Problem Set 6

CS 331H

Due Tuesday, May 3

1. For a hash family $\mathcal{H}$ from $[U]$ to $[n]$, and a set of items $S \subset [U]$, let $X_j(\mathcal{H}, S)$ be the random variable denoting the load in the jth bin:

   $$X_j := |\{i \in S \mid h(i) = j\}|$$

   as a distribution over $h \in \mathcal{H}$. Further, let $f(\mathcal{H}, S)$ denote the expected max load in any bin:

   $$f(\mathcal{H}, S) := \mathop{\mathbb{E}}_{h \in \mathcal{H}} \max_{j \in [n]} X_j$$

   (a) For any $t \geq 1$, and for any pairwise independent hash family $\mathcal{H}$ and any set S with $|S| = n$, show that

   $$\Pr[X_j \geq t] \leq O(1/t^2).$$

   Hint: bound $\mathbb{E}[X_j^2]$.

   (b) Show that for a pairwise independent family $\mathcal{H}$, that

   $$f(\mathcal{H}, S) = O(\sqrt{n})$$

   for any S with $|S| = n$.

   (c) Show that there exists a pairwise independent hash family $\mathcal{H}$ and set S with $|S| = n$ such that

   $$f(\mathcal{H}, S) = \Theta(\sqrt{n}).$$

2. Give an algorithm that takes a function $h : [U] \to [n]$ and finds pairs of inputs that collide. If h is perfectly random, your algorithm should use $O(1)$ space and $O(\sqrt{n})$ expected time. **Hint:** Relate this problem to the classic problem of finding a loop in a linked list.

3. The Python programming language uses hash tables for "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 (≥ 3.4); one is available at `linux.cs.utexas.edu`.

(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?

(e) (Optional) Using your algorithm from the previous problem, find other inputs that collide under Python's hash function.

4. Farmer John is conducting an experiment on how his cows play a variant of the repeated prisoners' dilemma. This variant of the prisoners' dilemma is a two player game, where each cow can choose whether to defect (D) or cooperate (C). The outcome of the game is as follows:

Cow A	Cow B	Result for cow A	Result for cow B
C	C	2	2
C	D	1	3
D	C	3	1
D	D	1	1

In this repeated prisoners' dilemma, cows A and B play the prisoners' dilemma T times. If A gets results $A_1, \ldots, A_T$ and B gets results $B_1, \ldots, B_T$ over the T rounds, then Farmer John gives A a total of $A_1 \times A_2 \times \cdots \times A_T$ dollars, and gives B a total of $B_1 \times B_2 \times \cdots \times B_T$ dollars.

Farmer John performs an experiment on his N cows, where each pair of cows plays a repeated prisoners' dilemma. For each of T rounds, each cow chooses to either cooperate in all her $N-1$ games, or to defect in all her $N-1$ games.

Given the transcript for each cow of whether they cooperated or defected in each round, tell Farmer John the total amount he must pay all the cows. Show how to do this in $O(T(N+2^T))$ time.