**CMPUT 498/501: Advanced Algorithms** **Fall 2025**

# Lecture 6 (Sep 22 2025): Randomized Algrithms

*Lecturer: Mohammad R. Salavatipour* *Scribe: Lyndon Hallett, Haoxin Sang*

## 6.1 Introduction

In the next few weeks we will talk about randomized algorithms specifically. There are verious reasons why randomized algorithms are useful:

- **Speed:** Are often faster than deterministic algorithms, even if their worst-case time is worse. An Example of this is QuickSort is the fastest sorting algorithm in practice even though its worste case running time is quadratic (has expected $O(n \log n)$ time vs).
- **Simplicity:** Are typically eeasier to implement (and faster as a result).
- **Necessity:** For many problems (e.g., online algorithms) these are the only algorithms that can solve the problem as it is the only way we can beat an adversary. Often it can be proved that any deterministic algorithm will be bad. There are other situations that we don't know of any deterministic algorithm at all, e.g. checking if a multivariate polynomial is identical to zero is hard deterministically, but easy with randomness.

We should note that study of Randomized algorithms differs from average-case analysis. In average-case analysis, we assume some distribution on the input (e.g., uniformly random) and study the performance of an algorithm under that distribution. However, when designing randomized algorithms, we do not place such assumptions (and is working for all settings).

**Types of Randomized Algorithms:**

- **Las Vegas algorithms:** Always find the correct answer, but their running time is a random function that can vary. We then compute the expected running time. An example is Randomized QuickSort, which runs in expected $O(n \log n)$ time, although the worst case is still $O(n^2)$.
- **Monte Carlo algorithms:** These may not always find the correct answer. We show that they find the correct answer with some probability $0 < \mathbb{P} < 1$.

Note that for Monte Carlo algorithms, we can always increase the probability by running multiple independent copies. The running time might also be bounded in expectation. Additionally, the two types of algorithms are not strictly disjoint categories. For example, a Las Vegas algorithm can be converted to a Monte Carlo algorithm by running it for a set time.

**Paradigms for Randomized Algorithms:**

- Avoiding adversarial inputs: randomly reorder the input.
- Fingerprinting & hashing: use smaller (randomly generated) fingerprints to compare large objects.

- Random sampling: used in approximate counting of large sets, median finding, etc.
- Online algorithms / load balancing: spread the load among resources randomly to reduce cost and/or time.
- Symmetry breaking: helpful for protocols in networks and avoiding deadlock in distributed systems.
- Probabilistic method: A very powerful technique in combinatorics. To show the existence, one can show the probability of a carefully chosen one having the desired property we want is greater than zero (e.g., Lovász Local Lemma). There are methods to turn these into algorithms.

## 6.2 Probability Background

- Probability space $S$: A set $S$ of possible outcomes of a (usually random) process.
  - E.g., rolling a die: $S = \{1, 2, 3, 4, 5, 6\}$.
- Event $\mathcal{E}$: Any subset of the probability space ($\mathcal{E} \subseteq S$).
- Probability distribution: A function $\Pr : S \mapsto [0, 1]$ such that $\sum_{e \in S} \Pr(e) = 1$.
- Random variable $X$: A function from $S$ to reals or integers. An example is the space of rolling a pair of dice. Define $X$ to be the random variable that is 0/1 depending on whether the parity of the sum of the two faces is even/odd.
- Expected value: The expected value of $X$ is defined as

$$\mathbb{E}[X] = \mu(X) = \sum_x x \Pr[X = x].$$

- Variance: The variance of $X$ is defined as

$$\begin{aligned} \mathrm{Var}(X) &= \mathbb{E}[(X - \mu(X))^2] \\ &= \mathbb{E}[X^2] - 2\mathbb{E}[X \cdot \mathbb{E}[X]] + \mathbb{E}[X]^2 = \mathbb{E}[X^2] - \mathbb{E}[X]^2. \end{aligned}$$

- Standard deviation: The standard deviation of $X$ is defined as

$$\sigma(X) = \sqrt{\mathrm{Var}(X)}.$$

- Rules for $\mathbb{E}[\cdot]$ and $\mathrm{Var}(\cdot)$:
  - Linearity of expectation: $\mathbb{E}[\sum_i \alpha_i X_i] = \sum_i \alpha_i \mathbb{E}[X_i]$.
  - If $X$ and $Y$ are independent: $\mathbb{E}[X \cdot Y] = \mathbb{E}[X] \cdot \mathbb{E}[Y]$.

