ECE608 Homework #7 Solution

(1) CLR 11.1-1

To find the maximum element of the set S, it requires searching the entire table T in the worst case. Note that NIL is returned if there are no elements in the table T; otherwise, the index to the largest element is returned.

GET-TABLE-MAX(T)

1. $maxIndex = \text{length}(T) - 1$
2. **while** $maxIndex \geq 0$
3. **do if** $T[maxIndex] = \text{NIL}$
4. **then** $maxIndex \leftarrow maxIndex$ - 1
5. **else return** $maxIndex$
6. **return** NIL

This procedure requires $\Theta(m)$ time in the worst case, where m is the table size.

(2) CLR 11.1-4

The difficulty of this problem results from the fact that an uninitialized array entry may contain garbage instead of a NIL. To address the fact that the initilization of the table is impractical, we use a stack data structure to perform a validity check. The stack size is of the maximum dictionary size. However, only the keys inserted in the dictionary are stored on the stack and the other entries are uninitialized. Let T be the huge table in the question and S be the stack. Let top represent the top of the stack S, which is initialized to 0.

The basic idea is that the entry in T validates S and entry in S validates T. This can be achieved by storing key i to the stack S and storing the stack index (from the bottom of the stack) in $T[i].index$, whenever a key i's satellite data is inserted into the table T.

Therefore, if a key i is present, then it can be validated as follows. Let $k = T[i].index$. Now,

$S[k] = i$ and $T[S[k]] = k$ and $1 \leq k \leq top(S)$

The INSERT operation does the bookkeeping to setup pointers correctly as described below. Observe INSERT takes $O(1)$ time. Note, the INSERT assumes key is not already there.

INSERT(x, i)
$top \leftarrow top + 1$
$S[top] \leftarrow i$
$T[i].data \leftarrow x$
$T[i].index \leftarrow top$

For SEARCH operation, the validation cycle as described above is checked. If it is not true, the slot does not contain valid data. Observe SEARCH operation takes $O(1)$ time.

SEARCH(i)
$k \leftarrow T[i].index$
if $S[k] = i$ and $T[S[k]] = k$ and $1 \leq k \leq top$
  then return *T[i].data*
  else return NOT FOUND

To delete the satellite data associated with key i, assuming the key is present, we need to break the validating cycle. The trick is to ensure the stack doesn't have a hole after deletion. This can be acheived by moving the stack top to the deleted entry.

DELETE(i)

$S[T[i].index] \leftarrow S[top]$

$T[S[T[i].index]] \leftarrow T[i].index$

$top \leftarrow top - 1$

$T[i].index \leftarrow 0$

$T[i].data \leftarrow$ NIL

(3) CLR 11.2-1

Let X_{ij} be an indicator random variable, such that $1 \leq i < j \leq n$ such that $X_{ij} = I\{h(i) = h(j)\}$. Note that the probability that two keys hash to the same slot is simply $\sum_{i=1}^{m} \frac{1}{m^2} = \frac{1}{m}$ based on simple uniform hashing and the fact that they can hash to any of the m slots in the table. Thus by Lemma 5.1, $E[X_{ij}] = \frac{1}{m}$.

Let X be the random variable that counts the number of pairs of keys that hash to the same slot (i.e., the number of collisions). Then $X = \sum_{i=1}^{n} \sum_{j=i+1}^{n} X_{ij}$.

Hence,

$$\begin{aligned}
E[X] &= E[\sum_{i=1}^{n} \sum_{j=i+1}^{n} X_{ij}] \\
&= \sum_{i=1}^{n} \sum_{j=i+1}^{n} E[X_{ij}] \\
&= \sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{1}{m} \\
&= \sum_{i=1}^{n} \frac{n-i}{m} \\
&= \sum_{k=0}^{n-1} \frac{k}{m} \\
&= \frac{1}{m} \sum_{k=1}^{n-1} k \\
&= \frac{1}{m} \frac{n(n-1)}{2}
\end{aligned}$$

$$= \frac{n(n-1)}{2m}$$

Hence, the expected number of collisions is $\frac{n(n-1)}{2m}$.

