Chapter 11

Michelle Bodnar, Andrew Lohr

May 5, 2017

Exercise 11.1-1

Starting from the first index in T, keep track of the highest index so far that has a non NIL entry. This takes time $O(m)$.

Exercise 11.1-2

Start with a bit vector b which contains a 1 in position k if k is in the dynamic set, and a 0 otherwise. To search, we return true if $b[x] == 1$. To insert x, set $b[x] = 1$. To delete x, set $b[x] = 0$. Each of these takes $O(1)$ time.

Exercise 11.1-3

You could have each entry in the table be either a pointer to a doubly linked list containing all the objects with that key, or NIL if there are none. search just returns the first element in the list corresponding to the given key. Since all the elements in the list have that same key, it doesn't matter which search returns. Insert just adds to the start of the doubly linked list. Finally, deletion can be done in constant time in a doubly linked list, see problem 10-1

Exercise 11.1-4

The additional data structure will be a doubly linked list S which will behave in many ways like a stack. Initially, set S to be empty, and do nothing to initialize the huge array. Each object stored in the huge array will have two parts: the key value, and a pointer to an element of S, which contains a pointer back to the object in the huge array. To insert x, add an element y to the stack which contains a pointer to position x in the huge array. Update position $A[x]$ in the huge array A to contain a pointer to y in S. To search for x, go to position x of A and go to the location stored there. If that location is an element of S which contains a pointer to $A[x]$, then we know x is in A. Otherwise, $x \notin A$. To delete x, delete the element of S which is pointed to by $A[x]$. Each of these takes $O(1)$ and there are at most as many elements in S as there are valid elements in A.

Exercise 11.2-1

Under the assumption of simple uniform hashing, we will use linearity of expectation to compute this. Suppose that all the keys are totally ordered $\{k_1, \ldots, k_n\}$. Let X_i be the number of $\ell > k_i$ so that $h(\ell) = h(k_i)$. Note, that this is the same thing as $\sum_{j>i} Pr(h(k_j) = h(k_i)) = \sum_{j>i} 1/m = (n-i)/m$. Then, by linearity of expectation, the number of collisions is the sum of the number of collisions for each possible smallest element in the collision. The expected number of collisions is $\sum_{i=1}^{n} \frac{n-i}{m} = \frac{n^2 - \frac{n(n+1)}{2}}{m} = \frac{n^2-n}{2m}$

Exercise 11.2-2

Label the slots of our table be $0, 1, 2, \ldots, 8$. Numbers which appear to the left in the table have been inserted later.

0	$\emptyset$
1	10, 19, 28
2	20
3	12
4	$\emptyset$
5	5
6	33, 15
7	$\emptyset$
8	17

Exercise 11.2-3

Both kinds of searches become expected runtime of $\Theta(1 + \lg(\alpha))$. Insertions and deletions stay $\Theta(1 + \alpha)$ because the time to insert into or delete from a sorted list is linear.

Exercise 11.2-4

The flag in each slot of the hash table will be 1 if the element contains a value, and 0 if it is free. The free list must be doubly linked. Search is unmodified, so it has expected time $O(1)$. To insert an element x, first check if $T[h(x.key)]$ is free. If it is, delete $T[h(x.key)]$ and change the flag of $T[h(x.key)]$ to 1. If it wasn't free to begin with, simply insert $x.key$ at the start of the list stored there. To delete, first check if $x.prev$ and $x.next$ are NIL. If they are, then the list will be empty upon deletion of x, so insert $T[h(x.key)]$ into the free list, update the flag of $T[h(x.key)]$ to 0, and delete x from the list it's stored in. Since deletion of an element from a singly linked list isn't $O(1)$, we must use a doubly linked list. All other operations are $O(1)$.

Exercise 11.2-5

There is a subset of size n hashing to the same spot, because if each spot only

had $n-1$ elements hashing to it, then the universe could only be size $(n-1)m$. The worst case searching time would be if all of the elements that we put in the hashtable were this subset of size n all going to the same spot, which is linear.

Exercise 11.2-6

Choose one of the m spots in the hash table at random. Let n_k denote the number of elements stored at $T[k]$. Next pick a number x from 1 to L uniformly at random. If $x < n_j$, then return the x^{th} element on the list. Otherwise, repeat this process. Any element in the hash table will be selected with probability $1/mL$, so we return any key with equal probability. Let X be the random variable which counts the number of times we must repeat this process before we stop and p be the probability that we return on a given attempt. Then $E[X] = p(1+\alpha)+(1-p)(1+E[X])$ since we'd expect to take $1+\alpha$ steps to reach an element on the list, and since we know how many elements are on each list, if the element doesn't exist we'll know right away. Then we have $E[X] = \alpha + 1/p$. The probability of picking a particular element is $n/mL = \alpha/L$, so we have $E[X] = \alpha + L/\alpha = L(\alpha/L + 1/\alpha) = O(L(1 + 1/\alpha))$ since $\alpha \leq L$.

