COMPSCI 614: Randomized Algorithms with Applications to Data Science

Prof. Cameron Musco
University of Massachusetts Amherst. Spring 2024.
Lecture 6

Logistics

- Problem Set 1 is due tomorrow at midnight.
- I am holding office hours directly after class today.
- No class or office hours on Thursday.
- Problem Set 2 will be posted later this week.

Summary

Last Time:

- Stronger concentration bounds for sums of independent random variables. I.e., exponential concentration bounds.
- Chernoff and Bernstein bound.
- Application to balls-into-bins and linear probing analysis.

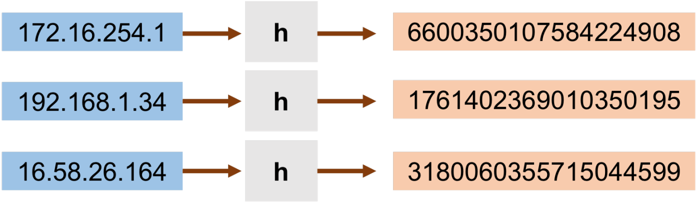
Today:

- Random hash functions and fingerprinting.
- Applications to pattern matching and communication complexity.

Random Hashing and Fingerprinting

Random Hash Functions

A random hash function maps inputs to random outputs.



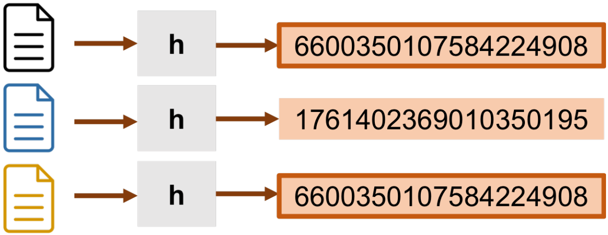
h is picked randomly, but after it is picked it is fixed – so a single input is always mapped to the same output.

```
import random
a = random.randint(1,100)
b = random.randint(1,100)
def myHash(x):
  return (a*x+b) % 100
```

```
import random
def myHash(x):
  a = random.randint(1,100)
  b = random.randint(1,100)
  return (a*x+b) % 100
```

Fingerprinting

Random hash functions are often used to reduce large files down to hash ‘fingerprints’, which can be used to check equality of files (deduplication), detect updates/corruptions, etc.



- Key requirement is that two distinct files are unlikely to have the same hash – low collision probability.
- In practice h is often a deterministic ‘cryptographic’ hash function like SHA or MD5 – hard to analyze formally.

Rabin Fingerprint

Rabin Fingerprint: Interpret a bit string $x_1, x_2, \ldots, x_n$ as the binary representation of the integer $x = \sum_{i=1}^{n} x_i \cdot 2^{i-1}$. Let

$$\mathbf{h}(x) = x \mod p,$$

where p is a randomly chosen prime in $[1, tn \log tn]$.

Prime Number Theorem: There are $\approx \frac{tn \log tn}{\log(tn \log tn)} = \Theta(tn)$ primes in $[1, tn \log tn]$. So p is chosen randomly from $\Theta(tn)$ possible values.

Claim: For $x, y \in [0, 2^n]$ with $x \neq y$, $\Pr[\mathbf{h}(x) = \mathbf{h}(y))] = O(1/t)$.

- If $\mathbf{h}(x) = \mathbf{h}(y)$, then it must be that $x - y \mod p = 0$. I.e., p divides $x - y$. So we must bound the probability of this occuring.
- **Note:** This is not a cryptographic hash function – it is relatively easy to find x, y with $\mathbf{h}(x) = \mathbf{h}(y)$ given p, or blackbox access to h. However, this is fine in many applications.

Rabin Fingerprint Analysis

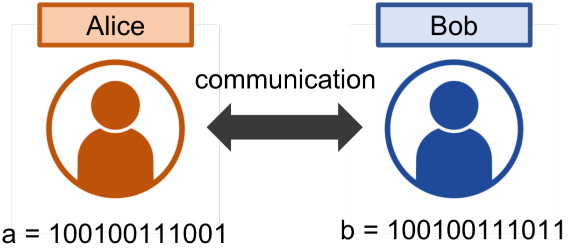
Think-Pair-Share 1: How many unique prime factors can an integer in $[-2^n, 2^n]$ have?

Think-Pair-Share 2: What is the probability that a random prime p chosen from $[1, tn \log tn]$ divides $x - y \in [-2^n, 2^n]$? I.e., that $h(x) = h(y)$? Recall: There are $\Theta(tn)$ primes in the range $[1, tn \log tn]$.

Fingerprinting Application 1: Communication Complexity

Fingerprinting for Equality Testing

Equality Testing Communication Problem: Alice has some bit string $a \in \{0,1\}^n$. Bob has some string $b \in \{0,1\}^n$. How many bits do they need to communicate to determine if $a = b$ with probability at least $2/3$?



Fingerprinting for Equality Testing

Equality Testing Protocol:

- Alice picks a random prime $p \in [1, tn \log tn]$ for some large constant t.
- Alice sends p, along with the Rabin fingerprint $\mathbf{h}(a) := a \mod p$ to Bob. $[O(\log p) = O(\log n)$ bits]
- Bob uses p to compute $\mathbf{h}(b) := b \mod p$.
- If $\mathbf{h}(a) = \mathbf{h}(b)$, Bob sends 'YES' to Alice. Else, he sends 'No'. [1 bit]

Correctness: If $a = b$ both Alice and Bob always output 'YES'. If $a \neq b$ they output 'NO' with probability $1 - O(1/t) \geq 2/3$ if t is set large enough.