(4) CLR 11.3-3

We know from p. 863 of CLR that if $a \equiv a' \bmod n$ and $b \equiv b' \bmod n$, then $a + b \equiv (a' + b') \bmod n$.

We have the hash function $h(k) = k \bmod m$, where $m = 2^p - 1$ and k is a character string in radix 2^p.

Given that the digits of x are $x_n x_{n-1} ... x_1 x_0$, we can evaluate the hash function $h(x) = (x_n \cdot 2^{pn} + x_{n-1} \cdot 2^{p(n-1)} + ... + x_0 \cdot 2^{p0}) \bmod (2^p - 1)$.

Since we know that:

$$\begin{aligned}
x_n \cdot 2^{pn} \bmod (2^p - 1) &= x_n \\
x_{n-1} \cdot 2^{p(n-1)} \bmod (2^p - 1) &= x_{n-1} \\
x_{n-2} \cdot 2^{p(n-2)} \bmod (2^p - 1) &= x_{n-2} \\
&\vdots \\
x_1 \cdot 2^{p1} \bmod (2^p - 1) &= x_1 \\
x_0 \cdot 2^{p0} \bmod (2^p - 1) &= x_0
\end{aligned}$$

Then it follows that $h(x) \equiv (x_n + x_{n-1} + ... + x_1 + x_0) \bmod (2^p - 1)$. Since the hash function is equivalent to the sum of the digits of $x \bmod (2^p - 1)$, any key derived by permuting the digits of x will have the same hash value.

Assume that a key x' is a permutation of the x_i's, $x' = x'_n x'_{n-1} \ldots x'_1 x'_0$. Since each x'_i has a 1-to-1 correspondence to an x_j, it follows that:

$$\begin{aligned} h(x') &= (x'_n \cdot 2^{pn} + x'_{n-1} \cdot 2^{p(n-1)} + ... + x'_0 \cdot 2^{p0}) \bmod (2^p - 1) \\ &= x'_n + x'_{n-1} + ... + x'_1 + x'_0 \bmod (2^p - 1) \\ &= x_n + x_{n-1} + ... + x_1 + x_0 \bmod (2^p - 1) \\ &= h(x) \end{aligned}$$

(5) CLR 11.3-5

Given $|U|$ keys to hash to a set of $|B|$ values, we argue that the least probability of collisions is achieved when each of the $|B|$ values is equally likely. Specifically if we view the keys as balls and the hash values as bins, then the collisions are minimized when each bin has the same number of balls (if $|U|$ is not a mutliple of $|B|$, then the number of balls in each bin may differ by 1). The proof is as follows: If there are x balls in a particular bin i.e., x keys hash to a particular hash-value, then the number of pairs of keys that collide are $\binom{x}{2} = \frac{x(x-1)}{2}$. Given a particular hash function h_i the $|U|$ keys hash to the $|B|$ possible values such that there are x_i keys that hash to the $i'th$ hash-value in the set $B = 1, 2, 3, ...|B|$. Also $\sum_{i=1}^{|B|} x_i = |U|$. The total number of pairs that collide can be computed as:

$$Y = \sum_{i=1}^{|B|} \frac{x_i(x_i-1)}{2} \quad .$$

Lemma: The real values $x_1, x_2, ...x_{|B|}$ that minimize the sum Y , while satisfying $\sum_{i=1}^{k} x_i = |U|$. To prove this suppose that $x_1, x_2, ...x_k$ minimize Y and they are not all equal in value. Then there must be at least one pair x_i and x_j that are unequal. Then we can show that:

$$\frac{x'_i(x'_i-1)}{2} + \frac{x'_j(x'_j-1)}{2} \geq \frac{m(m-1)}{2}$$

where $m = \frac{x'_i(x'_j+1)}{2}$. So assign value m instead of x_i and x_j and hence obtain a lower value for Y , contradicting our initial assumption. So it must be the case that the values of the x_is that minimizes the function Y are $x_1 = x_2 = x_3 ... = x_k = \frac{|U|}{|B|}$.

