# Discrete Mathematics
# Homework 3

Ethan Bolker

November 15, 2014

Due: ???

We're studying number theory (leading up to cryptography) for the next few weeks. You can read Unit NT in Bender and Williamson. This homework contains some programming exercises, some number theory computations and some proofs. The questions aren't arranged in any particular order. Don't just start at the beginning and do as many as you can – read them all and do the easy ones first. I may add to this homework from time to time.

## Exercises

1. Use the Euclidean algorithm to compute the greatest common divisor $d$ of 59400 and 16200 and find integers $x$ and $y$ such that

$$59400x + 16200y = d$$

I recommend that you do this by hand for the logical flow (a calculator is fine for the arithmetic) before writing the computer programs that follow. There are many websites that will do the computation for you, and even show you the steps. If you use one, tell me which one.

We wish to find $\gcd(16200, 59400)$.

**Solution**

*LaTeX note.* These are separate equations in the source file. They'd be better formatted in an `align*` environment.

$$\begin{aligned} 59400 &= 3(16200) + 10800 \quad \text{next find } \gcd(10800, 16200) \\ 16200 &= 1(10800) + 5400 \quad \text{next find } \gcd(5400, 10800) \\ 10800 &= 2(5400) + 0 \quad \gcd(16200, 59400) = 5400 \end{aligned}$$

$$\begin{aligned} 59400x + 16200y &= 5400 \\ 5400(11x + 3y) &= 5400 \\ 11x + 3y &= 1 \end{aligned}$$

By inspection, we can easily determine that $(x, y) = (-1, 4)$ (among other solutions), but let's use the algorithm.

$$\begin{aligned} 11 &= 3 \cdot 3 + 2 \\ 3 &= 2 \cdot 1 + 1 \end{aligned}$$

$$\begin{aligned} 1 &= 3 - (2) \cdot 1 \\ 1 &= 3 - (11 - 3 \cdot 3) \\ 1 &= 11(-1) + 3(4) \\ (x, y) &= (-1, 4) \end{aligned}$$

2. An interesting class of examples.

   (a) Use the Euclidean Algorithm *by hand* to find an integral solution to the equation

$$55x + 89y = 1$$

   You can probably see that 55 and 89 are mutually prime and so $\gcd(55, 89) = 1$, but let's run it through the Euclidean Algorithm.

$$\begin{aligned}
\gcd(55, 89) \quad & 89 = 1 \cdot 55 + 34 \\
\gcd(34, 55) \quad & 55 = 1 \cdot 34 + 21 \\
\gcd(21, 34) \quad & 34 = 1 \cdot 21 + 13 \\
\gcd(13, 21) \quad & 21 = 1 \cdot 13 + 8 \\
\gcd(8, 13) \quad & 13 = 1 \cdot 8 + 5 \\
\gcd(5, 8) \quad & 8 = 1 \cdot 5 + 3 \\
\gcd(3, 5) \quad & 5 = 1 \cdot 3 + 2 \\
\gcd(2, 3) \quad & 3 = 1 \cdot 2 + 1 \\
\gcd(1, 2) \quad & 2 = 1 \cdot 1 + 1 \\
\gcd(1, 1) \quad & 1 = 1 \cdot 1 + 0
\end{aligned}$$

   Now we run everything through backwards.

$$\begin{aligned}
1 &= 2 - 1 \cdot 1 \\
&2 - 1(3 - 1 \cdot 2) \\
&2 \cdot (2) - 3 \\
&2 \cdot (5 - 1 \cdot 3) - 3 \\
&2 \cdot 5 - 3 \cdot (3) \\
&2 \cdot 5 - 3(8 - 1 \cdot 5) \\
&5 \cdot (5) - 3 \cdot 8 \\
&5(13 - 1 \cdot 8) - 3 \cdot 8 \\
&5 \cdot 13 - 8 \cdot (8) \\
&5 \cdot 13 - 8(21 - 1 \cdot 13) \\
&13 \cdot (13) - 8 \cdot 21 \\
&13(34 - 1 \cdot 21) - 8 \cdot 21 \\
&13 \cdot 34 - 21 \cdot (21) \\
&13 \cdot 34 - 21(55 - 1 \cdot 34) \\
&34 \cdot (34) - 21 \cdot 55 \\
&34(89 - 1 \cdot 55) - 21 \cdot 55 \\
&34 \cdot 89 - 55 \cdot 55 = 1 \\
&(x, y) = (-55, 34)
\end{aligned}$$

(b) Your answer to the previous question should suggest a nice identity about the Fibonacci numbers. State it, then prove it by induction. (Look up "Fibonacci numbers" if you have to.)

