CS590R: Randomized Algorithms and Probabilistic Techniques in CS

Gopal Pandurangan

gopal@cs.purdue.edu

Office: CS128

webpage: www.cs.purdue.edu/homes/gopal/cs590-2004/

newsgroup: purdue.class.cs590r
“It is remarkable that this science, which originated in the consideration of games and chances, should have become the most important object of human knowledge... The most important questions of life are, for the most part, really only problems of probability”

Probability and Computing

• Randomized algorithms - random steps help!

• Probabilistic analysis of algorithms - Why hard problems are sometimes easy to solve in practice?

• Probabilistic deduction, statistical inference, machine learning...

Applications: Communication networks; Cryptography; Search engines; Fast data structures; Scheduling; Optimization algorithms; Simulation and Modeling, AI reasoning; Learning; Bioinformatics, Quantum Computing, Complexity Theory ...
Topics

- Basic (discrete) probability theory, moments, basic bounds.
- Randomized algorithms.
- Probabilistic analysis.
- Graph Algorithms.
- Combinatorial Optimization.
- Random walks - Markov chains.
- The Probabilistic Method.
- Online Algorithms.
- Lower bounds on Randomized Algorithms.
• Distributed Algorithms.

• .....
References

• Probabilistic Analysis and Randomized Algorithms by M. Mitzenmacher and E. Upfal. (Required – Available from Copymat, Chauncey Mall)

• Randomized Algorithms by R. Motwani and P. Raghavan. (Recommended)

• A First Course in Probability by S. Ross.

• Probability and Random Processes by G. Grimmett and D. Stirzaker.

The last three books are on reserve in the Math Science library.
Grading

Homeworks:

• 5 or 6 assignments.

• Individually written (in Latex) (some of them will be non-collaborative.)

• Concise and correct proofs.

• Work must be submitted on time.

Paper Presentation.

Class Participation.

**Academic Dishonesty policy**: All submitted work should be on your own. Copying or using other people’s work (including from the Web) or using unauthorized material (the reference books listed above are the only authorized material allowed) will result in $-MAX$. 
points, where $MAX$ is the maximum possible number of points for that assignment/problems/quiz. Repeat offense will result in getting a failure grade in the course and reporting to the Dean of students.
Verifying Polynomial Identities

**Problem:** Verify that \( P(x) \equiv Q(x) \).

**Example:** Check if

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

(Will use \( \equiv \) for polynomial identities, \( = \) for numerical equality.)
Deterministic solution:

\[ H(x) \equiv (x + 1)(x - 2)(x + 3)(x - 4)(x + 5)(x - 6), \]
\[ G(x) \equiv x^6 - 72x^3 + 25. \]

Transform \( H(x) \) to a "canonical" form

\[ H(x) \equiv \sum_{i=0}^{6} c_i x^i \]

\( H(x) \equiv G(x) \) iff the coefficients of all monomial are equal.
A Randomized Solution:

Choose a random integer $r$ in the range $[1, \ldots, 600]$.

Compute $H(r)$ and $G(r)$.

If $H(r) = G(r)$ output CORRECT else output FALSE.
Randomized Algorithm

We augment the standard RAM operation with a new operation:

Choose a random number uniformly from the set \( \{a_1, a_2, \ldots, a_k\} \).

We assume that this operation takes 1 step.

The output (might) depend on the choice of \( r \), thus it is a random variable.

What are the chances that the algorithm actually gives the correct answer???
Assume $r = 2$

$H(2) = 3 \times 1 \times 5 \times -1 \times 7 \times -4 = 420.$

$G(2) = 2^6 - 72^3 + 25 = 64 - 56 + 25 = 33.$

Since $H(2) \neq G(2)$ we proved that $H(x) \neq G(x).$
What happens if we have equality?

**Example 1:** Check if \((x + 1)(x - 1) \equiv x^2 - 1\).

Since the two sides of the equation are identical - any number that we try would give equality.

Is this algorithm **always** correct?
Example 2: Check if $x^2 + 7x + 1 = (x + 2)^2$.

If we try $r = 2$ we get

$LHS = 4 + 14 + 1 = 19, \quad RHS = 4^2 = 16$

showing that the two sides are not identical.

But for $r = 1$ we get:

$1 + 7 + 1 = (1 + 2)^2 = 9$ equality.

A bad choice of $r$ may lead to a wrong answer!
Some Algebra

Assume that $G(x) \not\equiv H(x)$, and that the sum of the degrees of $x$ in $H$ and $G$ is bounded by $d$.

\[ F(x) \equiv G(x) - H(x) \] is a polynomial of degree bounded by $d$.

**Theorem 1.** The equations

\[ F(x) = G(x) - H(x) = 0 \]

has no more than $d$ roots (solutions).
Analysis of the Algorithm

If the identity is correct, the algorithm always outputs a correct answer.

If the identity is NOT correct, the algorithm outputs the WRONG answer only if we randomly pick \( r \) which is a root of the polynomial \( F(x) = G(x) - H(x) = 0 \).

If we choose \( r \) in the range \([1, \ldots, 100d]\), the "chance" of returning a wrong answer is no more than 1%. 

A randomized technique gives a significantly simpler algorithm - at a cost of a small probability of error.
Getting an arbitrary small error probability

We can reduce the “error probability” at the expense of increasing the run-time of the algorithm:

1. Run the algorithm 10 times.

2. Output “CORRECT” if got “CORRECT” in all the 10 runs.

If the new algorithm outputs “CORRECT” The “chance” that $G(x) \not\equiv H(x)$ is less than $10^{-20} < 2^{-64}$. 
Events and Probability

Consider a random process with a set of outcomes.

Each outcome is an simple event (or sample point);

The sample space $\Omega$ is the set of all possible simple events.

An event is a set of simple events (a subset of the sample space).

Two events $A$ and $B$ are mutually exclusive if $A \cap B = \phi$.

With each simple event $s$ we associate a number $Pr(s)$ which is the probability of $s$. 
Probability Space

A $\sigma$-field $(\Omega, \mathcal{F})$ consists of a sample space $\Omega$ and a collection of subsets $\mathcal{F}$ satisfying the following conditions.

- $\phi \in \mathcal{F}$.

- $E \in \mathcal{F} \Rightarrow \bar{E} \in \mathcal{F}$.

- $E_1, E_2, \ldots \in \mathcal{F} \Rightarrow E_1 \cup E_2 \cup \ldots \in \mathcal{F}$. 
Axioms of Probability

Given a $\sigma$-field $(\Omega, \mathcal{F})$, a probability measure $\Pr: \mathcal{F} \rightarrow \mathbb{R}$ is a function that satisfies the following conditions:

- For any event $E \in \mathcal{F}$, $0 \leq Pr(E) \leq 1$.
- $Pr\{\Omega\} = 1$.
- For any finite or countably infinite sequence of pairwise mutually exclusive events $E_1, E_2, \ldots$:
  \[ Pr\{\bigcup_i E_i\} = \sum_i Pr\{E_i\} \]

A **Probability Space** $(\Omega, \mathcal{F}, \Pr)$ consists of a $\sigma$-field $(\Omega, \mathcal{F})$ with a probability measure $\Pr$ defined on it.

In a **discrete** probability space $\Omega$ is finite or countably infinite and $\mathcal{F} = 2^\Omega$. 
Examples:

Consider the random process defined by the outcome of rolling a dice.

\[ S = \{1, 2, 3, 4, 5, 6\} \]

We assume that all “facets” have equal probability, thus

\[ Pr(1) = Pr(2) = ....Pr(6) = 1/6. \]

The probability of the event “odd outcome”

\[ = Pr(\{1, 3, 5\}) = 1/2 \]
Assume that we roll two dice:

\[ S = \text{all ordered pairs } \{(i, j), 1 \leq i, j \leq 6\}. \]

We assume that each (ordered) combination has probability \(1/36\).

Probability of the event “sum = 2” =

\[ Pr((1, 1)) = 1/36. \]

Probability of the event “sum = 3”

\[ Pr(\{(1, 2), (2, 1)\}) = 2/36. \]
Let \( E_1 = \) “sum bounded by 6”,

\[
E_1 = \{(1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (2, 1), (2, 2), (2, 3), (2, 4), (3, 1), (3, 2), (3, 3), (4, 1), (4, 2), (5, 1)\}
\]

\[
Pr(E_1) = 15/36
\]

Let \( E_2 = \) “both dice have odd numbers”,

\[
Pr(E_2) = 1/4.
\]

\[
Pr(E_1 \cap E_2) =
Pr(\{(1, 1), (1, 3), (1, 5), (3, 1), (3, 3), (5, 1)\}) =
6/36 = 1/6.
\]
A simple event = a choice of $r$.

Sample Space = all integers in $[1, ..., 100d]$.

We assume that all integers in the range have equal probability, thus the probability of a simple event $r$ is $Pr(r) = \frac{1}{100d}$.

The “bad” events: choosing a root of the polynomial. There are no more than $d$ simple events in the bad event.

$Pr(“bad” \mbox{ event}) \leq \frac{d}{100d}$. 
Assume that we repeat the algorithm $k$ times.

If any iteration returns FALSE output FALSE, else output CORRECT.

A simple event = A sequence of $k$ choices $r_1, \ldots, r_k$.

The sample space = All sequences of $k$ numbers in the range $[1, \ldots, 100d]$.

The probability of a simple event = $(\frac{1}{100d})^k$.

The bad event = all $k$ choices are roots of the polynomial, there are no more than $d^k$ such simple events.

Probability of the bad event $\leq d^k(\frac{1}{100d})^k$. 
Principle of Inclusion-Exclusion

Let $E_1, E_2, \ldots, E_n$ be arbitrary events. Then

$$Pr(\cup_{i=1}^{n} E_i) =$$

$$\sum_i Pr(E_i) - \sum_{i<j} Pr(E_i \cap E_j) + Pr_{i<j<k} Pr(E_i \cap E_j \cap E_k) - \ldots + (-1)^{l+1} \sum_{i_1<i_2<\ldots<i_l} Pr(\cap_{r=1}^{l} E_{i_r}) + \ldots$$

**Boole’s inequality (union bound):** For any arbitrary sequence of events $E_1, E_2, \ldots, E_n$:

$$Pr(\cup_{i=1}^{n} E_i) \leq \sum_i Pr(E_i)$$
Conditional Probability

What is the probability that a random person born in Indiana is a student at Purdue.

\[ E_1 = \text{the event that a random person in the world is born in Indiana.} \]

\[ E_2 = \text{the event that a random person in the world is a student at Purdue.} \]

The conditional probability that a random person born in Indiana is a student at Purdue is denoted

\[ Pr(E_2 \mid E_1). \]
Computing Conditional Probabilities

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

By conditioning on \( B \) we restrict the sample space to the set \( B \).

Thus we are interested in \( Pr(A \cap B) \) “normalized” by \( Pr(B) \).
Example

What is the probability that in rolling two dice the sum is 8 given that the sum was even?

$E_1 = \text{“sum is 8”}$,

$Pr(E_1) = Pr((2, 6), (3, 5), (4, 4), (5, 3), (6, 2)) = \frac{5}{36}$

$E_2 = \text{“sum even”}$,

$Pr(E_2) = \frac{1}{2} = \frac{18}{36}.$

$Pr(E_1 \mid E_2) = \frac{Pr(E_1 \cap E_2)}{Pr(E_2)} = \frac{\frac{5}{36}}{\frac{18}{36}} = \frac{5}{18}.$
Example - a posteriori probability

We are given 2 coins. One is a fair coin $A$, the other coin, $B$, has head on both sides $B$.

We choose a coin at random (i.e. each coin is chosen with probability $1/2$) and toss.

Given that we got head, what is the probability that we chose the fare coin $A$???
Define a sample space of ordered pairs \((coin, outcome)\).

The sample space has three points

\[\{(A, h), (A, t), (B, h)\}\]

\[
Pr((A, h)) = Pr((A, t)) = 1/4
\]
\[
Pr((B, h)) = 1/2
\]

Define two events:

\[E_1 = \text{“Chose coin } A\text{”}.
\]
\[E_2 = \text{“Outcome is head”}.
\]

\[
Pr(E_1 \mid E_2) = \frac{Pr(E_1 \cap E_2)}{Pr(E_2)} = \frac{1/4}{1/4 + 1/2} = 1/3.
\]
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 $A_1, \ldots, A_n$ be a sequence of events.

Let $E_i = \bigcap_{j=1}^{i} A_i$

$$Pr(E_n) = Pr(A_n \mid E_{n-1})Pr(E_{n-1}) =$$

$$Pr(A_n \mid E_{n-1})Pr(A_{n-1} \mid E_{n-2}) \cdots Pr(A_2 \mid E_1)Pr(A_1)$$
Independence

Two events $A$ and $B$ are independent if

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

or (when $Pr(B) > 0$)

$$Pr(A \mid B) = \frac{Pr(A \cap B)}{Pr(B)} = Pr(A).$$
Independent events do not have to be related to independent physical processes.

Example: the probability that the outcome of a dice roll is \( \textit{even} \) \((= \frac{3}{6})\) is independent of the event ”the outcome is \( \leq 4" \) \((= \frac{4}{6})\).

The probability of ”an even outcome \( \leq 4" \) is

\[
\frac{2}{6} = \frac{12}{36} = \frac{3}{6} \cdot \frac{4}{6}
\]
Back to the Polynomial Identity Test

The probability of a wrong answer in one run of the algorithm is $1/100$. Runs of the algorithms are independent (i.e., the probability of error in each run is independent of other runs.) It’s enough to get a correct answer in one of the runs.

The probability of a wrong answer in $k$ runs of the algorithm is $(\frac{1}{100})^k$. 
The Monty Hall Game

A player is shown three doors, behind one door there is a prize, nothing behind the other two doors.

The player picks a door.

Before that door is opened the host opens another door with no prize.

The player can now stick with his original choice or pick the other door.

What should the player do???
1. The host just tries to confuse the player, the prize had equal chances to be behind each of the two closed doors.

2. Since the host had to open an empty door, it increases the chances of the other door.
Assumption: the prize is put behind each door with equal probability.

A priori any choice of the player has equal probability of leading to the prize.

Given that the host open one door we need to check if the a posteriori probabilities for winning on a given door had changed
We compute the probability that each of the two strategies win the prize.

The sample space represents the three possibility of placing the prize = \( \{A, B, C\} \), each sample point has probability \( 1/3 \).

Assume that the player chose door A.

The probability that A has the prize is \( 1/3 \).

If it does not switch doors, his winning probability remains \( 1/3 \).

If the player chooses the strategy of switching doors he is winning in the simple events \( \{B, C\} \). Thus, his winning probability is \( 2/3 \).
Principle of Inclusion-Exclusion

Let \( E_1, E_2, \ldots, E_n \) be arbitrary events. Then

\[
Pr(\bigcup_{i=1}^{n} E_i) = \\
\sum_i Pr(E_i) - \sum_{i<j} Pr(E_i \cap E_j) + \sum_{i<j<k} Pr(E_i \cap E_j \cap E_k) - \ldots + (-1)^{l+1} \sum_{i_1<i_2<\ldots<i_l} Pr(\cap_{r=1}^{l} E_{i_r}) + \ldots
\]

Boole’s inequality (union bound): For any arbitrary sequence of events \( E_1, E_2, \ldots, E_n \):

\[
Pr(\bigcup_{i=1}^{n} E_i) \leq \sum_i Pr(E_i)
\]
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 \( A_1, \ldots, A_n \) be a sequence of events.

Let \( E_i = \bigcap_{j=1}^{i} A_i \)

\[ Pr(E_n) = Pr(A_n \mid E_{n-1})Pr(E_{n-1}) = \]

\[ Pr(A_n \mid E_{n-1})Pr(A_{n-1} \mid E_{n-2}) \ldots Pr(A_2 \mid E_1)Pr(A_1) \]
The Birthday Paradox

What is the probability that among $m$ people no two have the same birthday?

Assumptions:

1. All birthdays are equally likely.

2. Birthdays are independent events.
The sample space is the set of all vectors \( S = \{(b_1, \ldots, b_m) | b_i \in [1, \ldots, N]\} \).

We need to compute \( Pr(E) \) where \( E = \{(b_1, \ldots, b_m | b_i \neq b_j \text{ for all } i \neq j\} \).
How many different atomic events are counted in $E$?

The number of possible $m$ different birthdays is $N.(N-1).(N-2)\ldots(N-m+1)$

$$Pr(E) = \frac{N.(N-1).(N-2)\ldots(N-m+1)}{N^m}$$

$$= \prod_{i=0}^{m-1} (1 - i/N)$$

$$\leq \prod_{i=0}^{m-1} e^{-i/N} = e^{-\sum_{i=0}^{m-1} i/N} = e^{-m(m-1)/2N}$$

For $m = \sqrt{2N} + 1 \leq 28$, 

$$Pr(E) < 1/e < 1/2.$$
Alternate Analysis

Assume that we choose one birthday after the other independently and uniformly at random from \([1 \ldots N]\).

Let \(E_i\): "the \(i\)th choice is different from the first \(i-1\) choices".

\[
\Pr(\bigcap_{i=1}^m E_i) = \Pr(E_1) \Pr(E_2|E_1) \Pr(E_3|E_2 \cap E_1) \ldots \Pr(E_m|\bigcap_{i=1}^{m-1} E_i) = \prod_{i=1}^m \left(1 - \frac{i-1}{N}\right)
\]

**Principle of Deferred Decisions.**
Verifying Matrix Multiplication

Given three $n \times n$ integer matrices $A$, $B$, and $C$, verify whether

$$AB = C$$

Simple matrix multiplication takes $\Theta(n^3)$ steps. There exist a $\Theta(n^{2.37})$ algorithm.
Randomized Algorithm

Choose a random vector $\bar{r} = (r_1, r_2, \ldots, r_n) \in \{0, 1\}^n$.

Compute $A(B\bar{r})$ and $C\bar{r}$.

If $A(B\bar{r}) \neq C\bar{r}$, then output $AB \neq C$.

Else output $AB = C$.

