Lecture 7: Fingerprinting

David Woodruff

How to Pick a Random Prime

- How to pick a random prime in the range {0, 1, …, M-1}?
  - Pick a random integer X in the range {0, 1, …, M-1}
  - Check if X is a prime. If so, output it. Else go back to the first step

- How to pick a random integer X?
  - Pick a uniformly random bit string of length $\lfloor \log_2 \mathrm{M} \rfloor + 1$
  - If it represents a number $< \mathrm{M}$, output X. Else go back to the last step
  - In expectation, repeat this step at most twice
- How to check if X is prime?
  - Miller-Rabin primality test very efficient but fails with tiny probability
  - Agrawal-Kayal-Saxena has a worse running time, but deterministic
- How likely is X to be prime?

Density of Primes

- Let $\pi(n)$ be the number of primes in the set {1, 2, …, n}
- Prime Number Theorem: $\lim_{n\to\infty} \frac{\pi(n)}{n/\ln} = 1$
- Chebyshev: $\pi(n) > n/\ln n$ for every $n \geq 2$
  - If we want at least k primes in {1, 2, …, n}, then $n \geq 2k \lg k$, if $k \geq 4$
- Dusart: For n > 60184, we have $\frac{n}{\ln n-1.1} > \pi(n) > \frac{n}{\ln n -1}$

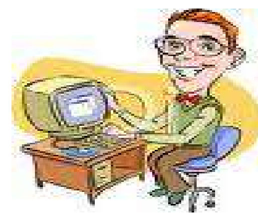
String Equality Problem

Alice

x

Is x = y?

Bob



y

- x and y are N-bit strings
- Alice and Bob want to exchange messages to decide if x = y
- Alice could send x to Bob but this takes N communication
    - Is there a more efficient scheme?

String Equality Problem

- Suppose we are OK if we achieve a probabilistic guarantee:
  - If x = y, then Pr[Bob says **equal**] = 1
  - If $x \neq y$, then Pr[Bob says **unequal**] $\geq 1 - \delta$

- Protocol
  - Alice chooses a random prime p from {1, 2, …, M} for $M = \lceil 2 \cdot (5N) \cdot \lg(5N) \rceil$
  - She sends Bob p and the value $h_p(x) = x \bmod p$, where we think of x as an integer in {0, 1, 2, …, 2^N-1}
  - If $h_p(x) = y \bmod p$, Bob says **equal**, else he says **unequal**

String Equality Problem

- Lemma: If x = y, then Bob always says **equal**
- Proof: If x = y, then x mod p = y mod p. So Bob's test will always succeed
- Lemma: If $x \neq y$, then Pr[Bob says **equal**] $\leq .2$
- Proof: Interpret $x, y \in \{0, 1, 2, \ldots, 2^N - 1\}$

  If Bob says **equal**, then x mod p = y mod p, i.e., (x-y) = 0 mod p

  So p divides D = |x-y|, and D < 2^N

  D = $p_1 \cdot p_2 \cdots p_k$ for primes $p_1, \ldots, p_k$ which may repeat

  Since each $p_i \geq 2$, we have k < N

  Pr[p divides D] $\leq \frac{N}{\text{number of primes in } \{1,2,\ldots,M\}} \leq \frac{N}{5N} = \frac{1}{5}$ why?

Communication Cost

- If Alice were to naively send x to Bob, would take N bits of communication
- Instead she sends a prime p and x mod p, where p is in {1, 2, …, M} and $\mathrm{M} = \lceil 2 \cdot (5\mathrm{N}) \cdot \lg(5\mathrm{N}) \rceil$
- Communication = O(log p) = O(log M) = O(log N + log log N) = O(log N) bits

Reducing the Error Probability

- We have 20% error probability, how to reduce it to δ?

- Repeat the scheme r = $\log_5(\delta^{-1})$ times independently with primes $p_1, \ldots, p_r$ ∈ {1, 2, ..., M}, and $M = \lceil 2 \cdot (5N) \cdot \lg(5N) \rceil$
  - Bob outputs **equal** if and only if x = y mod p_i for each i
  - If x = y, Bob outputs **equal** with probability 1
  - If $x \neq y$, Bob outputs **equal** with probability at most $\left(\frac{1}{5}\right)^{\lg_5(\frac{1}{\delta})} \leq \delta$
  - Communication cost is O(log$(1/\delta)$ $\log N$). Can we do better?

- If instead Alice sets M = $2 \cdot sN \lg(sN)$, the number of primes in {1, 2, ..., M} is at least sN, and so error probability is 1/s. Set s = $1/\delta$.
  - Communication is O(log M) = O(log s + log N) = O(log$(1/\delta)$ + log N)

Fingerprinting (the Karp-Rabin Method)

- In the string-matching problem, we have
  - A text T of length m
  - A pattern P of length n

- Goal: output all occurrences of the pattern P inside the text T
  - If T = abracadabra and P = ab, the output should be {0,7}

abracadabra

- Consider $h_p(x) = x \bmod p$ for $x \in \{0,1\}^n$, where we think of x as an integer in {0, 1, 2, …, 2^n-1}

Fingerprinting (the Karp-Rabin Method)