**Solution**

What you should have *noticed* about the previous calculation is that all the quotients on the way down are just 1 – the smallest they could possibly be. You should also have *recognized* the sequence

$$89, 55, 34, 21, 13, 8, 5, 3, 2, 1$$

as the *Fibonacci numbers* – I even gave that away in the hint.

That's not an accident. The Fibonacci numbers are defined by the recursion relation

$$\begin{aligned} F_1 &= 1 \\ F_2 &= 2 \\ F_{n+1} &= F_n + F_{n-1} \end{aligned} \tag{1}$$

Since we started with two adjacent Fibonacci numbers, that defining relation implies that the quotients in the Euclidean algorithm will all be 1 and that the remainders will count down the Fibonacci numbers to 1, the greatest common divisor.

Equation 1 is *not* the "nice identity" I asked for – it's just the definition of the Fibonacci numbers. What I hoped you'd notice is that the $x$ and $y$ in the identity

$$-55 \times 55 + 34 \times 89 = 1$$

are themselves both Fibonacci numbers. That suggests

**Theorem 1.** *For each $n > 2$,*

$$F_n^2 - F_{n+1}F_{n-1} = (-1)^n \tag{2}$$

*Proof.* Suppose we knew that Equation 2 was true for a particular value of $n$. Then its truth for $n+1$ follows from the computation

$$\begin{aligned} F_{n+1}^2 - F_{n+2}F_n &= F_{n+1}(F_n + F_{n-1}) - (F_{n+1} + F_n)F_n \\ &= F_{n+1}F_{n-1} - F_nF_n \\ &= -(-1)^n = (-1)^{n+1}. \end{aligned}$$

Since

$$2^2 - 3 \times 1 = 1,$$

Equation 2 is true for $n = 1$. Then by induction it's true for all $n$. □

If you found the different solution

$$34 \times 55 - 21 \times 89 = 1$$

you'd come up with a slightly different Fibonacci number identity to prove:

$$F_{n-1}F_n - F_{n-2}F_{n+1} = \pm 1.$$

3. Logarithmic time

(a) Prove that if you carry out *two steps* in the Euclidean algorithm for $\gcd(a, b)$ with $a > b$ the remainder is less than $a/2$.

**Solution**

This argument is Jiho Choi's. It's better than the one I knew (which is the one most of you either found or invented). The algorithm starts with

$$a = bq + r. \tag{3}$$

Since $a > b$, I know $q \geq 1$ so $a \geq b + r$. I also know $b > r$, so

$$a \geq b + r > 2r,$$

which implies $r < a/2$. This argument clearly works at each step of the algorithm. Along the way, the $a$ in Equation 3 is the remainder two before the $r$ in that equation.

(b) Prove that the Euclidean algorithm takes at most $2\log_2(a)$ steps. Show that is at most five times the number of decimal digits of $a$.

**Solution**

After $2\log_2(a)$ steps the remainder could be at most

$$\frac{a}{2^{\log_2(a)}} = 1$$

so the algorithm will have terminated.

Now $2\log_2(a)$ is (approximately) twice the number of binary digits of $a$. To get the number of decimal digits, multiply by $\log_2(10) = 3.32$. That says the algorithm terminates in at most 6.64 times the number of decimal digits of $a$ steps.

I asked you to prove the bound was five times the number of decimal digits. That's true, but you need a more sophisticated argument. Lamé first proved it, using the fact that the worst case for the Euclidean algorithm is the one that starts with adjacent Fibonacci numbers, making all the quotients 1.

4. Computer programs Since CS110 is a prerequisite for this course, you should all be able to write these programs. But for some of you your programming skills are so rusty that polishing them up so you can answer this question isn't worth the time. If that's the case, just say so and skip it. You may do this in any language you choose (there are easier ones than Java). You might even be able to write it in Excel without macros. I'd enjoy seeing that.

   (a) Write a program (function, method, procedure) that accepts two integers as input and produces their gcd as output. A well written program will do the right thing when the input values are any integers – positive, negative or zero. The only case that might require special treatment is $\gcd(0, 0)$. There is no right answer then. Just make sure your program doesn't crash.

   - If possible, the function that does the computation should not do any printing – it should return the answer. Then write a program that calls that function and prints the output. Printing is the responsibility of the calling program. If possible, the calling program should get the integer input values from the command line, or from stdin (System.in in Java). (Of course that's possible. But if your programming skills are so rusty that it's really difficult, don't spend time on it.)
   - It's really easy to find solutions to this problem on the web. I'd rather you wrote your own, but won't insist. If you do get one from the web you must acknowledge and understand the source and run the program to test it.
   - *Instrument* your program so that when a reporting flag is set it prints the number of iterations/recursions. Use your instrumentation to check the assertion in Problem 3b.
   - Submit hard copy of your program. If possible, do that in this LaTeX document using the `listings` package. This is a particularly useful part of the homework for cs students.

