Group Work on Primality Testing

Learning Objectives:

1. To learn about prime numbers (in order to be able to study cryptography later).
2. To understand what an algorithm is, and what it means to "analyze" an algorithm.
3. To improve creative problem solving skills.
4. To develop an appreciation for current computer science research.
5. To get to know each other and have fun.

Some terminology:

- An *algorithm* is a recipe for solving a problem – a sequence of unambiguous instructions that, when followed, are guaranteed to solve the problem in a finite amount of time for any reasonable input. Algorithms are of central importance in computer science partly because if we are interested in writing a computer program to solve a particular problem we must first develop an algorithm.
- To *analyze an algorithm* is to determine how the time and space requirements for the algorithm depend on the size of the input.
- An *integer* is any whole number. It can be positive, negative or zero.
- A *prime number* is any integer greater than 1 that is divisible only by 1 and itself.

The problem: Primality Testing

- **Input:** An integer n greater than 1.
- **Output:**
  - "Yes" if n is prime.
  - "No" if not.

Part I: Developing an algorithm.

1. Develop an algorithm to decide if a given number is prime. What I mean by that is the following. Imagine that your 10 year old sister (who knows how to read and how to divide, for example) is taking a quiz tomorrow on prime numbers. The teacher is going to give her a positive integer and she is going to have to decide whether or not it is prime. She is allowed to bring in a piece of paper with notes. You want to write down a list of instructions for her. You want your instructions to be clear and precise enough so that she would always get the right answer by following your directions to the letter. When your group has come up with an algorithm, please let me know so I can come talk to you about it. It's ok to call me over before you've written it down if you're wanting to test out some ideas.

Part II: Analyzing the speed of the algorithm.

1. How many divisions will be performed by your algorithm if the input is $n = 28$? What if $n = 29$? Count them carefully.

2. If n is not prime, how many divisions will be performed? This is what we call an "open-ended" question: there are many ways to answer it. Be as thorough and insightful as you can.

3. If n is prime, how many divisions will be performed? Here your answer should be a formula in terms of n. When you are satisfied with your answer to this question, let me know so I can come talk to you. While you are waiting for me, have a look at the next section.

Things to explore at home: One of our goals this semester is to expose you to the world of research in theoretical computer science. What problems are researchers trying to solve? Why are they interested in those particular problems? Who does such research, and who pays them to do it? How does one learn to be a researcher in theoretical computer science? What is the role of the Ph. D. advisor in learning how to do research? Do researchers tend to work alone or rather in collaboration? To start to get a feel for this world, I invite you to follow the leads below. The goal here is simply to give you a peek into the world of research, and to introduce you to some fun tools in case you want to poke around some more.

1. Read the New York Times article that I've printed for you about an algorithm discovered in 2002 for deciding if a number is prime.

2. In the article, the new algorithm is credited to three researchers. One of these ended up being the Ph. D. advisor of the other two. Find the Mathematics Genealogy Project on the internet. Use it to figure out who was the Ph. D. advisor.

3. Through the Hofstra Library web page, find the resource called MathSciNet. Use it to find the article that the three researchers above published in 2004 in the Annals of Mathematics. (The Annals is, some say, the most prestigious research journal in mathematics in the world.)

4. Find out what is meant by the "collaboration distance" between two researchers. Use the search tools in MathSciNet to find the collaboration distance between one of the authors of the paper above and another mathematician or theoretical computer scientist. You can use me, or you can use Erdos (who is that?) or you can use one of your other math or computer science professors here at Hofstra.

5. We will begin the next class with a short free write about what you have discovered. Anything interesting? puzzling? confusing? I will collect these at the start of class, just as a way to start to get to know you.