AEP 3: The Euclidean Algorithm and Extended Euclidean Algorithm

Overview

In this AEP, you will explore a fundamental algorithm for working with integers, divisibility, and modular arithmetic that has a special place in computing applications — the **Euclidean Algorithm** and its cousin the **Extended Euclidean Algorithm**.

Learning Targets associated with this AEP:

- **A.3**: I can compute `a % b` given integers `a` and `b` and perform arithmetic mod n.

Remember, AEPs do not have fixed deadlines; simply work on this item until you are ready to submit it. But remember the **Two Items Per Week Rule.**

Technology Background

No special technology skills are needed for this AEP.

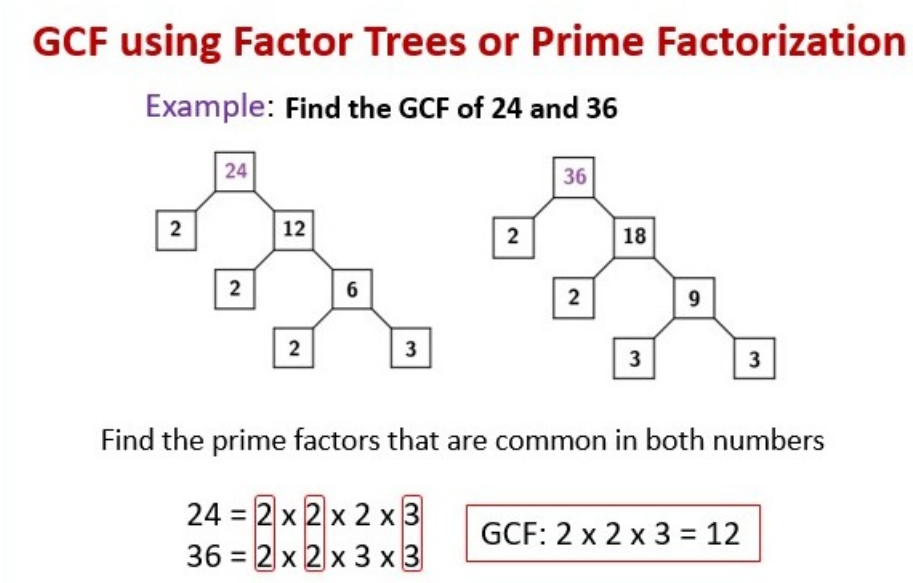
AEP Description and Tasks

What this AEP is about

An important arithmetic computation that has a surprising number of applications in computer science and cryptography is **finding the greatest common divisor or "GCD" of two positive integers**. The GCD of two integers is the largest integer that divides both of the integers we start with. For example, the greatest common divisor of 10 and 15 is 5; the GCD of 18 and 20 is 2; the GCD of 19 and 22 is 1.

The GCD has shown up for us already at least once in our side trips into cryptography, in the Diffie-Hellman key protocol. Recall that in this procedure, in which Alice and Bob agree on a secret key without ever exchanging private information in the clear, Alice and Bob first decide on a prime number p and then an integer g, and this integer *g must have no factors in common with $p-1$ other than 1.** That is, the GCD of g and $p-1$ must be 1. This criterion of having two numbers with a GCD of 1 is strangely common in cryptographic systems.

You probably learned how to find the GCD of two numbers in school by *factoring* the two numbers and looking for common factors that way. My kids learned this in elementary school by making a "factor tree", like this:



("GCF" means the same thing as "GCD".) This is fine for relatively small numbers. But if we needed to find the GCD of two large numbers — like something in the 200-300 decimal digit range — factoring simply takes too long.

Instead, a far more efficient algorithm for finding the GCD came to us almost 2000 years ago, from the mathematician **Euclid**.

Euclid is most famous for establishing geometry as a rigorous discipline through his multi-volume work *The Elements*. (Here's an interactive online edition of the *Elements* to play with in your spare time!) Notice the geometric figure on the scroll in the statue above. But in the *Elements*, Euclid also introduced some numerical procedures that today are used every day in computing. The one we focus on here, is the **Euclidean algorithm** that finds the GCD of two integers.

Setup (do this first)

Before doing the tasks below, you'll need to go learn about the Euclidean Algorithm and get familiar with using it. Here are videos about the Euclidean Algorithm:

- The Euclidean Algorithm (from GVSUMath) (6:57) https://www.youtube.com/watch?v=p5gn2hj51hs
- Euclidean Algorithm: An example (from Socratica) (2:03) https://www.youtube.com/watch?v=fwuj4yzoX1o

Here are some websites (that aren't videos):

- Article from Khan Academy (includes an embedded video, and some properties of the GCD) https://www.khanacademy.org/computing/computer-science/cryptography/modarithmetic/a/the-euclidean-algorithm
- Article from Rutgers University (includes some good CS discussion, e.g. why does the Euclidean

Algorithm always stop?) https://sites.math.rutgers.edu/~greenfie/gs2004/euclid.html

You will also need to learn about the **Extended Euclidean Algorithm** but resources for learning this are below.

