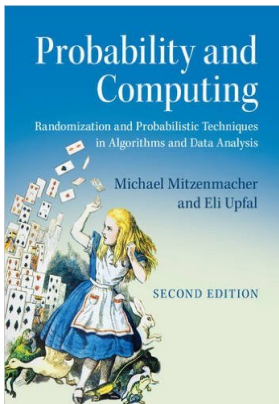


CS 1550/2540: Probabilistic Methods in Computer Science

Instructor: Eli Upfal, CIT 319.

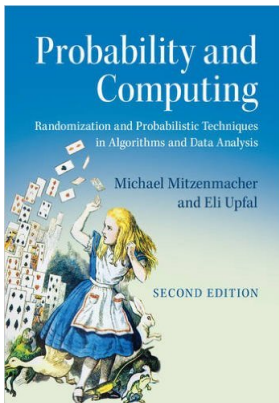
HTA: Daniel MA

TA's: Alice Wang, Joshua Kou



CS 1550/2540: Probabilistic Methods in Computer Science

Introduction + Chapter 1



Overview

- CSCI 1550/2540 is a **theory/math** course - analysis, theorems, proofs. No implementations.
- The course covers modern mathematics at the interface of probability theory and computation
- Formulates, and explains many of the great successes of computing, such as machine learning, cryptography, modern finance, computational biology, etc.
- The course focuses on methods, not applications.
- Goal: develop rigorous mathematical understanding of probabilistic and statistical methods.

This course is different from most courses in CS!!

Why Probabilistic Methods?

Almost any advance computing application today has some **randomization/statistical/machine learning** components:

- **Randomized algorithms** - random steps help!
 - Efficiency: Hashing, Quicksort, ...
 - Security and Privacy: Open key cryptography, one way function,...
 - Monte Carlo methods: scientific computing, finance, weather forecast,...
- **Probabilistic analysis of algorithms** - Theoretically "hard to solve" problems are often not that hard in practice.
- **Modeling data**
 - Statistical machine learning
 - Data mining
 - Recommendation systems

The "magic" of modern computing is that most probabilistic and statistical components are well hidden.

- A shopping site tells you that your credit card transaction is **secure**, not that it's **probabilistically secure**.
- A chatbot (LLM) gives you an answer. It doesn't specify the sequence of random steps that led to that answer, and may lead to another answer in a different execution.

Do you need to understand the probabilistic components of your computation?

Most software development work relies on "off-the-shelf", well tested applications, but someone must be able to understand the underlying theory, evaluate the quality of the results, and develop new applications.

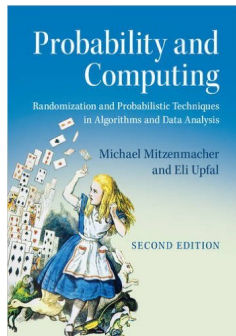
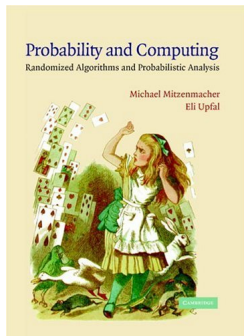
Course Details - Main Topics

- ① Basic randomized algorithms and probabilistic analysis of algorithms (and review of relevant probability theory concepts)
- ② Large deviation bounds: Chernoff and Hoeffding bounds
- ③ The probabilistic method
- ④ Martingale (in discrete space)
- ⑤ Sample complexity, statistical learning theory
- ⑥ Monte Carlo methods
- ⑦ Convergence of Monte Carlo Markov Chains
- ⑧ ...

This course emphasize rigorous mathematical approach, mathematical proofs, and analysis.

Course Details

- **Pre-requisite:** Discrete probability theory (first three chapters in the course textbook); mathematical maturity; interest in theory of algorithms
- **Course textbook:**



- **Follow:** Ed-discussion, Canvas, and the course website:
<https://cs.brown.edu/courses/csci1550/>

Course Work and Grading:

Homework Assignments:

- Weekly assignments (problem sets) are a major part of the course work.
- Submitting all assignments is required to take the final exam.
- Assignments submitted by their deadline will be graded for correctness and mathematical writing.
- Assignments' grades do not count toward the course grade, but at least half the midterm and final exams questions will be from the weekly assignments.

