Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №6 по курсу «Дискретный анализ»

Студент: С. А. Красоткин
Преподаватель: Н. С. Капралов
Группа: М8О-208Б
Дата: 30.03.2021
Оценка:
Подпись:

Москва, 2021

0.0 Лабораторная работа №6

Задача: Требуется разработать программу, реализующую арифметические операции и отношения над «длинными числами» со системой счисления не меньше 100000.

Исходные данные: Известно, что входные числа содержат не больше десяти тысяч десятичных разрядов. За каждой парой чисел следует один из операндов: $<, >, =, +, -, *, /,$. Запросы оканчиваются пустой строкой.

Выходные данные: На каждый запрос должен выводиться ответ с новой строки. В случае успеха: результат операции или true/false на вопрос об бинарном отношении. В случае неуспеха (вычитание из меньшего большим, деление на ноль, возведение нуля в степень ноль): Error.

0.1 Описание и исходный код

Цифры длинного числа храню в векторе. Каждый элемент цифра по основанию системы счисления 10^9. Порядок хранения в векторе начианаются с единиц (последние 9 цифр).

```
#include <iostream>
#include <iomanip>
#include <vector>
#include <algorithm>

using namespace std;

const long long BASE = 1e9;
const int DIGIT_LENGTH = 9;
const int ZERO = 0;
const int ONES = 1;
const int TWO = 2;
vector<long long> one_element_vector(1, 0);

int ReadInteger(string s, vector<long long> &thsInteger) {
    string tmp_str = "";
    int tmp_int = ZERO;
    for(int i = s.length(); i>= ZERO; i -= DIGIT_LENGTH) {
        if(i >= DIGIT_LENGTH) {
            tmp_str = s.substr(i-DIGIT_LENGTH, DIGIT_LENGTH);
            tmp_int = stoi(tmp_str);
            thsInteger.push_back(tmp_int);
        }
        else if(i > ZERO) {
            tmp_str = s.substr(ZERO, i);
            tmp_int = stoi(tmp_str);
            thsInteger.push_back(tmp_int);
        }
    }

    while(thsInteger.size () > 1 && thsInteger.back() == 0) {
        thsInteger.pop_back();
    }

    return 0;
}

int PrintInteger(const vector<long long> &thsInteger) {
    for(int i = thsInteger.size()-1; i >= ZERO; i--) {
        (i == thsInteger.size()-1) ? cout<<thsInteger[i] :
        cout<<setfill('0') <<setw(DIGIT_LENGTH)<<thsInteger[i];
    }
    cout<<"\n";

    return 0;
```

```
}

bool IsEqual(const vector<long long> &leftInteger, const vector<long long> &rightInteger) {
    if(leftInteger.size() != rightInteger.size()) {
        return false;
    }
    for(int i = leftInteger.size() - 1; i >= ZERO; i--) {
        if(leftInteger[i] != rightInteger[i]) {
            return false;
        }
    }
    return true;
}

bool IsGreater(const vector<long long> &leftInteger, const vector<long long> &rightInteger)
    if(leftInteger.size() != rightInteger.size()) {
      return leftInteger.size() > rightInteger.size() ?  true : false;
    }
    for(int i = leftInteger.size() - 1; i >= ZERO; i--) {
        if(leftInteger[i] == rightInteger[i] && (i > 0)) {
            continue;
        }
        else {
            return (leftInteger[i] > rightInteger[i]) ? true : false;
        }
    }
    return true;
}
int SumInteger(vector<long long> &firstInteger, const vector<long long> &rightInteger) {...

int SubInteger(vector<long long> &firstInteger, const vector<long long> &secondInteger)
{...}

int MulInteger(vector<long long> &firstInteger, const vector<long long> &secondInteger)
{...}

int ShortDivInteger(vector<long long> &fstInteger, const vector<long long> &scdInteger)
{...}

int PowInteger(vector<long long> &firstInteger, const vector<long long> &secondInteger)
{...}

int SlowDivInteger(vector<long long> &firstInteger, const vector<long long> &secondInteger)
