Topic 5: Fingerprinting

David Woodruff

How to Pick a Random Prime

- How to pick a random prime in the range \(\{1, 2, \ldots, M\}\)?
  - Pick a random integer \(X\) in the range \(\{1, \ldots, M\}\).
  - Check if \(X\) is a prime. If so, output it. Else go back to the first step.

- 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 primes in the set \(\{1, 2, \ldots, n\}\)

- Prime Number Theorem: \(\lim_{n \to \infty} \frac{\pi(n)}{n/\ln n} = 1\)

- Chebyshev: \(\pi(n) > n/\ln n\) for every \(n \geq 2\)
  - If we want at least \(k\) primes in \(\{1, 2, \ldots, n\}\), 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

- \(x\) 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 \( \Pr[\text{Bob says equal}] = 1 \)
  • If \( x \neq y \), then \( \Pr[\text{Bob says unequal}] \geq 1 - \delta \)

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

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 \( \{1, 2, \ldots, M\} \) and \( M = \lceil 2 \cdot (5N) \cdot \log(5N) \rceil \)

• 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 \( \delta \)?

• Repeat the scheme \( r = \log_2(6^{-1}) \) times independently with primes \( p_1, \ldots, p_r \in \{1, 2, \ldots, M\} \), and \( M = \lceil 2 \cdot (5N) \cdot \log(5N) \rceil \)
  • 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 \( \left(\frac{1}{2}\right)^{\log_2 p_i} \leq \delta \)
  • Communication cost is \( O(\log(1/\delta) \log N) \). Can we do better?

• If instead Alice sets \( M = 2 \cdot sN \log(sN) \), the number of primes in \( \{1, 2, \ldots, M\} \) 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 $T = \text{abracadabra}$ and $P = \text{ab}$, the output should be $\{0,7\}$

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

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

Fingerprinting (the Karp-Rabin Method)

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

1. Pick a random prime $p \in \{1,2,\ldots,M\}$ with $M = \lceil 2s \cdot n \cdot \lg(sn) \rceil$ for some $s$
2. Compute $h_p(P)$ and $h_p(2^n)$ and store the results
3. Compute $h_p(T[a..a+(n-1)])$ and check if it equals $h_p(P)$. If so, output match at location $a$
4. For each $i \in \{0,\ldots,m-n-1\}$, compute $h_p(T[i+1..i+n])$ using $h_p(T[i..i+n-1])$ and $h_p(2^n)$. If $h_p(T[i+1..i+n]) = h_p(P)$, 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 $\geq 99/100$
- $M = \lceil 2s \cdot n \cdot \lg(sn) \rceil = O(mn \cdot \log(mn))$, so $O(\log m + \log n)$ bits to store
- Since $p$ in $\{1,2,\ldots,M\}$, $p$ 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 given $h_p(2^n)$

- So $h_p(P)$, $h_p(2^n)$, and $h_p(T_{0, n-1})$ can be computed in $O(n)$ time

- Computing $h_p(T_{i+1, j+n})$ using $h_p(T_{i, j+n-1})$ and $h_p(2^n)$ can be done in $O(1)$ time!

- Total time is $O(m + n)$, which is optimal

Extensions

- How would you solve the following?

  - Given an $m_1 \times m_2$-bit rectangular binary text $T$, and an $n_1 \times n_2$-bit pattern $P$, where $n_1 \leq m_1$ and $n_2 \leq m_2$, find all occurrences of $P$ inside $T$. Show how to do this in $O(m_1 m_2)$ time

  - Assume you can do modular arithmetic of integers at most $\text{poly}(m_1 m_2)$ in $O(1)$ time

Fingerprinting Extensions

- Fingerprinting also works for strings $x \in \{0, 1, 2, ..., q - 1\}^n$
  - Think of $x$ as an integer $\sum_{i=0}^{n-1} q^i \cdot x_i$ in its $q$-ary representation
  - Drop the leftmost digit of $x$ to create $x'$, and append a digit to the right
    - If $x = x_{n-1}, x_{n-2}, x_{n-3}, ..., x_0$, then $x' = x_{n-2}, x_{n-3}, ..., x_0, x'_0$
    - $x' = q(x - x_{n-1} \cdot q^{n-1}) + x'_0$
    - $h_p(x') = (q \cdot h_p(x) - x_{n-1} \cdot h_p(q^n) + x'_0) \mod p$

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

Extensions

- Walk through the columns of $T$, and create fingerprints $h_q(T_{i+n_1-1,j})$ of the $n_1$ values $T_{i,j}, T_{i+1,j}, ..., T_{i+n_1-1,j}$
  - $q \leq \text{poly}(m_1 m_2 n_1)$

- Walk through the rows of $T$, and for the $(i,j)$-th entry, create a fingerprint of the $n_2$ values $h_q(T_{i+j+n_2-1,j}), h_q(T_{i+j+n_2-1,j+1}), ..., h_q(T_{i+j+n_2-1,j+n_2-1})$

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