Final grade:

- For 1550 credit: 40% midterm, 60% final.
- For 1550 credit as a capstone course: 30% midterm, 40% final, 30% project.
- For 2540 credit: 30% midterm, 40% final, 30% project.

Project:

- Submit a 6-8 page research project (with possible appendices) on any application of probabilistic method in CS.
- 15-minute presentation on your project in class.
- A project can be done by 1 or 2 students. Two students who submit a joint project will receive the same grade on their project.
- Students who choose to prepare projects will interact with the instructor and the TA's on preparing their project throughout the course.
- If you plan to submit a project: (1) let the HTA know by Oct. 15; (2) Submit a title and one paragraph plan by Oct 25.

We treat you as adults...

- You don't need to attend class - but you cannot ask the instructor/TA's to repeat information given in class.
- Don't postpone the HW assignments to the last day - you cannot do it in one evening!
- The TA's will not do your HW assignments!
- Make good use of course's resources:
 - Course's slides
 - book
 - TA's hours
 - Ed-Discussion
- **Think! Challenge yourself!**

Questions?

Verifying Matrix Multiplication

Given three $n \times n$ matrices **A**, **B**, and **C** in a Boolean field, we want to verify

$$\mathbf{AB} = \mathbf{C}.$$

Standard method: Matrix multiplication - takes $\Theta(n^3)$ ($\Theta(n^{2.37})$) operations (multiplications).

Verifying Matrix Multiplication

Freivalds' Randomized algorithm (takes $\Theta(n^2)$ multiplications):

- 1 Chooses a random vector $\bar{r} = (r_1, r_2, \dots, r_n) \in \{0, 1\}^n$.
- 2 Compute $B\bar{r}$;
- 3 Compute $A(B\bar{r})$;
- 4 Computes $C\bar{r}$;
- 5 If $A(B\bar{r}) \neq C\bar{r}$ return $AB \neq C$, else return $AB = C$.

Theorem

If $AB \neq C$, and $\bar{r}$ is chosen uniformly at random from $\{0, 1\}^n$, then

$$\Pr(AB\bar{r} = C\bar{r}) \leq \frac{1}{2}.$$

If $AB = C$ the algorithm is always correct. If $AB \neq C$, the algorithm gives the correct answer with probability $\geq 1/2$

Proof:

Assume $\mathbf{D} = \mathbf{AB} - \mathbf{C} \neq \mathbf{0}$.

$\mathbf{AB}\bar{\mathbf{r}} = \mathbf{C}\bar{\mathbf{r}}$ implies that $\mathbf{D}\bar{\mathbf{r}} = \mathbf{0}$.

Since $\mathbf{D} \neq \mathbf{0}$ it has some non-zero entry; assume d_{11} .

For $\mathbf{D}\bar{\mathbf{r}} = \mathbf{0}$, it must be the case that

$$\sum_{j=1}^n d_{1j} r_j = 0,$$

or equivalently

$$r_1 = -\frac{\sum_{j=2}^n d_{1j} r_j}{d_{11}}. \quad (1)$$

Here we use $d_{11} \neq 0$.

Let $a_2, \dots, a_n$ be the values chosen at random for the variables $r_2, \dots, r_n$.

$$\Pr \left(r_1 = -\frac{\sum_{j=2}^n d_{1j} a_j}{d_{11}} \right) \leq 1/2. \quad (2)$$

Only one value of r_1 satisfies the equation, and we choose r_1 randomly from a set of at least 2 values.

Theorem

If $\mathbf{AB} \neq \mathbf{C}$, and $\bar{r}$ is chosen uniformly at random from $\{0, 1\}^n$, then

$$\Pr(\mathbf{AB}\bar{r} = \mathbf{C}\bar{r}) \leq \frac{1}{2}.$$

$\leq 1/2$ because it must hold on all non-zero rows of D .

Probability Space

Probability is always with respect to a probability space!

