Sample Solution Sketch to Problem Set 3

Problem 1. (20 points) 2-Universal hash functions

We used a 2-universal hash function in the streaming algorithm for estimating the number of distinct elements in a stream. In this exercise, we construct one class of 2-universal hash functions.

Let $\mathcal{H}$ be a finite collection of hash functions that map a given universe $X = \{0,1\}^n$ into a range $Y = \{0,1\}^m$. The collection $\mathcal{H}$ is 2-universal if for each pair of distinct keys $x, x' \in X$ and pair of elements $y, y' \in Y$, if h is drawn uniformly at random from $\mathcal{H}$, then

$$\Pr[h(x) = y \text{ and } h(x') = y'] = \frac{1}{2^{2m}}.$$

Suppose we represent each element of X as an $n \times 1$ column $0-1$ vector and each element of Y as an $m \times 1$ column vector in the standard manner. For an arbitrary $m \times n$ matrix $A \in \{0,1\}^{m\times n}$ and vector $b \in \{0,1\}^m$, we define the function $h_{A,b} : X \to Y$ as $h_{A,b}(x) = Ax + b$, where all additions and multiplications are modulo two.

Prove that the collection $\mathcal{H} = \{h_{A,b} : A \in \{0,1\}^{n\times m}, b \in \{0,1\}^m\}$ is 2-universal.

Answer: Fix x, y, $x' \neq x$, and y'. The number of different pairs (A, b) is 2^{mn+m}. We will calculate how many of the pairs satisfy $Ax + b = y \bmod 2$ and $Ax' + b = y' \bmod 2$.

We first consider the number of A that satisfy $A(x-x') = y-y' \bmod 2$. We claim that this number is 2^{nm-m}. Since $x - x' \neq 0$, for every $z \in \{0,1\}^m$, we can find an A such that $A(x - x') = z$ (it is sufficient to use the fact that there exists an i such that $x_i \neq x'_i$). For any $z \in \{0,1\}^m$, let K_z denote the set $\{A : A(x - x') = z\}$. Since each K_z is nonempty, 0 is in K_0, it is easy to argue that every K_z is of the same size. So the number of A that satisfy $A(x - x') = y - y' \bmod 2$. is 2^{nm-m}.

For every A such that $A(x - x') = y - y' \bmod 2$, there is exactly one $b = Ax - y \bmod 2$ such that $Ax + b = y$. Thus, the number of (A, b) such that $Ax + b = y \bmod 2$ and $Ax' + b = y' \bmod 2$. is 2^{nm-m}. Hence, we obtain

$$\Pr[h(x) = y \text{ and } h(x') = y'] = \frac{1}{2^{2m}}.$$

Problem 2. (15 points) Estimating the number of distinct colors in an urn of balls

You are given an urn with a large finite number of colored balls. You are asked to estimate the number of distinct colors in the urn. The only operation you are allowed is to take *samples* from the urn, uniformly at random with replacement.

Give a procedure for yielding an unbiased estimator; that is an estimator, whose expectation equals the number of distinct colors. What is the variance of your estimator?

(*Remark:* I hope you find this to be a fun puzzle. Recall that in the streaming algorithm we studied for the second frequency moment, our unbiased estimator consisted of picking a random element in the stream and then returning the number of occurrences in the remainder of the stream. The expectation of this quantity was proportional to $\sum_i f_i \cdot f_i$ and hence gave the second frequency moment. Come up a similar estimator for the zeroth moment in this exercise.)

Answer: Here is a process that yields an unbiased estimate of the number of colors. Let c be the number of colors, labeled 1, ..., c. Let n_i be the number of balls of color i. Let $m = \sum_i n_i$.

Sample the first ball. The probability that the first ball is of color i is n_i/m. Similar to the estimator for the second frequency moment, let us consider a measure using this first sample. If we could ensure that the expected value of this measure, conditioned on the event that the first sample is of color i, is m/n_i, then we would the overall expectation of this estimate to be $\sum_i (n_i/m) \cdot (m/n_i)$, equal to c; precisely what we want.

Suppose the first ball is of color i. What measure would have an expectation of m/n_i? Well, n_i/m is the fraction of balls that have color i; and its inverse is precisely the expected number of balls we need to sample before we get a ball of color i.

Formally, the process is defined as follows. Let the first sample be of color i. Let next sample that is of color i be the kth sample. Return $k - 1$. As we argued above, the expectation of $k - 1$ is c.

Note that this process does not complete in a bounded amount of time with probability 1. If the number of colors exceeds 1, for any integer r, the number of samples needed by the process exceeds r with positive probability.

Problem 3. (10 + 10 = 20 points) Approximating graph bisection by reduction to trees

In the graph bisection problem, you are given an edge-weighted graph $G = (V, E)$ and are asked to determine a subset S of V of size $\lfloor V/2 \lfloor$ that minimizes the weight the cut $(S, \overline{S})$. In this exercise, we show that one can obtain a good approximation algorithm to this problem by reduction to trees.

(a) Give a polynomial-time algorithm to solve graph bisection optimally over trees.

Answer: There are multiple dynamic programming approaches for this problem. Most DP algorithms on trees arbitrarily root the tree on some vertex and do a top-down processing.

We present an approach that some students suggested. For convenience, we reduce the general tree case to one of a rooted binary tree with weights on edges and nodes. Arbitrary root the original tree at a node. Replace each node with more than two children with a binary tree whose leaves are the children; the edges of the binary tree have infinite weight while the vertices have 0 weight. This ensures that none of the newly added edges are cut in an optimal solution.

