CS 70 Discrete Mathematics and Probability Theory
Fall 2013 Week 3 Discussion

Modular Arithmetic

1. In class you learned how to compute $\gcd(a,b)$ using Euclid's algorithm. Now prove that for every number d such that $d|a$ and $d|b$, we must have $d|\gcd(a,b)$. Try to provide a mathematically rigorous proof.

   Solution: The intended solution is to use the extended Euclidean algorithm to find out that $\gcd(a,b) = ax+by$ for some x,y, and then the solution becomes obvious. It might be a good idea to help them reach the point

   It is perhaps a good starter question to help them understand the levels of rigor. It might be good to provide a solution using prime factorization and then discuss how we are using many hidden facts in that proof; that we are assuming every number has a prime factorization, and every divisor of a number must have exponents that are below the prime factorization's exponents.

   Paul suggests giving an intro like this:

   OK, whether this counts as a 'proof' or not just depends on how you are looking at things. Formally, a proof would be something that a machine could check, line by line, and nothing we are writing is close to that standard. Each line would be verbatim either a hypothesis or an axiom, or consist of a purely formulaic transformation of previous sentences. We don't do this here. Understanding the proofs we write in cs70 is a really hard AI problem, there is no simple algorithm. But informally, when humans read proofs they sometimes assume more or less; as the writer of the proof, you want to use a level of assumptions that is appropriate to the audience and context (this is a decision you all will have to make when turning in your HW!).

   As you change the level of rigor, there are tradeoffs: if you are more rigorous, you are less likely to accidentally make a false step, but your proof becomes longer. Every time you compress a step you save space, but by being less explicit you remove a safety check, and almost every working mathematician has at one time or another (or 100 times) compressed a "trivial" step that turned out not to be true. Most such errors can be fixed if you are more careful—you just need to change a few lines of the proof. But sometimes the problem is deeper. For example, in 1930's italy mathematicians "proved" many impressive theorems about geometry, relying on increasingly elaborate geometric intuitions that allowed them to compress proofs substantially. Unfortunately, it turned out that a number of these theorems were false, and an entire school of mathematics was eventually abandoned in part because of the problem.

   Anyway, we won't be doing anything so exotic in this class, but we do face the same tradeoffs. For example, I could write this proof this way [...], or I could be a bit more explicit [here], or I could be even more explicit [here]. When doing your HW, we are imagining a level of rigor like the one [here]. You can rely on statements that we have made in class and things that are essentially restatements of results from class. And you can rely on intuitions about what an algorithm does, say, when it is very obvious what it does. But beware of what is obvious! And now that we have moved on from induction, you can just say "We prove this claim by induction on n" and give a proof sketch, and then

you can use that as a lemma and go on to prove bigger and better things, and we'll understand that there was a working induction there that you just didn't spell out completely.

But unfortunately, almost every proof that people write relies on a lot of intuitive things which aren't just compressions of obvious steps, they are "shots in the dark" that are usually true but sometimes come back to bite you. And you shouldn't use those at all in your proofs

2. Find out whether the equation $51x = 1 \pmod{113}$ has a solution, and find one if it does. What about the equation $85x = 119 \pmod{221}$?

   Solution: For the first example they simply have to run the extended Euclidean algorithm to find the inverse of 51 mod 113 which is 82.

   In the second example 85 and 221 are not co-prime but have the common factor 17. Running the extended Euclidean algorithm gives us x such that $85x = 17 \pmod{221}$ (one solution is $x = 8$ but there are others). Now all we need to see is that 119 is divisible by 17, so we can just multiply x by $119/17 = 7$ to get for example 56 as a solution.

3. Prove that if $x = y \pmod 3$ and $x = y \pmod 5$ then $x = y \pmod{15}$.

   Solution: Again they should first transform these into $7|x-y$ and $5|x-y$ and then concluded that $35|x-y$. This is perhaps another point for discussing the level of rigor needed. While it might seem clear to them that when 3 and 5 divide a number, 15 also divides that number, under the hood they are appealing to hidden facts. In particular here they might be appealing to the unique prime factorization of integers. It is probably a good idea to show them how using the uniqueness of prime factorization one can prove this rigorously: if 3 or 5 did not appear in the prime factorization of $x-y$ then one could write another prime factorization of $x-y$ by adding 3 to the prime factorization of $(x-y)/3$ or 5 to the prime factorization of $(x-y)/5$. Because of the uniqueness 3 and 5 should both appear in the prime factorization and so 15 divides $x-y$. If we want to be more careful we also have to consider the case where $x-y$ is negative and also prove that 3 and 5 are both prime!

4. Prove the following:

   - If $x = y \pmod n$ and $z = w \pmod n$ then $xz = yw \pmod n$.
   - If x has two modular inverses y and z mod n, then $y = z \pmod n$.

   Solution: For the first one simply write $x = y + kn$ and $z = w + k'n$ and then multiply things out and factor n.

   For the second one, one needs to either appeal to the fact that when $n|x(y-z)$ and $\gcd(n,x) = 1$ then $n|y-z$. Another possible way is to note that xa when a ranges from 0 to $n-1$ covers all of $0,\ldots,n-1$. This is because for any number c we have $x(cy) = c \pmod n$. Therefore since $a \to xa$ is onto, it must be a bijection, which means that it is one-to-one. Again this is a fine point for discussing the level of rigor needed to prove these things. A nice exercise is to prove the first steps of the second proof (i.e. introduce the function $a \to xa$ and note that it is onto) and then ask students to complete the proof.

5. Prove that Fibonacci numbers mod n become periodic. Find an upper-bound on the length of the period as a function of n. Next, prove that if $a|b$ then $F_a|F_b$. **Challenge:** first prove that $\gcd(F_a,F_b) = \gcd(F_{(b \mod a)},F_a)$, and then prove that $\gcd(F_a,F_b) = F_{\gcd(a,b)}$.

   Solution: Just apply the pigeon-hole principle to the pairs of remainders of adjacent fibonacci numbers to get an upper-bound of n^2.

   For the second part one can observe that if $n = F_a$ then the sequence F_i starts with the pair $(0,1)$ until it reaches $(0,F_{a+1})$. Starting the Fibonacci sequence with $(0,x)$ is the same as starting it with $(0,1)$

and multiplying every term by x. Therefore the second period mod F_a is simply the first period times F_{a+1}. So at F_{2a} we again reach 0 and the argument continues.

In the challenge part, it is enough to observe that in the previous argument, every period is simply F_{a+1} times the previous period. But F_{a+1} is co-prime with F_a, therefore we never introduce any additional common factors with F_a when we go to the next period. So this means that $\gcd(F_b, F_a) = \gcd(F_{(b \mod a)}, F_a)$. Now we can apply the Euclidean algorithm to the indices, and see that $\gcd(F_a, F_b) = \gcd(F_{\gcd(a,b)}, F_0) = \gcd(F_{\gcd(a,b)}, 0) = F_{\gcd(a,b)}$.