## 6.3 Simple Deviation Bounds

**Theorem 1 (Markov's Inequality)** *Let $X$ be a random variable that takes non-negative values. For all $t > 0$,*

$$\Pr(X \geq t) \leq \frac{\mathbb{E}[X]}{t},$$

*or equivalently,*

$$\Pr\left(X \geq t\mathbb{E}[X]\right) \leq \frac{1}{t}.$$

**Proof.** Define an indicator random variable

$$I = \begin{cases} 1, & X \geq t \\ 0, & otherwise \end{cases}$$

Since $X \geq 0$, then $I \leq \frac{X}{t}$. Therefore,

$$\mathbb{E}[I] = \Pr[I = 1] = \Pr[X \geq t] \leq \mathbb{E}\Big[\frac{X}{t}\Big] = \frac{1}{t}\mathbb{E}[X].$$

■

**Theorem 2 (Chebyshev's Inequality)** *For any random variable $X$ and any $\lambda > 0$,*

$$\Pr(|X - \mathbb{E}[X]| \geq \lambda) \leq \frac{Var[X]}{\lambda^2},$$

*or equivalently,*

$$\Pr(|X - \mathbb{E}[X]| \geq \lambda\mathbb{E}[X]) \leq \frac{Var[X]}{\lambda^2\mathbb{E}[X]^2}.$$

**Proof.** First, notice that:

$$\begin{aligned} \Pr(|X - \mathbb{E}[X]| \geq \lambda) &= \Pr((X - \mathbb{E}[X])^2 \geq \lambda^2) \\ &\leq \frac{\mathbb{E}[(X - \mathbb{E}[X])^2]}{\lambda^2} = \frac{\mathrm{Var}[X]}{\lambda^2}, \end{aligned}$$

where the inequality holds by Markov's inequality. ■

Use of Markov's inequality is called the first moment method, and using Chebyshev's inequality is called the second moment method.

## 6.4 Karger's Randomized Algorithm for Min-Cut

**Definition 1 (Cut)** *Given an undirected graph $G = (V, E)$, a cut is a partition of $V$ into two non-empty subsets $S$ and $\overline{S} = V - S$. The size of the cut, $\delta(S, \overline{S})$, is the number of edges between $S$ and $\overline{S}$. If the graph has edge weights, the size of the cut is the sum of the weights of the edges crossing the cut.*

Minimum Cut Problem: Find a cut $(S, \overline{S})$ of minimum size. The size of the minimum cut is denoted by

$$\lambda_G = \min_{\emptyset \subsetneq S \subsetneq V} |\delta(S, \bar{S})|.$$

One can try to use Max-Flow-Min-Cut theorem to solve Minimum Cut. We can try each pair of vertices $(s, t)$, as source sink nodes respectively, and find a Min-$s, t$-cut.

If $F(m, n)$ be the time complexity of Max-Flow in graphs with $m$ edges ($|E| = m$) and $n$ vertices ($|V| = n$), then the running time of this approach is $O(n^2 \cdot F(m, n))$.

**Remark 1** This can be improved to $O(n \cdot F(m, n))$ by fixing one vertex $s$ and computing max-flows from $s$ to all other vertices $t$.

**Remark 2** The time complexity for the Max-Flow problem has improved over a long line of research to almost linear time $O(m^{1+o(1)})$ [CKL+23].

We now present Karger's randomized algorithm for the min-cut problem, which runs in $\tilde{O}(n^2)$ time [KS93].

