BigInt Implementation in C++

By: G-3
Deep Panchani ,BT20CSE105
Bharti Vaswani,BT20CSE119
Anurag Zalke,BT20CSE122

Subject: DPPL

Under The Guidance of Dr Milind R.Penurkar

Problem Statement

Write an implementation of BigInt(Java) in C++ which can handle integers greater than 64 bytes. Also, implement addition and subtraction functions for the same BigInt. Try to use the full capability of the underlying architecture.

Implementing multiplication and division functions is optional.

Structure of BigInt

Data Structure

Variable Name	Variable Type	Use of Variable
num	string	To store the input string.
numEle	int	To store the number of elements in dynamic array
sign	int	Sign Bit. 0 if number is +ive and 1 if -ive.
digits	int	To store number of digits in the bigint
bigArr	unsigned long long int*	Dynamic array to store and perform calculations on bigint

Functions

Constructor - bigint

Input Parameter: string bigNum
Use of Function: Initialises BigInt Object. It initialises all the variables in the object and converts the input string to a dynamic array.

StringToArray

Input Parameter: string num, int elements, int size
Return Type: unsigned long long int *
Use of Function: This function is used to convert the given string into a dynamic array. The number of elements the dynamic memory should have is stored in ***int elements.*** And the total number of digits in the string is stored in ***int size***. Each element of the array contains 17 digits. Except for the 1st element if ***size%17!=0***.

numOfDigit

Input Parameter: unsigned long long int num
Return Type: int
Use of Function: This function is used to find the number of digits in an integer recursively.

printBigInt

Return Type: string
Use of Function: This function returns the signed string of the corresponding bigint object.

operator +

Input Parameter: bigint const &num1
Return Type: bigint
Use of Function: Using operator overloading we overloaded + operator for the addition of two bigint. This function checks the sign of both the big integers and decides according to what sign the final big integer have and what operations to perform.

operator -

Input Parameter: bigint const &num1
Return Type: bigint
Use of Function: Using operator overloading we overloaded - operator for the subtraction of two bigint. This function checks the sign of both the big integers and decides according to what sign the final big integer have and what operations to perform.

add

Input Parameter: unsigned long long int *arr1,int digits1,int numEle1,unsigned long long int *arr2,int digits2,int numEle2
Return Type: string
Use of Function: This function essentially has the logic for addition. We pass 2 dynamic arrays that are to be added. Then we perform the addition of the array and loop through the elements until one of the arrays runs out of elements to add.

We are performing addition on 17 digits at a time. That is each element in the array contains 17 digits.

Then we prepend the rest of the elements to the final string.

sub

Input Parameter: unsigned long long int *arr1,int digits1,int numEle1,unsigned long long int *arr2,int digits2,int numEle2
Return Type: string
Use of Function: Same as add. This function essentially has the logic for subtraction. We pass 2 dynamic arrays that are to be subtracted. Then we perform the subtraction of the array and loop through the elements until one of the arrays runs out of elements.

We are performing subtraction on 17 digits at a time. That is each element in the array contains 17 digits.

Then we prepend the rest of the elements to the final string.

How to Use BigInt

Including bigint.hpp

You can use our bigint implementation in CPP by including our bigint.hpp C++ header file in you C++ program. Make sure you have bigint.hpp in the same folder as your C++ file.
#include "bigint.hpp"

Initialising BigInt Object

Initialising a bigint object is very easy. You just have to do:
bigint <object_name>(“<Number you want to assign>”);

For Eg.:
bigint obj1("4411891289491534861531591892302498896 1");

You can even give negative input as
bigint obj1("-1534861234234524234315918923024988961");

Addition of BigInt

As the bigint is immutable you will have to declare a new bigint to store the addition of two bigint.