- $\mathrm{h_p(x) = x \bmod p}$ for $\mathrm{x \in \{0,1\}^n}$
- Create x' by dropping the most significant bit of x, and appending a bit to the right
  - E.g., if x = 0011001, then x' could be 0110010 or 0110011
- Given $\mathrm{h_p(x) = z}$, can we compute $\mathrm{h_p(x')}$ quickly?
- Suppose $\mathrm{x'_{lb}}$ is the lowest-order bit of x', and $\mathrm{x_{hb}}$ is the highest order bit of x
- $\mathrm{x' = 2\left(x - x_{hb} \cdot 2^{n-1}\right) + x_{lb}{}'}$
- Since $\mathrm{h_p(a+b) = \left(h_p(a) + h_p(b)\right) \bmod p}$, and $\mathrm{h_p(2a) = 2h_p(a) \bmod p}$,

$$\mathrm{h_p(x') = \left(2h_p(x) - x_{hb} \cdot h_p(2^n) + x'_{lb}\right) \bmod p}$$

- *Given $h_p(x)$ and $h_p(2^n)$, this is just O(1) arithmetic operations mod p*

Fingerprinting (the Karp-Rabin Method)

- $T_{a \ldots b}$ denotes the string from the a-th to b-th positions of T, inclusive
- Goal: output all locations a in {0, 1, …, m-n} such that $T_{a \ldots a+(n-1)} = P$

1. Pick a random prime p ∈ {1, 2, …, M} with $M = \lceil 2s\, n\, \lg(sn) \rceil$ for some s
2. Compute $h_p(P)$ and $h_p(2^n)$ and store the results
3. Compute $h_p(T_{0 \ldots n-1})$ and check if it equals $h_p(P)$. If so, output **match** at location 0
4. For each i ∈ {0, …, m-n-1}, compute $h_p(T_{i+1 \ldots i+n})$ using $h_p(T_{i \ldots i+n-1})$ and $h_p(2^n)$. If $h_p(T_{i+1 \ldots i+n}) = h_p(P)$, output **match** at location i+1

Error Probability

- $m - n + 1 \leq m$ comparisons, each with probability at most 1/s of failure
- By a union bound, the probability there is at least one failure is at most m/s
- If s = 100m, we succeed on all comparisons with probability $\geq$ 99/100
- $M = \lceil 2s\, n \lg(sn) \rceil$ = O(mn log(mn)), so O(log m + log n) bits to store
- Since p in {1, 2, …, M}, p takes O(log m + log n) bits to store
- Assume unit-cost RAM model, so operations on O(log(mn)) bits take O(1) time

Running Time

- Computing $h_p(x)$ for n-bit x can be done in O(n) time. Why?
  - Generate powers of 2, or use shifting
- So $h_p(P)$, $h_p(2^n)$, and $h_p(T_{0,\ldots,n-1})$ can be computed in O(n) time
- Computing $h_p(T_{i+1\ldots i+n})$ using $h_p(T_{i\ldots i+n-1})$ and $h_p(2^n)$ can be done in O(1) time!
- Total time is O(m + n), which is optimal

Fingerprinting Extensions

- Fingerprinting also works for strings $x \in \{0, 1, 2, \dots, q-1\}^n$
- Think of x as an integer $\sum_{i=0,\dots,n-1} q^i \cdot x_i$ in its q-ary representation
- Drop the leftmost digit of x to create x', and append a digit to the right
  - If $x = \ x_{n-1}, x_{n-2}, x_{n-3}, \dots, x_0$, then $x' = x_{n-2}, x_{n-3}, \dots, x_0, x_0'$
- $x' = q\left(x - x_{n-1} \cdot q^{n-1}\right) + x_0{}'$
- $h_p(x') = \left(q \cdot h_p(x) - x_{n-1} \cdot h_p(q^n) + x_0'\right) \bmod p$

- Given $h_p(x)$ and $h_p(q^n)$, if q < p, computing $h_p(x')$ requires O(1) arithmetic operations mod p

Extensions

- How would you solve the following?

- Given an $\mathrm{m}_1 \mathrm{x}\ \mathrm{m}_2$-bit rectangular binary text T, and an $\mathrm{n}_1 \mathrm{x}\ \mathrm{n}_2$- bit pattern P, where $\mathrm{n}_1 \leq \mathrm{m}_1$ and $\mathrm{n}_2 \leq \mathrm{m}_2$, find all occurrences of P inside T. Show how to do this in $\mathrm{O}(\mathrm{m}_1 \mathrm{m}_2)$ time

- Assume you can do modular arithmetic of integers at most poly($\mathrm{m}_1 \mathrm{m}_2$) in O(1) time

Extensions

- Walk through the columns of T, and create fingerprints $h_q(T_{[i,i+n_1-1],j})$ of the n_1 values

$$T_{i,j}, T_{i+1,j}, \dots, T_{i+n_1-1,j}$$

- $q \leq \mathrm{poly}(m_1 m_2 n_1)$
- Walk through the rows of T, and for the (i,j)-th entry, create a fingerprint of the n_2 values

$$h_q(T_{[i,i+n_1-1],j}), h_q(T_{[i,i+n_1-1],j+1}), \dots, h_q(T_{[i,i+n_1-1],j+n_2-1})$$

- Note: the fingerprints are of q-ary instead of binary strings, but when fingerprinting these strings we can use a prime $p \leq \mathrm{poly}(m_1 m_2 n_1 n_2)$. Show this!

- Walking through the columns and rows and creating the fingerprints, and comparing with the hash of the pattern P, takes $O(m_1 m_2)$ time