CSE 521: Review

Larry Ruzzo
Complexity, I

Asymptotic Analysis
Best/average/worst cases
Upper/Lower Bounds
Big O, Theta, Omega
Analysis methods
  loops
  recurrence relations
  common data structures, subroutines
Complexity, I

Asymptotic Analysis
Best/average/worst cases
Upper/Lower Bounds
Big O, Theta, Omega
Analysis methods
  loops
  recurrence relations
  common data structures, subroutines
  “progress” arguments and general brute cleverness…
Graph Algorithms

Graphs

- Representation (edge list/adjacency matrix)
- Breadth/depth first search
- Bipartiteness/2-Colorability
- DAGS and topological ordering
Graph Algorithms

Graphs

- Representation (edge list/adjacency matrix)
- Breadth/depth first search
- Connected components
- Shortest paths/bipartit ness/2-Colorability
- DAGS and topological ordering
- DFS/articulation points/biconnected components
Graph Algorithms

Graphs
  - Breadth/depth first search
  - Connected components
  - Shortest paths/bipartitiveness/2-Colorability
  - DAGS and topological ordering
  - DFS/articulation points/biconnected components
  - Strongly connected components
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
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
Powerful Subproblems
Flow, Matching, Linear Programming
Design Paradigms

Greedy
   emphasis on correctness arguments, e.g. exchange

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

Dynamic Programming
   recursive solution, redundant subproblems, few
   do all in careful order and tabulate; OPT function
   (usually far superior to “memoization”)

Powerful Subproblems
   Flow, Matching, Linear Programming
Design Paradigms

Greedy
emphasis on correctness arguments, e.g. exchange

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

Dynamic Programming
recursive solution, redundant subproblems, few
do all in careful order and tabulate; OPT function
(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
  Counting Inversions
  Closest pair of points
  Integer multiplication (Karatsuba)
  Matrix multiplication (Strassen)
  Powering
Examples

Divide & Conquer

- Merge sort
- Closest pair of points
- Integer multiplication (Karatsuba)
- Matrix multiplication (Strassen)
- Powering
- FFT
Midterm Friday

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
Examples

Dynamic programming

  Fibonacci
  Making change/Stamps
  Weighted Interval Scheduling
  RNA
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

Dynamic programming
  Fibonacci
  Making change/Stamps, Knapsack
  Weighted Interval Scheduling
  RNA
  String Alignment
Examples

Dynamic programming
  Fibonacci
  Making change/Stamps, Knapsack
  Weighted Interval Scheduling
  RNA
  String Alignment
  (code generation)
Examples & Concepts

Flow and matching

Residual graph, augmenting paths, max-flow/min-cut, Ford-Fulkerson and Edmonds-Karp algorithms, (preflow-push), integrality, reductions to flow, e.g. bipartite matching
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 VertexCover example (how, why correct, why ≤_p, implications)
Definition of NP-completeness
2x approximation to Euclidean TSP
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 ⊆ NP ⊆ Exp (and worse)

Definition(s) of (polynomial time) reduction

- SAT ≤_p e.g., IndpSet, Knap, Ham, 3color: how, correctness, ≤_p, implications

Definition of NP-completeness

NP-completeness proofs

2x, 1.5x approximations to Euclidean TSP
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, clique, independent set, TSP, Hamilton cycle, coloring, max cut, knapsack
- P ⊆ NP ⊆ Exp (and worse)
- Definition(s) of (polynomial time) reduction
- SAT ≤p IndpSet, Knap examples (how, why correct, why ≤p, implications)
- Definition of NP-completeness
- NP-completeness proofs
- Asymmetry; SAT vs UNSAT, (polynomial hierarchy, PSPACE)
- 2x, 1.5x approximations to Euclidean TSP
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 unnecessary)

Comprehensive: All topics covered

assigned reading
slides
homework & solutions
Final Exam Mechanics

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

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

Comprehensive, w/ post-midterm bias

assigned reading

slides

homework & solutions
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
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
Some Typical Exam Questions

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

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

Understand parts of correctness proof for an algorithm
Some Typical Exam Questions

Give $O(\cdot)$ bound on $17n^*(n-3+\log n)$, or on code \{\texttt{for } i=1 \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

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 → 521 Final