Exercise 11.3-1

If every element also contained a hash of the long character string, when we are searching for the desired element, we'll first check if the hashvalue of the node in the linked list, and move on if it disagrees. This can increase the runtime by a factor proportional to the length of the long character strings.

Exercise 11.3-2

Compute the value of the first character mod m, add the value of the second character mod m, add the value of the third character mod m to that, and so on, until all r characters have been taken care of.

Exercise 11.3-3

We will show that each string hashes to the sum of it's digits mod $2^p - 1$. We will do this by induction on the length of the string. As a base case, suppose the string is a single character, then the value of that character is the value of k which is then taken mod m. Now, for an inductive step, let $w = w_1 w_2$ where $|w_1| \geq 1$ and $|w_2| = 1$. Suppose, $h(w_1) = k_1$. Then, $h(w) = h(w_1)2^p + h(w_2)$ mod $2^p - 1 = h(w_1) + h(w_2) \mod 2^p - 1$. So, since $h(w_1)$ was the sum of all but the last digit mod m, and we are adding the last digit mod m, we have the desired conclusion.

Exercise 11.3-4

The keys 61, 62, 63, 64, and 65 are mapped to locations 700, 318, 936, 554,

and 172 respectively.

Exercise 11.3-5

As a simplifying assumption, assume that $|B|$ divides $|U|$. It's just a bit messier if it doesn't divide evenly.

Suppose to a contradiction that $\epsilon > \frac{1}{|B|} - \frac{1}{|U|}$. This means that for all pairs k, ℓ in U, we have that the number $n_{k,\ell}$ of hash functions in $\mathcal{H}$ that have a collision on those two elements satisfies $n_{k,\ell} \leq \frac{|\mathcal{H}|}{|B|} - \frac{|\mathcal{H}|}{|U|}$. So, summing over all pairs of elements in U, we have that the total number is less than$\leq \frac{|\mathcal{H}||U|^2}{2|B|} - \frac{|\mathcal{H}||U|}{2}$.

Any particular hash function must have that there are at least $|B|\binom{|U|/|B|}{2} = |B|\frac{|U|^2-|U||B|}{2|B|^2} = \frac{|U|^2}{2|B|} - \frac{|U|}{2}$ colliding pairs for that hash function, summing over all hash functions, we get that there are at least $|\mathcal{H}|\left(\frac{|U|^2}{2|B|} - \frac{|U|}{2}\right)$ colliding pairs total. Since we have that there are at most some number less than this many, we have a contradiction, and so must have the desired restriction on ϵ.

Exercise 11.3-6

Fix $b \in \mathbb{Z}_p$. By exercise 31.4-4, $h_b(x)$ collides with $h_b(y)$ for at most $n-1$ other $y \in U$. Since there are a total of p possible values that h_b takes on, the probability that $h_b(x) = h_b(y)$ is bounded from above by $\frac{n-1}{p}$. Since this holds for any value of b, $\mathcal{H}$ is ((n-1)/p)-universal.

Exercise 11.4-1

This is what the array will look like after each insertion when using linear probing:

									10
22									10
22								31	10
22				4				31	10
22				4	15			31	10
22				4	15	28		31	10
22				4	15	28	17	31	10
22	88			4	15	28	17	31	10
22	88			4	15	28	17	59	31	10

For quadradic probing, it will look identical until there is a collision on inserting the fifth element. Then, it is

										10
22										10
22									31	10
22				4					31	10
22				4				15	31	10
22				4		28		15	31	10
22			17	4		28		15	31	10
22		88	17	4		28		15	31	10
22		88	17	4		28	59	15	31	10

Lastly, for double hashing, we have:

										10
22										10
22									31	10
22				4					31	10
22				4	15				31	10
22				4	15	28			31	10
22			17	4	15	28			31	10
22			17	4	15	28	88		31	10
22		59	17	4	15	28	88		31	10

Exercise 11.4-2

Algorithm 1 HASH-DELETE(T,k)

$i = HASH - SEARCH(T, k)$
$T[i] = DELETED$

We modify INSERT by changing line 4 to: **if** $T[j] == NIL$ or $T[j] == DELETED$.

Exercise 11.4-3

For the $\alpha = 3/4$ case, we just plug into theorems 11.6 and 11.8 respectively to get that for a unsuccessful search, the expected number of probes is bounded by 4. And for a successful search, the expected number of probes is bounded by $\frac{4}{3}\ln(4)$.