What is the probability that the sun will rise tomorrow?

Pierre-Simon de Laplace - 1749–1827.

Definition

A **probability space** has three components:

- 1 A **sample space** Ω , which is the set of all possible outcomes of the random process modeled by the probability space;
- 2 A family of sets $\mathcal{F}$ representing the allowable **events**, where each set in $\mathcal{F}$ is a subset of the sample space Ω ;
- 3 A **probability function** $\Pr : \mathcal{F} \rightarrow [0, 1]$ defining a measure.

In a **discrete** probability an element of Ω is a **simple** event, and $\mathcal{F} = 2^\Omega$.

Probability Function

Definition

A **probability function** is any function $\Pr : \mathcal{F} \rightarrow \mathbf{R}$ that satisfies the following conditions:

- 1 For any event E , $0 \leq \Pr(E) \leq 1$;
- 2 $\Pr(\Omega) = 1$;
- 3 For any finite or countably infinite sequence of pairwise mutually disjoint events $E_1, E_2, E_3, \dots$

$$\Pr \left(\bigcup_{i \geq 1} E_i \right) = \sum_{i \geq 1} \Pr(E_i).$$

In discrete sample space, the probability of an event is the sum of the probabilities of its simple events.

Lemma

Choosing $\vec{r} = (r_1, r_2, \dots, r_n) \in \{0, 1\}^n$ uniformly at random is equivalent to choosing each r_i independently and uniformly from $\{0, 1\}$.

Proof.

If each r_i is chosen independently and uniformly at random, each of the 2^n possible vectors $\vec{r}$ is chosen with probability 2^{-n} , giving the lemma. □

Proof:

Assume $\mathbf{D} = \mathbf{AB} - \mathbf{C} \neq \mathbf{0}$.

$\mathbf{AB}\bar{\mathbf{r}} = \mathbf{C}\bar{\mathbf{r}}$ implies that $\mathbf{D}\bar{\mathbf{r}} = \mathbf{0}$.

Since $\mathbf{D} \neq \mathbf{0}$ it has some non-zero entry; assume d_{11} .

For $\mathbf{D}\bar{\mathbf{r}} = \mathbf{0}$, it must be the case that

$$\sum_{j=1}^n d_{1j} r_j = 0,$$

or equivalently

$$r_1 = -\frac{\sum_{j=2}^n d_{1j} r_j}{d_{11}}. \quad (3)$$

Here we use $d_{11} \neq 0$.

Principle of Deferred Decision

Assume that we fixed $r_2, \dots, r_n$.

The RHS is already determined, the only variable is r_1 .

$$r_1 = -\frac{\sum_{j=2}^n d_{1j}r_j}{d_{11}}. \quad (4)$$

Probability that $r_1 = \text{RHS}$ is no more than $1/2$. (= $1/2$ in the Boolean field.)

Theorem

If $\mathbf{AB} \neq \mathbf{C}$, and $\bar{r}$ is chosen uniformly at random from $\{0, 1\}^n$, then

$$\Pr(\mathbf{AB}\bar{r} = \mathbf{C}\bar{r}) \leq \frac{1}{2}.$$

$\leq 1/2$ because it must hold on all non-zero rows of D .

More formally, summing over all collections of values $(x_2, x_3, x_4, \dots, x_n) \in \{0, 1\}^{n-1}$, we have

$$\begin{aligned}
 & \Pr(\mathbf{A}\mathbf{B}\bar{r} = \mathbf{C}\bar{r}) \\
 = & \sum_{(x_2, \dots, x_n) \in \{0, 1\}^{n-1}} \Pr((\mathbf{A}\mathbf{B}\bar{r} = \mathbf{C}\bar{r}) \cap ((r_2, \dots, r_n) = (x_2, \dots, x_n))) \\
 \leq & \sum_{(x_2, \dots, x_n) \in \{0, 1\}^{n-1}} \Pr\left(\left(r_1 = -\frac{\sum_{j=2}^n d_{1j}r_j}{d_{11}}\right) \cap ((r_2, \dots, r_n) = (x_2, \dots, x_n))\right) \\
 = & \sum_{(x_2, \dots, x_n) \in \{0, 1\}^{n-1}} \Pr\left(r_1 = -\frac{\sum_{j=2}^n d_{1j}r_j}{d_{11}}\right) \cdot \Pr((r_2, \dots, r_n) = (x_2, \dots, x_n)) \\
 \leq & \sum_{(x_2, \dots, x_n) \in \{0, 1\}^{n-1}} \frac{1}{2} \Pr((r_2, \dots, r_n) = (x_2, \dots, x_n)) \\
 = & \frac{1}{2}.
 \end{aligned}$$

