Colorado CSCI 5454: Algorithms
Homework 7

Instructor: Bo Waggoner
Due: October 31, 2019 at 11:59pm
Turn in electronically via Gradescope.

Remember to list the people you worked with and any outside sources used (if none, write "none").

Background: You are implementing a hash table for your company and are concerned that an attacker will try to submit a sequence of inputs, carefully constructed, so as to destroy your hash table's preformance. The first problem will develop some useful probabilistic tools, then the next problem will consider whether the attacker can succeed.

Problem 1 (6 points)

First, we need to analyze an important probabilistic process. Suppose we have n balls we are throwing into n bins. Each ball lands in each bin independently and uniformly at random, i.e. with probability $\frac{1}{n}$.

The goal of this problem is to understand the behavior of this process. The next problem will relate it back to hashing.

Part a (1 point) Fix a bin i. What is the probability that i contains exactly k balls?

Hint: first attempt the problem, then if you get stuck, look up the Binomial distribution.

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

Part b (1 point) Let Y_i be the number of balls in bin i. What is $\mathbb{E}[Y_i]$?

Hint: Let $X_{ij} = 1$ if ball j lands in bin i, zero otherwise. Write Y_i in terms of the X_{ij} variables.

Solution. $Y_i = \sum_{j=1}^{n} X_{ij}$. We have $\mathbb{E}[X_{ij}] = \frac{1}{n}$ for each j, so $\mathbb{E}[Y_i] = n\frac{1}{n} = 1$.

Part c (2 points) One version of the *Chernoff bound* says[1] that if Y is distributed Binomial(n, p) and $t \geq 6\,\mathbb{E}[Y]$, then $\Pr[Y \geq t] \leq 2^{-t}$. In other words, the probability of a large number of heads is going to zero exponentially fast.

Use this Chernoff bound to prove that for each bin i, for any $\delta \in (0, 1)$ and any n, as long as $\frac{n}{\delta} \geq 2^6$, we have $\Pr[Y_i \geq \log_2 \frac{n}{\delta}] \leq \frac{\delta}{n}$.

[1] Mitzenmacher and Upfal, *Probability and Computing*.

Solution. Y_i is distributed Binomial$(n, \frac{1}{n})$ and $\mathbb{E}[Y_i] = 1$. Let $t = \log_2 \frac{n}{\delta}$. If $\frac{n}{\delta} \geq 2^6$, then $t \geq 6 = 6\,\mathbb{E}[Y_i]$ by taking the log-base-2 of both sides. So by the Chernoff bound, $\Pr[Y_i \geq t] \leq 2^{-t} = \frac{\delta}{n}$.

Part d (2 points) The *union bound* says that if we have any different events $E_1, \ldots, E_j$, then $\Pr[E_1 \text{ or } E_2 \cdots \text{ or } E_j] \leq \sum_{j'=1}^{j} \Pr[E_{j'}]$. For example, the probability that "bin 1 is empty or bin 2 is empty" is at most the probability that "bin 1 is empty" plus the probability that "bin 2 is empty".

Use the union bound and previous answers to prove that the probability that "there exists a bin Y with at least $\log_2(\frac{n}{\delta})$ balls" is at most δ.[2]

Solution. Let $t = \log_2 \frac{n}{\delta}$. Let E_i be the event that $Y_i \geq t$. We showed that $\Pr[E_1] \leq \frac{\delta}{n}$. The probability that there exists a bin with at least t balls is $\Pr[E_1 \text{ or } \cdots \text{ or } E_n] \leq \sum_{i=1}^{n} \Pr[E_i] \leq n\frac{\delta}{n} = \delta$.

Problem 2 (12 points)

You are constructing a hash table with n slots, but an adversary is trying to hurt your performance by constructing the right sequence of inputs.

Part a (4 points) The universe of elements being hashed has size $M \geq n^2$. Prove that, if the adversary knows your hash function, she can choose a set of n inputs such that all of them hash to the same value. *Hint: start by recalling the Pigeonhole Principle.*

Solution. Suppose we hash every element of M into the n buckets. Then some bucket must have at least n elements in it. Proof: Otherwise, every bucket has at most $n-1$, and the total number of elements is at most $(n-1)(n) < n^2$, which is a contradiction.

