Fundamental Algorithms, Section 003
Homework 7, Fall 22. Due date: Wednesday, November 02, 4:55pm on Gradescope

1. This problem concerns using a universal hash function family to create a Bloom filter.

We will again have m hash functions, each drawn uniformly at random from a collection $\mathcal{H}$ of universal hash functions. But we will want to keep a separate array of bits for each hash function. Let $h_1, h_2, \ldots, h_m$ be the selected hash functions, and $B_1, B_2, \ldots, B_m$ be the corresponding arrays of bits. We suppose that for any pair $x \neq y$, the probability that a hash function h drawn uniformly at random from the collection $\mathcal{H}$ has $h(x) = h(y)$ is exactly $1/n$, where n is the hash table size.

Suppose there are at most s items in the set we are storing. We choose each array B_i to be of size $2s$.

a. For each item $x \notin S$ show that the probability that $B_i[h_i(x)] = 1$ is at most $\frac{1}{2}$. To prove this bound you will want to use the union bound for a collection of random events. This states that if events $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_r$ occur with probabilities $p_1, p_2, \ldots, p_r$, respectively, then one or more of these events occur with probability at most $p_1 + p_2 + \ldots + p_r$. e.g. Consider throwing a 6-sided die 3 times: the probability of getting a 4 and a 5 in some order on the first two throws is 1/18; likewise the probability of getting a 3 and a 5 in some order on the last two throws is 1/18. The probability of getting either a 4 and a 5 in some order on the first two throws, or a 3 and a 5 in some order on the last two throws, or both, is at most 1/9. (Actually, it's less than 1/9.)

b. Deduce that the probability that $x \notin S$ yet x is reported as being in S is at most $1/2^m$.

Comment. Having separate arrays for each hash function means that we don't need to worry about the collisions between the different hash functions that would occur if we used a single array as in the standard Bloom filter construction.

2. This problem concerns 2-level hash tables and builds on the analysis from problem 2 in this week's additional problems.

Suppose we have a set S of items, with each item $e = (s, v)$ having two attributes, size and value. Suppose that every pair of items is distinct, i.e. they differ on at least one of their attributes.

We are going to keep the items in a two level hash table. At the first level, we hash on the size. For each size, we will have a pointer into a second level hash table which is accessed by hashing on the value. Note that there is a separate second level hash table for each different size that occurs.

Suppose that initially S is the empty set and a total of m insertions, deletions and queries are performed. Show how to implement the two level hash table so that each query takes expected $O(1)$ time, and the m insertions and deletions require expected $O(m)$ time in total. In addition, at all times the overall table size must be $O(|S|)$.

Suppose that whenever a new array of a given size is requested that it is provided uninitialized in $O(1)$ time.

Comment: The overheads for managing the requests for arrays make this solution somewhat impractical.

3. In this problem we consider another universal hashing scheme. Suppose the key values are integers in the range $[0 : 2^j - 1]$ for some integer j, and suppose the data is being stored in a hash table of size 2^i. We define the following hash functions:

$$h_{a,b}(x) = \lfloor (ax + b \bmod 2^{i+j})/2^j \rfloor,$$

where $0 \leq a, b < 2^{i+j}$. The task is to show this family is strongly universal. More specifically, to prove that for $x \neq y$ and for any pair (c, d) with $0 \leq c, d, < 2^i$,

$$\Pr\left[h_{a,b}(x) = c \text{ and } h_{a,b}(y) = d\right] = \frac{1}{2^{2i}}.$$

Hint. To start, suppose that $x - y$ is an odd number. Now consider $h_{a,b}(x) - h_{a,b}(y) \bmod 2^i = \lfloor a(x-y) \bmod 2^{i+j}/2^j \rfloor$. The question we want to answer is for how many pairs (a, b) is this difference equal to $c - d \bmod 2^i$.
We begin by asking how many pairs (a, b) are there for which $ax + b = c' \bmod 2^{i+j}$ and $ay + b = d' \bmod 2^{i+j}$, where c' and d' are given values with $0 \leq c', d' < 2^{i+j}$.
a. First consider the condition $a(x-y) = c' - d' \bmod 2^{i+j}$. How many choices of a satisfy this condition? To answer this, we need to observe that as $x - y$ is odd it is invertible mod 2^{i+j}. This follows by showing that the values $x-y, 2(x-y), 3(x-y), \ldots, (2^{i+j)} - 1) \cdot (x-y) \bmod 2^{i+j}$ are all distinct, and therefore exactly one of these terms equals 1. Show this, and deduce what the inverse is. Now deduce how many choices of a there are.
b. Deduce how many pairs there are for which $ax + b = c' \bmod 2^{i+j}$ and $ax + b = d' \bmod 2^{i+j}$.
c. Now determine for how many pairs (c', d'), the desired conditions $h_{a,b}(x) = c$ and $h_{a,b}(y) = d$ hold.
d. Therefore, what is the probability that $h_{a,b}(x) = c$ and $h_{a,b}(y) = d$ when a and b are chosen uniformly at random?

The reasoning is similar when $x - y = 2^h$ for some integer $1 \leq h < j$. We leave the details as a challenge problem. What changes is that in step (a) we will find that there are 2^{i+j-h} distinct values, each being an integer multiple of 2^h. Each of the distinct values occurs 2^h times. So now only certain choices of c' and d' can be satisfied. Which ones?

Challenge problem. Do not submit.
Consider problem 2 on the additional problem set. Show how to do the table rebuilding so that every delete and insert operation runs in expected $O(1)$ time.