CS 170 HW 14

Due 2020-05-06, at 10:00 pm

1 Study Group

List the names and SIDs of the members in your study group. If you have no collaborators, you must explicitly write none.

2 Opting for releasing your solutions

We are considering releasing a subset of homework submissions written by students for students to see what a full score submission looks like. If your homework solutions are well written, we may consider releasing your solution. If you wish that your solutions not be released, please respond to this question with "No, do not release any submission to any problems". Otherwise, say "Yes, you may release any of my submissions to any problems".

3 Communicating across a galaxy

Alice is given an arbitrary string \( x \in \{0, 1\}^n \) and Bob is given an arbitrary string \( y \in \{0, 1\}^n \). Their goal is output yes if \( x = y \) and no otherwise (with “nontrivial” probability). They wish to come up with a communication protocol with as low communication complexity as possible. In this question we sketch the key idea of 3 algorithms to solve this problem; your job is to:

1. formalize each algorithm sketch,
2. analyze and upper bound the asymptotic communication complexity of your formalization of the algorithm,
3. lower bound the correctness probability of each algorithm.

The algorithms are as follows:

(a) **Prime number based idea.** Let \( n_x \) and \( n_y \) be the numbers that strings \( x \) and \( y \) represent in binary respectively.

- Alice chooses a uniformly random prime \( p \) in the interval \([1, n^2]\).
- Alice computes \( f_x := n_x \mod p \).
- Alice sends \( f_x \) and \( p \) to Bob.
- Bob computes \( f_y := n_y \mod p \).
- If \( f_x = f_y \), Bob sends 1 to Alice; otherwise Bob sends 0 to Alice.

You may use that for any \( N \) the number of primes less than \( N \) is \( \Theta(\frac{N}{\ln N}) \) without proof.

(b) **Polynomials based idea.**
Solution:

1. When \( x = y \), this protocol always succeeds. When \( x \neq y \), this protocol fails exactly when \( n_x - n_y = 0 \mod p \). This happens exactly when \( m := |n_x - n_y| \) is divisible by \( p \). Since \( m \leq 2^n \), it has at most \( n \) prime divisors. Since there are \( \Theta(n^2/\ln n) \) primes less than \( n^2 \), the probability that \( p \) is equal to one of the prime divisors of \( m \) is bounded by \( \Theta(\ln/n) \). The number of bits communicated is \( O(\log n) \).

2. Denote by \( q_x \) and \( q_y \) the polynomials that Alice and Bob respectively evaluate at \( r \). Once again, the protocol always succeeds when \( x = y \) and when \( x \neq y \) the protocol fails exactly when \( q_x(r) - q_y(r) = 0 \mod p \). Since the polynomial \( q := q_x - q_y \) is a nonzero polynomial of degree at most \( n \) it has at most \( n \) roots when evaluated \( \mod p \). The protocol fails exactly when \( r \) happens to be one of the roots of \( q \). This happens with probability at most \( \frac{n}{p^n} \) which is at most \( \frac{1}{n} \). The number of bits communicated is \( O(\log n) \).

3. If \( x = y \), this protocol always succeeds. If \( x \neq y \), then \( f_x^{(i)} - f_y^{(i)} = \sum_{j=1}^{n} (x_j - y_j)z_j^{(i)} \). Let \( j^* \) be an index such that \( x_{j^*} \neq y_{j^*} \) for any choice of the remaining bits (i.e. bits other than the \( j^* \)th bit) in the string \( z^{(i)} \), there is at most one choice of \( z_j^{(i)} \) to make the above sum 0, and hence \( f_x^{(i)} - f_y^{(i)} \) is nonzero with probability at least .5. By independence of the random strings the failure probability is at most \( \frac{1}{2^n} \).
4 Era of Ravens

(a) Design an algorithm that takes in a stream $z_1, \ldots, z_M$ of $M$ integers in $[n]$ and at any time $t$ can output a uniformly random element in $z_1, \ldots, z_t$. Your algorithm may use at most polynomial in $\log n$ and $\log M$ space. Prove the correctness and analyze the space complexity of your algorithm. Your algorithm may only take a single pass of the stream. 

Hint: $\frac{1}{t} = 1 \cdot \frac{1}{2} \cdot \frac{2}{3} \cdot \frac{3}{4} \cdots \frac{t-1}{t}$.

(b) For a stream $S = z_1, \ldots, z_{2n}$ of $2n$ integers in $[n]$, we call $j \in [n]$ a duplicative element if it occurs more than once. Prove that $S$ must contain a duplicative element, and design an algorithm that takes in $S$ as input and with probability at least $1 - \frac{1}{n}$ outputs a duplicative element. Your algorithm may use at most polynomial in $\log n$ space. Prove the correctness and analyze the space complexity of your algorithm. Your algorithm may only take a single pass of the stream.

