CS5330: Optional problems on Hashing

March 30, 2020

I strongly encourage you to think about these problems and to discuss with me, your TA, or your peers if you need help. I will post solution hints in about 1 week.

1. Construct a 2-universal family of hash functions $\mathcal{H}$ from[1] $[n]$ to $[n]$ such that:

$$\mathop{\mathbb{E}}_{h \in \mathcal{H}} \max_{j \in [n]} |\{i \in [n] : h(i) = j\}| \geq \Omega(\sqrt{n}).$$

Extend your construction to be strongly 2-universal, not just 2-universal.

2. For any family of hash functions from a finite set U to a finite set V, show that when h is chosen at random from that family, there exist $x, y \in U$ such that:

$$\Pr[h(x) = h(y)] \geq \frac{1}{|V|} - \frac{1}{|U|}.$$

3. (Adapted from Exercise 15.11 of MU) In a multi-set, each element can appear multiple times. Suppose that we have two multi-sets, S_1 and S_2, consisting of positive integers. We want to test identity between S_1 and S_2, meaning whether each item occurs the same number of times in S_1 and S_2. One way of doing this is to sort both sets and then compare the sets in sorted order. This takes $O(n \log n)$ time if each multi-set contains n elements.

Consider the following Monte Carlo algorithm. Hash each element of S_1 into a hash table with cn counters; the counters are initially 0 and the ith counter is incremented each time the hash of an element is i. Using another table of the same size and the same hash function, do the same for S_2. If all the counters in the first table and the second table are the same, then report that the sets are the same. Otherwise, report that the sets are different.

[1] $[n]$ denotes the set $\{1, \ldots, n\}$.

Assuming that the hash function is drawn from a 2-universal family, analyze the running time and error probability of the algorithm.

4. We discussed the count-min sketch on an *insertion stream*, that is, each occurrence of an item x in the stream contributes +1 to the count of x. In *turnstile streams*, the data stream consists of pairs (i_t, c_t) where i_t is an item and c_t is an integer (possibly negative). Moreover assume that at any time T and for any item x:

$$\sum_{t: i_t = x, 1 \le t \le T} c_t > 0$$

   Explain how you could modify or otherwise use count-min filters to find heavy hitters in this situation.

5. (Thanks to Eric Price for this problem.)

   The Python programming language uses hash tables (or "dictionaries") internally in many places. Until 2012, however, the hash function was not randomized: keys that collided in one Python program would do so for every other program. To avoid denial of service attacks, Python implemented hash randomization—but there was an issue with the initial implementation. Also, in python 2, hash randomization is still not the default: one must enable it with the `-R` flag.

   Find a 64-bit machine with both Python 2.7 and Python 3.5 or later.

   (a) First, let's look at the behavior of `hash`("a") − `hash`("b") over $n = 2000$ different initializations. If `hash` were pairwise independent over the range (64-bit integers, on a 64-bit machine), how many times should we see the same value appear?

   (b) How many times do we see the same value appear, for three different instantiations of python: (I) no randomization (`python2`), (II) python 2's hash randomization (`python2 -R`), and (III) python 3's hash randomization (`python3`)?

   (c) What might be going on here? Roughly how many "different" hash functions does this suggest that each version has?

   (d) The above suggests that Python 2's hash randomization is broken, but does not yet demonstrate a practical issue. Let's show that large collision probabilities happen. Observe that the strings "8177111679642921702" and "6826764379386829346" hash to the same value in non-randomized python 2.

   Check how often those two keys hash to the same value under `python2 -R`. What fraction of runs do they collide? Run it enough times to estimate the fraction to within 20% multiplicative error, with good probability. How could an attacker use this behavior to denial of service attack a website?