Problem Set 3 Solutions

This problem set is due at 11:59pm on Monday, March 11, 2013.

Problem 3-1. Lazy Hashing

Lazy Luke only attended enough 6.046 classes to learn the basics of hash functions, but didn’t learn how to handle collisions. Unfortunately, his new job requires him to implement a real hash table (of size \( m \)). Conveniently he is also guaranteed all keys to be inserted will be unique.

Luke informed his employer that he doesn’t intend to do any messy analysis, and rather than firing him, they generously agreed to grant him access to their private collection of uniform hash functions \( F \). The set \( F \) has the property that, for all \( a, b \) and \( x_1, x_2, \ldots, x_a \), if \( f_1, f_2, \ldots, f_b \) are sampled uniformly at random from \( F \), then the random variables \( f_i(x_j) \) are i.i.d. and uniformly distributed in \( \{0, 1, \ldots, m-1\} \).

(a) Luke spends a few minutes during his lunch break coming up with an idea, and decides to just put all collisions into a separate linked list. Specifically, he samples one function \( f \) from \( F \). He has one array of size \( m \) called \( \text{main} \) and one linked list (initially empty) called \( \text{extra} \). He then implements the following methods:

- **INSERT\( (x) \):** If \( \text{main}[f(x)] \) is empty, set \( \text{main}[f(x)] = x \). Otherwise, append \( x \) to \( \text{extra} \).
- **CONTAINS\( (x) \):** If \( \text{main}[f(x)] = x \), return true. Otherwise, return \( \text{extra}.\text{CONTAINS}(x) \).

Observe that \( \text{INSERT} \) runs in time \( O(1) \) always, but that \( \text{CONTAINS} \) may have to scan an entire linked list. Come up with a worst-case upper bound (worst case with respect to the inserts and the query) on the expected running time (expectation over the random choice of \( f \)) of \( \text{CONTAINS} \) after \( n \leq m \) distinct inserts have been made.

Solution: The worst-case runtime of \( \text{CONTAINS} \) after \( n \) inserts is exactly the number of collisions, so we just need to bound the expected number of collisions. Let \( X_i \) denote the random variable that is 1 if the \( i^{th} \) insert causes a collision, and 0 otherwise. Then \( X = \sum_i X_i \) is the number of collisions. At the time of the \( i^{th} \) insert, there are at most \( i-1 \) filled spots in \( \text{main} \), so the probability that the \( i^{th} \) element causes a collision (even for a worst-case sequence of inserts) is at most \( \frac{i-1}{m} \), and therefore \( \mathbb{E}[X_i] \leq \frac{i-1}{m} \).

Summing over all \( i \), we get:

\[
\mathbb{E}[X] = \sum_{i=1}^{n} \mathbb{E}[X_i] \leq \sum_{i=1}^{n} \frac{i - 1}{m} = \frac{n(n-1)}{2m}
\]
Show that, if \( n \geq 50\sqrt{m \ln m} \), then the same bound holds with probability at least \( 1 - 1/m^{100} \) (with respect to the random choice of \( f \)). Specifically, if \( g(n) \) is an upper bound from part (a) on the number of elements in \( extra \), show that, for any sequence of \( n \) distinct \( INSERT \)s followed by a \( CONTAINS \) query, with probability at least \( 1 - 1/m^{100} \) (over the choice of \( f \)) the actual length of \( extra \) during \( CONTAINS \) is at most \( \frac{3}{2}g(n) \).

**Hint:** Try to apply a Chernoff bound. The random variables you used in part (a) might not be independent, so you first have to relate them to random variables that are. Don’t be intimidated by the \( 50\sqrt{m \ln m} \), it will pop out at the end.

**Solution:** We can’t apply a Chernoff bound directly to \( \sum_i X_i \), because they aren’t independent (for instance, if \( X_j = 1 \) for all \( 1 < j < i \), then every element after the first had a collision, so only one spot in the table is taken and \( X_i \) is much more likely to be 0). So let’s first define \( X'_i \) in the following way: let \( Y_i \) be the number of collisions after inserting the first \( i - 1 \) elements (i.e. \( Y_i = \sum_{j<i} X_j \)). Pick any arbitrary \( Y_i \) empty slots and mark them as “unavailable.” Then define \( X'_i \) to be 1 if there is a collision on the \( i^{th} \) insert, or if the \( i^{th} \) insert gets mapped to an “ unavailable” slot. Now, we want to argue that it is okay to use a Chernoff bound on \( \sum_i X'_i \) instead.