Takes $\Theta(n^2)$ time.
Bounding the Error Probability

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

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

**Proof.** Let $D = AB - C \neq 0$. $D$ has some non-zero entry. W.l.o.g let it be $d_{11}$.

For $D\bar{r} = 0$, we should have

$$\sum_{j=1}^{n} d_{1j} r_j = 0, \text{ i.e.,}$$

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

Set $r_n, \ldots, r_2, r_1$ one by one.

The equality holds in at most one choice of $r_1$. \(\square\)
Verifying String Equalities

**Application:** Comparing the consistency of two replicates of a database.

Given two long strings $a = a_1, \ldots, a_n$ and $b = b_1, \ldots, b_n$ we want to check equality.

For a string $x = x_1, \ldots, x_n$ define the “fingerprint” of $x$

$$F_p(x) = \left(\sum_{i=1}^{n} x_i 2^{i-1}\right) \mod p$$
Theorem 1. Let \( p \) be a random prime number in the range \([1, ..., 2n^2 \log n]\).

\[
Pr(F_p(a) = F_p(b) \mid a \neq b) = O\left(\frac{1}{n}\right).
\]

Proof. If \( a \neq b \) then we get equality only if \( p \) divides
\[
c = \sum_{i=1}^{n} (a_i - b_i)2^{i-1}.
\]

Since \( c \leq 2^n \), the number of different primes that divide \( c \) is bounded by \( n \).

The number of primes in the range \([1, ..., T]\) is about \( T/\log T \).

Thus, there are \( \Omega(n^2) \) primes in the range \([1, ..., 2n^2 \log n]\). Less than \( n \) primes can divide \( c \). \( \square \)

Application: We can compare the consistency of two replicates of a database of \( n \) bits by exchanging only \( O(\log n) \) bits.
Min-Cut Algorithm

**Input:** A \( n \) node graph  
**Output:** A minimal set of edges that disconnects the graph.

1. **repeat** \( n - 2 \) **times:**  
   (a) Pick an edge uniformly at random.  
   (b) Contract the edge.  
**endrepeat**

2. **output** the set of edges connecting the two remaining vertices.
**Theorem 2.** The algorithm outputs a min-cut set of edges with probability $\geq \frac{2}{n(n-1)}$.

**Lemma 1.** Contraction operation (step 1(b)) does not reduce the size of the min-cut set.

**Proof.** Every cut set in the new graph is a cut set in the original graph. $\square$
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 \( C \).

**Lemma 2.** Let \( C \) be a min-cut set of \( G \). If the run of the algorithm did not contract any edge of \( C \), it also did not eliminate any edge of \( C \).

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

Let \( E_i = "\text{the edge contracted in iteration } i \text{ is not in } C\". \)
We want
\[
\Pr(\cap_{i=1}^{n-2} E_i) = \Pr(E_1) \Pr(E_2|E_1) \Pr(E_3|E_2 \cap E_1) \ldots \Pr(E_{n-2}|\cap_{i=1}^{n-3} E_i).
\]
\[
\Pr(E_1) \geq 1 - \frac{k}{nk/2} = 1 - \frac{2}{n}
\]
\[
\Pr(E_2|E_1) \geq 1 - \frac{k}{(n-1)k/2} = 1 - \frac{2}{n-1}
\]
\[
\Pr(E_i|\cap_{j=1}^{i-1} E_j) \geq 1 - \frac{2}{n-i+1}
\]

Thus,
\[
\Pr(\cap_{i=1}^{n-2} E_i) \geq \prod_{i=1}^{n-2} \left(1 - \frac{2}{n-i+1}\right) = 
\]
\[
\prod_{i=1}^{n-2} \left(\frac{n-i-1}{n-i+1}\right) = \frac{2}{n(n-1)}
\]
Min-Cut: High Probability Algorithm

**Input:** A $n$ node graph

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

1. **for** $i = 1$ to $n^2 \ln n$
   1.1. **repeat** $n - 2$ times:
       (a) Pick an edge uniformly at random.
       (b) Contract the edge.
   **endrepeat**
   1.2. Let $C_i$ be the set of edges connecting the two remaining vertices.

**endfor**

2. **output** the set with the minimum size among the $C_i$'s.
Theorem 3. The high probability algorithm outputs a min-cut set with probability at least $1 - 1/n^2$. 
Random Variable

Let \((S, Pr)\) be a discrete probability space.

Let \(V\) be a set of values.

A random variable \(X\) defined on \((S, Pr)\) is a function

\[
X : S \rightarrow V
\]

Let \(E(r) = \{s \in S \mid X(s) = r\}\)

\[
Pr(X = r) = Pr(E(r)) = \sum_{s \in E(r)} Pr(s).
\]

Two random variables \(X\) and \(Y\) (defined on the same sample space) are called independent if for all \(x\) and \(y\)

\[
Pr\{X = x \text{ and } Y = y\} = Pr\{X = x\} Pr\{Y = y\}
\]
Example 1: In rolling a dice, the number that comes up is a random variable.

Example 2: Consider a gambling game in which a player flips two coins, if he gets head in both coins we wins $3, else he losses $1. The payoff of the game is a random variable.
Definition 1. The expectation of a discrete random variable $X$ is

$$E[X] = \sum_{i \in \text{range}(X)} i Pr(X = i).$$

The expectation (or mean or average) is a weighted sum over all possible values of the random variable.

Example: The expected value of one dice roll is:

$$E[X] = \sum_{i=1}^{6} i Pr(X = i) = \sum_{i=1}^{6} \frac{i}{6} = 3\frac{1}{2}.$$
Consider a game in which a player chooses a number in $[1, \ldots, 6]$ and then rolls 3 dice.

The player wins $1 for each dice the matches the number, he losses $1 if no dice matches the number.

What is the expected outcome of that game:

$$-1\left(\frac{5}{6}\right)^3 + 1 \cdot 3\left(\frac{1}{6}\right)\left(\frac{5}{6}\right)^2 + 2 \cdot 3\left(\frac{1}{6}\right)^2\left(\frac{5}{6}\right) + 3\left(\frac{1}{6}\right)^3 = -\frac{17}{216}.$$
Linearity of Expectation

**Theorem 1.**  For any two random variables $X$ and $Y$

$$E[X + Y] = E[X] + E[Y].$$

**Proof.**

\[
E[X + Y] = \sum_{i \in \text{range}(X)} \sum_{j \in \text{range}(Y)} (i + j)Pr((X = i) \cap (Y = j)) =
\]

\[
\sum_{i} \sum_{j} iP r((X = i) \cap (Y = j)) + \sum_{j} \sum_{i} j Pr((X = i) \cap (Y = j)) =
\]

\[
\sum_{i} iP r(X = i) + \sum_{j} j Pr(Y = j).
\]

\(\square\)

(Since we sum over all possible choices of \(i\) \((j)\).)
Theorem 2. If $E_1, E_2, \ldots, E_k$ are disjoint events such that $\sum_{i=1}^{k} Pr(E_i) = 1$ then for any event $B$,

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

1. The expectation of the sum of two dice is 7, even if they are not independent.

2. Assume that we flip \( N \) coins, what is the expected number of heads?

   Using linearity of expectation we get \( N \cdot \frac{1}{2} \).

   By direct summation we get \( \sum_{i=0}^{N} i \binom{N}{i} 2^{-N} \).

   Thus we prove

\[
\sum_{i=0}^{N} i \binom{N}{i} 2^{-N} = \frac{N}{2}.
\]
3. Assume that $N$ people checked coats in a restaurant. The coats are mixed and each person gets a random coat.

How many people got their own coats?

It’s hard to compute $E[X] = \sum_{k=0}^{N} kPr(X = k)$. Instead we define $N$ 0-1 random variables $X_i$, where $X_i = 1$ iff $i$ got his coat.

\[
E[X_i] = 1 \cdot Pr(X_i = 1) + 0 \cdot Pr(X_i = 0) = Pr(X_i = 1) = \frac{1}{N}.
\]

\[
E[X] = \sum_{i=1}^{N} E[X_i] = 1.
\]
Randomized Quicksort

Procedure Q(S);

Input: A set S.

Output: The set S in sorted order.

1. If |S| ≤ 1 then return S, else

2(a) Choose a random element y uniformly from S.
(b) Compare all elements of S to y. Let

\[ S_1 = \{ x \in S - \{y\} | x \leq y \} \]

\[ S_2 = \{ x \in S - \{y\} | x > y \} \].

(Elements in \( S_1 \) and \( S_2 \) are in the same order as in \( S \).)
(c) Return the list:

\[ Q(S_1), y, Q(S_2). \]
Let $T =$ number of comparisons in a run of QuickSort.

**Theorem 3.**

$$E[T] = O(n \log n).$$
Let $s_1, \ldots, s_n$ be the elements of $S$ in sorted order.

For $i = 1, \ldots, n$, and $j > i$, define 0-1 random variable $X_{i,j}$, s.t.

$X_{i,j} = 1$ iff $s_i$ is compared to $s_j$ in the run of the algorithm.

The number of comparisons in running the algorithm is

$$T = \sum_{i=1}^{n} \sum_{j>i} X_{i,j}.$$ 

We are interested in $E[T]$. 
What is the probability that $X_{i,j} = 1$?

$s_i$ is compared to $s_j$ iff either $s_i$ or $s_j$ is chosen as a “split item” before any of the $j - i - 1$ elements between $s_i$ and $s_j$ are chosen.

Elements are chosen uniformly at random → elements in the set $[s_i, s_{i+1}, ..., s_j]$ are chosen uniformly at random.

$$Pr(X_{i,j} = 1) = \frac{2}{j - i + 1}.$$  

$$E[X_{i,j}] = \frac{2}{j - i + 1}.$$
\[ E[T] = E\left[ \sum_{i=1}^{n} \sum_{j>i} X_{i,j} \right] = \]
\[ \sum_{i=1}^{n} \sum_{j>i} E[X_{i,j}] = \sum_{i=1}^{n} \sum_{j>i} \frac{2}{j - i + 1} \leq \]
\[ \sum_{i=1}^{n} \sum_{k=1}^{n-i+1} \frac{2}{k} \leq 2 \sum_{i=1}^{n} \sum_{k=1}^{n} \frac{1}{k} = 2nH_n = n \log n + O(n) \]
Theorem 4. The expected run time of (deterministic) Quicksort on a random input, uniformly chosen from all possible permutation of $S$ is $O(n \log n)$.

Proof.

Set $X_{i,j}$ as before.

If all permutations have equal probability, all permutations of $S_i, \ldots, S_j$ have equal probability, thus

$$Pr(X_{i,j}) = \frac{2}{j-i+1}.$$  

$$E\left[\sum_{i=1}^{n} \sum_{j>i} X_{i,j}\right] = O(n \log n).$$
Randomized Algorithms:

- Analysis is true for any input.
- The sample space is the space of random choices made by the algorithm.
- Repeated runs are independent.

Probabilistic Analysis:

- The sample space is the space of all possible inputs.
- If the algorithm is deterministic repeated runs give the same output.
Randomized Algorithm classification

A **Monte Carlo Algorithm** is a randomized algorithm that may produce an incorrect solution.

For decision problems: A **one-side error** Monte Carlo algorithm errs only on one possible output, otherwise it is a **two-side error** algorithm.

A **Las Vegas** algorithm is a randomized algorithm that **always** produces the correct output.

In both types of algorithms the run-time is a random variable.
Occupancy Problems - The Coupon Collector Problem

Assume that $m$ balls are placed randomly in $n$ boxes.

- How many boxes are empty?
- How many boxes have at least $k$ boxes?
- What is the maximum number of balls in any box?
- How many randomly thrown balls are needed to fill all the boxes?
Let $X =$ number of balls till all boxes are not empty.

Let $X_i =$ number of balls from the time $i - 1$ boxes are not empty till a new empty box receives a ball.

$$X = \sum_{i=1}^{n} X_i.$$  

If $i - 1$ boxes are not empty, the probability of hitting a new box is

$$p_i = 1 - \frac{i - 1}{n}$$
The Geometric Distribution

Assume that an experiment has probability \( p \) for success \( 1 - p \) for failure. How many trials we need till the first success.

\[
Pr(X = i) = (1 - p)^{i-1}p.
\]

\( X \) has a Geometric distribution with parameter \( p \)

\( X \sim G(p) \).

\[
\sum_{i=1}^{\infty} Pr(X = i) = \sum_{i=1}^{\infty} (1 - p)^{i-1}p = p \frac{1}{1 - (1 - p)} = 1.
\]
Assume that $X$ get values in $\mathcal{N}$.

$$E[X] = \sum_{i \geq 0} iPr(X = i) = \sum_{i \geq 1} Pr(X \geq i)$$

Let $X \sim G(p)$,

$$E[X] = \sum_{i \geq 1} Prob(X \geq i) = \sum_{i \geq 1} (1 - p)^{i-1} = \frac{1}{1 - (1 - p)} = \frac{1}{p}.$$
The Geometric distribution is **memoryless**: 

For any \( k > r \),

\[
Pr(X > k \mid X > r) = \frac{(1 - p)^k}{(1 - p)^r} = (1 - p)^{k-r} = Pr(X > (k - r))
\]
Let $X = \text{number of balls till all boxes are not empty.}$

Let $X_i = \text{number of balls from the time } i - 1 \text{ boxes are not empty till a new empty box receives a ball.}$

$X = \sum_{i=1}^{n} X_i.$

If $i - 1$ boxes are not empty, the probability of hitting a new box is

$$p_i = 1 - \frac{i - 1}{n}$$

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

$$E[X] = \sum_{i=1}^{n} \frac{n}{n-i+1} = n \sum_{i=1}^{n} \frac{1}{i} = nH_n = n \log n + O(n).$$
Theorem 1. Assume that balls are placed randomly and independently in \( n \) boxes. The expected number of balls placed till no box is empty is

\[ n \ln n + O(n) \]
Theorem 2. Assume that $2n \log n$ balls are placed randomly into $n$ boxes, then with probability at least $1 - \frac{1}{n}$ no box is empty.

Proof. The probability that box $i$ is empty after $2n \log n$ balls is bounded by

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

The probability that any box is empty is bounded by

$$Pr(\bigcup_{i=1}^{n} E_i) \leq \sum_{i=1}^{n} Pr(E_i) \leq n^{-1}.$$ 

□
What is the maximum number of balls in any box?

Let $E_{i,\geq k}$ be the event box $i$ received $\geq k$ balls.

$$Pr(E_{i,\geq k}) \leq \binom{m}{k} \left(\frac{1}{n}\right)^k$$

The probability that any box received at least $k$ balls is bounded by

$$Pr(\bigcup_{i=1}^{n} E_{i,\geq k}) \leq \sum_{i=1}^{n} Pr(E_{i,\geq k})$$
For $m = n$, and $k = \frac{e \log n}{\log \log n}$ we get

$$n \left( \frac{n}{k} \right) \left( \frac{1}{n} \right)^k \leq n \left( \frac{e}{k} \right)^k \leq$$

$$n e^{-2 \log n} = o(1).$$

For $m = 2n \ln n$, and $k = 4e \log n$, we get

$$n \left( \frac{2n \log n}{k} \right) \left( \frac{1}{n} \right)^k \leq n \left( \frac{2en \log n}{k} \right)^k \left( \frac{1}{n} \right)^k \leq$$

$$n e^{-2 \log n} = o(1).$$
Consider a society of $n$ men and $n$ women. A marriage is a 1-1 function between men to women.

Each person (man or woman) has an ordered list of preferences.

A marriage is unstable if there are two couples $x_1, y_1$ and $x_2, y_2$ such that $x_1$ prefers $y_2$ on $y_1$ and $y_2$ prefers $x_1$ to $x_2$. Else the marriage is stable.
Stable Marriage Algorithm:

1. Start with all men and women unmarried.

2. **Repeat** till all men are married.
   
   (a) An unmarried man proposes to the first woman on his list that hasn’t rejected him yet.
   
   (b) The proposed woman accepts the proposal if she is currently unmarried, or if she is married but prefers the current proposer on her current mate (in that case the old mate becomes unmarried).
**Theorem 1.** The algorithm always terminates in $O(n^2)$ steps with a stable marriage.

**Proof.** Once a woman is married she remains married, when she changes mates her preference of the mates only improves.

If a man is not married there is an available woman in his list, since all the women he proposed to are married and there are $n$ women. Thus, $O(n^2)$ time.

Assume that output marriage is not stable, i.e. $x_1$ married to $y_1$ and prefers $y_2$ and $y_2$ prefers $x_1$ on $x_2$. We prove a contradiction.

If $x_1$ prefers $y_2$ he must have proposed $y_2$ before $y_1$. Since $y_2$ rejected $x_1$ at some point in the algorithm it must be married to a more desirable mate.

$\square$
Theorem 2. Assuming that the lists of preferences are chosen uniformly and independently at random, the expected number of proposals made by the algorithm is $O(n \log n)$, the maximum number of proposal any woman receives is bounded by $O(\log n)$.

Proof.

Instead of having a pre-selected preference lists we can assume that a man chooses a random woman that did not rejected him till now.

We simplify the analysis by assuming that each time a man proposed he proposed to a random woman chosen uniformly from among all women.
Since all the ”added” proposals are rejected the number of proposals in the new algorithm stochastically dominates the number of proposal in the original algorithm.

$$\forall x \ Pr(T_{new} > x) \geq Pr(T > x).$$

The algorithm terminates once all women receive at least one proposal. \qed
Expectation is not everything....

Which gambling game would you prefer:

- We flip one coin, you win $1 if head, loose one $1 if tail.

