Complexity, 1

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
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
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
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
Some Typical Exam Questions

Give $O(\cdot)$ bound on $17n^*(n-3+\log n)$
Give $O(\cdot)$ bound on some code \{for $i=1$ to $n$ \{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
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