

高精度库说明

我们在下发文件中提供了题面中提到的“祖传高精度整数运算库”，文件名是 `bigint.cpp`，直接将其内容全部复制到你的代码文件的开头即可使用，需要 C++11 或以上编译。

注意该代码使用了非标准的特性，需要在 x86-64 架构的 CPU，和一个兼容 GCC 内联汇编的编译器（GCC，Clang）上编译和运行。若你的电脑不支持请使用自定义测试或者在线 IDE。

该库提供了一个结构体 `bigint`，其中可以存储任何长度的非负整数，其内部存储方式是一个 `std::vector<unsigned long long> data`，其中第 i 个元素的从低到高第 j 位是该整数的从低到高第 $64i + j$ 位（所有位数均从 0 开始记），并且存储过程中保证 `data` 的最后一个元素非零。

这种简单的存储方式意味着若有需要你可以很方便实现你自己的函数。

令 n, m 为整数的位数，也就是最高的 1 所在位 $+1$ ， $w = 64$ 为位宽。

并且提供了关于 `bigint` 的以下函数：

默认的赋值运算符和拷贝构造函数。

空构造函数 `bigint()`，会初始化为 0。

从整型的构造函数 `bigint(unsigned long long x)`，复杂度 $O(1)$ 。

从字符串的构造函数 `bigint(const std::string &s)`，注意不能有前导零，复杂度 $O(n)$ 。

到字符串的显式转换函数 `explicit operator std::string()const`，复杂度 $O(n)$ 。

到布尔的显式转换函数 `explicit operator bool()const`，会返回是否非零，复杂度 $O(1)$ 。

到整型的显式转换函数 `explicit operator unsigned long long()const`，若超出则对 2^{64} 取模，复杂度 $O(1)$ 。

`std::size_t digit()const`，会返回 n ，复杂度 $O(1)$ 。

`bool operator==(const bigint &a)const`

`bool operator!=(const bigint &a)const`

`bool operator<(const bigint &a)const`

`bool operator>(const bigint &a)const`

`bool operator<=(const bigint &a)const`

`bool operator>=(const bigint &a) const`, 整数的比较运算符, 在 n, m 相同时复杂度 $O(n/w)$, 否则 $O(1)$ 。

`bigint &operator<<= (std::size_t n)`, 左移运算符, 复杂度 $O(n/w)$ 。

`bigint &operator>>= (std::size_t n)`, 右移运算符, 复杂度 $O(n/w)$ 。

`bigint &operator+=(const bigint &a)`, 加法运算符, 复杂度 $O((n+m)/w)$ 。

`bigint &operator-=(const bigint &a)`, 减法运算符, 注意若结果是负数则行为未定义, 复杂度 $O(n/w)$ 。

`bigint &operator*=(const bigint &a)`, 乘法运算符, 复杂度 $O(nm/w^2)$ 。

`bigint &operator/=(const bigint &a)`, 除法 (整除) 运算符, 若除数是 0 行为未定义, 复杂度 $O(n(n-m)/w)$ 。

`bigint &operator%=(const bigint &a)`, 取模运算符, 若模数是 0 行为未定义, 复杂度 $O(n(n-m)/w)$ 。

`bigint operator<<(std::size_t n)`

`bigint operator>>(std::size_t n)`

`bigint &operator++()`

`bigint &operator--()`

`bigint operator++(int)`

`bigint operator--(int)`

`bigint operator+(const bigint &a) const`

`bigint operator-(const bigint &a) const`

`bigint operator*(const bigint &a) const`

`bigint operator/(const bigint &a) const`

`bigint operator%(const bigint &a) const`, 二元运算符与自增自减运算符, 与上述一元运算符实现和复杂度相同。

还有以下非成员函数:

`std::istream &operator>>(std::istream &st, bigint &a)`, 流输入。

`std::ostream &operator<<(std::ostream &st, const bigint &a)`, 流输出。