Randomized Algorithms.
Lecture Notes Week 2.
Kasper Green Larsen.

1 Basic Probabilistic Inequalities

In this note, we introduce some of the basic probabilistic inequalities most commonly used when analysing the performance of randomized algorithms and data structures.

We motivate the inequalities by analysing the Hashing with Chaining solution to the dictionary problem. We first review the dictionary problem and the Hashing with Chaining data structure.

The Dictionary Problem. In the *dictionary problem*, we receive a set S of n keys, $x_1, \ldots, x_n$, from a universe $[U] = \{0, \ldots, U-1\}$. The goal is to store S in a data structure such that, given a query element $x \in [U]$, we can quickly determine whether $x \in S$.

Hashing with Chaining. The *Hashing with Chaining* data structure solves the dictionary problem as follows: Construct an array A with m entries, denoted $A[0], \ldots, A[m-1]$. Each array entry stores an initially empty linked list. We denote the list stored at entry i by List($A[i]$). Now pick a *random* hash function $h : [U] \to [m]$. In this note, we assume h is truly random and can be evaluated in constant time, i.e.

1. For any set of distinct elements $\{x_1, \ldots, x_k\} \subseteq [U]$ and any set of values $v_1, \ldots, v_k \in [m]$, we have $\Pr_h[h(x_1) = v_1 \wedge \cdots \wedge h(x_k) = v_k] = 1/m^k$. Equivalently, each value $h(x_i)$ is uniform random and independent of $h(x_1), \ldots, h(x_{i-1}), h(x_{i+1}), \ldots, h(x_k)$.
2. Given an element $x \in [U]$, we can compute $h(x)$ in $O(1)$ time.

After having chosen h, we process the elements $x_1, \ldots, x_n$ of S one at a time. For the element x_i, we compute $h(x_i)$ and append x_i to the list List($A[h(x_i)]$). To answer whether an element x is in S, we simply compute $h(x)$ and scan through the list List($A[h(x)]$) to see if it is there. Clearly this data structure solves the dictionary problem and uses $O(m+n)$ space. Analysing the query time and construction time is however more involved and motivates the basic probabilistic inequalities we will use repeatedly throughout the course.

Note that truly random hash functions are unrealistic in practice, but the assumption allows us to introduce the basic probabilistic inequalities in a much cleaner manner. If one was to actually implement a truly random hash function, one would need to fill a table T of size U with independent random numbers from $\{0, \ldots, m-1\}$. Then one could compute then hash value of an item $x \in [U]$ by looking up at position x in T. The issue with this, is that it requires U space, which may be huge, for instance if hashing 64-bit integers, i.e. $U = 2^{64}$.

1.1 Linearity of Expectation

In the following, we analyse the expected query time of the above Hashing with Chaining data structure. For this, let $x \in [U]$ be the query element and $S = \{x_1, \ldots, x_n\}$ the values stored in the data structure. The work spend when answering the query x is proportional to the number of elements from S also stored in List($A[h(x)]$). If we introduce an indicator random variable X_i for each x_i, taking the value 1 if $h(x_i) = h(x)$ and the value 0 otherwise, then the expected query time is $O(\mathbb{E}_h[\sum_i X_i])$. Here $\mathbb{E}_h[\cdot]$ is the *expected value* of the sum of the X_i's over the random choice of h. Recall that the expected value of a real valued discrete random variable X is

$$\mathbb{E}[X] = \sum_{x \in X} \Pr[X = x]x$$

where we sum over all possible outcomes x of the random variable X.

To bound the expectation of the sum $\sum_i X_i$, we need the first fundamental equality that we saw last time, namely *linearity of expectation*:

Lemma 1 (Linearity of Expectation)**.** *Let $X_1, \ldots, X_n$ be real valued discrete random variables and let $X = \sum_i X_i$ be the random variable giving their sum. Then $\mathbb{E}[\sum_i X_i] = \mathbb{E}[X] = \sum_i \mathbb{E}[X_i]$.*

What linearity of expectation says, is that the expectation of a sum of random variables, is equal to the sum of the expectations. We can thus use linearity of expectation to conclude that the expected length of the list $\text{List}(A[h(x)])$ is no more than $\mathbb{E}_h[\sum_i X_i] = \sum_i \mathbb{E}_h[X_i]$. Now $\mathbb{E}_h[X_i]$ is easy to bound since

$$\mathbb{E}_h[X_i] = \Pr[h(x_i) = h(x)] \cdot 1 + \Pr[h(x_i) \neq h(x)] \cdot 0 = \Pr[h(x_i) = h(x)].$$

If $x_i = x$, then $\Pr[h(x_i) = h(x)] = 1$, but only one element in S can equal x. If $x_i \neq x$ then by property 1. of h, we have $\Pr[h(x_i) = h(x)] = 1/m$. Thus the expected length of $\text{List}(A[h(x)])$ is

$$\sum_i \mathbb{E}_h[X_i] \leq 1 + \sum_{i:x_i \neq x} \mathbb{E}_h[X_i] \leq 1 + n/m.$$

If we choose $m = \Theta(n)$ then the expected query time is $O(1)$ and the space usage is $O(n)$.

