HW 7 CMSC 456. Morally DUE Nov 9

This homework is a little different.

There are FIVE programs here. You will upload your ANSWERS to be autograded and your programs to be manually looked at. We are only looking at your programs to make sure you actually wrote them, there are no secret tests. Thus, this homework should be a free 100.

The pdf you upload will mostly be five code snippets (you can don't have to include the whole program, screenshots of relevant function code is fine), plus a couple of short questions here and there.

You will upload to the autograder a txt file named "answers.txt". It will contain five lines, one for each problem. Each line will contain only integers separated by spaces. You will know what integers to include for each problem because it will say "IN YOUR ANSWER FILE:" in ALL CAPS.

You can use any non-esoteric language, because the autograder isn't running your program. As long as it runs on your machine, and it's readable, you are OK.

1. (a) Write a program that will, on input a, n, p, output $a^n \pmod{p}$. Make it efficient, so use repeated squaring. Some languages have this built in, but you are not allowed to use it.

   (b) Write a version of your exponentation program that counts the number of multiplications it does. Find an formula for the number of multiplications required given a, n, p. (It will be obvious once you find it, a single sentence is sufficient.) Include this in the manually-graded pdf.

   (c) Using your program compute $a^n \pmod{100}$ for all $a \in \{1, \ldots, 99\}$ and $n \in \{1, \ldots, 99\}$.

   (d) Calculate, for each $1 \leq n \leq 99$ how many a exist such that $a^n \equiv a \pmod{100}$. Call this number $f(n)$.

   (e) Sort your list of $f(n)$ from greatest to least such that $f(n_1)$ is the largest, $f(n_2)$ is the second largest, and so on.

   IN YOUR ANSWER FILE: include the 198 integers

$$n_1, f(n_1), n_2, f(n_2), \ldots, n_{99}, f(n_{99})$$

   in that order.

2. (a) Write a dumb algorithm to test for primality: on input n test if any number $\leq \sqrt{n}$ divides it. This is the naive algorithm.

   (b) Use the algorithm on the Oct 26 slide *A Primality Testing Algorithm*. For your "random" numbers, use $R = \{2^m - 1 \mid 2 \leq m \leq \lg n\}$. This is the fast algorithm.

   (c) Run both algorithms on every odd number between 101 and 2000.

   (d) IN YOUR ANSWER FILE: List any numbers n such that the two algorithms differed (if there are any).

   (e) Some of these differences are caused of our choice of R, some of them will differ no matter what R we choose. Try to find better choices for R. Which of your answers to part (d) are still present? Write down your answer in the manually graded portion.

   (f) Run the fast algorithm on all numbers that are $\leq 1,000$, then $\leq 10,000$, then $\leq 100,000$, then $\leq 1,000,000$. IN YOUR ANSWER FILE: Tell us how many primes you found with your fast algorithm for each bound.

   (g) Asymptotically, the number of primes $\leq n$ should be $n/\ln n$. For each bound you computed in the last part, how far off is your answer from your bound? IN YOUR ANSWER FILE: the error, rounded to the nearest integer, between the number of primes you found and $n/\ln n$ for each bound.

   (h) (5 pts, but open-ended) Conjecture about the error term, i.e. suggest a function for it and give reasoning or graphs to support.

3. (a) Write an algorithm that will, given a number n, test if its a SAFE prime. You MUST use the fast but imperfect algorithm from the last problem.

   (b) Run your algorithm on all numbers that are and $\leq 1,000,000$. IN YOUR ANSWER FILE: The number of safe primes found.

   (c) What percentage of PRIMES are safe primes? IN YOUR ANSWER FILE: round((number of safe primes / # of primes) * 10000)

4. (a) Write an program that will, given a number p will TEST if it is a safe prime and if it IS then find a generator by testing 2,3,4,... until you get one. KEEP TRACK of how many tries it took. Use the naive prime algorithm for this one, because accuracy is actually important here.

   (b) Bill has a different idea! He will do part (a), but instead of using $2, 3, 4, \ldots$, he will test $\{\lfloor p/2 \rfloor, \lfloor p/3 \rfloor, \lfloor p/4 \rfloor, \ldots\}$ in that order. Code it up and keep track of how many tries this method takes.

   (c) Run both algorithms on all numbers between 100 and 10,000 inclusive.

   IN YOUR ANSWER FILE: Let a_1 be the maximum number of tries for alg 1. Let b_1 be the average number of tries per prime for alg 1, so $b_1 =$ (sum total of tries / number of primes). Let a_2, b_2 be the same for alg 2.

   Print a_1, round($b_1 * 100$), a_2, round($b_2 * 100$).

   (d) Speculate on which alg is better. (Include this part in your manually graded pdf.)

5. Write a function implementing Diffie Hellman.

   It should take in a modulus p, a base g, and an unshared secret a or b.

   It should print some public data to console, read some data from console, and come up with a shared secret.

   If I run two copies of your program side by side, and send the output from one to the input of the other and vice versa, they should come up with the same shared secret.

   (a) $p = 2221$, $g = 123$.

   Alice's secret is $a = 555$. Bob publishes $g^b = 439$.

   IN YOUR ANSWER FILE: what Alice publishes, the shared secret.

   (b) $p = 7727$, $g = 1998$

   Alice's secret is $a = 911$. Bob's secret (not his published value) is $b = 311$.

   IN YOUR ANSWER FILE: what Alice publishes, what Bob publishes, the shared secret.