Lecture 28

Randomized Algorithms
• Contention resolution
• Global Minimum Cut

Sublinear-Time Algorithms
• Testing if a list is sorted

Reminder:
Fill out SRTEs online
• Don’t forget to click submit

Sofya Raskhodnikova
Randomization

Algorithmic design patterns.
- Greedy.
- Divide-and-conquer.
- Dynamic programming.
- Network flow.
- Randomization.

Randomization. Allow fair coin flip in unit time.

Why randomize? Can lead to simplest, fastest, or only known algorithm for a particular problem.

Ex. Symmetry breaking protocols, graph algorithms, quicksort, hashing, load balancing, Monte Carlo integration, cryptography.
Randomization

Algorithmic design patterns.

- Greedy.
- Divide-and-conquer.
- Dynamic programming.
- Network flow.
- Randomization.

Randomization. Allow fair coin flip in unit time.

- Why randomize? Can lead to simplest, fastest, or only known algorithm for a particular problem.
- Ex. Symmetry breaking protocols, graph algorithms, quicksort, hashing, load balancing, Monte Carlo integration, cryptography.

Not really a separate paradigm: integral part of algorithmic design
- E.g. Randomized divide & conquer is very common
  - Quicksort, “quickselect” median algorithm, ...

Tasks impossible deterministically: crypto, asynchronous consensus,...
13.1 Contention Resolution
Contention Resolution in a Distributed System

Contention resolution. Given \( n \) processes \( P_1, \ldots, P_n \), each competing for access to a shared database. If two or more processes access the database simultaneously, all processes are locked out. Devise protocol to ensure all processes get through on a regular basis.

Restriction. Processes can't communicate.

Challenge. Need symmetry-breaking paradigm.
Contestation Resolution: Randomized Protocol

Protocol. Each process requests access to the database at time t with probability \( p = \frac{1}{n} \).

Claim. Let \( S[i, t] = \) event that process \( i \) succeeds in accessing the database at time \( t \). Then \( \frac{1}{en} \leq \Pr[S(i, t)] \leq \frac{1}{2n} \).

Pf. By independence, \( \Pr[S(i, t)] = p (1-p)^{n-1} \).

- Setting \( p = \frac{1}{n} \), we have \( \Pr[S(i, t)] = \frac{1}{n} (1 - \frac{1}{n})^{n-1} \).

Useful facts from calculus. As \( n \) increases from 2, the function:
  - \( (1 - \frac{1}{n})^n \) converges monotonically from \( \frac{1}{4} \) up to \( \frac{1}{e} \)
  - \( (1 - \frac{1}{n})^{n-1} \) converges monotonically from \( \frac{1}{2} \) down to \( \frac{1}{e} \).

Most used inequality today: \( 1 - x \leq e^{-x} \).
Contention Resolution: Randomized Protocol

Claim. The probability that process $i$ fails to access the database in $e \cdot n$ rounds is at most $1/e$.
After $e \cdot n(c \ln n)$ rounds, the probability is at most $n^{-c}$.

Pf. Let $F[i, t] = \text{event that process } i \text{ fails to access database in rounds 1 through } t$. By independence and previous claim, $\Pr[F(i, t)] \leq \left(1 - \frac{1}{en}\right)^t$.

- Choose $t = \lceil e \cdot n \rceil$: $\Pr[F(i, t)] \leq \left(1 - \frac{1}{en}\right)^{en} \leq \frac{1}{e}$
- Choose $t = \lceil e \cdot n \rceil [c \ln n]$: $\Pr[F(i, t)] \leq \left(\frac{1}{e}\right)^{c \ln n} = n^{-c}$
Contestation Resolution: Randomized Protocol

Claim. The probability that all processes succeed within $2e \cdot n \ln n$ rounds is at least $1 - 1/n$.

Pf. Set $t = \lceil en \rceil \lceil 2 \ln n \rceil$. Let $F[t] = \text{event that at least one of the } n \text{ processes fails to access database in any of the rounds 1 through } t$.

$$\Pr[F[t]] = \Pr\left[ \bigcup_{i=1}^{n} F[i,t] \right] \leq \sum_{i=1}^{n} \Pr[F[i,t]] \leq n \cdot n^{-2} = 1/n.$$

Union bound. Given events $E_1, ..., E_n$, \[ \Pr\left[ \bigcup_{i=1}^{n} E_i \right] \leq \sum_{i=1}^{n} \Pr[E_i] \]
13.2 Global Minimum Cut
Global Minimum Cut

Global min cut. Given a connected, undirected graph $G = (V, E)$ find a cut $(A, B)$ of minimum cardinality.

Applications. Partitioning items in a database, identify clusters of related documents, network reliability, network design, circuit design, TSP solvers.

