6.856 — Randomized Algorithms

Handout #12, March 25, 2011 — Homework 5 Solutions

1.

The expected number of bits where the two Bloom filters differ is n times the probability that a fixed bit differs; call this p_b. If one of the r common elements gets mapped to the bit through one of the k hash functions, this bit will obviously be the same. The probability that this happens is $p_c = 1 - (1 - \frac{1}{n})^{rk}$. Now condition on none of the r common elements setting the bit. The probability that one of the $m - r$ elements unique to the first set makes this bit one is $p_d = 1 - (1 - \frac{1}{n})^{(m-r)k}$. The second set also have $m - r$ unique elements, so it has an independent probability of p_d that one of these elements sets the bit (the probabilities are independent because the elements are unique to each set). The probability that exactly one of the sets make the bit one is $2p_d(1 - p_d)$. This is all conditioned on none of the common elements setting the bit, so overall $p_b = (1-p_c)2p_d(1-p_d) = (1-\frac{1}{n})^{rk}2\cdot(1-\frac{1}{n})^{(m-r)k}(1-(1-\frac{1}{n})^{(m-r)k}) = 2(1-\frac{1}{n})^{mk}(1-(1-\frac{1}{n})^{(m-r)k})$. The expected number of differences is then np_b. We can get bounds on the deviation using the Poisson paradigm. Using this to estimate r is a matter of statistics and it's a bit different. We need to have some assumption on the distribution of r. Then, we determine an empirical $\widehat{p_b}$, and from this we can perform a maximum likelihood estimation of $\widehat{r}$. How good this estimate is depends on the assumed distribution of r and the value of r that we see (of course, we can't hope to reliably estimate a very small r, except with a linear number of trials). I believe that for an assumed uniform distribution on r and using the bound via the Poisson paradigm, we can approximate $\frac{r}{n}$ within an additive ε with constant probability, but I didn't check all calculations.

2.

We associate with each tree a polynomial, defined recursively as follows. For leaves, the polynomial is x_0. If the tree has height $h \geq 1$, the polynomial is $(x_h - P_{v_1}) \ldots (x_h - P_{v_k})$, where $v_1, \ldots, v_k$ are the children of the root, and P_{v_i} is the polynomial associated with the subtree rooted at v_i. We now argue that the polynomials associated with two trees are symbolically equal iff the trees are isomorphic. This is shown by induction on the heigh of the tree. It is obvious for $h = 0$ (leaves are always isomorphic, and $x_0 = x_0$). Now consider two trees of height h. If the trees are isomorphic, there is a bijection between the children of the roots, such that the subtrees are isomorphic. The two polynomials are products of terms corresponding to the subtrees; if the subtrees are isomorphic by some association, they will have equal polynomials, so the product will also be the same (it is commutative, so the association between the subtrees is irrelevant). Also, we use x_h for both trees, because if they are isomorphic, they must have the same height.

Now assume the polynomials are symbolically equal. For this, it must be the case that the trees have the same height, because the polynomials must have the same maximum i such that x_i appears in the polynomial. Now, the maximum degree of x_h is the number of children; thus, the two roots must have the same number of children k. Because both polynomials come from trees of height h, with roots having k children, they can be written as $(x_h - P_{v_1}) \ldots (x_h - P_{v_k}) = (x_h - P_{u_1}) \ldots (x_h - P_{u_k})$. The polynomials can be viewed as polynomials in one indeterminate x_h, with coefficients coming from the ring $\mathbb{R} = \mathbb{K}[x_1, \ldots, x_{h-1}]$. Then, P_{v_i} are the roots of the first polynomial, and P_{u_i} are the roots of the second polynomial. Since $\mathbb{R}[x_h]$ is a unique factorization domain, the roots of two equal polynomials must be equal. So there is some association

between $\{v_i\}$ and $\{u_i\}$ which makes the polynomials P_{u_i} and P_{v_j} symbolically equal. These polynomials characterize trees of heigh $h-1$ so by induction, the tress are isomorphic. But if we can group the subtrees in isomorphic pairs, the big trees are also isomorphic, qed.