Note that linearity of expectation does *not* need the variables $X_1, \ldots, X_n$ to be independent! It is thus an extremely useful inequality when analysing randomized algorithms with intricate dependencies. Also note that linearity of expectation cannot be used to conclude things such as $\mathbb{E}[X^2] = \mathbb{E}[X]^2$, which is not generally true.

1.2 Markov's Inequality

While the Hashing with Chaining data structure has expected constant query time for $m = \Theta(n)$, we might still want to say something about how often it spends significantly more time. One inequality that is useful for this is Markov's inequality. It says the following:

Inequality 1 (Markov's Inequality)**.** *Let X be a non-negative discrete random variable. Then for any $t > 0$ we have $\Pr[X > t] < \mathbb{E}[X]/t$ and $\Pr[X \geq t] \leq \mathbb{E}[X]/t$.*

If we let $X = \sum_i X_i$ denote the size of the linked list where x is hashed to (as in Section 1.1), we showed, using linearity of expectation, that $\mathbb{E}_h[X] \leq 1 + n/m$. Thus we can use Markov's inequality to conclude that if we insert n elements into a Hashing with Chaining data structure and then ask one query x, then the linked list $\text{List}(A[h(x)])$ that we have to scan through contains more than $t(1 + n/m)$ elements with probability less than $1/t$. This follows from Markov's by observing that:

$$\Pr[X > t(1 + n/m)] < \frac{\mathbb{E}[X]}{t(1 + n/m)} \leq \frac{(1 + n/m)}{t(1 + n/m)} = 1/t.$$

For $m = \Theta(n)$, this means that we spend more than $O(t)$ time with probability $1/t$.

Note that Markov's inequality requires the random variable to take non-negative values. This is indeed the case for the X_i's and thus for $X = \sum_i X_i$.

1.3 Chernoff Bound

Under our assumption of a truly random hash function h, we can sharpen the analysis given in Section 1.2 significantly. Consider the linked list at some index i, denoted $\text{List}(A[i])$. We wish to bound $|\text{List}(A[i])|$ after inserting the elements $S = \{x_1, \ldots, x_n\}$ using the truly random hash function h. For this, let Y_j denote the indicator random variable taking the value 1 if $h(x_j) = i$ and the value 0 otherwise. We have $|\text{List}(A[i])| = \sum_j Y_j$. Since h is truly random, the Y_j's are independent and $\mathbb{E}[Y_j] = 1/m$. We thus want to say something about the behaviour of $\sum_j Y_j$. Here we use the Chernoff bound (in its multiplicative form):

Inequality 2 (Chernoff Bound)**.** *Let $X_1, \ldots, X_n$ be independent* 0-1 *random variables and let $X = \sum_{i=1}^{n} X_i$. Then for any $0 < \delta < 1$ and any $\mu \leq \mathbb{E}[X]$, we have*

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

and for any $0 < \delta < 1$ and any $\mu \geq \mathbb{E}[X]$, we have

$$\Pr[X > (1+\delta)\mu] < e^{-\delta^2\mu/3}.$$

Finally, for $\delta \geq 1$ and any $\mu \geq \mathbb{E}[X]$, we have

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

Moreover, the strict inequalities $>, <$ may be replaced by $\geq, \leq$ in all of the above.

In our case, we have $\mathbb{E}[Y] = n/m$ (using linearity of expectation!) and for any $\delta \geq 1$, we get from the last inequality in the Chernoff bound that

$$\Pr[Y > (1+\delta)n/m] < \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^{n/m}.$$

For $m = n$, this implies for any $t \geq 2$ that

$$\Pr[|\operatorname{List}(A[i])| > t] = \Pr[Y > t] < \frac{e^{t-1}}{t^t} < (e/t)^t.$$

Comparing to the bound obtained using Markov's inequality, this can be used to show that we spend more than $O(t)$ time with probability decreasing as fast as t^{-t}. For Markov's inequality, the bound was $1/t$ on the probability. Thus we get an exponentially stronger bound. Note however that the Chernoff bound *requires* that the X_i's are independent $\{0,1\}$-random variables and thus the Chernoff bound is not always applicable.

1.4 Union Bound

The last inequality we will discuss is the union bound. The union bound is very simple but extremely useful. It says the following:

Inequality 3 (Union Bound)**.** *Let $E_1, \ldots, E_n$ be events (not necessarily independent). Then*

$$\Pr\left[\bigcup_i E_i\right] \leq \sum_i \Pr[E_i].$$

The union bound basically says that if we have a collection of events $E_1, \ldots, E_n$, then the probability that at least one of them happens is no more than the sum of the individual probabilities of the events. One crucial property of the union bound is that it does not need independence. When analysing algorithms and data structures, the union bound is most commonly used by defining the E_i's as some *bad events* that might happen during the execution of the algorithm. If the bad events all have small probability of happening, then the union bound can be used to show that with good probability, not even a single bad event will happen. Note that since we consider $\sum_i \Pr[E_i]$, it is important that this sum is less than 1 if you want anything useful. Thus the union bound is only used if you have a collection of rare events.

