{ csci 3|6907 | Lecture 1 }

Hoeteck Wee · hoeteck@gwu.edu

Today

- Overview of this course
- Course administration
- Two randomized algorithms

Randomization in Computer Science

- Algorithm Design
  - Basic algorithmic problems, e.g. PRIMALITY (1977, 2002)
  - Practical problems, e.g. load-balancing
- Cryptography
  - Randomness provides secrecy, e.g. 4-digit PIN random in $\{0000, \ldots, 9999\}$.
- Computational Models
  - Random processes, e.g. natural selection & mutation in biology
  - Complex networks, e.g. social networks and the Internet

This Course

- Randomized algorithms
  - Simplicity: Randomized min-cut, median-finding and 2-SAT
  - Efficiency: Sublinear-time algorithms
  - Average-Case "Goodness": Load balancing
- Tools and techniques for probabilistic analysis
  - Tail bounds, e.g. Markov's inequality and Chernoff bounds
- Computational models
  - Random graphs

Administration

- Basic Information
  - Course webpage `http://www.seas.gwu.edu/~hoeteck/s13`
  - Contacting me `hoeteck@gwu.edu`
  - Webpage + email for disseminating information
  - Textbook: Probability and Computing: ..., by Mitzenmacher & Upfal
- Pre-requisites
  - Strong background in basic probability; basic algorithms course

Course Evaluation

- Homework: $\sim$ once every two weeks
- One programming assignment
- Final project
- Class attendance and participation

Polynomial Identity Testing

Identity Testing

Given two polynomials $p(x)$ and $q(x)$, decide whether $p \equiv q$ (that is, whether p is "identical" to q).

- "polynomials": coefficients are integers or field elements; degree $\leq d$
- "$p \equiv q$": coefficients for each monomial are the same, e.g. $(x+1)(x-1) \equiv x^2 - 1$
- "given": (1) list of coefficients, or (2) as a formula, e.g. $((x-1)^2+1)^3 + 4x$.

Polynomial Identity Testing

Identity Testing

Given two polynomials $p(x)$ and $q(x)$, decide whether $p \equiv q$ (that is, whether p is "identical" to q).

Identity Testing (special case)

Given a polynomial $p(x)$, decide whether $p \equiv 0$.

- To solve the general case, check whether $p(x) - q(x)$ is identical to 0.

Polynomial Identity Testing

IDENTITY TESTING Algorithm

1. Pick a number r uniformly at random from $\{1, 2, \ldots, 2d\}$.
2. Evaluate $p(r)$. If the result is 0, accept; else, reject.

- If $p(x) \equiv 0$, then algorithm always accepts.
- If $p(x) \not\equiv 0$, then algorithm accepts with probability $\leq \frac{1}{2}$.

Fact

A non-zero degree d polynomial has at most d roots.

Polynomial Identity Testing

Identity Testing Algorithm

1. Pick a number r uniformly at random from $\{1, 2, \ldots, 2d\}$.
2. Evaluate $p(r)$. If the result is 0, accept; else, reject.

- If $p(x) \equiv 0$, then algorithm always accepts.
- If $p(x) \not\equiv 0$, then algorithm accepts with probability $\leq \frac{1}{2}$.

Question

How can we reduce the error (i.e. the probability of accepting $p(x) \not\equiv 0$)?

Polynomial Identity Testing

1. Try all r in $\{1, 2, \ldots, d+1\}$.
   - Always outputs correct answer.
   - Problem: d may be as large as 2^n. e.g.

$$\overbrace{((((x+1)^2+1)^2+1)^2 \cdots +1)^2}^{\longleftarrow\; n \text{ times} \;\longrightarrow}$$

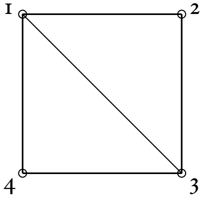
2. Replace $2d$ with $1000d$.
   - Reduces error to $1/1000$.
   - Disadvantage: need to compute with large numbers.

3. Repeat k times, using different random values r
   - Reduces error to $1/2^k$.
   - Advantage: works in general for any randomized algorithm.

Minimum cut

Definitions

1. Cut: set of edges whose removal render the graph disconnected
2. Minimum cut: cut of the smallest size (size = # edges in the cut)



- cut: $\{(1,2),(1,3),(1,4)\}$
- min-cut: $\{(1,4),(3,4)\}$ or $\{(1,2),(2,3)\}$

Easy Fact

$|$ minimum cut $| \leq$ minimum degree of any node.

