2WF90: Software Assignment Integer Arithmetic

Andreas Hülsing, Benne de Weger

September 2019

Contents

1 Introduction

The programming assignment for the first part of the course consists of writing code ('from scratch') for arithmetic with large integers, including modular arithmetic. This is mainly meant as preparation for the second programming assignment on polynomial and finite field arithmetic, which is described in a separate document.

1.1 Role in assessment

This programming assignment contributes $7\frac{1}{2}\%$ to your 2WF90 grade.

1.2 General guidelines for the programming assignments

There is a separate document with "General Guidelines", that also apply for this assignment.

2 Software Assignment on Integer Arithmetic

Write code 'from scratch' for basic arithmetic with large integers using representations with as radix a configurable integer $b \leq 16$). 'From scratch' means that you are not allowed to use large-integer libraries built into mainstream computer languages such as Java or Python,

but your code should use from such a language only elementary arithmetic operations on small integers of at most 32 bits. Your code should be able to do the following:

- formats, input, output:
  - deal with the standard radix b representation of integers of large word length (at least several hundreds); word length should be variable;
  - compute directly with the radix b, i.e. conversion back and forth to another radix is not allowed;
  - inputs and outputs of arithmetic operations should be in this standard format; your code should assume that the input file does have the required format, i.e. do not spend any time on writing error handling code for input that does not follow the specificed format;
  - deal with positive and negative integers, and with zero;
  - read input from a file called `input.txt` with a specific format (an example input file called `example.txt` is provided, this example file also specifies the format);
  - write output to a file called `output.txt` (with the same specified format);
- Integer arithmetic:
  - the following integer arithmetic operations should be supported:
    - addition, subtraction;
    - multiplication by the primary school method;
    - multiplication by the method of Karatsuba (dividing up long numbers in two parts, with recursion);
    - Euclid's Extended Algorithm for two large positive integers;
  - for the multiplication operations, the software should keep track of the number of elementary additions/subtractions it does, and the number of elementary multiplications, and to be able to show those operation counts in the output.
- Modular arithmetic:
  - modular arithmetic with a modulus m of large word length: the following modular arithmetic operations should be supported:
    - modular reduction (i.e. "long" division by m with remainder, where only the remainder is relevant output; see Algorithm 1.4);
    - addition, subtraction (mod m);
    - multiplication (mod m);
    - modular inversion for a large modulus m.

Present some examples of different word sizes and radices, and compare elementary operation counts of primary school and Karatsuba methods. Draw a conclusion based on the theory. Input and output files are not to be submitted.

The input/output format specification and test values are given in the example input file `example.txt`, see the Appendix.

