COMPSCI 690RA: Randomized Algorithms and Probabilistic Data Analysis

Prof. Cameron Musco

University of Massachusetts Amherst. Spring 2022.

Lecture 4

Logistics

- Problem Set 1 was due last night – solutions are posted on the Assignments page.
- Problem Set 2 will be posted by Friday.
- Next week I am away, so the lecture will be held over Zoom. I encourage you to attend live, but I will also record it. If you don't have a good place to Zoom from, you can of course come to the classroom and attend from there.
- Going forward, I will be posting quizzes a bit later on Wednesday/Thursday so that I can adapt them better to what is covered in class.

Summary

Last Time:

- Stronger concentration bounds for sums of independent random variables. I.e., exponential concentration bounds.
- Chernoff and Bernstein bound.
- Application to estimation via sampling and linear probing analysis.
- Start on randomized hash function and fingerprints.

Today:

- Finish fingerprints and applications to pattern matching and communication complexity.
- ℓ_0 sampling, with applications to graph sketching and streaming.

Quiz Question

You roll a fair 6-sided die n times independently. You look at the difference between the number of times you rolled a "1" the number of times you rolled a "2". Roughly, how big do we expect this difference to be in magnitude?

- a. $\Theta(n)$
- b. $\Theta(\sqrt{n})$
- c. $\Theta(\log n)$
- d. $\Theta\left(\frac{\log n}{\log \log n}\right)$

Quiz Question

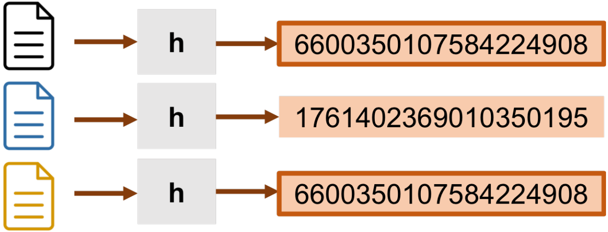
You roll a fair 6-sized die n times independently. Let S be the sum of rolls. $\mathbb{E}[S] = 3.5n$. What is the smallest threshold t for which $Pr[|S - 3.5n| > t] \leq 1/n$?

- a. $O(\log n)$
- b. $O(n \log n)$
- c. $O(\sqrt{n \cdot \log n})$
- d. $O(\sqrt{n} \cdot \log n)$
- e. $O(\sqrt{n})$

Random Hashing and Fingerprinting

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 $x - y$.
- **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.

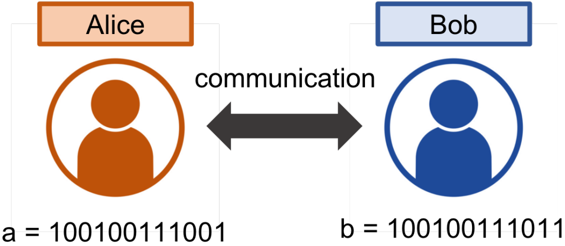
Think-Pair-Share 1: How many unique prime factors can an integer in $[-2^n, 2^n]$ have?

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

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?

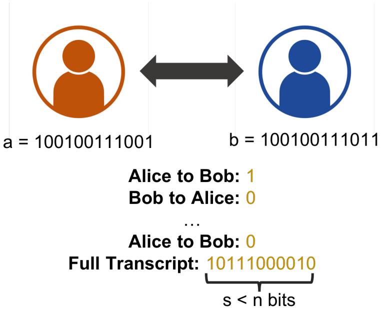
Claim: Any deterministic protocol for equality testing requires sending $\Omega(n)$ bits.

- 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

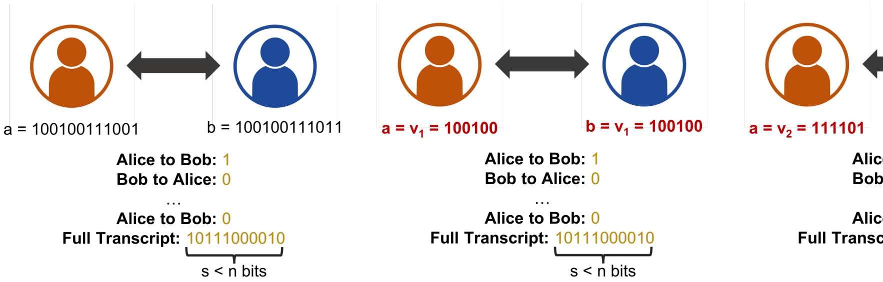
Claim: Any deterministic protocol for equality testing requires sending $\Omega(n)$ bits.

