Kelly Sovacool

CS315-001
21 Sept. 2017

Homework 2

Design

I chose to use an object-oriented design in the Python 3 programming language to implement the algorithms. I created a class called BinaryNumber which inherits from Python's built-in list type. It stores binary numbers as a list of integers, with the leftmost digit being the *least* significant digit. To solve problems 1, 2, and 3, I overloaded the addition, subtraction, and multiplication operators within the BinaryNumber class using the “school algorithms”. Each function operates on copies of the original numbers, which are equalized in length by adding zeroes to the most significant end of the smaller number. I also defined a handful of other methods to avoid code-redundancy.

The main function takes a single command-line argument for the input filename. The input file must be a tab-delimited text file with two binary numbers per line (x is the first number, y is the second). If no filename is supplied, instead the binary numbers can be entered manually. The binary numbers entered must have the leftmost digit being the *most* significant digit; these are reversed and fed into new instances of the BinaryNumber class. The main function then calls another module-level function that performs the arithmetic for all three problems and prints the results to standard output. If an input filename is given, a second command-line argument can also be supplied. If this second argument is equal to “human-readable”, objects of the BinaryNumber class are printed as strings of integers with the leftmost digit being the *most* significant digit. This aids in readability of the results since this is how numbers are traditionally represented. If the second argument is not specified or is not equal to “human-readable”, the binary numbers are printed as lists with the leftmost digit being the *least* significant, which is the same way they are represented in memory.

In the addition algorithm, the digits of x and y are iterated over concurrently from the least significant end. The new digit is calculated as the x digit plus the y digit plus the carry digit (which is initially zero) in decimal arithmetic. If the new digit is three, then the carry digit becomes one and the new digit becomes one. If the new digit is two, then the carry digit becomes one and the new digit becomes zero. Otherwise, the carry digit becomes zero and the new digit as calculated is used, because it is either zero or one. The new digit is appended to the result before the next iteration. After all of the iterations are complete, if a carry digit remains, it is also appended to the result.

In the subtraction algorithm, the digits of x and y are iterated over concurrently. The new digit is calculated as the x digit minus the y digit in decimal arithmetic. If the new digit is negative one, borrowing must occur. To borrow a one-digit, digits of x are iterated over from the current digit toward the significant end until a one-digit is found, changing any zeroes along the way to ones. The digit which is borrowed from becomes zero, and the new digit becomes one. The new digit is then appended to the result.

In the multiplication algorithm, the digits of y are iterated over. Within this loop, an intermediate binary number is initialized with a number of zeros to offset it by the current iteration number. For example, if the current y-digit is the second digit, a single zero is prepended to the intermediate binary number. Then, digits of x are iterated over, and the result of the x-digit times the y-digit is append to the current intermediate number. After iterating over the digits of x, the current intermediate number is added to the result. The outer loop then moves on to the next digit of y, until all have been iterated over.

Results

I created a tab-delimited text file (input.tsv) with the following x-y pairs of binary numbers:

```
111000110	101101111
11100011001	10110
1110001110000111000111	101010101010101
1110	101
1	0
1	1
0	0
10010	101
```

The first three lines contain the test cases given in the assignment instructions. The other lines contain additional test cases that I found useful for debugging. Here are the results for all three arithmetic operations on each pair of numbers, represented with the leftmost digit being the *most* significant:

x	y	x + y	x - y	x * y
111000110	101101111	1100110101	1010111	101000101011011010
11100011001	10110	11100101111	11100101111	1001110000100110
1110001110000111000000	101010101010101	1110011000111000011100	1110000011100011110010	1001011110110011100011011010000010011
1110	101	10011	1001	1000110
1	0	1	1	0
1	1	10	0	1
0	0	0	0	0
10010	101	10111	1101	1011010

Instructions

This program is written in Python 3. Either ensure that the hash-bang (#!) on the first line contains the proper path to Python 3 on your machine, or explicitly call the program using Python 3 on the command line. Also, give the user permission to execute the file. Usage examples:

```
python3 ./homework2.py
python3 ./homework2.py input_filename
python3 ./homework2.py input_filename human-readable
```