COMP 480/580 — Probabilistic Algorithms and Data Structure Sep 5, 2024

Lecture 3

Lecturer: Anshumali Shrivastava Scribe By: Bayzhan Mukatay (bm61), Jason Ludmir (jzl2), Kausar Alkaderi (ka47), Sungwon Chung (sc188)

1 PNG and Hashing

1.1 Pseudo-random Number Generator

A true random number generation function is not possible from a software standpoint, and thus a pseudo-random number generator (PNG) must be adopted. A PNG is a function of which the series of outputs seems random, but is deterministic. There is often an input seed which will determine the output sequence of the PNG function.

An example of a PNG function is the middle-square method, where an n digit number x is squared, resulting in and the middle n digits is used for the next segment of the pseudo-random output. Example from class: let $x = 1111$. Take $x^2 = 1234321$ and take the middle 4 digits. As 1234321 has an odd number of digits, add a leading 0. Thus the middle 4 digits of 01234321 are 2343 so that is our next series generated by the PNG. There are many more modern PNGs which have the same purpose of trying to generate sequences that are fit a statistical randomness. Testing the quality of a PNG is is done through statistical testing and looking at distributions of numbers that are generated. Note that no test can ever determine whether a function is truly random, as this would literally require an observation of every possible number sequence generated to an infinite degree, and for all purposes this absoluteness is not required: PNGs just have to be "good enough."

There is another "set" of PNGs which are cryptographically secure, known as cryptographically secure pseudo-random number generators (CSPRNGs). CSPRNGs attempt to make it more challenging for reverse engineering a PNG function, or predicting future outputs from the PNG. CSPRNGs generally work by implementing a source of entropy, in the form of bit streams often from hardware components. Multiple such bit streams can be combined to increase the sense of randomness and entropy. Note however this entropy still does not make the function a true RNG.

1.2 Universal Hash Function

Hashing is the process of mapping objects from a set to a (ideally) unique value. In computer science hashing is used extensively for fast queries.

Note that a realistic hash function will never be ideal, and different inputs to a hash function will at times result in the same value being returned (discussed in depth in subsequent sections).

A k-wise independent function family H is a family of functions mapping elements from a set S to a set T such that $\mathbb{P}(h(s_1) = t_1, h(s_2) = t_2, \ldots, h(s_n) = t_n) = \frac{1}{N^k}$ where k is the k-wise independence and N is the cardinality of set T, $s_i \in S, t_i \in T, h \in H$.

Intuitively, what this means that if we take a function h from our k-wise independent function family, the probability for h to map an element s_i from the domain S to a specific

t_i in the range T for k-many s_i's is $1/(N^k)$. You can think of throwing A-many labeled balls into B-many labeled bins. There is a $1/(B^A)$ chance for any single assortment of balls in bins, because you are picking one assortment out of a total of B^A total possibilities of where the balls can go.

A set of functions H mapping from S to T is said to be a K **universal hash function family** if $\mathbb{P}(h(s_1) = h(s_2) = \cdots = h(s_k)) \leq \frac{1}{N^{K-1}}$ where N is the cardinality of T and the s_i are distinct.

Consider now a 2-wise independent family of hash functions. We know from above that $\mathbb{P}(h(s_1) = t, h(s_2) = t) \leq \frac{1}{N^2}$ (by definition of 2 independent functions). From this it is easy to see that this 2-wise independent family of hash functions is a 2-universal hash family: To show this consider $\mathbb{P}(h(s_1) = h(s_2) = t)$. This probability is $\frac{1}{N^2}$, so then taking the sum of possible values t can take on (remember there are N such t) results in $\mathbb{P}(h(s_1) = h(s_2)) \leq \frac{1}{N}$. (The $\leq$ appears because the cardinality of the range is larger than the cardinality of the domain for all good hash functions). This satisfies the definition of 2-universal hash family.

Consider the function $h_{(a,b)}(x) = ((ax + b) \text{mod} P) \text{mod} N$ where $P > N$.

Then the hash family $H = \{h_{(a,b)} \mid a, b \in \{0, 1, \ldots, P - 1\}\}$

Note that an a, b is generated by the software for each hash function, but this does not change the property of $\mathbb{P}(h(s_1) = h(s_2) \leq \frac{1}{N}$ because of the **principle of deferred decision**. This principle states that the time in which a probabilistic experiment occurs does not affect the nature of the probabilistic experiment itself.