Smaller Error Probability

The test has a one side error, repeated tests are independent.

- Run the test k times.
- Accept $AB = C$ if it passed all k tests.

Theorem

The probability of making a mistake is $\leq (1/2)^k$.

Proof.

We use: if A and B are independent events then

$$Pr(A \cap B) = Pr(A)Pr(B)$$



Deciding Polynomial Identity

Problem: Verify that $P(x) \equiv Q(x)$, or $H(x) \equiv P(x) - Q(x) \equiv 0$.

Example: Check if

$$(x+1)(x-2)(x+3)(x-4)(x+5)(x-6) \equiv x^6 - 7x^3 + 25.$$

(We use $\equiv$ for polynomial identities, $=$ for numerical equality.)

Randomized Algorithm: (Decide if $H(x) \equiv 0$)

- 1 Choose a random integer r in the range $[1, \dots, m]$.
- 2 If $H(r) = 0$ output CORRECT else output FALSE.

Theorem

Assume that $H(x) \not\equiv 0$, with degree $\leq d$. If r is chosen uniformly at random from $[1, \dots, m]$, then

$$\text{Prob}(H(r) = 0) \leq \frac{d}{m}.$$

WHY?

Multivariate Polynomial Identity

Example:

$$H(x, y, z, w) = x^5 y^3 (z^3 - w^4) + x^3 (z^5 - w^3) (x^2 y^4 - w^2) \stackrel{?}{=} 0$$

Degree of Multivariate Polynomial = the maximum sum of the degrees in each of its monomials.

The degree of $H(x, y, z, w)$ is 12.

Theorem (Schwartz–Zippel)

Assume that $H(x_1, x_2, \dots, x_n) \not\equiv 0$, with degree $\leq d$. $(r_1, r_2, \dots, r_n)$ are chosen uniformly and independently at random from a set S ($(r_1, r_2, \dots, r_n)$ is chosen uniformly at random from S^n), then

$$\text{Prob}(H(r_1, \dots, r_n) = 0) \leq \frac{d}{|S|}.$$

No efficient deterministic algorithm is known. (A polynomial with n variables and degree d can have $\binom{n+d}{d}$ monomials.)

Proof of Schwartz–Zippel Lemma

Proof. By induction on n . For $n = 1$, H has at most d roots, so

$$\Pr[H(r_1) = 0] \leq \frac{d}{|S|}.$$

For $n > 1$, H can be written as a polynomial in x_n :

$$H(x_1, \dots, x_n) = \sum_{i \geq 0} x_n^i f_i(x_1, \dots, x_{n-1}),$$

Let $k \leq d$ be the largest index for which $f_k \neq 0$.

Since the degree of f_k is $\leq d - k$, by the induction hypothesis

$$\Pr[f_k(r_1, \dots, r_{n-1}) = 0] \leq \frac{d - k}{|S|}.$$

If $f_k(r_1, \dots, r_{n-1}) \neq 0$, then $H(r_1, \dots, r_{n-1}, x_n)$ is a nonzero univariate polynomial of degree k .

$$\Pr[H(r_1, \dots, r_n) = 0 \mid f_k(r_1, \dots, r_{n-1}) \neq 0] \leq \frac{k}{|S|}.$$

Conditional Probabilities

Definition

The **conditional probability** that event E_1 occurs given that event E_2 occurs is

$$\Pr(E_1 \mid E_2) = \frac{\Pr(E_1 \cap E_2)}{\Pr(E_2)}.$$

By conditioning on E_2 we restrict the sample space to the set E_2 . Thus we are interested in $\Pr(E_1 \cap E_2)$ “normalized” by $\Pr(E_2)$.