- We flip 10 coins, you win $2^{10}/2 = 0.5K$ if all heads, else you pay $1.$

- We flip 20 coins, you win $2^{20}/2 = M/2$ if all heads, else you pay $1.$
Bounding Deviation from Expectation

**Theorem 3. [Markov Inequality]** For any non-negative random variable

\[ \Pr(X \geq a) \leq \frac{E[X]}{a}. \]

**Proof.**

\[
E[X] = \sum_i i \Pr(X = i) \geq a \sum_{i \geq a} \Pr(X = i) = a \Pr(X \geq a).
\]

Example: What is the probability of getting more than \( \frac{3N}{4} \) heads in \( N \) coin flips?

\[
\leq \frac{N/2}{3N/4} \leq \frac{2}{3}.
\]
Variance

Definition 1. The variance of a random variable $X$ is

$$Var[X] = E[(X - E[X])^2].$$

Definition 2. The standard deviation of a random variable $X$ is

$$\sigma(X) = \sqrt{Var[X]}.$$
Example: Let $X$ be a 0-1 random variable with $Pr(X = 0) = Pr(X = 1) = 1/2$.

$$E[X] = 1/2.$$ 

$$Var[X] = \frac{1}{2}(1 - \frac{1}{2})^2 + \frac{1}{2}(0 - \frac{1}{2})^2 = \frac{1}{4}.$$
Chebyshev’s Inequality

**Theorem 4.** For any random variable

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

**Proof.**

\[ Pr(|X - E[X]| \geq a) = Pr((X - E[X])^2 \geq a^2) \]

By Markov inequality

\[ Pr((X - E[X])^2 \geq a^2) \leq \frac{E[(X - E[X])^2]}{a^2} = \frac{Var[X]}{a^2} \]
Theorem 5. For any random variable

\[ Pr(|X - E[X]| \geq a\sigma[X]) \leq \frac{1}{a^2}. \]

Theorem 6. For any random variable

\[ Pr(|X - E[X]| \geq \epsilon E[X]) \leq \frac{Var[X]}{\epsilon^2(E[X])^2}. \]
Theorem 7. If $X$ and $Y$ are independent random variable

$$E[XY] = E[X] \cdot E[Y],$$

Proof.

$$E[XY] = \sum_i \sum_j i \cdot j \Pr((X = i) \cap (Y = j)) =$$

$$\sum_i \sum_j ij \Pr(X = i) \cdot \Pr(Y = j) =$$

$$(\sum_i i \Pr(X = i)) (\sum_j j \Pr(Y = j)).$$

$\square$
Theorem 8. If $X$ and $Y$ are independent random variable

$$Var[X + Y] = Var[X] + Var[Y].$$

Proof.

$$Var[X + Y] = E[(X + Y − E[X] − E[Y])^2] =$$

$$E[(X − E[X])^2 + (Y − E[Y])^2 + 2(X − E[X])(Y − E[Y])] =$$

$$Var[X] + Var[Y] + 2E[X − E[X]]E[Y − E[Y]]$$

Since the random variables $X − E[X]$ and $Y − E[Y]$ are independent.

But $E[X − E[X]] = E[X] − E[X] = 0$. □
Assume again that we flip $N$ coins. Let $X$ be the number of heads.

$X_i = 1$ if the $i$-th flip was a head else $X_i = 0$.


$$Pr(X \geq 3N/4) \leq Pr(|X - E[X]| \geq N/4) =$$

$$Pr(|X - E[X]| \geq E[X]/2) \leq \frac{Var[X]}{(E[X])^2(1/4)} =$$

$$\frac{N/4}{(N^2/4)(1/4)} = 4/N.$$

A significantly better bound than $3/4$. 
Bernoulli Trial

Let $X$ be a 0-1 random variable such that

$$Pr(X = 1) = p, \quad Pr(X = 0) = 1 - p.$$

$$E[X] = 1 \cdot p + 0 \cdot (1 - p) = p.$$

A Binomial Random variable

Consider a sequence of \( n \) independent Bernoulli trials \( X_1, \ldots, X_n \). Let

\[ X = \sum_{i=1}^{n} X_i. \]

\( X \) has a **Binomial** distribution \( X \sim B(n, p) \).

\[ Pr(X = k) = \binom{n}{k} p^k (1 - p)^{n-k}. \]

\[ E[X] = np. \]

\[ Var[X] = np(1 - p). \]
Algorithm for Computing the Median

The **median** of a set \( X \) of \( n \) distinct elements is the \( \lceil \frac{n}{2} \rceil \) largest element in the set.

If \( n = 2k + 1 \), the median element is the \( k + 1 \)-th element in the sorted order.

Easily computed through sorting in \( O(n \log n) \) time. There exists a complicated \( O(n) \) deterministic algorithm.
Randomized Median Algorithm

**Input:** A set of $n = 2k + 1$ elements from a totally ordered universe.

**Output:** The $k + 1$-th largest element in the set.
1. Pick a (multi)-set \( R \) of \( s = n^{3/4} \) elements in \( S \), chosen independently and uniformly at random with replacement. Sort the set \( R \).

2. Let \( d \) be the \( \left( \frac{1}{2}n^{3/4} - \sqrt{n} \right) \)th smallest element in the sorted set \( R \).

3. Let \( u \) be the \( \left( \frac{1}{2}n^{3/4} + \sqrt{n} \right) \)th smallest element in the sorted set \( R \).

4. By comparing every element in \( S \) to \( d \) and \( u \) compute the set \( C = \{ x \in S : d \leq x \leq u \} \), and the numbers \( \ell_d = |\{ x \in S : x < d \}| \) and \( \ell_u = |\{ x \in S : x > u \}| \).

5. If \( \ell_d > n/2 \) or \( \ell_u > n/2 \) then FAIL.

6. If \( |C| \leq 4n^{3/4} \) then sort the set \( C \), otherwise FAIL.

7. Output the \( \left( \left\lfloor \frac{n}{2} \right\rfloor - \ell_d + 1 \right) \)st element in the sorted order of \( C \).
Algorithm for Computing the Median

The median of a set $X$ of $n$ distinct elements is the $\lceil \frac{n}{2} \rceil$ largest element in the set.

If $n = 2k + 1$, the median element is the $k + 1$-th element in the sorted order.

Easily computed through sorting in $O(n \log n)$ time. There exists a complicated $O(n)$ deterministic algorithm.
Randomized Median Algorithm

**Input:** A set of \( n = 2k + 1 \) elements from a totally ordered universe.

**Output:** The \( k + 1 \)-th largest element in the set.

2. Let $d$ be the $(\frac{1}{2}n^{3/4} - \sqrt{n})$th smallest element in the sorted set $R$.

3. Let $u$ be the $(\frac{1}{2}n^{3/4} + \sqrt{n})$th smallest element in the sorted set $R$.

4. By comparing every element in $S$ to $d$ and $u$ compute the set $C = \{x \in S : d \leq x \leq u\}$, and the numbers $\ell_d = |\{x \in S : x < d\}|$ and $\ell_u = |\{x \in S : x > u\}|$.

5. If $\ell_d > n/2$ or $\ell_u > n/2$ then FAIL.

6. If $|C| \leq 4n^{3/4}$ then sort the set $C$, otherwise FAIL.

7. Output the $(\lfloor \frac{n}{2} \rfloor - \ell_d + 1)$st element in the sorted order of $C$. 
Intuition

• We can sort sets of size $o(n)$ in linear time.

• The sample of $R$ elements are spaced “more or less” evenly among the elements of $X$.

• W.h.p. more than $\frac{1}{2}n^{3/4} - \sqrt{n}$ samples are smaller than the median.

• W.h.p. more than $\frac{1}{2}n^{3/4} - \sqrt{n}$ samples are larger than the median.

• W.h.p. the median is in the set $C$, and $|C| \leq 4n^{3/4}$. 
Let $Y_1$ be the number of samples below or equal to the median.

Let $Y_2$ be the number of samples above or equal to the median.

The algorithm computes the median in $O(n)$ time if all the following three events hold:

1. $E_1 : Y_1 \geq \frac{1}{2}n^{3/4} - \sqrt{n}$.

2. $E_2 : Y_2 \geq \frac{1}{2}n^{3/4} - \sqrt{n}$.

3. $E_3 : |C| \leq 4n^{3/4}$.

What is the probability that the three random variables $Y_1, Y_2$ and $|C|$ are all within the required ranges?
The sample space in execution of this algorithm is the set of all possible choices of \( n^{3/4} \) elements from \( n \), with repetitions. (The sample space has \( n^{n^{3/4}} \) points.)

Each point in the sample space defines values for \( Y_1, Y_2 \) and \( |C| \).

Computing the probabilities directly is too complicated, instead we use bounds on deviation from the expectation.
$Y_1$ = the number of samples below or equal the median.

What is the probability that $Y_1 < \frac{1}{2}n^{3/4} - \sqrt{n}$

Viewing $Y_1$ as the sum of $n^{3/4}$ independent 0-1 random variable, each with expectation $1/2$ and variance $1/4$ we prove:

$$E[Y_1] > \frac{1}{2}n^{3/4}.$$  

$$Var[Y_1] < \frac{1}{4}n^{3/4}.$$
Applying Chebyshev Inequality we get:

\[ Pr(\bar{E}_1: Y_1 < \frac{1}{2} n^{3/4} - \sqrt{n}) \leq Pr(|Y_1 - E[Y_1]| > \sqrt{n}) \leq \frac{Var[Y_1]}{n} = \frac{n^{3/4}}{4n} = \frac{1}{4n^{-1/4}}. \]

Similarly

\[ Pr(\bar{E}_2: Y_2 < \frac{1}{2} n^{3/4} - \sqrt{n}) \leq \frac{1}{4n^{-1/4}}. \]

\[ Pr(\bar{E}_1 \cup \bar{E}_2) \leq \frac{2}{4} n^{-1/4}. \]
Recall: $E_3 : |C| \leq 4n^{3/4}$.

**Lemma 1.**

$$\Pr(\bar{E}_3) \leq \frac{1}{2} n^{-1/4}.$$  

Define the following two events:

1. $E_{3,1}$: at least $2n^{3/4}$ elements of $C$ are greater than the median;

2. $E_{3,2}$: at least $2n^{3/4}$ elements of $C$ are smaller than the median.

If $|C| > 4n^{3/4}$, then at least one of the above two events occurs.
We bound $\mathcal{E}_{3,1}$: at least $2n^{3/4}$ elements of $C$ are greater than the median;

At least $2n^{3/4}$ elements of $C$ above the median $\Rightarrow$

$u$ is at least the $\frac{1}{2}n + 2n^{3/4}$ largest in $S$ $\Rightarrow$

$R$ had at least $\frac{1}{2}n^{3/4} - \sqrt{n}$ samples among the $\frac{1}{2}n - 2n^{3/4}$ largest elements in $S$.

Let $X$ be the number of samples among the $\frac{1}{2}n - 2n^{3/4}$ largest elements in $S$. Let $X = \sum_{i=1}^{n^{3/4}} X_i$ where

$$X_i = \begin{cases} 
1 & \text{the } i\text{-th sample in } \frac{1}{2}n - 2n^{3/4} \\
0 & \text{otherwise.}
\end{cases}$$
\[
E[X_i] = E[(X_i)^2] = \frac{1}{2} - 2n^{-1/4}
\]

\[
Var[X_i] = E[(X_i)^2] - (E[X_i])^2 \leq \frac{1}{4}.
\]

\[
E[X] = \frac{1}{2}n^{3/4} - 2\sqrt{n}
\]

\[
Var[X] \leq \frac{1}{4}n^{3/4}
\]

Applying Chebyshev’s Inequality yields

\[
Pr(\mathcal{E}_{3,1}) = Pr(X \geq \frac{1}{2}n^{3/4} - \sqrt{n}) \\
\leq Pr(|X - E[X]| \geq \sqrt{n}) \\
\leq \frac{Var[X]}{n} = \frac{n^{3/4}}{n} = \frac{1}{4}n^{-1/4}.
\]
Similarly,

\[ \Pr(\mathcal{E}_{3,2}) \leq \frac{1}{4} n^{-\frac{1}{4}}, \]

and

\[ \Pr(\bar{E}_3) \leq \Pr(\mathcal{E}_{3,1}) + \Pr(\mathcal{E}_{3,2}) \leq \frac{1}{2} n^{-\frac{1}{4}}. \]

The probability that the algorithm succeeds is

\[ \geq 1 - (Pr(\bar{E}_1) + Pr(\bar{E}_2) + Pr(\bar{E}_3)) \geq 1 - \frac{1}{n^{1/4}}. \]
A set of random variables $X_1, X_2, \ldots$ is said to be pairwise independent if for all $i \neq j$, $X_i$ and $X_j$ are independent.

Example: $X$ and $Y$ be independent r.v. each taking values -1 and 1 with probability 1/2 and let $Z = XY$.

**Theorem 1.** Let $n$ be a prime number and $\mathbb{Z}_n$ denote the field of integers modulo $n$. For $a$ and $b$ chosen independently and uniformly at random from $\mathbb{Z}_n$, let

$$Y_i = ai + b \pmod{n}$$

Then for $i \neq j \pmod{n}$, $Y_i$ and $Y_j$ are uniformly distributed on $\mathbb{Z}_n$ and pairwise independent.

**Proof.** Let $i \neq j$. Then in the field $\mathbb{Z}_n$, for any given fixed values of $y_i$ and $y_j$ we can solve the equations:

$$y_i = ai + b \pmod{n}$$
\[ y_j = aj + b(\text{mod } n) \]

uniquely for \( a \) and \( b \).

\[
a = (y_i - y_j)((i - j)^{-1}\text{mod } n)\text{mod } n
\]

\[
b = (y_i - ai)\text{mod } n
\]

That is, there is 1-1 correspondence between pairs \((a, b)\) and \((y_i, y_j)\).

If we pick \((a, b)\) uniformly at random, any pair \((y_i, y_j)\) is equally likely. \qed
Theorem 2. Let \( X_1, X_2, \ldots, X_m \) be pairwise independent r.v.s and \( X = \sum_{i=1}^{m} X_i \). Then
\[
\text{Var}(X) = \sum_{i=1}^{m} \text{Var}(X_i).
\]

Covariance of two random variables is defined as:
\[
\text{Cov}(X_i, X_j) = E[(X_i - E[X_i])(X_j - E[X_j])]
\]
\[
= E(X_iX_j) - E[X_i]E[X_j]
\]

\[
\text{Var}(X + Y) = \text{Var}(X) + \text{Var}(Y) + 2\text{Cov}(X_i, X_j)
\]
If \( X_i \) and \( X_j \) are independent \( \text{Cov}(X_i, X_j) = 0 \).

In general,
\[
\text{Var} \left( \sum_{i=1}^{n} X_i \right) = \sum_{i=1}^{n} \text{Var}(X_i) + 2\sum_{i<j} \text{Cov}(X_i, X_j)
\]
We consider a class of randomized algorithms called RP.

An RP algorithm $A$ for deciding whether input strings $x$ belong to a language $L$ is as follows:

Given $x$, $A$ picks a random $r$ from $\mathbb{Z}_n = \{0, \ldots, n-1\}$ for some prime $n$. Computes a binary function $f(x, r)$ such that:

If $x \in L$, then $f(x, r) = 1$ for at least half the possible values of $r$.

If $x \notin L$ then $f(x, r) = 0$ for all choices of $r$.

Repeating the algorithm $t$ times using $t$ random samples from $\mathbb{Z}_n$ the probability of error is at most $2^{-t}$. Uses $\Omega(t \log n)$ random bits.
Pairwise Independence and Sampling

A set of random variables \(X_1, X_2, \ldots\) is said to be pairwise independent if for all \(i \neq j\), \(X_i\) and \(X_j\) are independent.

Example: \(X\) and \(Y\) be independent r.v. each taking values -1 and 1 with probability 1/2 and let \(Z = XY\).

**Theorem 1.** Let \(n\) be a prime number and \(\mathbb{Z}_n\) denote the field of integers modulo \(n\). For \(a\) and \(b\) chosen independently and uniformly at random from \(\mathbb{Z}_n\), let

\[
Y_i = ai + b(mod\ n)
\]

Then for \(i \neq j (mod\ n)\), \(Y_i\) and \(Y_j\) are uniformly distributed on \(\mathbb{Z}_n\) and pairwise independent.

**Proof.** Let \(i \neq j\). Then in the field \(\mathbb{Z}_n\), for any given fixed values of \(y_i\) and \(y_j\) we can solve the equations:

\[
y_i = ai + b(mod\ n)
\]
\[ y_j = aj + b(\text{mod } n) \]

uniquely for \( a \) and \( b \).

\[ a = (y_i - y_j)((i - j)^{-1}\text{mod } n))\text{mod } n \]

\[ b = (y_i - ak)\text{mod } n \]

That is, there is 1-1 correspondence between pairs \((a, b)\) and \((y_i, y_j)\).

If we pick \((a, b)\) uniformly at random, any pair \((y_i, y_j)\) is equally likely.  \(\square\)
Theorem 2. Let $X_1, X_2, \ldots, X_m$ be pairwise independent r.v.s and $X = \sum_{i=1}^{m} X_i$. Then

$$Var(X) = \sum_{i=1}^{m} Var(X_i).$$

Covariance of two random variables is defined as:

$$Cov(X_i, X_j) = E[(X_i - E[X_i])(X_j - E[X_j])]$$

$$= E(X_iX_j) - E[X_i]E[X_j]$$

$$Var(X + Y) = Var(X) + Var(Y) + 2Cov(X_i, X_j)$$

If $X_i$ and $X_j$ are independent $Cov(X_i, X_j) = 0$.

In general, $Var(\sum_{i=1}^{n} X_i) = \sum_{i=1}^{n} Var(X_i) + 2\sum_{i<j} Cov(X_i, X_j)$
We consider a class of randomized algorithms called RP.

An RP algorithm $A$ for deciding whether input strings $x$ belong to a language $L$ is as follows:

Given $x$, $A$ picks a random $r$ from $\mathbb{Z}_n = \{0, \ldots, n-1\}$ for some prime $n$. Computes a binary function $f(x, r)$ such that:

If $x \in L$, then $f(x, r) = 1$ for at least half the possible values of $r$.

If $x \notin L$ then $f(x, r) = 1$ for all choices of $r$.

Repeating the algorithm $t$ times using $t$ random samples from $\mathbb{Z}_n$ the probability of error is at most $2^{-t}$. Uses $\Omega(t \log n)$ random bits.
Probability Amplification

How much can reduction in error by using only 2 samples from $\mathbb{Z}_n$?

Choose $a$ and $b$ independently and uniformly at random.

Let $r_i = ai + b(mod\ n)$ and compute $f(x, r_i), 1 \leq i \leq t$.

What is the error probability?

$Y = \sum_{i=1}^{t} f(x, r_i)$

$E[Y] \geq t/2$ and $Var(Y) \leq t/4$.