This lemma gives us a lower bound on the value of Y (and hence the number of collision pairs) for all possible families of universal hashing functions. Note that when $|U|$ is not a mutliple of $|B|$ the number of collision will be larger than this minimum bound because then it is not possible to make all $x_i = \frac{|U|}{|B|}$. Using the above lemma and the obtained value for x_i , we see that the minimum number of colliding pairs that any hash function produces is lower bounded by:

$$\sum_{i=1}^{|B|} \frac{x_i(x_i-1)}{2} \geq \frac{1}{2}|B|\frac{|U|}{|B|}(\frac{|U|}{|B|} - 1)$$

If there are n hash functions, then hashing all $|U|$ keys using all n functions would produce a minimum of $\frac{n}{2}|B|\frac{|U|}{|B|}(\frac{|U|}{|B|}-1)$ total colliding pairs. The total number of pairs obtainable from $|U|$ keys is $\binom{|U|}{2}$. So the average number of colliding hash functions per key-pair is:

$$\frac{n\frac{|B|}{2}\frac{|U|}{|B|}(\frac{|U|}{|B|}-1)}{\binom{|U|}{2}}$$

And there must be at least some key pair for which the number of colliding hash-functions is no less that the above stated average. When choosing the hash function uniformly randomly from n functions, the probability of a collision for this key pair can be written as:

$$p \geq \frac{\frac{|B|}{2}\frac{|U|}{|B|}(\frac{|U|}{|B|}-1)}{\binom{|U|}{2}}$$
$$p \geq \frac{|U|(\frac{|U|}{|B|}-1)}{|U|(|U|-1)}$$
$$p \geq \frac{(\frac{|U|}{|B|}-1)}{(|U|-1)}$$

$$p \geq \frac{(\frac{|U|}{|B|}-1)}{|U|}$$
$$p \geq \frac{|U|-|B|}{|U||B|}$$
$$p \geq \frac{1}{|B|} - \frac{1}{|U|}$$

We know that the above value of p gives a lower bound on the collision probablilty for any family of universal hashing functions. So for any -universal family of hash functions, the value of ϵ can not be smaller than p, since at least one key pair will have a collision proability of at least p in any hash function family. So ϵ is lower bounded by p which is lower bounded by the expression given above, as desired.

Informal :

Here is an intuitive way of thinking about the problem. We can think of key-pairs as bins. There are K bins, where K is $\binom{|U|}{2}$. We can think of hash functions as each placing (not tossing) balls in bins, placing either 0 or 1 ball in each bin. If hash function f places a ball in bin b, it means that that hash function collides that keypair.

Each hash function, f, places H_f balls into the bins. For simplicity, let $H_f = H$. If N hash functions do this, ($\sum_{f=1}^{N} H_f =$) NH balls have been placed into the bins. No matter how the placement is done, some bin must have gotten at least NH/K balls.

That bin represents a keypair that is collided by NH/K out of N hash functions. Choosing the hash function uniformly randomly, that means H/K probability of selecting a hash function that collides that keypair.

The lemma lower-bounds H. H/K lower bounds the collision probability of the most likely to collide keypair, and thus lower bounds epsilon.

(6) CLR 11-1

(a) Assuming uniform hashing, a particular hash is equally likely to take on any value among $1, 2, 3...m$. When the ith insertion is attempted, there are $i1$ slots occupied out of the total m. So the probability of collision in the first probe is $\frac{i-1}{m}$. In the

second probe, insertion would be attempted out of the remaining slots, $i-2$ of which are occupied. So probability of the second probe colliding given that the first one collided is $\frac{i-2}{m}$. Following the same scheme we can obtain the probability that the ith insertion requires k probes:

$$Pr(X_i) \geq k = \tfrac{i-1}{m} . \tfrac{i-2}{m} . \tfrac{i-3}{m} \dots \tfrac{i-k}{m}$$

As $i < n$ we can substitute to get an upper bound:

$$(Pr(X_i) \geq k) \leq \tfrac{n}{m} . \tfrac{n}{m} . \tfrac{n}{m} \dots \tfrac{n}{m}$$

$$(Pr(X_i) \geq k) \leq \left(\tfrac{n}{m}\right)^k$$

$$(Pr(X_i) \geq k) \leq (\alpha)^k$$

$$(Pr(X_i) \geq k) \leq \left(\tfrac{1}{2}\right)^k$$

$$(Pr(X_i) \geq k) \leq (2)^{-k}$$

(b) Use $k = 2lgn$ in the result of part (a).

