HW04 CMSC/MATH/ENEE 456. Morally DUE Oct 12
SOLUTIONS

1. (0 points)

   (a) What is the day and time of the midterm?

   (b) IF you CANNOT make the timed part of the midterm let me know NOW!!!!!!!!!!

GOTO NEXT PAGE

2. (20 points, programming question) The goal of this problem is to (1) get data on what fraction of numbers are safe primes, (2) write programs that will be used for both Diffie-Helman and RSA.

   (a) (0 points) Program EXP. 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) (0 points) Program TESTPRIME. Test for primality using the following method which is a variant of what was on the slides: To test if n is prime pick 5 distinct random numbers $a_1, a_2, a_3, a_4, a_5 \in \{2, \ldots, n-2\}$ and compute, for $1 \le i \le 5$, $a_i^{n-1} \pmod{p}$. If ALL are 1 then output 1. if ANY are not 1 then output 0. (So 1 means PRIME and 0 means NOT PRIME.)

   (c) (0 points) Program TESTSAFEPRIME. Given a number n, test if its a SAFE prime. If it is then output 1, if not then output 0.

   (d) (20 points) Program HOWMANYSAFEPRIME: Given n, determine how many numbers in $\{1, \ldots, n\}$ are safe primes.

   In your main method, you should take as input n and output the resulting integer from HOWMANYSAFEPRIME(n).

   (a) n will be given as a command line argument. Expect your filename to be the first command line argument and n to be the second. There will be no input given through standard input.

   (b) You should output HOWMANYSAFEPRIME(n) to standard output.

   (c) You should upload a **single** file ending in `.java`, `.py`, `.ml`, `.rb`, `.c`, `.cpp`, or `.scala`, corresponding to Java, Python3, OCaml, Ruby, C, C++, and Scala respectively.

COMMENTS

Students had questions on the following while doing this HW:

a) The calculation of $a^n \pmod{p}$ is taking too much time. The issue was that you need to do mod p after EVERY calculation so that the numbers do not get that large.

b) I get answers that are a wee bit different than what I think we should get. Two reasons.

- The test does NOT work on some numbers. 561 is the least one that it does not work on. These are *Carmichael Numbers* which are composits n such that, for ALL $a \in \{1, \ldots, n-1\}$, $a^n \equiv 1 \pmod{n}$. Hence these numbers will be declared PRIME even though they are NOT.
- If n is a composite (not a Carmichael number) then there is SOME a such that $a^{n-1} \not\equiv 1$. You may have gotten unlucky and picked a_1, a_2, a_3, a_4, a_5 such that for all $1 \le i \le 5$, $a_i^{n-1} \equiv 1 \pmod{n}$. (PROJECT IDEA: how common is this? if you find such a composite, do most a fail?)

END OF COMMENTS

GOTO NEXT PAGE

3. (20 points, written question) You will use your programs from question 2 for the following:

   (a) (10 points) Run HOWMANYSAFEPRIME on the inputs 10000, 20000, ..., 90000. Use this to determine what proportion of numbers in $\{1, ..., 10000\}, \{1, ..., 20000\}, ..., \{1, ..., 90000\}$ are safe primes.

   Report your results.

   (b) (10 points) Based on this data make a conjecture about $f(x + 10000) - f(x)$, where f is HOWMANYSAFEPRIME.

COMMENTS

The conjecture in math is that there is a constant c such that

Number of safe primes $\le n$ is $\frac{cn}{\ln n} + O(1)$.

This conjecture would imply that:

$$f(x+10000) - f(x) = c\left(\frac{x+10000}{\ln(x+10000)} - \frac{x}{\ln x}\right) + O(1)$$

$\ln(x+10000)$ is approximately $\ln(x)$ so this simplifies to

$$f(x+10000) - f(x) = c\left(\frac{10000}{\ln x}\right) + O(1)$$

I doubt your data would have lead you to this eqution since there just isn't that much data, and c is unknown. However, your data should lead you to:

$$f(x+10000) - f(x) \text{ is a decreasting function}$$

$$\lim_{x\to\infty} f(x+10000) - f(x) = 0$$

and you may have conjectured SOME function that goes to 0.

I may, on a later HW, ask you to estimate what c is and possibly what the additive term is.

END COMMENTS

GOTO NEXT PAGE

4. (20 points, programming question) The goal of this problem is to (1) get data on what fraction of numbers are generators, (2) write programs that will be used for both Diffie-Helman and RSA.

   (a) (0 points) Program TESTGEN. Given p and g do the following

      i. Test if p is a safe prime (if NOT then output 2 and stop, so 2 means BAD INPUT because NOT a safe prime.)

      ii. Test if $g \in \{2, \ldots, p-2\}$ (if NOT then output 3 and stop, so 3 means BAD INPUT because g is not in the right range).

      iii. (If you got this far then p is a safe prime and g is a candidate for a generator.) Find $q = \frac{p-1}{2}$. Note that this will be a prime. Compute $g^2 \pmod{p}$ and $g^q \pmod{p}$. If BOTH are not 1 then g is a generator. If EITHER is 1 then g is not a generator. Output 1 if g is a generator and output 0 if g is not.