A condition E_2 defines a new sample space, with a new probability function $P(\cdot \mid E_2)$

Let $E_1, \dots, E_k$ be a collection of events such that $\cup_{i=1}^k E_i = \Omega$ and $E_i \cap E_j = \emptyset$ for all $i \neq j$,

$$\sum_{i=1}^k \Pr(E_i \mid B) = \sum_{i=1}^k \frac{\Pr(E_i \cap B)}{\Pr(B)} = 1.$$

$$H(x_1, \dots, x_n) = \sum_{i \geq 0} x_n^i f_i(x_1, \dots, x_{n-1}),$$

$$\Pr[f_k(r_1, \dots, r_{n-1}) = 0] \leq \frac{d-k}{|S|}.$$

$$\Pr[H(r_1, \dots, r_n) = 0 \mid f_k(r_1, \dots, r_{n-1}) \neq 0] \leq \frac{k}{|S|}.$$

$$\begin{aligned} \Pr[H(r_1, \dots, r_n) = 0] &= \Pr[H(r_1, \dots, r_n) = 0 \mid f_k(r_1, \dots, r_{n-1}) \neq 0] \\ &\quad + \Pr[f_k(r_1, \dots, r_{n-1}) = 0] \\ &\leq \frac{d-k}{|S|} + \frac{k}{|S|} = \frac{d}{|S|}. \end{aligned}$$

We use

$$\begin{aligned} \Pr(A) &= \Pr(A \cap B) + \Pr(A \cap \bar{B}) \\ &= \Pr(A \mid B)\Pr(B) + \Pr(A \mid \bar{B})\Pr(\bar{B}) \\ &\leq \Pr(A \mid B) + \Pr(\bar{B}) \end{aligned}$$

Reducing the Error Probability

Theorem (Schwartz–Zippel)

If d is the degree of $H(x_1, x_2, \dots, x_n)$, and $(r_1, r_2, \dots, r_n)$ is chosen uniformly and independently at random from a set S , then

$$\text{Prob}(H(r_1, \dots, r_n) = 0 \mid H(x_1, \dots, x_n) \neq 0) \leq \frac{d}{|S|}.$$

Run the algorithm ℓ times with independent choices of $r_1, \dots, r_n$ for each run. Output $H(x_1, \dots, x_n) \equiv 0$ iff all runs gave $H(r_1, \dots, r_n) = 0$.

$$\text{Pr}(H(x_1, \dots, x_n) \neq 0 \mid H(r_1, \dots, r_n) = 0 \text{ in } \ell \text{ inde. runs}) \leq \left(\frac{d}{|S|}\right)^\ell$$

Independent Events

Definition

Two events E and F are **independent** if and only if

$$\Pr(E \cap F) = \Pr(E) \cdot \Pr(F).$$

More generally, events $E_1, E_2, \dots, E_k$ are mutually independent if and only if for **any** subset $I \subseteq [1, k]$,

$$\Pr\left(\bigcap_{i \in I} E_i\right) = \prod_{i \in I} \Pr(E_i).$$

If E and F are independent then the probability of E does not depend on F .

$$\Pr(E \mid F) = \frac{\Pr(E \cap F)}{\Pr(F)} = \frac{\Pr(E) \cdot \Pr(F)}{\Pr(F)} = \Pr(E)$$

Independent event are not necessarily physically independent. Dice roll ≤ 4 , roll gives an even number.

Min-Cut

A graph $G = (V, E)$, V -set of vertices, E set of edges.

A **Min-Cut set** - A minimum set of edges that disconnects the graph.



Min-Cut

A graph $G = (V, E)$, V -set of vertices, E set of edges.

A **Min-Cut set** - A minimum set of edges that disconnects the graph.

Fundamental computation problem in transportation, network reliability, arbitrage, image segmentation,...

Has a deterministic polynomial time solution.

Any ideas?

Min-Cut

A graph $G = (V, E)$, V -set of vertices, E set of edges.