{...}

int NormInteger(vector<long long> &fstInteger, vector<long long> &scdInteger) {
    long long n = scdInteger.size();
    long long d = BASE / (scdInteger[n-1] + 1);
    if(d == 1) {
```

```
        fstInteger.push_back(0);
        return 0;
    }
    one_element_vector[0] = d;
    MulInteger(fstInteger, one_element_vector);
    one_element_vector[0] = d;
    MulInteger(scdInteger, one_element_vector);
    return 0;
}

int DivInteger(vector<long long> &firstInteger, const vector<long long> &secondInteger) {..

int Operate(char operation, vector<long long> &firstInteger, vector<long long> &secondInteg
    vector<long long> baseFstInteger = firstInteger;
    vector<long long> baseSndInteger = secondInteger;
    vector<long long> tmpInteger = firstInteger;
    vector<long long> cpRes;
    switch (operation) {
        case '=':
            IsEqual(firstInteger, secondInteger) ? cout<<"true\n" : cout<<"false\n";
            break;
        case '>':
            IsGreater(firstInteger, secondInteger) ? cout<<"true\n" : cout<<"false\n";
            break;
        case '<':
            IsGreater(secondInteger, firstInteger) ? cout<<"true\n" : cout<<"false\n";
            break;
        case '+':
            if(SumInteger(firstInteger, secondInteger) == 0) {
                PrintInteger(firstInteger);
            }
            else {
                cout<<"Error\n";
            }
            break;
        case '-':
            if(SubInteger(firstInteger, secondInteger) == 0) {
                PrintInteger(firstInteger);
            }
            else {
                cout<<"Error\n";
            }
            break;
        case '*':
            if(MulInteger(firstInteger, secondInteger) == 0) {
                PrintInteger(firstInteger);
            }
            else {
                cout<<"Error\n";
```

```
            }
            break;
        case '/':
            if(DivInteger(firstInteger, secondInteger) == 0) {
                cpRes = firstInteger;
                secondInteger = baseSndInteger;
                MulInteger(firstInteger, baseSndInteger);
                SubInteger(tmpInteger, firstInteger);
                firstInteger = cpRes;
                if(!IsGreater(baseSndInteger, tmpInteger)) {
                    firstInteger = baseFstInteger;
                    secondInteger = baseSndInteger;
                    SlowDivInteger(firstInteger, secondInteger);
                }
                PrintInteger(firstInteger);
            }
            else {
                cout<<"Error\n";
            }
            break;
        case '^':
            if(PowInteger(firstInteger, secondInteger) == 0) {
                PrintInteger(firstInteger);
            }
            else {
                cout<<"Error\n";
            }
            break;
    }
    return 0;
}

int main() {
    string op_1;
    string op_2;
    char op;
    vector<long long> firstInteger;
    vector<long long> firstIntegerBase;
    vector<long long> secondInteger;
    vector<long long> secondIntegerBase;
    while(cin>>op_1>>op_2>>op) {
        ReadInteger(op_1, firstInteger);
        ReadInteger(op_2, secondInteger);

        Operate(op, firstInteger, secondInteger);

        firstInteger.clear();
        secondInteger.clear();
        firstInteger.shrink_to_fit();
```

```
        secondInteger.shrink_to_fit();
    }
}
```

1 Сложение

К первому слагаемому прибавляется второе поразрядное. Если сумма разрядов не меньше базы счисления, то избыток уходит в следующий разряд. Результат сохраняется в первое число.

