Tree

Yu-Tai Ching
Department of Computer Science
National Chiao Tung University

Tree

- A special kind of graph $G = (V, E)$, path between any pair of vertices is well defined.
- data are organized in a hierarchical manner.
- Used to present a structure, (Figure 5.1)
- Used to store data, a search structure,

Definition of a Tree and Termminology

Definition A "tree" is a finite set of one or more nodes s.t.

1. There is a specially designated node called the root.
2. The remaining nodes are partitioned into $n \geq 0$ disjoint sets T_1, ..., T_n, where each of these sets is a tree. T_1, ...,T_n are called the subtrees of the root.

- A recursive definition.
- Figure 5.2, root A, 3 substrees rooted at B, C, and D.
- Usually draw the root at the top.
- Number of subtrees of a node: degree, A is 3, B is 2, and F is 0.
- degree zero nodes are leaves, K, L, F, G, M, I, J.

- other nodes are referred to as nonterminal.
- Children, roots of subtrees of a node X are the children of X.
- X is the parent of its children,
- those children are siblin to each other.
- "degree of tree" is the maximum degree of the nodes in the tree.
- "ancestors" of a node X are the nodes on the path from X to the root.
- "level": root at level 1, node X is at level l, X's children are at level $l + 1$.
- "height" or "depth" of a tree is defined to be the maximum level of any node in the tree.

Tree Representation

Lemma 5.1: If T is a k-ary tree (a tree of degree k) with n nodes, each having a fixed size (k) linked (child) fields, then $n(k-1)$+1 of the nk child fields are 0.

Proof each non-zero child field points to a node except the root. There are $(n-1)$ child fields are non-zero. There are nk child fields, thus $nk-(n-1)$ = $n(k-1)+1$ child fields are zero.

- Lemma says that if each node has k child fields, we could wast a lot of memory space for large k.
- Generalized list
- left child-right sibling, each node has two child fields, the “left child” and “right sibling”, figures 5.5 and 5.6.
- a degree 2 tree, binary tree.

Binary Tree

Definition A “binary tree” is a finite set of nodes that either is empty or consists of a root and two disjoint binary trees called the “left subtree” and the “right subtree”.

- sub-trees are ordered in binary tree, figure 5.9
- Figure 5.10 a, skew tree, b balance (complete tree).

Properties of Binary Trees

Lemma 5.2 Maximum number of nodes:

1. The maximum number of nodes on level i of a binary tree is 2^{i-1}, $i \geq 1$.
2. The maximum number of nodes in a binary tree of depth k is $2^k - 1$, $k \geq 1$.

Proof of 1. By induction:

- Induction Bases: The root is the only node on level $i = 1$. The maximum number of nodes on level $i = 1$ is 2^{i-1} = 2^0 =1. (You can check for $i = 2$, $i = 3$, $\ldots$.)
- Induction Hypothesis: Let i be an arbitrary positive integer greater than 1. Assume that the maximum number of nodes on level j is 2^{j-1}, $\forall j < i$. We shall try to show level i has 2^{i-1} nodes.
- Induction Step: Since level $i - 1$ has 2^{i-2} nodes (by induction hypothesis), and since each each node in level $i - 1$ has at most two children, there are at most $2 \cdot 2^{i-2}$ nodes on level i. That is 2^{i-1}.

Proof of 2.,

- We know the maximum number of nodes on level i is 2^{i-1}.
- The maximum number of nodes in a tree is bounded by the sum of all nodes in all the levels.
- $\sum_{i=1}^{k}$(maximum number of nodes on level i) = $\sum_{i=1}^{k} 2^{i-1}$ = $2^k - 1$.

Properties of Binary Trees

Lemma 5.3 Relation between number of leaf nodes and degree-2 nodes:
For any nonempty binary tree, T, if n_0 is the number of leaf nodes and n_2 the number of degree 2 nodes, then $n_0 = n_2 + 1$.
Proof: Let n_1 be the number of degree 1 nodes. Since there are n nodes. ewe have

$$n = n_0 + n_1 + n_2.$$

Every node except the root has a branch leading to it. Let B be the number of branches,

$$n = B + 1.$$

All branches stem from a node of degree one or two. We have

$$B = n_1 + 2n_2.$$

Put all these togather,

$$n = B + 1 = n_1 + 2n_2 + 1 = n_0 + n_1 + n_2,$$

finally we get

$$n_0 = n_2 + 1.$$

Definition: A “full binary tree” of depth k is a binary tree of depth k having $2^k - 1$ nodes (the maximum number of nodes), $k \geq 0$. Figure 5.11.

Definition: A binary tree with n nodes and depth k is “complete” iff its nodes correspond to the nodes numbered from 1 to n in the full binary tree of depth k.

Important fact: height of a complete binary tree with n nodes is k = $\lceil \log_2(n+1) \rceil$. AND any binary tree cannot be more dense than a complete binary tree, thus the height is at least $\lceil \log_2(n+1) \rceil$.

Binary Tree Representation- Array

- Figure 5.11 suggests array representation of a tree
- the number is the index of the array (we don't use $A[0]$ in this case).
- In a tree, given a node i, we should be able to access the children of i or the parent i in $\Theta(1)$ time.

Lemma 5.4: If a complete binary tree with n nodes is represented sequentially, then for any node with index i, $1 \leq i \leq n$, we have

1. $parent(i)$ is at $\lfloor i/2 \rfloor$ if $i \neq 1$. If $i = 1$, i is at the root and has no parent.
2. $leftChild(i)$ is at $2i$ if $2i \leq n$. If $2i > n$, then i has no left child.
3. $rightChild(i)$ is at $2i + 1$ if $2i + 1 \leq n$. If $2i + 1 > n$ then i has no right child.

Proof We prove 2., 3. is a immediate consequence of 2.. And 1. follows from 2. and 3..

Proof of 2. By induction on i.

- For $i = 1$, clearly the left child is 2 unless $2 > n$.
- Assume that $\forall j$, $1 \leq j \leq i$, $leftChild(j)$ is $2j$. Based on this asumption, we try to argue that $leftChild(i + 1)$ is $2(i + 1)$.
- Note that the two nodes immediately preceeding $leftChild(i + 1)$ are the right child and left child of i.
- The left child is at $2i$. Hence the left child of $(i + 1)$ is at $2i + 2 = 2(i + 1)$. Unless $2(i + 1) > n$ in which case $i + 1$ has no left child.

- Array representation works fine for a complete binary tree.
- It works for an arbitrary binary tree
  - The space required is not linearly proportional to the number of nodes.
  - Consider a skew binary tree having of depth k, the space required is $2^k - 1$. Figure 5.12.

Binary Tree Representation- Linked Representation

template <class T> class $Tree$;
template <class T>
class $TreeNode\{$
friend class $Tree < T >$;
private:
    $Tdata$
    $TreeNode < T > *leftChild$;
    $TreeNode < T > *rightChild$;
};
template <class T>
class $Tree$ $\{$
public:
private:
    $TreeNode < T > *root$;
}; Figure 5.13, 5.14