A **Min-Cut set** - A minimum set of edges that disconnects the graph.

Fundamental computation problem in transportation, network reliability, arbitrage, image segmentation,...

Has a deterministic polynomial time solution: Run a max flow algorithm ($O(|V|^2)$ or $O(|V| \cdot |E|)$ complexity) between all pairs of nodes ($\binom{|V|}{2}$).

Instead, we present and analyze a simple **randomized algorithm** with a better run-time.

Min-Cut Algorithm

Input: An n -node graph G .

Output: A minimal set of edges that disconnects the graph.

① Repeat $n - 2$ times:

- ① Pick an edge uniformly at random.
- ② Contract the two vertices connected by that edge, eliminate all edges connecting the two vertices.

② Output the set of edges connecting the two remaining vertices.

How good is this algorithm?

Does it always give a correct result?

Min-Cut Algorithm

Input: An n -node graph G .

Output: A minimal set of edges that disconnects the graph.

① Repeat $n - 2$ times:

- ① Pick an edge uniformly at random.
- ② Contract the two vertices connected by that edge, eliminate all edges connecting the two vertices.

② Output the set of edges connecting the two remaining vertices.

Theorem

- ① The algorithm outputs a min-cut edge-set with probability $\geq \frac{2}{n(n-1)}$.
- ② The smallest set output in $O(n^2 \log n)$ iterations of the algorithm gives a correct answer with probability $1 - 1/n^2$.

Min-Cut Algorithm

Input: An n -node graph G .

Output: A minimal set of edges that disconnects the graph.

① Repeat $n - 2$ times:

- ① Pick an edge uniformly at random.
- ② Contract the two vertices connected by that edge, eliminate all edges connecting the two vertices.

② Output the set of edges connecting the two remaining vertices.

Theorem

The algorithm outputs a min-cut edge-set with probability

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

Analysis of the Algorithm

Assume that the graph has a min-cut set of k edges.

We compute the probability of finding one such set C .

$$\Pr(\text{Alg. returns any minimal cut set}) \geq \Pr(\text{Alg. returns } C)$$

Two parts proof:

- **Deterministic analysis part:**

Lemma

If no edge of C was contracted, the algorithms outputs C .

- **Probabilistic analysis part:**

Lemma

$$\Pr(\text{no edge of } C \text{ is contracted}) \geq \frac{2}{n(n-1)}.$$

Deterministic part:

Lemma (Correctness of a step)

If no edge of C was contracted, no edge of C was eliminated.

Proof.

Let X and Y be the two set of vertices cut by C .

If the contracting edge connects two vertices in X (res. Y), then all its parallel edges also connect vertices in X (res. Y). $\square$

Corollary (Correctness of a run)

If the algorithm terminates before contracting any edge of C , the algorithm gives a correct answer.

Lemma (One side error)

*Vertex contraction does not reduce the size of the min-cut set.
Every cut set in the new graph is a cut set in the original graph.*

Probabilistic Analysis:

Lemma

$$Pr(\text{no edge of } C \text{ is contracted}) \geq \frac{2}{n(n-1)}.$$

What's the probability space? It's a product of spaces corresponding to rounds of the loop.

The probability space at step i depends on the probability space at step $i - 1$.

Let $E_i =$ "the edge contracted in iteration i is not in C ."

Let $F_i = \bigcap_{j=1}^i E_j =$ "no edge of C was contracted in the first i iterations".

Since the minimum cut-set has k edges, all vertices have degree $\geq k$, and the graph has $\geq nk/2$ edges.

There are at least $nk/2$ edges in the graph, k edges are in C .

Thus, $Pr(E_1) = Pr(F_1) \geq 1 - \frac{2k}{nk} = 1 - \frac{2}{n}$.

Conditioning on E_1 , after the first vertex contraction we are left with an $n - 1$ node graph, with minimum cut set, and minimum degree $\geq k$. The new graph has at least $k(n - 1)/2$ edges, thus $Pr(E_2 \mid F_1) \geq 1 - \frac{k}{k(n-1)/2} \geq 1 - \frac{2}{n-1}$.