- 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 $x = x_1 x_2 \ldots x_n$ and a pattern $y = y_1 y_2 \ldots y_m$, find some j such that

$$x_j x_{j+1}, \ldots, x_{j+m-1} = y_1 y_2 \ldots y_m.$$

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 $Y = \mathsf{h}(y)$ be the Rabin fingerprint of the pattern.
- Let $H = \mathsf{h}(X_1)$ 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?

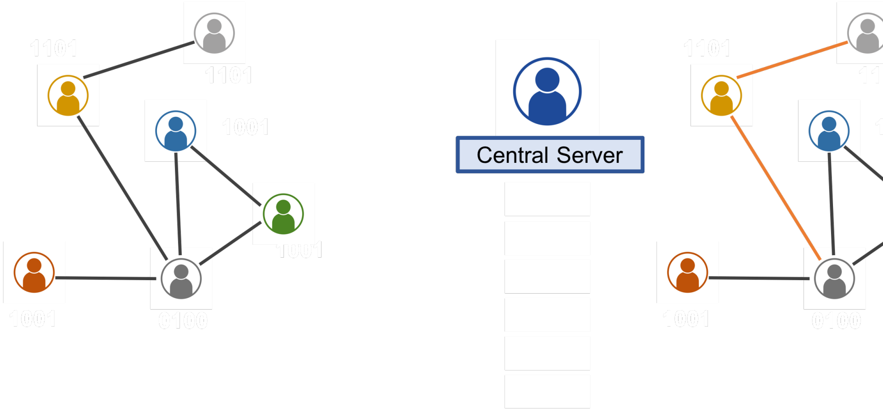
Interesting topics I am not covering:

- 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).

ℓ_0 Sampling and Graph Sketching

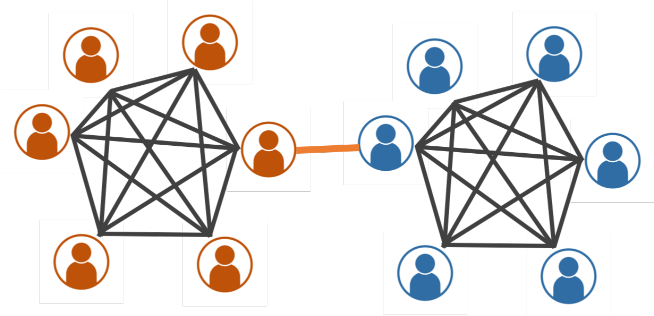
A Graph Communication Problem

Consider n nodes, each only knows its own neighborhood. They want to send messages to a central server, who will then determine if the graph is connected.



How large of messages (# bits) are needed to determine connectivity with high probability?

A Hard Case



- Surprisingly, for any input graph, the problem can be solved with high probability using just $O(\log^c n)$ bits per message!
- Solution will be based on a random linear sketch.

Key Ingredient 1: ℓ_0 Sampling

Theorem: There exists a distribution over random matrices $\mathbf{A} \in \mathbb{Z}^{O(\log^2 n)\times n}$ such that for any fixed $x \in \mathbb{Z}^n$, with probability at least $1 - 1/n^c$, we can learn (i, x_i) for some $x_i \neq 0$ from $\mathbf{A}x$.

Random sketching matrix **A** · x = **A**x

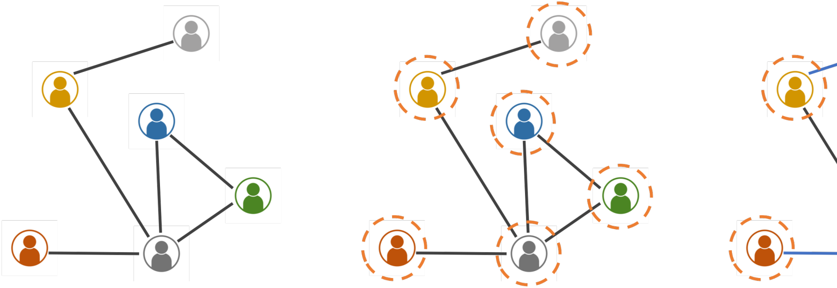
$$\begin{bmatrix} 1 & -1 & 0 & 0 & 1 & -1 & 0 & 1 \\ -1 & 0 & 1 & 1 & 0 & 0 & -1 & 0 \\ 1 & 1 & -1 & 0 & -1 & -1 & 0 & 1 \\ 0 & -1 & -1 & -1 & 1 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ -2 \\ 0 \\ 0 \\ 3 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ -2 \\ 1 \\ 5 \end{bmatrix}$$

Useful Property 1: Given t vectors $x_1, \ldots, x_t \in \mathbb{Z}^n$, can recover a nonzero entry from each with probability $\geq 1 - t/n^c$.