**Definition 2 (Contraction)** *Given an undirected graph $G = (V, E)$, a contraction of an edge $e = uv \in E$ gives a multigraph $G' = G/e$, where we contract nodes $u, v$ into a single node in $G'$. All of the edges in between $u$ and $v$ are deleted.*

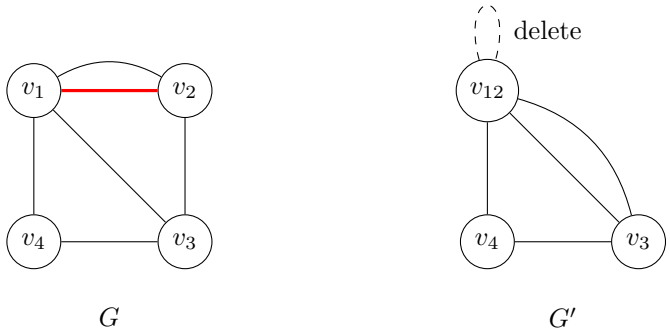


Figure 6.1: Contraction of edge $e = v_1v_2$ in graph $G$.

To represent the multigraph, we can keep the multiplicity of edges between any two nodes.

**Lemma 1** *For any edge $e \in G$, any cut in $G/e$ corresponds to a cut in $G$ of the same size.*

**Proof.** Consider any edge $e = uv \in G$ and suppose it is contracted into $v'$ in $G'$. Any cut $(S', \overline{S'})$ in $G'$ has $v'$ either in $S'$ or $\bar{S}'$. W.l.o.g., assume $v' \in S'$. Now consider the cut $(S, \overline{S})$ in $G$ where $S = (S' - \{v'\}) \cup \{u, v\}$. The edges crossing the cut remain the same, so the size of the cut is unchanged. That is, $\delta(S, \overline{S}) = \delta(S', \overline{S'})$. ■

**Corollary 1** *For any edge $e \in G$, $\lambda_{G/e} \geq \lambda_G$.*

The above lemma and corollary suggest algorithm 1.

**Algorithm 1** SIMPLE_MIN_CUT Algorithm

**Input**: Undirected graph $G = (V, E)$.
**Output**: A set of edges corresponding to a minimum cut $(S, S')$ of size $\lambda_G$.

```
G_0 ← G
i ← 0
while G_i has > 2 vertices do
    Pick a random edge e ∈ G_i
    G_{i+1} ← G_i/e
    i ← i + 1
end while
return edges of G_i
```

