6.856 — Randomized Algorithms

David Karger

Apr. 22, 2021 — Problem Set 10, Due 4/30

Problem 1. (Based on MR9.9) Consider any linear programming problem. Prove that if the constraints are given positive integer weights, and r constraints are sampled at random with probability proportional to the weights, then the expected weight of constraints that violate the optimum of the sample is at most a $d/(r-d)$ fraction of the total weight. **Hint:** For every constraint of weight w_h, replace it by w_h "virtual copies" of h, and consider sampling uniformly.

Problem 2. You are given a set of points in the plane and wish to find the two points that are closest to each other. Consider the following incremental algorithm. Maintain the minimum distance δ for a pair amount the points inserted so far. Without loss of generality, assume that the coordinates are all positive.

1. Multiply all coordinates by $2/\delta$ so that the minimum distance is exactly 2.
2. Place every point in a hash table, with the key being the integer parts of the x and y coordinates of the point.
3. Each time a new point is inserted, use the hash table to check whether δ has changed.
4. (Rebuild.) If δ has changed, rescale all the coordinates (keys) and rebuild the hash table from scratch.

Analyze the running time of this algorithm:

(a) Argue that no two points share a key in the hash table.

(b) Argue that when a new point is inserted, you can determine whether δ has changed in constant time.

(c) Use backward analysis to bound the expected time spend rebuilding the hash table, and thus the overall runtime of the algorithm.

Problem 3. Consider the problem of finding the smallest (minimum diameter) circle containing some set H of n points in the plane. We will assume that the points are in "general position"—no 3 points are colinear, and no 4 points are on the boundary of a common circle. This assumption can be achieved by small perturbations in the input. For any set of points S in the plane, let $O(S)$ denote the smallest circle containing S.

(a) Show that for any 3 non-colinear points, there is a unique circle having all 3 of those points on the circle boundary and that this circle (center and radius) can be computed in constant time from the points.

(b) Show that $O(H)$ contains either 2 or 3 of the input points on its boundary. We will call these points the "basis" of the circle (hint, hint) and refer to them as $B(H)$. Deduce a simple $O(n^4)$-time algorithm for solving the problem.

(c) Show that if a circle C excludes a point of H, then C cannot be the smallest circle containing $B(H)$.

(d) Show that if p is *not* contained in $O(S)$ for some S then p is on the boundary of $O(S \cup \{p\})$.

(e) Consider a set R of r points chosen at random from H. Bound the expected number of points of H outside $O(R)$.

(f) Generalize the previous part to where you have an "active" subset $S \subseteq H$ and compute the number of points outside $O(R \cup S)$.

(g) Give an $\tilde{O}(n)$ time algorithm for finding $O(H)$.

Problem 4. MR 5.12. Show how the method of conditional expectations can be used to deterministically build a 2-dimensional binary space partition of size $O(n \log n)$.

Problem 5. You are given a directed graph and wish to find a vertex partition (A, B) (a cut) maximizing the number of edges with tail in A and head in B.

(a) Give a simple randomized algorithm with approximation ratio 1/4

(b) Sketch how this algorithm can be derandomized

(c) Design an integer linear program for this problem

(d) Show how relaxation and randomized rounding of your ILP yields a 1/2-approximation algorithm.

Problem 6. Exercises with the Goemans-Williamson MAX-CUT algorithm (GW).

(a) Show that GW can also be used to approximate the s-t MAX-CUT problem, where two specified vertices s and t must be separated, to within .878.

(b) Prove that if a graph is bipartite, then GW will find the optimum solution.

(c) Generalizing the previous part, prove that for any $\varepsilon > 0$, there exists a $\delta > 0$, such that for any graph that has a max-cut of value exceeding $(1 - \varepsilon)m$, the Goemans-Williamson algorithm will find a cut of value at least $(1 - \delta)m$. How small a δ can you get in terms of a given ε? **Hint:** consider the value of $\arccos(x)$ near $x = -1$.