6.856 — Randomized Algorithms

David Karger

Handout #16, October 27, 2002 — Homework 7 Solutions

Problem 1

(a) We let each machine broadcast independently with probability $1/n$. Then the probability that exactly one machine tries to broadcast in a given timeslot is equal to

$$n \cdot \frac{1}{n} \cdot \left(1 - \frac{1}{n}\right)^{n-1} = \Theta\left(\frac{1}{e}\right),$$

i.e. larger than some constant independent of n.

(b) From (a) we know that the expected time until some machine is successful is $\Theta(e)$. Since the problem is symmetric in all n machines, the successful machine is equally likely to be any of the machines. This is also true for the next successful machine, and so on. So the question is how long it takes for all machines to be successful, if every $\Theta(e)$ steps, a random machine is successful. This is just the coupon collectors problem: we expect $\Theta(n \ln n)$ successful broadcasts before every machine actually sent their first packet. So the expected time for all machines to be successful is $\Theta(ne \ln n)$.

(c) The probability that a machine is successful in a particular slot is equal to the probability that a) the machine tries to broadcast, and b) nobody else tries to broadcast. Since the machines decide on broadcasts independently, there probabilities are also independent, and therefore

$$\Pr[\text{machine successful}] = \frac{1}{n} \cdot \left(1 - \frac{1}{n}\right)^{n-1} = \Theta\left(\frac{1}{ne}\right)$$

So over s time slots, the expected number of successful broadcasts is $\Theta(s/ne) = c \cdot s/ne$ (for some $c > 0$). Using Chernoff, we can estimate the probability that the machine is successful less than half that number of times:

$$\Pr[\text{sucessful} \leq cs/(2ne) \text{ times out of } s] \leq \exp(-cs/(8ne)) = \alpha^{s/n}$$

for some $\alpha < 1$. This shows high probability in terms of increasing s.

(d) We use 2-point-sampling to determine when a machine gets to go. For simplicity, assume that m is prime (all of the following is still true if we replace m by some prime in the

interval $[m, 2m]$). Assume that Aethernet broadcasts $2\lceil \log m \rceil$ bits that encode two random numbers $a, b \in \mathbb{Z}_m$. Then every machine takes its serial number s, and computes $as + b \mod m$. If that quantity falls in the interval $[0, m/2n]$, the machine broadcasts, otherwise it stays silent.

Let's analyze the probability that some particular machine broadcasts successfully. This is equal to

$$\begin{aligned} &\Pr[\text{our machine tries to broadcast}] \\ \cdot\ &\Pr[\text{no other machine tries to broadcast} \mid \text{our machine tries to broadcast}]. \end{aligned}$$

The first probability is obviously $1/2n$, since $as + b \mod m$ is equally likely to be any number between 0 and $m-1$. But since 2-point-sampling generates pairwise independent samples for two different machines, the probability that any other given machine tries to broadcast *given that* our machine tries to broadcast, is also $1/2n$. Therefore the expected number of other machines that tries to broadcast is $\leq (n-1)/2n \leq 1/2$. By Markov's inequality, that implies that the probability that at least one machine tries to broadcast (given that our machine tries to broadcast) is at most $1/2$.

This show that our machine successfully broadcast with probability at least $1/2n \cdot 1/2 = 1/4n$. Since the events of a successful broadcast are disjoint for different machines, the probability that some machine successfully broadcasts is $n \cdot 1/4n = 1/4 = \Omega(1)$.

(e) We just saw that if we pick the 2-point-sampling pair (a, b) at random, we obtain a pair that allows for a successful broadcast with probability $1/4$. But suppose we now generate those pairs (a, b) using two point sampling. This can be done by picking a prime p slightly larger than m^2, picking two seeds $u, v \in \mathbb{Z}_p$, and generating pairs as $(a_i, b_i) = (\lfloor \alpha_i/m \rfloor, \alpha_i \mod m)$, where $\alpha_i = (ui + v \mod p)$. We set $i = 0$ at the beginning of the protocol and then increase its value by one at every time slot (i.e. every machine keeps track of the current i).

So the pairs (a_i, b_i) can be generated by each machine just given the initial broadcast of u and v, which uses about $4 \log m$ bits. Since the (a_i, b_i) are chosen pairwise independently, we can use Chebyshev to determine the number of times a successful broadcast occurs over time.