Сложность: $O(max(n, m))$, где n, m - длины подаваемых в функцию чисел.

```
int SumInteger(vector<long long> &firstInteger, const vector<long long> &rightInteger) {
    vector<long long> cpScdInt = rightInteger;
    if(IsGreater(cpScdInt, firstInteger)){
        firstInteger.swap(cpScdInt);
    }
    bool addition = false;
    long long sum = 0;
    firstInteger.push_back(0);
    for(int i = 0; i < firstInteger.size(); i++) {
        if(i < cpScdInt.size()) {
            sum = firstInteger[i] + cpScdInt[i] + addition;
            (sum < BASE) ? addition = false : addition = true;
            firstInteger[i] = sum % BASE;
        }
        else if(addition) {
            sum = firstInteger[i] + addition;
            (sum < BASE) ? addition = false : addition = true;
            firstInteger[i] = sum % BASE;
        }
        else {
            break;
        }
    }
    while(firstInteger.size () > 1 && firstInteger.back() == 0) {
        firstInteger.pop_back();
    }
    return 0;
}
```

2 Вычитание

Из вычитаемого вычитается вычитатель и результат сохраняется в первое число.

Сложность: $O(max(n, m))$, где n, m - длины подаваемых в функцию чисел.

```
int SubInteger(vector<long long> &firstInteger, const vector<long long> &secondInteger) {
    if(IsGreater(secondInteger, firstInteger)){
        return -1;
    }
    bool deduction = false;
```

```
    long long difference = 0;
    for(int i = 0; i < firstInteger.size(); i++) {
        if(i < secondInteger.size()) {
            if(firstInteger[i] >= secondInteger[i]) {
                difference = firstInteger[i] - secondInteger[i];
                if(deduction) {
                    if(difference == 0) {
                        difference = BASE - deduction;
                    }
                    else {
                        difference -= deduction;
                        deduction = false;
                    }
                }
            }
            else {
                difference = BASE + firstInteger[i] - secondInteger[i] - deduction;
                deduction = true;
            }
        }
        else if(deduction) {
            if(firstInteger[i] == 0) {
                difference = BASE - deduction;
            }
            else {
                difference = firstInteger[i] - deduction;
                deduction = false;
            }
        }
        else {
            break;
        }
        firstInteger[i] = difference;
    }
    while(firstInteger.size () > 1 && firstInteger.back() == 0) {
        firstInteger.pop_back();
    }
    return 0;
}
```

3 Произведение

Два числа перемножаются столбиком со сложностью $O(n \cdot m)$, причём привлекается дополнительная память.

```
int MulInteger(vector<long long> &firstInteger, const vector<long long> &secondInteger) {
    vector<long long> cpScdInt = secondInteger;
    if(IsGreater(cpScdInt, firstInteger)){
        firstInteger.swap(cpScdInt);
```

```
    }
    int mult_blocks = firstInteger.size() + cpScdInt.size() - 1;
    long long mults = 0;
    long long remainderByBase = 0;
    vector<long long> multInteger(mult_blocks + 1, 0);
    for(int i = 0; i < cpScdInt.size(); i++) {
        for(int j = 0; j < firstInteger.size(); j++) {
            mults = firstInteger[j] * cpScdInt[i];
            remainderByBase = mults / BASE;
            multInteger[i+j] += mults % BASE;
            if(remainderByBase > 0) multInteger[i+j+1] += remainderByBase;
        }
    }
    for(int i = 0; i < multInteger.size(); i++) {
        if(multInteger[i] >= BASE) {
            remainderByBase = multInteger[i] / BASE;
            multInteger[i] = multInteger[i] % BASE;
            multInteger[i+1] += remainderByBase;
        }
    }
    if(multInteger.back() == 0) {
        multInteger.pop_back();
    }
    while(multInteger.size () > 1 && multInteger.back() == 0) {
        multInteger.pop_back();
    }
    firstInteger = multInteger;
    return 0;
}
```

4 Деление

Над входящими числами производится анализ. Если делитель короткий, а именно меньше базы, то производится обычное деление сохраняется частное в *cur* и остаток в *carry*. Иначе используется алгоритм, описанный в [1].