First, it is easy to see that all the \( X'_i \) are independent: there are always exactly \( i - 1 \) slots that contain an element or are unavailable, so the probability that \( X'_i \) is 1 is exactly \( \frac{i-1}{m} \), no matter what else happens. Second, it is easy to see that \( X_i \leq X'_i \) always. This is because whenever \( X_i = 1 \), there is a collision on the \( i^{th} \) insert, and therefore \( X'_i = 1 \) as well. The plan now is to upper bound \( X' = \sum_i X'_i \) with high probability, which we can do using a Chernoff bound. Because \( X \leq X' \) always, this will immediately give an upper bound on \( X \) as well.

So because \( X' \) is the sum of \( n \) independent \( \{0, 1\} \) random variables, we can apply the Chernoff bound to show that with high probability, \( X' \) is close to \( \mathbb{E}[X'] \). By the work in part (a), we know that \( \mathbb{E}[X'] = \frac{n(n-1)}{2m} \), so plugging into Chernoff’s bound we get:

\[
Pr[X' > (3/2)\frac{n(n-1)}{2m}] \leq e^{-\frac{n(n-1)}{24m}}
\]

So now we just need to find conditions for which \( e^{-\frac{n(n-1)}{24m}} \leq 1/m^{100} \). Without solving exactly, this holds whenever \( n \geq 50\sqrt{m \ln m} \). Putting everything together, we have just shown that as long as \( n \geq 50\sqrt{m \ln m} \), then with probability at least \( 1 - 1/m^{100} \), \( X' \leq \frac{3n(n-1)}{4m} \). Because \( X \leq X' \) always, we have also shown that with probability at least \( 1 - 1/m^{100} \), the number of collisions is at most \( \frac{3n(n-1)}{4m} \), as desired.

Unhappy with the performance of Luke’s first solution, his employers, rather than fire him, decide to give him more space, allowing him to store \( m\ell \) elements instead of just \( m \). During his second lunch break of the day, Luke decides to use the extra space for separate hash tables. Specifically, his new idea is the following. Sample \( \ell \) functions
$f_1, f_2, \ldots, f_\ell$ from $\mathcal{F}$. Store $\ell$ arrays of size $m$ called $\text{main}_1, \text{main}_2, \ldots, \text{main}_\ell$ and one linked list called $\text{extra}$. He then implements the following methods:

- **INSERT($x$)**: Set $i = 1$. If $\text{main}_i[f_i(x)]$ is empty, set $\text{main}_i[f_i(x)] = x$. Otherwise, if $i < \ell$, set $i = i + 1$ and repeat. If $i = \ell$, add $x$ to $\text{extra}$.
- **CONTAINS($x$)**: Set $i = 1$. If $\text{main}_i[f_i(x)] = x$, return true. Otherwise, if $i < \ell$, set $i = i + 1$ and repeat. If $i = \ell$ return $\text{extra.\text{CONTAINS($x$)}}$.

Observe that INSERT now takes time $O(\ell)$, but that CONTAINS may still have to scan an entire linked list. Show that, when $\ell \geq \log \log m + 1$, for any sequence of $n \leq m$ distinct INSERTS followed by a CONTAINS query, with probability at least $1 - 1/m^{99}$, the new actual runtime of CONTAINS is $\ell + O(\sqrt{m \log m})$.

**Solution:** First, it is clear the number of expected collisions only increases as we make more inserts, so it suffices to consider only the case where $n = m$. What we want to do now is bound (with high probability) the number of elements that are ever inserted into $\text{main}_2$. Elements are inserted into $\text{main}_2$ only when they cause a collision in $\text{main}_1$. So we can use our work from part (b): we know that with probability at least $1 - 1/m^{100}$, the number of collisions in $\text{main}_1$ does not exceed $3n(n-1)/4m$.