Consider one particular machine, and assume it wants to transmit during s time slots. In part (d) we determined, that it succeeds with probability $1/4n$ in each time slot, so the expected number of transmissions is $s/4n$. The variance of that distribution is $\sqrt{n(1/4n)(1-1/4n)} < 1/2$. Using Chebyshev, we see that the probability of sending out at least $s/8n$ broadcasts can be bounded by

$$\Pr[|\#broadcasts - s/4n| \geq s/8n] = \frac{1}{(s/4n)^2} = \frac{16n^2}{s^2}.$$

This is a high probability is terms of growing s.

Problem 2

(a) Every vertex gets deleted with probability $1/d$, so we expect n/d vertices to remain. An edge does *not* get deleted if both endpoints do not get deleted, which happens with probability $1/d^2$. Obviously, the events of edges getting deleted are not independent, but by linearity of expectation, be expect $(1/d^2) \cdot (nd/2) = n/2d$ edges to remain.

(b) After applying this sampling, we create an independent set as follows: for each edge in the resulting graph, delete one of the endpoints. After doing this for each edge, none of the remaining vertices are connected by any edges, i.e. form an independent set. If G' is the graph we obtain after sampling, we the expect the size of the independent set to be

$$\begin{aligned} E[\text{size of independent set}] &= E[\#\text{ vertices in } G' - \#\text{ edges in } G'] \\ &= E[\#\text{ vertices in } G'] - E[\#\text{ edges in } G'] \\ &= n/d - n/2d = n/2d. \end{aligned}$$

Since there will be at least one outcome with a value equal to (or greater than) the expectation, there must be an independent set of size $\geq n/2d$.

(c) Suppose the vertices are labeled $1, 2, \ldots, n$. Then we go through the vertices in this order, and compute the expected sizes of the independent set depending on whether we pick vertex i or not, given the choices we already made for vertices $1, 2, \ldots, i-1$. If we always choose the higher of the two values, then the method of conditional expectations guarantees that the independent set we finally end up with has size at least $n/2d$ – the expected value.

As seen above, to compute the expected size of the independent set, we have to compute the expected number of nodes and edges in our final sample – the size of the independent set is equal to their difference.

The expected number of nodes is easy to determine: suppose we picked k nodes among the set $\{1, 2, \ldots, i-1\}$. Then the expected number of nodes in the final sample is $k + (n-i)/d$ or $k + (n-i)/d + 1$, depending of whether we pick element i or not.

Slightly more complicated is determining the expected number of edges. Let $K \subseteq \{1, 2, \ldots, i-1\}$ the set of vertices that we already picked. Set $k_1 := |E \cap K \times K|$ to be the number of edges between already picked vertices – these will definitely be in the final sample. Let $k_2 := |E \cap K \times \{i+1, i+2, \ldots, n\}|$ be the edges between already picked vertices, and vertices which we have yet to decide on, each gets picked with probability $1/d$. Finally, $k_3 := |E \cap \{i+1, i+2, \ldots, n\} \times \{i+1, i+2, \ldots, n\}|$ are the edges that are picked with probability $1/d^2$ in the future.

Let k_1' and k_2' be the values of k_1 and k_2 if we replace K by $K \cup \{i\}$. Then the expected number of edges in the final sample is $k_1' + k_2'/d + k_3/d^2$, if we pick i, and $k_1 + k_2/d + k_3/d^2$ if we do not pick i. This shows that we easily compute the desired expectations.

Problem 3 We are going to modify the algorithm **CA** (Contraction Algorithm) slightly, so that it returns the second smallest cut instead of the smallest cut. First we observe that if the size of the second smallest cut is k, then the degree of all but one node must be at least k. Therefore, the graph contains at least $k(n-1)/2$ edges. Thus, when we contract an edge, the probability that we pick an edge from the second smallest cut is at most

$$\frac{k}{\frac{k(n-1)}{2}} = \frac{2}{n-1}.$$

If we perform a series of contractions down to 3 vertices, the probability that the second smallest cut survives is at least

$$\prod_{j=4}^{n}\left(1-\frac{2}{j-1}\right) = \prod_{j=4}^{n}\frac{j-3}{j-1} = \frac{2}{(n-1)(n-2)}.$$

Assuming the second smallest cut survives contraction to 3 vertices, we can find it by checking the degrees of the resulting 3 vertices. Thus, a single run of the contraction algorithm finds the second smallest cut with probability $\Omega(1/n^2)$. It follows that $O(n^2 \log n)$ trials will find the second smallest cut with high probability.

