Lab 3

C Strings
Binary Trees

September 22nd, 2010
James Marshall

Tips

- On hobbes, always type “bash”
  - Can use arrow keys
  - History
  - Auto complete
- = vs. ==
- Set pointer declarations to NULL
- Semicolons

Motivation

- For learning C: To graduate
- For Binary Trees: They are cool!

No really, Binary Trees are cool.

- Searching a balanced, sorted binary tree:

Elements in Tree	Steps to Locate Single Element
1,000	10
1,000,000	20
1,000,000,000	30
1,000,000,000,000	40
1,000,000,000,000,000	50

First, C Strings

- Always “\0” (NULL) terminated
- Can statically allocated arrays
  - Sets a limit on size
  - Waste of memory

Dynamic Allocation

- Slightly more complicated
- Size determined at run-time
- Can be any size needed
- Need to allocate and free memory

Trees in Comp Sci

- Very common
- A special type of graph
- “Grow” downwards
- Node
- Root
- Depth
- Leaf
- Ancestor, Parent, Children

Binary Search Tree

- Each node has 0 – 2 children
- Keys
- Left child key < parent key < right child key

Insertion

- Is current node NULL?
  - Done! Insert here.
- Is new node > current node?
  - Insert into RIGHT subtree
- Is new node < current node?
  - Insert into LEFT subtree

That's Recursion!

- Easy, right?
- Simple to perform operations on trees recursively
- Always:
  - Base Case
  - Recursive Case

Search

- Is current node NULL?
  - Done! But we didn't find it. :-(
- Does search key == current node key?
  - Done! You found it! :-)
- Is search key < current node key?
  - Search left tree.
- Is search key > current node key?
  - Search right tree.

Deletion

- More complicated
  - No children: Delete
  - One child: Delete, replace with child
  - Two children: Replace with next or previous predecessor, Delete the predecessor
- Not on homework

Traversals

- “Walking” the tree
  - Visit each node exactly once
- Defined by order nodes are visited
  - In-Order
  - Depth First
  - Breath First
- If unsorted, these can be searches

Depth First Traversal

- Visit
  - Self
  - Left
  - Right

Breath First Traversal

- Not as simple
- Queue of unvisited
- Visit
  - Self
  - Place left child in Queue
  - Place right child in Queue
  - Visit next in Queue

In-Order

- To visit nodes in the order of their keys:
- Visit
  - Left
  - Self
  - Right