   (b) Improve your solution to the previous problem so that your program both finds the gcd of its input values and also finds the coefficients for a linear integral combination of the inputs that produces the gcd. I'd still rather your function do no printing, but that's harder to arrange now that there are three integer outputs rather than just one. Do that if you can, but if you can't don't worry.

   (c) If you can, write both programs so that they run in constant space. In particular, no recursive calls, since that would create a logarithmic number of stack frames. This is easier for the first program than the second.

**Solution**

Here's one I wrote years ago in Java

```
// Implementing the Euclidean algorithm.
//
// Ethan Bolker
// October, 2008 for cs320
//
```

```
// algorithm courtesy of
// http://en.wikipedia.org/wiki/Extended_Euclidean_algorithm
// (I could have done it myself but this was quicker.)

public class Euclid
{
    private int m;
    private int n;
    private int a;
    private int b;
    private int d;
    private int steps;
    private double log2bound;

    /**
     *  Euclid constructor.
     *
     *  @param m the first integer
     *  @param m the second integer
     *  @exception NumberFormatException if both m and n are 0
     */
    public Euclid( int m, int n )
    {
        if ((m==0 && n==0)) {
            throw new NumberFormatException();
        }
        log2bound = Math.ceil(
              Math.log(Math.max(Math.abs(m), Math.abs(n)))/0.6931);
        this.m = m;
        this.n = n;
        steps = 0;
        // now do the work
        d = m;
        a = 1;
        b = 0;
        int nexta = 0;
        int nextb = 1;
        int q;
        int r = m;
        while (n != 0 ) {
            d = r;
            q = m/n; // integer arithmetic truncates
            r = m%n;

            m = n; // changes only local copy of m, not this.m
            n = r;

            int tmp = nexta;
            nexta = a - q*nexta;
            a = tmp;

            tmp = nextb;
            nextb = b - q*nextb;
            b = tmp;

            ++steps;
        }
    }

    // getters for all numbers of interest
```

```
    public int getM() { return m; }
    public int getN() { return n; }
    public int getA() { return a; }
    public int getB() { return b; }
    public int getD() { return d; }
    public int getSteps() { return steps; }

    public String toString()
    {
        return "gcd( " + getM() + ", " + getN() + ") = " + getD() +
            " = " +
            getA() + "*" + getM() + " + " + getB() + "*" + getN() +
            "\nsteps: " + getSteps() +
            "\nlog2(max(m,n)): " + log2bound;
    }


    public static void main( String[] args )
    {
        int m = 0;
        int n = 0;
        Euclid euclid = null;

        // collect arguments
        try {
            m = Integer.parseInt(args[0]);
            n = Integer.parseInt(args[1]);
        }
        catch( Exception e ) {
            System.out.println("usage: java Euclid m n");
            System.exit(0);
        }
        // create new Euclid object to compute answers
        try {
            euclid = new Euclid(m,n);
        }
        catch( NumberFormatException e ) {
            System.out.println("m and n can't both be 0");
            System.exit(1);
        }
        System.out.println(euclid);
    }

}
```

5. Calculate

$$54^{100} \pmod{101}$$

using the fast **Right-to-left binary method** described at `http://en.wikipedia.org/wiki/Modular_exponentiation`. Use a calculator along the way. If you don't like the wikipedia discussion you can find lots of others by googling `fast modular exponentiation`. This is the computation I botched at the end of lecture. Note: you should get 1 as the answer – that's Fermat's Little Theorem.

**Solution**

None provided here. Just about everyone managed to program themselves to follow the algorithm and get the right answer.

6. Large primes. Note: you can look up the answers to almost all these questions. In fact I've asked you to do that for the last few. I'd rather you didn't at the beginning – you'll learn more that way.

(a) Prove

**Theorem 2.** *If $2^n - 1$ is prime then $n$ is prime.*

Hint: If $n = ab$ is not prime then $2^n = (2^a)^b$. Then use a finite geometric series.

**Solution**

*Proof.* If $n$ is not prime then write $n = ab$ where both $a$ and $b$ are greater than 1. Then

$$\begin{aligned} 2^n - 1 &= (2^a)^b - 1 \\ &= (2^a - 1)((2^a)^{b-1} + (2^a)^{b-2} + \cdots + 1). \end{aligned}$$

That says $2^n - 1$ has a nontrivial factor $2^a - 1$ so isn't prime.

