

MA357, Spring 2008 — Problem Set 3

This problem set focuses on linear diophantine equations, prime factorization, and a little bit of congruences. A write-up of all the problems except the ones labelled “to explore” is due on **Friday, February 29**.

1. Silverman, problem 6.2.
 2. Silverman, problem 7.2. (Can you give a proof that doesn't use unique factorization?)
 3. Silverman, problem 7.5.
 4. Are there any prime numbers p of the form $p = n^3 - 1$? If so, how many of them are there?
 5. Three merchants found a purse along the way. One of them said, “If I secure this purse, I shall become twice as rich as both of you with your money on hand.” Then the second said, “I shall become thrice as rich as both of you.” The third man said, “I shall become five times as rich as both of you.” How much did each merchant have, and how much was in the purse?
 6. Silverman, problems 8.3 and 8.5.
 7. Prove that no number of the form $4k + 3$ can be equal to a sum of two squares.
 8. Show that no square has last digit 2, 3, 7, or 8.
 9. Are there any prime numbers p of the form $p = n^3 - 1$? If so, how many of them are there?
 10. Find the remainders when the following numbers are divided by 37: (a) 2^{50} , (b) 2^{72} , (c) 3^{17773} .
 11. The classical test for divisibility by 4 goes like this: n is divisible by 4 if and only if the number formed by its last two digits is divisible by 4. Use a congruence argument to show that this is true.
(In order to do this, you have to understand decimal notation, i.e., you have to know that when we write “3213” we really mean $3 \cdot 10^3 + 2 \cdot 10^2 + 1 \cdot 10^1 + 3$.)
- To Explore:** Given a number n , how can we find its factorization? Start from Silverman, problem 7.6; this asks you to actually write a computer program that will factor (small) integers by using a slightly improved form of trial division.

Here is a different method, based on the identity $a^2 - b^2 = (a + b)(a - b)$. Let n be the number to be factored.

- a. Let $a = \lfloor \sqrt{n} \rfloor + 1$, $i = 1$, $u = a^2$.
- b. If $u - n$ is a square, then set $u - n = b^2$ and then $n = (a - b)(a + b)$, end.
- c. If not, let $u = u + 2a + 1$, $a = a + 1$, $i = i + 1$.
- d. If $i \leq M$, go back to step (b).
- e. If $i > M$, report that the algorithm has failed to factor n .

As given, the algorithm needs a parameter M that limits the number of iterations it goes through. This allows us to give up when the algorithm is taking too long to find a factorization. The underlying factoring method is usually attributed to Fermat.

Here are some questions to think about:

- a. Why does this work?
- b. How high would M have to be set in order to be sure that we can factor any number n ?
- c. Does this method improve on trial division?
- d. Could we combine this method with trial division to get something more efficient?

Try writing factoring programs using (a) the trial division method, (b) the method outlined above, and (c) various modifications or combinations of both methods. Compare their speeds. To test your program, here are some numbers to try to factor:

- a. Mersenne numbers: $M_n = 2^n - 1$, with n prime, $n \leq 57$. (One of these is very hard.)
- b. Repunits: $R_n = (10^n - 1)/9$ = the n -digit number all of whose digits are 1, for $3 \leq n \leq 25$. (Three of these are hard.)
- c. $2^n + 1$ for $1 \leq n \leq 60$. (These are easy to start, but may be hard to finish. $2^{58} + 1$ is particularly hard.)
- d. $10^{22} + 1$ (This is hard to do with one method, but teaming up trial division with Fermat's method should work.)

Note: since you will be dealing with big numbers here, it is important to make sure that your program handles them correctly.