Of course, we need to know that it is the second smallest cut when we find it. But that is easy if we find the minimum cut first.

Problem 4

(a) Consider the $(k+1)^{st}$ item inserted. Since only k buckets (at worst) are occupied, the probability that *both* candidate locations are occupied is only $(k/n^{1.5})^2$. Thus, the expected number of times an item is actually inserted into an already-occupied bucket is at most

$$\begin{aligned}\sum_{k=0}^{n-1}(k/n^{1.5})^2 &= \frac{(n-1)(n)(2n-1)}{6n^3} \\ &\leq 1/3\end{aligned}$$

Now let's consider pairwise collisions. Item k collides with item $j<k$ only if (i) one of the candidate locations of item k is the location as item j (this has probability at most $2/n^{1.5}$) and (ii) the other candidate location for item k contains at least one element (probability $k/n^{1.5}$). Thus, the probability k collides with j is at most k/n^3. Summing over the k possible values of $j<k$, we find the expected number of collisions for item k is at most k^2/n^3. Summing over all k, we get the same result as above: $O(1)$ expected collisions.

(b) Start with a 2-universal family of hash functions mapping n items to $2n^{1.5}$ locations. Consider any particular set of n items. Consider choosing a random function from the hash family family. The probability that item k collides with item j is $1/2n^{1.5}$ by pairwise independence, implying by the union bound that the probability k collides with *any* item is at most $1/2\sqrt{n}$.

Now suppose that we allocate *two* arrays of size $2n^{1.5}$ and choose a random 2-universal hash function from the family independently for each array. If an item has no collision in *either* array, then it will be placed in an empty bucket by the bash function. We need merely analyze the probability that this happens for every item (this would make the bash function perfect).

The probability that item k has a collision in *both* arrays is at most $(1/2\sqrt{n})^2 = 1/4n$. It follows that the expected number of items colliding with some other item is at most $1/4$. This implies in turn that with probability $3/4$, every item is placed in an empty bucket by the (perfect) bash function. This in turn implies that *some* pair of 2-universal hash functions defines a perfect bash for our set of n items.

Since every set of items gets a perfect bash from this scheme, it follows that the family of pairs of 2-universal functions above is a perfect bash family. Since the 2-universal family has size polynomial in the universe, so does the family of pairs of 2-universal functions.

(c) If we map our n items to k candidate locations in an array of size $n^{1+1/k}$, our collision odds work out as above and we get a constant number of collisions. Similarly, k random 2-universal hash families, each mapping to a set of size $n^{1+1/k}$, has a constant probability of being perfect for any particular set of items, so the set of all such functions provides a perfect family (of polynomial size for any constant k). This gives a tradeoff of k probes for perfect hashing in space $O(n^{1+1/k})$.

Note that while we can achieve perfect hashing to $O(n)$ space, the resulting family does *not* have polynomial size (since a different, subsidiary hash function must be chosen for each sub-hash-table).

Problem 5

(a) n_1 is simply the number of balls that are placed in a nonempty bin, which occurs if and only if both of their candidate locations are nonempty. The probability that this happens to the k^{th} ball is at most $(k/n)^2$ (since at most k bins are nonempty). So the expected number of balls landing in nonempty bins is at most

$$\begin{aligned} \sum (k/n)^2 &= \frac{(n-1)(n)(2n-1)}{6n^2} \\ &\leq n/3 \end{aligned}$$

We'd like to use a Chernoff bound to deduce immediately that $n_1 \leq n/2$ (or any other constant multiple greater than $1/3$ of n) with high probability. A small concern arises regarding dependence: the *probability* that the k^{th} item lands in an occupied bin is dependent on the number of collisions among previous balls. But we are OK because this probability *must be* at most $(k/n)^2$ and because the *event* of the k^{th} item colliding (given the probability) is independent of previous events. More formally, suppose that as we insert random balls, each time a ball suffers a collision (so gets inserted into a nonempty bin) we add an extra artificial ball to one of the empty bins. The ensures

that when ball k is inserted there are exactly $k-1$ nonempty bins, which removes the dependence of the collision probability on past history and lets us apply the chernoff bound.

(b) This subpart had a bug: it conveys the target intuition without actually being true. What we are actually showing will be clear by the end.