For example: Suppose that after one hour you can either watch a movie or do laundry. You decide that you will flip a coin so that if the result is heads you will wash dishes, and if the result is tails you will watch a movie, you can either...

1. Perform the coin flip now and save and use the result later
2. Perform the coin flip later and use that result

In either case, the randomness of the coin being either heads or tails is equal, and the timing of when the coin flip took place does not change the "experiment."

Note: the expected time for a hash function is 1 with an ideal hash function. However, there are collisions in real world hash functions, so the expected value of a search can not be computed until we find a way to resolve collisions. (See next section.)

2 Collision Resolution

It is possible that two different objects have the same hash value, which is known as **hash collision**. There are several different techniques to address this which we collectively refer to as collision resolution techniques. We discuss how these techniques are used when implementing hash tables.

2.1 Separate Chaining

Separate chaining is a collision resolution technique where objects that have the same hash value are stored in a linked list at that specific position in the hash table. When we search for one of those objects, it first finds the index that the hash value points to and then iterates through the linked list to find the element being searched for (usually by comparing keys).

The worst-case time complexity for searching within the hash table would become $O(N)$ where N is the number of elements we store in our hash table. The reasoning for this is that all the elements we hash might result in the same hash value, meaning that all the elements would be stored in one linked list in one of the positions in the hash table.

2.2 Expected Search Time in a Hash Table using Chaining

To understand the importance of efficient search times in hash tables, consider real-world applications that require rapid data retrieval. In large-scale web services like search engines, where millions of queries are processed every second, hash tables are employed to offer constant-time performance for lookups, insertions, and deletions.

We can analyze this problem using indicator random variables. A random variable is a variable whose value is an outcome of a random experiment (see section 3.1). An indicator random variable is a special type of random variable where the value is either 0 or 1. The probability that it is 0 is $1 - p$ and the probability that it is 1 is p.

Consider a specific chain in our hash table. We'll denote the index of this chain by $h(q)$, where $h(q)$ represents the hash value that maps to this particular chain. Let's denote a hash table with N slots, and we wish to enter m items. We can define an indicator random variable x_i which represents the event that one object i will go into the chain at index $h(q)$. The outcome will be 1 with $P = \frac{1}{N}$ and 0 otherwise.

From here it follows using the definition of the expected value (section 3.3) of an indicator random variable and linearity of expectations (section 3.4): $E[\text{Length of chain at h(q)}] = E[x_1 + x_2 + ... + x_m] = \sum_{i=1}^{m} x_i < \frac{m}{N}$

Bringing it together, we have an O(1) time complexity for running the hash function itself. Since we are looking for the expected length of the chain assuming it exists in the table, we know that there must be at least one element in the linked list already so we only need to consider $m - 1$ elements. This means that the $E[\text{Length of chain at h(q)}] = E[x_1 + x_2 + ... + x_m - 1] = \sum_{i=1}^{m-1} x_i < \frac{m-1}{N}$. Therefore we get to the following analysis: $E[\text{Searching Time}] < 1 + \frac{m-1}{N}$. [2].

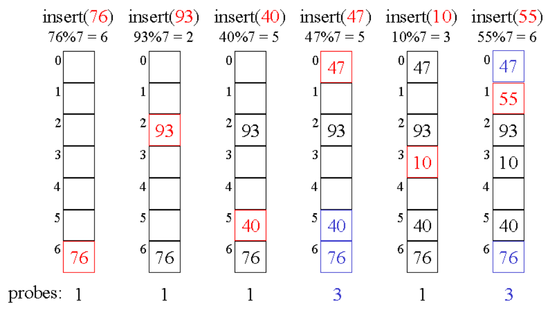
2.3 Issues with Chaining

The key issue with separate chaining is that since we are appending to linked lists as we insert elements to the hash array, the locations in memory of the various elements are not necessarily contiguous. This fragmentation leads to several performance drawbacks. CPUs rely on cache efficiency, but when data is scattered across memory, it's less likely to be found in the cache, resulting in more cache misses. Moreover, since each node in the linked list requires additional space for storing the pointer to the next node, there is some amount of memory overhead. The poor locality of reference is another issue. When traversing the linked list during a search, the CPU must repeatedly jump to different memory locations, which is slower than accessing contiguous memory in something like an array.

