COMP 251: Data Structures and Algorithms Fall 2006

Hash Functions

Lecturer/TA: Ethan Kim *October 5rd, 2006*

In this lecture, we survey some of the methods to design hash functions.

1 Hash functions: Heuristics

As discussed in last lecture, we would like to design our hash function h that approximates the simple uniform hashing(SUH), i.e., every element is equaly likely to hash into any of the m slots, independently of other element's positions in the hash table.

NOTE: The keys by which our data set are indexed may be of any data type. It is not difficult to convert this key into an integer, and in particular $Z \cup \{0\}$. Thus, we assume, for the remaining of the lecture, that the hash function takes $Z \cup \{0\}$ as input.

1.1 Division Method

$$h(k) = k \mod m$$

For this method, we should choose m with care. For example, if $k = 10052006$, and $m = 100$:

$$h(k) = 06 \quad \rightarrow \text{only 2 digits of the year}$$

For general rule, choose m to be prime, not close to a power of 2. (If m is a power of 2, we are simply taking the least significant digits of k.)

1.2 Multiplication Method

$$h(k) = \lfloor m(kA \mod 1) \rfloor$$

, where $0 < A < 1$, and $kA \mod 1$ means taking only the fractional part. Now, we have to choose both A and m.

Note that if we set m to be a power of 2 (i.e., $m = 2^p$), then the hash value $h(k)$ is the p most significant bits of $kA \mod 1$. (and this is easy to implement!) Consider the following trick: suppose k fits in a word of length w. We can choose the value for A to be $\frac{s}{2^w}$ for some integer $0 < s < 2^w$. To compute the hash value for k, we first multiply k by s. It is easy to see that the product ks now fits in $2w$ bits. (Because k fits in w bits and $s < 2^w$.) Hence, $ks = r_1 2^w + r_0$. So we have:

$$\begin{aligned}
&\lfloor m(kA \mod 1) \rfloor \\
=&\lfloor 2^p(\frac{ks}{2^w} \mod 1) \rfloor \\
=&\lfloor 2^p(\frac{r_1 2^w + r_0}{2^w} \mod 1) \rfloor \\
=&\lfloor 2^p(r_0) \rfloor
\end{aligned}$$

We now simply need to take the p most significant bits of r_0.

This method works well with any value of A, though the optimal choice depends on the data(keys) itself. An interesting (and striking) example is by using the Golden Ratio ϕ.

$$\phi = \frac{1+\sqrt{5}}{2}$$
$$\phi^{-1} = \frac{\sqrt{5}-1}{2} \approx 0.618036\ldots$$

Choose $A = \phi^{-1}$. (i.e., Set s as an integer relatively prime to 2^w such that $\frac{s}{2^w}$ is as close to as ϕ^{-1} as possible.) Then, if we consider a consecutive insertions with $k = 1, 2, \ldots$, each new point falls into the largest interval, partitioning it in the golden ratio.

2 Universal Hashing

With both methods presented above, one may come up with a sequence of keys that all get hashed into the same slot in a hash table. If this happens, we simply have one linked list with overhead of hash function: this makes it worse than the linked list implementation!

One way to resolve is to use a randomized algorithm: given a collection of hash functions, we can select the hash function at random at the beginning of execution. This causes the hashing behave differently on each execution, even with the same input.

Definition 1. *Let H be a finite collection of hash functions that map a universe of keys into $Z_m = \{0, 1, \ldots m-1\}$. We call this collection of hash functions* universal *if for each pair of keys k, l, the number of hash functions $h \in H$ such that $h(k) = h(l)$ is at most $\frac{|H|}{m}$.*

Stated differently, if we pick a hash function $h \in H$ at random, for every pair of keys k, l, the chance of a collision is no more than $\frac{1}{m}$. (Recall the definition of Simple Uniform Hashing!)

Let's assume that such collection of hash functions exist, for now. Then we can say the following:

Theorem 1. *Suppose a hash function h is chosen from H at random, and is used to hash n keys into a hash table T with m slots. If key k is not in the table, then the expected length $E[n_{h(k)}]$ of the chain that k would hash into is at most α. If k is already in the table, then the expected length of the list is at most $1 + \alpha$.*

Proof. First, define the indicator random vairable $X_{kl} = I\{h(k) = h(l)\}$. By the definition of universal hash functions, we have that $E[X_{kl}] \leq \frac{1}{m}$.

Then, define the random variable Y_k for each key k that equals the number of keys other than k that hash into the same slot as k.

$$Y_k = \sum_{l \in T, l \neq k} X_{kl}$$

We can compute the expected value for Y_k:

$$\begin{aligned} E[Y_k] &= E[\sum_{l \in T, l \neq k} X_{kl}] \\ &= \sum_{l \in T, l \neq k} E[X_{kl}] \\ &\leq \sum_{l \in T, l \neq k} \frac{1}{m}. \end{aligned}$$

Now we look at the two cases:

1. $k \notin T$:
   Then $n_{h(k)} = Y_k$, and $|\{l : l \in T \text{ and } l \neq k\}| = n$.
   So we have $E[n_{h(k)}] = E[Y_k] \leq \frac{n}{m} = \alpha$.
2. $k \in T$:
   Then $n_{h(k)} = Y_k + 1$, and $|\{l : l \in T \text{ and } l \neq k\}| = n - 1$.
   So we have $E[n_{h(k)}] = E[Y_k] + 1 \leq \frac{n-1}{m} + 1 = \alpha - \frac{1}{m} + 1 < 1 + \alpha$.

□

This asserts that using the universal hashing, any sequence of INSERT, DELETE, SEARCH operations with $O(m)$ INSERT operations would take $\theta(n)$ expected time overall.

3 Construction for Universal Hash Functions

Now that we know the universal hashing gives us good expected bound on the length of lists. But how do we design such class of functions? To do this, we need a bit of number theory(not too much!.)

First, pick a large prime p such that all possible keys belong to the set $\{0, 1, \ldots, p-1\}$. Let Z_p denote the set $\{0, 1, \ldots, p-1\}$, and let Z_p^* denote the set $\{1, 2, \ldots, p-1\}$. Now, let's assume $p > m$, since otherwise it's not interesting.

Define the following functions:

$$\begin{aligned} \tilde{h}_{a,b}(k) &= (ak + b) \mod p \\ h_{a,b}(k) &= \tilde{h}_{a,b}(k) \mod m = [(ak + b) \mod p] \mod m, \end{aligned}$$

for some $a \in Z_p^*$ and $b \in Z_p$.

e.g.) $p = 17$, $m = 6$, we have:

$$\begin{aligned} h_{3,4}(8) &= \tilde{h}_{3,4}(k) \mod 6 \\ &= [(3 \cdot 8 + 4) \mod 17] \mod 6 \\ &= 11 \mod 6 = 5 \end{aligned}$$

Now, we can construct a family of such functions:

$$H_{p,m} = \{h_{a,b} : a \in Z_p^* \text{ and } b \in Z_p\}$$

We claim that $H_{p,m}$ is universal.