Suppose that bucket is i and the n elements are $x_1, \ldots, x_n$. If the adversary chooses the inputs $x_1, \ldots, x_n$, then they will all hash to bucket i.

Part b (4 points) Now suppose you select a perfect hash function, i.e. each element of the universe is hashed to a value chosen uniformly and independently at random from $\{1, \ldots, n\}$. The adversary does not know the hash function, i.e. her choice of n elements is independent from the hash function choices.

Using your answer to previous problems, prove that for any $\delta > 0$ and n such that $\frac{n}{\delta} \geq 2^6$, with probability at least $1 - \delta$, at most $\log_2(\frac{n}{\delta})$ chosen elements are hashed to the same value.

[2]Note this is loose: more advanced techniques can prove an even smaller bound on the max number of balls in any bin.

Solution. We can fix the n elements chosen by the adversary and prove that for each such choice, the probability is at least $1 - \delta$. Let $t = \log_2 \frac{n}{\delta}$. Let E by the event that there exists a bin with at least t of the elements hashed to it. Because the hash function's value is chosen independently and uniformly for all elements, we can apply the solution to question 1d.

Question 1d implies that $\Pr[E] \leq \delta$. Then $\Pr[\neg E] \geq 1 - \delta$, where $\neg E$ is the event that E does *not* occur, i.e. for all bins at most t elements are hashed to that bin.

Part c (4 points) Your project manager wants to employ a hash table to store one million items in an array of one million buckets. But she wants confidence in performance of the table in terms of number of items hashing to the same location. For each $\delta = 2^{-1}, 2^{-2}, 2^{-3}, \cdots, 2^{-50}$, compute a k such that, with probability at least $1 - \delta$, no more than k elements will hash to the same location. Give your answer in the form of a plot where δ is on the horizontal axis in log scale and k is on the vertical axis. Attach the code used to generate the plot.

Problem 3 (6 points)

For each task below, state if you'd prefer to use a hash table, Bloom filter, or count-min sketch. Explain why (briefly).

Part a (2 points) You'd like to note the IP addresses that visit your website, but it gets lots of hits from one-time visitors. You only need to record the frequent IP addresses, so your data structure should check, when each visitor arrives, whether they've visited the site before.

Solution. The Bloom filter is the best choice for this because it allows us to check whether an IP address has already arrived, but it doesn't use extra space to count numbers of arrivals or store the IP addresses themselves.

Part b (2 points) When an IP address visits your website, you'd like to know how about many times it has visited. So your data structure should check, when each visitor arrives, how many times they've visited before.

Solution. The count-min sketch is the best choice, because we need to maintain counts of how many times we've seen an IP address, but we don't need to save all the IP addresses themselves.

Part c (2 points) At the end of the day, you'd like a list of all IP addresses that visited your website along with the number of times they've visited.

Solution. We can't use the Bloom filter or count-min sketch for this directly because they don't store the actual IP addresses. So the hash table is the best choice.

Problem 4 (Bonus: 1 point)

(Amortized analysis.)

Consider a hash table that starts with $n = 2$ buckets. It receives a sequence of m insertions.

Each time the number of elements in the table reaches the number of buckets, n, we double n, create a new array of size n, switch to a hash function with range $\{1, \ldots, n\}$, and insert all the existing $n/2$ elements into the new array using the hash function. Note insertions take $O(1)$ time (even if buckets have nontrivial load; we can add to the beginning of the linked lists).

Prove that the total time required is $O(m)$.

Solution. Start at $n = 2$. First, a single element is inserted and the table is half full. Then, we fill the second half of the table using $O(n)$ work, then transfer all of these to a new array using $O(n)$ more work, and double n. We repeat this until $n \geq m$. So we do big-O of $\sum_{j=1}^{\lceil \log_2(m) \rceil} 2(2^j)$ work. Note $\sum_{j=1}^{k} 2^j = 2^{k+1} - 1 \leq 2^{k+1}$, so this is $\leq 2 \cdot 2^{\lceil \log_2(m) \rceil} \leq 2 \cdot 2 \cdot 2^{\log_2(m)} = O(m)$, giving $O(m)$ total work.