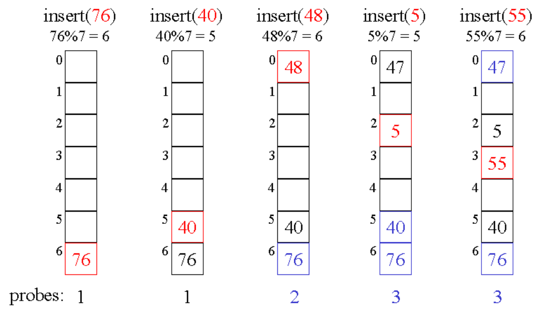
The frequent allocations for new nodes can also put stress on the memory allocation system, potentially leading to fragmentation of the heap itself. All these factors contribute to slower overall performance, particularly for large hash tables or in systems where memory access is a bottleneck. This is why alternatives like linear probing or quadratic probing might be preferred. These methods keep all elements in a contiguous array, improving cache efficiency and reducing memory fragmentation.

2.4 Linear Probing

A collision handling technique that avoids many of these pitfalls is **Linear Probing**. Rather than append to a linked list for each insertion as separate chaining does, linear probing attempts to insert the desired element into the hashed location. If that location is already occupied, then it begins probing sequentially for the next free slot in the hash array until it finds an unoccupied location. The probe uses a modulo function to ensure that when it reaches the end of the array



(a) Linear probing with hash array of size 7. Note how this leads to potential primary clustering as seen with the insertion of 47.



(b) The quadratic probing function $h(k, i) = (h(k) + i^2) \bmod 7$. 48 probes to slot 0 $(6 + 1^2)$, 5 to slot 2 $(5 + 2^2)$, and 55 to slot 3 $(6 + 2^2)$.

Figure 1: Examples of linear vs. quadratic probing [1].

it begins again at the start. Precisely, if the query is q and the length of the hash array is m, the value is hashed to location $q\ mod(m)$ in the hash array. See figure 1a for an example.

To perform a search operation for a particular query, the probe hashes to the initial hashed location. If the data element stored at that location is not the queried item, then the probe proceeds in a similar way as the insert, iterating sequentially over memory and repeatedly checking each element until it arrives at the queried item.

The benefit of using a linear probe is that there is no overhead for memory storage, and no need to perform jumps across potentially fragmented pieces of memory. Moreover, it makes efficient use of space in the hash array. However, linear probing is not without its drawbacks. The most significant issue is the tendency for elements to cluster together, something known as **primary clustering**. When multiple elements hash to nearby locations, they can create long chains of occupied slots. This clustering can lead to increased search times, as the probe may need to traverse a long sequence of occupied slots before finding the desired element or an empty slot. As the hash table becomes more full, the likelihood of collisions and the length of clusters increase. Deletion in linear probing hash tables is also more complex than in separate chaining. Simply marking a slot as empty can break the search chain for elements that were placed further along due to collisions. To address this, a special “tombstone” marker is often used, which complicates implementation.

2.5 Quadratic Probing

Quadratic probing is an alternative technique that aims to mitigate the primary clustering issue of linear probing. Instead of probing linearly (sequentially) through the hash array, quadratic probing uses a quadratic function to determine the next free location in the array. For example, if the initial hashed position of element k is $h(k)$ but the location is filled, subsequent probes are made to positions $(h(k) + i^2)\ mod(m)$, where m is the hash array size and i is the probe number. The idea here is that the distance of each probe increases quadratically from the previous one to reduce the risk of creating long chains of occupied slots, which thus alleviates the risk of primary clustering. See figure 1b for an example.

3 Probability Basics

3.1 Random Variable

A **random variable** X is a function that assigns a numerical value to each outcome in a sample space Ω. Formally, if Ω is the sample space of a probabilistic experiment, then a random variable $X : \Omega \to \mathbb{R}$ is a measurable function from Ω to the real numbers.

A random variable X can be either:

- **Discrete**, taking a countable number of distinct values.
- **Continuous**, taking an uncountable number of values, often represented as intervals on the real line.

