CS 224: Advanced Algorithms Spring 2023

Lecture 8 — February 6th, 2023

Prof. Prashant Pandey *Scribe: Taos Transue*

1 Overview

In the last lecture we discussed the game of Balls 'n' Bins to help us understand hashing, assuming that the balls land in any particular bin with uniform probability. That assumption says that we are dealing with a 'totally random' has function mapping balls to bins. In this lecture, we discuss other types of hash functions, and their use in hash tables.

2 Hash Functions

Let $U = \{0, 1, \ldots, u-1\}$ be the universe set, and $M = \{0, 1, \ldots, m-1\}$; then, $h : U \to M$ is a hash function.

2.1 Totally Random Hash Functions

Definition 1. *A hash function h is totally random if $\forall x, y \in U$ and $z, w \in M$, $\mathbb{P}[h(x) = z] = \frac{1}{m}$ and $\mathbb{P}[h(x) = z | h(y) = w] = \frac{1}{m}$.*

Complexity Considerations

- The space required to construct a totally random has function is $\Theta(u \log(m))$ bits of information.

2.2 Universal Hash Functions

Let H be a family of hash functions mapping U to M such that $\forall h \in H$, $\forall x, y \in U$ where $x \neq y$,

$$\mathbb{P}[h(x) = h(y)] = O\left(\frac{1}{m}\right) \tag{1}$$

then H is the family of universal hash functions. If $\mathbb{P}[h(x) = h(y)] \leq \frac{1}{m}$, then the hash functions of H are called strong. Unlike totally random hash functions, the events $h(x) = z, h(y) = w$ for $z, w \in M$ may not be independent.

The hash functions $h \in H$ have the form

$$h(x) = [(ax) \mod p] \mod m \tag{2}$$

where $a \in \{1, \dots, p-1\}$, p is prime, and $p \geq u$. When u, m are powers of two, h can be implemented on the computer as

$$h(x) = (ax) >> (\log(u) - \log(m)) \tag{3}$$

where $>>$ is the right bit-shift operator.

2.3 k-Wise Independent Hash Functions

Let H be a family of hash functions mapping U to M such that $\forall h \in H$, for $\{x_i\}_{i=1}^k \subset U$ distinct, and $\{t_i\}_{i=1}^k \subset M$,

$$\mathbb{P}[h(x_1) = t_1, \dots, h(x_k) = t_k] = O\left(\frac{1}{m^k}\right) \tag{4}$$

then H is the family of k-wise independent hash functions. When $k = 2$, h is called pairwise independent. Note that these hash functions are stronger than the universal hash functions.

Pairwise independent hash functions have the form

$$h(x) = [(ax + b) \mod p] \mod m \tag{5}$$

where $a \in \{1, \dots, p-1\}$, $b \in \{0, \dots, p-1\}$, p is prime, and $p \geq u$. More generally, k-wise independent hash functions have the form

$$h(x) = \left[\left(\sum_{i=0}^{k-1} a_k x^i\right) \mod p\right] \mod m \tag{6}$$

for $a_{k-1} \in \{1, \dots, p-1\}$ and $\{a_i\}_{i=0}^{k-2} \subset \{0, \dots, p-1\}$.

Complexity Considerations

- The time complexity to compute h is $O(k)$.

2.4 Simple Tabulation Hashing

Let x be the item to be hashed. We may view x as a vector of characters $x_1, \dots, x_c$. To hash x, first build a totally random hash table T_i for each x_i. Using these hash tables, the hash of x is

$$h(x) = T_1(x_1) \oplus \cdots \oplus T_c(x_c) \tag{7}$$

where $\oplus$ is the X-OR operation. Now, h is a 3-wise independent hash function.

Complexity Considerations

- The space complexity to construct the hash tables T_i is $O(c\sqrt{u})$.
- The time complexity to compute h is $O(c)$. Note this can be reduced to $O(1)$ if using the AVX-512 instruction set and x fits into a cache line ($x \leq\sim 64$ bytes).

3 Hash Tables

3.1 Chaining Hash Table

A chaining hash table handles collisions in the hash function by extending a linked-list from its buckets. The time required perform operations grows as the length of the linked-lists grow, so we would like to know how long the linked-lists may grow to be. Recall from the previous lecture that the number of balls in the fullest bucket is $O\left(\frac{\log(n)}{\log(\log(n))}\right)$ with high probability. Since a ball landing in a nonempty bucket represents a collision, the length of the longest linked-list in a chaining hash table is of the same scale.

Exploring the length of the linked-lists deeper, consider the expected length of the linked-lists. Let c_t be the length of the linked-list of bucket t in the hash table. Then,

$$\mathbb{E}[c_t] = \sum_i \mathbb{P}[h(x_i) = t] \tag{8}$$

To get an idea of the shape of the distribution of collisions, it is also helpful to look at the variance of c_t. If our hash function is strong, we expect the variance to be close to one (most buckets of the hash table only have one item).

$$V[c_t] = \mathbb{E}[c_t^2] - (\mathbb{E}[c_t])^2 \tag{9}$$

$$\mathbb{E}[c_t^2] = \frac{1}{m}\sum_s \mathbb{E}[c_s^2] = \frac{1}{m}\sum_{i,j} \mathbb{P}[h(x_i) = h(x_j)] \tag{10}$$

We can continue the last string on equalities for $\mathbb{E}[c_t^2]$ if the hash function h is universal:

$$\mathbb{E}[c_t^2] = \frac{1}{m} n^2 O\left(\frac{1}{m}\right) = O\left(\frac{n^2}{m^2}\right) = O(1) \quad \text{when } m = \Omega(n) \tag{11}$$

For the amortized performance bounds, see the lecture notes.

3.2 Perfect Hashing

This is a hash table where with each bucket is another hash table with $\Theta(c_t^2)$ buckets. Another name for this hash table is a 2nd-level hash table.

This hash table ultimately uses a bijective hash function, so there are no collisions. The top-level table has a hash function, as does each 2nd-level hash table. Collisions occur when a 2nd-level hash table has a collision. When this happens, the 2nd-level hash table is recomputed with a new hash function. The recomputation is repeated with new hash functions until there is no longer a collision. It has been proven that this collision-resolution process will terminate in finite iterations. However, this reveals the downside of this hash table: its operations are very sequential.

We can look at the expected number of collisions in a 2nd-level hash table:

$$\mathbb{E}[\text{\# of collisions in 2nd-level hash table}] = \sum_{i,j} \mathbb{P}[h(x_i) = h(x_j)] \tag{12}$$

$$h \text{ is Universal} \Rightarrow = c_t^2 O\left(\frac{1}{c_t^2}\right) = O(1) \tag{13}$$