1.0 概览

列表（CSDN博客链接）

- Algorithm Notes 1 字符串
- Algorithm Notes 2 数据结构
- Algorithm Notes 3 数论
- Algorithm Notes 4 动态规划 计算几何
- Algorithm Notes 5 图论
- Algorithm Notes 6 多项式
- Algorithm Notes 7 组合数学
- Algorithm Notes 8 分治
- Algorithm Notes 9 数学相关

代码检查

上机前

- 写代码的人应当**完整阅读过题面**，清楚题目的**所有范围限制。**
- **避免**出现**没想清楚就上机**的情况。

提交前

- [Optional] 若题目的样例较弱，可以尝试构造一些 corner cases，力求测试样例**覆盖所有的分支情况。**
- **逐一**检查**所有变量**的**类型**、**数组大小**、**取模/模数**、**多测清空**，以及**空间限制**、**是否存在无解**、**是否关闭了同步流**、**读入**是否超出 `int` 范围。
- **确保**所有函数都写了**返回值。**
- 大致扫描一遍代码整体，不要出现一些**明显的符号或引用错误。**
- 一旦出现错误，对代码修改后，应**重新过一遍上述流程。**

Rejected 后

- 依情况顺序执行以下步骤：
  - **重新认真地读一遍题目。**
  - 肉眼查错/小黄鸭调试法。
  - 构造小样例测试，极限数据测试，卡常。
  - （基本不要使用）对拍。

缺省源

```cpp
#include <bits/stdc++.h>

template <class T>
inline void read(T &res)
{
    char ch; bool flag = false; res = 0;
    while (ch = getchar(), !isdigit(ch) && ch != '-');
    ch == '-' ? flag = true : res = ch ^ 48;
    while (ch = getchar(), isdigit(ch))
        res = res * 10 + ch - 48;
    flag ? res = -res : 0;
}

template <class T>
inline void nonnegative_put(T x)
{
    if (x > 9)
        nonnegative_put(x / 10);
    putchar(x % 10 + 48);
}

template <class T>
inline void put(T x)
{
    if (x < 0)
        x = -x, putchar('-');
    nonnegative_put(x);
}

template <class T>
inline void CkMin(T &x, T y) {x > y ? x = y : 0;}
template <class T>
inline void CkMax(T &x, T y) {x < y ? x = y : 0;}
template <class T>
inline T Min(T x, T y) {return x < y ? x : y;}
template <class T>
inline T Max(T x, T y) {return x > y ? x : y;}
template <class T>
inline T Abs(T x) {return x < 0 ? -x : x;}
template <class T>
inline T Sqr(T x) {return x * x;}
//call Sqr((ll)x) when the type of returned value is "long long".

using std::map;
using std::set;
using std::pair;
```

```cpp
using std::bitset;
using std::string;
using std::vector;
using std::complex;
using std::multiset;
using std::priority_queue;

typedef long long ll;
typedef long double ld;
typedef complex<ld> com;
typedef pair<int, int> pir;
const ld pi = acos(-1.0);
const ld eps = 1e-8;
const int Maxn = 1e9;
const int Minn = -1e9;
const int mod = 998244353;
const int N = 1e5 + 5;
int T_data, n;

inline void solve()
{

}

int main()
{
    read(T_data);
    while (T_data--)
        solve();
}
```

关闭同步流

```cpp
using std::ios;
using std::cin;
using std::cout;

int main()
{
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
}
```

字符串

1. substr 函数：字符串截取函数，用于获取字符串的子串。

```cpp
// str.substr(begin, length)，用于截取 str 中以 begin 为下标长度为 length 的字串
string s = "asd";
s = s.substr(0, 1); // 结果为 "a"
```

2. find 函数：查找字符串中是否存在该字符。

```
string s = "asd";
int a = s.find('e'); //如果找到就返回 1，否则返回 -1
```

3. insert 函数：用于添加字符串。

```
string& insert(size_t pos, const string&str);
// 在位置 pos 处插入字符串 str
string& insert(size_t pos, const char *s);
// 在位置 pos 处插入字符串 s
string& insert(size_t pos, const char *ssize_t n);
// 在位置 pos 处插入字符串 s 的前 n 个字符
string& insert(size_t pos, size_t n, char c);
// 在位置 pos 处插入 n 个字符 c
```

4. erase 函数：用于作字符串删除操作。

```
// str.erase(begin, length)，用于删除 str 的从 begin 下标开始长度为 length 的字符串。
string str="abc";
str.erase(1, 1); // 结果为 "ac"
```

