Lab 3
Perfect Hashing

1 Overview

In this assignment, you're required to implement a perfect hashing data structure. We say a hash function is perfect for S if all lookups involve $O(1)$ work.

In section 2, background about universal hashing is provided. Sections 3 and 4 describe two methods for constructing perfect hash functions for a given set S. You're requied to design, analyze and implement a perfect hash table as described in sections 3 and 4.

2 Universal Hashing

A probability distribution H over hash functions from U to $\{1, ..., M\}$ is universal if for all $x \neq y$ in U, we have

$$Pr[h(x) = h(y)] \leq 1/M \tag{1}$$

2.1 Theorem 1

If H is universal, then for any set $S \subset U$, for any $x \in U$ (that we might want to insert or lookup), for a random h taken from H, the expected number of collisions between x and other elements in S is at most N/M.

2.2 Constructing a Universal Hash Family: the Matrix Method

Let's say keys are u-bits long. Say the table size M is power of 2, so an index is b-bits long with $M = 2^b$. What we'll do is pick h to be a random b-by-u 0/1 matrix, and define $h(x) = hx$, where we do addition mod 2. For instance:

$$\begin{array}{cccc} \text{h} & & \text{x} & \text{h(x)} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} & & \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} & = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \end{array}$$

We can show that for $x \neq y$, $Pr[h(x) = h(y)] = 1/M = 1/2^b$

3 $O(N^2)$-Space Solution

Say we are willing to have a table whose size is quadratic in the size N of our dictionary S. Then, here is an easy method. Let H be universal and $M = N^2$. Pick a random h from H and try it out, hashing everything in S. So, we just try it, and if we got any collisions, we just try a new h. On average, we will only need to do this twice.

4 $O(N)$-Space Solution

The main idea for this method is to use universal hash functions in a 2-level scheme.

The method is as follows. We will first hash into a table of size N using universal hashing. This will produce some collisions. However, we will then rehash each bin using Method 1, squaring the size of the bin to get zero collisions. So, the way to think of this scheme is that we have a first-level hash function h and first-level table A, and then N second-level hash functions $h_1, ..., h_N$ and N second-level tables $A_1, ..., A_N$. To lookup an element x, we first compute $i = h(x)$ and then find the element in $A_i[hi(x)]$.

5 Requirements

You're required to:

1. Implement an $O(N^2)$ as well as an $O(N)$-Space perfect hash table implemented as described in sections 3 and 4.
2. Verify that the hash table you constructed consumes $O(N^2)$-space in the the quadratic space method and $O(N)$-space in the linear space method.
3. Report the number of times required to re-build the hash table in the case of collision.
4. Deliver a report describing your implementation details.

You may assume that the keys will all be 32-bit integers. You will be provided with a text file, "keys.txt", containing a list of keys to be inserted in your hash table.

6 References

1. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to algorithms. MIT press.
2. Universal and Perfect Hashing, Lecture Notes by MIT OCW: http://www.cs.cmu.edu/ avrim/451f11/lectures/lect1004.pdf