   (b) (20 points) Program HOWMANYGEN: Given p (test if p is a safe prime and if its not output "*not safe man!*") determine how many numbers in $\{2, \ldots, p-1\}$ are generators.

In your main method, you should take as input p and output the result from HOWMANYGEN(p).

   (a) p will be given as a command line argument. Expect your filename to be the first command line argument and p to be the second. There will be no input given through standard input.

   (b) You should print HOWMANYGEN(p) to standard output, which should be either an integer or "*not safe man!*"

   (c) You should upload a **single** file ending in `.java`, `.py`, `.ml`, `.rb`, `.c`, `.cpp`, or `.scala`, corresponding to Java, Python3, OCaml, Ruby, C, C++, and Scala respectively.

GOTO NEXT PAGE

5. (20 points, written question) You will use your programs from question 4 for the following:

   (a) (20 points) Run HOWMANYGEN on:

   1019, 2027, 3023, 4007, 5087, 6047, 7079, 8039, 8963, and 10007

   Use this to determine what proportion of numbers in $\{2, ..., 1019 - 1\}$ are generators of 1019, what proportion of numbers in $\{2, ..., 2027 - 1\}$ are generators of 2027, etc. for all 10 safe primes listed above.

   Report your results.

   (b) (0 points) Based on this data make a conjecture about $g(p)$, where g calculates the proportion of generators in $\{2, ..., p - 1\}$ of p.

COMMENTS

The theorem in math is that there is a constant c such that

Number of gen mod p is $\frac{cp}{\ln lnp} + O(1)$.

This would imply

$$g(p) = \frac{cp}{\ln \ln p} + O(1).$$

I doubt your data would have lead you to this eqution since there just isn't that much data, and c is unknown. However, your data should lead you to:

$$g(p) \text{ is an increasing function}$$

and you may have conjectured SOME function that goes to infinity, perhaps a linear function with factor less than 1.

I may, on a later HW, ask you to estimate what c is.

END COMMENTS

GOTO NEXT PAGE

6. (20 points, programming question) The goal of this problem is to code up Diffie Helman.

   (a) FINDSAFEPRIME. On input L output a safe prime that is between 2^L and $2^{L+1}-1$ (so its L bits long) by doing the following: pick a random number r between 2^L and $2^{L+1}-1$ and test if its a safe prime. If so GREAT. If not then try $r+1, r+2, \ldots$ until you get one. (IF it ends up being over $2^{L+1}-1$, thats fine.)

   (b) FINDGEN. Given p a safe prime (if its not a safe prime output an appropriate insult) find a generator for p by testing random numbers in $\{2, ..., p-1\}$ until you get one. (NOTE- in the real world you would not do it this way, but on the HW we do it this way so its easier for you to do and for us to grade.)

   (c) SETUPDH. On input L, output a safe prime p and a generator for it g.

   (d) DHAlicesends. On input (p, g) pick a random $a \in \{2, \ldots, p-2\}$ and output g^a.

   (e) DHBobsends. On input (p, g) pick a random $b \in \{2, \ldots, p-2\}$ and output g^b.

   (f) DHAlicegetskey. On input (p, g, a, x) compute $x^a \pmod{p}$. Note that if $x = g^b$ then this will be g^{ab}.

   (g) DHBobgetskey. On input (p, g, b, x) computes $x^b \pmod{p}$. Note that if $x = g^a$ then this will be g^{ab}.

   GOTO NEXT PAGE FOR SUBMISSION DETAILS

In your main method, you should take as input L and output many different values computed throughout this problem.

(a) L will be given as a command line argument. Expect your filename to be the first command line argument and L to be the second. There will be no input given through standard input.

(b) You should output many different variables to standard output on separate lines as follows:

   i. On the first line, print your safe prime, p, from SETUPDH.
   ii. On the second line, print your generator, g, from SETUPDH.
   iii. On the third line, print your random value for a from DHAliceSends.
   iv. On the fourth line, print your computed value for g^a from DHAliceSends.
   v. On the fifth line, print your random value for b from DHBobSends.
   vi. On the sixth line, print your computed value for g^b from DHBobSends.
   vii. On the seventh line, print your computed g^{ab} from DHAliceGetsKey.
   viii. On the eighth line, print your computed g^{ab} from DHBobGetsKey. Note: this should match what is printed on the previous line. As confirmation that your programs work properly, it is highly recommended that you compute this value independently.

   A sample output file will be provided.

(c) You should upload a **single** file ending in `.java`, `.py`, `.ml`, `.rb`, `.c`, `.cpp`, or `.scala`, corresponding to Java, Python3, OCaml, Ruby, C, C++, and Scala respectively.