Компьютерная алгебра

(курс лекций)

Игорь Алексеевич Малышев

Computer.Algebra@yandex.ru

КСПТ

КОМПЬЮТЕРНЫЕ СИСТЕМЫ И ПРОГРАММНЫЕ ТЕХНОЛОГИИ

Лекция 4

Основы арифметических вычислений

Содержание лекции

- Целочисленная арифметика
- Полиномиальная арифметика

План лекции: тема подраздела

- **Целочисленная арифметика**
- Полиномиальная арифметика

Целочисленная арифметика

Постулаты

1) Объекты вычислений в целочисленной арифметике – короткие и длинные неотрицательные целые числа.

2) Система счисления (СС) – позиционная, с постоянным основанием.

3) Знак числа и позиция точки (разделителя целой и дробной частей) хранятся и обрабатываются отдельно.

4) Структура данных для представления объектов вычислений в целочисленной арифметике – список (i-я ячейка содержит коэффициент d_i при i-й степени основания системы счисления B).

5) Язык реализации – C++.

Замечание. Идентификатор основания СС (B) – BASE.

Целочисленная арифметика

Выбор основания СС (критерии) :

1) Основание должно подходить под один из базовых типов данных языка реализации арифметических вычислений.
2) Основание должно быть как можно больше, чтобы уменьшить размер представления длинных чисел и, соответственно, увеличить скорость выполнения арифметических операций над ними.
3) Основание должно быть достаточно малого размера, чтобы все операции с коэффициентами использовали только базовый тип данных.
4) Выбор значения BASE равным степени 10 удобен для организации вывода информации и отладки программ.
5) Выбор значения BASE равным степени 2 обеспечивает быстрое выполнений операций на низком уровне.

Целочисленная арифметика

Выбор основания СС (результаты) :

Разумный компромисс:

```
#define BASE 10000
```

Пример. Представление числа 20 !
(20 ! = 243 2902 0081 7664 0000)

$D = 0 + 7664 * BASE + 81 * BASE^2 + 2902 * BASE^3 + 243 * BASE^4$

Замечание. Далее для краткости изложения используется десятичная система счисления (BASE = 10).

Целочисленная арифметика

Объявление класса (длинное целое + простейшие операции) :

```
class BigInt {

public:
        // к этим членам можно закрыть доступ
        ulong Size, SizeMax;        // Size – текущая длина
                                    // SizeMax – максимальная длина
        short *Coef;                // Массив коэффициентов

        // в этом случае здесь также должны быть описаны дружественные функции
        // операций над большими числами, которые будут разобраны ниже.
        BigInt ();
        BigInt (ulong);
        BigInt (const BigInt&);
        virtual ~BigInt ();

        void zero ();               // Обнулить число
        void update ();

        BigInt& operator = (const BigInt&);
        operator long ();           // Оператор преобразования BigInt к типу long
};
```

Целочисленная арифметика

Конструкторы и деструктор объектов класса :

```
BigInt::BigInt () {
        SizeMax = Size = 0;             // Объявление вида BigInt A;
        Coef = NULL;                    // Создается полностью пустое число
}

BigInt::BigInt (ulong MaxLen) {
        Coef = new short [MaxLen]; //Объявление вида BigInt A(10);
        SizeMax = MaxLen;               // Выделяет память под MaxLen цифр
        Size = 0;
}

BigInt::BigInt (const BigInt &A) {  // Конструктор копирования
        SizeMax = A.SizeMax;            // Создает B, равное A
        Size = A.Size;
        Coef = new short [SizeMax];
        for (ulong i=0; i<SizeMax; i++) Coef [i] = A.Coef [i];
}

BigInt::~BigInt () {                    // Деструктор (Освобождает память)
        delete Coef;
}
```

Целочисленная арифметика

Простейшие операции над объектами класса :

Операция обнуления числа:

```
void BigInt::zero () {                  // A.zero () – обнулить число
        for (ulong i=0; i<SizeMax; i++) Coef [i] = 0;
        Size = 1;
}
```

Оператор вычисления значения числа в «обычном виде» (полезен при отладке, когда BASE=10 и числа небольшие):

```
BigInt::operator long () {
        long tmp = 0;                   // при вычислениях может произойти переполнение
        for (ushort i=0; i<Size; i++) tmp += Coef [i] * (long) pow ( BASE, (real) i );
        return tmp;
}
```

