CMSC 330 Algorithms: Design & Practice
Week of March 16: Problems, Solutions, Algorithms, Programs
Submission due on March 25th by 5p (Eastern Time)

This week's exercise is designed to get you out of the COVID-19 mode and think about computational problems, solutions, algorithms, and programs. At least for just a little bit. It will also serve as review of some of the fundamental ideas of computing that we have so far learned in this course. It is a simple exercise and will require you to have the following: pen/pencil, paper, a calculator, and access to a computer for writing programs in Python and Java. The program(s) will be no longer than a couple dozen lines of code. I promise. Ready?

Say we want to compute the square root of the number 2 (i.e. $\sqrt{2}$). This is essentially a numerical calculation task. While there are many methods for calculating the square root of a number[1], we will use one commonly called the Babylonian Method (circa 1000 BCE). It is also sometimes referred to as Heron's Method which dates back to AD 60.

To compute the square root $x = \sqrt{a}$ do the following:

1. Start with some guess $x_1 > 0$
2. Compute a sequence of guesses $x_1, x_2, \dots, x_n$ using the equation:

$$x_{n+1} = \frac{1}{2}(x_n + \frac{a}{x_n})$$

   until the numbers produced converge.

Do This: Put down this book and get a piece of scratch paper and pencil/pen. And, a calculator. You will need one. Compute the square root of 2 (i.e. $\sqrt{2}$) using the above formula. Start with $x_1 = 1$. Next, write down your answer for values of x_1, x_2, x_3, etc. I'll get you started. You fill out the remaining values (you might need more values than shown below, continue until numbers converge):

$x_1 = 1$
$x_2 = 1.5$
$x_3 = 1.416666667$

$x_4 =$ ____________________

$x_5 =$ ____________________

$x_6 =$ ____________________

$x_7 =$ ____________________

$x_8 =$ ____________________

$x_9 =$ ____________________

$x_{10} =$ ____________________

$x_{11} =$ ____________________

Computed value of $\sqrt{2}$ is: ____________________

Were you also able to compute $\sqrt{2}$ correctly?

[1] Look up the Wikipedia page for computing square roots. You will find many different ways, including the Babylonian/Heron method we use in this exercise.

On my calculator, I needed seven steps to get $\sqrt{2} = 1.414213562$. I was able to get values with a precision of 9-digits after the decimal. Yours may be different in the number of digits of accuracy. I stopped after the same value was produced twice in a row. That is, the sequence converged.

Well, congratulations! You were just a computer. And you did some computing!

The very small set of computations you did above even with the help of a calculator was a somewhat tedious exercise. Right? I agree. But, this is where computers come in. What you have seen so far is a concrete example of a **problem**: compute the square root of a number; and a possible **solution**: how to compute it.

What is programming?
P*rogramming is how you tell a computer what to do*. It is similar to how I told you above about how to go about computing the square root of a number. Try telling a friend and see how it goes. While you would not want to then say that you programmed your friend to compute square roots, when you do this to a computer, you are *programming* the computer.

To illustrate another important point, let us repeat the set of instructions to compute the square roots:

To compute the square roots $x = \sqrt{a}$ do the following:

1. Start with some guess $x_1 > 0$
2. Compute a sequence of guesses $x_1, x_2, \ldots, x_n$ using the equation:

$$x_{n+1} = \frac{1}{2}(x_n + \frac{a}{x_n})$$

   until the numbers produced converge.

While these were sufficient for us to follow and compute square roots, for a computer there are still many vagaries in the set of instructions above: What is "some guess"? what does it mean to say "until numbers converge"?

Let us try and eliminate these vagaries. First, look at step 1:

Start with some guess $x_1 > 0$

How do we know what to start with? It is a guess after all and as long as it is greater than zero, the method works. So, you could start with 1 (as we did earlier), or you could start with another number, say 10, or 100, etc. No matter where you start you will be able to converge to the correct square root of desired accuracy. Try computing $\sqrt{2}$ starting with $x_1 = 5$. In a few steps you will arrive at the correct value of $\sqrt{2}$. It will just take a few more repetitions of step 2 for it to converge. And, that brings us to the second vagary in Step 2:

until the numbers converge

Let us, for now, assume that a sequence of numbers converges whenever adjacent numbers are the same to a desired accuracy. If you only desired an accuracy of 3-digits, say to compute $\sqrt{2}$, starting with the first guess as 1. You will see convergence after five steps: 1, 1.5, 1.416, 1.41467, 1.41422. To get an accuracy of 16-digits you will need more steps:

```
1
1.5
1.4166666666666665
```

```
1.4142156862745097
1.4142135623746899
1.414213562373095
1.414213562373095
```

It took seven steps. Essentially, the number of steps it will take to converge depends on how "bad" the initial guess is and how much accuracy you desire. Here, "bad" means how far away the first guess is from the actual square root. Say you started with 1.4 as your initial guess. The sequence of numbers will converge after only four steps. Try it.

What is an algorithm?
One more time. Back to our example of computing square roots. In order to express the steps in a formal manner, we have to be more precise. This is where *algorithms* come in. For now, let us describe an algorithm as a *set of instructions* with some specific properties: they are precise and unambiguous (i.e. no vagaries). Here is a more precise and unambiguous set of instructions:

Algorithm: How to compute $\sqrt{a}$.

```
1. To compute $\sqrt{a}$.
2. Start with $x_i = 1$. This is our initial guess.
3. Compute the next guess:
       $x_{i+1} = \frac{1}{2}(x_i + \frac{a}{x_i})$
4. If $x_{i+1} \neq x_i$
           set $x_i$ to be same as $x_{i+1}$.
           And then repeat from Step 3.
   Otherwise because $x_{i+1} = x_i$, they have converged. Therefore, $\sqrt{a} = x_{i+1}$.
```

Computer scientists will recognize the above as an algorithm for computing $\sqrt{a}$. It is precise, unambiguous, and *effective*. An algorithm is effective if its steps can be followed to produce an answer.

Algorithm: A Definition
An algorithm is a precise, unambiguous, effective set of instructions on how to solve a problem.

From Algorithm to Program
Now that we have an algorithm, we can think about programming languages and *programs*. Programs are essentially algorithms expressed in a programming language. This is where you come in.

TASK: Express the algorithm described above in Python/Java/C so that you can compute the square root of any positive number on a computer. Write a function **sqRoot()** that uses the algorithm above and then use it as shown below:

```
for n ← 1 to 10 do
   print the value of n, and sqRoot(n)
```

Implement a complete program to do this.

WHAT TO SUBMIT
In PDF format, a print out of your program and its output as required above. For extra credit, write the program in another programming language of your choosing.

WEEKLY JOURNAL: Don't forget to write your journal entry for this week.
This week's topic: Algorithms in my life