Lab 0 Theory

This assignment is due **at 10:00pm ET** on **Thursday, September 16, 2021**.

Please make note of the following instructions:

- Remember that your solutions must be submitted on Gradescope. Please sign-up for 6.S060 Fall 2021 on Gradescope, with the entry code `KYRBPK`, using your MIT email.
- We require that the solution to the problems is submitted as a PDF file, **typeset on LaTeX**, using the template available on the course website (`https://6s060.csail.mit.edu/2021/`). Each submitted solution should start with your name, the course number, the problem number, the date, and the names of any students with whom you collaborated.
- You will often be called upon to "give an algorithm" to solve a certain problem. Your write-up should take the form of a short essay. A topic paragraph should summarize the problem you are solving and what your results are. The body of your essay should provide the following:
    1. A description of the algorithm in English and, if helpful, pseudocode.
    2. A proof (or indication) of the correctness of the algorithm.
    3. An analysis of the asymptotic running time behavior of the algorithm.
    4. Optionally, you may find it useful to include a worked example or diagram to show more precisely how your algorithm works.

Problem 0-1. Probability Practice: Dice Rolling [25 points]

If you need to review some probability concepts, you can find the probability review handout on the course website.

(a) [10 points] You are given a fair, 6-sided die. On average, how many times must the die be rolled until a 5 turns up twice in a row?

(b) [15 points] On average, how many times must the die be rolled until the sequence 56 appears (i.e., a 5 followed by a 6)?

Problem 0-2. Hashing and Collisions [75 points]

Each 6.S060 staff member and student requires access to 6.S060 material, including files private to each user. Assume that each of the user names corresponds to a distinct string with n or fewer characters.

(a) [15 points] Ben Bitdiddle is tasked with designing the system that provides users access. He designs a hash function to maps usernames to IDs by defining the hash of a string $s = (s_0, \ldots, s_{n-1})$ of length n as

$$\text{hash}(s) = s_0 + s_1 \cdot p + s_2 \cdot p^2 + \ldots + s_{n-1} \cdot p^{n-1} \mod m \;=\; \sum_{i=0}^{n-1} s_i \cdot p^i \mod m,$$

where $p \in \{1, \ldots, m-1\}$ is a constant. He uses the hash as an ID, and uses the ID to index into a direct-access array that provides access to the corresponding user's files. To keep memory requirements manageable, he sets the modulus $m = 2^{20}$. Ben figures that the total number of distinct users u that he needs to support will satisfy $u \ll m$, so he conjectures that each user will be assigned a unique hash/ID. Describe the problem with Ben's approach in one sentence.

(b) [15 points] Ben designs the system so usernames can only use lowercase characters from a to z, and have a length of 5, i.e., $n = 5$.

He maps characters to numbers, i.e., "a" $= 1$, "b" $= 2$, all the way to "z" $= 26$, and chooses $p = 31$.

Argue that there are at least two *possible* usernames with the same hash/ID in Ben's scheme.

(c) [15 points] Give an example of two different usernames with the same hash/ID. (You may want to write a simple program for this!)

(d) [30 points] Clearly, Ben's simple hash function does not suffice for his application. Unfortunately, all compressing hash functions—hash functions whose output is shorter than their input—have collisions. Alyssa P. Hacker tells Ben to look into cryptographic hash functions, which have collisions but these collisions are hard to find.

In the analysis of cryptographic constructions, we will often model a hash function $H\colon \{0,1\}^{2n} \to \{0,1\}^n$ as a "perfectly random" hash function. That is, for each input string $x \in \{0,1\}^{2n}$, we think of the hash value $H(x) \in \{0,1\}^n$ as being a string chosen independently and uniformly at random from the set of n-bit strings.

Now, given such a hash function H:

- Approximate the probability—over the random choice of the hash outputs—that after querying H in q different locations there are no collisions in the resulting q hash values.

- For which values of $q \gg 1$ do we expect to see a collision with a constant positive probability (i.e., with probability that is independent of n)? You can provide an Θ answer for q.

Hint: Use the approximation of $e^{-x} \approx 1 - x$ for small $x > 0$ (which follows from the Taylor series expansion).