Iterating this thought process out, we can define $Y_i$ to be the number of elements inserted into $\text{main}_i$. Then we know that $Y_1 = m$, and that $Y_{i+1}$ is also the number of collisions that occur in $\text{main}_i$. By our work in part (b), we know that as long as $Y_i \geq 50\sqrt{m \ln m}$, that with probability at least $1 - 1/m^{100}$ the number of collisions in $\text{main}_i$ is at most $\frac{3Y_i(Y_i-1)}{4m}$. For cleanliness, we will replace $Y_i - 1$ with $Y_i$ (this is a valid replacement as it only increases the upper bound). We can now take a union bound as $i$ ranges from 1 to $m$ to get that with probability at least $1 - 1/m^{99}$, for all $i \leq m$ such that $Y_i \geq 50\sqrt{m \ln m}$, the number of collisions in $\text{main}_i$ is at most $\frac{3Y_i^2}{4m}$.

So now, with probability at least $1 - 1/m^{99}$, the $Y_i$s satisfy the following recurrence (until $Y_i < 50\sqrt{m \ln m}$):

$$Y_{i+1} \leq \frac{3Y_i^2}{4m}$$

With this recursion in hand, we may now complete the proof by showing that $Y_{\log \log m + 2} = O(\sqrt{m \log m})$. Assume for contradiction that we have $Y_{\log \log m + 2} = \omega(\sqrt{m \log m})$. Then in particular, we have $Y_{\log \log m + 2} \geq 50\sqrt{m \ln m}$. This then implies that the above recurrence must hold for $Y_1$ through $Y_{\log \log m + 2}$. If the recurrence holds for $Y_1 = m$ through $Y_i$, then chasing out the recursion implies that $Y_i \leq (\frac{3}{4})^{2^{i-1}-1} m$. Substituting for $i = \log \log m + 2$ yields that $Y_{\log \log m + 2} \leq \frac{4}{3}m^{1-2\log_2(4/3)} = O(\sqrt{m \log m})$. So we must have $Y_{\log \log m + 2} = O(\sqrt{m \log m})$. Because it is clear that $Y_{i+1} \leq Y_i$ (there cannot be more collisions in $\text{main}_i$ than there were inserts), this implies that the number of collisions is also $O(\sqrt{m \log m})$.

To summarize, we first argued that with probability at least $1 - 1/m^{99}$, all large values of $Y_i$ satisfy a very strong decreasing recurrence. We then argued that as long as
this recurrence is satisfied, we have \( Y_{\log \log m + 2} = O(\sqrt{m \log m}) \). The only way the
recursion fails to hold is if some earlier value of \( Y_i \) satisfies \( Y_i \leq 50\sqrt{m \ln m} \). In both
cases, because \( Y_{i+1} \leq Y_i \), we see that the number of collisions is \( O(\sqrt{m \log m}) \).

(d) Show that, for \( \ell \geq 201, n = O(\sqrt{m \log m}) \), and any sequence of \( n \) distinct INSERTS
followed by a CONTAINS query, with probability at least \( 1 - O(1/m^{99}) \), the new actual
runtime of CONTAINS is \( \ell + O(1) \).

Hint: Don’t use a Chernoff bound for this part!

Solution: Let \( X_{ij} \) denote the random variable that is 1 if the \( i \)th element would cause
a collision if it were inserted into main\(_j\) (regardless of whether or not the insert is
actually attempted), and 0 otherwise. Define again \( X'_{ij} \) in the same way as part (b),
marking arbitrary spaces as unavailable so that the probability that \( X'_{ij} = 1 \) does not
depend on where the first \( i - 1 \) elements were inserted. Then each \( X'_{ij} \) is 1 with
probability \( O(\sqrt{\frac{\log m}{m}}) \) and 0 otherwise, and all \( X'_{ij} \) are independent. Let also \( X_i \)
denote the random variable that is 1 if element \( i \) is inserted to extra, and 0 otherwise. We
first claim that \( X_i = 1 \) if and only if \( X_{ij} = 1 \) for all \( j \). This is because we
cannot insert \( i \) into extra without getting a collision in each main. Therefore, we
see that \( X_i = \prod_j X_{ij} \). We can now define \( X'_{i} = \prod_j X'_{ij} \) and observe that \( X_i \leq X'_{i} \) always. The probability that \( X'_{i} = 1 \) is the probability that \( X'_{ij} = 1 \) for all \( j \),
which is \( O\left(\left(\frac{\log m}{m}\right)^{\ell/2}\right) \) because all \( X'_{ij} \) are independent. Because \( X_i \leq X'_{i} \) always,
we immediately get that the probability that element \( i \) is inserted into extra is also
\( O\left(\left(\frac{\log m}{m}\right)^{\ell/2}\right) \). Taking a union bound over all \( O(\sqrt{m \log m}) \) elements, we get that the
probability that any element is inserted to extra is \( O\left(\left(\frac{\log m}{m}\right)^{\ell/2-1}\right) \). When \( \ell \geq 201 \),
this probability is \( O(1/m^{99}) \), proving the desired claim.