Review Question:

- Using FF as a subroutine, how can you find the global min cut?
- How long does it take?
Global Minimum Cut

Global min cut. Given a connected, undirected graph \( G = (V, E) \) find a cut \( (A, B) \) of minimum cardinality.

Applications. Partitioning items in a database, identify clusters of related documents, network reliability, network design, circuit design, TSP solvers.

Network flow solution.
- Replace every edge \((u, v)\) with two antiparallel edges \((u, v)\) and \((v, u)\).
- Pick some vertex \(s\) and compute min \(s-v\) cut separating \(s\) from each other vertex \(v \in V\).

False intuition. Global min-cut is harder than min s-t cut.
Contraction algorithm. [Karger 1995]

- Pick an edge $e = (u, v)$ uniformly at random.
- **Contract** edge $e$.
  - replace $u$ and $v$ by single new super-node $w$
  - preserve edges, updating endpoints of $u$ and $v$ to $w$
  - keep parallel edges, but delete self-loops
- Repeat until graph has just two nodes $v_1$ and $v_2$.
- Return the cut (all nodes that were contracted to form $v_1$).

![Diagram showing contraction algorithm](attachment:image.png)
**Contraction Algorithm**

**Claim.** The contraction algorithm returns a min cut with prob $\geq 2/n^2$.

**Pf.** Consider a global min-cut $(A^*, B^*)$ of $G$. Let $F^*$ be edges with one endpoint in $A^*$ and the other in $B^*$. Let $k = |F^*| = \text{size of min cut}$.

- In first step, algorithm contracts an edge in $F^*$ probability $k / |E|$.
- Every node has degree $\geq k$ since otherwise $(A^*, B^*)$ would not be min-cut. $\Rightarrow |E| \geq \frac{1}{2}kn$.
- Thus, in this step, algorithm contracts an edge in $F^*$ with probability $\leq 2/n$. 

![Diagram](image-url)
Contraction Algorithm

Claim. The contraction algorithm returns a min cut with prob $\geq 2/n^2$.

Pf. Consider a global min-cut $(A^*, B^*)$ of $G$. Let $F^*$ be edges with one endpoint in $A^*$ and the other in $B^*$. Let $k = |F^*| = \text{size of min cut}.$

- Let $G'$ be graph after $j$ iterations. There are $n' = n-j$ supernodes.
- Suppose no edge in $F^*$ has been contracted. The min-cut in $G'$ is still $k$.
- Since value of min-cut is $k$, $|E'| \geq \frac{1}{2}kn'$.
- Thus, algorithm contracts an edge in $F^*$ with probability $\leq \frac{2}{n'}$.

Let $E_j = \text{event that an edge in } F^* \text{ is not contracted in iteration } j.$

$$\Pr[E_1 \cap E_2 \cdots \cap E_{n-2}] = \Pr[E_1] \times \Pr[E_2 \mid E_1] \times \cdots \times \Pr[E_{n-2} \mid E_1 \cap E_2 \cdots \cap E_{n-3}]$$

$$\geq (1 - \frac{2}{n})(1 - \frac{2}{n-1}) \cdots (1 - \frac{2}{4})(1 - \frac{2}{3})$$

$$\geq \frac{(n-2)}{n} \cdot \frac{(n-3)}{n-1} \cdots \frac{2}{4} \cdot \frac{1}{3}$$

$$\geq \frac{2}{n(n-1)}$$

$$\geq \frac{2}{n^2}$$
Contraction Algorithm

**Amplification.** To amplify the probability of success, run the contraction algorithm many times.

**Claim.** If we repeat the contraction algorithm $n^2 \ln n$ times with independent random choices, the probability of failing to find the global min-cut is at most $1/n^2$.

**Pf.** By independence, the probability of failure is at most

$$\left(1 - \frac{2}{n^2}\right)^{n^2 \ln n} = \left[\left(1 - \frac{2}{n^2}\right)^{\frac{1}{2}n^2}\right]^{2\ln n} \leq \left(e^{-1}\right)^{2\ln n} = \frac{1}{n^2}$$

$$\uparrow$$

$$(1 - 1/x)^x \leq 1/e$$
Global Min Cut: Context

Best known deterministic. [Nagamochi-Ibaraki 1992] \( O(mn + n^2 \log n) \).

Remark. Our algorithm is slow since we perform \( \Theta(n^2 \log n) \) iterations and each takes \( \Omega(m) \) time.

Improvement. [Karger-Stein 1996] \( O(n^2 \log^3 n) \).
- Early iterations are less risky than later ones: probability of contracting an edge in min cut hits 50% when \( n / \sqrt{2} \) nodes remain.
- Run contraction algorithm until \( n / \sqrt{2} \) nodes remain.
- **Recursively** run contraction algorithm twice on resulting graph, and return best of two cuts.