We will now use the union bound in combination with the Chernoff bound to say something highly non-trivial about the worst case query time of the Hashing with Chaining data structure.

In Section 1.3, we used the Chernoff bound to show that for any individual list $\operatorname{List}(A[i])$, we had $\Pr[|\operatorname{List}(A[i])| > t] < (e/t)^t$ when $m = n$. If we define E_i as the event that $|\operatorname{List}(A[i])| > 4\ln n/\ln\ln n$, then we get from the Chernoff bound that

$$\Pr[E_i] < ((e/4)\ln\ln n/\ln n)^{4\ln n/\ln\ln n} < (\ln\ln n/\ln n)^{4\ln n/\ln\ln n}.$$

Note that $\ln\ln n$ is undefined for $n < e$, hence we will simply assume $n \geq 3$ (it is anyways common with O-notation to state bounds only for n bigger than some constant).

The expression is a bit complicated to analyze due to both the $\ln\ln n$ and $1/\ln n$ in the base of the exponential expression. One trick to deal with this, is to upper bound $\ln\ln n$ in terms of something more closely related to $\ln n$. Here we use that $\ln\ln n \leq \sqrt{\ln n}$ for all $n \geq 3$. Intuitively, this should be true for n big enough as $\ln\ln n$ grows only logarithmically in $\ln n$ whereas $\sqrt{\ln n}$ grows polynomially. If one wants to be convinced that indeed $\ln\ln n \leq \sqrt{\ln n}$ for $n \geq 3$, we could for instance look at the derivative of $\sqrt{\ln n} - \ln\ln n$, which is (by the chain rule):

$$\frac{1}{2\sqrt{\ln n}} \cdot \frac{1}{n} - \frac{1}{\ln n} \cdot \frac{1}{n} = \frac{1}{n\sqrt{\ln n}} \left(\frac{1}{2} - \frac{1}{\sqrt{\ln n}} \right).$$

For $n \geq 3$, we always have $\frac{1}{n\sqrt{\ln n}} > 0$. Thus the sign of the expression equals that of $1/2 - 1/\sqrt{\ln n}$. This is clearly negative for $\sqrt{\ln n} < 2$, it equals 0 for $\sqrt{\ln n} = 2$ and is positive for $\sqrt{\ln n} > 2$. Hence it takes its minimum at $n = e^4$. For $n = e^4$, we have $\sqrt{\ln n} - \ln\ln n = 2 - \ln 4 > 0$. Since this n gives the minimum value for $\sqrt{\ln n} - \ln\ln n$, we have $\sqrt{\ln n} - \ln\ln n > 0$ for all $n \geq 3$. Using this, we now bound

$$\Pr[E_i] < \left(\frac{\ln\ln n}{\ln n} \right)^{4\ln n/\ln\ln n} \leq (\sqrt{\ln n}/\ln n)^{4\ln n/\ln\ln n} = (\ln n)^{-2\ln n/\ln\ln n}.$$

Using that $\ln n = e^{\ln\ln n}$, this equals:

$$(\ln n)^{-2\ln n/\ln\ln n} = \left(e^{\ln\ln n}\right)^{-2\ln n/\ln\ln n} = e^{(-2\ln n/\ln\ln n)\ln\ln n} = n^{-2}.$$

From the union bound it follows that

$$\Pr[\cup_i E_i] \leq \sum_i n^{-2} = n^{-1}.$$

This means that with probability $1 - 1/n$, there is not even a single list $\text{List}(A[i])$ with size more than $4\ln n/\ln\ln n$ (assuming $n \geq 3$). In this case, the worst case query time of the data structure is $O(\lg n/\lg\lg n)$ (no matter which linked list you scan through when answering your query, it has to be small). Thus the Hashing with Chaining data structure, with $m = n$, has worst case query time $O(\lg n/\lg\lg n)$ with probability $1 - 1/n$ when h is truly random. The construction time is $O(n)$ and the space is $O(n)$.

Always Worst Case. If one is not satisfied with only having worst case query time $O(\lg n/\lg\lg n)$ with probability $1 - 1/n$, but want this worst case query time always, then one can achieve this by changing to an expected $O(n)$ construction time as follows: After having inserted the keys $x_1, \ldots, x_n$, check the size of all the lists $\text{List}(A[0]), \ldots, \text{List}(A[n-1])$. If one of them exceeds $4\ln n/\ln\ln n$, discard the current data structure and try again with a freshly chosen h. This performs at least i iterations with probability no more than $(1/n)^{i-1}$. Thus (using linearity of expectation) the expected construction time is upper bounded by

$$\sum_{i=1}^{\infty} (1/n)^{i-1} \cdot O(n) = O(n).$$

We have thus achieved a worst case query time of $O(\lg n/\lg\lg n)$ with space $O(n)$ and expected construction time $O(n)$.