Massachusetts Institute of Technology
6.856J/18.416J: Randomized Algorithms
David Karger

April 2, 2021

Problem Set 5 Solutions

Problem 1. Let T and P be the text and pattern matrices respectively. For each (i, j), let

$$W_{i,j} = \sum_{0 \le i', j' < m} 2^{i'+j' \times m} T_{i+i', j+j'}.$$

In words, $W_{i,j}$ is the number we get by flattening out the $m \times m$ window (whose upper left corner is at i, j) and interpreting the resulting binary string as a binary number.

Let $P = \sum_{0 \le i', j' < m} 2^{i'+j' \times m} P_{1+i', 1+j'}$. Notice that $W_{i,j} - P$ has at most m^2 prime factors for a fixed i, j, which implies that

$$\Pr\left(W_{i,j} - P = 0 \pmod{p} | W_{i,j} \neq P\right) \le \frac{m^2}{\pi(\tau)} = O\left(\frac{1}{t}\right),$$

where p is a random prime smaller than $\tau = tm^2 \log(tm^2)$ and $\pi(\tau) = O(\tau / \log(\tau))$ denotes the number of primes smaller than τ. Since there are total of $O(n^2)$ fingerprints, the overall failure probability is $O\left(\frac{n^2}{t}\right)$ by the union bound. Thus by choosing $t = n^4$, it follows that by picking a random prime p in the range of τ, p does not divide any $W_{i,j} - P$ with high probability.

Now we simply have to check if $W_{i,j} - P = 0 \pmod{p}$. To get the required runtime $O(n^2)$, we present an efficient way to obtain the next fingerprint from the previous one.

Notice that

$$\begin{aligned}
W_{i+1,j} &= \sum_{0\le i',j'<m} 2^{i'+j'\times m} T_{i+1+i',j+j'} \\
&= \sum_{0\le j'<m} 2^{m-1+j'\times m} T_{m+i,j+j'} \\
&\quad + \sum_{0\le i',j'<m-1} 2^{i'+j'\times m} T_{i+1+i',j+j'} \\
&= \sum_{0\le j'<m} 2^{m-1+j'\times m} T_{m+i,j+j'} \\
&\quad + \frac{1}{2} \sum_{0\le j'<m-1, 1\le i'<m} 2^{i'+j'\times m} T_{i+i',j+j'} \\
&= \sum_{0\le j'<m} 2^{m-1+j'\times m} T_{m+i,j+j'} \\
&\quad + \frac{1}{2} W_{i,j} \\
&\quad - \frac{1}{2} \sum_{0\le j'<m} 2^{j'\times m} T_{i,j+j'}.
\end{aligned}$$

Let $R_{i,j} = \sum_{0\le j'<m} 2^{j'\times m} T_{i,j+j'}$. This is the horizontal “slice” located at i, j.

We have

$$W_{i+1,j} = \frac{1}{2}\left(W_{i,j} + 2^m R_{m+i,j} - R'_{i,j}\right).$$

In words, this means we can compute the value of the new window, which is slided down by one vertical unit, by subtracting one horizontal slice and adding one horizontal slice.

Notice that $R_{i,j}$ can be easily precomputed as follows (using the familiar trick in 1D):

$$\begin{aligned}
R_{i,j+1} &= \sum_{0\le j'<m} 2^{j'\times m} T_{i,j+1+j'} \\
&= \frac{1}{2^m}\left(R_{i,j} - T_{i,j} + 2^{m^2} T_{i,m+j}\right).
\end{aligned}$$

Runtime analysis:

1. Find $R_{i,0}$ for all i. [It takes $O(nm)$]
2. Use $R_{i,j+1} = \frac{1}{2^m}\left(R_{i,j} - T_{i,j} + 2^{m^2} T_{i,m+j}\right)$ and $R_{i,0}$ to find $R_{i,j}$. [$O(n^2)$]
3. Use the formula $W_{i,j} = \sum_{0\le i'<m} 2^{i'} R_{i+i',j}$ to find $W_{0,j}$ for all j. [$O(nm)$]
4. Use $W_{i+1,j} = \frac{1}{2}\left(W_{i,j} + 2^m R_{m+i,j} - R'_{i,j}\right)$ to find $W_{i,j}$. [$O(n^2)$]
5. Compute P. [$O(m^2)$]

Problem 2. **(a)** 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 $p_b = (1 - p_c)2p_d(1 - p_d) = (1 - \frac{1}{n})^{rk}2 \cdot (1 - \frac{1}{n})^{(m-r)k}(1 - (1 - \frac{1}{n})^{(m-r)k}) = 2(1 - \frac{1}{n})^{mk}(1 - (1 - \frac{1}{n})^{(m-r)k}) \approx 2e^{-mk/n}(1 - e^{(m-r)k/n})$.