5. replace 函数：用来对字符串的字串作替换处理。

```
// str.replace(begin, length, string s)，用于把下标为 begin，长度为 length 的字串替换为 s。
string str = "abc";
str = str.replace(1, 2, "lk"); // str = "alk"
// str.replace(s.begin(), s.begin() + 3, "aaa"); 给出两个起始位置和终点，把其中的字符串替换为 "aaa"。
// str.replace(1, 3, "123456", 3, 5); 用 "123456" 子串的下标 [3,5] 替换 str 的 [1,3] 的位置。
// str.replace(1, 3, 5, 'a'); 用 5 个 'a' 代替 str 的 [1,3] 的位置。
```

6. to_string 函数：用于把非 string 类的数据类型转化为 string 类。

```
int a = 134;
string s = a.to_string(); // s = "134"
```

7. stoi 函数：把 string 类转化为 int 类型。

```
string str = "123";
int a = stoi(str);
```

9. 整行读取

```cpp
// scanf/getchar()
scanf("%d\n", &k); // 注意这个换行符，也可以用单个 getchar 替换
while (ch = getchar(), ch != '\n') // gets 在新的 C++17 标准中已经不能使用
    s[++n] = ch;

// cin/getline
cin >> k;
char ch; cin.get(ch);
getline(cin, s);
```

随机打乱

```cpp
    std::mt19937 rng(time(0));
    std::shuffle(a + 1, a + n + 1, rng);
```

linux 下对拍

- 以 A + B problem 对拍为例，编译器环境为 codeblocks。
- test/main.cpp

```cpp
#include <bits/stdc++.h>

using std::vector;

int main()
{
    int a, b;
    freopen("../test_gen/test.in", "r", stdin);
    freopen("../test_check/test.out", "w", stdout);

    scanf("%d%d", &a, &b);
    printf("%d\n", a + b);

    fclose(stdin); fclose(stdout);
    return 0;
}
```

- test_bf/main.cpp

```cpp
#include <iostream>

using namespace std;

int main()
{
    freopen("../test_gen/test.in", "r", stdin);
    freopen("../test_check/test_bf.out", "w", stdout);

    int a, b;
    std::cin >> a >> b;
    std::cout << a + b << std::endl;

    fclose(stdin); fclose(stdout);
    return 0;
```

```
}
```

- test_gen/main.cpp

```cpp
#include <iostream>

using namespace std;
const int mod = 1e9;
int main()
{
    freopen("test.in", "w", stdout);

    srand(time(0));
    printf("%d %d\n", rand() % mod + 1, rand() % mod + 1); // 64bit 的 ubunutu  rand 的最大值一般是 2^31 - 1
    return 0;
}
```

- test_check/main.cpp

```cpp
#include <iostream>

using namespace std;

int main()
{
    while (1)
    {
        system("g++ ../test_gen/main.cpp -o test_gen");
        system("g++ ../test/main.cpp -o test");
        system("g++ ../test_bf/main.cpp -o test_bf");

        system("./test_gen");
        system("./test");
        system("./test_bf");

        if (system("diff test.out test_bf.out"))
            break ;
        puts("checking");
    }
    return 0;
}
```

bitset

C++中__builtin内置函数是GCC、Clang等编译器所提供的一系列高效的内联函数，其中包括许多与二进制相关的函数。下面是所有与二进制相关的__builtin函数：

- __builtin_popcount(x): 返回x的二进制表示中1的个数。

```
unsigned int x = 65535u;
int count = __builtin_popcount(x); // count的值为16
```

- __builtin_clz(x): 返回x的二进制表示中从最高位开始连续0的个数，如果x的值为0，则返回所在类型的位宽。

```
unsigned int x = 0xf00000u;
int count = __builtin_clz(x); // count的值为8
```

- __builtin_ctz(x): 返回x的二进制表示中从最低位开始连续0的个数，如果x的值为0，则返回所在类型的位宽。

```
unsigned int x = 0xf0u;
int count = __builtin_ctz(x); // count的值为4
```

- __builtin_parity(x): 返回x的二进制表示中1的个数是否为奇数，是则返回1，否则返回0。

```
unsigned int x = 0xfu;
int parity = __builtin_parity(x); // parity的值为0
```


- __builtin_bswap16(x): 将x的二进制表示中的16位进行字节交换。

```
unsigned short x = 0xaabb;
unsigned short y = __builtin_bswap16(x); // y的值为0xbbaa
```