Целочисленная арифметика

Оператор присваивания :

```
inline BigInt& BigInt::operator = (const BigInt &A) {
        const short *a = A.Coef;
        if (this == &A) return *this;              // Если присваивание вида A=A, выйти
        if ( SizeMax < A.Size ) {                  // Если размера не хватает,
                                                   // переинициализация
                if (Coef) delete Coef;
                Coef = new short [A.Size];
                SizeMax = Size = A.Size;

        } else Size = A.Size;

        for (ulong l=0; l<Size; l++) Coef [l] = a [l];

        return *this;
}
```

Учтены, в том числе, особые случаи применения оператора:

1) **Случай A = A;**
2) **Случай необходимости выделения дополнительной памяти под коэффициенты, если размеры операндов не совпадают;**
3) **Случай A = B = C (интерпретируется как A = (B = C)).**

Целочисленная арифметика

Выбор типа интерфейса операций – 1 :

Тип 1. Оператор – часть класса.

Пример. Реализация оператора «+» (сложение): C = A + B

BigInt& operator + (BigInt&);

Достоинство – инфиксный оператор (перегрузка оператора «+»).

Недостаток – необходимость дополнительной памяти и времени (для промежуточных результатов: TEMP = A + B; C = TEMP).

Тип 2. Оператор – внешняя функция.

Достоинство – все операнды (вход/выход) – параметры функций.

Недостаток – «непривычность» описания математического оператора в форме вызова программного кода.

Целочисленная арифметика

Выбор типа интерфейса операций – 2 :

- **Выбор типа 1 (оператор – часть класса) для реализации операций ввода / вывода (<< , >>).**

- **Выбор типа 2 (оператор – внешняя функция) для реализации арифметических операций (+ , - , * , /).**

Замечание.

Принятые решения о выборе способа реализации операций преследуют исключительно учебные цели. В реальных системах компьютерной алгебры пользовательский интерфейс должен быть максимально «дружественным» (все операции должны быть инфиксными, а все функции – префиксными).

Целочисленная арифметика

Схема выполнения операции вывода на печать :

Общие правила:

- **Основание СС является степенью 10 (BASE = 10000 = 10^4), поэтому количество цифр в одном знаке длинного целого числа будет равно этой степени (BASE_DIG = 4).**
- **Внешний цикл осуществляет перебор коэффициентов (при степенях основания СС), начиная со старшего.**
- **Внутренний цикл производит последовательное выделение (десятичных) цифр текущего коэффициента и выдает эти цифры на печать.**

Особые случаи:

Нет.

Целочисленная арифметика

Оператор вывода на печать:

```
#define BASE_DIG 4       // Полагаем, что BASE = 10000

ostream& operator << (ostream& os, const BigInt& A) {          // Перегрузка оператора <<
        long j, Digit=0;
        short Pow, Dec, Coef;

        os << A.Coef[A.Size-1];
        for (long i=A.Size-2; i>=0; i--) {                     // Цикл вывода коэффициентов
                Pow = BASE/10;
                Coef = A.Coef[i];
                for (j=0; j<BASE_DIG; j++) {                   // Цикл, выводящий каждый коэффициент
                        Dec = Coef/Pow;
                        Coef -= Dec*Pow;
                        Pow /= 10;                             // Очередная цифра получается делением
                        os << Dec;                             // коэффициента на 10^j
                        Digit++;
                        // Каждые 1000 цифр сопровождаются переходом строки
                        if (Digit%1000==0) os << "\n\n";
                        else if (Digit%50==0) os << "\t: " << Digit << "\n";
                }
        }
        return os;
}
```

Целочисленная арифметика

Схемы выполнения аддитивных операций :

Схема сложения.

Общие правила:

- Складываем цифры слева направо (цифры хранятся в обратном порядке).
- Если зафиксировано переполнение (т.е. сумма цифр больше основания системы счисления), то формируем перенос в следующий разряд.

Особые случаи:

Нет.

Схема вычитания.

Общие правила:

- Просматривая цифры слева направо (цифры хранятся в обратном порядке), вычитаем цифру вычитаемого из цифры уменьшаемого.
- Если цифра вычитаемого больше цифры уменьшаемого, то формируем заимствование из следующего (старшего) разряда.