(b) If B is the number of indices on which the Bloom filters differ, then part (a), shows that,

$$\mathbb{E}[B] = 2n\left(1 - \frac{1}{n}\right)^{mk}\left(1 - \left(1 - \frac{1}{n}\right)^{(m-r)k}\right) \approx 2ne^{-km/n}(1 - e^{-(m-r)k/n})$$

Assuming B is concentrated around it's expectation, we can estimate r as,

$$m + \frac{n}{k}\log\left(1 - \frac{B}{2ne^{-mk/n}}\right)$$

Problem 3. **(a)** We hash m elements to a $\frac{n}{r}$ space with a universal hash function. Fix some x not in the set. The expected number of elements with which it collides is m times the probability that it collides with a fixed element of the set. By universality (even weak universality suffices), this is $\frac{r}{n}$, so we expect x to collide with $\frac{mr}{n}$ elements. Let B denote x's bit. By Markov, the probability that B is set is at most $\frac{mr}{n}$, since

$$\Pr(B \geq 1) \leq \frac{\mathbb{E}[B]}{1} = \mathbb{E}[B] = \frac{mr}{n}.$$

We are using r hash functions, chosen at random. The probability computed above is for fixed elements, over the choice of the hash function. So, since the hash functions are independent, these probabilities are independent, and x is a false positive with probability at most $(\frac{mr}{n})^r$.

(b) We are trying to minimize $(\frac{mr}{n})^r$ as a function of r. Since $x \to \ln x$ is increasing for $x > 0$, this is equivalent to minimizing $\ln \cdots = r(\ln r + \ln \frac{m}{n})$. Taking the derivative with respect to r, and setting this to zero, we get $\ln r + \ln \frac{m}{n} + r(\frac{1}{r}) = 0$, or $\ln r = -1 - \ln \frac{m}{n} = \ln \frac{n}{em}$. So the best choice is $r = \frac{n}{em}$ (it can be seen that we actually found a minimum because for very small and very large r we have high probabilities). So the optimal probability of false positives is $e^{-n/em} \approx (.6922)^{n/m}$.

There was a paper in SODA'05 showing how to get optimal error, $(.5)^{-n/m}$ (where m is the space in bits), even with universal hashing.

Problem 4. 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 $(x_h - P_{v_1}) \dots (x_h - P_{v_k})$, where $v_1, \dots, v_k$ 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 height 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 $(x_h - P_{v_1}) \dots (x_h - P_{v_k}) = (x_h - P_{u_1}) \dots (x_h - P_{u_k})$. The polynomials can be viewed as polynomials in one indeterminate x_h, with coefficients coming from the ring $\mathbb{K} = \mathbb{F}[x_1, \dots, x_{h-1}]$. Then, P_{v_i} are the roots of the first polynomial, and P_{u_i} are the roots of the second polynomial. Since $\mathbb{K}[x_h]$ 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 at most n by a simple inductive argument. We also need to bound the maximum absolute value of the coefficients. We will instead bound the sum of absolute values of the coefficients by 2^{n-1}. The base case is simple since $n = 1$ and the sum of absolute values of coefficients is 1. If we use $n(v)$ to denote the number of vertices in the subtree rooted by v, then by induction, the sum of absolute values of coefficients of P_v can be bounded by

$$(1 + 2^{n(v_1)-1})(1 + 2^{n(v_2)-1}) \cdots (1 + 2^{n(v_k)-1}),$$

which is at most

$$(2^{n(v_1)})(2^{n(v_2)}) \cdots (2^{n(v_k)}) = 2^{n(v)-1}.$$

Thus, maximum absolute value of any coefficient is 2^{n-1} and the maximum absolute value of any coefficient of the difference between two polynomials is at most 2^n. Thus, if two trees are not isomorphic, the difference between their corresponding polynomials will have a nonzero coefficient whose absolute value is at most 2^n, which contains at most n distinct prime factors. Thus, if we pick p randomly from the first n^2 primes, two non-isomorphic trees will have different polynomials even over the field $\mathbb{F} = \mathbb{F}_p$ with probability at least $1 - 1/n$. We can sample such prime p by repeatedly sampling integers of value $O(n^2 \lg n)$, and use a randomized primality test to check if the sampled integer is a prime. By the Schwartz-Zippel lemma, plugging in random values from $\mathbb{F}$ 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.

Problem 5. **(a)** Let the vector $m = [m_1, \cdots, m_k]$ denote the vector of the message sent to the entry vertices. As the value at each vertex is a linear combination of the values m_i, for each vertex $v \in V$, we assign a row vector $a(v) \in R^k$ which shows the coefficients of this linear combination. That is, $A(v) = a(v) \cdot m^T$ is the sum of the messages arrived at each node. Note that using this notation, for an edge $e = (u, v)$, the value u sends to v, is equal to $(w(e)A(u))$. Now it is enough to prove that each $a(v)_i$ is equal to the sum of all the weights of the paths from s_i to v, because rows of M are $a(t_1), \cdots, a(t_k)$. For simplicity we assume that the weight of a zero length path from a node to itself is equal to 1.

We prove the argument by induction. Since the graph is a DAG, we can consider the topological order of the nodes in which all the edges go forward. The base case, suppose v is the first vertex in this order. Since v has no incoming edge, it receives no message and hence $a(v) = \vec{0}$ and clearly there is no path from any s_i to v in this case and thus the argument holds.