□

Primes of this form are called "Mersenne primes". The first few are $3 = 2^2 - 1$, $7 = 2^3 - 1$, $31 = 2^5 - 1$ and $127 = 2^7 - 1$.

(b) State the *converse* of Theorem 2. Then show that it is false. Hint. Try to continue the list above in the obvious way.

**Solution** The converse of Theorem 2 is

$$\text{If } n \text{ is prime then } 2^n - 1 \text{ is prime.}$$

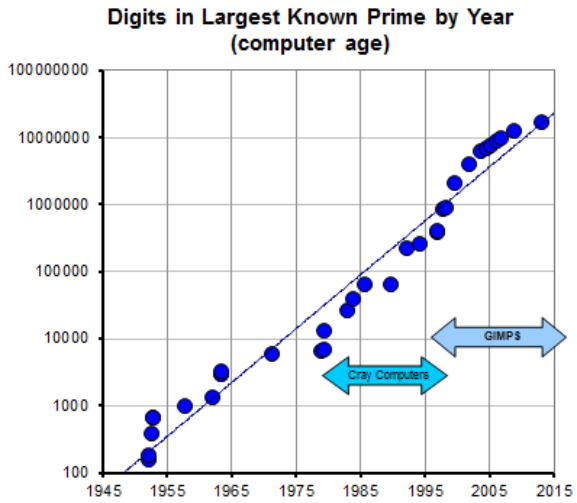
One counterexample is enough to show it's false.

$$2^{11} - 1 = 2047 = 23 \times 89.$$

(c) Look up some information about how the largest known prime has increased over time. Argue from the data that the logarithm of the logarithm of the largest known prime is growing. That means the largest known prime is growing much faster than exponentially!

**Solution**

This image from `https://primes.utm.edu/notes/by_year.html`




shows that the number of digits in the largest known prime is growing linearly on a logarthmic scale. That means the number of digits is growing exponentially, so the number itself is growing at a doubly exponential rate.

(d) Check Chris Caldwell's Prime Pages at `http://primes.utm.edu/`. Read about GIMPS at `http://www.mersenne.org/`. Tell me something you found particularly interesting (not just something from the first page).

**Solution** None here. I enjoyed finding out what you enjoyed finding out.

7. Show that $2^{340} \equiv 1 \mod 341$ even though 341 is not prime. What is the connection between this result and the converse of Fermat's Little Theorem?

   **Solution**

   The computation is easy. The converse of Fermat's Little Theorem says that if $a^{n-1} \equiv 1 (\mod n)$ then $n$ is prime. Since $341 = 11 \times 31$ this computation provides a counterexample.

   There's a long history to this example and more general ones. Look up "Carmichael numbers" to find more.

8. Eratosthenes' RSA public key is

$$n = 10967535067$$
$$e = 1051$$

   Archimedes encrypts his message with this key and sends Eratosthenes

$$C = 1963501580$$

   (a) Break the encryption. (Use any tools you like; tell me how you did it.)

   **Solution**

   I used Wolfram alpha (`http://www.wolframalpha.com/`) to create the key, so I'll use it to break the encryption too.

   First I asked it to `factor 1963501580` and was told

   `104723 104729  (2 distinct prime factors)`

   Then `1051^-1 mod (104722*104728)` led to `6188034339`.

   Finally, `1963501580^6188034339 (mod 10967535067)` is `3141592653`.

   That's the beginning of the decimal expansion of $\pi$!

   (b) Why is the message appropriate?

   The message is appropriate because Archimedes invented an elegant method for finding rational approximations to $\pi$ and used it to determine that

$$3\frac{1}{7} < \pi < 3\frac{10}{71}$$

   – a truly remarkable calculation for his time. His friend Eratosthenes measured the diameter of the Earth. He'd have been interested in the latest news about $\pi$. Of course Archimedes couldn't have told him this way – no RSA encryption, no Wolfram Alpha, no decimal representation for numbers!

   (c) What is each of these Greeks famous for?

   The previous answer describes one accomplishment of each. Eratosthenes also invented his eponymous [1] sieve for finding primes. Archimedes was famous for lots of other contributions to mathematics and physics.