Property: For a discrete random variable X with possible values $x_1, x_2, \ldots$, the sum of all probabilities must equal 1:

$$\sum_i \Pr(X = x_i) = 1.$$

3.2 Probability Density Function (PDF)

The **probability density function (PDF)** of a continuous random variable X is a function $f_X(x)$ that describes the likelihood of X taking a particular value. The PDF has the following properties:

- $f_X(x) \geq 0$ for all $x \in \mathbb{R}$.
- The total area under the curve of $f_X(x)$ is 1:

$$\int_{-\infty}^{\infty} f_X(x)\, dx = 1.$$

- The probability that X falls within a particular interval $[a, b]$ is given by:

$$\Pr(a \leq X \leq b) = \int_a^b f_X(x)\, dx.$$

3.3 Expectation (Expected Value)

The **expectation** or **expected value** of a random variable X, denoted by $\mathbb{E}[X]$, is a measure of the center or average of the distribution of X.

For a discrete random variable, it is defined as:

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

where the sum is over all possible values x that X can take.

For a continuous random variable, the expectation is defined as:

$$\mathbb{E}[X] = \int_{-\infty}^{\infty} x\, f_X(x)\, dx,$$

where $f_X(x)$ is the probability density function of X.

3.4 Linearity of Expectation

The **linearity of expectation** states that for any random variables $X_1, X_2, \dots, X_n$ and any real numbers $a_1, a_2, \dots, a_n$, the expectation of a linear combination is the linear combination of expectations:

$$\mathbb{E}\left[\sum_{i=1}^{n} a_i X_i\right] = \sum_{i=1}^{n} a_i \mathbb{E}[X_i].$$

This property holds regardless of whether the random variables X_i are independent.

3.4.1 Proof for Discrete Random Variables

Let $X_1, X_2, \dots, X_n$ be discrete random variables, and let $a_1, a_2, \dots, a_n$ be real constants. Define Y as a linear combination of these random variables:

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

We want to show that:

$$\mathbb{E}[Y] = \sum_{i=1}^{n} a_i \mathbb{E}[X_i].$$

Using the definition of expectation for a discrete random variable:

$$\mathbb{E}[Y] = \mathbb{E}\left[\sum_{i=1}^{n} a_i X_i\right] = \sum_{\omega \in \Omega} \left(\sum_{i=1}^{n} a_i X_i(\omega)\right) \Pr(\omega),$$

where ω represents an outcome in the sample space Ω.

Since summation is a linear operation, we can interchange the sums:

$$\mathbb{E}[Y] = \sum_{i=1}^{n} a_i \sum_{\omega \in \Omega} X_i(\omega) \Pr(\omega).$$

Recognizing the inner sum as the expectation of X_i:

$$\mathbb{E}[Y] = \sum_{i=1}^{n} a_i \mathbb{E}[X_i].$$

Note that X_i's don't have to be pairwise independent for the proof to hold.

3.4.2 Proof for Continuous Random Variables

Let $X_1, X_2, \dots, X_n$ be continuous random variables with a joint probability density function $f(x_1, x_2, \dots, x_n)$. The expectation of a linear combination $Y = \sum_{i=1}^{n} a_i X_i$ is:

$$\mathbb{E}[Y] = \mathbb{E}\left[\sum_{i=1}^{n} a_i X_i\right] = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \left(\sum_{i=1}^{n} a_i x_i\right) f(x_1, x_2, \dots, x_n)\, dx_1\, dx_2 \cdots dx_n.$$

Using the linearity of integration, we can interchange the sum and the integrals:

$$\mathbb{E}[Y] = \sum_{i=1}^{n} a_i \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} x_i f(x_1, x_2, \ldots, x_n)\, dx_1\, dx_2 \cdots dx_n.$$

Recognizing each integral as the expectation of X_i:

$$\mathbb{E}[Y] = \sum_{i=1}^{n} a_i \mathbb{E}[X_i].$$

Again, note that Xi's don't have to be pairwise independent for the proof to hold.

