

COMP 460 Lecture Notes

R. I. Greenberg

1 Hash Tables

Supports “dictionary” operations:

- Insert: add elt. to set
- Delete: delete elt. from set
- Search: find elt. in set

General notation:

- $U = \{0, 1, 2, \dots, |U| - 1\}$ is universe of possible keys.
- K = set of n keys actually stored in “dictionary” at a given time.
- $T[0..m - 1]$ comprises *hash table* for storing K .

1.1 Hashing w. Chaining (Open Hashing)

Object w. key k goes into slot $h(k)$ of hash table. When keys hashed to same slot (collision), put them in a linked list.

1.1.1 Worst-Case Times

Insert: $O(1)$

Delete: $O(1)$ if *doubly*-linked lists (given ptr. to object to delete).

Same as searching if singly-linked lists.

Search: proportional to length of list of objects w. same *hash value* as key we’re searching for.

Worst-case $\Theta(n)$ but better avg. case.

1.1.2 Avg. Case Analysis of Searching

Search for object w. key k assuming keys used are random (\Rightarrow any given key equally likely to go into any of the m slots given a reasonable hash fn.)

Def.: load factor $\alpha = n/m =$ avg. no. of elts. in a chain

- Unsuccessful search (usual case if search for a random key):

Must walk through entire linked list in slot $h(k)$. Expected no. of elts. examined is α .

- Successful search

If search for an elt. that is in the table, we will examine the cell containing that elt. and, on average, cells for half the other elts. in slot $h(k)$. Expected no. of elts. examined is

$$1 + \frac{1}{2} \cdot \frac{n-1}{m} = 1 + \frac{\alpha}{2} - \frac{\alpha}{2n} .$$

- Search for a random key

Expected no. of elts. examined is

$$\frac{n}{|U|} \left(1 + \frac{\alpha}{2} - \frac{\alpha}{2n}\right) + \left(1 - \frac{n}{|U|}\right) \alpha = \alpha + \frac{n}{|U|} \left(1 - \frac{\alpha}{2} - \frac{\alpha}{2n}\right) .$$

In all three cases, the total search time is $\Theta(1 + \alpha)$, which equals $\Theta(1)$ if $n = O(m)$.

1.1.3 Universal Hashing

Similar to our move from QUICKSORT to RANDOMIZED-QUICKSORT. Instead of fixing the hash fn. (pivot location) in advance, choose it at random from a collection of good hash functions (all array positions). Then for any input, you will probably get good results.

Def.: A colln. \mathcal{H} of hash functions is universal if

$$\forall \text{ distinct } k, l \in U, |\{ h \in \mathcal{H} : h(k) = h(l) \}| \leq |\mathcal{H}| / m .$$

Thm: If h is chosen randomly from such a collection \mathcal{H} , the expected search time is $\Theta(1 + \alpha)$ regardless of the sequence of hash table operations (as long as the sequence of operations is not chosen by a malicious adversary who knows which member of \mathcal{H} has been selected).

Pf.: Essentially same as avg. case argument above, but now no need to assume random keys.

Corollary: Universal hashing w. chaining in a table w. m slots takes expected time $\Theta(n)$ time for *any* sequence of n Insert, Search, and Delete operations w. $O(m)$ Insert operations (or even an $O(m)$ bound on $|K|$ throughout the sequence).

$$\mathcal{H}_{p,m} = \{ h_{a,b} : a \in \mathbb{Z}_p^* \wedge b \in \mathbb{Z}_p \} \quad \text{w. } h_{a,b}(k) = ((ak + b) \bmod p) \bmod m \quad 3$$

1.2 Hash Functions

Keys can be non-numeric; easy to convert them to numbers, e.g.,

$$\begin{aligned} \mathbf{a} &= 1, \mathbf{b} = 2, \mathbf{c} = 3, \dots, \mathbf{z} = 26 \\ \mathbf{abc} &= 1 + 2 \times 26 + 3 \times 26^2 \end{aligned}$$

Then:

- division method: $h(k) = k \bmod m$.

A good choice for m is a prime not too close to a power of 2.

- multiplication method: $h(k) = \lfloor m(kA \bmod 1) \rfloor$ where $0 < A < 1$.

($x \bmod 1$ denotes the fractional part of x .)

Choosing m as a power of 2 eases computer implementation via bit operations. Knuth: $A \approx (\sqrt{5} - 1)/2$ is a good choice.

- universal hashing (one method):

Def.: For p prime, $\mathbb{Z}_p = \{0, 1, \dots, p-1\}$ and $\mathbb{Z}_p^* = \{1, \dots, p-1\}$.

