CSE417: Review

Larry Ruzzo

Winter 2009

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
- Bipartitness/2-Colorability
- DAGS and topological ordering

Design Paradigms

Greedy

Dynamic Programming

- recursive solution, redundant subproblems, few
- do all in careful order and tabulate

Divide & Conquer

- recursive solution
- superlinear work
- balanced subproblems

Examples

Greedy

- Interval Scheduling Problems
- Huffman Codes

Examples

Dynamic programming

- Fibonacci
- Making change/Stamps
- Weighted Interval Scheduling
- RNA

Divide & Conquer

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

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, VertexCover, quadratic Diophantine equations, clique, independent set, TSP, Hamilton cycle, coloring, max cut
- P $\subseteq$ NP $\subseteq$ Exp
- Definition of (polynomial time) reduction
- SAT $\le_p$ VertexCover example (how, why correct, why $\le_p$, implications)
- Definition of NP-completeness
- 2x approximation to Euclidean TSP

Some Typical 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

Convert a simple recursive alg to a dynamic programming solution

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