Announcements

- Office Hours During the Final Week
  - Lin's: Wed/Thur 3:00pm - 4:00pm
  - TA's : Mon/Wed 2:00pm - 3:30pm

- Extra Help Session Before Final:

  12/12/05 in SN115, 3:30pm – 5:00pm

Materials from Text

- Algorithmic Basics: Chapter 1-5
- Sorting Methods: Chapter 6-9
- Hash Tables: Chapter 11
- BST and R-B trees: Chapter 12 & 13
- Dynamic Programming: Chapter 15
- Graph Algorithms: Chapter 22
- Greedy Algorithms: Chapter 16, 23-24

Cheat Sheet

- Important Formulas: summations, Master Theorem for recurrences, etc.
- Definitions: MST, Shortest Paths, etc.
- Basic Lemmas & Theorems
- Essences of Key Algorithms

Algorithmic Basics & Math

- Asymptotic Notation
- Solving Recurrences
- Summations
- Proof Techniques

Sorting Methods

- There are a number of $\Theta(n \lg n)$ sorting methods: merge sort, heapsort, quicksort. Given a list of n numbers, the kth smallest can be found in $\Theta(n)$ time
- Assuming comparisons are used, you cannot sort faster than $\Omega(n \lg n)$ time. Any comparison-based algorithm can be written as a decision tree, and there are $n!$ possible outcomes, so even a perfectly balanced tree would require height of at least $\Omega(\lg n!) = \Omega(n \lg n)$.
- Counting sort takes $\Theta(n + k)$ time, given n integers in a range of 1 to k
- Radix Sort takes $\Theta(d(n + k))$ time for n integers of d digits, each ranging from 1 to k

Hash Tables

- Operations: Insert, Search, Delete
- For all operations: expected time $O(1)$, worst case $\Omega(n)$
- Chaining vs open addressing

Search Trees

- Operations: Insert, Search, Delete, Predecessor, Successor, Minimum, Maximum
- Running time proportional to tree height
  - Can be linear if tree = chain
- Red-black trees guarantee approximate balance
  - Red-black properties
  - Use rotation to maintain balance
  - Rules for adding, deleting elements

Graph Algorithms

- *Representation*: adjacency lists & matrices
- *BFS*: Traverse a graph in a breadth-first order. It can be applied to compute the shortest paths from a single source in a digraph without edge weights in $\Theta(n+e)$ time.
- *DFS*: Traverse a graph in a depth-first order. Useful for many problems in $\Theta(n+e)$ time.
  - Topological sort: Sort the vertices of a DAG in topological order. Vertices finish in reverse topological order.
  - Connected components: Break a graph into its connected components. Each DFS tree is a connected component.
  - Cycle Detection: Determine if a (di)graph has a cycle.
  - Strong Components: Break a graph into its strong CC’s.
  - Articulation points: Determine “cut vertices” in a graph.

Greedy Algorithms

By making a locally optimal (greedy) choice, greedy algorithms seek a global optimal solution. They do NOT always yield optimal solutions, but for many problems they do……

- **MST: Compute the minimum spanning tree of a weighted graph. Two algorithms presented in class that run in $\Theta((n+e)\lg n)$ time: Kruskal's and Prim's algorithm.**
- **Single Source Shortest Paths: Compute the shortest path from a single source vertex in a weighted graph with non-negative weights: Dijkstra's algorithm that runs in $\Theta((n+e)\lg n)$ time.**

Dynamic Programming

Similar to divide-and-conquer, but more……

- **Substructure:** decompose problem into smaller sub-problems. Express the solution of the original problem in terms of solutions for smaller problems.
- **Table-structure:** Store the answers to the sub-problem in a table, because sub-problem solutions may be used many times.
- **Bottom-up computation:** combine solutions on smaller sub-problems to solve larger sub-problems, and eventually arrive at a solution to the complete problem.

Possible Questions

- True or False (with justifications)
- Short Questions
- Design Algorithms to solve problems
  - Dynamic Programming
  - Greedy Algorithms
  - Graph Algorithms
  - Sorting & Selection
  - Binary Trees & Hash Tables

Preparing for Finals

- Do the assigned weekly readings
- Review the previous homework, (practice) exam & quizzes
- Go over all lecture notes
- Understand (not memorize)
- Get plenty of sleep

Techniques for Taking Exams

- Be neat, clear, precise, formal
  - You’ll be graded on correctness, simplicity, elegance & clarity
  - Clearly state your assumptions!!!
- Take time to read and think about the problems before you start writing
- Make intelligent guesses on T/F and short questions, if you’re not sure…
- Move on if you get stuck and come back to it later if you have time……

Plan to make more than one pass through the exam

- Try to get an idea on how to do each “hard problem”
  - No more than 5 min on any one problem at first
  - Make a note of any ideas you have
  - If you can’t get an idea really soon, at least try to understand what the problem is asking for
- Then go back and do the details or make a second try at figuring out how to do it
- Uses your subconscious
- Helps you follow last advice from previous slide