Review for Final

Chapters 10 – 13

CSc 212 Data Structures

Trees and Traversals

- Tree, Binary Tree, Complete Binary Tree
  - child, parent, sibling, root, leaf, ancestor,...
- Array Representation for Complete Binary Tree
  - Difficult if not complete binary tree
- A Class of binary_tree_node
  - each node with two link fields
- Tree Traversals
  - recursive thinking makes things much easier
- A general Tree Traversal
  - A Function as a parameter of another function

Binary Search Trees (BSTs)

- Binary search trees are a good implementation of data types such as sets, bags, and dictionaries.
- Searching for an item is generally quick since you move from the root to the item, without looking at many other items.
- Adding and deleting items is also quick.
- But as you'll see later, it is possible for the quickness to fail in some cases -- can you see why? (unbalanced)

Heaps

- Heap Definition
  - A complete binary tree with a nice property
- Heap Applications
  - priority queues (chapter 8), sorting (chapter 13)
- Two Heap Operations – add, remove
  - reheapification upward and downward
  - why is a heap good for implementing a priority queue?
- Heap Implementation
  - using binary_tree_node class
  - using fixed size or dynamic arrays

B-Trees

- A B-tree is a tree for sorting entries following the six rules
- B-Tree is balanced - every leaf in a B-tree has the same depth
- Adding, erasing and searching an item in a B-tree have worst-case time O(log n), where n is the number of entries
- However the implementation of adding and erasing an item in a B-tree is not a trivial task.

Trees - Time Analysis

- Big-O Notation :
  - Order of an algorithm versus input size (n)
- Worse Case Times for Tree Operations
  - O(d), d = depth of the tree
- Time Analysis for BSTs
  - worst case: O(n)
- Time Analysis for Heaps
  - worst case O(log n)
- Time Analysis for B-Trees
  - worst case O(log n)
- Logarithms and Logarithmic Algorithms
  - doubling the input only makes time increase a fixed number

Searching

- Applications
  - Database, Internet, AI...
- Most Common Methods
  - Serial Search – O(n)
  - Binary Search – O(log n)
  - Search by Hashing - O(k)
- Run-Time Analysis
  - Average-time analysis
  - Time analysis of recursive algorithms

Quadratic Sorting

- Both Selectionsort and Insertionsort have a worst-case time of $O(n^2)$, making them impractical for large arrays.
- But they are easy to program, easy to debug.
- Insertionsort also has good performance when the array is nearly sorted to begin with.
- But more sophisticated sorting algorithms are needed when good performance is needed in all cases for large arrays.

O(NlogN) Sorting

- Recursive Sorting Algorithms
  - Divide and Conquer technique
- An O(NlogN) Heap Sorting Algorithm
  - making use of the heap properties
- STL Sorting Functions
  - C++ sort function
  - Original C version of qsort

Graphs

- Examples/Applications
- Terminologies
- Representations
- Graph Traversal