OOP 第十周作业文档

2019010175 孔瑞阳 土木 92

一、项目信息

1、功能说明

使用 vector<unsigned char>实现了一个大整数类(C_Integer)。
(按照大作业 2.21 要求进行定义，方便进行大作业时代码复用)
这个复数类支持与复数类 C_Integer 进行运算。
支持的运算包括：+、-、*、/、==、!=、>=、<=、>、<。
以及支持 C_Integer / int 的运算。

2、软件构件介绍

文件	功能介绍
random.h/cpp	随机数类
C_Integer.h/cpp	实现的大整数类
C_IntegerTest.h/cpp	大整数类的测试（包括自动/手动）
C_IntegerMain.cpp	主程序

3、测试环境

CPU	Intel(R) Core(TM)i7-9750H CPU @ 2.6Ghz 6 核 12 线程
GPU	NVIDIA GeForce RTX2070
RAM	DDR4 16G+16G
Operating System	Microsoft Windows 版本 1909
Compiler	MSVC++ 14.24

二、模型

1、大整数的表示

```
protected:
    IntegerStatus m_status;          表示数的类型
    vector<unsigned char> m_data;    表示数的绝对值
```

其中，IntegerStatus 定义如下：

```
    INTEGER_INVALID = -3, //非数
    INTEGER_NEG_INF = -2, //负无穷大
    INTEGER_NEG_VALUE = -1, //常规负数      本例会用到
    INTEGER_ZERO = 0, //0                   本例会用到
    INTEGER_POS_VALUE = 1, //常规正数      本例会用到
    INTEGER_POS_INF = 2 //正无穷大
```

2、大整数的加减法

如果两个数不同号，则加法变成减法，减法变成加法，所以只考虑同号的情况。

对于加法：按位相加，如果某一位大于 9 则进位。

对于减法：先保证绝对值大的数减绝对值小的数，再按位相减，如果某一位小于 0 则退位。计算完成后去掉最前面的若干个 0，如果变成了 0 则更改数的类型。

3、大整数的乘法

设第一个大整数的位数为 n，第二个大整数的位数为 m。

如果朴素地枚举两个大整数的每一位进行乘法运算再相加，则时间复杂度为 O(nm)，当 n.m 达到 10^5 的级别时，乘法就会变得非常慢。

所以采用快速傅里叶变换(FFT)，用复数单位元进行乘法优化，则时间复杂度为 O((n+m)log(n+m))，时间效率更高。

某个介绍 FFT 进行乘法优化的博客：

https://www.cnblogs.com/zwfymqz/p/8244902.html

4、大整数的除法

如果可以整除的话用快速数论变换(NTT)进行多项式逆元，也可以在 O(nlogn)的时间复杂度内完成(假设 n、m 同阶)。但是不能整除的时候处理麻烦。

所以使用二分法，先确定一个答案可能的范围[l,r]。

每次二分一个可能的结果，用上述的大整数的乘法的操作进行判断，乘之后的结果是否大于被除数。因为可以按照位数估算 l 和 r 的位数，所以可以保证 l<=r<=99l，二分的次数为常数。所以时间复杂度依然是 O(nlogn)。

三、单元测试

1、手动测试(integerTest)

加法

等价类	选取案例	结果
正数+正数	1919810+114514	2034324
正数+0	12345678910111213+0	12345678910111213
负数+负数	(-575687684)+(-687685451)	-1263373135
负数+0	-3141592653589+0	-3141592653589
正数+负数(正绝对值大)	8101919+(-514)	8101405
正数+负数(负绝对值大)	114+(-8101919)	-8101805

减法

等价类	选取案例	结果
正数-负数	114-(-8101919)	8102033
正数-0	12345678910111213-0	12345678910111213
0-正数	0-12345678910111213	-12345678910111213

正数-正数（左绝对值大）	1919810-114514	1805296
正数-正数（右绝对值大）	114514-1919810	-1805296
负数-正数	-8101919-114	-8102033
负数-0	-3141592653589-0	-3141592653589
0-负数	0-(-3141592653589)	3141592653589
负数-负数（左绝对值大）	(-687685451)-(-575687684)	-111997767
负数-负数（右绝对值大）	(-575687684)-(-687685451)	111997767

乘法

等价类	选取案例	结果
正数*正数	1919810*114514	219845122340
正数*0	12345678910111213*0	0
负数*负数	(-575687684)*(-687685451)	395892044606685484
负数*0	-3141592653589*0	0
正数*负数	114*(-8101919)	-923618766

除法

等价类	选取案例	结果
正数/正数	1919810/114514	16
0/正数	0/12345678910111213	0
负数/负数	(-687685451)/(-57568768)	11
0/负数	0/(-3141592653589)	0
正数/负数	8101919/(-514)	-15762
负数/正数	-8101919/114	-71069

2、自动测试(integerAutoTest)

实现了赋值构造函数 C_Integer(long long x);

采用**先将整数赋值到大整数类、再大整数类进行运算**与**整数先运算、再赋值到大整数类**进行对拍。

每次随机生成两个整数 x1,x2（正/负/0）进行对拍。

对于它们之间的所有 10 种运算(+-*//小整数各两种)全部测试一遍。如果出现错误则输出错误的数据，否则一直进行循环。（当先 x1 或 x2 随机出 0 时，不进行/x1 或/x2 的测试。）

经过 10 分钟的对拍，没有出现错误。

根据估算，10 分钟大致可以进行千亿(10^10)次计算，基本可以验证程序的正确性。