Overview

• HW7 due on Tuesday Nov 30. Work on it!
• Last Lecture:
  – Shortest Path Problem
  – Traveling salesman problem
  – Trees, Spanning Trees
• This Lecture: Trees & Counting/Combinatorics
  – Rooted Trees, Binary Trees
  – Rooted Tree Apps & Tree Representations
  – Tree Traversal Algorithms

6.2 Introduction to Trees

© by Kazunori Okada, 2021
• Definition: A graph $G$ is said to be a **tree** if it is connected and has no cycle (**acyclic**).

• $G$ is said to be a **forest** if it consists of several trees.

$G$ is not connected!

• Definition: a **spanning tree** $G'$ of an undirected graph $G$ satisfies:
  - $G'$ is a subgraph of $G$
  - $G'$ consists of all vertices in $G$
  - $G'$ is a tree (forest then you have **spanning forest**)

Several Spanning trees $G'$ of $G$

Several Spanning forests $G'$ of $G$
**Kruskal Algorithm** (to find a spanning tree):

**Input:** a connected graph \( G = (V, E) \)

**Output:** a spanning tree \((V, T)\) of \(G\) \(\forall E\)

\[
T = \emptyset \quad \text{— initial tree}
\]

for each \( e \in E \) \{

\[
\text{if } (V, \{e\} \cup T) \text{ is acyclic then}
\]

\[
T = T \cup \{e\}
\]

\}

return \((V, T)\)

Note: above algorithm can be modified to obtain **minimum cost spanning tree**. How?

→ Sort \( E \) by weights in the increasing order
Definition: A rooted tree $T$ is a connected acyclic graph with one node designated as the root of the tree.

- The nodes $r_1, r_2, \ldots, r_t$ are children of $r$, and $r$ is a parent of $r_1, r_2, \ldots, r_t$.
- A node with no children is called a leaf; all nonleaves are internal nodes.
- The depth of a node in a tree is the length of the path from the root to the node.
- The height of the tree is the maximum depth of any node in the tree.
- Branches: subtrees/subgraphs
- Convention of drawing

**Binary Trees** is a rooted tree where each node has at most 2 children.

- A full binary tree occurs when all internal nodes have 2 children and all leaves are the same depth.
- A complete binary tree is an almost-full binary tree except for the deepest depth.

Note that a complete tree is not a complete graph!
A child can choose one jellybean out of two jellybeans (red, black), and one gummy bear out of three gummy bears (yellow, green, white). How many different sets of candy can the child have?

\[
\begin{align*}
\text{choose jellybean} & \quad \text{choose gummy bear} \\
R & \quad \text{R, Y} \\
G & \quad \text{R, G} \\
W & \quad \text{R, W} \\
B & \quad \text{Y, B, Y} \\
G & \quad \text{B, G} \\
W & \quad \text{B, W}
\end{align*}
\]

\[\Rightarrow 6 \text{ outcomes}\]

• Family tree - not only interesting but also useful for research in medical genetics.
• Files on your computer are organized in a hierarchical (treelike) structure (nested folders, file system).
• Algebraic expression involving binary operations can be represented by a labeled binary tree. (Compiler!)
**Binary Tree Representation**

Binary trees have special characteristics: the identity of the left and right child.

Binary tree

1 2 3
2 4 5
3 0 6
4 0 0
5 0 0
6 0 0

**Left child-right child array representation**

1 2 3
2 4 5
3 0 6
4 0 0
5 0 0
6 0 0

**Pointer representation**

1
2
3
4
5
6

**Tree Traversal Algorithms**

**Traversal of Tree**: visit every nodes of a tree in a systematic order

The 3 common tree traversal algorithms are: **preorder**, **inorder**, and **postorder** traversal.

These terms refer to the order in which the root of a tree is visited compared to the subtree nodes.

In these traversal methods, it is helpful to use the **recursive** view of a tree, where the root of a tree is a parent of the roots of subtrees.
ALGORITHM Preorder

\[ \text{PR}(\text{tree } T) \]

// Writes the nodes of a tree // with root \( r \) in preorder
\[
\begin{array}{l}
\text{write}(r) \\
\text{for } i = 1 \text{ to } t \\
\quad \text{PR}(T_i) \\
\text{end for}
\end{array}
\]

end

ALGORITHM Inorder

\[ \text{IN}(\text{tree } T) \]

// Writes the nodes of a tree // with root \( r \) in inorder
\[
\begin{array}{l}
\text{IN}(T_1) \\
\quad \text{write}(r) \\
\text{for } i = 2 \text{ to } t \\
\quad \text{IN}(T_i) \\
\text{end for}
\end{array}
\]

end

ALGORITHM Postorder

\[ \text{PO}(\text{tree } T) \]

// Writes the nodes of a tree // with root \( r \) in Postorder
\[
\begin{array}{l}
\text{for } i = 1 \text{ to } t \\
\quad \text{PO}(T_i) \\
\text{end for}
\end{array}
\]

\text{write}(r)

end

e.g. Do a preorder, inorder, and postorder traversal of the tree.

Preorder: \( a, b, e, f, c, d, g, i, h \)
Inorder: \( e, b, f, a, c, i, g, d, h \)
Postorder: \( e, f, b, c, i, g, h, d, a \)
Algebraic expressions represented as binary trees

- **Infix notation**
  - Operation symbol appears between the 2 operands.

- **Prefix notation**
  - Operation symbol precedes its operands. (Lisp)

- **Postfix notation**
  - Operation symbol follows its operands. (PS)

Inorder traversal: $(2 + x) \ast 4$

Preorder traversal: $\ast + 2 \ x \ 4$

Postorder traversal: $2 \ x \ + \ 4 \ \ast$