CSE 521: Algorithms Design, 2006 Winter

Instructor: Venkatesan Guruswami

Lecture 1 (Jan 3)

- Stable Matching Problem
  - Gale-Shapley algorithm

Lecture 2 (Jan 5)

- Some classical problems
  - Interval Scheduling
  - Weighted Interval Scheduling
  - Bipartite Matching
  - Independent Set (generalize the above problems)
  - Competitive Facility Location
- Greedy Algorithms
  - Interval Scheduling
    - earliest starting time first (works arbitrarily bad)
    - minimum size first (tight 2-approximation)
    - earliest ending time first (optimal, proof)

Lecture 3 (Jan 10)

- Greedy Algorithms
  - Scheduling to minimize the maximum lateness
    - earliest deadline first (optimal, proof)
    - Q: minimize total lateness ?
  - Optimal caching
    - evict item whose next request is farest (optimal, proof)

Lecture 4 (Jan 12)

- Divide and Conquer
  - Mergesort algorithm
    - break numbers arbitrarily into half
  - Integer multiplication
    - break integers into first half and second half
  - Polynomial multiplication (The Fast Fourier Transform)
    - break the degree of polynomials and the corresponding seeking values into half

Lecture 5 (Jan 17)

- Divide and Conquer
  - Closest Pair of Points
    - break points into half according to their x-coordinate
    - when combine two solutions together, for each point p, consider all points on the other side that are "close" enough to p.
- Dynamic Programming
  - Weighted Interval Scheduling
    - recursion on whether the current last job is scheduled or not
  - Knapsack
    - recursion on the size of knapsack, given the fixed order of items
    - Note: this gives a pseudo-polynomial time algorithm

Lecture 6 (Jan 19)

- Dynamic Programming
  - RNA Secondary Structure
    - recursion on the distance of any two symbols in the string
  - Traveling Salesman Problem (TSP)
    - recursion on the number of remaining vertices to the destination
    - Note: this does not give a polynomial time algorithm, but it's much better than the trivial $O(n!)$ one.
  - Shortest Paths (with negative weights and without negative cycles)
    - recursion on the number of edges used in the shortest path

Lecture 7 (Jan 24)

- Dynamic Programming
  - Negative cycles in a graph
    - negative cycle detection
    - find a negative cycle
    - Q: zero weight cycle?
- Network Flow
  - Introduction

Lecture 8 (Jan 26)

- Network Flow
  - Ford-Fulkerson algorithm
    - idea: residual graph with backward edges, augmenting path
    - feasibility
    - termination
    - correctness

Lecture 9 (Jan 31)

- Network Flow
  - Max-flow Min-cut theorem
  - Implementation of the Ford-Fulkerson algorithm in polynomial time
- Applications of Network Flow
  - Bipartite matching
    * idea: augmenting path
  - Disjoint paths in graphs
    * idea: reduce to maximum flow problem

Lecture 10 (Feb 2)

- Network Flow
  - Min-cost perfect matching
  - Min-cost max flow
  - Min-cost circulation
- Linear Programming
  - Introduction

Lecture 11 (Feb 7)

- Linear Programming
  - Examples
  - History
  - Simplex method

Lecture 12 (Feb 9)

- Linear Programming
  - Duality
  - Strong duality theorem

Lecture 13 (Feb 14)

- Linear Programming
  - Week and Strong Duality Theorem
  - Example: Max-flow Min-cut
  - Complementary slackness condition
- Approximation Algorithms
  - Vertex Cover
    - idea: rounding (approximation ratio: 2)

Lecture 14 (Feb 16)

- Approximation Algorithms
  - Vertex Cover
    - idea: primal-dual (approximation ratio: 2)
    - integrality gap of the linear program
    - open Q: approximation ratio better than 2?
  - Set Cover
    - idea: greedy (approximation ratio: $O(n \log n)$)

Lecture 15 (Feb 21)

- Approximation Algorithms
  - Center Selection Problem
    - idea: greedy (approximation ratio: 2)
    - tightness of the ratio
  - Knapsack
    - idea: pseudo-polynomial time algorithm and rounding (approximation ratio: $1 + \epsilon$, for any $\epsilon > 0$)
    - polynomial time approximation scheme (PTAS)

Lecture 16 (Feb 23)

- Approximation Algorithms
  - Knapsack (cont.)
    - idea: pseudo-polynomial time algorithm and rounding (approximation ratio: $1 + \epsilon$, for any $\epsilon > 0$)
    - polynomial time approximation scheme (PTAS)
- Randomized Algorithms
  - Min-cut
    - idea: contract vertices randomly

Lecture 17 (Feb 28)

- Randomized Algorithms
  - Max Exact 3-SAT
    - idea: toss a fair coin for each variable (achieves the best we can approximate unless $P = NP$)
    - polynomial time in expectation, use waiting time bound
  - Coupon Collection
    - problem: n coupons, get a random one every day, how long to collect all coupons?
    - idea: X_i number of days to get the i-th coupon after getting the $(i-1)$-th coupon
    - $\theta(nlogn)$ in expectation, again use waiting time bound
  - MAXSAT (LP)
    - idea: randomized rounding to LPR
    - $\frac{e}{e-1}$-approximation in expectation

Lecture 18 (Mar 2)

- Hashing
  - problem
    - static: support search only, can assume a fixed set
    - dynamic: search and update, must deal with a dynamic set
  - concepts: dictionary, universe, hash function, collision
  - how to hash?
    - deterministic hashing linear worst-case search
    - totally random hash achieves constant worst-case search, but need large specification (linear in universe size)
  - randomized hashing
    - 2-universal family
    - universal hashing lemma
    - linear total number of collision in expectation

Lecture 19 (Mar 7)

- Hashing
  - static case: worst-case constant search
    - soln 1: use quadratic space
      - repeat till finding a collision-free hash function for the given set
      - each trial successful with 1/2 probability (Markov inequality)
      - surprisingly, this achieves the best we can approximate unless $P = NP$
    - soln 2: linear space, FKS-perfect-hashing (1973)
      - 2-level hashing
      - for each cell with collisions, build a quadratic space hash table
      - repeat till finding a linear space hash scheme (each trial successful with 1/2 probability)

- dynamic hashing
  - dynamic perfect hashing (1994): constant worst-case search, constant update w.h.p.
  - cuckoo hashing (2001): conceptually simpler, much smaller constant factor (2 vs. 35)
    - use a pair of hash functions
    - constant worst-case search and deletion, constant insertion in expectation
    - require $O(logn)$-wise independent family (in practice, work well with weaker assumptions)

Lecture 20 (Mar 9)

- cuckoo hashing
- summary