Extensions. Naturally generalizes to handle positive weights.

Best known. [Karger 2000] \( O(m \log^3 n) \).
\( \backslash \) faster than best known max flow algorithm or deterministic global min cut algorithm
Sublinear-Time Algorithms

A sneak preview of the Spring ’12 course

Register soon: only 21 seats left!

First time at Penn State!
Vast Data: Impossible to Access All of It

Massive datasets
- world-wide web, online social networks, census data, genome project, scientific measurements, sales logs, ...

Long access time
- communication bottleneck (low-bandwidth connection)
- implicit data (an experiment per data point)
What Can We Hope For?

• What can an algorithm compute if it
  – reads only a tiny portion of the data?
  – runs in sublinear time?

• For most interesting problems algorithms must be
  – approximate
  – randomized
A Sublinear-Time Algorithm

Quality of approximation vs. Resources

- number of queries
- running time
Types of Approximation

Classical approximation

• need to compute a value
  ➢ output is close to the desired value
  ➢ example: average

• need to compute the best structure
  ➢ output is a structure with “cost” close to optimal
  ➢ example: minimum spanning tree

Property testing

• need to answer YES or NO
  ➢ intuition: output is a correct answer for a given input,
    or at least some input close to it
Property Tester Definition

[Rubinfeld Sudan 96, Goldreich Goldwasser Ron 98]

Probabilistic Algorithm

**YES**
- Accept with probability $\geq \frac{2}{3}$

**NO**
- Reject with probability $\geq \frac{2}{3}$

Property Tester

**YES**
- Accept with probability $\geq \frac{2}{3}$

**Far from YES**
- Don’t care
- Reject with probability $\geq \frac{2}{3}$

$\epsilon$-far = differs in many places ($\geq \epsilon$ fraction of places)
A Simple Example

Input: a list of \( n \) numbers \( x_1, x_2, \ldots, x_n \)

- **Question:** Is the list sorted?
  Requires reading entire list: \( \Omega(n) \) time

- **Approximate version:** Is the list sorted or \( \epsilon \)-far from sorted?
  (An \( \epsilon \) fraction of \( x_i \)'s have to be changed to make it sorted.)
  
  [Ergün Kannan Kumar Rubinfeld Viswanathan]: \( O((\log n)/\epsilon) \) time

- **Attempts:**
  1. **Test:** Pick a random \( i \) and reject if \( x_i > x_{i+1} \).
     
     Fails on: \( 1111111100000000 \) \( \leftarrow \) 1/2-far from sorted

  2. **Test:** Pick random \( i < j \) and reject if \( x_i > x_j \).
     
     Fails on: \( 10213243546576 \) \( \leftarrow \) 1/2-far from sorted
Is a list sorted or $\varepsilon$-far from sorted?

Idea: Associate positions in the list with vertices of the directed line.

Construct a graph (2-spanner)

- by adding a few “shortcut” edges $(i, j)$ for $i < j$
- where each pair of vertices is connected by a path of length at most 2

$\leq n \log n$ edges
Is a list sorted or $\epsilon$-far from sorted?

**Test** [Dodis Goldreich Lehman Raskhodnikova Ron Samorodnitsky]

Pick a random edge $(x_i,x_j)$ from the 2-spanner and **reject** if $x_i > x_j$.

![Graph](image)

**Analysis:**

- Call an edge $(x_i,x_j)$ **violated** if $x_i > x_j$, and **good** otherwise.
- If $x_i$ is an endpoint of a **violated** edge, call it **bad**. Otherwise, call it **good**.

**Claim.** All **good** numbers $x_j$ are sorted.

**Proof:** Consider any two good numbers, $x_i$ and $x_j$.

They are connected by a path of (at most) two **good** edges $(x_i,x_k), (x_k,x_j)$.

$\Rightarrow$ $x_i \leq x_k$ and $x_k \leq x_j$

$\Rightarrow$ $x_i \leq x_j$
Is a list sorted or \( \varepsilon \)-far from sorted?

**Test** [Dodis Goldreich Lehman Raskhodnikova Ron Samorodnitsky ]

Pick a random edge \((x_i, x_j)\) from the 2-spanner and **reject** if \(x_i > x_j\).

![Diagram of a graph with labeled vertices and edges]

**Analysis:**
- Call an edge \((x_i, x_j)\) **violated** if \(x_i > x_j\), and **good** otherwise.
- If \(x_i\) is an endpoint of a **violated** edge, call it **bad**. Otherwise, call it **good**.