3.5 Independence Property

Let X_i and X_j be two independent random variables. We want to show that:

$$\mathbb{E}[X_i X_j] = \mathbb{E}[X_i]\mathbb{E}[X_j].$$

3.5.1 Proof for Discrete Random Variables

If X_i and X_j are discrete random variables, their independence implies that for any values x_i and x_j, the joint probability can be expressed as the product of their marginal probabilities:

$$\Pr(X_i = x_i, X_j = x_j) = \Pr(X_i = x_i) \Pr(X_j = x_j).$$

The expectation of the product $X_i X_j$ is then given by:

$$\mathbb{E}[X_i X_j] = \sum_{x_i} \sum_{x_j} x_i x_j \Pr(X_i = x_i, X_j = x_j).$$

Substitute the expression for the joint probability:

$$\mathbb{E}[X_i X_j] = \sum_{x_i} \sum_{x_j} x_i x_j \Pr(X_i = x_i) \Pr(X_j = x_j).$$

Rearrange the sums by grouping terms involving X_i and X_j separately:

$$\mathbb{E}[X_i X_j] = \left(\sum_{x_i} x_i \Pr(X_i = x_i) \right) \left(\sum_{x_j} x_j \Pr(X_j = x_j) \right).$$

Recognize that each term in parentheses is the expected value of the corresponding random variable:

$$\mathbb{E}[X_i X_j] = \mathbb{E}[X_i]\mathbb{E}[X_j].$$

Thus, for discrete random variables, the property holds.

3.5.2 Proof for Continuous Random Variables

If X_i and X_j are continuous random variables with joint probability density function $f_{X_i,X_j}(x_i, x_j)$ and marginal densities $f_{X_i}(x_i)$ and $f_{X_j}(x_j)$, their independence implies:

$$f_{X_i,X_j}(x_i, x_j) = f_{X_i}(x_i) f_{X_j}(x_j).$$

The expectation of the product $X_i X_j$ is:

$$\mathbb{E}[X_i X_j] = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x_i x_j f_{X_i,X_j}(x_i, x_j)\, dx_i\, dx_j.$$

Substitute the expression for the joint density:

$$\mathbb{E}[X_i X_j] = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x_i x_j f_{X_i}(x_i) f_{X_j}(x_j)\, dx_i\, dx_j.$$

Since the product of integrals is the integral of products when the variables are independent, we can separate the integrals:

$$\mathbb{E}[X_i X_j] = \left(\int_{-\infty}^{\infty} x_i f_{X_i}(x_i)\, dx_i \right) \left(\int_{-\infty}^{\infty} x_j f_{X_j}(x_j)\, dx_j \right).$$

Recognize that each integral is the expected value of the corresponding random variable:

$$\mathbb{E}[X_i X_j] = \mathbb{E}[X_i]\mathbb{E}[X_j].$$

Thus, the property holds for continuous random variables as well.

3.6 Variance

The **variance** of a random variable X, denoted by $\mathrm{Var}(X)$, measures the spread of its values around the mean. It is defined as:

$$\mathrm{Var}(X) = \mathbb{E}[(X - \mathbb{E}[X])^2].$$

Variance can also be computed using the formula:

$$\mathrm{Var}(X) = \mathbb{E}[X^2] - (\mathbb{E}[X])^2.$$

Property: Variance is always non-negative:

$$\mathrm{Var}(X) \geq 0.$$

3.7 Bienaymé's Formula

Bienaymé's Formula provides a relation for the variance of the sum of pairwise independent random variables. If X_i, $i = 1, 2, \ldots, n$, are pairwise independent random variables, then:

$$\mathrm{Var}\left(\sum_{i=1}^{n} X_i \right) = \sum_{i=1}^{n} \mathrm{Var}(X_i).$$

3.8 Proof of Bienaymé's Formula

Let $S = \sum_{i=1}^{n} X_i$ where X_i, $i = 1, 2, \ldots, n$, are pairwise independent random variables. We want to show that:

$$\mathrm{Var}(S) = \sum_{i=1}^{n} \mathrm{Var}(X_i).$$

By the definition of variance, we have:

$$\mathrm{Var}(S) = \mathbb{E}[S^2] - (\mathbb{E}[S])^2.$$

Substitute $S = \sum_{i=1}^{n} X_i$:

$$\mathrm{Var}(S) = \mathbb{E}\left[\left(\sum_{i=1}^{n} X_i\right)^2\right] - \left(\mathbb{E}\left[\sum_{i=1}^{n} X_i\right]\right)^2.$$

Expand the square:

$$\mathrm{Var}(S) = \mathbb{E}\left[\sum_{i=1}^{n} X_i^2 + 2\sum_{i<j} X_i X_j\right] - \left(\sum_{i=1}^{n} \mathbb{E}[X_i]\right)^2.$$

