v and all v 's descendants.
 - (Ex: The subtree rooted at h includes h , c , g , and k and the edges between them.)



Excercise

- Which vertices are **ancestors** of vertex n?
- Which vertices are the **descendants** of vertex i?
- List the **leaves** in the tree.
- What is the **level** of vertex d?
- What is the **height** of the tree?
- List the level **four** vertices.
- Draw the **subtree** rooted at vertex h.
- What are the **siblings** of vertex i?



Outline

- 1 Introduction to trees
- 2 Tree application examples
 - Game Tree
 - Prefix codes
- 3 Properties of trees
- 4 Tree Traversal

Applications for Trees

Two applications for the trees will be covered:

- Game trees.
- Prefix codes.

Applications for Trees

Two applications for the trees will be covered:

- Game trees.
- Prefix codes.

We only highlight the data structure used by these applications. The algorithms/methods used are out of the scope of the course.

Outline

- 1 Introduction to trees
- 2 Tree application examples
 - Game Tree
 - Prefix codes
- 3 Properties of trees
- 4 Tree Traversal

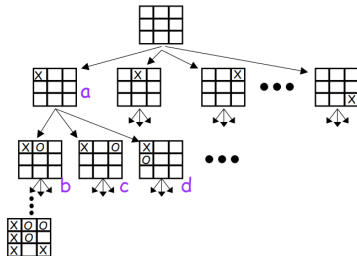
Game trees

A game tree is a **strategy** that examines all possible moves of game, and its results, in an attempt to ascertain the **optimal move**.

Game trees

A game tree is a **strategy** that examines all possible moves of game, and its results, in an attempt to ascertain the **optimal move**.

- Players **alternate** moves as in Tic-tac-toe.
- Artificially intelligent player **evaluate** the game tree to have a best move for each game state.
- Leafs are either **wins** or **loss** situations.
- The game tree can get **extremely large** in complex games such as Chess.
- In complex games, **partial tree evaluation** may be possible to overcome storage issues.



Outline

- 1 Introduction to trees
- 2 Tree application examples
 - Game Tree
 - Prefix codes
- 3 Properties of trees
- 4 Tree Traversal

Prefix codes

- Text files are stored on computers as binary strings in which each **character** is assigned a **binary code** representing that character.
- The standard way to represent text is to have a **fixed length code** for every symbol in the file.
- ASCII and Unicode are examples of fixed length encodings (**8 bits per character**).

Motivation

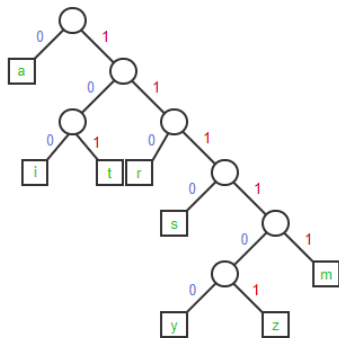
- Some characters are more frequent than the others.
- More space-efficient encodings can be achieved by **variable length codes**
- Algorithms for **generating variable length codes** are out of the scope.

Variable Length Codes

Trees are a convenient way to represent **variable length codes** for translating between text and binary.

Ex.

VLC Tree



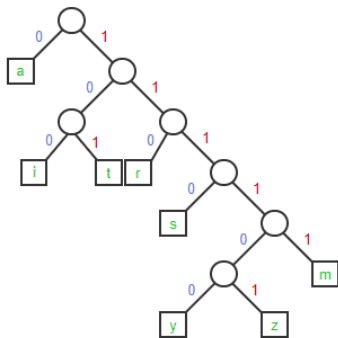
- Encode : mist

Variable Length Codes

Trees are a convenient way to represent **variable length codes** for translating between text and binary.

Ex.

VLC Tree



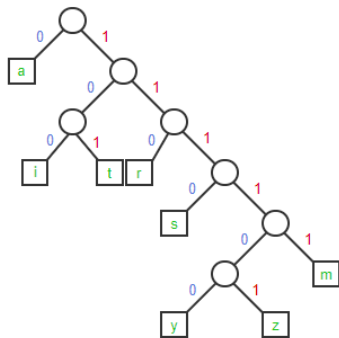
- Encode : mist
 - 111111001110101
- Decode : 11101010110

Variable Length Codes

Trees are a convenient way to represent **variable length codes** for translating between text and binary.

