Lecture 10

- Trees
  - Definiton of trees
  - Uses of trees
  - Operations on a tree

Why use a Tree?

- Trees combines the advantages of two other data structures – an ordered array and a linked list.
- You can search a tree quickly as you can an ordered array
- You can also insert and delete items quickly as you can with a linked list

Definition of Trees

- A tree is a data structure. It is used for
  - storing files
  - compiler design
  - text processing
  - searching algorithms
- A tree can be defined non-recursively or recursively.
- Non-recursive definition: A tree consists of a set of nodes and a set of directed edges that connect pairs of nodes.
  - One node is distinguished as the root
  - Every node c, except the root, is connected by an edge from exactly one other node p. Node p is c's parent, and c is one of p's children
  - A unique path traverses from the root to each node. The number of edges that must be followed is the path length.
  - A leaf is a node that has no children.

More Definitions

- A tree with N nodes must have N-1 edges, every node except the root has an incoming edge
- Depth of a node is the length of the path from the root to the node
- Height of a node is the length of the path from the node to the deepest leaf
- Nodes with the same parents are called siblings
- If there is a path from node u to node v, then u is an ancestor of v and v is a descendant of u
- Size of a node is the number of descendants the node has including the node itself. The size of a tree is the size of the root.
- A node may be considered the root of a sub-tree. A node's sub-tree contains all its descendants.
- A node is visited when the program control arrives at the node for the purpose of carrying out some operation – checking the value of one of its data fields or displaying it. Merely passing over a node from one node to another is not considered visiting the node

More Definitions

- To traverse a tree means to visit all the nodes in some specified order.
- Level of a particular node refers to how many generations the node is from the root. It is the same thing as the depth.
- One data item in a node is usually designated the key value. This value is used to search for the item or perform other operations on it.
- In a binary tree each node can have at most two children – the left child and the right child
- In a binary search tree a node's left child must have a key less than its parent, and node's right child must have a key greater than or equal to its parent.

- A recursive definition of a tree
  - Either a tree is empty or it consists of a root and zero or more nonempty sub-trees T_1, T_2, ..., T_k each of whose roots are connected by an edge from the root.

Uses of Trees

- Expression tree is used in compiler design. The leaves of an expression tree are operands (constants or variables) and the other nodes contain operators.
- Huffman coding tree is used to implement a data compression algorithm. Each symbol in the alphabet is stored at a leaf. Its code is obtained by following the path to it from the root. A left link corresponds to 0 and a right link to a 1.
- Binary search trees allow logarithmic time insertions and access of items.

Finding a Node

- Use variable current to hold a node that it is examining
- Start at the root
- In the while loop compare value to be found (key) with the iData value (key field). If key is less than this value then current is set to the node's left child
- If key is greater than this value then current is set to the node's right child.
- If current becomes null, then we have not found the node
- If node is found then while loop is exited and the found node is returned

Inserting a Node

- First find the place to insert the node. (Same process as finding a node that turns out not to exist)
- Follow path from the root to the appropriate node which will become the parent of the new node.
- Once the parent has been found the new node is connected as its left or right child, depending on whether the new node's key is less than or greater than that of the parent.

Traversing a Tree

- Inorder traversal – the left child is recursively visited, the node is visited, and the right child is recursively visited.
- Preorder traversal – a node is visited and then its children are visited recursively.
- Postorder traversal – a node is visited after both children are visited.

Parsing Expressions

- Infix notation: A * (B + C)
- Prefix notation: * A + B C
- Postfix notation : A B C + *

Finding Minimum and Maximum Values

- To get the minimum go to the left child of the root; then go to the left child of that child, and so on, until you come to a node that has no left child. That node is a minimum.
- To get the maximum go to the right child of the root, then go to the right child of that child, and so on, until you come to a node that has no right child. That node is a maximum.

Deleting a Node

- Find the node you want to delete. Once you have found the node there are three cases to consider:
  - The node is a leaf (has no children)
  - The node has one child
  - The node to be deleted has two children
- To delete a leaf node simply change the appropriate child field in the node's parent to be null. The space in memory occupied by that node will be reclaimed by the Java garbage collector.
- If the node to be deleted has one child then directly connect the parent of this node to the child of this node.
- If the node to be deleted has two children then replace this node with its inorder successor.

Finding the inorder successor

- The inorder successor is the smallest node of the set of nodes that are larger than the original node.
- When you go to the original node's right child, all the nodes in the resulting sub-tree are greater than the original node.
- To obtain the minimum value in this sub-tree follow the path down all the left children. The first node that has no left child is the successor node.

Number of Nodes per Level

No. of Levels	**No. of Nodes**
1	1
2	3
3	7
4	15
5	31
...	...
10	1023
...	...
20	1048575

Num of Nodes = $2^{\text{Level}} - 1$

Efficiency of Binary Trees

- Most operations with trees involve descending from level to level.
- In a full tree, there is one more node on the bottom row than in the rest of the tree. This about half of all operations require finding a node on the lowest level.
- In an unordered array or a linked list containing 1,000,000 items it would take on the average 500,000 comparisons to find the one you wanted. In a tree of 1,000,000 items it takes 20 (or fewer) comparisons.