Def.: $h_{a,b}(k) = ((ak + b) \bmod p) \bmod m$

Consider a family of $p(p-1)$ hash functions based on a prime $p \geq |U| > m$ defined by

$$\mathcal{H}_{p,m} = \{ h_{a,b} : a \in \mathbb{Z}_p^* \wedge b \in \mathbb{Z}_p \} .$$

Thm.: $\mathcal{H}_{p,m}$ is universal. (Recall def.: \forall distinct $k, l \in U, |\{ h \in \mathcal{H} : h(k) = h(l) \}| \leq |\mathcal{H}|/m$)

Pf.: Consider two distinct keys k and l . ($k \not\equiv l \pmod{p}$, since $p \geq |U|$.)

For $h_{a,b} \in \mathcal{H}_{p,m}$, let

$$\begin{aligned} r &= (ak + b) \bmod p \\ s &= (al + b) \bmod p \end{aligned}$$

$r \neq s$, since $r - s \equiv a(k - l) \bmod p$, and a and $k - l$ are nonzero mod p . (Number theory in Chapter 31 indicates a product of nonzero nos. is nonzero, even mod a prime.)

Furthermore, each choice of (a, b) (in $\mathbb{Z}_p^* \times \mathbb{Z}_p$) yields a different (r, s) .

(Same principle as above:

Suppose

$$r = (ak + b) \bmod p = (ck + d) \bmod p \quad (1)$$

$$s = (al + b) \bmod p = (cl + d) \bmod p \quad (2)$$

$$\mathcal{H}_{p,m} = \{ h_{a,b} : a \in \mathbb{Z}_p^* \wedge b \in \mathbb{Z}_p \} \quad \text{w. } h_{a,b}(k) = ((ak + b) \bmod p) \bmod m \quad 4$$

Then

$$\begin{aligned} (a - c)k + (b - d) &\equiv 0 \pmod{p} \text{ from (1)} \\ (a - c)l + (b - d) &\equiv 0 \pmod{p} \text{ from (2)} \end{aligned}$$

which implies $(a - c)(k - l) \equiv 0 \pmod{p}$. Since $k - l$ is nonzero mod p , and $a, b, c, d \in \mathbb{Z}_p$, we have $a = c \wedge b = d$.)

So there is a 1-1 correspondence between the $p(p - 1)$ pairs $(a, b) \in \mathbb{Z}_p^* \times \mathbb{Z}_p$ and the $p(p - 1)$ pairs $(r, s) \in \mathbb{Z}_p \times \mathbb{Z}_p$ w. $r \neq s$.

So the no. of fns. in $\mathcal{H}_{p,m}$ that map k and l to the same slot is the no. of $(r, s) \in \mathbb{Z}_p \times \mathbb{Z}_p$ w. $r \neq s$ and $r \equiv s \pmod{m}$. There are p choices for r . For each one, the no. of $s \in \mathbb{Z}_p$ w. $r \equiv s \pmod{m}$ is at most $\lceil p/m \rceil$, and the no. of such values $\neq r$ is at most

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

so the no. of (r, s) pairs meeting our criteria is $\leq p(p - 1)/m$. Thus the fraction of the $p(p - 1)$ fns. in $\mathcal{H}_{p,m}$ that map k and l to the same slot is at most $1/m$.

Q. E. D.

1.3 Open Addressing (Closed Hashing)

Instead of using chaining to handle collisions, look at a sequence of table positions

$$h(k, 0), h(k, 1), h(k, 2), \dots, h(k, m - 1)$$

until find an empty one to insert into. (Need $n \leq m$ here.)

Similarly, searching looks through seq. of positions until find the key or an empty slot. This makes deletion problematical. Either outlaw deletion, or replace deleted keys w. a special key, DELETED.

- Linear probing: $h(k, i) = (h'(k) + i) \bmod m$.

Probes consecutive table locations. Tends to have bad clustering (*primary clustering*).

- Quadratic probing: $h(k, i) = (h'(k) + c_1 i + c_2 i^2) \bmod m$.

Better than linear probing, but probe seq. still depends only on $h'(k)$. *secondary clustering*.

- Double hashing: $h(k, i) = (h_1(k) + i h_2(k)) \bmod m$.

Comes closest to the theoretical ideal of *uniform hashing*, where each key is equally likely to have any of the $m!$ permutations of $\{0, 1, \dots, m - 1\}$ as its probe seq.

Book analyzes uniform hashing.