Finally, here are a couple of computational tools to help you:

- Wolfram|Alpha will find a GCD — just enter in, for example `gcd(15,20)` to find the GCD of 15 and 20.
- Python 3 has a GCD function, but it's not built-in. When you start Python, only the bare essential functions are loaded so as to save memory. But a vast assortment of functions are available to be **imported** from different **libraries**, which are repositories of additional functions you can include in Python if you want. The GCD function is one of those, and it's found in a standard library called `math`. To load it into your session for use, first enter `from math import gcd` and hit enter. Now the `gcd` function is available, and you can enter, for example, `gcd(15,20)` to get the GCD of 15 and 20.

Tasks for this AEP

1. Go to https://www.random.org/integers/ and generate two random 5-digit integers. Then use the Euclidean algorithm to find their GCD. Show all your work and clearly indicate the GCD. (Be sure to check your work using a computational tool!)

2. Let's suppose you were using the Diffie-Hellman algorithm to find a secret key. You would need to pick a large prime number p and then find a random integer g so that $\gcd(g, p-1) = 1$. The pair (g, p) would be made public. Go through the following process to generate a public key:

   (a) Choose a prime number p that is at least 10 digits long. You can just look one of these up and use it, but explain how you got your prime number.

   (b) Choose a random integer g and use the Euclidean Algorithm to prove that your choice has no common factors with $p-1$ other than 1. Show your work on the Euclidean Algorithm computation.

3. An interesting and highly useful feature of GCD's is that the GCD of two integers can always be expressed as combination of the two integers themselves. In general, if $\gcd(a, b) = n$, then there are integers x and y such that $ax + by = n$. For example, $\gcd(20, 32) = 4$ and notice that $4 = 20(-3) + 32(2)$. Another example: $\gcd(1970, 2020) = 10$, and notice $10 = 2020(79) + 1970(-81)$. If we use the Euclidean Algorithm to find the GCD, we can use the data from the algorithm to find the values of x and y to make the combination work out, by running the algorithm in reverse. The process of doing so is called the **Extended Euclidean Algorithm**. Go watch this video to see how it works: https://www.youtube.com/watch?v=hB34-GSDT3k (There are many others you can watch too.) Then do the following: Look back at the 5-digit integers you used in task 1. Call those a and b, and let the GCD be called n. Use the Extended Euclidean Algorithm to find values of x and y such that $ax + by = n$. Again, for example, if $a = 12345$ and $b = 34567$, you should have found that $\gcd(12345, 34567) = 1$; you would use the Extended Euclidean Algorithm to find that $12345(13824) + 34567(-4937) = 1$. Obviously show all the work in the Extended Euclidean Algorithm computation, and verify at the end that the combination does actually equal the GCD.

Assignment Expectations and Grading Criteria

Please note, it is the case with all AEP's that **your grade is primarily based on your explanations and writing, and only secondarily on the precision and correctness of your computations.** Correct computations with insufficient explanation will need to be revised and may receive an "N" grade.

AEPs are graded using the "EMRN" rubric found in the syllabus. **A grade of E or M requires all of the following to be met:**

- Task 1 must show all work in a clear, organized, and legible manner, and the GCD must be clearly indicated.
- Task 2 must clearly state where the prime number p was found; the prime number has to be the requested size; and the value of g must work according to the Diffie-Hellman specifications (i.e. it must have no common factors with $p-1$ other than 1). And all work has to be shown in a clear, organized, and legible manner.
- Task 1 must show all Extended Euclidean Algorithm work in a clear, organized, and legible manner. You do not need to repeat the work from the Euclidean Algorithm in task 1.

A grade of "E" is given if all of the above expectations are met, and the work is of superior quality in terms of the clarity of explanations and work, appearance of the writeup, and precision of the mathematics.

Submitting your work

AEP submissions must be typewritten and saved as either a PDF or MS Word file. No part of your submission may involve handwriting; work that is submitted that contains handwriting will be graded N and returned without feedback. This includes electronic handwritten docments, for example using a stylus and a note-taking app. To type up your work, you can use MS Word or Google Docs (both of which have equation editors for mathematical notation) or any other computer-based math typesetting tool. Just make sure you save your work as a Word document or PDF (no `.odt`, `.rtf`, or other file extensions are allowed).

When you have your work typed up, double-check it for neatness, correctness, and clarity. Then, go to Blackboard, then **Assignments**, then **AEP**, then **AEP 3**. Clicking on the text "AEP 3" will take you to a place on Blackboard where you can upload your work. All grading and revisions of labs are done entirely on Blackboard. **Do not email your work to the professor** – only Blackboard submissions are accepted.

Getting Help

Please note that according to the syllabus, for AEP's **"no interactions at all with another person or with unauthorized sources on the internet is allowed."** Violations of this rule include *any* consultation with other students or former students, including Math Center tutors; using work from another student or former student; submitting the problem set to an online help site such as Chegg or Coursehero; or asking

for help in an online forum. All such violations will be treated as academic dishonesty and will result in a grade of "N" and being banned from revising the work.

You **may** ask me (Talbert) for help on this assignment in the form of **specific mathematical or technical questions**. If I cannot answer a question because it would give too much away, I'll tell you so. **However please note: I will not "look over your work" before you submit it to give you feedback on the overall submission**; the expectations are clearly laid out above, so just follow those directions and submit your best work, and you'll be allowed to revise it if needed.

You can ask technology related questions to anyone at any time. For example if you need help with Desmos, or with figuring out how to type up your work, there are no restrictions on that. I recommend the `#tech` channel on Campuswire so that you'll reach a large audience.