CS 224: Advanced Algorithms Spring 2017

Lecture 3 — January 31, 2017

Prof. Jelani Nelson *Scribe: Saketh Rama*

1 Overview

In the last lecture we covered Y-fast tries and Fusion Trees.

In this lecture we start our discussion of hashing. We will study load balancing, k-wise independence, and the dynamic dictionary problem solved using hashing with chaining and linear probing.

2 Load Balancing

Formally, consider jobs with IDs in the universe $[u]$, and machines labeled $1, \ldots, m$. The task of *load balancing* studies the assignment of jobs to machines so that no machine is too overloaded. For example, we could have a centralized scheduler which decides where jobs should go. However, local decisions for scheduling are preferable for complexity reasons – this motivates our study of hashing.

The idea is to have a "random" function $h : [u] \to [m]$.

2.1 Chernoff Bound

We will assume there are n jobs in the system, with $n \ll u$, and focus on the case where $m = n$. Studying this case will motivate our statement and proof of *Chernoff bounds*.

2.1.1 Application of Bound

Load balancing corresponds to a small probability $\mathbb{P}(\text{max load of any machine } > T)$. We can restate this as follows:

$$\begin{aligned}\mathbb{P}(\text{max load of any machine } > T) &= \mathbb{P}(\exists \text{ machine } i : load(i) > T) \\ &\leq \sum_{i=1}^{n} \mathbb{P}(load(i) > T) \\ &= n \cdot \mathbb{P}(load(i) > T)\end{aligned}$$

where the inequality follows from the union bound. We can now apply the Chernoff bound, which will prove later.

Lemma 1. ***Chernoff Bound (Discrete Case).*** *Let random variables $X_1, \ldots, X_R \in \{0, 1\}$ be independent, with $X = \sum_i X_i$ and $\mathbb{E}[X] = \mu$. Then for all $\delta > 0$,*

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

To apply this to hashing, define $R = n$ indicator variables

$$X_i = \begin{cases} 1 & h(i) = 1 \\ 0 & \text{o.w.} \end{cases}.$$

Then

$$\mu = \mathbb{E}[X] = \sum_i \mathbb{E}[X_i] = n \cdot \frac{1}{n} = 1.$$

We can now analyze the probability in load balancing.

$$\begin{aligned} \mathbb{P}(load(1) > T) \cdot n &< \frac{e^{T-1}T^T}{\cdot} n \\ &< \left(\frac{e}{T}\right)^T \cdot n < n \cdot e^T \cdot (1/T)^T \end{aligned}$$

By setting $T = \Theta\left(\frac{\lg n}{\lg \lg n}\right)$ such that $(1/T)^T \ll 1/n^2$, we get $n \cdot e^T \cdot (1/T)^T < 1/n^{1-o(1)}$.

In load balancing jargon, we say that if the left-hand condition is satisfied, then the max load is $O\left(\frac{\lg n}{\lg \lg n}\right)$ ***with high probability***.

2.1.2 Proof of Bound

Because X_i is Bernoulli, $\mathbb{E}[X_i] = p_i$ implies $\mu = \sum_i p_i$. We will make use of the following inequality to bound the probability using an expectation.

Lemma 2. ***Markov's Inequality.*** *Let X be a nonnegative r.v. Then for all $\lambda > 0$,*

$$\mathbb{P}(X > \lambda) < \frac{\mathbb{E}[X]}{\lambda}.$$

Because f is strictly increasing, we can say that

$$\mathbb{P}(X > (1+\delta)\mu) = \mathbb{P}(f(x) > f((1+\delta)\mu)).$$

As a somewhat magical step, choose $f(z) = e^{tz}$, such that we can guess at the form of z.

$$\begin{aligned}
\mathbb{P}(e^{tX} > e^{t(1+\delta)\mu}) &< e^{-t(1+\delta)\mu} \cdot \mathbb{E}(e^{t\sum_i X_i}) \text{ (by Markov's inequality)} \\
&= e^{-t(1+\delta)\mu} \cdot \mathbb{E}(\prod_i e^{tX_i}) = e^{-t(1+\delta)\mu} \cdot \prod_i \mathbb{E}(e^{tX_i}) \\
&= e^{-t(1+\delta)\mu} \cdot \prod_i (1 - p_i + p_i \cdot e^t) \\
&\leq e^{-t(1+\delta)\mu} \cdot \prod_i (e^{p_i(e^t-1)}) \\
&= e^{-t(1+\delta)\mu} \cdot e^{\mu(e^t-1)}
\end{aligned}$$

This establishes that $\mathbb{P}(X > (1+\delta)\mu) < \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^\mu$.