[1] Look up the meaning of this word if you don't know it.

```
% Math 320 hw3
%
\documentclass{article}
\pagestyle{empty}
\usepackage[textheight=10in]{geometry}
\usepackage{amsmath}
\usepackage{amsthm}
\usepackage{listings}
\usepackage{graphicx}
\usepackage{verbatim}
\usepackage{hyperref}
\newtheorem{theorem}{Theorem}
\newcommand{\coursehome}
{http://www.cs.umb.edu/~eb/320}
\title{Discrete Mathematics \\
Homework 3
}
\author{Ethan Bolker}
%\date{September 1, 2014}
\newcommand{\ZZ}{\mathbb{Z}}
%create (mod n) macro
\newcommand{\mm}[1]{%
\ensuremath{(\text{mod } #1)}}
\begin{document}
\maketitle
\noindent
Due: ???

We're studying number theory (leading up to cryptography) for the next
few weeks. You can read Unit NT in Bender and Williamson.
This homework contains some programming exercises, some number
theory computations and some proofs. The questions aren't arranged in
any particular order. Don't just start at the beginning and do as many
as you can -- read them all and do the easy ones first.
I may add to this homework from time to time.
\section*{Exercises}
\begin{enumerate}
\item Use the Euclidean algorithm to compute the greatest common
divisor $d$ of
$59400$ and $16200$ and find integers $x$ and $y$ such that
\begin{equation*}
59400x + 16200y = d
\end{equation*}
I recommend that you do this by hand for the logical flow (a
calculator is fine for the arithmetic) before writing the computer
programs that follow.
There are many websites that will do the computation for you, and even
show you the steps. If you use one, tell me which one.

We wish to find $\gcd(16200, 59400)$.

\textbf{Solution}

\emph{\LaTeX{} note.} These are separate equations in the source
  file. They'd be better formatted in an \verb!align*! environment.


\[59400=3(16200)+10800\ \quad \text{next find } \gcd(10800,16200)\]
\[16200=1(10800)+5400 \quad \text{next find } \gcd(5400,10800)\]
\[10800=2(5400)+0 \quad \gcd(16200,59400) = 5400\]
```

```
\[59400x+16200y = 5400\]
\[5400(11x+3y)=5400\]
\[11x+3y=1\]

By inspection, we can easily determine that $(x,y) = (-1,4)$ (among
other solutions), but let's use the algorithm.

\[11=3\cdot 3+2\]
\[3=2\cdot 1+1\]

\[1=3-(2)\cdot 1\]
\[1=3-(11-3 \cdot 3)\]
\[1=11(-1)+3(4)\]
\[(x,y)=(-1,4)\]

\item An interesting class of examples.
\begin{enumerate}
\item Use the Euclidean Algorithm \emph{by hand} to find an integral
solution to the equation
%
\begin{equation*}
55x + 89y = 1
\end{equation*}

You can probably see that 55 and 89 are mutually prime and so
$\gcd(55,89) = 1$, but let's run it through the Euclidean Algorithm.


\[\gcd(55,89)\quad 89=1\cdot 55+34\]
\[\gcd(34,55)\quad 55=1\cdot 34+21\]
\[\gcd(21,34)\quad 34=1\cdot 21+13\]
\[\gcd(13,21)\quad 21=1\cdot 13+8\]
\[\gcd(8,13)\quad 13=1\cdot 8+5\]
\[\gcd(5,8)\quad 8=1\cdot 5+3\]
\[\gcd(3,5)\quad 5=1\cdot 3+2\]
\[\gcd(2,3)\quad 3=1\cdot 2+1\]
\[\gcd(1,2)\quad 2=1\cdot 1+1\]
\[\gcd(1,1)\quad 1=1\cdot 1+0\]

Now we run everything through backwards.

\[1=2-1 \cdot 1\]
\[2-1(3-1 \cdot 2)\]
\[2\cdot (2)-3\]
\[2\cdot (5-1 \cdot 3)-3\]
\[2\cdot 5-3 \cdot (3)\]
\[2\cdot 5-3(8-1\cdot 5)\]
\[5\cdot (5)-3 \cdot 8\]
\[5(13-1\cdot 8)-3\cdot 8\]
\[5\cdot 13 - 8\cdot (8)\]
\[5\cdot 13-8(21-1\cdot 13)\]
\[13\cdot (13)-8\cdot 21\]
\[13(34-1\cdot 21)-8\cdot 21\]
\[13\cdot 34-21\cdot (21)\]
\[13\cdot 34-21(55-1\cdot 34)\]
\[34\cdot (34)- 21\cdot 55\]
\[34(89-1\cdot 55)-21\cdot 55\]
\[34\cdot 89-55\cdot 55=1\]
\[(x,y) = (-55,34)\]
```

```
\item
Your answer to the previous question should suggest a nice identity
about the Fibonacci numbers. State it, then prove it by
induction. (Look up ‘‘Fibonacci numbers’’ if you have to.)

\textbf{Solution}

What you should have \emph{noticed} about the previous calculation is
that all the quotients on the way down are just $1$ -- the smallest
they could possibly be. You should also have \emph{recognized} the
sequence
\begin{equation*}
89, 55, 34, 21, 13, 8, 5, 3, 2, 1
\end{equation*}
as the \emph{Fibonacci numbers} -- I even gave that away in the hint.

That’s not an accident. The Fibonacci numbers are defined by the
recursion relation
%
\begin{align}\label{eq:fib}
F_1 & = 1 \notag \\
F_2 & = 2 \notag \\
F_{n+1} & = F_{n} + F_{n-1}
\end{align}

Since we started with two adjacent Fibonacci numbers, that defining
relation implies that the quotients in the Euclidean algorithm will
all be $1$ and that the remainders will count down the Fibonacci
numbers to $1$, the greatest common divisor.

Equation~\ref{eq:fib} is \emph{not} the ‘‘nice identity’’ I asked for
-- it’s just the definition of the Fibonacci numbers. What I hoped
you’d notice is that the $x$ and $y$ in the identity
%
\[
-55 \times 55 + 34 \times 89 = 1
\]
%
are themselves both Fibonacci numbers. That suggests
%
\begin{theorem}
For each $n > 2$,
\begin{equation}\label{eq:fibid}
F_n^2 - F_{n+1}F_{n -1} = (-1)^n
\end{equation}
\end{theorem}
\begin{proof}
Suppose we knew that
Equation~\ref{eq:fibid} was true for a particular value of $n$. Then its
