6.856 — Randomized Algorithms

David Karger

Handout #13, October 16, 2000 — Midterm, Due 10/23

M. R. refers to this text:
Motwani, Rajeez, and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge: Cambridge University Press, 1995.

NO COLLABORATION is allowed on this problem set.

1. The Aethernet corporation has developed an exciting new protocol to let a collection of m machines communicate on a shared wire. Time is divided into fixed slots. In a slot, any machine that wants to send information to another can broadcast it onto the wire. If exactly one request is broadcast in the slot, that broadcast is successfully completed. If more than one broadcast is attempted, none are successful.

   (a) Assuming that the number of machines n that wish to broadcast is known, devise a randomized protocol that will, with no communication overhead, let some broadcast succeed with constant probability in each time slot, without favoring any particular machine.

   (b) Assume that there is a fixed set of n machines, each of which repeatedly tries (in every slot) to send (an infinite sequence of) packets using the protocol from the previous part. What is the expected time until every machine has sent its first packet?

   (c) For this subproblem and the remainder, assume that in each time slot, *some* set of n machines decides to broadcast a packet. However, in each time slot, the set of machines might be different.

   Prove that over a sufficient length of time (number of slots), with high probability every machine broadcasts at least $\Omega(1/n)$ of the times that it wants to (thus achieving a fair share of the bandwidth on the wire). In other words, show that if there are s slots in which M is one of the n machines trying to broadcast, then with high probability, M broadcasts in $\Omega(s/n)$ slots.

   (d) For a competitive advantage, Aethernet believes they can cut costs by using fewer random bits to make broadcast decisions. Assume that each of the m machines has a distinct integer serial number in $1, \ldots, m$. Assume that Aethernet corp can broadcast to the entire group of machines a single $O(\log m)$-bit random number per slot, but that no other randomness is used. And assume again that n is known. Devise a scheme that achieves the same result as part (a) under this model. That is, in each slot achieves $\Pr[\text{some broadcast succeeds}] = \Omega(1)$. There is no need to achieve fairness, but it should be clear that the scheme is mostly fair.

   (e) (optional) Show that it suffices for the system to broadcast a single random $O(\log m)$ bit value at the start of computation and still achieve the goals of the previous part.

2. (Based on MR 5.3). An *independent set* in a graph is a set of vertices with no edges connecting them. Let G be a graph with $nd/2$ edges ($d > 1$), and consider the following probabilistic experiment for finding an independent set in G: delete each vertex of G (and all its incident edges) independently with probability $1 - 1/d$.

   (a) Compute the expected number of vertices and edges that remain after the deletion process. Now imagine deleting one endpoints of each remaining edge.

   (b) From this, infer that there is an independent set with at least $n/2d$ vertices in any graph with on n vertices with $nd/2$ edges.

   (c) Apply the method of conditional expectations to the above experiment to define a deterministic algorithm for finding an independent set of size $n/2d$.

3. A cut in a graph is a partition of the vertices into two parts A and B. The size of the cut is the number of edges with one end in A and the other in B. A minimum cut is a cut of minimum size. Give a Monte Carlo algorithm that finds the *second smallest* cut in a graph with high probability (if a graph has two distinct minimum cuts, the second smallest cut is a minimum cut).

4. A *bi-bucket hash (bash)* function maps each item in a universe U to *two* possible values. A bash-function can be used in a dictionary: to lookup an item, check in both possible locations. An advantage of the bash-function is that you can choose which of two locations to place the item in, which gives some improved load-balancing opportunities. In particular, consider an insertion rule that places each inserted item into the less-occupied of its two locations.

   (a) Suppose that n items are inserted into $n^{1.5}$ locations using a random bash function. Give a reasonably good upper bound on the expected number of collisions.

   (b) A bash is perfect for a set S if there is some assignment of the items to their candidate locations such that no two elements of S share a location. Under the obvious generalization of the definition, prove that there exists a perfect bash family mapping n items to $O(n^{1.5})$ locations of size polynomial in n and log(|universe|).

   (c) Generalize your results to achieve smaller space usage with perfect tri-bucket (trash), quad-bucket (quash) and higher bucket hash families.

5. Bash functions also have interesting applications for load balancing. In class we discussed that when n balls are thrown randomly into n bins, the maximum number of balls in a bin is $\Theta(\log n)$ with high probability. Suppose instead that we insert them using a (truly random) bash function. We will prove that with high probability the maximum load drops to $\Theta(\log \log n)$, an exponential improvement. When the i^{th} item is inserted in a random bin, let h_i denote the (random) number of items already in that bin. Let n_h denote the (random) number of items with $h_i \geq h$.

(a) Argue that with high probability $n_1 \leq n(1 - 1/e)$.

(b) Suppose that $n_h \leq \epsilon n$. Argue that $E[n_{h+1}] \leq \epsilon^2 n$ conditioned on this fact.

(c) Prove that there is some $h = O(\log \log n)$ such that $n_h = O(\log n)$ with high probability

(d) Using the h from the previous section, prove that with high probability there are no balls with $h_i \geq h + O(1)$.