Suppose the input tree was a binary tree with root r. Let T_u denote the subtree rooted at u. Let $B(u, k)$ be an optimal partition (X, Y) of T_u such that the part containing u is of weight k; we use the convention that X contains u.

Consider the case where u has two children u_1 and u_2. The case where u has one child is similar. To calculate $B(u, k)$, we consider four possibilities: u and u_1 are on the same part, u_2 on the other; u and u_2 are on the same part, u_1 on the other; u is in one part, while u_1 and u_2 are on the other part; u, u_1, and u_2 are all in the same part.

Consider the first possibility. For any m, let $B(u_1, k - m) = (X_m, Y_m)$ and $B(u_2, |T_{u_2}| - m -$

$1) = (X'_m, Y'_m)$. Then, one candidate for $B(u, k)$ is the partition $(X_m \cup \{u\} \cup Y'_m, Y_m \cup X'_m)$ that has the smallest cut weight, among all choices of r. We obtain a similar formula for the other three cases, and set $B(u, k)$ to be the best among the four choices.

The final solution is given by the better of the solutions $B(r, \lceil n/2 \rceil)$ and $B(r, \lfloor n/2 \rfloor)$.

For a bound on the running time, we note that the number of choices for u is n, and for k is also n. The computation of $B(u, k)$, for a given u and k, takes time proportional the number of choices of m, which is also at most n. So the total running time is $O(n^3)$. A more careful argument may reduce the running time to close to quadratic.

Suppose for any graph edge-weighted $G = (V, E)$, we can compute, in polynomial-time, a probability distribution $\mathcal{D}$ over edge-weighted trees (with new weights) over the same set V of vertices, with the following property for some $\alpha \geq 1$: for any cut $(S, V - S)$ of G, (i) the weight of the cut in any tree in $\mathcal{D}$ is at least the weight of the cut in G; and (ii) the expected weight of the cut in a tree drawn uniformly at random from $\mathcal{D}$ is at most α times the weight of the cut in G.

(b) Show that there exists a randomized polynomial-time algorithm that computes a solution to the graph bisection problem for any graph of expected cost at most α times optimal.

Answer: We draw a tree T according to distribution $\mathcal{D}$ and return the partition given by the optimal bisection B for T. This partition is clearly also a bisection in G. Let the weight of B in T be $w_T(B)$ and in G be $w_G(B)$. Let B^* be an optimal bisection with weight $w_T(B^*)$ in T and $w_G(B^*)$ in G. By the construction of $\mathcal{D}$, we have the following:

$$E[w_G(B)] \leq E[w_T(B)] \leq E[w_T(B^*)] \leq \alpha w_G(B^*).$$

Note that the expectation is over the random spanning tree T drawn from $\mathcal{D}$. This gives the desired α-approximation.

Problem 4. (20 points) Multiset multicover problem

The *multiset multicover problem* is a generalization of set cover in which we have multisets instead of sets and an element may be required to be covered multiple times. Formally, we are given a universe $\mathcal{U}$ of elements, a positive integer r_e for each element $e \in \mathcal{U}$, a collection $\mathcal{C}$ of multisets over $\mathcal{U}$, and a cost $c(S)$ for each multiset $S \in \mathcal{C}$. (A multiset contains a specified number of copies of each element.)

The goal of the multiset multicover problem is to determine a minimum-cost multiset of multisets (thus, you are allowed to pick multiple copies of a multiset) such that each element e is covered at least r_e times. The cost of picking a multiset S, k times, is $k \cdot c(S)$. (You may assume that the number of times that an element e appears in any multiset S is at most r_e.)

Generalize either the greedy algorithm or the randomized rounding algorithm for set cover to achieve an $O(\log(r + n))$ approximation, where n is the number of elements in $\mathcal{U}$ and r is the sum of r_e over all e.

Answer: We present a randomized rounding algorithm, which in fact yields an $O(\log n)$ approximation. We set up the LP similar to that for set cover. Let x_S denote the variable for multiset

S. We first select $\lfloor x_S \rfloor$ copies of multiset S. When we do that for every multiset S, we have satisfied some of the requirements, so we can reduce those requirements. For the remaining problem, $y_S = x_S - \lfloor x_S \rfloor$ is a valid fractional solution.

We will now do the rounding for the remaining problem. We add another copy of S with probability y_S. The expected total cost is the same as the LP optimal. The expected number of times that an element e is covered is at least r_e. But satisfying this coverage is not guaranteed.

If we repeat the above process $O(\log n)$ times one can argue using Chernoff bounds that the probability that element e is not completely covered is at most $1/n^c$ for constant $c > 0$ that can be made large by adjusting the hidden constant in the big-Oh term. This is because the expected amount of coverage in $\Theta(\log n)$ iterations is $r_e \cdot \Theta(\log n)$, and we are bounding the deviation to within a $\Theta(\log n)$ factor of this expectation. To ensure that the argument works, we need to ensure that no multiset covers e more than r_e times (since this maximum bound appears in the Chernoff bounds). We can achieve this by simply setting the coverage of any multiset that contains e more than r_e times to r_e. This does not affect the coverage problem in any way.

Thus, the probability that all elements are covered up to their requirement is at least $1 - 1/n$, by choosing c appropriately.

We now consider the cost. The expected cost of the solution is $O(\log n)$ times optimal LP value. The probability that the cost exceeds $O(\log n)$ times optimal is at most $1/4$, using Markov's inequality (and selecting the hidden constant appropriately).

Now, we have a situation similar to what we had for set cover. We just repeat the above process until we get a solution whose cost is at most $O(\log n)$ times optimal LP value and every element is covered.