Linguistics

Parlez-vous Deutsch, amigo?

During spring break of 2015 at BCIT, I went and redid one of my assignments a few times in a few different languages: C++, Haskell, JavaScript, PowerShell, and Python. C++ was the course I was studying and I had done a fair bit of python before. JavaScript I’d touched a couple times, but Haskell and PowerShell were completely unfamiliar to me.

The assignment I chose was a small lab assignment to calculate which set of 7 numbers had the most permutations that were prime numbers.

Considering 3-digit numbers, as a simpler example, gives a maximum of 4 permutations. There are 3 different solutions tied with 4 permutations.

* 149, 419, 491, 941
* 179, 197, 719, 971
* 379, 397, 739, 937

3-digit numbers are easy to work with, but when you get up to 7-digit numbers it becomes more important how you choose to solve it. I really enjoyed solving this one. So much so that I resolved it in a few different languages. The code is the purpose of this article so I’m about to give away how to solve it. Stop now if you want to do it yourself.

The general solution I used is

1. Generate all the prime numbers in the range you need: e.g. [1000000:9999999]
2. Write a function that will take a number in that same range, and turn it into a key based on the digits it contains, so that 1045, 1405, 5041, and 4501 all give the same key
3. Group all the prime numbers that have the same key, or just keep the largest group if you don’t care about ties.
4. Report the largest group(s).

One other note: Do not take the results below as a benchmark for a language speed comparison. I ran different code in each of the languages, some of which I had no knowledge of prior to starting this project. Haskell was run through a virtual machine, C++ through Cygwin, JavaScript through chromium.

# C++

In C++ the code was quite straightforward. I had been writing in C++ recently and was able to translate the algorithm into code in less than an hour.

I was able to take advantage of the bitset type to both shrink my volume of data and probably make it faster to manipulate. Since strings are mutable in C++, most people in my class converted the number to a string and then ran quicksort on it. I felt like doing something different so I wrote a function that would generate a key using only numeric transformations. My solution would work up to 9-digit numbers, although it could be generalized.

<code>

Running the code in c++ took about 2 seconds on my laptop for a 7-digit number. No other language that followed came close to meeting that speed.

# Haskell

Haskell was an interesting language to investigate. This was my first time installing it and coding in it. It’s a functional language, so it feels like all I’m doing is setting up the relationships between variables, and then, only at the very end, I’m requesting a value to print. Once that value is requested, Haskell works backwards to resolve all the dependencies and calculate the necessary intermediate values to give me what I wanted to print out.

Another thing that I found really interesting was that all the functions were pure. There were no global variables in the background being manipulated, nor any object state. It was pure functions with no side effects. It practically eliminates runtime errors, and it feels good too.

<code>

Another cool thing about Haskell is that it comes with a fancy profiler. I ran it on my program and it gave me this as the result:

<profiler results>

All in all it was a really insightful exploration into other programming languages.

# JavaScript

JavaScript is written to do the same thing as C++, but instead of using a bitset, it generates a list of primes. That means each prime is a Number and not a single bit. What JavaScript does bring to the table to is integration with HTML, and the ability to easily make the code visually interactive, with a GUI.

<Code>

<active implementation>

It takes several seconds to calculate the answer for 7 digits, considerably slower than the C++ version, but not as much slower as I expected. I was pleasantly surprised!

# PowerShell

It took a little bit to wrap my head around PowerShell being built for piping commands into commands, but it is actually really cool, and powerful. I had a working product in PowerShell faster than in any other language.

<code>

I was not patient enough to see how long it would take to calculate the 7-digit number, but 6 digits took about 4 minutes. Getting the primes was the most expensive part at 3 minutes, and grouping them took a minute more. I read a little about how to profile and optimize, but did not get far. I read that using things like [Math]::Pow() load up C# libraries, but avoiding C# did not appear to make a difference to computation times. Interesting experiment, and PowerShell is kind of fun, but I don’t have need for its strengths right now.

# Python

Python is a favorite of mine, so naturally I wrote the program up in counter here too.

<code>

Python 2.7.6 and 3.3.3 both took about 10 seconds to calculate the result.