(e) Show that, when \( \ell = \log \log m + 202 \), for any sequence of \( n \leq m \) distinct INSERTS
followed by a CONTAINS query, with probability at least \( 1 - O(1/m^{98}) \), the new actual
runtime of CONTAINS is \( \ell + O(1) \).

Solution: Part (c) guarantees that with probability at least \( 1 - 1/m^{99} \), only \( O(\sqrt{m \log m}) \)
elements are inserted beyond the \((\log \log m + 1)\)th array. Part (d) guarantees that
with probability at least \( 1 - O(1/m^{99}) \), if \( O(\sqrt{m \log m}) \) elements are inserted to the
\((\log \log m + 2)\)th array, that there will be no collisions in the last array (and therefore
extra will be empty). Therefore, taking a union bound guarantees that with probability
at least \( 1 - O(1/m^{98}) \) the new expected runtime of CONTAINS is \( \ell + O(1) \).

Problem 3-2. Shuffling Cards
In this problem, we will use the theory of Markov chains to prove that a card shuffling algorithm achieves its goal of returning a uniformly random arrangement of the cards.

Consider the following algorithm for shuffling a deck of $k$ cards:

**SHUFFLE-DECK**($deck$, $n$)
1. for $i ← 1$ to $n$
2. insert the top card of deck in a uniformly random position in the deck
3. return $deck$

Note that $k$ is the number of cards in the deck, while $n$ is the number of iterations of the shuffling loop. At every iteration of the loop, there are exactly $k$ possible positions in which the top card can be inserted.

For parts (a) and (b), assume that $k = 3$.

**(a)** Model the evolution of the entire configuration of the deck as a Markov chain. Draw a weighted directed graph whose six nodes represent states, whose edges represent transitions, and whose edge weights represent probabilities. Give the transition matrix that correctly describes the shuffling process.

**Solution:** Since $k = 3$ (and call the cards A, B, C), there are $3! = 6$ possible arrangements of the cards. Then our state can be the current arrangement of the cards.

Let us now define the transition matrix. We will order the arrangements alphabetically (that is, row and column 1 regard the state ABC, and row and column 2 regard the state ACB, · · ·, row and column 6 regard the state CBA). Then the transition matrix $P$ is as follows:

$$
\begin{pmatrix}
\frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} & 0 & 0 \\
0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} & \frac{1}{3} \\
\frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\
\frac{1}{3} & \frac{1}{3} & 0 & 0 & \frac{1}{3} & 0 \\
0 & 0 & \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3}
\end{pmatrix}
$$

**(b)** Find a stationary distribution, and show that it is the unique stationary distribution of this Markov chain.

**Hint:** Notice something special about the column sums of your transition matrix in part (a).

**Solution:** To show that there is a unique stationary distribution, that $P$ satisfies two sufficient conditions:
Problem Set 3 Solutions

- $P$ is aperiodic: We see that because each state has a self-transition.
- $P$ is irreducible: Let $x$ and $y$ be any two states, i.e. two arrangements of cards. Then $P^3[x][y] > 0$, since there is clearly a non-zero (however small) probability that the arrangement $x$ will have its 3 cards transformed into those of $y$ in 3 steps.

Also, since $P$ is a doubly-stochastic matrix (that is, its columns also add up to one) it has a left-eigenvalue of 1 corresponding to the uniform distribution (that is, the one that assigns $1/6$ to each of the six arrangements) is one of its stationary distributions. Since we have shown there is a unique stationary distribution, the uniform distribution over arrangements is indeed the unique stationary distribution.

For part (c), do not assume that $k = 3$.

(c) Conclude that the algorithm achieves its goal: argue that, as $n \to \infty$, SHUFFLE-DECK($deck, n$) produces a uniformly random permutation of the cards in $deck$.

Solution: In the general case, all one must do is note that the column sums are still 1. We can argue that by by noting that there are $k$ possible arrangements that transition into any given state, and each of those transitions happens with probability $1/k$.

Hence the argument from part (b) follows as is, and we again conclude that the uniform distribution over arrangements is the unique stationary distribution.