**Claim.** All **good** numbers \(x_j\) are sorted.
- All sorted lists are accepted.
- If a list is \(\varepsilon\)-far from sorted, it has \(\geq \varepsilon n\) **bad** numbers.  (Claim)
  - Each **violated** edge contributes 2 **bad** numbers.
  \[ \geq \varepsilon n/2 \text{ **violated** edges, i.e., } \geq \varepsilon/(2 \log n) \text{ fraction of edges are **violated**.} \]
  - If \(s = (4\log n)/\varepsilon\) edges are checked, \(\Pr[\text{no **violated** edge discovered}]\)
    \[ \leq (1 - \varepsilon/(2 \log n))^s \leq e^{-\varepsilon/(2 \log n)s} = e^{-2} < \frac{1}{3} \]
We can determine if a list of $n$ numbers is sorted or $\epsilon$-far from sorted in $O\left(\frac{\log n}{\epsilon}\right)$ time.

[Fischer 01]: This cannot be improved.
13.3 Linearity of Expectation
Expectation

Expectation. Given a discrete random variables $X$, its expectation $E[X]$ is defined by:

$$E[X] = \sum_{j=0}^{\infty} j \Pr[X = j]$$

Waiting for a first success. Coin is heads with probability $p$ and tails with probability $1-p$. How many independent flips $X$ until first heads?

$$E[X] = \sum_{j=0}^{\infty} j \cdot \Pr[X = j] = \sum_{j=0}^{\infty} j (1-p)^{j-1} p = \frac{p}{1-p} \sum_{j=0}^{\infty} j (1-p)^j = \frac{p}{1-p} \cdot \frac{1-p}{p^2} = \frac{1}{p}$$

$\uparrow$  \hspace{0.5cm}  \uparrow$

$j$-1 tails  \hspace{0.5cm}  1 head
Expectation: Two Properties

Useful property. If $X$ is a 0/1 random variable, $E[X] = \Pr[X = 1]$.

\[
E[X] = \sum_{j=0}^{\infty} j \cdot \Pr[X = j] = \sum_{j=0}^{\infty} j \cdot \Pr[X = j] = \Pr[X = 1]
\]

Linearity of expectation. Given two random variables $X$ and $Y$ defined over the same probability space, $E[X + Y] = E[X] + E[Y]$.

Decouples a complex calculation into simpler pieces.
Guessing Cards

**Game.** Shuffle a deck of $n$ cards; turn them over one at a time; try to guess each card.

**Memoryless guessing.** No psychic abilities; can't even remember what's been turned over already. Guess a card from full deck uniformly at random.

**Claim.** The expected number of correct guesses is 1.

**Pf.** (surprisingly effortless using linearity of expectation)

- Let $X_i = 1$ if $i^{th}$ prediction is correct and 0 otherwise.
- Let $X = \text{number of correct guesses} = X_1 + \ldots + X_n$.
- $E[X_i] = \Pr[X_i = 1] = 1/n$.
- $E[X] = E[X_1] + \ldots + E[X_n] = 1/n + \ldots + 1/n = 1$. ▪

↑

linearity of expectation
**Guessing Cards**

**Game.** Shuffle a deck of $n$ cards; turn them over one at a time; try to guess each card.

**Guessing with memory.** Guess a card uniformly at random from cards not yet seen.

**Claim.** The expected number of correct guesses is $\Theta(\log n)$.

**Pf.**
- Let $X_i = 1$ if $i^{th}$ prediction is correct and 0 otherwise.
- Let $X = \text{number of correct guesses} = X_1 + \ldots + X_n$.
- $E[X_i] = \Pr[X_i = 1] = 1 / (n - i - 1)$.
- $E[X] = E[X_1] + \ldots + E[X_n] = 1/n + \ldots + 1/2 + 1/1 = H(n)$.

\[\ln(n+1) < H(n) < 1 + \ln n\]

\[\text{linearity of expectation}\]
Coupon Collector

**Coupon collector.** Each box of cereal contains a coupon. There are n different types of coupons. Assuming all boxes are equally likely to contain each coupon, how many boxes before you have $\geq 1$ coupon of each type?

**Claim.** The expected number of steps is $\Theta(n \log n)$.

**Pf.**
- Phase $j = \text{time between j and j+1 distinct coupons}$.
- Let $X_j = \text{number of steps you spend in phase j}$.
- Let $X = \text{number of steps in total} = X_0 + X_1 + \ldots + X_{n-1}$.

\[
E[X] = \sum_{j=0}^{n-1} E[X_j] = \sum_{j=0}^{n-1} \frac{n}{n-j} = n \sum_{i=1}^{n} \frac{1}{i} = nH(n)
\]

prob of success = $(n-j)/n$
⇒ expected waiting time = $n/(n-j)$