CS 124–Spring, 2017 Homework 5 Out: March 19, 2017
Due: March 31, 2017 (4:59 pm)

1. Suppose each person gets a random hash value from the range $[1 \ldots n]$. (For the case of birthdays, n would be 365.) Show that for some constant c_1, when there are at least $c_1\sqrt{n}$ people in a room, the probability that no two have the same hash value is at most $1/\mathrm{e}$. Similarly, show that for some constant c_2 (and sufficiently large n), when there are at most $c_2\sqrt{n}$ people in the room, the probability that no two have the same hash value is at least 1/2. Make these constants as close to optimal as possible.

   Hint: you may use the fact that
   $$e^{-x} \geq 1 - x$$
   and
   $$e^{-x-x^2} \leq 1 - x \quad \text{for } x \leq \frac{1}{2}.$$
   You may feel free to find and use better bounds.

2. A problem with Bloom filters is that you cannot delete elements from a Bloom filter by changing the corresponding 1s to 0s. (This was discussed in section.) Counting Bloom filters are a variation of Bloom filters where you do not use an array of bits, but an array of small counters. Instead of changing 0s to 1s when an element hashes to a Bloom filter location, you increment the counter. To delete an element, you decrement the counter. To check if an element is in the set, you just check if all of the counter entries are bigger than 0; if they are, then you return the element is in the set. If you see a 0, you say that the element is not in the set.

   Deletions will work properly for a Counting Bloom filter as long as the counters never overflow; once a counter overflows, you would not have an accurate count of the number of elements currently hashed to that location. A standard choice in this setting in practice to use 4 bit counters. Find an expression for the probability that a specific counter overflows for the cases of using 3 bit counters, 4 bit counters, and 5 bit counters when n elements are hashed into m total locations using k hash functions. (You may have a summation in your expression.) Calculate these probabilities when $\lceil k = (m/n) \cdot \ln 2 \rceil$, $n = 10^5$, and $m = 10^6$, reasonable numbers for some practical situations. What sized counter do you think might be appropriate to use in practice?

3. For the document similarity scheme described in class, it would be better to store fewer bytes per document. Here is one way to do this, that uses just 48 bytes per document: take an original sketch of a document, using 84 different permutations. Divide the 84 permutations into 6 groups of 14. Re-hash each group of 14 values to get 6 new 64 bit values. Call this the *super-sketch*. Note that for each of the 6 values in the super-sketch, two documents will agree on a value when they agree on all 14 of the corresponding values in the sketch. Why does it make sense to simply assume that this is the only time a match will occur?

   Consider the probability that two documents with resemblance r agree on two or more of the six sketches. Write equations that give this probability and graph the probability as a function of r. Explain and discuss your results.

   What happens if instead of using a 64 bit hash value for each group in the supersketch, we only use a 16 bit hash? An 8 bit hash?

4. In class we examined a fingerprint method for pattern matching. We considered the technique searching for a single pattern. Suppose instead we had to search for k patterns within the document. Obviously we can just do k sequential searches, but this would increase the running time by a factor of k. Can you explain how to instead do the search in k searches at the same time, in parallel? Specifically, you should only do one pass

through the document to compute fingerprints (although you could use more than one fingerprint, if desired, as we suggested in class). Explain your specific changes to the algorithm, and how it affects the probability of a false positive at each step in the algorithm (and/or the expected number of false positives overall).

5. Prove that 636127 is composite by finding an appropriate witness. Be sure to give ample evidence showing that your witness is in fact witnesses. (Note: do not use a factor as a witness! Sure, these numbers are small enough that you can exhaustively find a factor; that is not the point. A factor is not a witness, according to our definition.) Hint: you will want to write some code. You will preferably use a package that deals with big integers appropriately, as you may want to use some of this code for the next problem (RSA). We don't need a code listing for this problem– a short summary of the output should suffice.

   The number 294409 is a Carmichael number. Prove that it is composite by finding a witness. Briefly explain why Fermat's little theorem won't help.

6. My RSA public key is: (469478487497204305296287390 81,372674862636792350620645369 73). Convert the message

   Give me an A

   into a number, using ASCII in the natural way. (So for "A b": in ASCII, A = 65, space = 32, and b = 98; translating each number into 8 bits gives "A b" = 010000010010000001100010 in binary.) Encode the message as though you were sending it to me using my RSA key, and write for me the corresponding encoded message in decimal.

7. For the following problem you will write code to do some simulations. Suppose I throw one billion balls into one billion bins (independently and uniformly at random). We will say for this experiment that the maximum load is the largest number of balls in any bin. Do this 10 times, and make a table showing the frequency of the maximum load (that is, how many times the maximum load is x for any relevant value of x).

   Note: you do NOT want to create an array of one billion buckets, with the number of balls in each bucket. All buckets with the same number of balls are indistinguishable in this experiment. So suppose you keep an array A where $A[i]$ is the number of bins with i balls. Can you do your simulation just keeping track of the $A[i]$? (Trust me, this will be much easier.)

   Now modify your experiment as follows. Suppose I throw one billion balls into one billion bins, sequentially, the following way. When I throw a ball, I choose two bins independently and uniformly at random, and I put the ball in the least loaded of the two bins. (That is, I put the ball in the bin with fewer balls, and break ties whatever way you like.) Again, do this 10 times, and make a table showing the frequency of the maximum load.

   Write a few sentences describing your findings, and how you interpret them.

8. (Easy bonus question – no points will be given, do not turn it in, it's just for you to consider.) Suppose that Harvard ID numbers were issued randomly, with replacement. That is, your Harvard ID would consist of just 8 randomly generated digits, and no check was made to ensure that the same number was not issued twice. You might use the last four digits of your ID number as a password. How many people would you need to have in a room before it was more likely than not that two had the same last four digits? How many numbers could be issued before it would be more likely than not that there is a duplicate Harvard ID number? What would the answers for the above questions be if there were 12 digit ID numbers? Try to give exact numerical answers.

9. (Hard bonus question – no points will be given, do not turn it in, it's just for you to consider.) How many people do you need in the same room before it is more likely than not that some 3 people in the room share the same birthday? You may solve this problem purely mathematically (by developing an appropriate formula– be careful, this is a little tricky!) or by doing experiments by writing a program.