Особые случаи:

- Если вычитаемое больше по размеру, чем уменьшаемое, то выходим по ошибке (т.к. все числа, включая результат, ТОЛЬКО положительны).
- Если длины чисел одинаковы, но значение вычитаемого больше значения уменьшаемого, то последнее заимствование останется неиспользованным. Результат будет дополнением до $BASE^{B.Size}$.

Целочисленная арифметика

Оператор сложения - 1:

```
// Вычисление C = A+B, работает вызов вида Add (A, B, A).
// Максимальный размер C должен быть достаточен для хранения суммы

void Add(const BigInt &A, const BigInt &B, BigInt &C) {

        ulong i;
        long temp;      // temp здесь и далее играет роль “временной” цифры,
                        // до выполнения переноса. Возможно, temp > BASE.

        // Здесь и в следующих примерах
        // для быстрого доступа к коэффициентам
        // объявляются временные указатели a, b, c.
        const short *a=A.Coef, *b=B.Coef; short *c=C.Coef;

        carry = 0;      // перенос в следующий разряд

        // Добиваемся B.Size ≤ A.Size.
        if ( A.Size < B.Size ) {
                Add(B,A,C);
                return;
        }
```

Целочисленная арифметика

Оператор сложения - 2:

```
// Теперь B.Size ≤ A.Size. Складываем два числа, i -номер текущей цифры
for (i=0; i<B.Size; i++) {
        temp = a[i] + b[i] + carry;
        if (temp >= BASE) {    // Переполнение. Перенести единицу.
                        c[i] = temp - BASE;    carry = 1;  }
        else {
                        c[i] = temp;    carry = 0;  }
}

// Меньшее число кончилось
for (; i < A.Size; i++) {
        temp = a[i] + carry;
        if (temp >= BASE) {
                        c[i] = temp - BASE;   carry = 1;   }
        else { c[i] = temp;    carry = 0;   }
}

// Если остался перенос – добавить его в дополнительный разряд
if (carry) { c[i] = carry;   C.Size = A.Size+1;  }
else C.Size = A.Size;
}
```

Целочисленная арифметика

Оператор вычитания - 1:

```
// C = A-B, должно быть A.Size >= B.Size. Работает вызов Sub(A, B, A).
// Если длины равны, но A<B: возвращается -1, результат будет дополнением.

int Sub (const BigInt& A, const BigInt& B, BigInt& C) {

        const short *a=A.Coef, *b=B.Coef;
        short *c=C.Coef;
        ulong i;
        long temp, carry=0;
        if ( A.Size < B.Size ) error ("BigSub: size error");

        for (i=0; i<B.Size; i++) {
                temp = a[i] - b[i] + carry;
                if (temp < 0) {
                                c[i] = temp + BASE;    carry = -1; }
                else { c[i] = temp;       carry = 0;  }
        }
```

Целочисленная арифметика

Оператор вычитания - 2:

```
        for (; i<A.Size; i++) {
                temp = a[i] + carry;
                if (temp < 0) {
                                c[i] = temp + BASE;  carry = -1;
                }
                else {
                                 c[i] = temp;  carry = 0;
                }
        }

        // Размер результата может быть гораздо меньше, чем у исходного числа
        // Устанавливаем его по первому положительному разряду
        i = A.Size-1;
        while ( (i>0) && (c[i]==0) ) i--;
        C.Size = i+1;
        return carry;
}
```

Сложность аддитивных операций :

Функция времени вычислений:

$$T_{ADD}(A,B) = T_{SUB}(A,B) = O\{\max [L(A), L(B)]\},$$

где A и B – операнды (в виде списков разрядов длинных целых чисел);
L(A), L(B) – размеры (длины) списков A и B.

Количество ячеек (разрядов) списка результата:

$$L(C) \le \max [L(A), L(B)] + 1$$

Целочисленная арифметика

Схемы выполнения операции умножения - 1 :

Упрощённая схема :

<длинное_целое> = <длинное_целое> * <короткое_целое>.

