Homework 3 Due Thursday Sept 30

- CLRS 8-3 (sorting variable-length items)
- CLRS 9-2 (weighted median)
- CLRS 11-1 (longest probe bound for hashing)

Chapter 11: Hashing

We use a table of size $m \ll n$ and select a function $h : U \to \mathcal{Z}_m$, which we call a **hash function**.

We put an element with key k to the slot $h(k)$, where collision is resolved by **chaining** the elements with the same "hash value."

Load Factor

To analyze efficiency of hashing we use the **load factor**, α, which is the average number of elements in a slot. This is a quantity that changes over time as the table acquires or loses elements.

What is the load factor of an m-slot hash table holding q objects?

Fundamental operations in hashing

Insertion

Insert the given item with key k *somewhere* in the list at the slot $h(k)$.

Where in the list should the item be inserted?

And how does the strategy influence the running time?

It should go at the beginning of the list.

Then the time for insertion is constant excluding the time for evaluation the hash function.

If all the elements happen to have the same hash value, then the time for insertion is proportion to the number of elements in the table, again excluding time for evaluation the hash function.

Deletion and Searching

To find or delete an element with key k, we scan the list at slot $h(k)$ to find it.

The worst-case scenario in searching and deletion is when the item is at the very end of the list.

Selection of the Hash Function

The performance of dynamic table operations is dependent on the choice of h.

Suppose that, for each of the three operations, selection of the target element is subject to a probability distribution P. That is, for each key x, $0 \leq x \leq n-1$, the probability that the key x is selected for an operation is $P(x)$.

Ideal hashing can be achieved when the hash function has a property such that for all y, $0 \leq y \leq m-1$, $\sum_{x:h(x)=y} P(x) = \frac{1}{m}$. Such a situation is called **simple uniform hashing**.

What is the expected number of elements in a slot under simple uniform hashing?

Under simple uniform hashing, for each slot, the probability that the target element is assigned to the slot is $\frac{1}{m}$. If there are q elements in the table, then for every slot the expected number of elements in the slot is q/m, which is the load factor. The expect time for searching in a list of length L is $L/2$ for successful search and L for unsuccessful search. So, we have the following theorem.

Theorem A If h is computable in a constant time **searching** under simple uniform hashing takes $\Theta(1+\alpha)$ on the average.

Unfortunately, designing a simple uniform hash function is usually impossible because P is not known.

Heuristics for Hash Functions

1. The division method

For all k, $h(k) = k \bmod m$.

It often happens that the keys are character strings interpreted in radix 2^p. Then

- $m = 2^p$ maps two keys with the same last character to the same hash value, and
- $m = 2^p - 1$ maps two keys composed of the same set of characters to the same hash value.

A heuristic choice for m is a prime far apart from any powers of 2, e.g. the prime closest to $2^p/3$.

2. The multiplication method

For all k, $h(k) = \lfloor m(kA - \lfloor kA \rfloor) \rfloor$, where $A \in (0\,..\,1)$ is a constant.

It is known that the value of m is not critical.

Universal Hashing

Suppose that a situation in which an application that employs hashing is repeatedly executed and in which the hash function is selected from a pool of hash functions at each execution.

Let $\mathcal{H}$ be the pool of hash functions.

We say that $\mathcal{H}$ is **universal** if, for all keys x and y, $x \neq y$, it holds that

(*) $\|\{h \in \mathcal{H} \mid h(x) = h(y)\}\| = \frac{\|\mathcal{H}\|}{m}$.

Suppose that at each execution the hash function h is chosen from $\mathcal{H}$ uniformly at random. Then, for all pairs (x, y), $x \neq y$, the probability that $h(x) = h(y)$ is $1/m$.

Usefulness of Universal Hashing

Theorem B Let $\mathcal{H}$ be a universal family of hash functions. Let S be a nonempty set of keys having cardinality at most m. Let x be any key in S. For $h \in \mathcal{H}$ chosen uniformly at random, the expected number of collisions in S with x is less than 1.

Proof Let E be the expected number in question. Then

$$E = \frac{\sum_{h \in \mathcal{H}} \sum_{y \in S, x \neq y} \sigma(h, x, y)}{\|\mathcal{H}\|},$$

where $\sigma(h, x, y) = 1$ if $h(x) = h(y)$ and 0 otherwise. This quantity is equal to

$$\frac{\sum_{y \in S, x \neq y} \sum_{h \in \mathcal{H}} \sigma(h, x, y)}{\|\mathcal{H}\|}.$$

By (*), this is

$$\sum_{y \in S, x \neq y} \frac{1}{m} \leq \frac{m-1}{m} < 1.$$

Designing Universal Hash Functions

Choose a prime p greater than all keys k.
Choose $a \in \{1...p-1\}$
Choose $b \in \{0...p-1\}$
$h_{a,b}(k) = ((ak+b) \pmod p) \pmod m$

Lemma C The class $\mathcal{H}_{p,m}$ is universal.