- __builtin_bswap32(x): 将x的二进制表示中的32位进行字节交换。

```
unsigned int x = 0xaabbccddu;
unsigned int y = __builtin_bswap32(x); // y的值为0xddccbbaa
```

- __builtin_bswap64(x): 将x的二进制表示中的64位进行字节交换。

```
unsigned long long x = 0xaabbccddeeff1122ull;
unsigned long long y = __builtin_bswap64(x); // y的值为0x2211ffeeddccbbaa
```

Python

- 读入多个以空间间隔的整数：

```
string = input().split() #a list of strings
n = int(string[0])
k = int(string[1])
```

- 取整数长度：

```
Len = len(str(n))
```

- 格式化输出：

```
print(f'hello world {n} {k}')
```

- 数组的简单使用：

```
ex = [1]
for i in range(1, 101):
    ex.append(ex[i - 1] * 10)
```

- `range(l,r,d)` 分别表示起点、终点和步幅：

```
>>> list(range(-10, -100, -30))
[-10, -40, -70]
```

- 按格式输出

```
print('.2f' % a)
```

高精度

- 仅供参考，且仅支持非负整数，有需求优先考虑 Python。

```
#include<bits/stdc++.h>

using std::pair;
using std::make_pair;
const int N_bigint = 2500;
char s[N_bigint * 10];

template <class T>
inline T Max(T x, T y) {return x > y ? x : y;}

int askLenx(int x)
{
    x = abs(x);
    int cnt = 0;
    while (x)
    {
        x /= 10;
        ++cnt;
```

```
    }
    return cnt;
}

struct bigint
{
    typedef long long ll;
    const ll base = 1e8;
    ll a[N_bigint];
    int len;

    void Clear() {memset(a, 0, sizeof(a)); a[len = 1] = 0;}
    bigint() {Clear();}
    bigint(ll x) {*this = x;}
    bigint operator = (const bigint &b)
    {
        memset(a, 0, sizeof(a));
        len = b.len;
        for (int i = 1; i <= len; ++i)
            a[i] = b.a[i];
        return *this;
    }

    bigint operator + (const bigint &b) const
    {
        int L = Max(len, b.len);
        bigint tmp;
        for (int i = 1; i <= L; ++i)
        {
            if (i > len)
                tmp.a[i] += b.a[i];
            else if (i > b.len)
                tmp.a[i] += a[i];
            else
            {
                tmp.a[i] += a[i] + b.a[i];
                if (tmp.a[i] >= base)
                {
                    tmp.a[i] -= base;
                    ++tmp.a[i + 1];
                }
            }
        }
        if (tmp.a[L + 1]) tmp.len = L + 1;
            else tmp.len = L;
        return tmp;
    }
    bigint operator - (const bigint &b) const
    {
        int L = Max(len, b.len);
        bigint tmp;
        for (int i = 1; i <= L; ++i)
        {
            tmp.a[i] += a[i] - b.a[i];
            if (tmp.a[i] < 0)
            {
                tmp.a[i] += base;
                --tmp.a[i + 1];
```

```
            }
        }
        while (L > 1 && !tmp.a[L]) --L;
        tmp.len = L;
        return tmp;
    }
    bigint operator * (const bigint &b) const
    {
        int L = len + b.len;
        bigint tmp;
        for (int i = 1;i <= len; ++i)
            for (int j = 1;j <= b.len; ++j)
            {
                tmp.a[i + j - 1] += a[i] * b.a[j];
                if (tmp.a[i + j - 1] >= base)
                {
                    tmp.a[i + j] += tmp.a[i + j - 1] / base;
                    tmp.a[i + j - 1] %= base;
                }
            }
        tmp.len = len + b.len;
        while (tmp.len > 1 && !tmp.a[tmp.len])
            --tmp.len;
        return tmp;
    }
    pair<bigint, bigint> Divide(const bigint &a, bigint b) const
    {
        int L = a.len; bigint c, d;
        for (int i = L; i; --i)
        {
            c.a[i] = 0;
            d = d * base;
            d.a[1] = a.a[i];
            ll l = 0, r = base - 1, mid;
            while (l < r)
            {
                mid = (l + r + 1) >> 1;
                if (b * mid <= d) l = mid;
                    else r = mid - 1;
            }
            c.a[i] = l;
            d -= b * l;
        }
        while (L > 1 && !c.a[L])
            --L;
        c.len = L;
        return make_pair(c, d);
    }
    bigint operator / (ll x) const
    {
        ll d = 0; bigint tmp;
        for (int i = len; i; --i)
        {
            d = d * base + a[i];
            tmp.a[i] = d / x;
            d %= x;
        }
        tmp.len = len;
```

```
        while (tmp.len > 1 && !tmp.a[tmp.len])
            --tmp.len;