truth for $n+1$ follows from the computation
%
\begin{align*}
F_{n+1}^2 - F_{n+2}F_{n} & =
F_{n+1}(F_n + F_{n-1}) - (F_{n+1} + F_n)F_n \\
& = F_{n+1}F_{n-1} -  F_nF_n \\
& = -(-1)^n = (-1)^{n+1}.
\end{align*}

Since
```

```
\begin{equation*}
2^2 - 3 \times 1 = 1,
\end{equation*}
%
Equation~\ref{eq:fibid} is true for $n=1$. Then by induction it's true
for all $n$.
\end{proof}

If you found the different solution
\begin{equation*}
34 \times 55 - 21 \times 89 =1
\end{equation*}
you'd come up with a slightly different Fibonacci number identity to
prove:
%
\begin{equation*}
F_{n-1}F_{n} - F_{n-2}F_{n+1} = \pm 1.
\end{equation*}

\end{enumerate}

\item Logarithmic time
\begin{enumerate}
\item Prove that if you carry out \emph{two steps} in the Euclidean
algorithm for $\text{gcd}(a,b)$ with $a >b$ the remainder is less
than $a/2$.

\textbf{Solution}

This argument is Jiho Choi's. It's  better than the one I knew (which
is the one most of you either found or invented). The algorithm starts with
\begin{equation}\label{eq:euclid}
a = bq + r .
\end{equation}
%
Since $a > b$, I know $q \ge 1$ so $a \ge b+r$. I also know $b > r$,
so
\begin{equation*}
a \ge b + r > 2r ,
\end{equation*}
which implies $r < a/2$.
%
This argument clearly works at each step of the algorithm. Along the
way, the $a$ in Equation~\ref{eq:euclid} is the remainder two before
the $r$ in that equation.

\item \label{logtime} Prove that the Euclidean algorithm takes at most
$2\log_{2}(a)$ steps. Show that is at most five times the number
of decimal digits of $a$.

\textbf{Solution}

After $2\log_{2}(a)$ steps the remainder could be at most
\begin{equation*}
\frac{a}{2^{\log_{2}(a)}} =1
\end{equation*}
so the algorithm will have terminated.

Now $2\log_{2}(a)$ is (approximately) twice the number of binary digits of
$a$. To get the number of decimal digits, multiply by
```

```
$\log_{2}(10) = 3.32$. That says the algorithm terminates in at most
6.64 times the number of decimal digits of $a$ steps.

I asked you to prove the bound was five times the number of decimal
digits. That's true, but you need a more sophisticated
argument. Lam\'e first proved it, using the fact that the worst case
for the Euclidean algorithm is the one that starts with adjacent
Fibonacci numbers, making all the quotients $1$.

\end{enumerate}
\item Computer programs
Since CS110 is a prerequisite for this course, you should all
be able to write these programs. But for some of you your programming
skills are so rusty that polishing them up so you can answer this
question isn't worth the time. If that's the case, just say so and
skip it.
You may do this in any language you choose (there are easier ones than
Java). You might even be able to write it in Excel without macros. I'd
enjoy seeing that.
\begin{enumerate}
\item Write a program (function, method, procedure) that accepts two
integers as input and produces their gcd as output.
A well written program will do the right thing when the input values
are any integers -- positive, negative or zero. The only case that
might require special treatment is $\text{gcd}(0,0)$. There is no right
answer then. Just make sure your program doesn't crash.
\begin{itemize}
\item
If possible, the function that does the computation should not do any
printing -- it should return the answer. Then write a program that
calls that function and prints the output.
Printing is the responsibility of the calling program. If possible, the
calling program should get the integer input values from the command
line, or from \lstinline!stdin! (System.in in Java). (Of course that's
possible. But if your programming skills are so rusty that it's really
difficult, don't spend time on it.)
\item It's really easy to find solutions to this problem on the
web. I'd rather you wrote your own, but won't insist. If you do
get one from the web you must acknowledge and understand the source
and run the program to test it.
\item \emph{Instrument} your program so that when a reporting flag is
set it prints the number of iterations/recursions. Use your
instrumentation to check the assertion in Problem~\ref{logtime}.
\item Submit hard copy of your program. If possible, do that in this
\LaTeX{} document using the \verb!listings! package. This is a
particularly useful part of the homework for cs students.
\end{itemize}
\item
Improve your solution to the previous problem so that your program
both finds the gcd of its input values and also finds the coefficients
for a linear integral combination of the inputs that produces the
gcd.
I'd still rather your function do no printing, but that's harder to
arrange now that there are three integer outputs rather than just
one. Do that if you can, but if you can't don't worry.
\item If you can, write both programs so that they run in constant
space. In particular, no recursive calls, since that would create a
logarithmic number of stack frames.
This is easier for the first program than the second.
\end{enumerate}
```