$\Pr(Y = 0) \leq \Pr(|Y - E[Y]| \geq t/2) \leq 1/t$.

Improvement over the naive bound of $1/4$. 
Hashing

Given a set of possible keys $U$, such that $|U| = u$ and a table of $m$ entries, a **Hash function** $h$ is a mapping from $U$ to $M = \{1, \ldots, m\}$.

A **collision** occurs when two hashed elements have $h(x) = h(y)$.

**Definition 1.** A hash function $h : U \rightarrow M$ is **perfect** for a set $S$ if it causes no collisions for pairs in $S$.

For any given $S$ such that $|S| \leq m$ there is a perfect hash function.

For any $S$ such that $|S| > m$ there is no perfect hash function.
Chaining

$h(.)$ - hash function.

A table $T[1..m]$ such that $T[k]$ is a pointer to a linked list of all the elements hashed to $T[k]$.

Insert $k$: add $k$ to the linked list $T[h(k)]$.

Search/delete $k$: search (+ delete) in $T[h(k)]$.

The cost is proportional to the length of the link lists.
Hash Functions

\[ h(k) = k \mod m \]
\[ h(k) = (ak + b) \mod m, \]
\[ H = \{h(k) \mid 1 \leq a \leq m - 1, \ 0 \leq b \leq m - 1\} \]

If \( m \) not a prime, let \( p > m \) be a prime

\[ h(k) = (((ax + b) \mod p) \mod m \]
Analysis of Hashing with Chaining

Let $n$ be the number of keys stored in the table.

The load factor $\alpha = \frac{n}{m}$.

Worst case insert time $O(1)$.

Worst case search/delete time $O(n)$.

For simple probabilistic analysis:

**Simple Uniform Assumption:** Keys are hashed to uniformly random and independent locations.

Assume that $h(.)$ is computed in $O(1)$ time.
Theorem 3. In a hash table in which collisions are resolved by chaining, under the assumption of simple uniform hashing,

1. An unsuccessful search takes $\Theta(1 + \alpha)$ expected time.

2. A successful search takes $\Theta(1 + \alpha)$ expected time.
Universal Hash Functions

**Definition 2.** A family $H$ of hash functions from $U$ to $M$ is **universal (2-universal)** if for all $x, y \in U$, such that $x \neq y$, and for a randomly chosen function $h$ from $H$

$$\Pr(h(x) = h(y)) \leq \frac{1}{m}.$$ 

Let $H$ be the set of all functions from $U$ to $M$, then $H$ is universal.

**Problem:** There are $m^u$ functions from $U$ to $M$ - requires $u \log m$ bits to choose, represent and store as a table.
Theorem 4. Assume that we hash \( n \) keys to a table of size \( m \), \( n \leq m \), using a hash function \( h \) chosen at random from a universal family of hash functions, and we resolve collisions by chaining. Then searching for a key takes expected time at most \( 1 + \alpha \).

Proof. For every pair of distinct keys \( k \) and \( l \), \( X_{kl} = 1 \) iff \( h(k) = h(l) \), else 0.

\[
E[X_{kl}] \leq 1/m.
\]

For each key \( k \), define the r.v. \( Y_k \) that equals the number of keys other than \( k \) that hash to the same slot as \( k \):

\[
Y_k = \sum_{l \in T, l \neq k} X_{kl}
\]

Thus \( E[Y_k] = \sum_{l \in T, l \neq k} E[X_{kl}] \leq \sum_{l \in T, l \neq k} 1/m \)

If key \( k \notin T \): \( E[Y_k] \leq n/m = \alpha \)

If key \( k \in T \): \( E[Y_k] \leq (n - 1)/m + 1 < 1 + \alpha \)
Corollary 1. Using universal hashing and collision resolution by chaining in a table with $m$ slots, it takes expected time $\Theta(s)$ to handle any sequence of $s$ INSERT, SEARCH, and DELETE operations containing $O(m)$ INSERT operations.
Constructing universal hash functions

Choose a prime number $p$ such that $0 \leq k \leq p - 1$.

Let $\mathcal{Z}_p = \{0, 1, \ldots, p-1\}$ and $\mathbb{Z}_p^* = \{1, 2, \ldots, p-1\}$.

Let

$$h_{a,b}(k) = ((ak + b) \mod p) \mod m$$

be a hash function for any $a \in \mathbb{Z}_p^*$ and any $b \in \mathcal{Z}_p$.

Define the set of hash functions:

$$\mathcal{H}_{p,m} = \{h_{a,b} : a \in \mathbb{Z}_p^*, b \in \mathcal{Z}_p\}$$
Theorem 5. $\mathcal{H}_{p,m}$ is universal.

Proof. Consider two distinct keys $k$ and $l$ from $\mathbb{Z}_p$. For a given hash function $h_{a,b}$ we let

$$r = (ak + b) \mod p$$
$$s = (al + b) \mod p$$

$r \neq s$.

Moreover, each distinct pair $(a, b)$ with $a \neq 0$ yields a distinct pair $(r, s)$ with $r \neq s$.

Thus the probability that keys $k$ and $l$ collide is equal to the probability that $r = s \pmod{m}$ with $r \neq s$.

For a given $r$, the number of values of $s$ such that $r \neq s$ and $r = s \pmod{m}$ is at most $\left\lceil \frac{p}{m} \right\rceil - 1 \leq \frac{p + m - 1}{m} - 1 = \frac{p - 1}{m}$.

Thus probability that $s$ collides with $r$ is $\leq 1/m$. $\square$
Perfect Hashing

The **worst case** time to search is $O(1)$.

**Definition 1.** A family $H$ of hash functions from $U$ to $M$ is said to be a **perfect hash family** if for each set $S \subseteq U$ of size $|S| \leq |M|$ there exists a hash function $h \in H$ that is perfect for $S$. 
A two-level hashing scheme

1. Use universal hashing (from the class $\mathcal{H}_{p,m}$) to hash a given set of $n$ keys to a table of $m = n$ slots.

2. For $0 \leq j \leq m - 1$:
   - let $n_j$ be the number of elements hashed to slot $j$;
   - use universal hashing again (from the class $\mathcal{H}_{p,n_j^2}$) to hash these in a secondary table of size $n_j^2$.

Theorem 1. The above scheme guarantees a worst case search time of $O(1)$ assuming that the set of keys is static.
Theorem 2. If we store $k$ keys in a hash table of size $k^2$ using universal hashing (from the class $\mathcal{H}_{p,k^2}$), then the probability of there being any collision is $< 1/2$.

Proof. $X$ be the number of collisions.

$$E[X] = \binom{k}{2} \frac{1}{k^2} < 1/2$$

Markov’s inequality gives the result. \qed

Corollary 1. Under the conditions of the above theorem, there exists an hash function that that will not produce any collision. Moreover, such a function can be found in expected constant number of trials.
The Probabilistic method

1. If a random object in a set satisfies some property with positive probability then there is an object in that set that satisfies that property.

2. Any random variable assumes at least one value that is no smaller than its expectation and at least one value that is no greater than its expectation.
**Theorem 3.** In the two-level hashing scheme the expected amount of storage required for all the secondary hash tables is:

\[ E \left[ \sum_{j=0}^{m-1} n_j^2 \right] < 2n \]

**Proof.**

\[
E \left[ \sum_{j=0}^{m-1} n_j^2 \right] = E \left[ \sum_{j=0}^{m-1} (n_j + 2\binom{n_j}{2}) \right] \\
= E \left[ \sum_{j=0}^{m-1} n_j \right] + 2E \left[ \sum_{j=0}^{m-1} \binom{n_j}{2} \right] \\
\leq n + 2\binom{n}{2}/m = n + 2n(n-1)/2n < 2n \quad \square
\]

**Corollary 2.** In the two-level hashing scheme the probability that the total storage used for secondary hash tables exceeds $4n$ is $< 1/2$. 
Moment Generating Function

The moment generating function $M(t)$ of the random variable $X$ is defined for all real values $t$ by

$$M(t) = E[e^{tX}]$$

The $n$th derivative of $M(t)$ is given by

$$M^n(t) = E[X^n e^{tX}] \quad n \geq 1$$

$$M^n(0) = E[X^n] \quad n \geq 1$$
Examples

Binomial Distribution $B(n, p)$:

$M(t) = E[e^{tX}] = \sum_{k=0}^{n} e^{tk} \binom{n}{k} p^k (1 - p)^{n-k}$

$= \sum_{k=0}^{n} \binom{n}{k} (pe^t)^k (1 - p)^{n-k}$

$= (pe^t + 1 - p)^n$

Unit Normal Distribution $N(0, 1)$:

$M(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{tx} e^{-x^2/2} dx$

$= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-(x-t)^2/2 + t^2/2} dx$

$= e^{t^2/2}$

Normal Distribution with parameters $\mu$ and $\sigma^2$, $N(\mu, \sigma^2)$:

$e^{\sigma^2 t^2/2 + \mu t}$
Important Properties of MGFs

1. The Moment generating function of the sum of independent random variables equals the product of the individual moment generating functions.

2. MGF uniquely determines the distribution.
Chernoff Bounds

Let $X$ be a r.v. and $M$ be its MGF.

\[
\Pr(X \geq a) \leq e^{-ta}M(t)
\]
for all $t > 0$

\[
\Pr(X \leq a) \leq e^{-ta}M(t)
\]
for all $t < 0$

If $X$ is the standard normal r.v.

\[
P(X \geq a) \leq e^{-ta} e^{t^2/2} \text{ for all } t > 0
\]

RHS is minimized for $t = a$. Thus, for $a > 0$

\[
P(X \geq a) \leq e^{-a^2/2}
\]

Similarly, for $a < 0$,

\[
\Pr(X \leq a) \leq e^{-a^2/2}
\]
Theorem 4. Let $X_1, X_2, \ldots, X_n$ be independent indicator random variables such that, for $1 \leq i \leq n$, $\Pr[X_i = 1] = p_i$, where $0 < p_i < 1$. Then, for $X = \sum_{i=1}^{n} X_i$, $\mu = E[X] = \sum_{i=1}^{n} p_i$, and any $\delta > 0$, 

$$\Pr(X > (1 + \delta)\mu) < \left(\frac{e^\delta}{(1 + \delta)(1+\delta)}\right)^\mu$$

For $0 < \delta < 1$,

$$\Pr(X > (1 + \delta)\mu) \leq e^{-\mu\delta^2/3}$$

For $R \geq 6\mu$

$$\Pr(X \geq R) \leq 2^{-R}$$

For $0 < \delta < 1$,

$$\Pr(X < (1 - \delta)\mu) < e^{-\mu\delta^2/2}$$
Theorem 1. Let $X_1, X_2, \ldots, X_n$ be independent indicator random variables such that, for $1 \leq i \leq n$, $	ext{Pr}[X_i = 1] = p_i$, where $0 < p_i < 1$. Then, for $X = \sum_{i=1}^{n} X_i$, $\mu = \mathbb{E}[X] = \sum_{i=1}^{n} p_i$, and any $\delta > 0$,

$$\text{Pr} (X > (1 + \delta)\mu) < \left( \frac{e^{\delta}}{(1 + \delta)(1+\delta)} \right)^{\mu}$$

For $0 < \delta < 1$: $\text{Pr} (X > (1 + \delta)\mu) \leq e^{-\mu \delta^2 / 3}$

For $R \geq 6\mu$: $\text{Pr}(X \geq R) \leq 2^{-R}$

For $0 < \delta < 1$,

$$\text{Pr}(X < (1 - \delta)\mu) < e^{-\mu \delta^2 / 2}$$
Proof. Upper tail: For any positive real $t$,

$$\Pr (X > (1 + \delta) \mu) = \Pr (e^{tX} > e^{t(1+\delta)\mu})$$

By Markov’s inequality,

$$\Pr (X > (1 + \delta) \mu) < \frac{E[e^{tX}]}{e^{t(1+\delta)\mu}}$$

$$= \frac{E[e^{t \sum_{i=1}^{n} X_i}]}{e^{t(1+\delta)\mu}} = \frac{E[\prod_{i=1}^{n} e^{tX_i}]}{e^{t(1+\delta)\mu}} = \frac{\prod_{i=1}^{n} E[e^{tX_i}]}{e^{t(1+\delta)\mu}}$$

$$= \frac{\prod_{i=1}^{n} (p_i e^t + 1 - p_i)}{e^{t(1+\delta)\mu}} = \frac{\prod_{i=1}^{n} (1 + p_i (e^t - 1))}{e^{t(1+\delta)\mu}}< \frac{\prod_{i=1}^{n} e p_i (e^t - 1)}{e^{t(1+\delta)\mu}} = \frac{e \sum_{i=1}^{n} p_i (e^t - 1)}{e^{t(1+\delta)\mu}} = \frac{e (e^t - 1) \mu}{e^{t(1+\delta)\mu}}$$

$$\leq \left( \frac{e^\delta}{(1 + \delta)^{1+\delta}} \right)^\mu$$

for $t = \ln(1 + \delta)$

Using $\delta - (1 + \delta) \ln(1 + \delta) \leq -\delta^2/3$ for $0 < \delta < 1$ we get

$$\Pr (X > (1 + \delta) \mu) \leq e^{-\mu \delta^2/3}$$
For $R \geq 6\mu$, $\delta \geq 5$.

\[
Pr(X \geq (1 + \delta)\mu) \leq \left( \frac{e^{\delta}}{(1 + \delta)^{(1+\delta)}} \right)^\mu \\
\leq \left( \frac{e}{6} \right)^R \\
\leq 2^{-R}.
\]

**Lower tail:**

\[
Pr(X < (1 - \delta)\mu) = Pr(e^{-tX} > e^{-t(1-\delta)\mu})
\]

By Markov’s inequality,

\[
Pr(X < (1 - \delta)\mu) < \frac{E[e^{-tX}]}{e^{-t(1-\delta)\mu}}
\]

Similar calculations yield

\[
< \frac{e^{(e^{-t}-1)\mu}}{e^{-t(1-\delta)\mu}}
\]
For \( t = \ln(1/(1 - \delta)) \)

\[
\leq \left( \frac{e^{-\delta}}{(1 - \delta)^{(1-\delta)}} \right)^{\mu}
\]

Since \( (1 - \delta)^{(1-\delta)} > e^{-\delta+\delta^2/2} \) we have

\[
\Pr(X < (1 - \delta)\mu) < e^{-\mu\delta^2/2}
\]

□
Example

Theorem 2. Consider \( n \) coin flips, let \( X \) be the number of heads,

\[
\Pr(|X - \frac{n}{2}| > \frac{1}{2} \sqrt{6n \log n}) \leq \frac{2}{n}
\]

Proof.

\[E[X] = n/2\]

We need

\[
\frac{n}{2} - \frac{1}{2} \sqrt{6n \log n} \leq X \leq \frac{n}{2} + \frac{1}{2} \sqrt{6n \log n}
\]

or

\[X = \frac{n}{2}(1 \pm \sqrt{\frac{6 \log n}{n}})\]

Fixing \( \delta = \sqrt{\frac{6 \log n}{n}} \)

\[
\Pr(X < (1 - \delta)n/2) \leq e^{-\frac{n\delta^2}{2}} \leq 1/n
\]
$$\Pr(X > (1 + \delta)n/2) \leq e^{-\frac{n \delta^2}{2 \cdot 3}} \leq 1/n$$
What is the probability of more than $\frac{3N}{4}$ heads in $N$ coin flips?

1. Using Markov Inequality:

   $$Pr(X \geq 3N/4) \leq 2/3.$$ 

2. Using Chebyshev’s Inequality:

   $$Pr(X \geq 3N/4) \leq 4/N.$$ 

3. Using the Chernoff bound:

   $$Pr(X \geq 3N/4) \leq e^{-\frac{N}{243}}.$$
Randomized Quicksort Revisited

View an execution of the randomized quicksort algorithm as the following binary tree of (sub-)problems.

- The root of this tree is the (initial) problem of sorting a given set of $n$ distinct numbers; and a leaf is a subproblem of sorting a singleton set.

- An internal node of this tree is a subproblem of sorting a set $S$ (of size greater than 1).

- Its left child (if any) is the subproblem of sorting a set $S_1 \subset S$ consisting of elements smaller than the pivot.

- Its right child (if any) is the subproblem of sorting a set $S_2 \subset S$ consisting of elements larger than the pivot.

Thus, a run of a quicksort algorithm is described by the above execution tree.
High Probability Analysis

Theorem 3. Randomized Quicksort runs in $O(n \log n)$ time with high probability, i.e., with probability at least $1 - 1/n^b$, for some constant $b > 1$.

Proof. Suppose the size of the set to be sorted at a particular node is $S$. A node in the execution tree is labeled good if the pivot element divides the set into two parts, each of size not exceeding $2S/3$. Otherwise the node is called bad.

Then we can show that:

1. The probability of a node being labeled good is $1/3$.

2. The number of good nodes in any root to leaf path is bounded by $\log_{3/2} n < c \log n$ for some constant $c$. 
What is the probability that a path of length $ac \log n$ (for some constant $a > 1$) will have at most $c \log n$ good nodes?

The mean $\mu = 1/3(ac \log n)$. Using the Chernoff bound

$$\Pr(X < c \log n) = \Pr(X < (1 - (1 - \frac{3}{a})) \mu)$$

$$\leq e^{-\mu(1-3/a)^2(1/2)} \leq 1/n^2$$

for a suitably large constant $a$.

Thus with probability at most $1/n$ no root to leaf path has length more than $ac \log n$. Since the total work done at each level of the tree is $O(n)$, the running time is bounded by $O(n \log n)$ with high probability. $\square$
Application: Set Balancing

Given an $n \times n$ matrix $A$ with entries in $\{0, 1\}$, let

$$
\begin{pmatrix}
a_{11} & a_{12} & \ldots & a_{1n} \\
a_{21} & a_{22} & \ldots & a_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
a_{n1} & a_{n2} & \ldots & a_{nn}
\end{pmatrix}
\begin{pmatrix}
b_1 \\
b_2 \\
\vdots \\
b_n
\end{pmatrix}
=
\begin{pmatrix}
c_1 \\
c_2 \\
\vdots \\
c_n
\end{pmatrix}.
$$

Find a vector $\bar{b}$ with entries in $\{-1, 1\}$ that minimizes

$$
||A\bar{b}||_\infty = \max_{i=1,\ldots,n} |c_i|.
$$
Theorem 4. For a random vector $\bar{b}$, with entries chosen independently and with equal probability from the set $\{-1, 1\}$,

$$Pr(\|A\bar{b}\|_\infty \geq \sqrt{12n \ln n}) \leq \frac{4}{n}.$$
Consider the $i$-th row $\bar{a}_i = a_{i,1}, \ldots, a_{i,n}$. Let $k$ be the number of 1’s in that row.