Считается, что первое число (u) состоит из $n + m$ разрядов, а второе (v) из m. База счисления обозначается как b. Тогда.

1. Нормируем оба числа следующим образом. Коэффициент нормировки: $d = \frac{b}{v_{n-1}+1}$. $u_b = u_b * d$, $v_b = v_b * d$.

2. Заводится счётчик $j = m$ и цикл по нему.

3. $\widehat{q} = \frac{u_{j+n} \cdot b + u_{j+n-1}}{v_{n-1}}$, $\widehat{r}$ - остаток. После чего если выполняются неравенства $\widehat{q} = b || \widehat{q} v_{n-2} > b\widehat{r} + u_{j+n-2}$ уменьшить $\widehat{q}$ на 1 и увеличить $\widehat{r}$ на v_{n-1} и потворить проверку при $\widehat{r} < b$. Получается досаточно проверить три числа.

4. Заменить $(u_{j+n}u_{j+n-1} \dots u_j)_b$ на $(u_{j+n}u_{j+n-1} \dots u_j)_b - \widehat{q}(v_{n-1} \dots v_1 v_0)_b$. Если разность отрицательна, то добавить b^{n+1}.

5. $q_j = \widehat{q}$.

6. Если 2 шага назад был отрицательный результат, то уменьшить разряд частного на этом шаге на один и добавить $(0v_{n-1}\dots v_1v_0)_b$ к $(u_{j+n}u_{j+n-1}\dots u_{j+1}u_j)_b$.