Useful Property 2: Given sketches $\mathbf{A}x_1$ and $\mathbf{A}x_2$, can easily compute $\mathbf{A}(x_1 + x_2)$ and recover a nonzero entry from $x_1 + x_2$ with high probability.

Key Ingredient 2: Boruvka's Algorithm

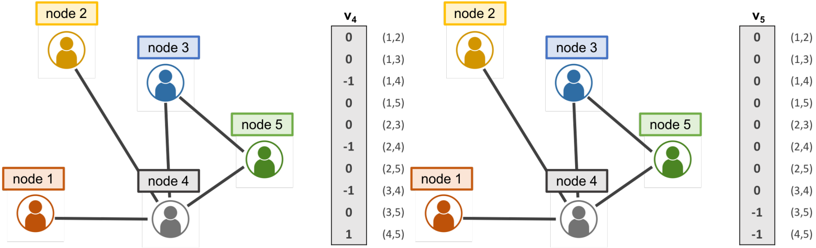
1. Initialize each node as its own connected component.
2. For each connected component, select an outgoing edge. Merge any newly connected components.
3. Repeat until no connected component has an outgoing edge. If at this point, all nodes are in the same component, then the graph is connected.



Converges in $\leq \log n$ rounds

Key Ingredient 3: Neighborhood Sketches

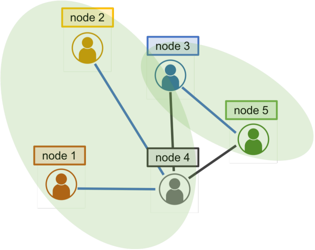
Each node i, can compute a vector $\mathbf{v}_i \in \mathbb{Z}^{\binom{n}{2}}$. v_i has a ± 1 for every edge in the graph and incident to node i. $+1$ is used for edges (i, j) and -1 for edges (j, i).



- Given an ℓ_0 sampling matrix $\mathbf{A} \in \mathbb{Z}^{O(\log^2 n) \times \binom{n}{2}}$, each node can compute $\mathbf{A}v_i \in \mathbb{Z}^{O(\log^2 n)}$ and send it to the central server.
- Using these sketches, with probability $\geq 1 - 1/n^c$, the central server can identify one edge incident to each node – i.e., they can simulate the first iteration of Boruvka's algorithm.

Simulating Boruvka's Algorithm via Sketches

- For independent ℓ_0 sampling matrices $\mathbf{A}_1, \ldots, \mathbf{A}_{\log_2 n}$, each node computes $\mathbf{A}_j v_i$ and sends these sketches to the central server. $O(\log^c n)$ bits in total.
- The central server uses $\mathbf{A}_1 v_1, \ldots, \mathbf{A}_1 v_n$ to simulate the first step of Boruvka's algorithm.
- For each subsequent step j, let $S_1, S_2, \ldots S_c$ be the current connected components. Observe that $\sum_{i \in S_k} v_i$ has non-zero entries corresponding exactly to the outgoing edges of S_k.



v_3	+	v_5	=		
0		0		0	(1,2)
0		0		0	(1,3)
0		0		0	(1,4)
0		0		0	(1,5)
0		0		0	(2,3)
0		0		0	(2,4)
0		0		0	(2,5)
1		0		1	(3,4)
1		-1		0	(3,5)
0		-1		-1	(4,5)

- So, from $\mathbf{A}_j \sum_{i \in S_k} v_i = \sum_{i \in S_k} \mathbf{A}_j v_i$, the server can find an outgoing

Implementing ℓ_0 Sampling

ℓ_0 Sampling Construction

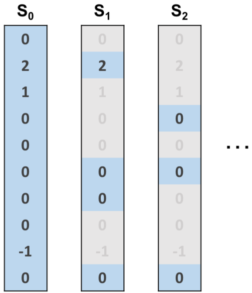
Theorem: There exists a distribution over random matrices $\mathbf{A} \in \mathbb{Z}^{O(\log^2 n) \times n}$ such that for any fixed $x \in \mathbb{Z}^n$, with probability at least $1 - 1/n^c$, we can learn (i, x_i) for some $x_i \neq 0$ from $\mathbf{A}x$.