If $k \leq \sqrt{12n \ln n}$ clearly $|\bar{a}_i \cdot \bar{b}| \leq \sqrt{12n \ln n}$.

If $k > \sqrt{12n \ln n}$, let

$$X_i = |\{ j \mid a_{i,j} = 1 \text{ and } b_j = 1 \}|$$

and

$$Y_i = |\{ j \mid a_{i,j} = 1 \text{ and } b_j = -1 \}|.$$

Thus, $X_i$ counts the number of $+1$’s in the sum $\sum_{j=1}^n a_{i,j} b_j$, and $Y_j$ counts the number of $-1$’s. Using Chernoff bounds,
\[ Pr \left( X_i \geq \frac{k}{2} \left( 1 + \sqrt{\frac{12 \ln n}{k}} \right) \right) \leq e^{- \left( \frac{k}{2} \right) \left( \frac{1}{3} \right) \left( \frac{12 \ln n}{k} \right)} \leq e^{- 2 \ln n} \]

Hence, for a given row,

\[ Pr \left( |X_i - Y_i| \geq \sqrt{12n \ln n} \right) \leq \frac{2}{n^2} \]

Since there are \( n \) rows, the probability that any row exceeds that bound is bounded by \( \frac{2}{n} \).
Chernoff Bound for Sum of \([-1, +1]\) Random Variables

Theorem 1. Let \(X_1, \ldots, X_n\) be independent random variables with

\[
Pr(X_i = 1) = Pr(X_i = -1) = \frac{1}{2}.
\]

Let \(X = \sum_{1}^{n} X_i\). For any \(a > 0\),

\[
Pr(X \geq a) \leq e^{-a^2/2n}
\]
Proof. For any $t > 0$,

$$E[e^{tX_i}] = \frac{1}{2}e^t + \frac{1}{2}e^{-t}.$$  

$$e^t = 1 + t + \frac{t^2}{2!} + \ldots + \frac{t^i}{i!} + \ldots$$

and

$$e^{-t} = 1 - t + \frac{t^2}{2!} + \ldots + (-1)^i \frac{t^i}{i!} + \ldots$$

Thus,

$$E[e^{tX_i}] = \frac{1}{2}e^t + \frac{1}{2}e^{-t} = \sum_{i \geq 0} \frac{t^{2i}}{2i!}$$

$$\leq \sum_{i \geq 0} \frac{(\frac{t^2}{2})^i}{i!} = e^{t^2/2}$$
\[ E[e^{tX}] = \prod_{i=1}^{n} E[e^{tX_i}] \leq e^{nt^2/2}, \]

\[ Pr(X \geq a) = Pr(e^{tX} > e^{ta}) \leq \frac{E[e^{tX}]}{e^{ta}} \leq e^{t^2n/2-ta}. \]

Setting \( t = a/n \) yields

\[ Pr(X \geq a) \leq e^{-a^2/2n}. \]
By symmetry we also have

**Corollary 1.** Let $X_1, \ldots, X_n$ be independent random variables with

$$\Pr(X_i = 1) = \Pr(X_i = -1) = \frac{1}{2}.$$  

Let $X = \sum_{i=1}^{n} X_i$. Then for any $a > 0$,

$$\Pr(|X| > a) \leq 2e^{-a^2/2n}.$$
Theorem 2. For a random vector $\bar{b}$, with entries chosen independently and with equal probability from the set \{-1, 1\},

$$Pr(\|A\bar{b}\|_\infty \geq \sqrt{4n \ln n}) \leq \frac{2}{n} \tag{1}$$

Proof. Consider the $i$-th row $\bar{a}_i = a_{i,1}, \ldots, a_{i,n}$.

Let $k$ be the number of 1’s in that row.

$$Z_i = \sum_{j=1}^{k} a_{i,j} b_{i,j}.$$ 

If $k \leq \sqrt{4n \ln n}$ then clearly $Z_i$ satisfies the bound.

If $k > \sqrt{4n \ln n}$, the $k$ non-zero terms in the sum $Z_i$ are independent random variables, each with probability $1/2$ of being either +1 or -1.

Using the Chernoff bound:

$$Pr \left\{ |Z_i| > \sqrt{4n \log n} \right\} \leq 2e^{-4n \log n / 2k} \leq \frac{2}{n^2},$$
where we use the fact that $n \geq k$.
What is the probability of more than \( \frac{3N}{4} \) heads in \( N \) coin flips?

1. Using Markov Inequality:

\[
Pr(X \geq 3N/4) \leq 2/3.
\]

2. Using Chebyshev’s Inequality:

\[
Pr(X \geq 3N/4) \leq 4/N.
\]

3. Using the Chernoff bound:

\[
Pr(X \geq 3N/4) \leq e^{-\frac{N}{243}}.
\]
Packet Routing on Parallel Computer

Communication network:

- Nodes - processors, switching nodes.
- edges - communication links.
The $n$-cube:

$N = 2^n$ nodes.

Let $\bar{x} = (x_1, ..., x_n)$ be the number of node $x$ in binary.

Nodes $x$ and $y$ are connected by an edge iff their binary representations differ in exactly one bit.

Bit-wise routing: correct bit $i$ in the $i$-th transition - route has length $n$. 
A permutation communication request: each node is the source and destination of exactly one packet.

Up to one packet can cross an edge per step, each packet can cross up to one edge per step.

What is the time to route an arbitrary permutation on the $n$-cube?
Definition 1. A routing algorithm is oblivious if the path taken by one packet is independent of the source and destinations of any other packets in the system.

Theorem 3. Given an \( n \)-node network with maximum degree \( d \) the routing time of any deterministic oblivious routing scheme is

\[
\Omega\left(\sqrt{\frac{n}{d^3}}\right).
\]
Two phase routing algorithm:

1. Send packet to a randomly chosen destination.

2. Send packet from random place to real destination.

Path: Correct the bits, starting at $x_0$ to $x_{n-1}$.

Any greedy queuing method - if some packet can traverse an edge one does.
Theorem 4. The two phase routing algorithm routes an arbitrary permutation on the $n$-cube in $O(\log N) = O(n)$ parallel steps with high probability.

We focus first on phase 1. We bound the routing time of a given packet $M$.

Let $e_1, \ldots, e_m$ be the $m \leq n$ edges traversed by a given packet $M$ is phase 1.

Let $X(e)$ be the total number of packets that traverse edge $e$ at that phase.

Let $T(M)$ be the number of steps till $M$ finished phase 1.
Lemma 1.

\[ T(M) \leq \sum_{i=1}^{m} X(e_i). \]

\[ E[T(M)] \leq \sum_{i=1}^{m} E[X(e_i)]. \]

Let \( P = (e_1, \ldots, e_m), \ (m \leq n) \) be any path followed by the bit fixing algorithm. Nodes are \( v_0, \ldots, v_m \).

For any path \( P \) let \( T(P) = \sum_{i=1}^{m} X(e_i) \).

We bound the probability that \( T(P) \) is large for any \( P \).
A packet is **active** at a node $v_{i-1}$ if it reaches $v_{i-1}$ and has the possibility of traversing $e_i$.

A packet is active if it is active at some node i.e., reaches, some node of path $P$.

Since traversing $e_i$ “fixes” the $j$-th bit, a packet can cross that edge only in its $j$-th transition.

Assume that $e_i$ connects $v_{i-1} = (a_1, ..., a_{j-1}, a_j, ..., a_n)$ to $v_i = (a_1, .., \bar{a}_j, ..., a_n)$.

Only packets that started in address

$$(*, ..., *, a_j, ...., a_n)$$

can reach edge $v_{i-1}$, before the $j$th bit is fixed and only if their destination addresses are

$$(a_1, ...., a_{j-1}, *, *, ...., *)$$

.
There are $2^{j-1}$ possible packets, each has probability $2^{-(j-1)}$ to reach $v_i$.

Thus expected number of active packets per vertex is 1.

Let $H_{k, i = 1 \ldots N}$ be the indicator r.v. for the packet starting from node $k$ to be active.

$$H = \sum_{i=1}^{N} H_k.$$ 

$$E[H] \leq m \leq n.$$ 

$$\Pr(H \geq 6n \geq 6E[H]) \leq 2^{-6n}$$ 

$$\Pr(T(P) \geq 30n) \leq \Pr(H \geq 6n) + \Pr(T(P) \geq 30n | H < 6n) \leq 2^{-6n} + \Pr(T(P) \geq 30n | H < 6n)$$
Lemma 1. If a packet leaves a path (of another packet) it cannot return to that path in the same phase.

Proof. Leaving a path at the $i$-th transition implies different $i$-th bit, this bit cannot be changed again in that phase. □

Lemma 2. The number of transitions that a packet takes on a given path is distributed $\mathcal{G}(\frac{1}{2})$.

Proof. The packet has probability $1/2$ of leaving the path in each transition. □
Conditioning on $H \leq 6n$, if $Z = \sum_{i=1}^{n} Z_i > 30n$, then we had less than $6n$ successes in $36n$ trails with probability $1/2$ for success.

$$Pr(Z = \sum_{i=1}^{n} Z_i > 30n \mid H \leq 6n) \leq e^{-18n\frac{1}{2}(\frac{2}{3})^2} \leq 2^{-3n-1}.$$ 

For a given packet the probability that it spent more than $30n$ steps in phase 1 is bounded by

$$\leq 2^{-6n} + Pr(T(P) \geq 30n \mid H < 6n) \leq 2^{-3n}.$$ 

Since there are at most $2^{2n}$ possible packets in the hypercube, the probability that there is any packet path with $T(P) \geq 30n$ is at most $2^{2n}2^{-3n} = O(1/N).$
The proof of phase 2 is by symmetry:

The proof of phase 1 argued about the number of packets crossing a given path, no “timing” considerations.

The path from “one packet per node” to random locations is similar to random locations to “one packet per node” in reverse order.

Thus, the distribution of the number of packets that crosses a path of a given packet is the same.

The total number of packet traversals across the edges of any packet path during Phase 1 and 2 together is bounded by $60n$ with probability $O(1/N)$. 
Multi-Commodity Flow

Consider a **directed** network with $s$ sources $x_1, \ldots, x_s$ and $s$ sinks $y_1, \ldots, y_s$.

Node $x_i$ is the source of commodity $i$, $y_i$ is the destination of commodity $i$.

For each edge $e$ there is a capacity bound $T$ on the amount of flow through that edge.

Assume that we earn $r_i$ for each unit of flow of commodity $i$.

Let $F^i$ be the amount of flow of commodity $i$, $0 \leq F^i \leq 1$.

We want to maximize the gain function

$$G = \sum_{i=1}^{s} r_i F^i.$$
Let $IN(v)$ and $OUT(v)$ denote the sets of edges leading into and out of vertex $v$.

Let $F^i_e$ denote the amount of commodity $i$ flowing through $e$.

We can formulate this problem as a linear programming problem:

Maximize $G = \sum_{i=1}^{s} r_i F^i$,

such that:

For any source $x_i$, $\sum_{e \in OUT(x_i)} F^i_e = F_i$,

For any sink $y_i$, $\sum_{e \in IN(y_i)} F^i_e = F_i$,

For any edge $e$, $\sum_{i=1}^{s} F^i_e \leq T$,

for any internal node $v$, and commodity $i$, $\sum_{e \in IN(v)} F^i_e = \sum_{e \in OUT(v)} F^i_e$.

For any edge and commodity $F^i_e \geq 0$. 
Linear Programming

Let \( A \) be an \( k \times n \) matrix, \( \bar{x}, \bar{c}, \bar{b} \in R^n \).

Maximize

\[
G(\bar{x}) = \bar{c}^T \bar{x}
\]

Subject to

\[
A\bar{x} \leq \bar{b}.
\]
Each constraint
\[ \sum_{j=1}^{n} a_{i,j} x_j \leq b_i \]
defines a **half space** of \( \mathbb{R}^n \).

Each
\[ \sum_{j=1}^{n} a_{i,j} x_j = b_i \]
is a **hyperplane**.

A finite intersection of hyperplanes defines a **polyhedron**.

We are looking for a point in the polyhedron defined by
\[ A\bar{x} \leq \bar{b} \]
that maximizes
\[ G(\bar{x}) = \bar{c}^T \bar{x} \]
The Simplex Algorithm

Since the target function is a convex function on a convex set

1. A solution of a linear problem is always in a vertex of the polyhedron.

2. There are no local maxima or minima vertices.
The simplex algorithm:

1. Start at an arbitrary vertex of the polyhedron.

2. If the value of the target function in this vertex is better than in all its neighbors STOP.

3. Move to the neighbor vertex with best target function value.

   The algorithm is finite, and always terminate with the optimal value.

   The number of vertices visited can be exponential in the number of constrains and variables.

   Linear programming problem can be solved efficiently (in polynomial time).

   Integer linear programming (optimal solution with integer values) is $NP$-hard.
Randomized Rounding - Basic Idea

1. Assume that each source generates one unit of commodity.

2. Run the relaxed linear programming problem for $0 \leq F_e^i \leq 1$.

3. The flow of commodity $i$ is divided between some $m$ different (not always disjoint) paths. Let $0 \leq X_j^i \leq 1$ be the amount of commodity using path $j$.

4. Send ALL commodity $i$ through one of the $m$ paths. Choose the path randomly with probabilities $X_1^i, \ldots, X_m^i$. 

For a given edge $e$ the relaxed linear programming solution gives $\sum_{i=1}^{s} F_{e}^i \leq T$.

The expected flow through an edge in the rounded solution is $\leq T$.

The integer solution is the sum of 0-1 independent random variables.

Using the Chernoff bound the probability that any of the $N$ edges has flow more than $O(T + \log N)$ is bounded by $1/N$.

The expected rounded flow of each commodity equal the relaxed flow. Thus,

$$E[\tilde{G}] = G$$
Randomized Rounding - Algorithm

1. Assume that we have one unit of each commodity.

2. Run the relaxed linear programming problem for $0 \leq F_i^e \leq 1$.

3. The integer solution is constructed as follows:

   (a) If $v$ is a source, the commodity exits through vertex $e$ with probability $F_i^e$. 
   (b) If $\tilde{F}_i^e = 1$ and $e$ leads to vertex $v$: The commodity exits vertex $v$ using edge $e'$ chosen with probability

\[
\frac{F_i^{e'}}{\sum_{e \in \text{OUT}(v)} F_i^e}.
\]
Conditional Expectation

Consider the following game: A player rolls a dice. If the outcome of the dice is $i$ it flips $i$ independent coins. The payoff of the game, $X$, is the number of heads in the $i$ coin flips.

What is the expected payoff of this game?

Let $Y$ be the outcome of the dice.

$$E[X \mid Y] = Y/2$$

$$E[X] = E[E[X \mid Y]] = E[Y/2] = \sum_{i=1}^{6} \left( i \frac{1}{2} \right) \frac{1}{6}$$
The conditional expectation of a random variable is defined as

\[ E[Y|Z = z] = \sum_y \Pr(Y = y|Z = z) \]

where the summation is over all \( y \) in the range of \( Y \).

A useful identity is

\[ E[X] = \sum_y \Pr(Y = y)E[X|Y = y] \]

The expression \( E[Y|Z] \) is a random variable \( f(Z) \) that takes the value \( E[Y|Z = z] \) when \( Z = z \).
Theorem 1.

\[ E[X] = E[E[X \mid Y]] \]

Proof.

\[ E[X] = \sum_i iPr(X = i) \]

\[ = \sum_i i(\sum_j Pr(X = i \mid Y = j)Pr(Y = j)) \]

\[ = \sum_j (\sum_i iPr(X = i \mid Y = j))Pr(Y = j) = E[E[X \mid Y]] \]

\[ \square \]
We’ll show by induction on the distance from source \( i \) that for any edge \( e, E[\tilde{F}_e^i] = F_e^i \).

The claim is clearly true for the edges adjacent to source \( i \).

Assuming the the claim is true for all edges leading to vertex \( v \), then the expected flow of commodity \( i \) through vertex \( v \) in the rounded solution is equal to the flow in the relaxed solution.
Let $X_v^i = 0, 1$ be the rounded flow of commodity $i$ through vertex $v$. Let $e'$ be an edge leading out of vertex $v$.

$$E[F_{e'}^i \mid X_v^i] = X_v^i \frac{F_{e'}^i}{\sum_{e \in \text{OUT}(v)} F_e^i}$$

$$E[F_{e'}^i] = E[E[F_{e'}^i \mid X_v^i]]$$

$$= E[X_v^i \frac{F_{e'}^i}{\sum_{e \in \text{OUT}(v)} F_e^i}] = F_{e'}^i$$

since

$$E[X_v^i] = \sum_{e \in \text{IN}(v)} F_e^i = \sum_{e \in \text{OUT}(v)} F_e^i$$

Thus, $E[\tilde{G}] = G$.

What is the probability that the rounded flow on any of the $N$ original edges is more than $6T + 2 \log N$?
$N 2^{-(6T + 2 \log N)} \leq \frac{1}{N}$
Randomized Approximation algorithm

Definition 1. A randomized approximate algorithm has a ratio bound $\rho(n)$, if for any input of size $n$, the optimal solution $C^*(n)$ and the expected cost of the algorithm's solution $C(n)$ satisfy the relation:

$$\max \left[ \frac{C(n)}{C^*(n)}, \frac{C^*(n)}{C(n)} \right] \leq \rho(n).$$
Randomized Rounding

General Idea to approximate:

1. (Try to) Formulate the problem as an Integer linear program.

2. Relax the ILP by relaxing the integer constraint.

3. Solve the relaxed LP.

4. **Randomized Rounding**: Treat the fractional (part) of the solution to the LP as probabilities and round based on the probabilities.