Общие правила:

- **Каждую цифру длинного целого умножаем на короткое целое.**
- **При этом может возникнуть переполнение, поэтому вычисляем переносы, которые могут принимать значения от 0 до BASE-1.**
- **Для определения правильного значения цифры вместо медленной (программно реализуемой процессором) операции вычислений остатка по модулю BASE применяем быстрые (аппаратно реализуемые процессором) операции – вычитание с умножением.**

Особые случаи:

Нет.

Целочисленная арифметика

Схемы выполнения операции умножения - 2 :

Общая схема :

<длинное_целое> = <длинное_целое> * <длинное_целое>.

Общие правила:

- **Обнуляем значение произведения (C = 0).**
- **Обнуляем указатель текущего перемножаемого разряда длинного целого (i=0).**
- **Вычисляем временный результат – умножаем i-ую цифру множимого (A) на множитель (B) и прибавляем его к C, начиная с i-ой позиции.**

  В обычном («школьном») алгоритме умножения «столбиком» последовательно перемножаем цифры сомножителей, сохраняя временные результаты. Затем все ранее полученные временные результаты суммируем с учётом разрядов.
- **Если значение очередного вычисленного разряда C оказалось больше BASE, то выполняем перенос.**
- **Если разряды множимого не закончились, то инкрементируем указатель (i++) и продолжаем умножение; в противном случае – умножение выполнено.**

Особые случаи:

Нет.

Целочисленная арифметика

Оператор умножения (упрощённая схема) :

```
// C = A * B, работает вызов Smul (A, B, A).
void SMul (const BigInt &A, const short B, BigInt &C) {
        ulong i, temp;
        const short *a=A.Coef;
        short *c=C.Coef, carry=0;
        for (i=0; i<A.Size;i++) {
                temp = a[i]*B + carry;
                carry = temp / BASE;
                c[i] = temp - carry*BASE;          // c[i] = temp % BASE ( Так очень медленно ! )
        }
        if (carry) {
                // Число удлинилось за счет переноса нового разряда
                c[i] = carry;
                C.Size = A.Size+1;
        }
        else C.Size = A.Size;
}
```

Целочисленная арифметика

Оператор умножения (общая схема) – 1 :

```
// C = A * B, работает вызов Mul (A, B, A)

void Mul (const BigInt &A, const BigInt &B, BigInt &C) {

        ulong i, j;
        const short *a=A.Coef, *b=B.Coef;
        short *c=C.Coef;
        ulong temp, carry;

        // Обнулить необходимую для работы часть C
        for ( i=0; i <= A.Size + B.Size; i++ ) c[i]=0;
```

Целочисленная арифметика

Оператор умножения (общая схема) – 2 :

```
        // Выполнение основного цикла умножения
        for ( i = 0; i < A.Size; i++) {
                carry = 0;
                // вычисление временного результата с одновременным прибавлением
                // его к c[i+j] (делаются переносы)
                for (j = 0; j < B.Size; j++) {
                        temp = a[i] * b[j] + c[i+j] + carry;
                        carry = temp / BASE;
                        c[i+j] = temp - carry*BASE;
                }
                c[i+j] = carry;
        }
        // Установить размер по первой ненулевой цифре
        i = A.Size + B.Size - 1;
        if ( c[i] == 0 ) i--;
        C.Size = i+1;
}
```

Сложность операции умножения :

Функция времени вычислений:

$T_{MUL}(A,B) = O\,[L(A) * L(B)]$,

где A и B – операнды (в виде списков разрядов длинных целых чисел);
L(A), L(B) – размеры (длины) списков A и B.

Количество ячеек (разрядов) списка результата:

$L(C) \le [L(A) * L(B)] + 1$

Целочисленная арифметика

Схема выполнения операции деления :

Упрощённая схема :

<длинное_целое> = <длинное_целое> / <короткое_целое>.

Общие правила:

- **Осуществляем перебор цифр делимого, начиная со старшего разряда длинного целого.**
- **Каждую текущую цифру делимого делим на делитель (короткое целое): целую часть (очередную цифру) результата добавляем (конкатенацией справа) к общему частному, остаток от деления переносим для участия в обработке следующей цифры делимого.**
- **Последний перенесённый остаток является остатком от всего деления.**

Особые случаи:

Нет.