7. Продолжить ходить по циклу.

```
int ShortDivInteger(vector<long long> &fstInteger, const vector<long long> &scdInteger) {
    long long carry = 0;
    long long cur;
    for(int i = fstInteger.size() - 1; i >= 0; i--) {
        cur = fstInteger[i] + carry * BASE;
        fstInteger[i] = cur / scdInteger[0];
        carry = cur % scdInteger[0];
    }
    while(fstInteger.size() > 1 && fstInteger.back() == 0) {
        fstInteger.pop_back();
    }

    return 0;
}

int SlowDivInteger(vector<long long> &firstInteger, const vector<long long> &secondInteger)

    vector<long long> res(firstInteger.size());
    vector<long long> tmpVector(secondInteger.size());
    vector<long long> shiftFactor = one_element_vector;
    shiftFactor[0] = 0;
    shiftFactor.push_back(1);
    tmpVector.front() = firstInteger.back();

    for(int i = 0; i < firstInteger.size(); i++) {
        long long l = -1;
        long long r = BASE;
        vector<long long> factor;
        vector<long long> nesMult;
        while(l + 1 < r) {
            long long m = (l + r) / 2;
            factor.push_back(m);
            MulInteger(factor, secondInteger);
            if(IsGreater(factor, tmpVector)) {
                r = m;
            }
            else {
                l = m;
            }
            factor.clear();
        }
        res[i] = l;
        factor.push_back(l);
```

```
        MulInteger(factor, secondInteger);
        SubInteger(tmpVector, factor);
        MulInteger(tmpVector, shiftFactor);
        if(i + 1 < firstInteger.size()) {
            tmpVector.front() = firstInteger[firstInteger.size()-i-2];
        }
    }

    reverse(res.begin(), res.end());
    while(res.back() == 0 && res.size() > 1) {
        res.pop_back();
    }
    firstInteger = res;

    return 0;
}

int DivInteger(vector<long long> &firstInteger, vector<long long> &secondInteger) {
    vector<long long> baseFstInteger = firstInteger;
    vector<long long> baseSndInteger = secondInteger;
    if(secondInteger.size() == 1 && secondInteger.back() == 0) {
        return -1;
    }
    if(IsGreater(secondInteger, firstInteger)){
        firstInteger.clear();
        firstInteger.shrink_to_fit();
        firstInteger.push_back(0);
        return 0;
    }
    else if(IsEqual(secondInteger, firstInteger)){
        firstInteger.clear();
        firstInteger.shrink_to_fit();
        firstInteger.push_back(1);
        return 0;
    }
    else if(secondInteger.size() == 1) {
        ShortDivInteger(firstInteger, secondInteger);
        return 0;
    }

    NormInteger(firstInteger, secondInteger);
    if(firstInteger.size() - secondInteger.size() == 0) {
        firstInteger.push_back(0);
    }

    bool overcup = false;
    int n = secondInteger.size();
    int m = (firstInteger.size() - n) > 0 ? (firstInteger.size() - n) : 1;
    long long q_cup;
```

```
long long r_cup;
long long remainderByBase;
vector<long long> partVector;
vector<long long> tmpVector;
vector<long long> powN (n, 0);
powN.push_back(1);
vector<long long> dvnInteger(m, 0);

for(int j = m - 1; j >= 0; j--) {
    q_cup = (firstInteger[j+n]*BASE+firstInteger[j+n-1]) / (secondInteger[n-1]);
    r_cup = (firstInteger[j+n]*BASE+firstInteger[j+n-1]) % (secondInteger[n-1]);

    while(q_cup == BASE || (q_cup * secondInteger[n-2] > (BASE * r_cup + firstInteger[j
        q_cup--;
        if(r_cup + secondInteger[n-1] < BASE) {
            r_cup += secondInteger[n-1];
        }
        else {
            break;
        }
    }

    for(int k = 0; k <= n; k++) {
        partVector.push_back(firstInteger[k+j]);
    }
    one_element_vector[0] = q_cup;
    tmpVector = one_element_vector;
    MulInteger(tmpVector, secondInteger);

    if(IsGreater(partVector, tmpVector) || IsEqual(partVector, tmpVector)) {
        SubInteger(partVector, tmpVector);
        for(int k = 0; k <= n; k++) {
            firstInteger[j+k] = partVector[k];
        }
    }
    else {
        SumInteger(partVector, powN);
        SubInteger(partVector, tmpVector);
        overcup = true;
    }

    dvnInteger[j] = q_cup % BASE;
    bool overflow = false;
    if(q_cup >= BASE) {
        overflow = true;
        remainderByBase = q_cup / BASE;
        for(int k = j + 1; k < m && overflow; k++) {
            dvnInteger[k] += remainderByBase;
            if(dvnInteger[k] >= BASE) {
```

```
                    remainderByBase = dvnInteger[k] / BASE;
                    dvnInteger[k] = dvnInteger[k] % BASE;
                }
                else {
                    dvnInteger[k] = dvnInteger[k] % BASE;
                    overflow = false;
                }
            }
            if(overflow) {
                dvnInteger.push_back(remainderByBase);
            }
        }

        if(overcup) {
            dvnInteger[j] -= 1;
            SumInteger(partVector, secondInteger);
            for(int k = 0; k <= n; k++) {
                firstInteger[k+j] = partVector[k];
            }
        }

        overcup = false;
        partVector.clear();
        tmpVector.clear();
        if(powN.size() > 2) {
            powN.pop_back();
            powN.pop_back();
            powN.push_back(1);
        }
        else if(powN.size() == 2) {
            powN.pop_back();
            powN[0] = 1;
        }
        else {
            powN.clear();
        }
    }

    while(dvnInteger.size () > 1 && dvnInteger.back() == 0) {
        dvnInteger.pop_back();
    }
    firstInteger = dvnInteger;

    return 0;
}
```