5. Make sure you have feasibility.
Randomized Rounding technique for Minimum Set Cover

The minimum set cover problem: Given a set $U$ with $n$ elements, a collection $F$ of subsets of $U$, and a cost function $c : F \rightarrow \mathbb{R}^+$, find a minimum cost sub-collection of $F$ that covers all elements of $U$.

variable $x_S$ for each set $S \in F$. The ILP is:

minimize $\sum_{S \in F} c(S) x_S$

subject to

$$\sum_{S: e \in S} x_S \geq 1, \ e \in U$$

$$x_S \in \{0, 1\}, \ S \in F$$

LP relaxation:

minimize $\sum_{S \in F} c(S) x_S$

subject to

$$\sum_{S: e \in S} x_S \geq 1, \ e \in U$$
\[ x_S \geq 0, \quad S \in F \]
\[ x_S \leq 1, \quad S \in F \]

**Randomized Rounding:**
Let \( x_S, \quad S \in F \) be an optimal solution to the linear program. Pick a set \( S \in F \) with probability \( x_S \).
Theorem 2. LP relaxation plus randomized rounding gives an $O(\log n)$ approximation algorithm for minimum set cover problem.

Proof. Let $C$ be the collection of sets picked. The expected cost of $C$ is

$$\sum_{S \in F} \Pr[S \text{ is picked }] c_S = \sum_{S \in F} p_S c_S = \text{LOPT} \leq \text{OPT}$$

where LOPT and OPT are the optimal solutions to the LP and set cover respectively.

Consider an element $e \in U$. Suppose $e$ occurs in $k$ sets of $F$. Then

$$\Pr[ e \text{ is covered by } C ] \geq 1 - (1 - 1/k)^k \geq 1 - \frac{1}{e}$$
Pick \( c \log n \) such collections and take their union - call it \( C' \). Then for a suitably large constant \( c \),

\[
\Pr[\text{e is not covered by } C'] \leq \left( \frac{1}{e} \right)^{c \log n} \leq \frac{1}{2n}
\]

Summing over all elements \( e \in U \)

\[
\Pr[\text{some } e \in U \text{ is not covered by } C'] \leq n \times \frac{1}{2n} \leq \frac{1}{2}
\]

Therefore \( C' \) is a set cover with probability at least \( 1/2 \). If \( C' \) is not a set cover repeat the above procedure.

The expected cost is at most \( 2 \times OPT \times c \log n = O(\log n)OPT \). \( \square \)
Back to “Balls Into Bins” Model

Assume that $m$ balls are thrown randomly into $n$ bins, i.e., the location of each ball is independently and uniformly chosen at random from the $n$ possibilities.

- How many randomly thrown balls are needed to fill all the bins? - **Coupon Collector** Problem.

- What is the maximum number of balls in any bin?

- How many bins are empty?

- What is the distribution of the balls in the bins?
Number of Empty Bins

Let $X_i$ be the indicator variable for an empty bin.

$$\Pr(X_i = 1) = (1 - \frac{1}{n})^m \approx e^{-m/n}$$

Let $X$ denote the total number of empty bins.

Then

$$E[X] = \sum_{i=1}^{n} X_i = n(1 - 1/n)^m \approx ne^{-m/n}$$

For any constant $r$, the probability that a bin has $r$ balls is

$$\binom{m}{r} \left( \frac{1}{n} \right)^r (1 - \frac{1}{n})^{m-r} = \frac{1}{r!} \frac{m(m-1)\ldots(m-r+1)}{n^r} (1 - 1/n)^{m-r} \approx \frac{e^{-m/n} (m/n)^r}{r!}$$

which is he expected fraction of bins with exactly $r$ balls.

The number of balls is approximately Poisson distribution with mean $m/n$. 
Application to Geometric Random Graphs

Assume that $n$ points are thrown randomly in a unit square.

The random geometric graph $G(n, r)$ has $n$ nodes corresponding to the $n$ points and two nodes are connected by an edge if they are within a distance of $r$ of each other. We assume the $L_\infty$ norm, i.e., for two points $u = (x_1, y_1)$, $v = (x_2, y_2)$, $d(u, v) = \max(|x_1 - x_2|, |y_1 - y_2|)$.

A popular model for Sensor Networks.
Connectivity of $G(n, r)$

**Theorem 1.** If $r \geq \sqrt{\frac{c\log n}{n}}$, where $c$ is a constant $> 4$, then $G(n, r)$ is connected asymptotically almost surely, i.e., $\Pr(G(n, r) \text{ is connected}) \to 1$ as $n \to \infty$.

**Proof.** Divide the unit square into bins of size $r/2 \times r/2$.

Number of bins is $4/r^2$.

$G(n, r)$ is connected if no bin is empty.

Let $r = \sqrt{\frac{c\log n}{n}}$.

The probability that a bin is empty is

$$(1 - r^2/4)^n \leq e^{-nr^2/4} = n^{-c/4}$$

Let $X$ be the number of empty bins.

$$E[X] = \frac{4}{r^2}n^{-c/4} = \frac{4n}{c\log n}n^{-c/4} \leq n^{1-c/4} \to 0 \text{ if } c > 4.$$ 

$$\Pr(X > 0) \leq E[X] \to 0. \quad \square$$
The Poisson Distribution

A Poisson r.v. $X$ with parameter $\mu$ is given by the following probability distribution:

$$\Pr(X = j) = \frac{e^{-\mu}(\mu)^j}{j!}$$

$$E[X] = Var[X] = \mu$$

The sum of a finite number of independent Poisson r.v. is a Poisson r.v. whose mean is the sum of the means of the independent Poisson r.v.s.
Theorem 1. For a Poisson r.v. $X$ with parameter $\mu$:

1. If $x > \mu$ then $\Pr(X \geq x) \leq \frac{e^{-\mu}(e\mu)^x}{x^x}$.

2. If $x < \mu$ then $\Pr(X \leq x) \leq \frac{e^{-\mu}(e\mu)^x}{x^x}$.

Proof. $E[e^{tX}] = \sum_{k=0}^{\infty} \frac{e^{-\mu}\mu^k}{k!} e^{tk}$

$= e^{\mu(e^t-1)} \sum_{k=0}^{\infty} \frac{e^{-\mu t}(\mu e^t)^k}{k!} = e^{\mu(e^t-1)}$

For any $t > 0$ and $x > \mu$

$\Pr(X \geq x) = \Pr(e^{tX} > e^{tx}) \leq \frac{E[e^{tX}]}{e^{tx}} = e^{\mu(e^t-1)-xt}$

For $t = \ln(x/\mu) > 0$ gives

$\Pr(X \geq x) \leq e^{x-\mu-x\ln(x/\mu)} = \frac{e^{-\mu}(e\mu)^x}{x^x}$

Lower tail bound is similar. □
Alternate Approach

**Theorem 2.** Let $X$ be a Poisson r.v. with mean $\mu$. Then we have the following Chernoff bounds:

For $0 < \delta < 1$,

$$\Pr(X < (1 - \delta)\mu) < e^{-\mu\delta^2/2}$$

For $\delta > 0$,

$$\Pr(X > (1 + \delta)\mu) < \left( \frac{e^\delta}{(1 + \delta)(1+\delta)} \right)^\mu$$

For $0 < \delta < 1$:

$$\Pr(X > (1 + \delta)\mu) \leq e^{-\mu\delta^2/3}$$
Proof

For any fixed $s$,

$$\Pr(X = s) = \lim_{n \to \infty} \Pr(B(n, \frac{\mu}{n}) = s).$$

$$\Pr(B(n, \frac{\mu}{n}) = s) = \binom{n}{s} \left(\frac{\mu}{n}\right)^s (1 - \frac{\mu}{n})^{n-s}$$

$$= \frac{n(n-1)\ldots(n-s+1)}{s!} \left(\frac{\mu}{n}\right)^s \frac{(1-\frac{\mu}{n})^n}{(1-\frac{\mu}{n})^s}$$

$$= \frac{(\mu)^s}{s!} e^{-\mu} \text{ for } n \to \infty.$$

Use Chernoff bounds for the sum of indicator r.v.s.
Poisson Approximation

Let $m$ balls be thrown into independently and uniformly at random and let $X_i^m$ be the number of balls in the $i$-th bin, $1 \leq i \leq n$.

Let $Y_1^m, \ldots, Y_n^m$ be independent Poisson r.v.s with mean $m/n$.

**Theorem 3.** The distribution of $(Y_1^m, \ldots, Y_n^m)$ conditioned on $\sum_i Y_i^m = k$ is the same as $(X_1^k, \ldots, X_n^k)$ regardless of the value of $m$.

**Proof.** The probability that $(X_1^k, \ldots, X_n^k) = (k_1, \ldots, k_n)$ for any $k_1, \ldots, k_n$ satisfying $\sum_i k_i = k$ is given by

$$\frac{1}{n^k} \binom{k}{k_1,k_2,\ldots,k_n} = \frac{k!}{k_1!k_2!\ldots k_n!n^k}$$

$$\Pr((Y_1^m, \ldots, Y_n^m) = (k_1, \ldots, k_n) | \sum_{i=1}^n Y_i^m = k) = \frac{\Pr((Y_1^m=k_1) \cap \ldots \cap (Y_n^m=k_n))}{\Pr(\sum_{i=1}^n Y_i^m=k)}$$
\[
\begin{align*}
&= \left( \prod_{i=1}^{n} \frac{e^{-m/n} (m/n)^{k_i}}{k_i!} \right) \frac{k!}{e^{-m} m^k} \\
&= \frac{k!}{k_1! k_2! \ldots k_n! n^k} \quad \square
\end{align*}
\]
Theorem 4. Let \( f(x_1, \ldots, x_n) \) be a non-negative function. Then

\[
E[f(X_1^m, \ldots, X_n^m)] \leq e\sqrt{m}E[f(Y_1^m, \ldots, Y_n^m)]
\]

Proof.

\[
E[f(Y_1^m, \ldots, Y_n^m)] = \\
\sum_{k=0}^{\infty} E[f(Y_1^m, \ldots, Y_n^m) | \sum_{i=1}^{n} Y_i^m = k] \Pr(\sum_{i=1}^{n} Y_i^m = k) \\
\geq E[f(Y_1^m, \ldots, Y_n^m) | \sum_{i=1}^{n} Y_i^m = m] \Pr(\sum_{i=1}^{n} Y_i^m = m) \\
= E[f(X_1^m, \ldots, X_n^m)] \Pr(\sum_{i=1}^{n} Y_i^m = m) \\
\geq E[f(X_1^m, \ldots, X_n^m)] \frac{m^m e^{-m}}{m!}
\]

Using \( m! < e\sqrt{m}(m/e)^m \) we have the result. \( \square \)
Relationship between Bernoulli and Poisson Models

**Bernoulli Model**: $m$ balls are thrown into $n$ bins independently and uniformly at random.

**Poisson Model**: Each of the $n$ bins receive balls determined by independent Poisson r.v.s with mean $m/n$.

**Corollary 1.** Any event that takes place with probability $p$ in the Poisson model takes place with probability at most $pe^{\sqrt{m}}$ in the Bernoulli model.
Theorem 5. When \( n \) balls are thrown independently and uniformly at random into \( n \) bins, the maximum load is at least \( \frac{\ln n}{\ln \ln n} \) with probability at least \( 1 - \frac{1}{n} \) for \( n \) sufficiently large.

Proof. Consider the Poisson model. The probability that no bin has load at least \( M \) is at most

\[
(1 - \frac{1}{eM!})^n \leq e^{-n/(eM!)} \leq \frac{1}{n^2}
\]

if \( M! \leq \frac{n}{2e\ln n} \) i.e., \( \ln(M!) \leq \ln n - \ln \ln n - \ln(2e) \).

When \( n \) and hence \( M = \ln n/(\ln \ln n) \) are suitably large:

\[
M! \leq e\sqrt{M}(M/e)^M \leq M(M/e)^M
\]

\[
\ln M! \leq M \ln M - M + \ln M
\]

\[
= \frac{\ln n}{\ln \ln n} (\ln \ln n - \ln \ln \ln n) - \ln n/(\ln \ln n) + (\ln \ln n - \ln \ln \ln n)
\]
\[ \leq \ln n - \ln n / (\ln \ln n) \]
\[ \leq \ln n - \ln \ln n - \ln(2e). \]
The Coupon Collector Problem

Assume that there are $n$ distinct coupons. Every time you go to a shop you collect a random coupon i.e., every coupon is equally likely to show up. How many coupons you have to collect till you have all the $n$ distinct coupons?

**Theorem 6.** 1. The expected number of coupons needed to collect all the $n$ distinct coupons is $n \ln n + O(n)$.

2. With probability at least $1 - 1/n$, the number of coupons needed is at most $2n \ln n$.

3. Further, with probability at least $1 - 1/n$ no coupon is collected more than $9 \ln n$ number of times.
Proof.

1. Let $X =$ number of coupons needed.

Let $X_i =$ number of coupons from the time $i - 1$ coupons are collected till a new coupon is got.

$$X = \sum_{i=1}^{n} X_i.$$ 

The probability of getting the $i$th new coupon, given that $i - 1$ coupons are already collected is

$$p_i = 1 - \frac{i - 1}{n}$$

$$E[X_i] = \frac{1}{p_i} = \frac{n}{n - i + 1}$$

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

2. The probability that coupon $i$ is not collected after $3n \ln n$ number of trials is:

$$\Pr(E_i) = \left(1 - \frac{1}{n}\right)^{3n \ln n} \leq e^{-3 \ln n} \leq 1/n^3$$
The probability that no coupon is collected is bounded by

\[
\text{Pr}(\bigcup_{i=1}^{n} E_i) \leq \sum_{i=1}^{n} \text{Pr}(E_i) \leq 1/n^2 \leq 1/n
\]

3. The expected number of times a particular coupon is collected in \(3n \ln n\) trials is \(3 \ln n\).

Using the Chernoff bound, with probability at least \(1 - 1/n^3\), this coupon was not collected more than \(9 \ln n\) times. Summing up the probabilities for \(n\) coupons we have the result.

\(\square\)
Sharp Threshold for Coupon Collector

**Theorem 7.** Let $X$ be the number of coupons observed before obtaining one of each of $n$ types of coupons. Then for any constant $c$,

$$\lim_{n \to \infty} \Pr(X > n \ln n + cn) = 1 - e^{-e^{-c}}$$
Theorem 1. Let $X$ be the number of coupons observed before obtaining one of each of $n$ types of coupons. Then for any constant $c$,

$$\lim_{n \to \infty} \Pr(X > n \ln n + cn) = 1 - e^{-e^{-c}}$$
Proof

Assume that the number of balls in each bin is a Poisson r.v. with mean $\ln n + c$.

Then the probability that a specific bin is empty is:

$$e^{-(\ln n + c)} = e^{-c}/n$$

And the probability that no bin is empty (denoted by the event $E$) is:

$$(1 - e^{-c}/n)^n = e^{-e^{-c}} \text{ for } n \to \infty.$$ 

That is, $\lim_{n \to \infty} \Pr(E) = e^{-e^{-c}}$.

We show that the Poisson assumption is accurate enough.
The mean total number of balls is $m = n \ln n + cn$ in the Poisson assumption.

\[
\Pr(E) = \Pr(E||X - m| \leq \sqrt{2m \ln m}) \Pr(|X - m| \leq \sqrt{2m \ln m}) \\
+ \Pr(E||X - m| > \sqrt{2m \ln m}) \Pr(|X - m| > \sqrt{2m \ln m})
\]

Using Chernoff bounds, we can show that

\[
\Pr(|X - m| > \sqrt{2m \ln m}) = o(1)
\]

Next we will show that

\[
|\Pr(E||X - m| \leq \sqrt{2m \ln m}) - \Pr(E|X = m)| = o(1).
\]

Then,

\[
\Pr(E) = \Pr(E||X - m| \leq \sqrt{2m \ln m})(1 - o(1)) + o(1) \\
= \Pr(E|X = m)(1 - o(1)) + o(1)
\]
Hence

$$\lim_{n \to \infty} \Pr(E) = \lim_{n \to \infty} \Pr(E|X = m)$$
To show:

$$|\Pr(E|X - m| \leq \sqrt{2m \ln m}) - \Pr(E|X = m)| = o(1).$$

Since $\Pr(E|X = k)$ is increasing in $k$, it is enough to show that

$$\Pr(E|X = m + \sqrt{2m \ln m}) - \Pr(E|X = m - \sqrt{2m \ln m}) = o(1).$$

This is equivalent to the probability the additional $2\sqrt{2m \ln m}$ balls will cover all empty bins left after throwing $m - \sqrt{2m \ln m}$ balls.

This is at most $2\sqrt{2m \ln m} \frac{1}{n} = o(1)$. 


Random Graphs

A probabilistic model of graphs for probabilistic analysis of graph algorithms.

\( G(n, p) \) random graph model: Given a \( n \)-vertex labeled graph, an (undirected) edge occurs between a pair of vertices with probability \( p \), independently of other pairs.