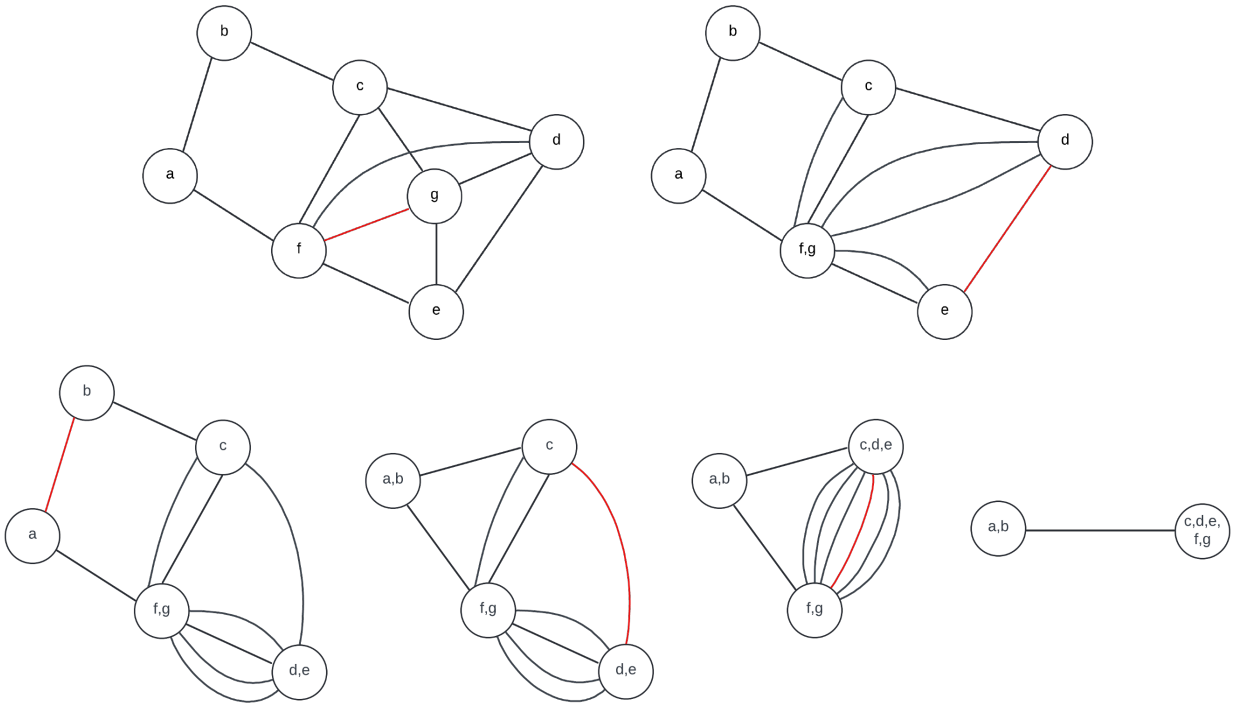


Figure 6.2: An example execution of Algorithm 1. In each step (following top-left to the bottom-right), an randomly selected edge is contracted (highlighted in red).

We now discuss the time complexity of Algorithm 1. Note that the algorithm will perform $n-2$ iterations of the while loop, until we have two components remaining. Each contraction step can be done in $O(n)$ time, as there are at most $O(n)$ edges that can change. Therefore, the total running time is $O(n^2)$.

Applying Lemma 1 inductively, along with Corollary 1, implies the following statement.

**Corollary 2** *Every cut in $G_i$, for $0 \leq i \leq n-1$, corresponds to a cut of the same size in $G$. Furthermore, $\lambda_{G_i} \geq \lambda_G$.*

It is also a fact for every graph $G$, if $\lambda_G = k$ then $m \geq \frac{kn}{2}$. Using this fact, we obtain the following.

**Corollary 3** $|E(G_i)| \geq \frac{(n-i)k}{2}$.

Now suppose $C \subseteq E$ is a fixed min-cut of $G$. Clearly, the probability that the algorithm returns $C$ is the same probability that none of the edges of $C$ are contracted. Let $|C| = k$.

Let $\varepsilon_i$ be the event that we didn't pick any edge of $C$ in iteration $i$. Therefore, since $G$ has at least $\frac{kn}{2}$ edges, then

$$\Pr(\varepsilon_1) \geq 1 - \frac{k}{\frac{nk}{2}} = 1 - \frac{2}{n}.$$

For general $i$, we have

$$\Pr\left(\varepsilon_1 | \bigcap_{j=1}^{i-1} \varepsilon_j\right) \geq 1 - \frac{2}{n-i+1}.$$

Let $\varepsilon$ be the event in which no edge of $C$ is ever selected. Since the events are independent, we have

$$\begin{aligned}\Pr(\varepsilon) &= \prod_{i=1}^{n-2} \Pr\left(\varepsilon_1 | \bigcap_{j=1}^{i-1} \varepsilon_j\right) \geq \prod_{i=1}^{n-2} \left(1 - \frac{2}{n-i+1}\right) \\ &= \left(\frac{n-2}{n}\right)\left(\frac{n-3}{n-1}\right)\left(\frac{n-4}{n-2}\right)\cdots\left(\frac{2}{4}\right)\left(\frac{1}{3}\right) = \frac{2}{n(n-1)}.\end{aligned}$$

Therefore, the probability of failure is at most $1 - \frac{2}{n(n-1)}$.

If we repeat the algorithm $t$ times and pick the smallest cut, the probability of failure is at most $\left(1 - \frac{2}{n(n-1)}\right)^t \leq \left(1 - \frac{2}{n^2}\right)^t$. Recall the well-known inequality that holds $\forall c$:

$$\left(1 + \frac{c}{x}\right)^x \leq e^c. \tag{6.1}$$

Using Eq. (6.1) and $t = n^2 \log n$, the probability of failure is at most $e^{-2\log n} = O(n^{-2})$.

