CSE 417: Review

Larry Ruzzo
Complexity, I

Asymptotic Analysis
Best/average/worst cases
Upper/Lower Bounds
Big O, Theta, Omega
definitions; intuition
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/bipartitiveness/2-Colorability
DAGS and topological ordering
DFS/articulation points/biconnected components
Design Paradigms

Greedy
emphasis on correctness arguments, e.g. stay ahead, structural characterizations, exchange arguments

Divide & Conquer
recursive solution, superlinear work, balanced subproblems, recurrence relations, solutions, Master Theorem

Later:
Dynamic 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)
Matrix multiplication (Strassen – see HW)
Powering
Some Typical Exam Questions

Give $O(\cdot)$ bound on $17n^*(n-3+\log n)$

Give $O(\cdot)$ bound on some code  
{\textbf{for} i=1 \textbf{to} n \{\textbf{for} j \cdots \}}

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 on given input
Midterm Friday, 5/9/2014

Closed book, no notes

(no bluebook needed; scratch paper may be handy; calculators unnecessary)

All up through “Divide & Conquer”

assigned reading up through Ch 5;

slides

homework & solutions
Final Review
Final Exam Coverage

Comprehensive, all topics covered
(but with post-midterm bias)
assigned reading
slides
homework & solutions
midterm review slides still relevant, plus those below
Design Paradigms

Greedy
- emphasis on correctness arguments, e.g. stay ahead, structural characterizations, exchange arguments

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; \textbf{OPT table}
  - (usually far superior to “memoization”)


Examples

Dynamic programming
  Fibonacci
  Making change/Stamps
  Weighted Interval Scheduling
  RNA
  Knapsack
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 ⊆ NP ⊆ Exp (and worse)

- Definition of (polynomial time) reduction
- SAT ≤_p Independent Set example
- SAT ≤_p Knapsack example

- Definition of NP-completeness
- 2x approximation to Euclidean TSP
Abstract

We prove NP-hardness results for five of Nintendo’s largest video game franchises: Mario, Donkey Kong, Legend of Zelda, Metroid, and Pokémon. Our results apply to Super Mario Bros. 1, 3, Lost Levels, and Super Mario World; Donkey Kong Country 1–3; all Legend of Zelda games except Zelda II: The Adventure of Link; all Metroid games; and all Pokémon role-playing games. For Mario and Donkey Kong, we show NP-completeness. In addition, we observe that several games in the Zelda series are PSPACE-complete.
Final Exam Mechanics

Closed book, 1 pg notes (8.5x11, 2 sides, handwritten)

(no bluebook needed; scratch paper may be handy; calculators probably probably unnecessary)
Some Typical Exam Questions

Give $O(\cdot)$ bound on $17n^*(n-3+\log n)$
Give $O(\cdot)$ bound on some code $\{\text{for } i=1 \text{ to } n \text{ \{for } j \ldots\}\}$
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
Good Luck!