Целочисленная арифметика

Оператор деления (упрощённая схема) :

```
// Частное Q = A/B.  Остаток R = A%B. A, Q – длинные целые. B, R – короткие целые.
void SDiv(const BigInt &A, const short B, BigInt &Q, short &R) {
        short r=0, *q=Q.Coef;
        const short *a=A.Coef;
        long i, temp;

        for ( i=A.Size-1; i>=0; i--) {          // идти по A, начиная от старшего разряда
                temp = r*BASE + a[i];           // r – остаток от предыдущего деления
                                                // вначале r=0, temp – текущая цифра A с
                                                // учетом перенесенного остатка

                q[i] = temp / B;                // i-я цифра частного
                r = temp - q[i]*B;              // остаток примет участие в вычислении
                                                // следующей цифры частного
        }
        R = r;
        // Размер частного меньше, либо равен размера делимого
        i = A.Size-1;
        while ( (i>0) && (q[i]==0) ) i--;
        Q.Size = i+1;
}
```

Целочисленная арифметика

Общая схема операции деления :

Цель операции деления (длинного) целого A на (длинное) целое B – определение двух (длинных) целых Q и R, которые удовлетворяют свойству делимости с остатком:

$$A = B * Q + R \ , \ 0 \le R \le B$$

Пусть

$$A = (a_0, a_1, a_2, \ldots a_{n+m-1})$$
$$B = (b_0, b_1, b_2, \ldots b_{n-1})$$

Общие правила:

- **На каждом (i-м) шаге (кроме первого) необходимо выполнить деление (n+1)-разрядной части делителя на n-разрядный делитель:**
  - **угадать i-ю цифру частного Q[i];**
  - **не создавая временных чисел, вычесть «сдвинутое» произведение B * Q[i] из A. (При этом сдвиг B относительно A на каждом шаге уменьшается).**

Проблема: Каким образом обеспечить угадывание ?

Целочисленная арифметика

Пример. Деление длинных целых чисел.

A = 68971 B = 513 n = 5 m = 2

Шаг 1. i=4 (индекс старшего коэффициента A)

a. Угадываем Q[0] = 1

b. Вычитаем сдвинутое B * 1 = 513 из A

(на бумаге пишем только значимую для следующего шага часть A)

Шаг 2. i=3

a. Угадываем Q[1] = 3

b. Вычитаем сдвинутое B * 3 = 1539 из A

Шаг 3. i=2

a. Угадываем Q[2] = 4

b. Вычитаем сдвинутое B * 4 = 2052 из A

Шаг 4. I < m = 2

Деление завершено: Q=134 R=229

```
 68971 | 513
-513   |-----
 1767  | 134
-1539
 2281
-2052
  229
```

Целочисленная арифметика

Правило угадывания цифр частного - 1

Ключевая идея угадывания.

Анализ значения отношения наиболее значимых цифр
текущего делимого
(на первом шаге оно равно A,
на последующих шагах – ***текущему остатку*** U)
и делителя (B) с помощью вычисления
пробного частного (Q_{PROBE})
от деления двузначного числа U[n] * BASE + U[n-1]
на наиболее значимую цифру делителя B[n-1],
где $U = (u_0, u_1, u_2, \ldots u_n)$, $B = (b_0, b_1, b_2, \ldots b_{n-1})$.

Целочисленная арифметика

Правило угадывания цифр частного - 2

Теорема о пробном частном (Дональд Кнут).

Если делитель нормализован, т.е. его старшая цифра больше половины основания СС (B[n-1] > BASE/2), то верны следующие утверждения:

(1) $Q_{PROBE} - 2 \le Q \le Q_{PROBE}$, т.е. пробное частное не меньше истинного частного, но может быть больше на 1 или на 2.

(2) Если дополнительно выполнено следующее неравенство:

$Q_{PROBE} * B[n-2] > BASE * R_{PROBE} + U[n-2]$,

где R_{PROBE} – остаток при вычислении Q_{PROBE} и $Q_{PROBE} \neq BASE$,

то $Q_{PROBE} - 1 \le Q \le Q_{PROBE}$,

причем вероятность события $Q_{PROBE} = Q + 1$ приблизительно равна 2/BASE.