\textbf{Solution}

Here's one I wrote years ago in Java

\begin{lstlisting}

```
// Implementing the Euclidean algorithm.
//
// Ethan Bolker
// October, 2008 for cs320
//
// algorithm courtesy of
// http://en.wikipedia.org/wiki/Extended_Euclidean_algorithm
// (I could have done it myself but this was quicker.)

public class Euclid
{
    private int m;
    private int n;
    private int a;
    private int b;
    private int d;
    private int steps;
    private double log2bound;

    /**
     *  Euclid constructor.
     *
     *  @param m the first integer
     *  @param m the second integer
     *  @exception NumberFormatException if both m and n are 0
     */
    public Euclid( int m, int n )
    {
        if ((m==0 && n==0)) {
            throw new NumberFormatException();
        }
        log2bound = Math.ceil(
              Math.log(Math.max(Math.abs(m), Math.abs(n)))/0.6931);
        this.m = m;
        this.n = n;
        steps = 0;
        // now do the work
        d = m;
        a = 1;
        b = 0;
        int nexta = 0;
        int nextb = 1;
        int q;
        int r = m;
        while (n != 0 ) {
            d = r;
            q = m/n; // integer arithmetic truncates
            r = m%n;

            m = n; // changes only local copy of m, not this.m
            n = r;

            int tmp = nexta;
            nexta = a - q*nexta;
```

```
            a = tmp;

            tmp = nextb;
            nextb = b - q*nextb;
            b = tmp;

            ++steps;
        }
    }

    // getters for all numbers of interest
    public int getM() { return m; }
    public int getN() { return n; }
    public int getA() { return a; }
    public int getB() { return b; }
    public int getD() { return d; }
    public int getSteps() { return steps; }

    public String toString()
    {
        return "gcd( " + getM() + ", " + getN() + ") = " + getD() +
            " = " +
            getA() + "*" + getM() + " + " + getB() + "*" + getN() +
            "\nsteps: " + getSteps() +
            "\nlog2(max(m,n)): " + log2bound;
    }


    public static void main( String[] args )
    {
        int m = 0;
        int n = 0;
        Euclid euclid = null;

        // collect arguments
        try {
            m = Integer.parseInt(args[0]);
            n = Integer.parseInt(args[1]);
        }
        catch( Exception e ) {
            System.out.println("usage: java Euclid m n");
            System.exit(0);
        }
        // create new Euclid object to compute answers
        try {
            euclid = new Euclid(m,n);
        }
        catch( NumberFormatException e ) {
            System.out.println("m and n can't both be 0");
            System.exit(1);
        }
        System.out.println(euclid);
    }

}
\end{lstlisting}
\item Calculate
%
\begin{equation*}
54^{100} \mm{101}
```

```
\end{equation*}
%
using the fast
\textbf{Right-to-left binary method} described at
\url{http://en.wikipedia.org/wiki/Modular_exponentiation}.
Use a calculator along the way.
If you don't like the wikipedia discussion you can find lots of others
by googling \verb!fast modular exponentiation!.
This is the computation I botched at the end of lecture.
Note: you should get $1$ as the answer -- that's Fermat's Little
Theorem.

\textbf{Solution}

None provided here. Just about everyone managed to program themselves
to follow the algorithm and get the right answer.

\item Large primes.
Note: you can look up the answers to almost all these questions. In
fact I've asked you to do that for the last few. I'd rather you didn't
at the beginning -- you'll learn more that way.
\begin{enumerate}
\item Prove
\begin{theorem}\label{thm:mersenne}
If $2^n -1$ is prime then $n$ is prime.
\end{theorem}
Hint: If $n = ab$ is not prime then $2^n = (2^a)^b$. Then use a finite
geometric series.

\textbf{Solution}

\begin{proof}
If $n$ is not prime then write $n = ab$ where both $a$ and $b$ are
greater than $1$. Then