For the induction step, there are two cases. First we consider the case where $v \neq s_l$ for any value of l. Suppose the set of incoming edges to v is $e_1, \cdots, e_t$

having their other vertex at $u_1, \cdots, u_t$, then the value v receives is equal to

$$\begin{aligned} A(v) &= \sum_{j=1}^{t} w(e_j)A(u_j) \\ &= \sum_{j=1}^{t}\sum_{i=1}^{k} w(e_j)a(u_j)_i m_i \\ &= \sum_{i=1}^{k} m_i \sum_{j=1}^{t} w(e_j)a(u_j)_i \\ &= \sum_{i=1}^{k} m_i a(v)_i = a(\vec{v})\vec{m}^T \end{aligned}$$

Since the argument holds for all vertices u_j, it is true that $a(u_j)_i$ shows the sum of the weights of the paths from s_i to u_j. Therefore, $w(e_j)a(u_j)_i$ shows the sum of the weights of the paths from s_i to v which use the edge e_j and thus $a(v)_i$ is equal to the sum of all the weights of the paths from s_i to v and the argument holds for v.

The other case is when $v = s_l$. This case is similar to the previous one except that since none of the u_j could have received a message from s_l, we have $a(u_j)_l = 0$. However s_l is receiving the message m_l from the source and thus the total coefficient of m_l in $A(v)$ is equal to 1, i.e. $a(v)_l = 1$ which by our assumption is equal to the weight of the path from s_l to s_l.

(b) The determinant formula is $det(M) = \sum_\pi (-1)^{sgn(\pi)} \prod_{i=1}^{k} M_{i,\pi(i)}$ where π takes value over all permutations of $[k]$. From part (a), $M_{i,\pi(i)} = \sum_{p\in P_i} w(p)$ where P_i is the set of paths from s_i to $t_{\pi(i)}$ and $w(p)$ shows the weight of a path. Therefore we have

$$\prod_{i=1}^{k} M_{i,\pi(i)} = \prod_{i=1}^{k}\sum_{p\in P_i} w(p) = \sum_{p_1\in P_1,\cdots,p_k\in P_k} \prod_{i=1}^{k} w(p_i)$$

where the inner term $\prod_{i=1}^{k} w(p_i)$ by definition is exactly the weight of a path system. Also since different permutation have disjoint set of path systems, the determinant is a signed sum of the weights of path systems.

(c) Any value k flow from s to t is a set of k vertex (and thus edge) disjoint paths from s_i to $t_{\pi(i)}$ for some permutation π over $[k]$. Suppose $E' = \{e_1, \cdots, e_t\}$ is the set of all edges in these paths. It can be seen that given a set of feasible edges E' for a path system, there is only one path system which uses exactly edges of E' (because they are vertex disjoint, by following the edges going out from the entry vertices we can find the paths and they never collide and never there are more than one option). So there are no two terms corresponding to E' in the sum to cancel out each other and each feasible E' contributes $(+or-)\prod_{e\in E'} w(e)$ to the sum.

Note: We can also prove a stronger version of this result, namely, that non-vertex disjoint path systems always cancel each other out in $det(M)$. In particular, consider some path system $S = \{p_1, \ldots, p_k\}$ that is not vertex disjoint and let i be the smallest index such that p_i intersects with some other path in the path system. Let j be the smallest index of paths that intersect with p_i and let v denote some vertex, where p_i and p_j intersect. Now we construct a path p'_i as follows: Follow p_i until we hit v, then follow p_j. Similarly, we can also construct p'_j, and from there a path system S', where we replace p_i and p_j by p'_i and p'_j, respectively. Moreover, note that both S and S' appear in $det(M)$ for different permutations π, π'. In particular, S' was constructed from S by swapping two paths, i.e., swapping the destination nodes $t_{\pi(i)}$, $t_{\pi(j)}$ for the two nodes s_i, s_j. In other words, $t_{\pi(i)} = t_{\pi'(j)}$ and vice versa. This implies that the permutation π' associated with S' must have opposite parity of the permutation of π associated with S, thus the sign of S is opposite of the sign of S'. Finally, we note that $w(S) = w(S')$, where $w(S)$ denotes the weight of the path system S, and conclude that since $w(S)$ and $w(S')$ have different signs in $det(M)$, they cancel each other. Since the above argument holds for any non-vertex disjoint path system, only vertex disjoint path systems remain.

(d) From parts (b) and (c), it can be seen that the determinant is a polynomial of degree at most $|E|$, therefore by Schwartz-Zippel Lemma, the probability that a random assignment of weights in Z_p is a root is at most $\frac{|E|}{p}$ and thus it is enough to choose $p \geq |E|^2$ to reduce the probability to $\frac{1}{|E|}$.

(e) From part (d), with high probability $det(M) \neq 0$ and thus M is invertible. So having $A(t_1), \cdots A(t_k)$, we can compute $m_1, \cdots, m_k$.