Now all we have to do is test that the two polynomials are symbolically equal. Note that they have degree $h+1 \leq n$. Also, all coefficients are ± 1. Thus, the polynomials are also symbolically equal if working over the field $\mathbb{K} = \mathbb{F}_p$, for $p \geq n^2$. By the Schwartz-Zippel lemma, plugging in random values from $\mathbb{K}$ for the variables will yield the same evaluation with probability $\leq \frac{1}{n}$. This can be amplified to whp by trying $O(1)$ times. Evaluating the polynomial at these random points can be done in $O(n)$. All values fit in $O(\lg n)$ bits, and we can evaluate the polynomial in linear time by recursing on the tree, by the definition of the polynomial.

3.

(a) The algorithm is as follows. Multiply all weights by $2m^2 + 10$. Add to each edge a random integer drawn from $\{1 \dots 2m\}$. Find the min-weight perfect matching as in the algorithm described in the class (min-weight with respect to the new weights). The result is also min-weight perfect matching (with respect to the initial weights) with probability at least $1/2$.

To see the correctness, let M be the value of the min-weight perfect matching (wrt to initial weights), and M' the value of the min-weight perfect matching (wrt to new weights). Note that $(2m^2+10)M \leq M' \leq (2m^2+10)M + 2m^2 < (2m^2+10)(M+1)$. This means that if a perfect matching is found (wrt to new weights), then it has minimum weight perfect matching wrt to initial weights too.

Also, note that, with probability at least $1/2$, there will exist exactly one perfect matching that minimizes sum of new weights modulo $2m^2 + 10$ (due to the isolating lemma). This directly implies that with probability $\geq 1/2$, there will exists exactly one perfect matching that minimizes the sum of new weights. This means that with probability $\geq 1/2$, the algorithm succeeds in finding a perfect matching, which is the min-weight wrt to initial weights.

(b) If an edge has weight w then the corresponding Tutte matrix entry is given value 2^w for the determinant calculation. Thus, if the input weights have a polynomial number of bits, then the number in the matrix will have an exponential number of *bits* so we will not be able to perform arithmetic operations on the efficiently.

(c) Given the graph G, build a complete bipartite graph G' where edge (i,j) has weight 1 if the $(i,j) \in G$ and weight 2 if $(i,j) \notin G$. Any matching in G that has k edges corresponds to a matching in G' of total weight $k + 2(n-k) = 2n - k$. Thus a matching of minimum weight in G' corresponds to a maximum matching in G.

4.

Let the distance matrix be D. We assume $A_{ij} = \infty$ if no edge between i and j exists.

The new APSP algorithm is a slight modification of the APSP algorithm from the book (page 288). Note that for any node i, and any neighbor k, we have that $D_{ij} - B \leq D_{kj} \leq D_{ij} + B$. Let $\Delta = 2B + 3$. Then, for any i, if $A_{ik} < \infty$ and $D_{kj} + A_{ik} \equiv D_{ij} (\mod \Delta)$, then k is a valid candidate for S_{ij}.

We iterate over q from 0 to B, filling in entries of S_{ij}.

For a fixed value of q, we compute matrices $D^{(s)}$, for $s = 0 \dots \Delta - 1$, where $D^{(s)}_{kj} + q \equiv s (\mod \Delta)$. Compute A', where $A'_{ij} = 1$ iff $A_{ij} = q$ (and 0 otherwise). Finally, compute $W^{(s)}_{ij} = \mathrm{BPWM}(A', D^s)$. Then, for all i, j, if there is a witness $W^{(s)}_{ij}$ (i.e., if $(A'D^{(s)})_{ij} > 0$), then $S_{ij} = W^{D_{ij} \mod \Delta}_{ij}$.

We have to verify two properties: 1) S_{ij} is a valid successor and 2) all values S_{ij} are filled in.

For property one, consider any $W_{ij}^{(D_{ij} \mod \Delta)}$ which represents a witness k (i.e., the case when $(A' D^{(D_{ij} \mod \Delta)})_{ij} > 0$) for some q. This means that $D_{ij} \equiv A_{ik} + D_{kj} (\mod \Delta)$. Due to property $D_{ij} - B \leq D_{kj} \leq D_{ij} + B$, we have that $D_{ij} = A_{ik} + D_{kj} (\mod \Delta)$.

For property two, note that there must be some k such that $D_{ij} = A_{ik} + D_{kj} (\mod \Delta)$. Therefore, S_{ij} is filled in when $q = A_{ik}$.

Running time is $O(B^2 MM(n) \log^2 n)$.