%
\begin{align*}
2^n -1 &  = (2^a)^b -1 \\
 & = (2^a -1)((2^a)^{b-1} + (2^a)^{b-2} + \cdots + 1) .
\end{align*}

That says  $2^n - 1$ has a nontrivial factor $2^a -1$ so isn't prime.

\end{proof}

Primes of this form are called ``Mersenne primes''. The first few are
$3 = 2^2-1$, $7=2^3-1$, $31=2^5-1$ and $127=2^7-1$.
\item State the \emph{converse} of
Theorem~\ref{thm:mersenne}. Then show that it is false.
Hint. Try to continue the list above in the obvious way.

\textbf{Solution} The converse of
Theorem~\ref{thm:mersenne} is

\begin{center}
If $n$ is prime then $2^n -1$ is prime.
\end{center}

One counterexample is enough to show it's false.

\begin{equation*}
```

```
2^{11} - 1 = 2047 = 23 \times 89.
\end{equation*}

\item Look up some information about how the largest known prime has
increased over time. Argue from the data that the logarithm of the
logarithm of the largest known prime is growing. That means the
largest known prime is growing much faster than exponentially!


\textbf{Solution}

This image from \url{https://primes.utm.edu/notes/by_year.html}

\begin{center}
\includegraphics[width=3in]{largestprime}
\end{center}

shows that the number of digits in the largest known prime is growing
linearly on a logarthmic scale. That means the number of digits is
growing exponentially, so the number itself is growing at a doubly
exponential rate.

\item Check Chris Caldwell's Prime Pages at
\url{http://primes.utm.edu/}. Read about GIMPS at
\url{http://www.mersenne.org/}. Tell me something you found
particularly interesting (not just something from the first page).

\textbf{Solution} None here. I enjoyed finding out what you enjoyed
finding out.

\end{enumerate}
\item Show that $2^{340} \equiv 1 \mod{341}$ even though $341$ is not
prime. What is the connection between this result and the converse
of Fermat's Little Theorem?

\textbf{Solution}

The computation is easy. The converse of Fermat's Little Theorem says
that if $a^{n-1} \equiv 1 \mm{n}$ then $n$ is prime. Since $341 = 11
\times 31$ this computation provides a counterexample.

There's a long history to this example and more general ones. Look up
``Carmichael numbers'' to find more.

\item Eratosthenes' RSA public key is
%
\begin{align*}
n & = 10967535067 \\
e & = 1051
\end{align*}
Archimedes encrypts his message with this key and sends Eratosthenes
%
\begin{equation*}
C = 1963501580
\end{equation*}
\begin{enumerate}
\item Break the encryption. (Use any tools you like; tell me how you
did it.)

\textbf{Solution}
```

```
I used Wolfram alpha (\url{http://www.wolframalpha.com/}) to create
the key, so I'll use it to break the encryption too.

First I asked it to \verb!factor 1963501580! and was told

\verb!104723 104729  (2 distinct prime factors)!

Then \verb!1051^-1 mod (104722*104728)! led to
\verb!6188034339!.

Finally, \verb!1963501580^6188034339 (mod 10967535067)! is
\verb!3141592653!.

That's the beginning of the decimal expansion of $\pi$!

\item Why is the message appropriate?

The message is appropriate because Archimedes invented an elegant
method for finding rational approximations to $\pi$ and used it to
determine that
%
\begin{equation*}
3\frac{1}{7} < \pi < 3\frac{10}{71}
\end{equation*}
%
-- a truly remarkable calculation for his time. His friend
Eratosthenes measured the diameter of the Earth. He'd have been
interested in the latest news about $\pi$. Of course Archimedes
couldn't have told him this way -- no RSA encryption, no Wolfram
Alpha, no decimal representation for numbers!

\item What is each of these Greeks famous for?

The previous answer describes one accomplishment of each. Eratosthenes
also invented his eponymous
\footnote{Look up the meaning of this word if you don't know it.}
sieve for finding primes. Archimedes was famous for lots of other
contributions to mathematics and physics.
\end{enumerate}
\end{enumerate}
\newpage
\verbatiminput{\jobname}
\end{document}
```