Using the linearity of expectation:

$$\mathrm{Var}(S) = \sum_{i=1}^{n} \mathbb{E}[X_i^2] + 2\sum_{i<j} \mathbb{E}[X_i X_j] - \sum_{i=1}^{n} \mathbb{E}[X_i]^2 - 2\sum_{i<j} \mathbb{E}[X_i]\mathbb{E}[X_j].$$

Pairwise independence implies $\mathbb{E}[X_i X_j] = \mathbb{E}[X_i]\mathbb{E}[X_j]$ for $i \neq j$, so the terms $2\sum_{i<j} \mathbb{E}[X_i X_j]$ and $2\sum_{i<j} \mathbb{E}[X_i]\mathbb{E}[X_j]$ cancel each other:

$$\mathrm{Var}(S) = \sum_{i=1}^{n} \mathbb{E}[X_i^2] - \sum_{i=1}^{n} \mathbb{E}[X_i]^2.$$

Recognizing that $\mathrm{Var}(X_i) = \mathbb{E}[X_i^2] - (\mathbb{E}[X_i])^2$, we have:

$$\mathrm{Var}(S) = \sum_{i=1}^{n} \mathrm{Var}(X_i).$$

Thus, Bienaymé's Formula is proved.

4 Tail Bounds: Markov, Chebyshev, and Chernoff Inequalities

Consider a game where you can win or lose 1 million dollars, each with a probability of 0.5. While the expected outcome $\mathbb{E}[X] = 0$ suggests a break-even scenario, this single value fails to capture the significant risk involved.

This example illustrates a limitation of using only expectations to characterize random variables. We need to also examine the probabilities of deviations from the mean. Tail bounds, such as Markov's, Chebyshev's, and Chernoff's inquealities, provide upper limits on probabilities of random variables taking values far outside their expected values.

4.1 Markov's Inequality

Markov's Inequality provides a bound for non-negative random variables. For any non-negative random variable X and any $a > 0$:

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

4.1.1 Proof of Markov's Inequality

1. Decompose $\mathbb{E}[X]$ using the law of total expectation:

$$\mathbb{E}[X] = \Pr(X < a) \cdot \mathbb{E}[X \mid X < a] + \Pr(X \geq a) \cdot \mathbb{E}[X \mid X \geq a].$$

2. Since X is non-negative, $\mathbb{E}[X \mid X < a] \geq 0$. Thus:

$$\mathbb{E}[X] \geq \Pr(X \geq a) \cdot \mathbb{E}[X \mid X \geq a].$$

3. For $X \geq a$, we have $\mathbb{E}[X \mid X \geq a] \geq a$, so:

$$\mathbb{E}[X] \geq \Pr(X \geq a) \cdot a.$$

4. Dividing both sides by a gives:

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

4.2 Chebyshev's Inequality

Chebyshev's Inequality provides a bound on the probability that a random variable deviates from its mean by a certain amount. For any random variable X with finite mean $\mu = \mathbb{E}[X]$ and finite variance $\sigma^2 = \mathrm{Var}(X)$, Chebyshev's Inequality states that for any $k > 0$:

$$\Pr(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2}.$$

This inequality gives an upper bound on the probability that the value of X is more than k standard deviations away from its mean.

4.2.1 Proof of Chebyshev's Inequality Using Markov's Inequality

We can derive Chebyshev's Inequality using Markov's Inequality. Recall Markov's Inequality: for a non-negative random variable Y and any $a > 0$:

$$\Pr(Y \geq a) \leq \frac{\mathbb{E}[Y]}{a}.$$

To apply Markov's Inequality to prove Chebyshev's Inequality, consider the random variable $Y = (X - \mu)^2$, which represents the squared deviation of X from its mean. Notice that $Y \geq 0$ is always non-negative.

