CS 240: Data Structures and Data Management
Module 7 Study Guide

Taylor J. Smith — Spring 2017

Key Concepts

- **Hashing** allows us to achieve $O(1)$ operations (on average) for the Dictionary ADT.
- The simplest "hash", called **direct addressing**, stores each key k in its own position in an array of size M.
- To save space, we can use hash functions that may map multiple keys to the same position of an array.
- A hash function is a map $h : U \to \{0, 1, \ldots, M - 1\}$ from a universe of keys U to an index value.
- Collision resolution:
  - **Chaining** — use a linked list to maintain keys that map to the same position in a hash table
  - **Linear probing** — $h(k, i) = (h(k) + i) \bmod M$ for some hash function h
  - **Double hashing** — $h(k, i) = (h_1(k) + i \cdot h_2(k)) \bmod M$ for some hash functions h_1 and h_2
  - **Cuckoo hashing** — insert a key using h_1; if another key is displaced, reinsert that key using h_2
- Choosing a good hash function:
  - Hash functions should be efficient to compute, unrelated to patterns in the data, and dependent on all parts of the key
  - **Division method** — $h(k) = k \bmod M$
  - **Multiplication method** — $h(k) = \lfloor M(kA - \lfloor kA \rfloor) \rfloor$ for some $0 < A < 1$
  - Often, an irrational constant like $\varphi = \frac{\sqrt{5}-1}{2}$ works well for the value A
- We can hash multidimensional data by flattening it in some way (e.g., converting English letters to ASCII)

Suggested Readings

- **Sedgewick:** Chapter 14 (Hashing)
- **CLRS:** Chapter 11 (Hash Tables)
- **Goodrich/Tamassia:** 2.5 (Dictionaries and Hash Tables)

Practice Questions

Sedgewick

14.10. Prove that $(((ax) \bmod M)+b) \bmod M = (ax+b) \bmod M$, assuming that a, b, x, and M are all nonnegative integers.

14.16. How long could it take in the worst case to insert N keys into an initially empty table, using chaining with *(i)* unordered lists and *(ii)* ordered lists?

14.17. Give the contents of the hash table that results when you insert items with the keys E A S Y Q U T I O N in that order into an initially empty table of $M = 5$ lists, using chaining with unordered lists. Use the hash function $h(k) = 11k \bmod M$ to transform the kth letter of the alphabet into a table index.

14.18. Answer Exercise 14.17, but use ordered lists. Does your answer depend on the order in which you insert the items?

14.24. How long could it take, in the worst case, to insert N keys into an initially empty table, using linear probing?

14.25. Give the contents of the hash table that results when you insert items with the keys E A S Y Q U T I O N in that order into an initially empty table of size $M = 16$ using linear probing. Use the hash function $h(k) = 11k \bmod M$ to transform the kth letter of the alphabet into a table index.

14.26. Answer Exercise 14.25 for $M = 10$.

14.31. Give the contents of the hash table that results when you insert items with the keys E A S Y Q U T I O N in that order into an initially empty table of size $M = 16$ using double hashing. Use the hash function $h_1(k) = 11k \bmod M$ for the initial probe and the second hash function $h_2(k) = (k \bmod 3) + 1$ for the search increment (when the key is the kth letter of the alphabet).

14.32. Answer Exercise 14.31 for $M = 10$.

CLRS

11.1-1. Suppose that a dictionary S is represented by a direct-address table T of length m. Describe a procedure that finds the maximum element of S. What is the worst-case performance of your procedure?

11.2-1. Suppose we use a hash function h to hash n distinct keys into an array T of length m. Assuming simple uniform hashing, what is the expected number of collisions? More precisely, what is the expected cardinality of $\{\{k, l\} \mid k \neq l \text{ and } h(k) = h(l)\}$?

(*Note:* "Simple uniform hashing" means that any given element is equally likely to hash into any of the m slots, independently of where any other element has hashed to.)

11.2-2. Demonstrate what happens when we insert the keys $5, 28, 19, 15, 20, 33, 12, 17, 10$ into a hash table with collisions resolved by chaining. Let the table have 9 slots, and let the hash function be $h(k) = k \bmod 9$.

11.2-3. Professor Marley hypothesizes that he can obtain substantial performance gains by modifying the chaining scheme to keep each list in sorted order. How does the professor's modification affect the running time for successful searches, unsuccessful searches, insertions, and deletions?