$$(Pr(X_i) \geq k) \leq (2)^{-k}$$

$$(Pr(X_i) \geq k) \leq (2)^{-2lgn}$$

$$(Pr(X_i) \geq k) \leq (n)^{-2lg2}$$

$$(Pr(X_i) \geq k) \leq \tfrac{1}{n^2}$$

(c) Define the random variable $X = max_{1 \leq i \leq n} X_i$ then:

$$X = max_{1 \leq i \leq n} X_i$$

$$(Pr(X > 2lgn) = Pr(\text{One out of } n \ X_i s \text{ is } > 2lgn)$$

$$(Pr(X > 2lgn) = Pr(X_1 > 2lgn \text{ or } X_2 > 2lgn \text{ or } ....X_n > 2lgn)$$

$$(Pr(X > 2lgn) \leq Pr(X_1 > 2lgn) + Pr(X_2 > 2lgn) + Pr(X_3 > 2lgn) + ... + Pr(X_n > 2lgn)$$

$$(Pr(X > 2lgn) \leq \tfrac{1}{n^2} + \tfrac{1}{n^2} + \tfrac{1}{n^2} \cdots \tfrac{1}{n^2}$$

$$(Pr(X > 2lgn) \leq n\tfrac{1}{n^2}$$

$$(Pr(X > 2lgn) \leq \tfrac{1}{n}$$

(d) Given that $Pr(X > 2lgn) \leq \frac{1}{n}$ we can compute the expected value of X as:

$$Pr(X > 2lgn) \leq \tfrac{1}{n}$$

Put $i = 2lgn$, or $n = 2^{\frac{i}{2}}$:

$$Pr(X > i) \leq \ 2^{-\frac{i}{2}}$$

$$Pr(X > 2lgn) \leq \tfrac{1}{n}$$

$$E[X] = \sum_{i=1}^{n} Pr(X \geq i)$$

$$E[X] = \sum_{i=1}^{n} 2^{-\frac{i}{2}} \leq \sum_{i=1}^{n} \frac{1}{2+i} \leq \sum_{i=1}^{n} \frac{1}{i}$$

$$E[X] \leq \sum_{i=1}^{n} \frac{1}{i}$$

$$E[X] \leq lgn$$

$$E[X] = O(lgn)$$

(7) CLR 11-2

(a) A particular key is hashed to a particular slot with probability $\frac{1}{n}$. Suppose we select k particular keys. The probability that these k keys are inserted into the slot in question and that all the other keys are inserted elsewhere is $(\frac{1}{n})^k(1-\frac{1}{n})^{n-k}$. Since there are $\frac{n!}{k!(n-k)!}$ ways we can choose our k keys, we get $Q_k = (\frac{1}{n})^k(1-\frac{1}{n})^{n-k}\frac{n!}{k!(n-k)!}$.

(b) Let x_i be the number of keys in slot i.

$$\begin{aligned}
P_k &= \Pr\{M = k\} \\
&= \Pr\{\max_i x_i = k\} \\
&= \Pr\{\exists i \text{ such that } x_i = k \text{ and } \forall i, x_i \leq k\} \\
&\leq \Pr\{\exists i \text{ such that } x_i = k\} \\
&= \Pr\{x_1 = k \text{ or } x_2 = k \text{ or } \ldots \text{ or } x_n = k\} \\
&\leq \sum_i \Pr\{x_i = k\} \text{ (by Equation (C.13))} \\
&= nQ_k
\end{aligned}$$

(c)

$$\begin{aligned} Q_k &= (\frac{1}{n})^k \cdot (1-\frac{1}{n})^{n-k} \cdot \frac{n!}{k!(n-k)!} \\ &< \frac{n!}{n^k k!(n-k)!}, \text{ because } (1-\frac{1}{n})^{n-k} < 1 \\ &< \frac{1}{k!}, \text{ because } \frac{n!}{(n-k)!} < n^k \end{aligned}$$

Because of Stirling's approximation, $\frac{1}{k!} \leq \frac{1}{\sqrt{2\pi k}\cdot(\frac{k}{e})^{k+\frac{1}{12k}}} < \frac{1}{(\frac{k}{e})^k} = (\frac{e}{k})^k$, thus, $Q_k < \frac{1}{k!} < \frac{e^k}{k^k}$.