Appendix

Here is the contents of the input/output file `example.txt` for the Integer Arithmetic assignment, containing the specification of the input and output format. The file itself is provided on Canvas.

```
#############################################################################################
#
# Example input / output file your program should be able to process
#
# Explanation:
#
# empty line:        separates two computations
#
# line starts with
# #:                 comment, to be ignored by the program
# [radix]:           the radix used in the next computation
# [add]:             apply addition to the following two numbers
# [subtract]:        apply subtraction to the following two numbers
# [multiply]:        apply primary school multiplication to the following two numbers
# [karatsuba]:       apply Karatsuba multiplication to the following two numbers
# [x]:               first number to which the operation is to be applied
# [y]:               second number to which the operation is to be applied
# [answer]:          to be ignored if the file is input, may even be absent in input files;
#                    the correct answer of the preceding computation if the file is output
# [count-add]:       to be ignored if the file is input;
#                    count of the elementary additions and subtractions if the file is output
# [count-mul]:       to be ignored if the file is input;
#                    count of the elementary multiplications if the file is output
# [m]:               modulus to which the operation is to be applied
#                    (absent if integer arithmetic is done)
# [reduce]:          modular reduction
# [inverse]:         modular inversion
# [euclid]:          Euclid's Extended Algorithm
# [answ-a]:          to be ignored if the file is input;
#                    a such that gcd(x,y) = a x + b y if the file is output
# [answ-b]:          to be ignored if the file is input;
#                    b such that gcd(x,y) = a x + b y if the file is output
# [answ-d]:          to be ignored if the file is input;
#                    gcd(x,y) if the file is output
#
# the [multiply] and [karatsuba] examples below use the same values for [x] and [y]
# (for the same [radix]); the answers your program produces should be equal too;
# also below the correct answers are shown (but not the operation counts)
#
# all spaces and tabs should be ignored
#
#############################################################################################

# integer arithmetic, radix 2

[radix]  2
```

```
[add]
[x]      1100110001011011110011101011100100000001010101111101001111010101111100110110001100010100001
[y]      1001011010110100010111100101110011001010001010001001010100111001010011000101001110000101000
[answer] 1011000110001000000101101000101011100101110000000011010010000111100111111101101101001100100

[radix]  2
[subtract]
[x]      1111001001111000100100000101010011011111111101100110100101110101101100100110001010011101011
[y]      1001011010100110111010101000101110010010111011100100111101010001011000110110100111011000110
[answer] 1011011110100011010010111001001010011010000100000011010001001000100111011111000110001001010

[radix]  2
[multiply]
[x]      1111001110011011011101001100010101111000011010100100001101011101111101101110010001111010001
[y]      1110110000001110100000101100010100101010110001101100011011101000101001001001000100001111010
[answer] 1110000010100001000111101001011011111110011100000101011110110010010000110000001001101111100
[count-add] ...
[count-mul] ...

[radix]  2
[karatsuba]
[x]      1111001110011011011101001100010101111000011010100100001101011101111101101110010001111010001
[y]      1110110000001110100000101100010100101010110001101100011011101000101001001001000100001111010
[answer] 1110000010100001000111101001011011111110011100000101011110110010010000110000001001101111100
[count-add] ...
[count-mul] ...

# integer arithmetic, radix 3

[radix]  3
[add]
[x]      1021002222220100111012120000201102021221
[y]      1012200010211212111211101000010010220000
[answer] 2110210010202020000000221000211120011221

[radix]  3
[subtract]
[x]      2021112002201010101201001221200211210001
[y]      1211222222202101122110112222212210012122
[answer] 1021120022212012020201112212110011201 02

[radix]  3
[multiply]
[x]      2010220211120222202210012012101022110011
[y]      -2102021102022210001211210202210112011002
[answer] -1201200020101002210011011012121122112222120010101012101122110100122011021111111022
[count-add] ...
[count-mul] ...

[radix]  3
[karatsuba]
```

```
[x]      201022021112022220221001201210102211001 1
[y]      -21020211020222100012112102022101120110002
[answer] -120120002010100221001101101212112211222120010101012101122110100122011021111110022
[count-add] ...
[count-mul] ...

# integer arithmetic, radix 16

[radix]  16
[add]
[x]      df9e76d113895821c567
[y]      6a20b188675ab39e17a2
[answer] 149bf28597ae40bbfdd09

[radix]  16
[subtract]
[x]      36d2b5154ab14bfabbf2
[y]      f9d1495cfafe396ae4b1
[answer] -c2fe9447b04ced7028bf

[radix]  16
[multiply]
[x]      -eed50d6aa53e51691add
[y]      -f9027b863f654daae6a8
[answer] e84f8af471ab1bb45d20f1a95313171b2ade2f08
[count-add] ...
[count-mul] ...

[radix]  16
[karatsuba]
[x]      -eed50d6aa53e51691add
[y]      -f9027b863f654daae6a8
[answer] e84f8af471ab1bb45d20f1a95313171b2ade2f08
[count-add] ...
[count-mul] ...

# modular arithmetic, radix 16

[radix]  16
[reduce]
[x]      ffbd238907b7d47c8f011379ad54173b9502beb1
[m]      c7eb8a91fbad0d1c1f03
[answer] c0808380322a6abc359a

[radix]  16
[add]
[x]      54311bd480c5d7f89db4
[y]      96389ae5100438574eaf
[m]      c7eb8a91fbad0d1c1f03
[answer] 227e2c27951d0333cd60
```

```
[radix]  16
[subtract]
[x]      62f73b5b5c02ab69e6f5
[y]      7a3e5237d2111e1d46fd
[m]      c7eb8a91fbad0d1c1f03
[answer] b0a473b5859e9a68befb

[radix]  16
[multiply]
[x]      44105f31659258bdf082
[y]      86347b5906a96ca11cc2
[m]      c7eb8a91fbad0d1c1f03
[answer] 9c1b2bab5c0ca148e260

[radix]  16
[euclid]
[x]      5896363941d32eccd5c
[y]      c7eb8a91fbad0d1c1f03
[answ-d] 1
[answ-a] 96998fcd4268440ce6a5
[answ-b] -42bb80ba0313b9aff19

[radix]  16
[inverse]
[x]      5896363941d32eccd5c
[m]      c7eb8a91fbad0d1c1f03
[answer] 96998fcd4268440ce6a5

[radix]  16
[inverse]
[x]      b99ab2815ee4b4a5f842
[m]      c7eb8a91fbad0d1c1f03
[answer] inverse does not exist
```