Project 2 - Factoring

Due July 23, 2017 at 11:55pm

1 Background

We talked briefly in class about the difficulties of factoring numbers. In general factoring a number into its prime factors is an NP-complete problem, meaning there is no polynomial time algorithm for every case.

This difficulty is what RSA is based on, if it is difficult to factor integers then it is difficult to break RSA encryption, as discussed in our lectures.

2 Project

For this project you will be factoring numbers, significantly smaller than the numbers used for RSA of course, but they will still be fairly large numbers. So large in fact that if you are using `C++` you may not be able to natively represent these numbers. If you choose to use `C++` for this project you will need to determine a way to store the number so that they can be used in your program. I would recommend the GNU Multiple Precision Library (GMP). Examine their webpage for download and use instructions.

In addition to factoring there will be some RSA messages given to you that you may choose to decipher, if you can factor the public key.

3 Expectations

You will be given a list of many numbers and asked to factor them as quickly as you can. We only briefly talked about factoring integers in class with the naive algorithm. You will certainly have to find speed ups. There are some obvious speed ups, you don't have to check every power of two, or every power of 3 or 5 or 7 etc this allows you to skip significantly many numbers. Using this you will be able to factor some of the earlier numbers in a reasonable amount of time.

There are other methods to factoring. All of the numbers in the given list are of the form pq for primes p and q. Meaning that once you have one factor you have both. This means that really all you need are prime numbers, if you can generate a list of large enough primes then you can factor all of these numbers.

Other methods of factoring include Pollard's Rho, Pollard's $p-1$, Quadratic Sieve. You may choose to look into some of these advanced options to help speed up your program, though if you decide not to that is fine as well.

There are other methods of factoring as well. The wiki page on Integer Factorization might be helpful.

4 Deliverables

For this project you will turn in a tarball of all of your work. If you do not know how to make a tarball let me know.

Your work should include:

- your program which reads numbers from a file and attempts to factor them
- the list of numbers you are factoring
- the output of your program in the form of

  $$n = p * q$$

  for each n in the input file
- a Makefile that compiles the program any way that you want it to and has a run option. To run your program I should just have to do

  ```
  $ make
  $ make run
  ```

  and see the results.
- Any libraries that you used, this would be libraries that you might need to download so that you can represent numbers large enough.
- a readme file that explains in a high level what your code does and how it works. You should discuss difficulties you ran into on this project. Also this would be where you mention any of your decryptions.

5 Restrictions

There are numerous libraries that have already implemented algorithms for use. You should not use these in your final turn in. All code that you submit should be yours. If your project just calls someonelse's library for factoring numbers you will not do well on this project.

Similarly for primes, there are numerous lists of primes online, it might be helpful to use them to practice factoring numbers but your final turn in should not have this. That is if you need a list of primes for your project you should generate that list yourself.

Absolutely nothing should be hard-coded. Once you factor some of the numbers you should not store the result so that future runs are faster. Though you may take advantage of any patterns you find.

6 Grading

If you make a decent attempt at this project and it is clear you spent more than 2 hours on it you will get a 70 on this project (easy right!). But there are very easy ways to change this grade. Things that may help:

- Your code compiles correctly
- Your code works correctly on numbers it claims to factor
- Your code can handle reading arbitrarily large numbers
- Your code is very well documented with many comments explaining what each function is doing. (In my opinion this is handled very well at the beginning of a function (or above the function in `C++`), rather than comments dispersed throughout the function)
- You can factor some of the 15 digit numbers in less than 30 seconds
- You can factor some of the 20 digit numbers in less than 30 seconds

Some of the more advanced tactics that will significantly help your grade:

- Discussing in your readme about various approaches that you took and how you decided between them.
- Noticeably trying to improve performance
- Decrypting some of the messages
- Factoring all of the numbers

In particular if you can manage to factor all of the numbers faster than I could (1.12 seconds for all numbers, think of this as a crazy stretch goal).

7 Final thoughts

Start this project immediately. This project is not conceptually difficult (I hope!) but there are a lot of details involved. You should start immediately to run into any issues as soon as possible. If you do run into issues you should be well aware of the help you can receive. You may work with other students as much as you would like. HOWEVER you must write your own code. You may talk with others about the structure of the program. If there is a specific part of the program you do not how to code you may ask others for examples, but at the end of the day you must write all of your own code.

You may of course ask me any questions you want and I will be very helpful. This is expected to be:

1. interesting – factoring numbers and dealing with prime numbers are fun.

2. difficult – handling the details of this program can be very difficult to keep straight.

3. illuminating – seeing how some of these algorithms are implemented and why they work can be very illuminating.

I will be testing your code on a Linux machine meaning that you should write your code to run on a Linux machine. You may do most of your coding on Mac or Windows but before submission you should run it on the VM and make sure it works flawlessly (though possibly slower, due to VM) there.

You may find it beneficial to combine different forms of factoring, to help speed up your program.

If at any point in this project you find that you have questions you should not hesitate to ask.