Exercise Sheet 1: Hashing and Bloom filters

Advanced Algorithms

Please feel free to discuss these problems on the unit discussion board. If you would like to have your answers marked, please either hand them in in person at the lecture or email them to me with the email subject "Problem sheet 1" by the deadline stated.

1 Weakly-universal Hashing

A hash function family $H = \{h_1, h_2, \dots\}$ is weakly-universal iff for randomly and uniformly chosen $h \in H$, we have $\Pr(h[x] = h[y]) \leq 1/m$ for any distinct $x, y \in U$. Consider the following hash function families. For each one, prove that it is weakly universal or give a counter-example.

1. Let p be a prime number and m be an integer, $p \geq m$. Consider the hash function family where you pick at random $a \in \{1, \dots, p-1\}$ and then define $h_a : \{0, \dots, p-1\} \to \{0, \dots, m-1\}$ as $h_a(x) = (ax \mod p) \mod m$.

2. Let p be a prime and m be an integer such that $p \geq m$. Consider the hash function family where you pick at random $b \in \{0, \dots, p-1\}$ and then define $h_b : \{0, \dots, p-1\} \to \{0, \dots, m-1\}$ as $h_b(x) = ((x + b) \bmod p) \bmod m$.

3. Let p be a multiple of m. Consider the hash function family where you pick at random $a \in \{1, \dots, m-1\}$ and $b \in \{0, \dots, m-1\}$. Define $h_{a,b} : \{0, \dots, p-1\} \to \{0, \dots, m-1\}$ as $h_{a,b}(x) = ((ax + b) \bmod p) \bmod m)$.

2 Cuckoo Hashing

1. This question is about cuckoo hashing. Consider a small variant of cuckoo hashing where we use two tables T_1 and T_2 of the same size and hash function h_1 and h_2. When inserting a new key x, we first try to put x at position $h_1(x)$ in T_1. If this leads to a collision, then the previously stored key y is moved to position $h_2(y)$ in T_2. If this leads to another collision, then the next key is again inserted at the appropriate position in T_1, and so on. In some cases, this procedure continues forever, i.e. the same configuration appears after some steps of moving the keys around to dissolve collisions.

   (a) Consider two tables of size 5 each and two hash functions $h_1(k) = k \bmod 5$ and $h_2(k) = \lfloor \frac{k}{5} \rfloor \bmod 5$. Insert the keys 27, 2, 32 in this order into initially empty hash tables, and show the result.

   (b) Find another key such that its insertion leads to an infinite sequence of key displacements.

2. In order to use cuckoo hashing under an unbounded number of key insertions, we cannot have a hash table of fixed size. The size of the hash table has to scale with the number

of keys inserted. Suppose that we never delete a key that has been inserted. Consider the following approach with Cuckoo hashing. When the current hash table fills up to its capacity, a new hash table of doubled size is created. All keys are then rehashed to the new table. Argue that the average time it takes to resize and rebuild the hash table, if spread out over all insertions, is constant in expectation. That is, the expected amortised cost of rebuilding is constant.

3 Bloom Filters

1. Answer the following three questions about Bloom filters:

   (a) What operations do we perform on Bloom filters?

   (b) What is the difference between hash tables and Bloom filters in terms of which data we can access?

   (c) Why is there is a problem when deleting elements from a Bloom filter?

2. Suppose you have two Bloom filters A and B (each having the same number of cells and the same hash functions) representing the two sets A and B. Let $C = A\&B$ be the Bloom filter formed by computing the bitwise Boolean *and* of A and B.

   (a) C may not always be the same as the Bloom filter that would be constructed by adding the elements of the set (A intersect B) one at a time. Explain why not.

   (b) Does C correctly represent the set (A intersect B), in the sense that it gives a positive answer for membership queries of all elements in this set? Explain why or why not.

   (c) Suppose that we want to store a set S of $n = 20$ elements, drawn from a universe of $U = 10000$ possible keys, in a Bloom filter of exactly $N = 100$ cells, and that we care only about the accuracy of the Bloom filter and not its speed. For this problem size, what is the best choice of the number of hash functions (the parameter r in the lecture)? (That is what value of r gives the smallest possible probability that a key not in S is a false positive?) What is the probability of a false positive for this choice of r?

4 Perfect Hashing

This question is about perfect hashing:

1. Our perfect hashing scheme assumed the set of keys stored in the table is static. Suppose instead that we want to add a few new items to our table after the initial construction. Suggest a way to modify our intitial construction so that we can insert these new items using no new space and without making significant changes to our existing table (in particular, we don't want to change our initial hash function). Your scheme should still do lookups of all items in $O(1)$ time, but you may use a bit more initial space.

2. Suppose now that we want to delete some of our initial items. Describe a simple way to support deletions in our perfect hashing scheme.