To add two bigint you can use + operator as shown below:
bigint <result_object> = <object_1> + <object_2>;

For Eg.:
bigint obj1(“49820398472938498273490”);
bigint obj2(“1000000424002340”);
bigint answer = obj1 + obj2;

Subtraction of BigInt

As the bigint is immutable you will have to declare a new bigint to store the subtraction of two bigint.

To subtract two bigint you can use - operator as shown below:
bigint <result_object> = <object_1> - <object_2>;

For Eg.:
bigint obj1(“498202938498273490”);
bigint obj2(“1003289474234002340”);
bigint answer = obj1 - obj2;

Printing BigInt

We made a function to print bigint using printBigInt() function. It returns a string so you can store it in another string or initialise a new bigint or even directly print using cout.

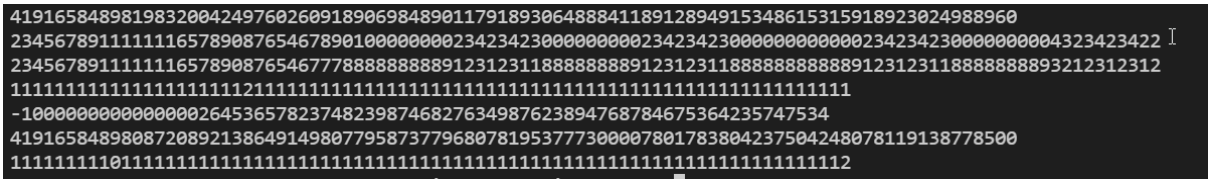
For eg:
string temp = obj.printBigInt();
bigint obj1(obj.printBigInt());
cout << obj.printBigInt();

Example

```
#include <iostream>
#include "bigint.hpp"

int main(){

    bigint
p1("4191658489819832003249760260918906984890117918930648884118912894915
34861531591892302498896 1");
    bigint
p2("9999999999999999999999999999999999999999999999999999999999999999999999
99999");
    bigint
p3("2345678911111116578908765467890000000000234234230000000000234234230
0000000000023423423000000000 4323423423");
    bigint
p4("1111111111111111111111111111111111111111111111111111111111111111111
111111111");
    bigint
p5("1000000000000000000000000000000000000000000000000000000000");
```

```
    bigint
n1("-26453657823748239874682763498762389476878467536423574753
5");
    bigint
n2("-10000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000
0000000000000000");
    bigint
n3("-99999999999999999999999999999999999999999999999999999999
999999999999999999999999999999999999999999999999999999999999
");
    bigint
n4("-11111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111
11111111111111");
    bigint
n5("-65467897856979000000000006745367485957846356475000000000000056456387
5467534685764753462635700007654656545");

    bigint a1=p1+p2;
    cout << a1.printBigInt() << endl;

    bigint a2=p2+p3;
    cout << a2.printBigInt() << endl;

    bigint a3=p3-p4;
    cout << a3.printBigInt() << endl;

    bigint a4=p4+p5;
    cout << a4.printBigInt() << endl;

    bigint a5=n1-p2;
    cout << a5.printBigInt() << endl;

    bigint a6=p1+n4;
    cout << a6.printBigInt() << endl;

    bigint a7=p4+n3;
    cout << a7.printBigInt() << endl;

    return 0;
}
```

Output



Resources and References

Class Recordings and Lecture PPTs
BigInt using Strings:
https://www.youtube.com/watch?v=SzfViApUBzw&ab_channel=CodeChef

GMP Lib Tutorial:
https://home.cs.colorado.edu/~srirams/courses/csci2824-spr14/gmpTutorial.html

Java Big Int: https://www.geeksforgeeks.org/biginteger-class-in-java/
https://www.javatpoint.com/java-biginteger
https://www.tutorialspoint.com/java/math/java_math_biginteger.htm

GFG C++ implementation of Big Int(strings):
https://www.geeksforgeeks.org/bigint-big-integers-in-c-with-example/

C++ Implementation of Big Int(Linked List):
https://www.sanfoundry.com/cpp-program-linked-list-add-two-large-numbers/