The probability of a graph \( G \) of \( k \) edges is given by

\[
p^k (1 - p) \binom{n}{2}^{-k}
\]

The probability that a graph \( G \in G(n, p) \) contains \( k \) edges is binomially distributed:

\[
\binom{n}{k} p^k (1 - p) \binom{n}{2}^{-k}
\]
Algorithm for finding a Hamiltonian Path

1. Choose an arbitrary vertex \( x_0 \) to start the path. 
   \[ HEAD = x_0. \]

2. Repeat until all vertices are connected:
   
   (a) Choose a random vertex (say \( u \)) from HEAD’s adjacency list.
   (b) remove \((HEAD, u)\) from HEAD’s and \(u\)’s lists.
   (c) \textbf{If} \( u \) is not in the path then \( HEAD = u; \) 
       \textbf{else} use the edge to ROTATE path and set HEAD to the successor of \( u \) in the path.
Algorithm for finding a Hamiltonian Path

1. Choose an arbitrary vertex $x_0$ to start the path. 
   $HEAD = x_0$.

2. Repeat until all vertices are connected:
   (a) Choose a random vertex (say $u$) from HEAD’s adjacency list.
   (b) remove $(HEAD, u)$ from HEAD’s and $u$’s lists.
   (c) If $u$ is not in the path then $HEAD = u$; 
       else use the edge to ROTATE path and set HEAD to the successor of $u$ in the path.
Rotation

Let

\[ P = v_1, v_2, \ldots, v_k \]

be a simple path in an undirected graph \( G \). Let \((v_k, v_i)\) be an edge of \( G \). Then

\[ P' = v_1, \ldots, v_i, v_k, v_{k-1}, \ldots, v_{i+2}, v_{i+1} \]

is a rotated simple path in \( G \).
Theorem 1. If no adjacency list is empty before termination, then whp:

1. The algorithm finds a Hamiltonian path;

2. The algorithm terminates after $O(n \log n)$ iterations of the while loop;

3. No more than $a \log n$ edges (for some constant $a > 0$) are removed from any list.
Proof. Consider a "less efficient" algorithm that for each vertex $u$ keeps two lists:

1. $\text{new\_edges}(u)$ - adjacent edges that were not used yet;
2. $\text{old\_edges}(u)$ - edges that were already used.

When $u$ is at the head of the path we choose:

1. A random element in $\text{old\_edges}(u)$ with probability $\frac{|\text{old\_edges}(u)|}{n}$.
2. $u$ with probability $1/n$.
3. with probability $1 - \frac{1}{n} - \frac{\text{old\_edges}(u)}{n}$, a random element in $\text{new\_edges}(u)$ (and move it to $\text{old\_edges}(u)$).

Make the successor (in the path) of the chosen node to be the new head by rotation. (In case $u$ is chosen, then the successor is the last vertex in the path).
The number of iterations of the modified algorithm stochastically dominates the number of iterations of the original algorithm i.e.,

$$\forall x \; \Pr(I_{mod} > x) \geq \Pr(I_{orig} > x)$$

The number of edges removed from a $new\_edges(u)$ by the modified algorithm has the same distribution as in the original algorithm.

The probability that a given vertex becomes HEAD at a given iteration of the modified algorithm is $1/n$.

By the coupon collector analysis, if no list is empty before termination, then whp:

1. The algorithm terminates with an Hamiltonian path in $3n \ln n$ iterations.

2. No vertex is chosen more than $9 \ln n$ times.
Corollary 1. If \( p \geq \frac{36 \log n}{n} \) and \( G \in G(n, p) \), then whp \( G \) has a Hamiltonian path and the algorithm will find it.

Proof. What should be \( p \) so that whp every vertex has more than \( 18 \ln n \) neighbors?

The expected number of neighbors for a vertex is \( (n - 1)p \); using the Chernoff bound and the union bound we can show that if \( p \geq \frac{36 \ln n}{n} \), then every vertex has more than \( 18 \ln n \) neighbors. \( \square \)
The Probabilistic Method

1. If a random object in a set satisfies some property with positive probability then there is an object in that set that satisfies that property.

2. If \( E[X] = C \), then there are values \( c_1 \leq C \) and \( c_2 \geq C \) such that \( Pr(X = c_1) > 0 \) and \( Pr(X = c_2) > 0 \).
Monochromatic Complete Subgraphs

Given a complete graph on 1000 vertices, can you color the edges in two colors such that no clique of 20 vertices is monochromatic?

**Theorem 1.** If \( n \leq 2^{k/2} \) then it is possible to edge color the edges of a complete graph on \( n \) points \( (K_n) \), such that is has no monochromatic \( K_k \) subgraph.
**Proof.** Consider a random coloring.

For a given set of $k$ vertices, the probability that the clique defined by that set is monochromatic is bounded by

$$2 \times 2^{-\binom{k}{2}}.$$

There are $\binom{n}{k}$ such cliques, thus the probability that any clique is monochromatic is bounded by

$$\binom{n}{k} 2 \times 2^{-\binom{k}{2}} \leq \frac{n^k}{k!} \times 2^{-k(k+1)/2+1}$$

$$\leq 2^{(k/2)+1} \frac{1}{k!} < 1.$$

Thus, there is a coloring with the required property. \qed
Maximum Satisfiability

Given \( m \) clauses in CNF (Conjunctive Normal Form), assume that no clause contains a variable and its complement.

**Theorem 2.** For any set of \( m \) clauses there is a truth assignment that satisfy at least \( m/2 \) of the clauses.

**Proof.** Assign random values to the variables. The probability that a given clause (with \( k \) literals) is not satisfied is bounded by

\[
1 - 2^{-k} \geq \frac{1}{2}.
\]

\( \Box \)
**Theorem 3.** Given any undirected graph $G = (V, E)$ with $n$ vertices and $m$ edges, there is a partition of $V$ into two disjoint sets $A$ and $B$ such that at least $m/2$ edges connect a vertex in $A$ to a vertex in $B$.

**Proof.** Construct sets $A$ and $B$ by randomly assign each vertex to one of the two sets.

The probability that a given edge connect $A$ to $B$ is $1/2$, thus the expected number of such edges is $m/2$.

Thus, there exists such a partition. □
A Las Vegas Algorithm

We lower bound the probability that the cut produced is of value at least $m/2$.

Let $C(A, B)$ be the set constructed by the random partition. Then $p = \Pr(C(A, B) \geq m/2)$.

$$E[C(A, B)] = \sum_{i \leq m/2 - 1} i \Pr(C(A, B) = i)$$

$$+ \sum_{i \geq m/2} i \Pr(C(A, B) = i)$$

$$\leq (1 - p)(m/2 - 1) + pm$$

which gives

$$p \geq \frac{1}{m/2 + 1}$$
Derandomization

Using "The Method of Conditional Probabilities" we derandomize a randomized algorithm to construct a deterministic algorithm.

Let $C(A, B)$ be the cut generated by a random partition.

Initially let $A$ and $B$ be empty.

Place the vertices one by one deterministically in some order, say $v_1, \ldots v_n$ according to the following rule:

Place $v_{k+1}, k \geq 1$, in the set $x_{k+1}$ (either $A$ or $B$) such that

$$E[C(A, B)|x_1, x_2, \ldots, x_k]$$

$$\leq E[C(A, B)|x_1, x_2, \ldots, x_{k+1}]$$

How to find $x_{k+1}$?
Suppose you place \( v_{k+1} \) with equal probability in \( A \) or \( B \). Let \( Y_{k+1} \) denote where it is placed. Then,

\[
E[C(A, B) | x_1, x_2, \ldots, x_k] =
\]

\[
\frac{1}{2} E[C(A, B) | x_1, x_2, \ldots, x_k, Y_{k+1} = A]
\]

\[
+ \frac{1}{2} E[C(A, B) | x_1, x_2, \ldots, x_k, Y_{k+1} = B]
\]

Thus,

\[
\max(E[C(A, B) | x_1, x_2, \ldots, x_k, Y_{k+1} = A],
\]

\[
E[C(A, B) | x_1, x_2, \ldots, x_k, Y_{k+1} = B])
\]

\[
\geq E[C(A, B) | x_1, x_2, \ldots, x_k]
\]

Choose the set which gives a larger conditional expectation.
**Theorem 1.** Let $G = (V, E)$ be a graph on $n$ vertices with $dn/2$ edges. Then $G$ has an independent set with at least $n/2d$ vertices.

**Proof:**

Consider the following randomized algorithm:

1. Delete each vertex of $G$ (and its incident edges) independently with probability $1 - 1/d$.

2. For each remaining edge, remove it and one of its adjacent vertices.
"Sample and Modify" Technique.

Let $X$ denote the number of vertices that survive step 1.

$E[X] = n/d$.

Let $Y$ be the number of edges that survive the first step.

$E[Y] = \frac{nd}{2} \left( \frac{1}{d} \right)^2 = \frac{n}{2d}$

The algorithm outputs a set of size at least $X - Y$, so

$E[X - Y] = \frac{n}{2d}$
Dominating Set

A dominating set of an undirected graph \( G = (V, E) \) is a set \( U \subseteq V \) such that every vertex \( v \in V - U \) has at least one neighbor in \( U \).

Given an arbitrary graph, finding the minimum dominating set is NP-complete.

**Theorem 2.** Let \( G = (V, E) \) be a graph on \( n \) vertices, with minimum degree \( \delta > 1 \). Then \( G \) has a dominating set of size at most \( n \frac{1 + \ln(\delta + 1)}{\delta + 1} \).

**Proof:**

For some \( p \in [0, 1] \) pick randomly and independently each vertex of \( V \) with probability \( p \).

Let \( X \) be the set of vertices picked.

\[
E[|X|] = np.
\]

Let \( Y_X \) denote the set of vertices in \( V - X \) that do not have any neighbor in \( X \).
\[ E[|Y_X|] \leq n(1 - p)^{\delta + 1} \]
\[ E[|X| + |Y_X|] \leq np + n(1 - p)^{\delta + 1} \]

Thus there is at least one choice of \( X \subseteq V \) such that \( U = X \cup Y_X \) which is a dominating set of at most this size.

We can optimize \( p \). A crude bound is:

\[ |U| \leq np + ne^{-p(\delta + 1)} \]

The RHS is minimized for \( p = \frac{\ln(\delta + 1)}{\delta + 1} \).
High Girth and High Chromatic Number

Girth is the length of the smallest cycle in the graph.

Chromatic number is the minimum number of colors needed to color the vertices of the graph such that no two adjacent vertices have the same color.

**Theorem 3.** For all $k, l$ there exists a graph $G$ with $\text{girth}(G) > l$ and $\chi(G) > k$.

**Proof:** If $\alpha(G)$ is the size of the largest independent set of $G$ then $\chi(G)\alpha(G) \geq n$. We want to show that there exists a graph with small $\alpha$ and large girth.

Fix $\theta < 1/l$ and choose a random graph $G \in G(n, p)$ with $p = n^{\theta - 1}$.

Let $X$ be the number of cycles of size at most $l$.

$$E[X] = \sum_{i=3}^{l} \binom{n}{i} \frac{(i-1)!}{2} p^i$$

$$\leq \sum_{i=3}^{l} n^i p^i = \sum_{i=3}^{l} n^{\theta i} = o(n)$$
since $\theta l < 1$.

$$\Pr(X \geq n/2) = o(1)$$

Set $x = \frac{3}{p} \ln n$.

$$\Pr(\alpha(G) \geq x) \leq \binom{n}{x} (1-p)^{\left(\frac{x}{2}\right)} \leq [ne^{-p(x-1)/2}]^x = o(1)$$

Let $n$ be sufficiently large so that both these events happen with probability less than $1/2$.

Then there exists a graph $G$ with less than $n/2$ cycles of length at most $l$ and with $\alpha(G) < \frac{3}{p} \ln n = 3n^{1-\theta} \ln n$.

Remove from $G$ a vertex from each cycle of length at most $l$. This gives a graph $G'$ with at least $n/2$ vertices.

$G'$ has girth $> l$ and $\alpha(G') \leq \alpha(G)$. Thus

$$\chi(G') \geq \frac{n/2}{3n^{1-\theta} \ln n} = \frac{n^\theta}{6 \ln n} > k$$

for a sufficiently large $n$. 

6
First and Second Moment Method

**Theorem 1.** Let $X$ be a non-negative integer-valued r.v. Then

\[
\Pr(X \geq 1) \leq E(X)
\]

\[
\Pr(X = 0) \leq \frac{Var(X)}{(E(X))^2} = \frac{E(X^2)}{(E(X))^2} - 1
\]

**Proof:**

\[
\Pr(X = 0) \leq \Pr(|X - E(X)| \geq E(X)) \leq \frac{Var(X)}{(E(X))^2}
\]

These help in showing $\Pr(X = 0) \to 1$ and $\Pr(X = 0) \to 0$. 
Thresholds in Random Graphs

**Theorem 2.** For the $G(n, p)$ random graph model, let $p = c \frac{\log n}{n}$. If $c > 1$ then almost all graphs have no isolated vertices and if $c < 1$ almost all graphs have at least one isolated vertex.

**Proof:**

Upper Threshold: Let $X$ denote the number of isolated vertices in a random $G \in G(n, p)$.

Let $X_i$ be the indicator r.v. for a vertex to be isolated.

$$E[X] = \sum_{i=1}^{n} X_i = n(1 - p)^{n-1}$$

$$(1 - p)^n = e^{n \log(1-p)} = e^{n(-p - p^2/2 - p^3/3 ...)}$$

$$= e^{-np} e^{-np^2(1/2+p/3+...)}$$

$$\sim e^{-np} \text{ provided } np^2 \rightarrow 0.$$  

$$E[X] = n(1 - p)^{n-1} \sim ne^{-np} \sim n^{1-c}$$
Thus, $E(X) \to 0$ if $c > 1$ and

$E(X) \to \infty$ if $c < 1$. 
Lower Threshold:

\[ E[X^2] = \sum_{i=1}^{n} E(X_i^2) + \sum_{i \neq j} X_i X_j \]

\[ = E(X) + n(n - 1)E(X_1X_2) \]

\[ = E(X) + n(n - 1)(1 - p)^{2(n-2)+1} \]

\[ \Pr(X = 0) \leq \frac{E(X^2)}{(E(X))^2} - 1 \]

\[ = \frac{1}{E(X)} + \frac{n(n-1)(1-p)^{2(n-2)+1}}{n^2(1-p)^{2(n-1)}} - 1 \]

\[ \sim \frac{1}{E(X)} \rightarrow 0 \text{ if } c < 1. \]
The Lovasz Local Lemma

Let $A_1, \ldots, A_n$ be a set of “bad” events. We want to show that

$$Pr(\bigcap_{i=1}^{n} \bar{A}_i) > 0.$$ 

1. If $\sum_{i=1}^{n} Pr(A_i) < 1$ then $Pr(\bigcap_{i=1}^{n} \bar{A}_i) > 0$.

2. If all the $A_i$’s are mutually independent and for all $i$ $Pr(A_i) < 1$ then $Pr(\bigcap_{i=1}^{n} \bar{A}_i) > 0$.

3. If each $A_i$ depends only on a few other events: The Lovasz Local Lemma.
Definition 1.  An event $E$ is mutually independent of the events $E_1, \ldots, E_n$, if for any $T \subset [1, \ldots, n],$

$$Pr(E \mid \cap_{j \in T} E_j) = Pr(E).$$

Definition 2.  A dependency graph for a set of events $E_1, \ldots, E_n$ has $n$ vertices $1, \ldots, n$. Events $E_i$ is mutually independent of any set of events $\{E_j \mid j \in T\}$ iff there is no edge in the graph connecting $i$ to any $j \in T$. 
Theorem 3. Let $E_1, \ldots, E_n$ be a set of events. Assume that

1. For all $i$, $\Pr(E_i) \leq p$;

2. The degree of the dependency graph is bounded by $d$.

3. $4dp \leq 1$

then

$$\Pr(\cap_{i=1}^{n} \bar{E}_i) > 0.$$
Proof

Let $S \subset \{1, \ldots, n\}$. We prove by induction on $s = 1, \ldots, n$ that if $|S| \leq s$, for all $k$

$$\Pr(E_k \mid \cap_{j \in S} E^c_j) \leq 2p$$

For $s = 0$, $S = \emptyset$, trivial.

W.l.o.g. renumber so that $S = \{1, \ldots, s\}$, and $(k, j)$ is not an edge of the dependency graph for $j > d$. 


\[
\Pr(E_k|E_1^c, \ldots, E_s^c) = \frac{\Pr(E_k E_1^c \ldots E_s^c)}{\Pr(E_1^c \ldots E_s^c)} = \frac{\Pr(E_k E_1^c \ldots E_d^c|E_{d+1}^c \ldots E_s^c)}{\Pr(E_1^c \ldots E_d^c|E_{d+1}^c \ldots E_s^c)}
\]

\[
\Pr(E_k E_1^c \ldots E_d^c|E_{d+1}^c \ldots E_s^c) \leq \Pr(E_k|E_{d+1}^c \ldots E_s^c) = \Pr(E_k) \leq p
\]

Using the induction hypothesis:

\[
\Pr(E_1^c \ldots E_d^c|E_{d+1}^c \ldots E_s^c) \geq 1 - \sum_{i=1}^{d} \Pr(E_i|E_{d+1}^c \ldots E_s^c) \geq 1 - \sum_{i=1}^{d} 2p \geq 1 - 2pd \geq 1/2
\]

Thus,

\[
\Pr(E_k|E_1^c, \ldots, E_s^c) \leq \frac{p}{1/2} = 2p
\]
\[
\Pr(E_1^c \ldots E_n^c) = \prod_{i=1}^{n} \Pr(E_i^c | E_1^c \ldots E_{i-1}^c)
\]

\[
= \prod_{i=1}^{n} (1 - \Pr(E_i | E_1^c \ldots E_{i-1}^c)) \geq \prod_{i=1}^{n} (1 - 2p) > 0
\]
Given a network, and $n$ pairs of users, we want to find $n$ edge-disjoint paths between the $n$ pairs. Assume that each pair, $i = 1, \ldots, n$ can choose a path from a collection $F_i$ of $m$ paths. Then,

**Theorem 1.** If for each $i \neq j$, any path in $F_i$ shares edges with no more than $k$ paths in $F_j$, where $8nk/m \leq 1$, then there is a way to choose $n$ edge-disjoint paths connecting the $n$ pairs.

**Proof:** Consider the probability space defined by each pair choosing a path independently and uniformly at random from its set of $m$ paths.

$E_{ij}$: event that the paths chosen by pairs $i$ and $j$ share at least one edge.

$p = \Pr(E_{ij}) \leq k/m.$

The degree of the dependency graph of the events $d < 2n$. 
Since $4dp < 8nk/m \leq 1$, by the Lovasz Local Lemma:

$$\Pr(\cap_{i \neq j} E_{i,j}^c) > 0.$$
Satisfiability

**Theorem 2.** Consider a CNF formula with $k$ literals per clause. Assume that each variable appears in no more than $T = \frac{2^k}{4k}$ clauses, then the formula has a satisfying assignment.
Proof. Assume that the formula has $m$ clauses.