Since we run the algorithm $O(n^2 \log n)$ times and each run is $O(n^2)$, the total runtime for this algorithm is $O(n^4 \log n)$. While there does exist more efficient algorithms, one advantage of this one is that it is fairly simple, and shows how randomization can help produce more efficient algorithms.

## 6.5 Coupon Collector Example

We illustrate previously introduced deviation bounds through an example. Consider the Coupon Collector's Problem. We have a deck of $n$ cards. Each round, we randomly pick a card, note it's value, and put it back in the same place. We want to determine the expected number of rounds before each card is drawn at least once. We frame the problem in terms of coupons instead of cards. Let $X$ be the number of rounds before all coupon types are collected. If $X_i$ denotes the number of rounds until the next new coupon is collected when $i$ coupons have been collected already, then

$$\mathbb{E}[X] = \sum_{i=0}^{n-1} \mathbb{E}[X_i].$$

Let $p_i = \frac{n-i}{n}$ be the probability of drawing a new coupon when $i$ coupons have already been seen. Since $X_i$ follows the geometric distribution[1], the mean is given by

$$\mathbb{E}[X_i] = \frac{1}{p_i} = \frac{n}{n-i}. \tag{6.2}$$

By Eq. (6.2), we have

$$\mathbb{E}[X] = \sum_{i=0}^{n-1} \frac{n}{n-i} = n \sum_{i=1}^{n} \frac{1}{i} = nH_n = n \ln n + \Theta(n),$$

[1]The probability distribution of the number $X$ of Bernoulli trials needed to get one success.

where $H_n$ denotes the $n$-th Harmonic number.

Using Markov's inequality, we have that

$$\Pr(X \geq 2\mathbb{E}[X]) = \Pr(X \geq 2nH_n) \leq \frac{1}{2}.$$

This means that the probability that we need twice as many rounds as expected to collect all coupons is at most $\frac{1}{2}$. A better bound can be achieved using Chebyshev's inequality. Notice that since $X_i$ is a geometric random variable, we have that $\text{Var}[X_i] = \frac{1-p_i}{p_i^2} \leq \frac{1}{p_i^2}$. Therefore, since $X_i$, $\forall\ 0 \leq i \leq n-1$, are independent random variables, we have

$$\text{Var}[X] = \sum_{i=0}^{n-1} \text{Var}[X_i] \leq \sum_{i=0}^{n-1} \Big(\frac{n}{n-i}\Big)^2 = n^2 \sum_{i=1}^{n} \frac{1}{i^2} \leq \frac{n^2\pi^2}{6}.$$

Therefore, using Chebyshev's inequality, we have

$$\Pr((X - \mathbb{E}[X]) \geq nH_n) \leq \frac{O(n^2)}{n^2 \ln^2 n} = O\Big(\frac{1}{\ln^2 n}\Big).$$

## 6.6 Chernoff-Hoeffding's Bound

Chernoff-Hoeffding's bound is one of the strongest tail bounds for independent random variables, and perhaps the most useful. We provide the bound as follows.

**Theorem 3** *Assume $X_1, \ldots, X_n$ are independent random variables that take values in [0,1]. Then for $X = \sum_{i=1}^{n} X_i$ and $\mu = \mathbb{E}[X]$, we have the following:*

1. *For any $0 < \delta$:* $\Pr(X \geq (1+\delta)\mu) \leq \Big(\frac{e^\delta}{(1+\delta)^{(1+\delta)}}\Big)^\mu$,
2. *For any $0 \leq \delta \leq 1$:* $\Pr(x \geq (1+\delta)\mu) \leq e^{-\mu\delta^2/3}$,
3. *For any $r \geq 6\mu$:* $\Pr(X \geq r) \leq 2^{-r}$.

We provide a proof of statement 1 in Theorem 3, to provide further details regarding it's derivation.

**Proof.** Suppose for ease that $X_1, \ldots, X_n$ are Bernoulli random variables with $X_i \in \{0, 1\}$ and $\mathbb{E}[X_i] = \mu$, $\forall\ 1 \leq i \leq n$. We can remove this assumption and prove the claim for any independent random variables that take values in $[0, 1]$.

