COMP 480/580 — Probabilistic Algorithms and Data Structures August 31, 2023

Lecture 4

Lecturer: Anshumali Shrivastava *Scribe By: Stephen Xu, Zilin Xiao, Gyuheon Oh*

1 Hash Functions

Previously we've gone through the definition of a perfect hash function, that is:

$$h(O_1) \neq h(O_2) \text{ if and only if } O_1 \neq O_2$$

Recall the following construction of a two-universal hash function family:

$$h(x) = ax + b \pmod{R}$$

for some chosen values of a, b and R. One key property of hash functions in his family is that for all possible distinct x, y pairs:

$$P(h(x) = h(y)) \leq \frac{1}{R}$$

We can analogously define three-universal hash function families as

$$h(x) = ax^2 + bx + c \pmod{R}$$

Here, if we have distinct x, y and z, we can also conclude that

$$P(h(x) = h(y) = h(z)) \leq \frac{1}{R^2}$$

This is an improvement, but hash functions in this family require more cost and computing time. A conclusion we can draw is that there are no perfect hash functions, but for practical purposes, these slightly imperfect hash functions shown above can suffice. We can still roughly achieve all the desired properties of perfect hash functions through these cheap constructions.

2 Random Variables

We now shift our attention to random variables. Consider the following:

- How large can a random variable get?
- In other words, how far can a value that the random variable takes be from its mean?

Recall that the **expectation** of a random variable X with possible outcomes $x_1, x_2, \cdots, x_n$ is

$$E(X) = \sum_{i=1}^{n} x_i \cdot P(x_i)$$

Also, recall the following properties of expectation:

$$E(X_1 + X_2 + \cdots + X_n) = E(X_1) + E(X_2) + \cdots + E(X_n)$$

$$E(aX + b) = a \cdot E(X) + b$$

We desire to perform some deeper analysis on the expectation of a random variable. In particular, if the expectation of a random variable X is μ, what can we say about the likelihood of X being some value $x >> \mu$? It can be helpful to know the probability of "bad" regions where values deviate far from the mean.

For example,

- X is the number of steps an algorithm takes.
- If $\mu = E(X)$ is the expected computing time in a randomized algorithm, how likely would it be for the algorithm to take 2μ units of time? 10μ units?

In order for us to answer these questions, we need to define some new terms.

Definition 2.1 (**Variance**)**.** let X be a random variable on a sample space S. The variance of X, denoted by $V(X)$, is

$$V(X) = E[(X - E(X))^2]$$

Equivalently,

$$V(X) = E(X^2) - [E(X)]^2$$

The following relationships hold for expectation and variance. First of all, linearity of variance is also possible, but under some special conditions.

Theorem 2.1 (**Bienayme's Formula**)**.** If X_i for $i = 1, 2, \cdots, n$, are pairwise independent random variables on S, then

$$V\left(\sum_{i=1}^{n} X_i\right) = \sum_{i=1}^{n} V(X_i)$$

Note that if the X_i variables are **not** pairwise independent, we have to consider the covariances between the random variables. That is,

$$V\left(\sum_{i=1}^{n} X_i\right) = \sum_{i=1}^{n} V(X_i) + 2 \cdot \sum_{i<j} \mathrm{Cov}(X_i, X_j)$$

Next, we look into some inequalities that can help us bound the likelihood of random variables falling within a certain range.

3 Markov's Inequality

Theorem 3.1. Let X be a random variable that takes only nonnegative values. Then, for every real number $a > 0$, we have

$$P(X \geq a) \leq \frac{E(X)}{a}$$

Proof. We perform some basic manipulation of the expectation formula:

$$\begin{aligned} E(X) &= \sum_{i=0}^{\infty} P(X = i) \cdot i \\ &= \sum_{i<a} P(X = i) \cdot i + \sum_{i>a} P(X = i) \cdot i \\ &\geq 0 + a \cdot \sum_{i>a} P(X = i) \\ &= a \cdot P(X \geq a) \end{aligned}$$

Hence,

$$P(X \geq a) \leq \frac{E(X)}{a}$$

□

3.1 Remark

As simple as this inequality is, in practice, it often gives far too loose of an upper bound. If we desire a tighter upper bound, we should favor an inequality that considers the variance in X.