For $i = 1, \ldots, m$, let $E_i$ be the event “The random assignment does not satisfy clause $i$”.

$$Pr(E_i) = \frac{1}{2^k}.$$  

The event $E_i$ is mutually independent of all the events related to clauses that do not share variables with clause $i$.

The degree of $E_i$ in the dependency graph is bounded by $kT$.

Since

$$4dp \leq 4kT2^{-k} = 4k\frac{2^k}{4k}2^{-k} = 1$$

$$Pr(\overline{E_i}, \ldots, \overline{E_m}) > 0.$$  

\[\square\]
Algorithm

Assume $m$ clauses, $\ell$ variables, each clause has $k$ literals, each variable appears in no more than $T = 2^{\alpha k}$ clauses.

**First Part:**

A clause is **Dangerous** at a given step if both

1. The clause is not satisfied;

2. At least $k/2$ of its variables were fixed.

For $i = 1$ to $\ell$
If $x_i$ is not in a dangerous clause assign it a random value in $\{0, 1\}$. 
A **surviving clause** is a clause that is not satisfied at the end of phase one.

A surviving clause has no more than \( k/2 \) of its variables fixed.

A **deferred** variable is a variable that was no assigned value in the first part.

**Lemma 1.** There is an assignment of values to the deferred variables such that all the surviving clauses are satisfied (thus the formula is satisfied).
Lemma 2. Let $G'$ be the dependency graph on the surviving clauses. With high probability all connected components in $G'$ have size $O(\log m)$.

Part Two:

Using exhaustive search assign values to the deferred variable to complete the truth assignment for the formula.

If a connected component has $O(\log m)$ clauses it has $O(k \log m)$ variables. Assuming $k = O(1)$ we can check all assignments in polynomial in $m$ number of steps.
Lemma 3. There is an assignment of values to the deferred variables such that all the surviving clauses are satisfied (thus the formula is satisfied).

Proof. At the end of the first phase we have $m'$ “surviving clauses’ (all the rest are satisfied), each surviving clause has at least $k/2$ deferred variables.

Consider a random assignment of the deferred variables.

Let $E_i$ be the event clause $i$ (of the surviving clauses) is not satisfied.

\[ p = Pr(E_i) \leq 2^{-k/2}. \]
The degree of the dependency graph is bounded by

\[ d = kT \leq k2^{\alpha k}. \]

Since

\[ 4dp = 4k \cdot 2^{\alpha k/2} \cdot 2^{-k/2} \leq 1 \]

there is a satisfying assignment of the deferred variables that (together with the assignment of the other variables) satisfies the formula. \( \square \)
Lemma 4. Let $G'$ be the dependency graph on the surviving clauses. With high probability all connected components in $G'$ have size $O(\log m)$.

Proof.

Assume that there is a connected component $R$ of size $r = |R|$.

Since the degree of a vertex in $R$ is bounded by $d$, there must be a set $T$ of $t = r/d^3$ vertices in $R$ which are at distance at least 4 from each other.

A clause “survives” the first part if it is at distance at most 1 from a dangerous clause. Thus, for each clause in $T$ there is a distinct dangerous clause, and these dangerous clauses are at distance 2 from each other.

The probability that a given clause is dangerous is at most $2^{-k/2}$.

The probability that a clause survives is at most $(d + 1)2^{-k/2}$. 


These events are independent for vertices in \( T \). Thus the probability of a particular connected component of \( r \) vertices is bounded by

\[
\left((d + 1)2^{k/2}\right)^{r/d^3}
\]

How many possible connected components of size \( r \) are in a graph of \( m \) nodes and maximum degree \( d \)?
Lemma 5. There are no more than $md^{2r}$ possible connected components of size $r$ in a graph of $m$ vertices and maximum degree $d$.

Proof. A connected component of size $r$ has a spanning tree of $r - 1$ edges.

We can choose a “root” for the tree in $m$ ways.

A tree can be defined by an Euler tour that starts and ends at the root and traverses each edge twice.

At each node the tour can continue in up to $d$ ways. Thus, for a given root there are no more than $d^{2r}$ different Euler tours. □
Thus, the probability that at the end of the first phase there is a connected component of size $r = \Omega(\log m)$ is bounded by

$$md^{2r}((d + 1)2^{-k/2})^{r/d^3} = o(1)$$

for $d = k2^{\alpha k}$, $\alpha > 0$ sufficiently small. $\square$
Each deferred variable appears in only one component. A component of size $O(\log m)$ has only $O(\log m)$ variables. Thus, we can enumerate (try) all possibilities in time polynomial in $m$.

**Theorem 3.** Given a CNF formula of $m$ clauses, each clause has $k = O(1)$ literals, each variable appears in up to $2^{\alpha k}$ clauses. For a sufficiently small $\alpha > 0$ there is an algorithm that finds a satisfying assignment to the formula in time polynomial in $m$. 
Online Paging (Caching)

Assume that a processor has a fast memory device - cache that can store $k$ items (pages), and a slower, much larger, secondary memory.

If a processor requests an item that is in the cache (a hit) it has no cost.

If the processor requests an item that is not in the cache (a miss) it costs one unit of time to fetch it to the cache from the secondary memory.

When an item is fetched to the cache, another item is evicted.
Possible eviction policies?

- Least Recently Used (LRU)
- First-in First-out (FIFO).
- Least Frequently Used (LFU).

What is the optimal eviction policy?
Definition 1. An on-line algorithm receives a sequence of requests, each request must be satisfied before the next request is known to the algorithm.

Theorem 1. The worst-case complexity of any on-line algorithm on an arbitrary sequence of $n$ requests is $\Omega(n)$. 
Definition 2. An off-line algorithm satisfies a sequence of requests after reading the whole sequence.

Let $f_A(\bar{R}_n)$ be the cost of on-line algorithm $A$ on a sequence of $n$ requests $\bar{R}_n$.

Let $f_O(\bar{R}_n)$ be the cost of the best off-line algorithm on the sequence $\bar{R}_n$.

Definition 3. The competitive ratio of algorithm $A$ is $C_A$, if for any $n \geq N_0$, and for any sequence $\bar{R}_n$,

$$f_A(\bar{R}_n) \leq C_A f_O(\bar{R}_n) + B,$$

where $B$ is a constant independent of $n$. 
Theorem 2. For any deterministic $k$-paging online algorithm $A$, 
\[ C_A \geq k. \]

Proof.

There are a total of $k + 1$ items. Assume that the first $k$ are in $A$’s cache at time 0.

During a run of algorithm $A$ the adversary always requests the one item that is not in the cache. Thus, $A$ misses on each request.

Since $A$ is deterministic the above defines a sequence of requests. We now show that an off-line algorithm can process the requests with no more than one miss per $k$ requests.
The off-line algorithm starts with the same set of items in its cache at time 0.

Partition the sequence of requests into rounds. A round is the maximal sequence of requests to up to $k$ distinct items (each item may be requested several times).

Let $r$ be the first item requested in the second round.

The off-line algorithm evicts $r$ when satisfying the first request.

The off-line algorithm had one miss in the first round.

At the end of the first round the off-line and on-line algorithms have the same items in their caches.

We can repeat the above argument for the next round. □
Randomized Online Algorithms

Let \( E[f_A(\vec{R}_n)] \) be the expected cost of a randomized on-line algorithm \( A \) on a sequence of \( n \) requests \( \vec{R}_n \).

We assume an oblivious adversary who knows \( A \) but does not know the random choices made by \( A \) while processing the request sequence.

Let \( f_O(\vec{R}_n) \) be the cost of the best off-line algorithm on the sequence \( \vec{R}_n \).

**Definition 4.** The competitive ratio of algorithm \( A \) against the oblivious adversary is \( C_A \), if for any \( n \geq N_0 \), and for any sequence \( \vec{R}_n \),

\[
E[f_A(\vec{R}_n)] \leq C_A f_O(\vec{R}_n) + B,
\]

where \( B \) is a constant independent of \( n \).
A Randomized Algorithm

Each cache location has a one bit marker.

The Marker Algorithm:

1. Set all markers to 0.

2. Repeat

   (a) If requested item is in cache set its marker to 1, else
      i. If all markers are 1, set all markers to 0.
      ii. evict a random element among the elements with marker 0, fetch the requested item and set the marker of that location to 1.
Theorem 3. The marker algorithm is $2H_k = 2 \sum_{j=1}^{k} \frac{1}{j}$-competitive.
Theorem 1. The marker algorithm is $2H_k = 2 \sum_{j=1}^{k} \frac{1}{j}$-competitive.

Proof. Assume that the on-line and off-line algorithms start with same set of items in their caches.

Assume that the first request is a miss (for both algorithms).

The execution of the marker algorithm partitions the sequence of requests to rounds. A round starts with setting all the markers to 0 and ends when all markers are set to 1, and the new request is a miss.

Each round starts with a miss.

In each round plus the next step there are requests to $k+1$ distinct elements.
Consider requests at given round of the marker algorithm.

An item (in or out of the cache) is **stale** if it unmarked, but was marked in the previous round.

An item is **clean** if it is not marked and not stale.

Let $\ell$ be the number of requests to clean items.

We’ll show:

1. The off-line algorithm had $\geq \ell/2$ misses (amortized).

2. The marker algorithm had no more than $\ell H_k$ misses.
Let $\ell$ be the number of requests to clean items.

Let $d_I$ be the number of elements that are in the off-line cache and not in the on-line cache at the \textbf{beginning} of the round.

Let $d_F$ be the number of elements that are in the off-line cache and not in the on-line cache at the \textbf{end} of the round.

Let $M_o$ be the number of misses of the off-line algorithm in the round.
A request to a clean item is a request to an item that is not in the cache of the on-line algorithm at the beginning of this round.

\[ M_o \geq \ell - d_I \]

At the end of the round the \( k \) items in the on-line cache are items requested in this round.

Thus, \( M_o \geq d_F \).

\[ M_o \geq \text{MAX}\{\ell - d_I, d_f\} \geq \frac{\ell - d_I + d_F}{2} \]

Summing over \( t \) rounds we get a bound of \( t\frac{\ell}{2} \pm O(k) \)
Each of the $\ell$ requests to clean items is a miss.

There are $k - \ell$ requests to stale items.

A request to a stale item is a miss if it was evicted before that request. That probability is maximized when all the requests for clean items precede that request.

The probability that the $i$-th request to a stale item is a miss is
\[
\leq 1 - \frac{k-i}{(k-i+1)}\frac{\ell}{\binom{k-i+1}{\ell}}
\]
\[
= 1 - \frac{k-i+1-\ell}{k-i+1} = \frac{\ell}{k-i+1}.
\]
The expected cost from stale requests at a round is bounded by

\[
\sum_{i=1}^{k-\ell} \frac{\ell}{k-i+1} = \ell \sum_{i=\ell+1}^{k} \frac{1}{i} \sim \ell H_k
\]

The expected cost of the marker algorithm is

\[
\ell + \ell (H_k - H_\ell) \leq \ell H_k.
\]

\( \square \)

How good is that result?

Can we prove lower bounds on randomized computation?
Consider a two player game.

Assume that the game is deterministic (no coin or dice).

Assume that the sum of the outcomes of the two players is always 0 (a zero-sum game).

A strategy for player $A$ specifies in each position (state) of the game the next move of player $A$. 


Consider an \( n \times m \) matrix \( \mathcal{M} \). The rows of the matrix correspond to the possible strategies of player \( A \), the columns corresponds to the possible strategies of player \( B \).

The \( m_{i,j} \) entry is the payoff of player \( A \) when \( A \) plays strategy \( i \) and \( B \) plays strategy \( j \).

Player \( A \) tries to maximize the payoff, \( B \) tries to minimize the payoff.
If \( A \) chooses strategy \( i \), it is guarantees a payoff of

\[
\min_j m_{i,j}.
\]

Thus, \( A \)'s optimal strategy is the \( i \) that maximizes \( \min_j m_{i,j} \).

\[
V_A = \max_i \min_j m_{i,j}
\]
is a lower bound on the payoff of \( i \) when it uses its optimal strategy.

\[
V_B = \min_j \max_i m_{i,j}
\]
is a lower bound on the payoff of \( j \) when it uses its optimal strategy.

In general

\[
V_A = \max_i \min_j m_{i,j} \leq \min_j \max_i m_{i,j} = V_B
\]

If \( V_A = V_B \) the game has a **Solution** and \( V = V_A = V_B \) is the **value** of the game.
Some Game Theory

Consider a two player game.

Assume that the game is deterministic (no coin or dice).

Assume that the sum of the outcomes of the two players is always 0 (a zero-sum game).

A strategy for player $A$ specifies in each position (state) of the game the next move of player $A$. 
Consider an $n \times m$ matrix $\mathcal{M}$. The rows of the matrix correspond to the possible strategies of player $A$, the columns corresponds to the possible strategies of player $B$.

The $m_{i,j}$ entry is the payoff of player $A$ when $A$ plays strategy $i$ and $B$ plays strategy $j$.

Player $A$ tries to maximize the payoff, $B$ tries to minimize the payoff.
If $A$ chooses strategy $i$, it is guarantees a payoff of

$$Min_j m_{i,j}.$$

Thus, $A$’s optimal strategy is the $i$ that maximizes $Min_j m_{i,j}$.

$$V_A = \text{Max}_i \, Min_j \, m_{i,j}$$

is a lower bound on the payoff of $i$ when it uses its optimal strategy.

$$V_B = \text{Min}_j \, \text{Max}_i \, m_{i,j}$$

is a upper bound on the payoff of $j$ when it uses its optimal strategy.

In general

$$V_A = \text{Max}_i \, Min_j \, m_{i,j} \leq Min_j \, \text{Max}_i \, m_{i,j} = V_B$$

If $V_A = V_B$ the game has a Solution and $V = V_A = V_B$ is the value of the game.
Lemma 1.  In any real value matrix

\[ V_A = \max_i \min_j m_{i,j} \leq \min_j \max_i m_{i,j} = V_B \]

Proof.

For an arbitrary \( i \) and any \( j \)

\[ \min_j m_{i,j} \leq m_{i,j} \]

\[ \min_j m_{i,j} \leq \max_i m_{i,j} \]

Since true for any \( i \): \[ \max_i \min_j m_{i,j} \leq \max_i m_{i,j} \]

Since true for any \( j \) and left side fixed

\[ \max_i \min_j m_{i,j} \leq \min_j \max_i m_{i,j}. \]

\( \square \)
Mixed Strategies

**Definition 1.** A **mixed strategy** is a probability distribution on the set of (pure) strategies. The player picks the strategy it plays according to that distribution.

Let $\bar{p} = (p_1, \ldots, p_n)$ be the mixed strategy for $A$, $\bar{q} = (q_1, \ldots, q_m)$ the mixed strategy for $B$.

$$E[\text{payoff}] = \bar{p}^T M \bar{q} = \sum_{i=1}^{n} \sum_{j=1}^{M} p_i m_{i,j} q_j.$$  

$$V_A = \max_{\bar{p}} \min_{\bar{q}} \bar{p}^T M \bar{q}$$

$$V_B = \min_{\bar{q}} \max_{\bar{p}} \bar{p}^T M \bar{q}$$
The Minmax Theorem:

**Theorem 1.** For any two-person zero-sum game

\[ V_A = \max_p \min_q \bar{p}^T M \bar{q} = V_B = \min_q \max_p \bar{p}^T M \bar{q}. \]

Let \( e_i \) denote a vector with 1 in the \( i \)-th place 0 elsewhere.

**Theorem 2.**

\[ V_A = \max_p \min_j \bar{p}^T M e_j = V_B = \min_q \max_i e_i^T M \bar{q}. \]
Consider a problem with a finite set of inputs $\mathcal{I}$ (of a fixed size) and a finite set of algorithms $\mathcal{A}$. For input $I \in \mathcal{I}$ and algorithm $A \in \mathcal{A}$ let $C(I, A)$ be the run-time of the algorithm.

Let $\bar{p}$ be a probability distribution over $\mathcal{I}$, $I_p$ an input chosen with that distribution.

Let $\bar{q}$ be a probability distribution over $\mathcal{A}$, $A_q$ an algorithm chosen according to this distribution.

$$
\max_p \min_A \mathbb{E}[C(I_p, A)] = \min_q \max_I \mathbb{E}[C(I, A_q)].
$$
The Yao's minimax principle:

\[ \min_A E[C(I_p, A)] \leq \max_I E[C(I, A_q)]. \]

The expected run-time of the best deterministic algorithm on some input distribution is a lower bound for the expected run-time of the best randomized algorithm on an arbitrary input.
Example: Lower bound for sorting

Assume the comparison model.

**Theorem 3.** The worst-case expected time of any randomized sorting algorithm is $\Omega(n \log n)$.

**Proof:** Consider a deterministic sorting algorithm and assume that the input is a random permutation of $n$ distinct numbers.

If the algorithm sorts all possible permutations the decision tree must have $n!$ leaves.

Since $n!/2 > 2^{\alpha n \log n}$ for some constant $\alpha > 0$, at least $1/2$ of the leaves are at distance $\Omega(n \log n)$ from the root.

Thus, the expected run-time of any deterministic algorithm on this random input is $\Omega(n \log n)$.

Applying the Yao’s principle, any randomized algorithm has expected run-time $\Omega(n \log n)$ on worst-case input.
Back to Caching

**Theorem 4.** The competitive ratio of any randomized paging algorithm for a cache of size $k$ is $\geq H_k$.

**Lemma 2.** There is a random distribution on request sequences so that any deterministic algorithm on that distribution has competitive ratio $\geq H_k$. 
**Proof.** Consider a set of $k+1$ items. Consider request sequences of length $N >> k$ generated at random:

- The first request is chosen uniformly at random from the $k+1$ items.
- Request $j$ is chosen uniformly at random from the $k$ items not requested in request $j-1$.

Partition the sequence of requests into rounds. A round is the shortest sequence that includes a request to each of the $k+1$ items.

What is the length of a round?

$kH_k$ since it is the time to pick another $k$ distinct items when choosing at random.
The off-line algorithm evicts at the end of each round the element that is requested at the end of the next round.

The off-line has one miss per round.

The probability that the on-line algorithm has a miss is each step is $\frac{1}{k}$.

Thus, the competitive ratio is $\frac{kH_k}{k} = H_k$. □