Similarly, $Pr(E_i \mid F_{i-1}) \geq 1 - \frac{k}{k(n-i+1)/2} = 1 - \frac{2}{n-i+1}$.

How do we combine $Pr(E_i \mid F_{i-1})$'s to compute F_{n-1} ?

Useful identities:

$$Pr(A \mid B) = \frac{Pr(A \cap B)}{Pr(B)}$$

$$Pr(A \cap B) = Pr(A \mid B)Pr(B)$$

$$Pr(A \cap B \cap C) = Pr(A \mid B \cap C)Pr(B \cap C)$$

$$= Pr(A \mid B \cap C)Pr(B \mid C)Pr(C)$$

Let $E_1, \dots, E_n$ be a sequence of events. Let $F_i = \bigcap_{j=1}^i E_j$

$$Pr(F_n) = Pr(E_n \mid F_{n-1})Pr(F_{n-1}) =$$

$$Pr(E_n \mid F_{n-1})Pr(E_{n-1} \mid F_{n-2}) \dots Pr(E_2 \mid F_1)Pr(F_1)$$

We need to compute

$$Pr(F_{n-2}) = Pr(\cap_{j=1}^{n-2} E_j)$$

We have

$$Pr(E_1) = Pr(F_1) \geq 1 - \frac{2k}{nk} = 1 - \frac{2}{n}$$

and

$$Pr(E_i \mid F_{i-1}) \geq 1 - \frac{k}{k(n-i+1)/2} = 1 - \frac{2}{n-i+1}.$$

$$Pr(F_{n-2}) = Pr(E_{n-2} \cap F_{n-3}) = Pr(E_{n-2} \mid F_{n-3})Pr(F_{n-3}) =$$

$$Pr(E_{n-2} \mid F_{n-3})Pr(E_{n-3} \mid F_{n-4})....Pr(E_2 \mid F_1)Pr(F_1) =$$

$$Pr(F_1) \prod_{j=2}^{n-2} Pr(E_j \mid F_{j-1})$$

The probability that the algorithm computes the minimum cut-set is

$$\begin{aligned} Pr(F_{n-2}) &= Pr(\cap_{j=1}^{n-2} E_j) = Pr(F_1) \prod_{j=2}^{n-2} Pr(E_j \mid F_{j-1}) \\ &\geq \prod_{i=1}^{n-2} \left(1 - \frac{2}{n-i+1}\right) = \prod_{i=1}^{n-2} \left(\frac{n-i-1}{n-i+1}\right) \\ &= \left(\frac{n-2}{n}\right) \left(\frac{n-3}{n-1}\right) \left(\frac{n-4}{n-2}\right) \cdots \\ &\quad \frac{2}{n(n-1)}. \end{aligned}$$

Theorem

The algorithm outputs a min-cut edge-set with probability
 $\geq \frac{2}{n(n-1)}.$

Theorem

Assume that we run the randomized min-cut algorithm $n(n-1)\log n$ times and output the minimum size cut-set found in all the iterations. The probability that the output is not a min-cut set is bounded by $\frac{1}{n^2}$.

Proof.

The algorithm has a **one side error**: the output is **never smaller** than the min-cut value.

The probability that **C** is **not** the output of any of the $n(n-1)\log n$ runs is (using independence)

$$\leq \left(1 - \frac{2}{n(n-1)}\right)^{n(n-1)\log n} \leq e^{-2\log n} = \frac{1}{n^2}.$$



$$\left(1 - \frac{2}{n(n-1)}\right)^{n(n-1) \log n} \leq e^{-2 \log n} = \frac{1}{n^2}.$$

The Taylor series expansion of e^{-x} gives

$$e^{-x} = 1 - x + \frac{x^2}{2!} - \dots$$

Thus, for $x < 1$,

$$1 - x \leq e^{-x}.$$

Theorem

- ① *The algorithm outputs a min-cut edge set with probability $\geq \frac{2}{n(n-1)}$.*
- ② *The smallest output in $O(n^2 \log n)$ iterations of the algorithm gives a correct answer with probability $1 - 1/n^2$.*