# CS 561, Final Review

Jared Saia
University of New Mexico

# Final

- About 5 main problems
- As usual, the solutions will require lots of thinking, but not much writing.
- I expect a class mean of between 60 :( and 70 :) percent

# Topics Covered

- Probability and Randomized Algorithms: Linearity of Expectation, Union Bounds, Markov's inequality. Randomized Quicksort, Count-min sketch, Bloom Filters.
- Recurrence Relations and Induction (Chapter 3 and 4): Defns of big-O and friends, recursion trees, Master method, annihilators and change of variables; Proof by induction!
- Dynamic Programming: String Alignment, Matrix Multiplication, Longest Common Subsequence (Chapter 15)
- Greedy Algorithms: Activity selection, fractional knapsack, MST, proof via exchange arguments (Chapter 16)
- Amortized Analysis: Aggregate Method, Accounting Method, Potential Method, Dynamic Array (Chapter 17)

# Topics Covered

- Disjoint-Sets: Union by Rank and Path Compression, Amortized Costs (Chapter 21)
- Minimum Spanning Trees: Kruskal's and Prim's Algorithm, Safe Edge Theorem and Corollary
- Shortest Paths: Dijkstra's, Bellman-Ford, Floyd-Warshall (Chapters 22 23,24,25)
- NP-Hard Problems: Definitions of P, NP, co-NP, NP-Hard, and NP-Complete; Reductions (i.e. how to show that a problem is NP-Hard); Classic NP-Hard problems: CIRCUIT-SAT, SAT, 3-SAT, COLORING, CLIQUE, VERTEX COVER, INDEPENDENT SET, HAMILTONIAN CYCLE, TSP. (Chapter 34)
- Approximation Algorithms: Vertex Cover, TSP.
- Linear Programming and Gradient Descent. Primarily how to use.

# Example Problem - Short Answer

Collection of true/false questions, matching and short answer questions. Some examples:

- T/F questions covering all topics
- Multiple Choice e.g. I give you some "real world" problems and ask you which algorithm we've studied in class that you would use to solve each of them; I give you some problems and ask you how fast they can be solved, etc.
- Know the resource bounds for all algorithms covered.

# Induction/Recurrences

Possibilities:

- **Proof by Induction**
- **Recurrence Relations** Uses: Analyzing algorithms. Creating Dynamic Programs, Greedy Algorithms, Linear Programming, etc.
- **Exchange Arguments** for Greedy Algorithms

# Dynamic Programming

- Key focus: getting the correct recurrence relation
- Likely will be related to some problem we did in class and/or homework
- Practice solving a big problem by using solutions to sub-problems

# Graph Theory

- Possibility 1: MST and Safe Edge Theorem
- Possibility 2: Single Source Shortest Paths (Dijkstra's and Bellman-Ford)
- Possibility 3: All Pairs Shortest Paths

# NP-Hard/Complete Problems

Some Possibilities:

- I give you a problem and ask you to prove it's NP-Hard by a reduction from another NP-Hard Problem. You have to choose which problem to reduce from.
- Be careful to get the direction right!
- I give you an NP-Hard Problem and ask you give an approximation algorithm for it (e.g. a variant of something already seen in class)
- Be prepared to show that problem is NP-Complete

# Beyond Worst Case

- Probabilistic Analysis: Linearity of Expectation, Markov's, Union Bounds, etc.
- Amortized Analysis: Aggregate, Accounting and Potential Methods

# LP and Gradient Descent

- Know how to use Linear Programming and Gradient Descent
- Key focus: Transforming one problem into another
- Tools to do this:
  - Graph Problems $\rightarrow$ solving recurrence relations
  - recurrence relations $\rightarrow$ recurrence inequalities (SSSP)
  - Maximizing products $\rightarrow$ Maximizing sums of logs

# How to Study?

A: Solve Problems! Start with worked examples from lecture and hw problems. Next, problems from old finals.

1. Cover up the answer
2. Try to solve
3. If you get stuck, uncover some lines of the solution
4. Rinse, Wash, Repeat

# More Problems

Hungry for more problems? Good!

1. Redo HW problems
2. Do worked examples from our textbook
3. Problems from 561 and 362 finals
4. Do problems in Jeff Erickson's book *Algorithms*. This book is free, the link is on class web page.
5. Website `leetcode.com` is a great resource. Click on the tag "dynamic programming" or "greedy algorithm" for job interview type questions in that area.