# CSE 421: Review

Larry Ruzzo

Summer 2011

# Complexity, I

Asymptotic Analysis

Best/average/**worst** cases

Upper/Lower Bounds

Big O, Theta, Omega

Analysis methods

- loops
- recurrence relations
- common data structures, subroutines

# Graph Algorithms

Graphs

- Representation (edge list/adjacency matrix)
- Breadth/depth first search
- Connected components
- Shortest paths/Bipartitness/2-Colorability
- DAGS and topological ordering
- DFS/Articulation points/Biconnected components

# Design Paradigms

Greedy

Divide & Conquer

- recursive solution, superlinear work, balanced subproblems, recurrence relations, solutions, Master Theorem

Dynamic Programming

- recursive solution, redundant subproblems, few
- do all in careful order and tabulate
- (usually far superior to "memoization")

Powerful Subproblems

- Flow, Matching, Linear Programming

# Examples

Greedy

- Interval Scheduling Problems (3)
- Huffman Codes
- Examples where greedy fails (stamps/change, scheduling, knap, RNA,…)

# Examples

Divide & Conquer

- Merge sort
- Closest pair of points
- Integer multiplication (Karatsuba)
- Powering

# Examples

Dynamic programming

- Weighted Interval Scheduling
- Max Subarray Sum
- Knapsack
- String Search with Wildcards
- Edit Distance/String Alignment
- Counting Solutions
- Shortest Paths
- RNA Folding

# Examples

Flow and matching

- Residual graph, augmenting paths, max-flow/min-cut, Ford-Fulkerson and Edmonds-Karp algorithms, integrality, reducing bipartite matching to flow

# Complexity, II

P vs NP

- Big-O and poly vs exponential growth
- Definition of NP – hints/certificates and verifiers
- Example problems from slides, reading & hw
  - SAT, 3-SAT, circuit SAT, vertex cover, quadratic Diophantine equations, clique, independent set, TSP, Hamilton cycle, coloring, max cut, knapsack
- $P \subseteq NP \subseteq Exp$ (and worse)
- Definition(s) of (polynomial time) reduction
- $SAT \leq_p VertexCover$ example (how, why correct, why $\leq_p$, implications)
- Definition of NP-completeness
- NP-completeness proofs
- 2x, 1.5x approximations to Euclidean TSP

# Some Typical Exam Questions

Give O( ) bound on 17n*(n-3+logn)

Give O( ) bound on some code `{for i=1 to n {for j …}}`

True/False: If X is $O(n^2)$, then it's rarely more than $n^3$ +14 steps.

Explain why a given greedy alg is/isn't correct

Give a run time recurrence for a recursive alg, or solve a simple one

Simulate any of the algs we've studied

Give an alg for problem X, maybe a variant of one we've studied, or prove it's in NP

Understand parts of correctness proof for an algorithm or reduction

Implications of NP-completeness

Reductions

NP-completeness proofs


~~Hell's library~~ → 421 Final