(d) Remark: Notice that when $n = 2$, $\lg\lg n = 0$, so to be precise, we need to assume $n \geq 3$. Since we showed $Q_{k_0} < \frac{e^{k_0}}{k_0^{k_0}}$, it suffices to show that $\frac{e^{k_0}}{k_0^{k_0}} < \frac{1}{n^3}$ or equivalently $\frac{k_0^{k_0}}{e^{k_0}} > n^3$.

Taking lg of both sides,

$$\begin{aligned} 3\lg n &< k_0(\lg k_0 - \lg e) \\ &= \frac{c\lg n}{\lg\lg n}(\lg c + \lg\lg n - \lg\lg\lg n - \lg e) \\ &\quad \text{because } k_0 = \frac{c\lg n}{\lg\lg n} \\ 3 &< \frac{c}{\lg\lg n}(\lg c + \lg\lg n - \lg\lg\lg n - \lg e) \\ &= c(1 + \frac{\lg c - \lg e}{\lg\lg n} - \frac{\lg\lg\lg n}{\lg\lg n}) \end{aligned}$$

Let x be the last expression in parentheses: $x = (1 + \frac{\lg c - \lg e}{\lg\lg n} - \frac{\lg\lg\lg n}{\lg\lg n})$. We need to show that there is a $c > 1$ such that $3 < cx$. Note that $\lim_{n\to\infty} x = 1$, so there exists n_0 such that $n \geq n_0 \to x \geq \frac{1}{2}$. Thus, $c > 6$ works for $n \geq n_0$.

But what about smaller values of n? In particular, $3 \leq n < n_0$?

Since n is an integer, there are a finite number of such n's, so we can plug each such n into the expression for x in $3 < cx$, derive a lower bound on c ($c >$ something), and take c big enough to satisfy all those bounds as well

as the $c > 6$ bound determined above. Then $Q_{k_0} < \frac{1}{n^3}$, as desired. By part (b), $P_k \leq nQ_k$, so $P_{k_0} < \frac{1}{n^2}$. In fact, we will show that we can pick c such that $Q_k < \frac{1}{n^3}$ for all $k \geq k_0$, and thus conclude that $P_k < \frac{1}{n^2}$ for all $k \geq k_0$. Just pick c big enough so that $k_0 > 3 > e$. Then for any $k \geq k_0$, $\frac{e}{k} < 1$ and $\frac{e^k}{k^k}$ decreases as k increases, thus $Q_k < \frac{e^k}{k^k} \leq \frac{e^{k_0}}{k^{k_0}} < \frac{1}{n^3}$ for $k \geq k_0$.

(e) The expectation of M is

$$\begin{aligned}
\mathrm{E}[M] &= \sum_{i=0}^{n} i\Pr\{M = i\} \\
&= \sum_{i=0}^{k_0} i\Pr\{M = i\} + \sum_{i=k_0+1}^{n} i\Pr\{M = i\} \\
&\leq k_0 \sum_{i=0}^{k_0} \Pr\{M = i\} + n \sum_{i=k_0+1}^{n} \Pr\{M = i\} \\
&= k_0\Pr\{M \leq k_0\} + n\Pr\{M > k_0\}
\end{aligned}$$

which is what we were asked to show, since $k_0 = \frac{c \lg n}{\lg \lg n}$.

To show that $\mathrm{E}[M] = O(\frac{\lg n}{\lg \lg n}) = O(k_0)$, note that $\Pr\{M \leq k_0\} \leq 1$ and

$$\begin{aligned}
\Pr\{M > k_0\} &= \sum_{i=k_0+1}^{n} \Pr\{M = i\} \\
&= \sum_{i=k_0+1}^{n} P_i \\
&< \sum_{i=k_0+1}^{n} \frac{1}{n^2} \text{ by part (d)} \\
&\leq n\frac{1}{n^2} \\
&= \frac{1}{n}
\end{aligned}$$

Thus, $\mathrm{E}[M] \leq k_0 + 1 = O(k_0) = O(\frac{\lg n}{\lg \lg n})$.