Construction:

- Let $S_0, S_1, \ldots, S_{\log_2 n}$ be random subsets of $[n]$. Each element is included in S_j independently with probability $1/2^j$.
- For each S_j, compute $a_j = \sum_{i \in S_j} x_i$, $b_j = \sum_{i \in S_j} x_i \cdot i$ and $c_j = \sum_{i \in S_j} x_i \cdot r^i \mod p$, where r is a random value in $[p]$ and p is a prime with $p \geq n^c$ for some large constant c.
- **Exercise:** Show that the vector $[a_1, \ldots, a_{\log_2 n}, b_1, \ldots, b_{\log_2 n}, c_1, \ldots, c_{\log_2 n}]$ can be written as $\mathbf{A}x$, where $\mathbf{A} \in \mathbb{Z}^{3 \log_2 n \times n}$ is a random matrix.

Construction Intuition

We will recover a nonzero element from a sampling level when there is exactly one nonzero element at that level.



With good probability, there is will exactly one element at some level. Can improve success probability via repetition.

Recovering Unique Nonzeros

Recall: $S_0, \ldots, S_{\log_2 n}$ are random subsets of $[n]$, sampled at rates $1/2^j$. $a_j = \sum_{i \in S_j} x_i$, $b_j = \sum_{i \in S_j} x_i \cdot i$ and $c_j = \sum_{i \in S_j} x_i \cdot r^i \mod p$, where r is a random value in $[p]$ and $p = n^c$ for large enough constant c.

Claim 1: If there is a unique $i \in S_j$ with $x_i \neq 0$, then $a_j = x_i$ and $b_j = x_i \cdot i$. So, from these quantities we can exactly determine (i, x_j).

Claim 2: c_j lets us test if there is a unique such i. In particular, we check that $\frac{b_j}{a_j} \in [n]$ and that $c_j = a_j \cdot r^{b_j/a_j} \mod p$.

- If there is a unique $i \in S_j$ with $x_i \neq 0$, the test passes.
- If not, it fails with probability at most $\frac{n}{p} = \frac{1}{n^{c-1}}$.

The problem of recovering a unique $i \in S_j$ with $x_i \neq 0$ is called 1-sparse recovery.

Recovering Unique Nonzeros

Claim 2: c_j lets us test if there is a unique such i. In particular, we check that $\frac{b_j}{a_j} \in [n]$ and that $c_j = a_j \cdot r^{b_j/a_j} \mod p$.

- If there is a unique $i \in S_j$ with $x_i \neq 0$, the test passes.
- If not, it fails with probability at most $\frac{n}{p} \leq \frac{1}{n^{c-1}}$.

Proof via polynomial identity testing: If $|\{i \in S_j : x_i \neq 0\}| > 1$, then

$$p(r) = c_j - a_j r^{b_j/a_j} \mod p = \sum_{i \in S_j} x_i r^i - a_j r^{b_j/a_j} \mod p$$

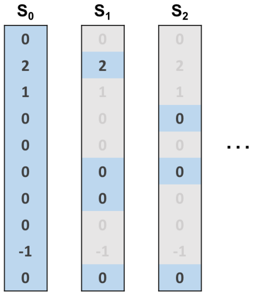
is a non-zero polynomial of degree at most n over $\mathbb{Z}_p$.

- This polynomial has $\leq n$ roots, so for a random $r \in [p]$, $\Pr[p(r) = 0] \leq \frac{n}{p}$.
- Thus, $c_j = a_j r^{b_j/a_j}$ with probability $\leq \frac{n}{p} \leq \frac{1}{n^{c-1}}$.

Completing The Analysis

Recall: $S_0, \ldots, S_{\log_2 n}$ are random subsets of $[n]$, sampled at rates $1/2^j$.

- If any S_j contains a unique i with $x_i \neq 0$, we will recover it.
- It remains to show that with good probability, at least one S_j contains such an i.



Claim: For j with $2^{j-2} \leq \|x\|_0 \leq 2^{j-1}$, $\Pr[|\{i \in S_j : x_i \neq 0\}| = 1] \geq 1/8$.