6.856 — Randomized Algorithms

David Karger

Handout #23, Nov. 28th, 2002 — Homework 13 (Last One), Due 12/6

Answers to some of these problems are in textbooks. Don't use them. There is **no collaboration** permitted on this problem set.

1. We are going to analyze an algorithm for generating a random spanning tree of an undirected, d-regular connected graph G. The algorithm is as follows:

   - Choose a starting vertex uniformly at random
   - Follow the standard random walk over vertices
   - The first time a vertex is entered by the random walk, add the edge used to enter it to the spanning tree (the starting vertex is "entered" at time 0 so receives no entering edge)
   - Stop when all vertices have been entered at least once

   To analyze the algorithm, we need to examine a Markov chain that, on the surface, seems unrelated. The Markov Chain is on *rooted spanning trees of* G, in other words, spanning trees with a specific vertex selected as the root. Once a root r is selected, each vertex $v \neq r$ in the tree has a unique *parent*: the first vertex on the (unique) path from v to r. Consider the following random transition rule from tree T with root r:

   - Choose a random neighbor v of r to be the new root
   - Delete the parent edge of v from T
   - Add the edge (r, v) to the T (this is now r's parent edge)

   Note that as the rooted-tree markov chain evolves, the root vertex is tracing out a standard random walk on G

   (a) Prove that (once we add self loops) this Markov Chain has a unique stationary distribution

   (b) Prove that the stationary distribution is uniform over rooted spanning trees. Conclude that if we ignore the root, then the *tree* of the stationary distribution is a (uniformly) random spanning tree. **Hint:** prove that exactly $d+1$ distinct rooted trees can transition to any particular rooted tree (T, r), and apply previous homework.

(c) Now we connect this Markov chain to the tree generation algorithm described initially. Suppose that the markov chain is initiated in its stationary distribution infinitely far in the past, and consider its state at time 0. Argue that to know the rooted tree at time 0, it is sufficient to know, for each vertex v, the identity of the last vertex to replace v as root in the Markov chain before time 0. In other words, that the state at time 0 is defined by the sequence of roots $r_{-\infty}, \ldots, r_{-1}, r_0$ encountered before time 0, and that we only need a suffix of this sequence that includes every vertex at least once.

(d) Argue that setting r_0 to be a (uniformly) random vertex of G and then iteratively setting $r_{-(k+1)}$ to be a random neighbor of r_{-k} (ie, running the standard Markov chain on G) gives the correct distribution on sequences of roots. **Hint:** this follows from the fact that the standard walk on G is a reversible Markov chain.

(e) Conclude that the spanning tree generation algorithm proposed above does yield a uniformly random spanning tree.

(f) Give a one-line proof that the expected running time of the algorithm is polynomial.

2. The previous algorithm for generating a random spanning tree generalizes to arbitrary graphs (with nonuniform degrees and even parallel edges). Take this as given (you are welcome to prove it). We will use this generator to estimate the number of spanning trees in an undirected graph (there is a deterministic exact algorithm, but this randomized approximation is faster).

   (a) Consider a particular edge e in the graph. Argue that there is an FPRAS for *either* (i) the fraction of spanning trees containing e *or* (ii) the fraction of spanning trees *not* containing e.

   (b) Given the previous part, show how to reduce the problem of approximately counting spanning trees in G to one of approximately counting spanning trees in a smaller graph **Hint:** consider contraction and/or deletion of an edge.

   (c) Give (and prove) an FPRAS for the number of spanning trees in an undirected graph.

   (d) Conversely, prove that given *any* algorithm for *counting* the number of spanning trees in an arbitrary graph, you can *sample* a random spanning tree using at most $2m$ calls to the counting algorithm. Thus, the connection from sampling to counting is bidirectional. **Hint:** use the same self-reduction as above to decide about one edge at a time whether or not it is in the spanning tree.

3. Summary.

   (a) What was the most useful/interesting topic covered in class? Why?

   (b) What was the least useful/interesting topic covered in class? Why?

   (c) Explain what (if anything) you wish had been covered that was not (we'll do computational geometry next week).