Ex.

VLC Tree



- Encode : mist
 - 11111001110101
- Decode : 11101010110
 - star

Why called prefix code? (Advantage 1)

- A prefix code has the property that the code for one character **can not be a prefix** of the code for another character.

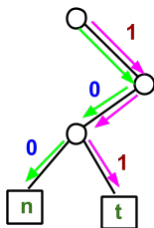
s: 101
t: 101001

s is a *prefix* of t

r: 1011
t: 101001

r is not a *prefix* of t

- The fact that the codes are organized as a tree in which characters are **only stored at the leaves** ensures the prefix property.

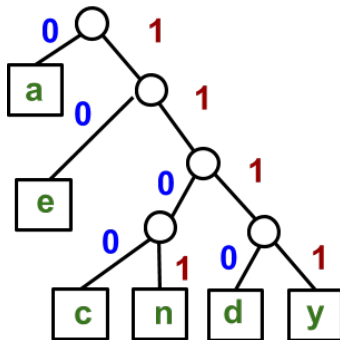


Paths to different leaves must eventually diverge.

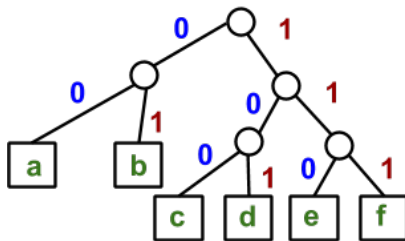
Excercise 1

- Use the tree to encode "day".
- Use the tree to encode "candy".
- Use the tree to decode "1110101101".
- Use the tree to decode "111001101110010".

VLC Tree



Excercise 2



A text file contains only characters from the set $\{a, b, c, d, e, f\}$. The frequency of each letter in the file is:

a: 5% b: 5% c: 10% d: 15% e: 25% f: 40%

- What is the average number of bits per character used in encoding the file?
- Is there a prefix tree for the set $\{a, b, c, d, e, f\}$ that would result in fewer bits per character on average for the given frequencies of the characters?

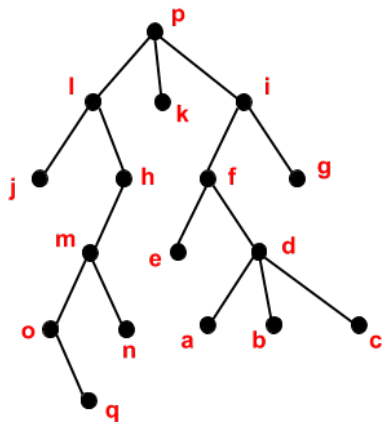
Outline

- 1 Introduction to trees
- 2 Tree application examples
 - Game Tree
 - Prefix codes
- 3 Properties of trees
- 4 Tree Traversal

Unique paths in trees.

Theorem

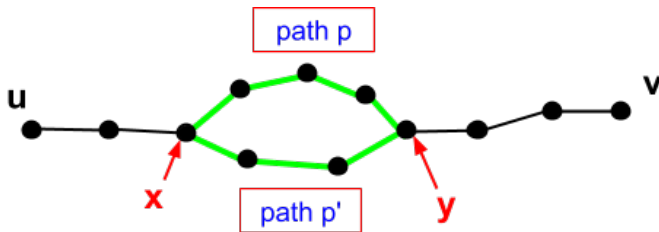
There is a unique path between every pair of vertices in a tree.



Unique paths in trees.

Theorem

There is a unique path between every pair of vertices in a tree.



- For pair of vertices u and v such that there are two distinct paths between u and v , the graph must have a cycle.
- Not a tree.

Number of leaves in a tree.

Theorem

Any *free tree* with at least two vertices has at least two leaves.

- Recall, the leaf is a vertex with no children. In other words, A leaf has a is a vertex of degree at most 1.

Number of edges in a tree.

Theorem

Let T be a tree with n vertices and m edges, then $m = n - 1$.

