6.856 — Randomized Algorithms

David Karger

Mar. 17, 2021 — Problem Set 5, Due 3/26

Problem 1. Consider the problem of *image matching*. You are given an $n \times n$-bit matrix as "text" and an $m \times m$-bit "pattern" you want to find in the text. Devise an efficient (expected time $O(n^2)$) algorithm for the problem. **Hint:** The key is rapidly updating a fingerprint as you shift the "window" over the matrix. You may first want to consider the case of an $n \times m$-bit "text." Can you transform this into a standard string-matching problem?

Problem 2. Bloom filters can be used to estimate the difference between two sets. Suppose that you have sets X and Y, each with m elements, and with r elements in common. Create an n-bit Bloom filter for each, using the same k hash functions.

(a) Determine the expected number of bits where the two Bloom filters differ, as a function of m, n, k, and r.

(b) Explain how this could be used as a technique for estimating r. **Hint:** For this part, assume that the number of differing bits is tightly concentrated around its expectation.

(c) **Optional:** Give a deviation bound showing that your estimate is accurate to within some bound with high probability.

Problem 3—This problem should be done without collaboration. In class we modeled Bloom filters by treating each hash function as if it were a random function, and argued that with the right setting of parameters, the probability of a false positive for an n-bit bloom filter on m items was roughly $(.6185)^{n/m}$. This problem will consider a more theoretically grounded approach. Divide the n bits into r "blocks" of n/r bits. Associate a (random) 2-universal hash function with each block. For each item, hash it to one bit in each block using the block's hash function, and set those bits to 1. Now consider an item x *not* in the inserted set, and suppose we hash it into the blocks in the same way. We wish to bound the probability we get a false positive in this data structure (i.e. all bits 1).

(a) Bound the probability that in a particular block, x hashes to a set bit, and from that bound derive the probability that x is a false positive.

(b) Determine a best choice of r to minimize the above probability as a function of m and n.

Problem 4. Two rooted trees T_1 and T_2 are said to be isomorphic if there exists a one to one mapping f from the nodes of T_1 to those of T_2 satisfying the following condition: v is a child of w in T_1 if and only if $f(v)$ is a child of $f(w)$ in T_2. Observe that no ordering is assumed on the children of any vertex. Devise an efficient randomized algorithm for testing the isomorphism of rooted trees and analyze its performance. **Hint:** Associate a polynomial P_v with each vertex v in a tree T. The polynomials are defined recursively, the base case being that the leaf vertices all have $P = x_0$. An internal vertex v of height h with children $v_1, \ldots, v_k$ has its polynomial defined to be

$$(x_h - P_{v_1})(x_h - P_{v_2}) \cdots (x_h - P_{v_k}).$$

Note that there is exactly one indeterminate for each level in the tree.

Problem 5. In this problem we're going to generalize the "network coding" technique from bipartite graphs to general graphs. The most challenging part of this problem is likely to be understanding the question.

Consider any directed acyclic graph with n vertices, a source vertex s with edges directed from it to k distinct "entry" vertices $s_1, \ldots, s_k$ and a sink vertex t with edges directed to it from k distinct "exit" vertices $t_1, \ldots, t_k$. We will think of each vertex as having the capacity to transmit a single message (represented as an integer over some field Z_p). We know that if there is a flow of k *vertex disjoint* paths from s to t then it can carry a set of k distinct "messages" from s to t such that at most one message traverses each vertex.

Rather than computing that flow, we will use the network coding technique. The vertex s will send a distinct one of its k messages to each of the k entry vertices (alternatively, it can send each vertex a distinct random linear combination of its messages). Every other vertex will receive a message on each of its incoming edges, compute a random linear combination of those messages over Z_p, and transmit the result on all its outgoing edges. We can think of this as placing a random coefficient $w(e)$ on each edge and multiplying the message on that edge by the edge's coefficient, then summing the messages a vertex receives. We will show that enough information is received at the exit vertices (and thus by t) to reconstruct all the original messages.

Note that the value received at any t_i is a linear combination of the values sent from the s_i. Thus, we can represent the mapping from inputs (at s_i) to outputs (at t_i) as a $k \times k$ *transfer matrix* M.

Note that the k vertex disjoint paths of any s-t flow must connect each s_i to a distinct t_j and vice versa. We will refer to any such path set (using disjoint paths or not) as a *path system*. The *sign* of a path system is the sign of the permutation of entry indices to exit indices. Define the *weight* of a path to be the product of the weights on its edges, and the weight of a path system to be the product of the weights of its paths.

(a) Prove that M_{ij} is equal to the sum of the weights of all paths from s_i to t_j.

(b) Considering the Leibniz formula for the determinant as we did in class, show that the $\det(M)$ can be written as a signed sum of weights of path systems.

(c) Show that any value-k vertex-disjoint flow from s to t yields a term in the determinant formula that is not canceled (symbolically) by any other terms.

(d) Show that in this case, with reasonable probability the random assignment of weights proposed above yields a matrix with nonzero determinant over Z_p.

(e) Conclude that any set of original messages on the s_i can be reconstructed from the received values at the t_i.

An important advantage of the network coding technique is that this analysis applies simultaneously to *every* sink that has k disjoint paths to the source; thus, while a flow can be used to send k messages to a single sink, network coding can use the same network capacity to simultaneously *broadcast* the k messages to every sufficiently connected recipient. In particular, network coding can be used to broadcast to all nodes of a k-connected graph.