11.3-4. Consider a hash table of size $m = 1000$ and a corresponding hash function $h(k) = \lfloor m\,(kA \bmod 1)\rfloor$ for $A = (\sqrt{5} - 1)/2$. Compute the locations to which the keys 61, 62, 63, 64, and 65 are mapped.

11.4-1. Consider inserting the keys $10, 22, 31, 4, 15, 28, 17, 88, 59$ into a hash table of length $m = 11$ using open addressing with the auxiliary hash function $h'(k) = k$. Illustrate the result of inserting these keys using linear probing, using quadratic probing with $c_1 = 1$ and $c_2 = 3$, and using double hashing with $h_1(k) = k$ and $h_2(k) = 1 + (k \bmod (m - 1))$.

11.4-2. Write pseudocode for the operation HASH-DELETE, and modify the pseudocode for the operation HASH-INSERT to handle the special value DELETED.

(*Note:* The special value DELETED marks the position of deleted elements in a hash table, and is distinct from the special value NIL, which marks an empty position.)

```
HASH-INSERT($T, k$)
  $i = 0$
  repeat
      $j = h(k, i)$
      if $T[j]$ == NIL
          $T[j] = k$
          return $j$
      else $i = i + 1$
  until $i$ == $m$
  error "hash table overflow"
```

11.4-4. Suppose that we use double hashing to resolve collisions—that is, we use the hash function $h(k, i) = (h_1(k) + ih_2(k)) \bmod m$. Show that if m and $h_2(k)$ have greatest common divisor $d \geq 1$ for some key k, then an unsuccessful search for key k examines $(1/d)$th of the hash table before returning to slot $h_1(k)$. Thus, when $d = 1$, so that m and $h_2(k)$ are relatively prime, the search may examine the entire hash table.

(*Hint:* See Chapter 31 of CLRS for more details about number-theoretic algorithms.)

Goodrich/Tamassia

R-2.19. Draw the 11-item hash table resulting from hashing the keys 12, 44, 13, 88, 23, 94, 11, 39, 20, 16, and 5, using the hash function $h(k) = (2k + 5) \bmod 11$ and assuming collisions are handled by chaining.

R-2.20. What is the result of the previous exercise, assuming collisions are handled by linear probing?

R-2.22. What is the result of Exercise R-2.19, assuming collisions are handled by double hashing using a secondary hash function $h'(k) = 7 - (k \bmod 7)$?

R-2.24. Show the result of rehashing the hash table shown in Figure 2.55 into a table of size 19 using the new hash function $h(k) = 2k \bmod 19$.

Additional Practice Questions

1. (Goodrich and Tamassia, *Algorithm Design*)

   The *universal hashing* method is guaranteed to give us good performance when hashing a set of keys in the range $[0..N-1]$ into a table of size M. With this method, we take p to be a prime number between N and $2N$, and we define our hash function to be of the form

$$h_{a,b}(k) = (ak + b \bmod p) \bmod M$$

   for integers $0 < a < p$ and $0 \leq b < p$.

   Implement a method for finding such a prime p by using the *sieve algorithm*. In this algorithm, we allocate a $2N$ cell Boolean array A, such that cell i is associated with the integer i. We then initialize the array cells to all be "true" and we "mark off" all the cells that are multiples of 2, 3, 5, 7, and so on. This process can stop after it reaches a number larger than $\sqrt{2N}$.

   (*Note:* The fact that there exists at least one prime number p with $N < p < 2N$ for all $N > 1$ is known as "Bertrand's postulate".)

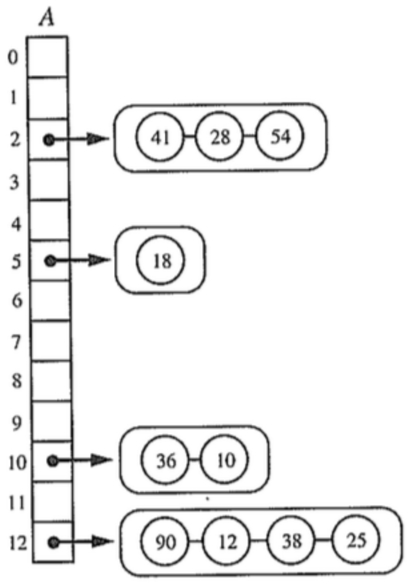

Figure 2.55: Example of a hash table of size 13, storing 10 integer keys, with collisions resolved by the chaining method. The hash function in this case is $h(k) = k \bmod 13$.