4 Chebyshev's Inequality

Theorem 4.1. Let X be a random variable. For every real number $r > 0$,

$$P(|X - E(X)| > a) \leq \frac{V(X)}{a^2}$$

Often times, this bound is tighter and more useful than bound from Markov's Inequality.

Corollary 4.1 (**A Corollary of Chebyshev**)**.** We can extend this inequality to a series of random variables. Given independent random variables $X_1, X_2, \cdots, X_n$, where for all $1 \leq i \leq n$:

$$E(X_i) = \mu_i$$

$$V(X_i) = \sigma_i^2$$

then for any $a > 0$,

$$P\left(\left|\sum_{i=1}^{n} X_i - \sum_{i=1}^{n} \mu_i\right| \geq a\right) \leq \frac{1}{a^2} \cdot \sum_{i=1}^{n} \sigma_i^2$$

5 Illustration: Estimating π using the Monte Carlo Method

Here is a simple algorithm for estimating π:

- Throw darts at a unit square with an inscribed circle of radius $\frac{1}{2}$.
- The probability of a randomly thrown dart landing inside the circle is $\frac{\pi}{4}$. Hence, we can create an empirical estimate of π through a series of n random darts.

  $$\hat{\pi}(n) = 4 \cdot \frac{X_1 + X_2 + \cdots + X_n}{n}$$

  where X_i is an indicator variable corresponding to the success or failure of throw i. The following holds for each X_i:

  $$E(X_i) = \frac{\pi}{4}$$
  $$V(X_i) = \frac{\pi}{4}\left(1 - \frac{\pi}{4}\right)$$

Question: How large should n be for us to get a good estimate of π?

6 Chernoff Bounds

Let $X = X_1 + X_2 + \cdots + X_n$, where all the X_i's are independent and each X_i is a Bernoulli variable with probability p. if μ is the expected value of X, then for $\delta > 0$

$$P(|X - \mu| \geq \delta\mu) \leq 2 \cdot \exp\left(-\frac{\delta^2\mu}{2+\delta}\right)$$

We can compare the three bounds we've discussed through the following scenario: a fair coin is tossed 200 times. How likely is it to observe at least 150 heads?

- Markov: ≤ 0.6666
- Chebyshev: ≤ 0.02
- Chernoff: ≤ 0.017

7 Analysis of Hashing and Chaining

Separate Chaining is a hashing strategy in hash tables where each key points to a linked list. Elements that hash to the same value are appended to the corresponding linked list in the hash table. Suppose there are m objects desired to be inserted and n total keys in the hash table. First, define the **load factor** as

$$\alpha = \frac{m}{n}$$

Let's do some quick analysis of search times in this particular hash table. At a quick glance, we can observe that

- Best Case: $O(1)$. If the desired element corresponds to a linked list of length 1, it takes constant time to compare.
- Worst Case: $O(m)$. If everything hashes to the same slot, we end up just traversing a linked list of m elements.

Furthermore, we can argue that the expected search time is

$$E(\text{time to search}) \leq 1 + \alpha$$

Beyond pure intuition, how can we mathematically arrive at this result? To do so, we try to find the expected number of elements in a particular bucket on the hash table.

Fix a particular bucket j in the hash table and let ℓ_j be the length of the linked list at bucket j. Our goal is to find $E(\ell_j)$, the expected number of elements that get randomly hashed into bucket j in the hash table. First, define a set of indicator variables $X_1, X_2, \cdots, X_m$, where X_i is defined as follows:

$$X_i = \begin{cases} 0 \text{ if element } i \text{ is not hashed to slot } j \\ 1 \text{ if element } i \text{ is hashed to slot } j \end{cases}$$

Through this definition, we see that

$$\ell_j = \sum_{i=1}^{n} X_i \implies E(\ell_j) = E\left(\sum_{i=1}^{n} X_i\right)$$

From linearity of expectation, we have

$$E(\ell_j) = \sum_{i=1}^{n} E(X_i) = E(X_1) + E(X_2) + \cdots + E(X_n)$$

Assuming independent uniform hashing, any key i is equally likely to hash to any of the m slots in the hash table, so

$$E(X_i) = 0 \cdot P(X_i = 0) + 1 \cdot P(X_i = 1) = \frac{1}{m}$$

As a result,

$$E(\ell_j) = \frac{m}{n}$$

This concludes the discussions for today's lecture.