For $\alpha = 7/8$. The bound for expected number of probes of unsuccessful is 8, and for successful is $\frac{8}{7}\ln(8)$.

Exercise 11.4-4

Write $h_2(k) = dc_1$ and $m = dc_2$ for constants c_1 and c_2. When we have examined $(1/d)^{th}$ of the table entries, this corresponds to $m/d = c_2$ of them. The entry that we check at this point is $[h_1(k) + (m/d)h_2(k)] \mod m = [h_1(k) + (m/d)(dc_1)] \mod m \equiv h_1(k)$. When $d = 1$, we may have to examine the entire

hash table.

Exercise 11.4-5

We will try to find a rational solution, let $\alpha = m/n$. Using theorems 11.6 and 11.8, we have to solve the equation

$$\frac{1}{1-\alpha} = \frac{2}{\alpha}\ln(\frac{1}{1-\alpha})$$

Unfortunately, this transcendental equation cannot be solved using simple techniques. There is an exact solution using the Lambert W function of

$$\frac{1}{2}\frac{1+2LambertW(-1,-(\frac{1}{2})*exp(-\frac{1}{2}))}{LambertW(-1,-(\frac{1}{2})*exp(-\frac{1}{2}))}$$

Which evaluates to approximately $\alpha = .7153$.

Exercise 11.5-1

Let $A_{j,k}$ be the event that j and k hash to different things. Due to uniform hashing, $Pr(A_{j,k}) = \frac{m-1}{m}$. Also, we can say that there is a negative correlation between the events. That is, if we know that several pairs of elements hashed to the same thing, then we can only decrease the likelihood that some other pair hashed to different things. This gets us that the probability that all events happen is $\leq$ the probability that the all happened if they were independent, so,

$$Pr(\cap_{j,k}A_{j,k}) \leq \left(\frac{m-1}{m}\right)^{\binom{n}{2}} \leq \left(e^{-1/m}\right)^{\frac{n(n-1)}{2}} = e^{\frac{-n(n-1)}{2m}}$$

So, if we have that that n exceeds m, the $-n^2/(2m)$ term is much bigger than the $n/2m$ term, so, the exponent is going to $-\infty$, which means the probability is going to 0.

Problem 11-1

a. The index for each probe is computed uniformly from among all the possible indices. Since we have $n \leq m/2$, we know that there are at least half of the indices empty at any stage. So, for more than k probes to be required, we would need that in each of k first probes, we probed a vertex that already had an entry, this has probability less than $1/2$, so the probability of it happening each time is $< 1/(2^k)$.

b. Using the result from the previous part with $k = 2\lg(n)$, we have that the probability that so many probes will be required is

$$< 2^{-2\lg(n)} = 2^{\lg(n^{-2})} = n^{-2} = \frac{1}{n^2}$$

c. We apply a union bound followed by the results of the previous part

$$Pr\{X > 2\lg(n)\} = Pr\{\vee_i X_i > 2\lg(n)\} \leq \sum_i Pr\{X_i > 2\lg(n)\} \leq \sum_i \frac{1}{n^2} = \frac{n}{n^2} = \frac{1}{n}$$

d. The longest possible length of a probe sequence is n, as we would try checking every single entry already placed in the array. We also know that the probability that a sequence of length more than $2\lg(n)$ is required is $\leq 1/n$. So, we have that the largest the expected value can be is

$$E[X] \leq Pr\{X \leq 2\lg(n)\}2\lg(n) + Pr\{X > \lg(n)\}n = \frac{n-1}{n}2\lg(n) + \frac{1}{n}n = 2\lg(n) + 1 - 2\frac{\lg(n)}{n} \in O(\lg(n))$$

Problem 11-2

a. Let Q_k denote the probability that exactly k keys hash to a particular slot. There are $\binom{n}{k}$ ways to select the k keys, they hash to that spot with probability $\left(\frac{1}{n}\right)^k$, and the remaining $n-k$ keys hash to the remaining $n-1$ slots with probability $\left(\frac{n-1}{n}\right)^{n-k}$. Thus,

$$Q_k = \left(\frac{1}{n}\right)^k \left(\frac{n-1}{n}\right)^{n-k} \binom{n}{k}.$$

b. The probability that the slot containing the most keys contains exactly k is bounded above by the probability that some slot contains k keys. There are n slots which we could select to contain those k, so $P_k \leq nQ_k$.

c. Using the fact that $\binom{n}{k} \leq \left(\frac{ne}{k}\right)^k$ we have $Q_k \leq \frac{(n-1)^{n-k}}{n^n}\left(\frac{ne}{k}\right)^k \leq e^k/k^k$.

