# CS 124 Homework 5: Spring 2021

**Your name:**

**Collaborators:**

**No. of late days used on previous psets:**
**No. of late days used after including this pset:**

Homework is due Wednesday 2021-04-07 at 11:59pm ET. You are allowed up to **twelve** (college)/**forty** (extension school) late days through the semester, but the number of late days you take on each assignment must be a nonnegative integer at most **two** (college)/**four** (extension school).

Try to make your answers as clear and concise as possible; style will count in your grades. Be sure to read and know the collaboration policy in the course syllabus. Assignments must be submitted in pdf format on Gradescope. If you do assignments by hand, you will need to scan in your results to turn them in.

For all homework problems where you are asked to design or give an algorithm, you must prove the correctness of your algorithm and prove the best upper bound that you can give for the running time. Generally better running times will get better credit; generally exponential time algorithms (unless specifically asked for) will receive no or little credit. You should always write a clear informal description of your algorithm in English. You may also write pseudocode if you feel your informal explanation requires more precision and detail, but keep in mind pseudocode does NOT substitute for an explanation. Answers that consist solely of pseudocode will receive little or not credit. Again, try to make your answers clear and concise.

1. Suppose each person gets an indendent uniformly random hash value from the range $[1 \ldots n]$. (For the case of birthdays, $n$ would be 365.)

   (a) **(10 points)** 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 $1/2$.

   (b) **(10 points)** 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. (You may wish to use the same hint.)

   (c) **(0 points, optional)**[1] Make these constants as close to optimal as possible.
   **Hints:** You may wish to use the fact that

   $$(1 - \frac{1}{n})^n < \frac{1}{e} < (1 - \frac{1}{n})^{n-1}$$

   for all $n > 1$.You may want to also use the facts 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) **(5 points)** In our description of Bloom filters in class, we could only add and look up items. Suppose we try to implement deletion of an item from a Bloom filter by changing the corresponding elements to 0s. Give an example sequence of operations (adds, deletes, and lookups) for which this Bloom filter has both a false positive and a false negative.

   (b) **(15 points)** Counting Bloom filters are a variant 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

[1] We won't use this question for grades. Try it if you're interested. It may be used for recommendations/TF hiring.

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 when using 4 bit counters when $n$ elements are hashed into $m$ total locations using $k$ hash functions. (You may have a summation in your expression.) Calculate this probability when $m/n = 8$ and $k = 5$, for $n = 1000, 10000$, and $100000$, and calculate the expected number of counters that overflow for each case.

(c) **(5 points)** Suppose that you use $m$ counters for $n$ elements and $k$ hash functions for a Counting Bloom filter. What is the false positive probability (assuming there has never been an overflow), and how does it compare with the standard Bloom filter?

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.

   (a) **(5 points)** Why does it make sense to simply assume that this is the only time a match will occur?

   (b) **(15 points)** 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.

   (c) **(10 points)** 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. Suppose we want to search for $k$ patterns, each of length $m < n$, as substrings (not subsequences: "ac" is a subsequence but not a substring of "abc") of a document of length $n$, and report which of them are in the document.

   (a) **(5 points)** Give a simple $O(nk)$ algorithm for this problem. Using as a black box the document-searching scheme from class, your algorithm shouldn't take more than a sentence or two to describe.

   (b) **(24 points)** Give an algorithm to instead do the search in $O(n + km)$ time.

5. (a) **(10 points)** Prove that 53586077 is composite by finding a witness in the form of a positive integer $a < 53586077$ such that $a^{53586077-1} \neq 1$. Give any information necessary to show that your witness in fact witnesses. (Note: do not use a factor as a witness! Sure, 53586077 is small enough that you can exhaustively find a factor, but we want to test your knowledge of the primality-testing algorithm without using awkwardly-big numbers.)

   (b) **(10 points)** The number 294409 is a Carmichael number. Prove that it is composite by finding a witness in the form of a nontrivial square root of 1.

   (You can find the answers by whatever method you like, but we recommend writing some code; if you use a language like C that doesn't natively deal with big integers nicely, you'll want some package that does. You don't need to submit code.)

6. **(0 points, optional)**[2] 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.

[2] We won't use this question for grades. Try it if you're interested. It may be used for recommendations/TF hiring.