CCPS616 Lab 2 – Divide and Conquer

Preamble

In this lab you will write a C++ program that performs large integer multiplication. Your implementation must be from the ground up without linking any external libraries. Your lab should be completed in a single cpp file called **lab2.cpp**, and should compile at the command line very simply as follows:

```
g++ -o lab2 lab2.cpp
```

Please do not submit labs that don't even compile. We're past that. I'm not going to fix your syntax errors. If you can't make any progress on the lab whatsoever, submit a simple program that prints a message saying as much. For this, I will award you a single mark.

Lab Description

First, implement the so called "Schoolbook" O(n^2) method that goes digit by digit (seen on or around slides 63/64 in lecture 4). Your function should be called `mult4()`, and it will accept two strings (either `std::string` or `char*`, up to you) of decimal digits as input. Keep in mind that the whole point of implementing an algorithm like this is for use on integers too large to be represented by existing integer data types (pretend Python's integers don't exist). Thus, we use character arrays to represent our large integers precisely.

Second, implement Karatsuba's algorithm that performs multiplication in O($n^{1.585}$). Call this function `mult3a()`. Its input arguments should take the same form as `mult4()`. This implementation should recurse all the way down to n = 1.

Third, implement a modified version of Karatsuba's algorithm that recurses only so long as the result of a multiplication would overflow a 32-bit integer. This should be a very simple modification to the version above. Call this function `mult3b()`.

In all cases, you may assume the input integer strings are positive.

Testing

To test your implementations, in your `main()` function, you will randomly generate strings of *k* digits, where *k* goes from 10 to N. The value N is the point at which you get fed up waiting for your computer to complete the multiplication (at minimum, find N such that the calculation takes a couple seconds). Once you've got a testing loop properly implemented, progressively increasing N should be a very simple matter.

Perform each test (k=10 up to k=N) on both of your multiplication functions and verify that the advertised complexity of each holds true. Average each test over 10 trials.

Results

Your program should write out test results to a plain text file. That is, running your program should produce a plain text file containing the experimental results described previously. Thus, your source code file must contain your implementations for `mult4()`, `mult3a()`, and `mult3b()`, as well as all the test code that will produce the output file. Name your output file CCPS616_Lab2_ LastName1_LastName2.txt, where my name is replaced with yours. Format your output file in a way that's easy for me to read. An example is below. All quantities in the table will of course be filled with real data produced by your program:

```
CCPS616 - Lab 2 – Partner1, Partner2
Each result is the average of 10 trials

k =     mult4()    mult3a()   mult3b()
10      XXXms      XXXms      XXXms
x2      XXXms      XXXms      XXXms
x3      XXXms      XXXms      XXXms
x4      XXXms      XXXms      XXXms
x5      XXXms      XXXms      XXXms
x6      XXXms      XXXms      XXXms
x7      XXXms      XXXms      XXXms
N       XXXms      XXXms      XXXms
```

As you can see, you need not write out the result for every value of k from 10 to N. Choose an appropriate step such that you print lines for 8-10 values of k between 10 and N.

Submission

Lab 2 may be done in **pairs**! Submit your source file (**lab2.cpp**) and a sample output file (**CCPS616_Lab2_LastName1_LastName2.txt**) produced by your program to D2L.

"You do one half of the assignment I'll do the other half and then we'll join them together"