**Implementation of Advanced Data Structures and Algorithms**

**Long Project 1**

***Group Members***

*Giridara Varma Elumalai*

*Srikanth Kannan*

*Praveen Erode Murugesan*

**Problem Statement:**

The aim is to implement big number library that enables to perform various arithmetic operations (add, subtract, multiply, divide etc) on numbers that will not fit into regular data type boundaries.

**Approach:**

The number is stored as a list of digits (words in the case of bases greater than 10) in the reverse order. In order to make the computation faster for very large numbers, use of different base is facilitated. The sign of the number is also considered for the implementation of arithmetic operations.

**Implementation:**

The digits in the number are maintained as an Arraylist<Long>. The number can be represented in any base. The information about the base and the sign of the number is maintained as a separate information in the number

The various arithmetic operations are implemented using different methodologies as follows

**1. Add:**

The basic naïve algorithm to sum two numbers is being implemented.

**2. Subtract:**

The difference between two numbers is found in different ways depending on their performance on empirical observations.

* For numbers of different length,

Basic subtraction algorithm is implemented by using borrow principle as Larger No – Smaller No

* For numbers of same length,

B’s complement method is used.

**2. Multiplication:**

Multiplication forms the fundamental operation for rest of the arithmetic operations. So, Karatsuba’s multiplication algorithm (Divide and Conquer Algorithm) is implemented, which has better time complexity of O(nlog3).

**3. Division & Modulo:**

Division and Modulo operation make use of binary search and multiplication operation to find the quotient and remainder respectively.

**Intuition of Implementation:**

Quotient = Dividend/ Divisor => Quotient\* Divisor= Dividend

Values between 1 and the dividend is multiplied against the divisor to get the number equal to dividend. (This is the quotient of division). If the number is not divisible properly then there is need to find two successive numbers (n & n+1) when n \* Divisor< Dividend and n+1\*Divisor > Dividend. Here n is the quotient and Dividend- n\*Divisor gives the remainder.

**4. Power:**

Power is implemented using Divide and Conquer with the base case being the BigNumber to the power of a single Long digit.

a^b = a^(b2\*base + b1) = (a^b2)^base \* a^b1

**5. Square Root:**

Square root is also implemented in the same way as that of division using multiplication and binary search. The slight modification in the intuition is as follows

Let Number n be the input for which square root s is to be found.

s\*s <= n and (s+1) \* (s+1) >n.

Choose numbers for s in between 1 and n iteratively until the above condition holds true.

**6. Factorial:**

Factorial of a number n is implemented with the basic algorithm of repeated multiplication of numbers from n till 1.

**Running time Analysis:**

Empirical studies show that the arithmetic operation such as factorial and power operation perform slower when the number is very larger (say the number n has 100s of digits). But the implemented algorithm has performed far better on the increase of the base b in which the number is represented. The results are as follows.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input A | Input B | Function | Time taken in ms for **Base 10** | Time taken in ms for Base **Sqrt(Long\_Max\_Value)** |
| 1234567890123456789012345678901234567890 | 999 | Power | 47208 | 680 |
| 1234567890123456789012345678901234567890 |  | Squareroot | 219 | 25 |
| 2000 |  | Factorial | Too long to note | 7344 |

**Conclusion:**

The Project helped understand the ways to use different Data Structures in implementing large number arithmetic operations which are not otherwise supported by the programming language. Since the program states that the numbers are too large, it gave us a sense of how to think about Data Structures and Algorithms when dealing with very large data and also how small changes can cause significant differences in performance.