Now, for any $k > 0$, we have:

$$\Pr(|X - \mu| \geq k\sigma) = \Pr((X - \mu)^2 \geq k^2\sigma^2).$$

Applying Markov's Inequality to the non-negative random variable $Y = (X - \mu)^2$ with $a = k^2\sigma^2$:

$$\Pr((X-\mu)^2 \geq k^2\sigma^2) \leq \frac{\mathbb{E}[(X-\mu)^2]}{k^2\sigma^2}.$$

Since $\mathbb{E}[(X-\mu)^2] = \sigma^2$ (by the definition of variance), we can substitute this into the inequality:

$$\Pr((X-\mu)^2 \geq k^2\sigma^2) \leq \frac{\sigma^2}{k^2\sigma^2} = \frac{1}{k^2}.$$

Thus:

$$\Pr(|X-\mu| \geq k\sigma) \leq \frac{1}{k^2}.$$

This completes the proof of Chebyshev's Inequality.

4.3 Chernoff Bounds

Chernoff Bounds provide exponential bounds on the tail probabilities for the sum of independent random variables. These bounds are particularly effective for establishing the likelihood of large deviations from the mean.

Consider independent random variables $X_1, X_2, \ldots, X_n$ where each $X_i \sim \text{Bernoulli}(p_i)$. Let:

$$X = X_1 + X_2 + \ldots + X_n, \quad \text{and} \quad \mu = \mathbb{E}[X] = \sum_{i=1}^{n} p_i.$$

Then, for any $\delta > 0$, the Chernoff bound is:

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

4.3.1 Alternative Forms of the Chernoff Bound

The bound can also be written in the following forms:

1. For $\delta > 0$:

$$\Pr(X \geq (1+\delta)\mu) \leq e^{-\frac{\delta^2\mu}{2+\delta}}.$$

2. For $1 > \delta > 0$:

$$\Pr(X \leq (1-\delta)\mu) \leq e^{-\frac{\delta^2\mu}{2}}.$$

4.3.2 Proof of the Chernoff Bound

To prove the Chernoff bound, we use a series of lemmas:

Lemma 1: Given a random variable $Y \sim \text{Bernoulli}(p)$, for all $s \in \mathbb{R}$:

$$\mathbb{E}(e^{sY}) \leq e^{p(e^s-1)}.$$

Lemma 2: For independent random variables $X_1, X_2, \ldots, X_n$, and $X = \sum_{i=1}^{n} X_i$, for $s \in \mathbb{R}$:

$$\mathbb{E}(e^{sX}) = \prod_{i=1}^{n} \mathbb{E}(e^{sX_i}).$$

Lemma 3: Let $X_1, X_2, \ldots, X_n$ be independent random variables (Bernoulli distributed), and $X = \sum_{i=1}^{n} X_i$, with $\mathbb{E}(X) = \sum_{i=1}^{n} p_i = \mu$. Then, for $s \in \mathbb{R}$:

$$\mathbb{E}(e^{sX}) \leq e^{(e^s-1)\mu}.$$

Proof Strategy:
To establish the Chernoff bound, use Markov's inequality:

$$\Pr(X \geq a) = \Pr(e^{sX} \geq e^{sa}) \leq \frac{\mathbb{E}[e^{sX}]}{e^{sa}}.$$

Apply Lemma 3 and the inequality $\ln(1+x) \leq x - \frac{x^2}{2+x}$ for $x > 0$. Set $a = (1+\delta)\mu$ and $s = \ln(1+\delta)$ to obtain:

$$\Pr(X \geq (1+\delta)\mu) \leq e^{-\frac{\delta^2\mu}{2+\delta}}.$$

This completes the proof of the Chernoff bound.

4.4 Comparison: Markov, Chebyshev, and Chernoff Bounds

These three inequalities provide different bounds depending on the assumptions about the random variable:

- **Markov's Inequality:** Applies to any non-negative random variable. It is the most general bound but often the least tight because it only uses the mean and makes no other assumptions about the distribution of the variable.
- **Chebyshev's Inequality:** Assumes the random variable has a finite mean and variance. It provides a tighter bound than Markov's by incorporating information about the spread (variance) of the random variable, thus giving more insight into the likelihood of deviations from the mean.
- **Chernoff Bound:** Assumes that the random variable is a sum of independent random variables (or similar conditions). This bound is the tightest among the three, especially for large deviations, because it leverages the independence and exponential concentration properties of the sum, resulting in exponentially small probabilities for large deviations.

