6.856 — Randomized Algorithms

David Karger

Apr. 7, 2021 — Problem Set 8, Due 4/14

Problem 1. Consider an arbitrary set S of vectors in d-dimensional space. Suppose that you sample each of those vectors independently with probability p. Prove that there are at most d/p vectors of S in expectation that are not spanned by your sample, regardless of the size of S.

Problem 2. Prove that the sampling MST algorithm runs in $O(m+n)$ time with (very) high probability. **Hint:** Each recursive step has a first subproblem generated by sampling, and a second subproblem generated by sensitivity analysis. First show the total size of second subproblems is small with high probability by examining our sampling analysis (**Hint:** what happens if you just *concatenate* the decision process of all the sampling analyses?). Then show the total size of first subproblems is small by considering how long any given edge will persist through many sampling steps (similar to the Quicksort analysis).

Problem 3—This problem should be done without collaboration. Improving the Recursive Contraction Algorithm. In class we designed a recursive contraction algorithm that branched on two subproblems and contracted each to $n/\sqrt{2}$ vertices. Is this the ideal balance?

(a) Suppose that instead of recursing on two size-$n/\sqrt{2}$ subproblems, we recursed on three. What runtime would result (including repetitions needed for high probability of success)?

(b) Suppose that instead of contracting two subproblems to $n/\sqrt{2}$ vertices, we contracted to $n/2$. What runtime would result (including repetitions needed for high probability of success)?

Problem 4. An *r-way cut* is a partition of the vertices of a graph G into r subsets; its *value* is the number (or total weight) of edges with endpoints in different subsets. The *minimum r-way cut* problem is to find the r-way cut of minimum value. If r can be specified as a (nonconstant) part of the input, the problem is NP-hard. However, it can be solved in polynomial time for any constant r. We will focus on unweighted graphs, though the weighted case is similar. Let c (now) denote the number of edges in the minimum r-way cut

(a) Prove that if G has n vertices then

$$c \leq \left[1 - (1 - \frac{r-1}{n})(1 - \frac{r-1}{n-1})\right] m$$

Hint: consider selecting a random r-way cut by taking $r-1$ vertices at random as singleton components. What is its expected value?

(b) Show that for constant r, any given r-way min-cut survives contraction to r vertices with probability $\Omega(n^{-2(r-1)})$

(c) Conclude that there are only polynomially many minimum r-way cuts and all can be found in polynomial time with high probability

(d) Using ideas from the recursive contraction algorithm, give the best time bound you can for finding all r-way min-cuts.

Problem 5. A *flow* in an undirected graph is a set of edge-disjoint paths from a *source* vertex s to a *sink* vertex t. The *value* of the flow is the number of edge disjoint paths. The *s-t maximum flow problem* aims to a flow of maximum value. This quantity turns out to be equal to the *s-t minimum cut value:* the minimum number of edges that must be removed from the graph in order to disconnect vertex s from vertex t. There is an *augmenting path algorithm* that, given an s-t flow of value v, finds an s-t flow of value $v+1$ in $O(m)$ time on an m-edge graph, or else reports that v is the maximum flow.

Consider any undirected graph with m edges, s-t maximum flow v, and minimum cut c:

(a) Prove for any constant ϵ, an s-t cut of value at most $(1+\epsilon)v$ can be found in $\tilde{O}(mv/c^2)$ time.

(b) Prove that for any constant ϵ, a flow of value $(1-\epsilon)v$ can be found in $\tilde{O}(mv/c)$ time.

(c) Sketch an algorithm that finds the maximum $s-t$ flow in $\tilde{O}(mv/\sqrt{c})$ time, and give a informal argument as to its correctness.

(d) Use the algorithm of part (c) to improve the running times of the algorithms in parts (a) and (b). The improvement can be in terms of ϵ.