CMPT 225

Lecture 15 – Introduction to Trees
Last Lecture

- We saw how to ...
  - Describe how quick sort function
  - Analyze the time efficiency of quick sort
  - Discuss how to improve quick sort’s time efficiency
  - Analyze its space efficiency
  - Discuss how to improve quick sort’s space efficiency
Learning Outcomes

- At the end of the next few lectures, a student will be able to:
  - Define the following data structures:
    - Binary search tree
    - Balanced binary search tree (AVL)
    - Binary heap
    as well as demonstrate and trace their operations
  - Implement the operations of binary search tree and binary heap
  - Implement and analyze sorting algorithms: tree sort and heap sort
  - Write recursive solutions to non-trivial problems, such as binary search tree traversals
Today’s menu

- Introducing tree data structure
- Defining some tree-related terms and concepts
Way back in lecture 3, we introduced ...

Categories of data organizations

- **Linear**
  - Data organization in which each element has a unique predecessor (except for the first element, which has none) and a unique successor (except for the last element, which has none)

- **Non-Linear**
  - Data organization in which there is no first element, no last element and for each element, there is no concept of a predecessor and a successor

Categories of data organizations – cont’d

- **Hierarchical**
  - Data organization in which each element has only one predecessor -> its parent (except for the first element, which has none) and up to many successors (except for the last element(s), which has none)

- **Graph**
  - Data organization in which each element can have many predecessors and many successors
Example of data represented as hierarchical data organizations -> tree
**Example**

This structure chart shows the hierarchical relationship between the methods within a computer program. Here the line symbol represents a method call and the data arrows represent arguments and return values. For instance, it shows that the program has a method to Calculate Deductions that receives the Gross Pay as an argument and returns the Total Withheld.

Source: http://faculty.salina.k-state.edu/tmertz/Java/210proceduralprogramdesign/02proceduraldesignmethodology.pdf
What is a Tree?

Definition:
- A set of zero or more nodes partitioned into a root node and subtrees (of the root)
- Root is the access point into tree

NOTE: node != class Node of our linked list
Recursive nature of tree and subtrees
Tree Classification

- We classify trees by the maximum number of children (or subtrees) a node of the tree can have.

- Examples:
Tree terminology

- Root
- Node (vertex)
- Edge
- Leaf
- Parent
- Child
- Sibling
- Ancestor
- Descendant
Ancestors and Descendants

- If there is a path from node \( a \) to node \( b \), then \( a \) is called an \textit{ancestor} of \( b \) and \( b \) is called a \textit{descendant} of \( a \)
Path and Path Length

- A path is a sequence of nodes $v_1 \ldots v_m$ where $v_i$ is a parent of $v_{i+1}$ ($1 \leq i \leq m-1$)

- The path length is the number of nodes in a path
- From node $v_1$ to another node $v_k$
  -> path length is $k$
Activity
Height

- Height of a node $V$:
  - Length of longest path from node $V$ to a leaf

- Height of a tree $T$:
  - Height of a tree $T$'s root
Level

- If node $V$ is the root of tree $T$, then node $V$ is at level 1, otherwise level++ every time we reach root of subtree containing node $V$. 
Full tree of height $H$

- **Full tree** of height $H$ is a tree in which all nodes at level $< H$ have max number of children
- Nodes @ level $H$ have 0 children (they are leaves)
Complete tree of height $H$

- **Complete tree** of height $H$ is a tree that is full all the way down to level $H-1$ with level $H$ filled in from left to right without any gap.
Balanced tree

- A $N$-ary tree in which all $N$ subtrees of any nodes have height that differ by at most 1
Learning Check

- We can now ...
  - Define some tree-related terms and concepts
Next Lectures

- Binary Tree and Binary Search Trees