Universality of the Family

The class $\mathcal{H}_{p,m}$ is universal.

Proof For two distinct keys $k \neq l$:

$$r = (ak + b) \pmod{p}$$

$$s = (al + b) \pmod{p}$$

$$r - s = a(k - l) \pmod{p}$$

$$r \neq s$$

Furthermore we can solve for a and b:

$$a = ((r - s)((k - l)^{-1} \pmod{p})) \pmod{p}$$

$$b = (r - ak) \pmod{p}$$

So there is a one-to-one correspondence between pairs (a, b) and (r, s). If we choose (a, b) uniformly at random, (r, s) are uniformly distributed.

(r, s) are uniformly distributed.

Collision when $r = s \bmod m$. Given r, the number of colliding s is at most

$$\begin{aligned} \lceil p/m \rceil - 1 &\leq \frac{(p + m - 1)}{m} - 1 \\ &= (p - 1)/m \end{aligned}$$

$$\begin{aligned} Pr\{r = s \bmod m\} &\leq \frac{(p-1)/m}{p-1} \\ &= 1/m \end{aligned}$$

Open Addressing

Open addressing is an alternative to chaining, where collision is resolved by putting the element into an open slot.

To do this we assign to each key a sequence of addresses to search for an open slot. Formally, we extend the hash function to one that takes two inputs, namely a mapping from $U \times \mathcal{Z}_m$ to $\mathcal{Z}_m$, where for each $k \in U$ the slots $h(k,0), \ldots, h(k,m-1)$ are examined in this order and the first open one is used to store k. The sequence $\langle h(k,0), \ldots, h(k,m-1) \rangle$ is called the **probe sequence for k**. We design that each probe sequence is a permutation of $\mathcal{Z}_m$.

Deletion with Open Addressing

We cannot simply delete an element. When deleting an element we store in the slot a special value "DELETED" to signify that a key has been deleted. This means that the computation time for deletion depends on not the load factor in the original sense but on the load factor that even counts the slots that have the "DELETED" flag.

Can we store an item in a slot with "DELETED" label?

Insertion with Open Addressing

To insert an element with key k, we put it in the first open (either completely empty or having "DELETED") slot in the probe sequence for k.

Searching with Open Addressing

Searching is subject to the probe sequence of the key. It goes on until either the key is found or a completely open slot is encountered.

Three probe sequence schemes

1. Linear probing: Define

$$h(k,i) = (h'(k) + i) \bmod m,$$

where h' is an ordinary hash function from U to $\mathcal{Z}_m$.

2. Quadratic probing: Define

$$h(k,i) = (h'(k) + c_1 i + c_2 i^2) \bmod m,$$

where h' is an ordinary hash function and $c_1, c_2 \not\equiv 0 \pmod m$.

3. Double hashing: Pick two ordinary hash functions h_1, h_2 of U to $\mathcal{Z}_m$. Define

$$h(k,i) = (h_1(k) + i h_2(k)) \bmod m.$$

Primary clustering

Primary clustering is a situation in which there is a long line of occupied slots. Primary clustering is observed typically in linear probing

In linear probing, if every other slot is occupied, then the average unsuccessful search takes 1.5 probes. On the other hand, if there is a cluster of one half of the slots, then the average number of probes is

$$\frac{1}{m}\cdot\left(\frac{m}{2}+\sum_{i=1}^{m/2} i\right) = \frac{1}{2}+\frac{1}{2m}\cdot\frac{m}{2}\left(\frac{m}{2}+1\right) = \frac{m}{8}+\frac{3}{4}.$$

Analysis of Open-Address Hashing

Let m be the number of slots and let n be the number of occupied slots, including those that hold the "DELETED" label. Let $\beta = \frac{n}{m}$.

Theorem D Suppose that all the probe sequences (all $m!$ permutations) are equally likely to occur and that $\beta < 1$. Then, in an open-address hashing, the expected number of probes in an unsuccessful search is $\leq \frac{1}{1-\beta}$.

Proof For each $i \geq 0$, define p_i (respectively, q_i) to be the probability that exactly (respectively, at least) i probes are made before finding an open slot. The expected number of probes is $1 + \sum_{i=1}^{n} ip_i$.

For all i, $1 \leq i \leq n$, $q_i = \sum_{j=i}^{n} p_i$. So,

$$\sum_{i=1}^{n} ip_i = \sum_{i=1}^{n} q_i.$$

Note that

$$\begin{aligned} q_i &= \left(\frac{n}{m}\right)\left(\frac{n-1}{m-1}\right)\cdots\left(\frac{n-i+1}{m-i+1}\right) \leq \left(\frac{n}{m}\right)^i \\ &= \beta^i. \end{aligned}$$

So, the expected number of probes is at most $1 + \sum_{i=1}^{n} \beta^i \leq \sum_{i=0}^{\infty} \beta^i = \frac{1}{1-\beta}$. ∎