Solution:

1. Maintain a counter $n_1$ initially 0 to keep track of how many elements have arrived so far and maintain a “current element” $x$ initially 0. When an element $z$ arrives, increment $n_1$ by 1 and replace $x$ with $z$ with probability $1/n_1$. When queried at any time, output $x$. To analyze the probability that $z_t$ is outputted at time $t' > t$, observe that $z_t$ must be chosen and then never replaced. This happens with probability

$$\left(\frac{1}{t}\right) \cdot \left(\frac{t}{t+1} \cdot \frac{t+1}{t+2} \cdots \frac{t'-1}{t'}\right).$$

2. $S$ must contain a duplicative element by the pigeonhole principle. The algorithm is to maintain $\log n$ independent instantiations of the sampling algorithm from part (a) and when a new element $z$ arrives at time $t$, first query all the instantiations and if any of them outputs $(z,*)$, ignore the rest of the stream and output $z$ as the ‘duplicative element’ at the end of the stream. Otherwise, stream $(z,t)$ as input to each of the independent instantiations of the sampling algorithm and continue. If the stream ends without the algorithm ever ‘committing’ to a $z$, output ‘failed’. Note that if the algorithm returns an element, it is certainly a duplicative element. We now turn our attention to upper bounding the probability that the algorithm returns ‘failed’. Suppose the algorithm returns failed, then let $(x_1,t_1), \ldots, (x_{\log n},t_{\log n})$ be the samples held by each of the instantiations at the end of the stream. Each $t_i$ is a uniformly random number between 1 and $2n$. Note that for the algorithm to have failed $t_i$ must be the time of the final occurrence of element $x_i$ (otherwise $x_i$ would have been identified as a duplicative element in the next occurrence of $x_i$). Since there are at most $n$ distinct values for $x_i$, there are at most $n$ distinct times $t$ such that the element $z$ that arrived at time $t$ is the final occurrence of $z$ in the stream. Thus, the probability that $t_i$ is the timestamp of a final occurrence is bounded by $1/2$. Thus the probability of the algorithm failing is bounded by $1/2^{\log n} = 1/n$. 


5 Evasions

In this problem for a hash function \( h \in H \), we refer to \( n_h(j) \) as the number of elements \( i \) such that \( h(i) = j \).

(a) Let \( H \) be the set of all functions from \([n] \to [n]\). Suppose we choose a uniformly random \( h \) from \( H \), show that except with probability \( o(n) \),

\[
\max_j n_h(j) \leq O \left( \frac{\log n}{\log \log n} \right).
\]

Hint: You may use the fact that \( \binom{n}{t} \leq \left( \frac{en}{t} \right)^t \) without proof.

(b) Let \( H \) be a 2-universal hash family. Suppose we choose a uniformly random \( h \) from \( H \), show that except with probability \( \frac{1}{t^2} \),

\[
\max_j n_h(j) \leq C t \sqrt{n}
\]

for some absolute constant \( C > 0 \).

Solution:

1. Let \( T = 2e \log n / \log \log n \). Let \( x_1, \ldots, x_T \) be any \( T \) elements of \([n]\). For a fixed \( j \), the probability that \( h(x_1), \ldots, h(x_T) \) are all equal to \( j \) is equal to \( \frac{1}{n^T} \). By a union bound, the probability that there exists some collection of \( T \) elements that all map to \( j \) is bounded by \( \frac{1}{n^T} \left( \frac{e n}{T} \right)^T = \left( \frac{e}{T} \right)^T \leq \frac{1}{n^T} \). Now taking a union bound over all \( j \) reveals that

\[
\max_j n_h(j) \leq T
\]

except with probability \( \frac{1}{n^T} \).

2. Let \( X_{u,v} \) be the indicator random variable that \( h(u) = h(v) \). On one hand we have

\[
\sum_{1 \leq u < v \leq n} X_{u,v} = \sum_{j=1}^{n} n_h(v)(n_h(v) - 1) / 2 = \left( \sum_{j=1}^{n} n_h(v)^2 / 2 \right) - n / 2 \geq \max_j n_h(j)^2 / 2 - n / 2,
\]

and rearranging the above we get:

\[
\frac{n}{2} + \sum_{1 \leq u < v \leq n} X_{u,v} \geq \max_j n_h(j)^2 / 2.
\]

Applying linearity of expectation to the above gives:

\[
n \geq E \max_j n_h(j)^2.
\]

By Markov’s inequality, except with probability \( \frac{1}{T^2} \), \( \max_j n_h(j)^2 \leq t^2 n \), which is equivalent to

\[
\max_j n_h(j) \leq C t \sqrt{n}.
\]