The intuitive argument is clear. An item gets height exceeding h only if both bins it maps to have height h or greater. If there are only ϵn items of height h or greater, then there are at most ϵn buckets of height h or greater. This means a random location has height h or greater with probability at most ϵ. Thus the probability that both (random) locations for an item have height h or greater is at most ϵ^2. This means that the expected number of items of height $h+1$ or greater is at most $\epsilon^2 n$.

The technical challenge arises from the conditioning. We want to condition on the event that $n_h \leq \epsilon n$. But this is certainly not independent of the event of any particular item having height exceeding h. We will show how to solve the conditioning problem *when the conditioning event holds with high probability*. Intuitively, conditioning on a high probability event does not change things very much (just as conditioning on a probability 1 event does not change things at all).

So supppose that our event A_h that $n_h \leq \epsilon n$ holds with high probability. We aim to show that $E[n_{h+1}]$ is close to $\epsilon^2 n$. Conditioning on A_h, our first task is to determine the probability of the event X_i that $h_i > h$. The intuitive expectation argument above says that with this conditioning we believe that $\Pr[X_i] \leq \epsilon^2$, which gives what we want. But perhaps the conditioning changes this probability?

To analyze this it is helpful to consider the event B_i that there are less than ϵn height h bins *when item i is inserted*. Note that event A_h (that there are at most ϵn balls of height h at the end) implies B_i. We can then write

$$\begin{aligned}
\Pr[X_i \mid A_h] &= \frac{\Pr[X_i \wedge A_h]}{\Pr[A_h]} \\
&\leq \frac{\Pr[X_i \wedge B_i]}{\Pr[A_h]} \text{since } A_h \text{ implies } B_i \\
&\leq \frac{\Pr[X_i \mid B_i]\Pr[B_i]}{1-1/n^2}
\end{aligned}$$

(since our assumption that A_h is true with high probability means we can take $\Pr[A_h] \geq 1-1/n^2$)

$$\begin{aligned}
&\leq \frac{\epsilon^2}{1-1/n^2} \\
&\leq 1.1\epsilon^2
\end{aligned}$$

Now that we know that the probability of X_i is at most $1.1\epsilon^2$ conditioned on A_h, we can deduce that $E[n_{h+1}] \leq 1.1\epsilon^2 n_h$ conditioned on A_h.

(c) We are going to define values ϵ_h such that, with high probability, $n_h \leq \epsilon_h n$. We already have $\epsilon_1 = 1/2$. Define the ϵ_h will satisfy the recurrence $\epsilon_{h+1} \leq 1.1\epsilon_h^2$ which means that $\epsilon_h = 1/n$ for some $i = O(\log \log n)$.

The argument of the previous section tells us that conditioned on the event A_h of $n_h \leq \epsilon_h n$, the probability that any particular ball has height exceeding h is at most $1.1\epsilon^2$. This suggests that we apply the Chernoff bound, arguing that so long as $\epsilon_h^2 \gg \log n$, the actual value of n_{h+1} is close to its expectation with high probability. Of course, we have to worry about independence. But we can avoid this the same way we did in the first subpart. Although the $\Pr[h_i > h]$ is determined by the previous history, the *event* that $h_i > h$ is independent given the probability. In other words, we can imagine increasing the probability that $h_i > h$ so that it is always exactly $1.1\epsilon^2 n$; this makes it independent of the previous history and gives a distribution that stochastically dominates the true distribution. Since these modified variables are independent we can apply the Chernoff bound as desired.

(d) By the previous part there is an $h = O(\log \log n)$ such that $n_h = O(\log n)$ with high probability. The Chernoff bound is not longer powerful enough to take us lower. But much of the analysis of the previous section still works. Conditioning on $n_h = O(\log n)$, we can conclude that the probability any particular ball height exceeds $h + 1$ is at most $1.1\epsilon^2$ where $\epsilon = O((\log n/)n)$. As above, the number of balls of height exceeding h is stochastically dominated by a sum of n independent random variables, each 1 with probability ϵ^2. The probability that some set of k of these variables is all 1 is at most

$$\begin{aligned}\binom{n}{k}(1.1\epsilon^2)^k &= O(n^k(\frac{\log n}{n})^2 k) \\ &= O((\frac{\log^2 n}{n})^k) \\ &= O(n^{1-k})\end{aligned}$$

Proving that with high probability, at most a constant number of balls have height exceeding h. This of course implies that the maximum height is $h + O(1) = O(\log \log n)$.