4.5 Example: Comparing Markov, Chebyshev, and Chernoff Bounds

Suppose we toss a fair coin 200 times. Let X be the random variable representing the number of heads obtained. Since each toss is independent and has a probability $p = 0.5$ of showing heads, X follows a binomial distribution:

$$X \sim \text{Binomial}(n = 200, p = 0.5).$$

The expected value and variance of X are:

$$\mathbb{E}[X] = np = 200 \cdot 0.5 = 100, \quad \text{Var}(X) = np(1-p) = 200 \cdot 0.5 \cdot 0.5 = 50.$$

We want to find an upper bound for the probability of obtaining at least 150 heads, i.e., $\Pr(X \geq 150)$, using Markov's, Chebyshev's, and Chernoff bounds.

4.5.1 Markov's Inequality

Markov's inequality states that for any non-negative random variable X and $a > 0$:

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

For $a = 150$:

$$\Pr(X \geq 150) \leq \frac{\mathbb{E}[X]}{150} = \frac{100}{150} = \frac{2}{3} \approx 0.6667.$$

This bound is quite loose, as Markov's inequality only uses the expectation of X.

4.5.2 Chebyshev's Inequality

Chebyshev's inequality states that for any random variable X with finite mean μ and variance σ^2, and for any $k > 0$:

$$\Pr(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2}.$$

Here, $\mu = 100$, $\sigma = \sqrt{\mathrm{Var}(X)} = \sqrt{50} \approx 7.071$. We want:

$$\Pr(X \geq 150) = \Pr(X - 100 \geq 50).$$

Define $k = \frac{50}{\sigma} = \frac{50}{7.071} \approx 7.07$. Then:

$$\Pr(X \geq 150) \leq \Pr(|X - 100| \geq 50) \leq \frac{1}{k^2} = \frac{1}{7.07^2} \approx \frac{1}{50} = 0.02.$$

Chebyshev's inequality provides a tighter bound than Markov's by incorporating variance.

4.5.3 Chernoff Bound

The Chernoff bound provides a much tighter bound for the probability of deviations from the mean for sums of independent random variables. For a binomial random variable X, the upper tail bound is:

$$\Pr(X \geq (1 + \delta)\mu) \leq \exp\left(-\frac{\delta^2\mu}{2 + \delta}\right),$$

where $\mu = \mathbb{E}[X] = 100$ and $\delta = \frac{150-100}{100} = 0.5$.

Substitute these values into the Chernoff bound:

$$\Pr(X \geq 150) \leq \exp\left(-\frac{(0.5)^2 \cdot 100}{2 + 0.5}\right) = \exp\left(-\frac{25}{2.5}\right) = \exp(-10).$$

Calculating the exponential:

$$\exp(-10) \approx 0.000045.$$

Chernoff bound gives an exponentially small probability, which is much tighter than both Markov and Chebyshev bounds.

4.5.4 Conclusion

The upper bounds for $\Pr(X \geq 150)$ using the different inequalities are:

- **Markov's Inequality:** ≤ 0.6667
- **Chebyshev's Inequality:** ≤ 0.02
- **Chernoff Bound:** ≤ 0.000045

This comparison illustrates that Chernoff bounds are significantly tighter for large deviations, especially when the random variable is a sum of independent random variables. Chebyshev's inequality improves over Markov's by incorporating variance, but Chernoff's bound, leveraging independence, provides the sharpest estimate.

4.5.5 Actual Probability Calculation

To compute the actual probability of getting at least 150 heads in 200 fair coin tosses, we use the binomial distribution.

Given:

$$X \sim \text{Binomial}(n = 200, p = 0.5),$$

we want to find:

$$\Pr(X \geq 150).$$

The probability of obtaining exactly k heads in 200 tosses is:

$$\Pr(X = k) = \binom{200}{k}(0.5)^{200}.$$

Thus, the probability of obtaining at least 150 heads is:

$$\Pr(X \geq 150) = \sum_{k=150}^{200} \binom{200}{k}(0.5)^{200} \approx 2.753 \times 10^{-7}.$$

References

[1] University of Washington. Avl trees. `https://courses.cs.washington.edu/courses/cse326/00wi/handouts/lecture16`, 2000. Accessed: September 9, 2024.

[2] Cornell University. Cs 2112 fall 2021 object-oriented design and data structures (honors). `https://www.cs.cornell.edu/courses/cs2112/2021fa/lectures/lecture.html?id=hashtables#:~:text=If%20we%20search%20for%20an,be%20traversed%20to%20find%20it.`, 2021. Accessed: September 11, 2024.