Minimum cut

Definitions

1. Cut: set of edges whose removal render the graph disconnected
2. Minimum cut: cut of the smallest size (size = # edges in the cut)

Minimum Cut Problem

On input an undirected graph with n vertices, output a minimum cut.

Applications

- network reliability (nodes = machines, edges = connections)
- clustering webpages (nodes = webpages, edges = hyperlinks)

Minimum cut

Definitions

1. Cut: set of edges whose removal render the graph disconnected
2. Minimum cut: cut of the smallest size (size = # edges in the cut)

Minimum Cut Problem

On input an undirected graph with n vertices, output a minimum cut.

Algorithms

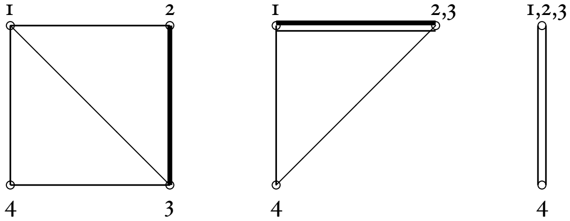
- "naive": compute s-t minimum cut n^2 times, total $O(mn^3)$ time.
- next: randomized algorithm based on edge contractions in $O(n^4)$ time.

Edge contraction

Operation Edge Contraction

Input: edge (u, v) in undirected graph

1. Merge vertices u and v.
2. Remove any self-loops and keep multi-edges.



remove $(2, 3)$ remove $(1, 2), (1, 3)$ left with $(1, 4), (3, 4)$

Randomized min-cut (Karger, 1993)

RandMinCut Algorithm

Input: undirected graph G.

1. Repeat: contract a random edge
2. Output the edges connecting the remaining two vertices.

- correctness?
- Each edge contraction reduces # vertices by 1.
- number of repetitions = $n - 2$
- running time = $O(n^2)$

Analysis

Fact 1

Let C be a cut. If we never contract an edge in C, then C remains a cut.

- PROOF: only contract edges, so $\langle$ edges, vertices $\rangle$ on left side of C stay on left side, and the same for right side.

Analysis

FACT 1

Let C be a cut. If we never contract an edge in C, then C remains a cut.

- fix a min-cut C of size k
- INTUITION: $\exists$ lots of edges, so we're unlikely to contract an edge in C
- GOAL: bound $\Pr[E_i]$ where E_i is "C survives the first i iterations".

Base case: $\Pr[E_1]$

1. degree of every vertex $\geq k$
2. # edges $\geq nk/2$
3. $\Pr[E_1] = 1 - \frac{k}{\text{\# edges}} \geq 1 - \frac{k}{nk/2} = \frac{n-2}{n}$

Analysis

Fact 2

Min-cut size never decreases.

- CLAIM: Any cut C in the new graph is also a cut in the original graph.
- PROOF: Induction. Any “contracted edge” must lie on same side of C.

Analysis

Fact 2

Min-cut size never decreases.

Iterative step: $\Pr[E_{i+1} \mid E_i]$

1. By Fact 2, min-cut size $\geq k$, so degree $\geq k$
2. # vertices $= n - i$
3. # edges $\geq (n-i)k/2$
4. $\Pr[E_{i+1} \mid E_i] = 1 - \frac{k}{\#\text{ edges}} \geq 1 - \frac{k}{(n-i)k/2} = \frac{n-i-2}{n-i}$

Analysis

$$\begin{aligned} & \Pr[\textsc{RandMinCut} \text{ outputs } C] \\ = \; & \Pr[E_{n-2}] = \Pr[E_{n-2} \mid E_{n-1}] \cdots \Pr[E_2 \mid E_1] \cdot \Pr[E_1] \\ \geq \; & \left(\tfrac{n-2}{n}\right)\left(\tfrac{n-3}{n-1}\right)\left(\tfrac{n-4}{n-2}\right)\left(\tfrac{n-5}{n-3}\right) \cdots \left(\tfrac{4}{6}\right)\left(\tfrac{3}{5}\right)\left(\tfrac{2}{4}\right)\left(\tfrac{1}{3}\right) \\ = \; & \frac{2}{n(n-1)} \end{aligned}$$

Question

How can we increase the probability of returning a min-cut?

- Repeat $\frac{n(n-1)}{2} \ln n$ times and output the smallest cut.
- $\Pr[\text{fails to output } C] \leq \left(1 - \frac{2}{n(n-1)}\right)^{\frac{n(n-1)}{2} \ln n} \leq \frac{1}{n}$

(Almost) the End

- Next week: review basic probability
- Homework 1 to be posted by Fri, due Jan 30 (Wed).