5 Возведение в степень

При возведении в степень используется идея бинарного возведения в степень. Для любого чётного числа верно: $a^n = (a^{\frac{n}{2}}) \cdot a^{\frac{n}{2}})$, значит можно потратить одну операцию. В нечётном случае можно снизить степень: $a^n = a * a^{n-1}$. Сложность такого подхода: $O(log \quad n)$.

```
int PowInteger(vector<long long> &firstInteger, const vector<long long> &secondInteger) {
    vector<long long> cpScdInt = secondInteger;
    if(firstInteger.size() == 1) {
        if(firstInteger.back() == 0) {
            if(cpScdInt.size() == 1 && cpScdInt.back() == 0) {
                return -1;
            }
            return 0;
        }
        if(firstInteger.back() == 1) {
            return 0;
        }
    }

    vector<long long> tmpVector = firstInteger;
    vector<long long> resVector = one_element_vector;
    resVector[0] = ONES;
    vector<long long> zeroes = one_element_vector;
    zeroes[0] = ZERO;
    vector<long long> ones = one_element_vector;
    ones[0] = ONES;
    one_element_vector[0] = TWO;


    while(IsGreater(cpScdInt, zeroes)) {
        if(cpScdInt[0] & 1) {
            MulInteger(resVector, firstInteger);
            SubInteger(cpScdInt, ones);
        }
        MulInteger(firstInteger, firstInteger);
        ShortDivInteger(cpScdInt, one_element_vector);
    }

    firstInteger = resVector;

    return 0;
}
```

0.2 Тест производительности

Сравню деление длинным способом, если делитель не меньше базы, и тем что реплизован в моей программе.

```
goku@debian:~/Documents/Vuz/MAI/4sem/DA/Labs/6/Report$ g++ ShortSlowDiv.cpp
goku@debian:~/Documents/Vuz/MAI/4sem/DA/Labs/6/Report$ g++ ShortSlowNormDiv.cpp -o SSND
goku@debian:~/Documents/Vuz/MAI/4sem/DA/Labs/6/Report$ time ./a.out < test.txt > out1.txt
real    0m0.017s
user    0m0.017s
sys     0m0.000s
goku@debian:~/Documents/Vuz/MAI/4sem/DA/Labs/6/Report$ time ./SSND < test.txt > out1.txt

real    0m0.007s
user    0m0.007s
sys     0m0.000s
```

Как видно, бинарное деление в 2 раза дольше способа из книги [1].

0.3 Выводы

Впервые выполнив лабораторную работу, научился работать с длинными числами. Длинная арифметика применяется в криптографии, математическом и финансовом ПО, так что, возможно, мои новые знания мне когда-то помогут.

Благодаря этой работе много работал с тестированием. Подбирал тесты, генерировал их, сравнивал с грубой реализацией. Я делал это почти что впервые, так как боялся по незнанию трогать какие-то ни было скрипты.

Моя реализация почти неделю не проходила проверяющую систему из-за множества тонких мест в реализации деления, на мой взгляд, из-за недостаточного раскрытия в источнике [1] и моего неумения. В частности остался нерешённым вопрос с компенсацией сложения. Примечательно, этот сегмент выполняется с вероятностью $0,000000002$, что значительно затрудняет дебаг. По итогу ошибку так и не нашёл, хотя обнаружил тесты на которых она падает. Отчаялся и добавил функцию бинарного деления в таких случаях. Я этим совершенно не расстроен, потому что получившаяся реализация превосходит программу почти с полным бинарным делением.

Интересно, конечно, найти что именно пошло не так, но времени нет. Полагаю следующие шаги по поиску: всё-таки найти что не так в случае с отриацательной реализацией; попробовать с другой нормализацией, описанной в [1], и сравнить результаты; использовать переборную эвристику: пусть частное найдено, тогда умножаем его на делитель и проверяем разность, если она не удовлетворительна, то начиная со старших разрядов пробуется добавить по единичке или убрать и сверяться и так спуститься до разряда единиц.

В будущем в случае свободного времени, например, на летних каникулах интересно попробовать умножение и деление с помощью быстрого преобразования Фурье.

Литература

[1] Дональд Кнут *Искусство программирования, том 2* — . Перевод с английского: В. Штонда, Г. Петриковец, А. Орлович. — 788 с.