CS 6301.002

Implementation of advanced data structures and algorithms

Long Project 1: Integer arithmetic with arbitrarily large numbers

G-07: Salil Kansal, Twinkle Sharma, Sujit Sajja

Report

Methods used:

* Addition:

Adding digit by digit and forwarding the carry for positive numbers. For negative numbers calling the subtraction method based on logic.

* Subtraction:

Subtracting digit by digit and forwarding the borrow for positive numbers. For negative numbers calling the add function based on the logic.

* Multiplication:

Finding the product using divide and conquer approach which uses three recursive calls to the same function. The base case is when multiplying with a single digit. Then we are directly adding that number to itself n times.

* Division:

Finding the Quotient by binary search for a number between 1 and divisor. Multiplying it and comparing it with the desired value.

* Mod:

For this we are calling the division method, finding the quotient, calling the product method, finding the nearest multiple and then subtracting it with our number

* Power:

Multiplying two number raised to the same power which reduces half of the computations when compared to multiplying the number n times.

* Square root:

For this method we are finding the value by binary search starting in the range 1 to number/2. Each time we guess a value we find its square and compare it with given number. If it is less we increase our guess value and if it’s more then we decrease our guess value.

Problems faced:

* Handling the negative numbers during addition and subtraction like A – (-B) = A + B;

Initial approach was to remember the sign of result, change the sign of number and perform the operation. The problem with this approach was using the same variable for different operation. As we are changing the sign it will reflect other operations as well. Solution to this is to change back the symbol after performing the operation.

* Converting from one base to the other.
* For division operation we need to guess the value of Quotient by binary search which needs the operation of divide by 2. It was a deadlock. So we had to implement another function which only finds the middle element given two big numbers.

Design Descisions:

* For implementing negative numbers we have a Boolean flag negative. First we thought of keeping the first bit as1 for negative and 0 for positive. But it would cause lots of problem in the iterator. Thus, we stayed with keeping a flag only.

Test results:

* All input files provided are giving the required output. It is working for all bases between 2 and 10,000. All functions are implemented and Level 1,2,3 are fully working.

References:

StackOverflow

Wikipedia