Design and Analysis of Algorithms: Lecture 8

Ben Chaplin

Contents

1 Problem

1.1 Dictionary problem

The dictionary problem asks for a data structure with the following requirements:

- Maintain a dynamic set of items (where each item has a distinct key)
- Support:
  - INSERT(item)
  - DELETE(item)
  - SEARCH(key)

1.2 Hashing

"Hashing" the items' keys in a hash table gives $O(1)$ time per operation. We will use the following variables with regards to a hash table:

- u : number of possible keys
- n : number of items currently in the table
- m : size of the table
- $h : \{0, 1, \ldots, u-1\} \to \{0, 1, \ldots, m-1\}$ the hash function

With chaining, hashing takes $\Theta(1+\frac{n}{m})$ time. The proof of this fact assumes **simple uniform hashing**.

Definition. A function provides **simple uniform hashing** when, for two random distinct keys k_1, k_2, the probability the function outputs the same hash is $\dfrac{1}{m}$.

But what kind of hash functions guarantee this, no matter the universe of keys?

2 Universal Hashing

Definition. Let $\mathcal{H}$ be a set of hash functions. $\mathcal{H}$ is **universal** if for any two distinct keys k_1, k_2, the probability a random hash function $h \in \mathcal{H}$ outputs the same hash is at most $\frac{1}{m}$.

Note that contrary to the definition of simple uniform hashing, this definition includes a probability over *all hash functions*. Simple uniform hashing was defined by a probability over *all pairs of distinct keys*.

Theorem 1. *Let $\mathcal{H}$ be universal. For n arbitrary distinct keys and a random $h \in \mathcal{H}$, the expected number of colliding keys is at most $1 + \frac{n}{n}$.*

Proof. Take keys $k_1, \ldots, k_n$. Define an "indicator" random variable:

$$I_{i,j} = \begin{cases} 1 & \text{if } h(k_i) = h(k_j) \\ 0 & \text{else} \end{cases}$$

$$\begin{aligned}
E[\text{number of keys with the same hash as } k_i] &= E\left[\sum_{i \neq j} I_{i,j}\right] + I_{i,i} \\
&= E\left[\sum_{i \neq j} I_{i,j}\right] + 1 \\
&= \left(\sum_{i \neq j} E[I_{i,j}]\right) + 1 \\
&= \left(\sum_{i \neq j} Pr(I_{i,j} = 1)\right) + 1 \\
&\leq \left(\sum_{i \neq j} \frac{1}{m}\right) + 1 && \text{by universality} \\
&= \frac{n-1}{m} + 1
\end{aligned}$$

□

3 Dot product hash family

Definition. Assume m is prime and $u = m^r$ for some $r \in \mathbb{Z}^+$. For each key k, define a vector $\bar{k} = \langle k_0, k_1, \ldots, k_{r-1} \rangle$ to be the digits of k in base m. The **dot product hash family** is defined:

$$\mathcal{H} = \left\{ h_a(k) = (\bar{a} \cdot \bar{k}) \mod m \mid a \in \{0, \ldots, u-1\} \right\}$$

Note here that the hash functions in $\mathcal{H}$ are completely determined by the choice of a.

Theorem 2. *The dot product hash family is universal.*

Proof. Let $k \neq k'$ be keys. Then, some digit of $\bar{k}$ and $\bar{k}'$ differs, say $k_d \neq k'_d$. For a random key a:

$$
\begin{aligned}
Pr(h_a(k) = h_a(k')) &= Pr\left(\sum_{i=0}^{r-1} a_i k_i = \sum_{i=0}^{r-1} a_i k'_i \mod m\right) \\
&= Pr\left(\sum_{i=0}^{r-1} a_i (k_i - k'_i) = 0 \mod m\right) \\
&= Pr\left(a_d (k_d - k'_d) + \sum_{i=0, i\neq d}^{r-1} a_i (k_i - k'_i) = 0 \mod m\right) \\
&= Pr\left(a_d = -(k_d - k'_d)^{-1} \cdot \sum_{i=0, i\neq d}^{r-1} a_i (k_i - k'_i) \mod m\right) && m \text{ is prime}
\end{aligned}
$$

Or, in other words, $Pr(h_a(k) = h_a(k'))$ is the same as the probability that a_d is equal to some integer n mod m. Note that n does not rely on a_d at all, so the relevant probability is the same as the probability that a_d is equal to random number modulo m.

$$
\begin{aligned}
&= E_{a_i \neq d}\big[Pr_{a_d}(a_d = n \mod m)\big] \\
&= \frac{1}{m}
\end{aligned}
$$

□