Note for all $t > 0$,

$$\begin{aligned}\Pr(X \geq (1+\delta)\mu) &= \Pr(tX \geq t(1+\delta)\mu) \\ &= \Pr(e^{tX} \geq e^{t(1+\delta)\mu}).\end{aligned}$$

Using Markov's inequality, we have

$$\Pr(e^{tX} \geq e^{t(1+\delta)\mu}) \leq \frac{\mathbb{E}[e^{tX}]}{e^{t(1+\delta)\mu}} = \frac{\mathbb{E}[\prod_i e^{tX_i}]}{e^{t(1+\delta)\mu}} = \frac{\prod_i \mathbb{E}[e^{tX_i}]}{e^{t(1+\delta)\mu}},$$

where the first equality holds because the random variables are independent. Note that since we have Bernoulli random variables,

$$e^{tX_i} = \begin{cases} e^t, & w.p.\ \mu_i, \\ 1, & w.p.\ 1-\mu_i. \end{cases}$$

Also note that $1 + x \leq e^x$, $\forall\ x \geq 0$. Therefore, we have that

$$\mathbb{E}[e^{tX_i}] = e^t\mu_i + (1-\mu_i) = 1 + \mu_i(e^t - 1) \leq e^{\mu_i(e^t-1)}.$$

Altogether, we have the following:

$$\begin{aligned} \Pr(X \geq (1+\delta)\mu) &\leq \frac{\prod_i (e^{\mu_i(e^t-1)}}{e^{t(1+\delta)\mu}} \\ &= \frac{e^{(e^t-1)\sum_i \mu_i}}{e^{t(1+\delta)\mu}} = \frac{e^{(e^t-1)\mu}}{e^{t(1+\delta)\mu}} \end{aligned}$$

Letting $t = \ln(1+\delta)$, we obtain

$$\Pr(X \geq (1+\delta)\mu) \leq \Big(\frac{e^\delta}{(1+\delta)^{(1+\delta)}}\Big)^\mu,$$

as desired. ■

We have the following corollary, which implies that the Chernoff-Hoeffding's bounds are double sided.

**Corollary 4** *For $0 < \delta < 1$:*

$$\Pr(|X - \mu| \geq \delta\mu) \leq 2e^{-\mu\delta^2/3}.$$

Corollary 4 implies that with high probability, $X \sim (1 \pm \delta)\mu$.

We finish this section with applying the deviation bounds presented in the lecture on an example.

Consider randomly flipping $n$ unbiased coins. Let $X_i = 1$ if and only if coin $i$ is a head, and $X = \sum_i X_i$. Note that $\mu = \mathbb{E}[X] = \frac{n}{2}$ and $\mathrm{Var}[X] = \frac{n}{4}$.

- Using Markov's inequality, letting $\lambda = \mu$, we have

$$\Pr(X > \lambda + \mu) \leq \frac{1}{2}.$$

- Using Chebyshev's inequality, letting $\lambda = \sqrt{n}$, we have

$$\Pr(X > \mu + \lambda) = \frac{\frac{n}{4}}{\lambda^2} \leq \frac{1}{4}.$$

- Using Chernoff bounds, letting $\lambda = \sqrt{3n\ln n}$, i.e., $\frac{\lambda^2}{3\mu} = 2\ln n$, we have

$$\Pr(X \geq \mu + \lambda) = \Pr(X \geq \mu(1 + \frac{\lambda}{\mu})) \leq e^{\frac{-\lambda^2}{3\mu}} = e^{-2\ln n} = \frac{1}{n^2}.$$

Therefore, using a Chernoff bound gives us a stronger tail bound than the simpler deviation bounds.

## 6.7 References

`https://en.wikipedia.org/wiki/Las_Vegas_algorithm`

[CKL+23] Chen et al. "Maximum Flow and Minimum-Cost Flow in Almost-Linear Time" (2023). URL: `https://arxiv.org/pdf/2203.00671`

[KS93] David R. Karger and Clifford Stein. "An $\tilde{O}(n^2)$ algorithm for minimum cuts". In STOC, 1993.