        return tmp;
    }
    ll operator % (ll x) const
    {
        ll d = 0;
        for (int i = len; i; --i) d = (d * base + a[i]) % x;
        return d;
    }
    bigint operator / (const bigint &b) const {return Divide(*this, b).first;}
    bigint operator % (const bigint &b) const {return Divide(*this, b).second;}
    bigint &operator += (const bigint &b) {*this = *this + b; return *this;}
    bigint &operator -= (const bigint &b) {*this = *this - b; return *this;}
    bigint &operator *= (const bigint &b) {*this = *this * b; return *this;}
    bigint &operator ++() {bigint T; T = 1; *this = *this + T; return *this;} //前缀++
    bigint &operator --() {bigint T; T = 1; *this = *this - T; return *this;} //前缀--
    bigint operator ++(int) {bigint T, tmp = *this; T = 1; *this = *this + T; return tmp;} //后缀++
    bigint operator --(int) {bigint T, tmp = *this; T = 1; *this = *this - T; return tmp;} //后缀--
    bigint operator + (ll x) const {bigint T; T = x; return *this + T;}
    bigint operator - (ll x) const {bigint T; T = x; return *this - T;}
    bigint operator * (ll x) const {bigint T; T = x; return *this * T;}
    bigint operator *= (ll x) {*this = *this * x; return *this;}
    bigint operator += (ll x) {*this = *this + x; return *this;}
    bigint operator -= (ll x) {*this = *this - x; return *this;}
    bigint operator /= (ll x) {*this = *this / x; return *this;}
    bigint operator %= (ll x) {*this = *this % x; return *this;}
    bool operator == (ll x) const {bigint T; T = x; return *this == T;}
    bool operator != (ll x) const {bigint T; T = x; return *this != T;}
    bool operator <= (ll x) const {bigint T; T = x; return *this <= T;}
    bool operator >= (ll x) const {bigint T; T = x; return *this >= T;}
    bool operator < (ll x) const {bigint T; T = x; return *this < T;}
    bool operator > (ll x) const {bigint T; T = x; return *this > T;}
    bigint operator = (ll x)
    {
        len = 0;
        while (x)
            a[++len] = x % base, x /= base;
        if (!len) a[++len] = 0;
        return *this;
    }
    bool operator < (const bigint &b) const
    {
        if (len < b.len) return 1;
        if (len > b.len) return 0;
        for (int i = len; i; --i)
        {
            if (a[i] < b.a[i]) return 1;
            if (a[i] > b.a[i]) return 0;
```

```
        }
        return 0;
    }
    bool operator == (const bigint &b) const
    {
        if (len != b.len) return 0;
        for (int i = len; i; --i)
            if (a[i] != b.a[i]) return 0;
        return 1;
    }
    bool operator != (const bigint &b) const {return !(*this == b);}
    bool operator > (const bigint &b) const {return !(*this < b || *this == b);}
    bool operator <= (const bigint &b) const {return (*this < b) || (*this == b);}
    bool operator >= (const bigint &b) const {return (*this > b) || (*this == b);}

    void str(char *s)
    {
        int l = strlen(s);
        ll x = 0, y = 1; len = 0;
        for (int i = l - 1; i >= 0; --i)
        {
            x = x + (s[i] - '0') * y;
            y *= 10;
            if (y == base)
            {
                a[++len] = x;
                x = 0;
                y = 1;
            }
        }
        if (!len || x)
            a[++len] = x;
    }
    void read()
    {
        scanf("%s", s);
        this->str(s);
    }
    void print() const
    {
        printf("%d", (int)a[len]);
        for (int i = len - 1; i; --i)
        {
            for (int j = base / 10; j >= 10; j /= 10)
            {
                if (a[i] < j) putchar('0');
                    else break;
            }
            printf("%d", (int)a[i]);
        }
        putchar('\n');
    }
    int askLen() const
    {
        return (len - 1) * 8 + askLenx(a[len]);
```

```
    }
}a, b;

int main()
{
    a.read();
    b.read();
    (a + b).print();
    if (a >= b)
        (a - b).print();
    else
    {
        putchar('-');
        (b - a).print();
    }
    (a * b).print();
    pair<bigint, bigint> t = a.Divide(a, b);
    t.first.print(); t.second.print();
}
```