COS 521: Advanced Algorithms Design February 3, 2004
Prof. Moses Charikar and Dr. Chandra Chekuri

Lecture 1: Hashing - Applications to Data Structures

Lecture: Prof. Moses Charikar *Scribe: Shien Jin Ong*

1 The Fundamental Data-structuring Problem

Consider the problem of implementing a dictionary to determine whether an element x from the universe $\mathcal{U}$ is in a given set S. The universe $\mathcal{U}$ is of size m, and the set S is of size s, with $S \subset \mathcal{U}$. We typically think of $s \ll m$. The operations to be supported are the following.

MakeSet(S): creates a new empty set S.

Insert(i, S): inserts item i into set S.

Delete(i, S): deletes item i from set S.

Find(k, S): returns 1 if $k \in S$, else returns 0.

A standard solution would be to store the elements of S as a balanced search tree. Then, all the above operations will run in $O(\log s)$ worst-case. In fact, representing S as random treaps or random skip lists would give an expected $O(\log s)$ running time. Could we do better, say creating a data structure such that all the above operations take only $O(1)$ time?

In the comparison-based model (where we are only allowed to do relative comparison between the elements), we cannot achieve a $o(\log s)$ running time, for or else we can sort n numbers in $o(n \log n)$ steps.

2 Hash Tables

If we go beyond the comparison-based model and assume that the (keys of the) elements are represented as integers, we will be able to achieve an $O(1)$ search time for the dictionary problem. We will work in the full RAM model, where accessing an integer and doing an arithmetic operation take $O(1)$ time. In fact, we achieve $O(1)$ search time for both the *static* and *dynamic* version of the dictionary problem.

1. In the *static* version, we are given a set S and preprocess it into a data structure that supports efficient Find(k, S) queries.

2. In the *dynamic* version, we are given an intermingled series of Find, Insert, and Delete operations. We are required to perform these operations efficiently.

Recall that m represents size of the universe $\mathcal{U}$. We think of m are a huge number. For instance, if $\mathcal{U}$ is the universe of all 32-bit integers, then $m = 2^{32}$. A trivial hashing solution that achieves an $O(1)$ search time would be to maintain an m-bit hash table H. Hash table H would have the property that $H(x) = 1$ if $x \in S$, and $H(x) = 0$ if $x \notin S$. The major drawback of this naive approach is the high storage requirement of the hash table, and the high initialization cost of the table.

Formally, we can define a *hash table* to be a data structure that consists of the following two components: a table T of size n indexed by $N = \{0, 1, \ldots, n-1\}$, and a hash function h mapping $\mathcal{U}$ to N. We index $\mathcal{U}$ by $\{0, 1, \ldots, m-1\}$. In addition, to avoid trivial hash functions we assume that $n < m$.

Ideally, we would like to store $x \in S$ in $T[h(x)]$. Then, to check if $x \in S$, we compute $h(x)$ and check that location in table T. Nevertheless, this poses a problem if too many elements in S hashes to the same value.

Definition 1 *A hash function $h : \mathcal{U} \to N$ is* perfect *for $S \subset \mathcal{U}$ if for all $x \neq y \in S$, we have that $h(x) \neq h(y)$.*

While it is clear that a perfect hash function can always be constructed for a specific set S of size less than n, there is no single hash function that is perfect for all sets. Therefore, to solve the dictionary problem, what we need is family (collection) of hash functions such that for any set S that is small, most hash functions in that family is perfect for it.

Definition 2 *Let H be a family of functions mapping $\mathcal{U}$ to N. We say that H is* 2-universal *if for all $x \neq y \in \mathcal{U}$, we have*

$$\Pr_{h \leftarrow H}[h(x) = h(y)] \leq \frac{1}{n}$$

Note that a (truly) random hash gives a collision probability of exactly $1/n$, thus a 2-universal family can be interpreted as a somewhat random hash function. Most construction of 2-universal hashing actually yields a stronger property known as *strongly 2-universal.*

Definition 3 *Let H be a family of functions mapping $\mathcal{U}$ to N. We say that H is* strongly 2-universal *if for all $x \neq y \in \mathcal{U}$ and for all $n_1, n_2 \in N$, we have*

$$\Pr_{h \leftarrow H}[h(x) = n_1 \textit{ and } h(y) = n_2] = \frac{1}{n^2}$$

If we have an efficient implementation of H satisfying the 2-universal property, we can solve both the static and dynamic version of the dictionary problem. Given the inputs, we randomly choose a hash function h from the family H and use it hash elements of the universe $\mathcal{U}$. We store $x \in S$ in $T[h(x)]$. If there are more than one element that hashes to the same value, we store them using a link list. The claim is that each link list in the table T will have expected depth $O(1)$. Thus, the Insert, Delete, and Find operations will run in expected time $O(1)$. The proof of the claim is left as an exercise for the students in the class.

The crucial property of 2-universal hash functions is that the collision probability is small, in our definition, less than $1/n$. While it would be in our interest to reduce this probability as much as possible, the following lemma shows that we cannot do much better when $m \gg n$.

Lemma 4 *For any family H of functions from $\mathcal{U}$ to N there exist $x, y \in \mathcal{U}$ such that*

$$\Pr_{h \leftarrow H}[h(x) = h(y)] \geq \frac{1}{n} - \frac{1}{m}$$

Proof: Define the indicator function for collision of x and y under h as follows.

$$\delta(x, y, h) = \begin{cases} 1 & \text{for } h(x) = h(y) \text{ and } x \neq y. \\ 0 & \text{otherwise.} \end{cases}$$

Consider a fixed hash function h, and for each $z \in N$ define

$$A_z = \{x \in \mathcal{U} : h(x) = z\}.$$

We know that $\sum_z |A_z| = m$. Because the total number of collisions is minimized when the sets A_z are all of the same size, we have

$$\begin{aligned} \sum_{x \in \mathcal{U}} \sum_{y \in \mathcal{U}} \delta(x, y, h) &= \sum_z |A_z|(|A_z| - 1) \\ &\geq n \left(\frac{m}{n}\left(\frac{m}{n} - 1\right)\right) \\ &= m^2 \left(\frac{1}{n} - \frac{1}{m}\right) \end{aligned}$$

The above equation holds for any fixed h. Therefore,

$$\sum_{x \in \mathcal{U}} \sum_{y \in \mathcal{U}} \sum_{h \in H} \delta(x, y, h) \geq m^2 |H| \left(\frac{1}{n} - \frac{1}{m}\right)$$

Consequently, there must exists a pair x, y such that

$$\sum_{h \in H} \delta(x, y, h) \geq |H| \left(\frac{1}{n} - \frac{1}{m}\right)$$

Since $\Pr_{h \leftarrow H}[h(x) = h(y)] = \frac{1}{|H|} \sum_{h \in H} \delta(x, y, h)$, our proof is complete. ■