- Can be proved by induction. (Not required)

Exercise 1

Draw a tree with the given set of properties or explain why no such tree can exist.

- Tree, seven vertices, total degree = 14.
- Tree, four internal vertices, four leaves.
- Tree, all vertices have degree 2.

Excercise 2

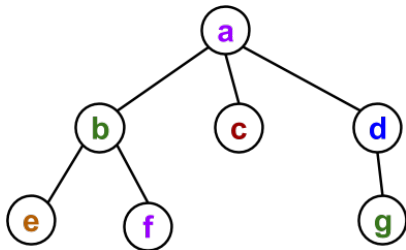
- If T is a tree with n vertices, what is the most leaves that it can have?
- Draw a tree with eight vertices that has the most number of leaves possible.

Outline

- 1 Introduction to trees
- 2 Tree application examples
 - Game Tree
 - Prefix codes
- 3 Properties of trees
- 4 Tree Traversal**

Tree Traversal

It is a way to process the information stored in the vertices by systematically visiting each vertex in a particular order.



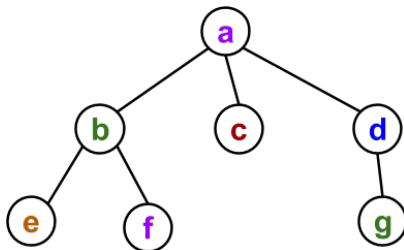
Two famous methods used in **computer science** for tree traversal

- Pre-order
- Post-order

Pre-order

list the root value, then "traverse" the children sub-trees from left to right in (Pre-order).

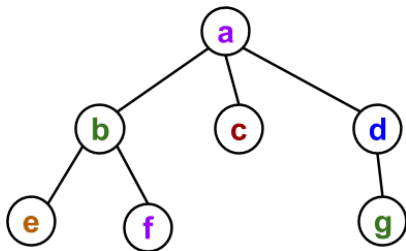
Ex.



Pre-order

list the root value, then "traverse" the children sub-trees from left to right in (Pre-order).

Ex.

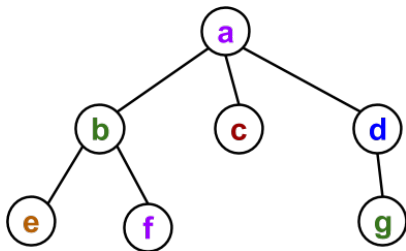


a, b, e, f, c, d, g

Post-order

"traverse" the children sub-trees from left to right in (Post-order) , then list the root value.

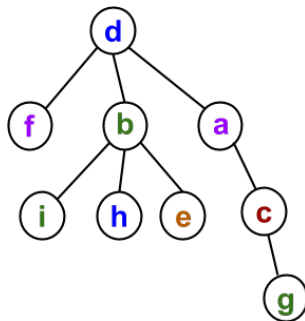
Ex.



e, f, b, c, g, d, a

Excercise

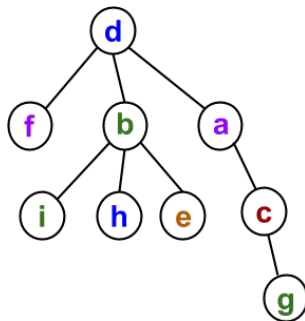
For the tree below:



- Give the order in which the vertices of the tree are visited in a post-order traversal.

Excercise

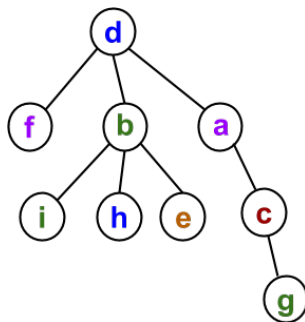
For the tree below:



- Give the order in which the vertices of the tree are visited in a post-order traversal.
 - f, i, h, e, b, g, c, a, d
- Give the order in which the vertices of the tree are visited in a pre-order traversal.

Excercise

For the tree below:



- Give the order in which the vertices of the tree are visited in a post-order traversal.
 - f, i, h, e, b, g, c, a, d
- Give the order in which the vertices of the tree are visited in a pre-order traversal.
 - d, f, b, i, h, e, a, c, g



Questions 