Complexity: Uses just $O(\log p) = O(\log n)$ bits of communication in total.

Deterministic Equality Testing

How many bits must Alice and Bob send if they want to check equality of $a, b \in \{0, 1\}^n$ without using randomness?

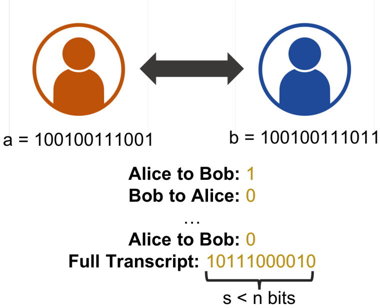
Claim: Any deterministic protocol for equality testing requires sending $\Omega(n)$ bits.

- An exponential separation between randomized and deterministic protocols!
- Unlike for running times, for communication complexity problems there are often large provable separations between randomized and deterministic protocols.

Deterministic Equality Testing Lower Bound

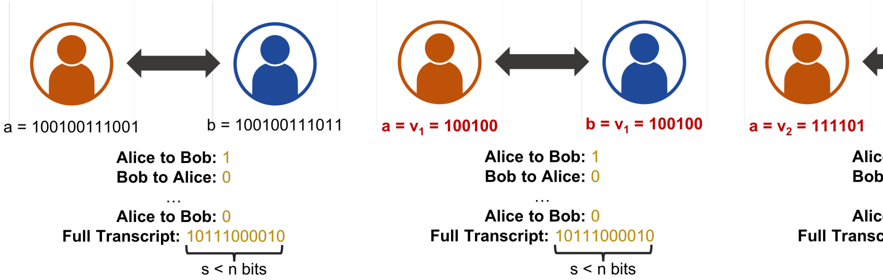
Claim: Any deterministic protocol for equality testing requires sending $\Omega(n)$ bits.

- Assume without loss of generality that Alice and Bob alternate sending 1 bit at a time – at most doubles the number of bits.
- If Alice and Bob send $s < n$ bits, in total, there are 2^s possible conversations they may have.



Deterministic Equality Testing Lower Bound

If Alice and Bob send $s < n$ bits, in total, there are 2^s possible conversations they may have.



- Since there are $2^n > 2^s$ possible inputs, there must be two different inputs $v_1 \neq v_2$, such that given $a = b = v_1$ or $a = b = v_2$, the protocol outputs 'YES' and has identical transcripts.
- But then the players will send the same messages and output 'YES' also when Alice is given $a = v_1$ and Bob is given $b = v_2$. This violates correctness!

Application 2: Pattern Matching

Pattern Matching

Given some document $x = x_1x_2 \dots x_n$ and a pattern $y = y_1y_2 \dots y_m$, find some j such that

$$x_jx_{j+1}, \dots, x_{j+m-1} = y_1y_2 \dots y_m.$$

x = The quick brown **fox** jumped across the pond…

y = fox

Can assume without loss of generality that the strings are binary strings.

What is the 'naive' running time required to solve this problem?

Rolling Hash

We will use the fact that the Rabin fingerprint is a rolling hash.

- Letting $X_j = \sum_{i=0}^{m-1} x_{j+i} \cdot 2^{m-1-i}$ be the integer value represented by the binary string $x_j x_{j+1}, \ldots, x_{j+m-1}$, we have

$$X_{j+1} = 2 \cdot X_j - 2^m x_j + x_{j+m}.$$

- Thus, since for any X, $\mathbf{h}(X) = X \mod p$,

$$\mathbf{h}(X_{j+1}) = 2 \cdot \mathbf{h}(X_j) - 2^m x_j + x_{j+m} \mod p.$$

- Given $\mathbf{h}(X_j)$, this hash value can be computed using just $O(1)$ arithmetic operations.

Rabin-Karp Algorithm

The Rabin-Karp pattern matching algorithm is then:

- Pick a random prime $p \in [1, tm \log mt]$, for $t = cn$.
- Let $Y = \mathrm{h}(y)$ be the Rabin fingerprint of the pattern.
- Let $H = \mathrm{h}(X_1)$ be the Rabin fingerprint of the first block of text.
- For $j = 1, \ldots, x_{n-m+1}$
  - If $Y == H$, return j.
  - Else, $H = 2 \cdot H - 2^m x_j + x_{j+m} \mod p$.

Runtime: Takes $O(m + n)$ time in total. $O(m)$ for the initial hash computations, and $O(1)$ for each iteration of the for loop.

Correctness: The probability of a false positive at any step is upper bounded by $\frac{1}{t} = \frac{1}{cn}$. Thus, via a union bound, the probably of a false positive overall is at most $\frac{n}{cn} = \frac{1}{c}$.

Questions on Random Hashing?

Interesting topics I am not covering:

- Constructions of universal hash functions.
- Constructions of k-wise independent hash functions.
- Concentration bounds and hash table analysis using k-wise independent hash functions. See Lectures 3-4 of Jelani Nelson's course notes for some material on this (link on schedule page).
- Connections to pseudorandom number generators (PRGs).