

Department of Mathematics and Statistics
620-351: Number Theory 2008
Assignment 1

Answers to this assignment should be returned by
11.00am on Wednesday 27 August.

Questions 1 and 2 require calculations. You are free to use calculators or computers to assist with these calculations but you should present your answer as though the working had been done by hand. Thus all the main details of the calculation should be visible.

1. Find a solution in *positive* integers x and y to the equation

$$351x + 620y = 300,000 + sn$$

where sn is (the number formed from) the last five digits of your student number.

2. Find all solutions of the congruence

$$x^2 - 2x + 160 \equiv 0 \pmod{175}.$$

3. There are two options for this question. Both look at the number of operations required to complete a Euclidean algorithm. The following is relevant to both options.

A) We think of a general Euclidean algorithm in the following form.

$$\begin{aligned} a &= q_0b + r_1 \\ b &= q_1r_1 + r_2 \\ &\vdots \\ r_{i-1} &= q_i r_i + r_{i+1} \\ &\vdots \\ r_{k-1} &= q_k r_k + 0 \end{aligned}$$

B) The *Fibonacci* numbers are defined by $F_1 = F_2 = 1$ and, for $n > 1$, $F_{n+1} = F_n + F_{n-1}$. They can also be expressed as

$$F_n = \frac{1}{\sqrt{5}} (\alpha^n - \beta^n)$$

where α is the positive root and β the negative root of $x^2 - x - 1 = 0$. (You need not prove this).

Either:

- (a)
- i. Show that, for all i , $r_{i+1} < r_{i-1}/2$. (Hint: first show that, if $r_i > r_{i-1}/2$ then $q_i = 1$.)
 - ii. Show that $k \leq 1 + 2 \log_2 b$.
 - iii. Assume that a "division with remainder" of two numbers of n -decimal digits take no more than Kn^2 operations for some constant K . Show that the Euclidean algorithm for two n -digit numbers takes no more than Ln^3 operations for a suitable constant L .
 - iv. If $a = F_{n+1}$ and $b = F_n$, show that the Euclidean algorithm takes $n - 2$ steps. Deduce that, in this case, there is a constant M so that the Euclidean algorithm for a and b takes *at least* $M \log_2 b$ steps.

or:

- (b) The aim of this option is to determine *experimentally* the relationship between number of digits and time taken for a Euclidean algorithm. You should aim to do two major steps:
- i. Use Pari-gp to find a good approximation for the value of s so that the time taken to calculate the gcd of two random n -digit numbers is of the form An^s for some constant A .
 - ii. The same problem but for two consecutive Fibonacci numbers.

Some comments for option (b)

- You should choose the range of numbers so that the times taken to calculate the gcd are at least a few milliseconds. It is probably wise if the largest times are no more than a few minutes.
- The problem requires you to use data generated within Pari-gp to estimate a good value for s . This estimation need not use Pari-gp. It may use other software or may involve (non-computer) graph plotting. Those who know statistical techniques for this sort of estimation are free to use them.
- A useful function *not* mentioned in the Notes is **gettime()**. This measures the time since the last use of `gettime()` (or since the start of the session). It is useful for measuring the time of processes within a procedure. For example, `"gettime(); gcd(a,b); gettime()"` will measure the time taken by the gcd function only and will not include the time taken by any other parts of the process.