The above proof required a magical step of guessing at the function's form. We can also consider Chernoff bounds more intuitively as a moment bound for large p in the expression derived from a repeated application of Markov's inequality:

$$\mathbb{P}(|X - \mathbb{E}[X]|^p) < \frac{\mathbb{E}(|X - \mathbb{E}[X]|)^p}{\lambda^p}.$$

2.2 Alternative Analysis

We can also approach load balancing more directly.

$$\begin{aligned}
\mathbb{P}\left(\text{max load } > T\right) < n \cdot \mathbb{P}(load(1) > T) & \\
&= n \cdot \mathbb{P}(\exists T \text{ jobs mapping to machine 1}) \\
&< n \cdot \binom{n}{T} \frac{1}{n^T}
\end{aligned}$$

We can bound $\frac{1}{n^T}$ as follows. For $I = \{i_1, \ldots, i_T\}$ with $i_1 < \cdots < i_T$, let

$$X_I = \begin{cases} 1 & \text{if all } i\text{'s map to 1} \\ 0 & \text{o.w.} \end{cases}.$$

Then

$$\mathbb{P}(\exists T \text{ jobs mapping to 1}) = \mathbb{P}(\exists I : X_I = 1) \leq \sum_I \mathbb{P}(X_I = 1)$$

by the union bound.

Note that

$$\binom{n}{T} \cdot \frac{1}{n^T} = \frac{n!}{T!(n-T!)} \cdot \frac{1}{n^T} = \frac{n(n-1)\cdots(n-T+1)}{T!} \cdot \frac{1}{n^T} < \frac{1}{T!}.$$

Here, we can either use Stirling's approximation or be sloppier with the inequality $T! > (T/2)^{T/2}$. We choose the latter, and so $\frac{1}{T!} < \frac{1}{q^q}$ where $q = T/2$. This quantity is much smaller than $1/n$ for $T = q = \Theta\left(\frac{\lg n}{\lg \lg n}\right)$.

3 k-wise Independence

It turns out that the above analysis only requires k-wise independence (where $k = T$), a concept which we will now study.

Note that $\mathbb{P}(X_I = 1) = \mathbb{P}_h(\wedge_{j=1}^{T} h(i_j) = 1) = \prod_{j=1}^{T} \mathbb{P}_h(h(i_j) = 1) = (1/n)^T$, where the probability is taken over the randomness of the hash function.

Definition 3. *A family $\mathcal{H}$ of functions $h : [u] \to [m]$ is a* **k-wise independent** hash family *if for any $i_1 < \cdots < i_k \in [u]$ and $y_1, \ldots, y_k \in [m]$, we have*

$$\mathop{\mathbb{P}}_{h \in \mathcal{H}}(\wedge_{j=1}^{T} h(i_j) = y_j) = \prod_{j=1}^{k} \mathop{\mathbb{P}}_{h}(h(i_j) = y_j).$$

This condition is useful because a totally random hash function would require $u \lg m$ bits to store. With a less restrictive k-wise independence, we can get away with less storage.

Note that if $\mathcal{H}$ is the set of all functions mapping $[u]$ to $[m]$, then a random $h \in \mathcal{H}$ is what we just analyzed.

3.1 Example

Let $u = p$ where p is prime, with $p \geq 2m$. Define $\mathcal{H} = \{h : h(x) = \sum_{i=0}^{k-1} a_i x^i \bmod p\}$. Then $|\mathcal{H}| = p^k$, and so $\lg |\mathcal{H}| = k \lg p$ bits.

We will omit the analysis of this example for the purposes of this course. It can be derived using polynomial interpolation.

4 Dynamic Dictionary Problem

The dynamic dictionary problem is a data structure problem. The goal is to maintain items $S \subseteq [u]$ as keys where each $x \in S$ has an associated value in the universe $[u]$ subject to the following operations:

1. $insert(x, v)$ – associate key x with v
2. $query(x)$ – return value of key x
3. $del(x)$ – remove x from S

4.1 First Solution: Hashing with Chaining

We can define an array $A[1 \ldots m]$ whose entries are pointers to linked lists with key-value pairs as nodes. The hash function maps the universe into this array.

It turns out that if $\mathcal{H}$ is 2-wise independent with $m \geq |S|$, then $\mathbb{E}[\text{time per op}] = o(1)$. The analysis of this is available in the notes for CS 124/125 and CLRS.

4.2 Second Solution: Linear Probing

The approach we will focus on in this course is linear probing. We again have an array $A[1dotsm]$. To insert a key k, we start at $h(k)$ and move right until we find an empty slot. (For now, we will consider the simpler case which does not support deletions.)

Linear probing first appeared in an IBM 701 program by Samuel, Amdahl, and Boehme in 1954, and was subsequently analyzed by Knuth in 1963 in the case where $\mathcal{H}$ is all functions [1].

Knuth showed that if $m \geq (1+\epsilon)^n$, where $n = |S|$, then $\mathbb{E}(\text{time per op}) \leq O(1/\epsilon^2)$.

Pagh, Pagh, and Ružić showed more recently that if $m \geq c \cdot n$ (e.g., $c = 3$ works), then 5-wise independence guarantees constant expected time as well, which we will prove in the next lecture.

In the case of 4-wise independence, Pătraşcu and Thorup showed that there exist $\mathcal{H}$ which have expected runtime $\Omega(\lg n)$, which is *not* constant.

References

[1] Donald Knuth. *The Art of Computer Programming* (2nd ed.). Addison Wesley, pp. 513–558, 1998.

[2] Anna Pagh, Rasmus Pagh, and Milan Ružić. Linear probing with 5-wise independence. *SIAM Review* 53.3 (2011):547-558, 2011.

[3] Mihai Pătraşcu, and Mikkel Thorup. On the k-independence required by linear probing and minwise independence. *International Colloquium on Automata, Languages, and Programming*. Springer Berlin Heidelberg, 2010.