Dictionaries

Remember ADTs?

Dictionary
- MakeEmpty, IsEmpty
- Find
- Insert, Delete

Using Dictionaries

Dictionaries are everywhere in computers
## Implementations of Dictionaries

<table>
<thead>
<tr>
<th>Implementation</th>
<th>Time to Find</th>
<th>Time to Insert</th>
<th>Time to Delete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unordered List</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unordered Array</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sorted Array</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Trees

Trees in the World

![Tree Image](image_url)

### Definitions

**Anatomy of a Tree**

- **Root**: The topmost node of the tree.
- **Node**: Any non-root node.
- **Edge**: A connection between two nodes.
- **Children**: The nodes directly connected to a given node.
- **Siblings**: Nodes that have the same parent.
- **Leaves**: Nodes with no children.
### Depth and Height

- Depth = 2
- Height = 2
- Height = 1

### Recursive Definition

A tree is...

A root

A root with distinct child trees

### Binary Trees

- Empty Tree
- Left Child
- Right Child
- Full
- Perfect
- Complete
- Level 0
- Level 1
- Level 2
- Level 3
Counting with Perfect Trees

- How many nodes in level \( i \) of a perfect tree?

- How many nodes, total?

- What fraction of the tree is in the last level?

Binary Tree Traversals

- A traversal is an order for visiting all nodes of a tree

- Easy to think of in terms of expression trees

Three Kinds of Traversals

- Preorder: \(+ \times 3 2 4\)
- Inorder: \(3 \times 2 + 4\)
- Postorder: \(3 2 \times 4+\)
  - For an expression, this is prefix order
### Code for Traversals

```cpp
void TreeNode::PreOrder() {
    Visit();
    if (LeftChild()) LeftChild()->PreOrder();
    if (RightChild()) RightChild()->PreOrder();
}

void TreeNode::InOrder() {
}

void TreeNode::PostOrder() {
}
```

### Binary Search Trees

```
Node* Node::Find(Key k) {
    if (key == k) return this;
    if (k < key && LChild())
        return LChild()->Find(k);
    if (k > key && RChild())
        return RChild()->Find(k);
    return NULL;
}
```

```
void Node::Insert(Key k) {
    if (k <= key) {
        if (LChild())
            LChild()->Insert(k);
        else
            SetLChild(new Node(k));
    } else
        SetRChild(new Node(k));
}
```

### Traversals on BSTs

```
Traversals on BSTs

- Preorder
- Inorder
- Postorder
```
BST Deletion IV

- Locate node to delete
- If no children, delete
- If one child, replace with child
- Otherwise,
  - Locate successor (or predecessor)
  - Replace with successor (or predecessor)

BST Analysis

- Perfect Tree
- Unbalanced Tree

Complete? Full?
Making an Unbalanced Tree