d. We'd like to make $Q_{k_0} < n^3$. Taking the log of both sides and expanding using log rules this becomes

$$\frac{c\lg n}{\lg\lg n}(\lg e + \lg\lg\lg n - \lg c - \lg\lg n) < -3\lg n.$$

dividing both sides by $-\lg\lg n$ this is equivalent to

$$3 < -\frac{c\lg e}{\lg\lg n} - \frac{\lg\lg\lg n}{\lg\lg n} + \frac{\lg c}{\lg\lg n} + c.$$

As $n \to \infty$, all terms except the last go to 0, so choosing $c = 4$ works for all $n > N$ for some fixed value of N. Let c_i denote the value that works for $n = i$. Then by letting $c = \max_i(c_i, 4)$ we obtain the desired value of c. By part (b), we have $P_{k_0} \leq nQ_{k_0} < n/n^3 = 1/n^2$. Since Q_k is a decreasing function of k, we conclude that $P_k < 1/n^2$ for all $k \geq k_0$.

e. This comes from computing expectation by conditioning, and bounding M from above by n in the first term. From the bound in (d) we have $nP(M > \frac{c \lg n}{\lg \lg n}) < 1$. For the second term, $\sum_{i=1}^{d} P(M = i) \leq 1$ where $d = c \lg n / \lg \lg n$, so the asymptotic expectation follows.

Problem 11-3

a. At each step, we are increasing the amount we increase by by 1, so, this leads to the "Gauss numbers" which have formula $\frac{i^2+i}{2}$. So, we have $c_1 = c_2 = \frac{1}{2}$.

b. To show that this algorithm examines every number, we will show that every number that it examines is distinct. Then, since it examines m numbers total, this will imply that every number is visited. Suppose that we visited the same position on two distinct rounds $i < i' < m$, then,

$$h(k) + \frac{i + i^2}{2} \equiv h(k) + \frac{i' + i'^2}{2} \mod m$$

Which means

$$\frac{i + i^2}{2} \equiv \frac{i' + i'^2}{2} \mod m$$

So,

$$i + i^2 \equiv i' + i'^2 \mod 2m$$

Rearranging,

$$i'^2 - i^2 = (i + i')(i' - i) \equiv i' - i \mod 2m$$

So,

$$(i' - i)(i' + i - 1) \equiv 0 \mod 2m$$

Now, we split into two cases based on whether $i - i'$ is even or odd. If $i' - i$ is odd, then we can cancel it on both sides, because it has no common factor with 2m (because m is a power of 2). This leaves us with

$$i + i' + 1 \equiv 0 \mod 2m$$

Since $i < i'$, we have $i + i' + 1 \leq 2i' < 2m$, so, we must have $i + i' + 1 = 0$, a contradiction to our assumption that such i and i' existed.

If we have that $i' - i$ is even, then $i + i' + 1$ is odd, so it has no factors in common with 2m. Similarly, this means that we can cancel out that factor to get $i' - i \equiv 0 \mod 2m$. However, we have that $|i' - i| \leq i' + i < 2i' < 2m$. So, we must have $i' - i = 0$, a contradiction to how we picked i and i'.

Problem 11-4

a. Let $k, l \in U$ be arbitrary. Then $P(h(k) = h(l)) = P(< h(k), h(l) >=< x, x >$) for some $x \in [m]$. Since h comes from a set of 2-universal hash functions, this is equally likely to be any of the m^2 sequences. There are m possible values of x which would cause a collision, so the probability of collision is $\frac{m}{m^2} = \frac{1}{m}$.

b. The probability of collision is $\frac{1}{p}$, so $\mathcal{H}$ is universal. Now consider the tuple $z =< 0, 0, \ldots, 0 >$. Then $h_a(x) = h_b(x) = 0$ for any $a, b \in U$, so the sequence $< z, x^{(2)} >$ is equally likely to be any of p sequences starting with 0, but can't be any of the other $p^2 - p$ sequences, so $\mathcal{H}$ is not 2-universal.

c. Let $x, y \in U$ be fixed, distinct n-tuples. As a_i and b range over $\mathbb{Z}_p$, $h'_{ab}(x)$ is equally likely to achieve every value from 1 to p since for any sequence a, we can let b vary from 1 to $p - 1$. Thus, $< h'_{ab}(x), h'_{ab}(y) >$ is equally likely to be any of the p^2 sequences, so $\mathcal{H}$ is 2-universal.

d. Since $\mathcal{H}$ is 2-universal, there are p other functions which map m to $h(m)$, and the adversary has no way of knowing which one of these Alice and Bob have agreed on in advance, so the best he can do is try one of them, which will succeed in fooling Bob with probability $1/p$.