Целочисленная арифметика

Общая схема операции деления : (исправленная и дополненная)

Общие правила:

- На каждом шаге вычисляем пробное частное:
  Q_{PROBE} = (U[n] * BASE + U[n-1]) / B[n-1] при B[n-1] ≥ (BASE / 2)
  и декрементируем его, пока не станут выполняться условия:
  Q_{PROBE} * B[n-2] > BASE * R_{PROBE} + U[n-2] и $Q_{PROBE} \neq$ BASE
  В результате получим либо правильное частное, либо с вероятностью 2/BASE на единицу большее число.
- Если в результате угадывания получено Q_{PROBE} = Q + 1, то в пункте b. (см. пример) требуется выполнять вычитание с помощью оператора Sub (см. выше), которое вместо отрицательного числа выдаст дополнение до следующей степени основания. Возникший при этом последний перенос будет равен borrow = -1, что сигнализирует о необходимости коррекции – восстановлении остатка (путем сложения его с делителем). Вновь возникший перенос carry = 1 следует игнорировать.

Особые случаи:

- Если делитель не нормализован, т.е. условие B[n-1] ≥ (BASE / 2) не выполнено, то перед вычислением пробного частного выполняем нормализацию делителя, домножая делитель и делимое на число scale = BASE / (B[n-1] + 1).

Целочисленная арифметика

Оператор деления (общая схема) – 1 :

```
// Частное Q = A/B.  Остаток R = A%B.
// Все операнды – длинные целые.

void Div (const BigInt &A, BigInt &B, BigInt &Q, BigInt &R) {

        // Вырожденный случай 1. Делитель больше делимого.
        if ( A.Size < B.Size ) {
                Q.zero();
                R=A;
                return;
        }

        // Вырожденный случай 2. Делитель – короткое целое.
        if ( B.Size == 1) {
                SDiv ( A, B.Coef[0], Q, R.Coef[0]);
                R.Size = 1;
                return;
        }
```

Целочисленная арифметика

Оператор деления (общая схема) – 2 :

```
// Создать временный массив U, равный A
// Максимальный размер U на цифру больше A, с учетом
// возможного удлинения A при нормализации
BigInt U(A.Size+1); U = A; U.Coef[A.Size]=0;
// Указатели для быстрого доступа
short *b=B.Coef, *u=U.Coef, *q=Q.Coef;

long n=B.Size, m=U.Size-B.Size;
long uJ, vJ, i;
long temp1, temp2, temp;
short scale;              // коэффициент нормализации
short qProbe, r;          // догадка для частного и соответствующий остаток
short borrow, carry;      // переносы

// Нормализация
scale = BASE / ( b[n-1] + 1 );
if (scale > 1) {
        SMul (U, scale, U);
        SMul (B, scale, B);
}
```

Целочисленная арифметика

Оператор деления (общая схема) – 3 :

```
// Главный цикл шагов деления.
// Каждая итерация дает очередную цифру частного.
// vJ - текущий сдвиг B относительно U, используемый при вычитании,
// по совместительству - индекс очередной цифры частного.
// uJ – индекс текущей цифры U

for (vJ = m, uJ=n+vJ; vJ>=0; --vJ, --uJ) {
        qProbe = (u[uJ]*BASE + u[uJ-1]) / b[n-1];
        r = (u[uJ]*BASE + u[uJ-1]) % b[n-1];

        // Пока не будут выполнены необходимые условия, уменьшать частное.
        while ( r < BASE) {
                temp2 = b[n-2]*qProbe;
                temp1 = r*BASE + u[uJ-2];
                if ( (temp2 > temp1) || (qProbe==BASE) ) {
                        // условия не выполнены, уменьшить qProbe
                        // и досчитать новый остаток
                        --qProbe;
                        r += b[n-1];
                } else break;
        }
```

Целочисленная арифметика

Оператор деления (общая схема) – 4 :

```
// Теперь qProbe - правильное частное или на единицу больше q
// Вычесть делитель B, умноженный на qProbe из делимого U,
// начиная с позиции vJ+i
carry = 0; borrow = 0;
short *uShift = u + vJ;
// цикл по цифрам B
for (i=0; i<n;i++) {
        // получить в temp цифру произведения B*qProbe
        temp1 = b[i]*qProbe + carry;
        carry = temp1 / BASE;
        temp1 -= carry*BASE;
        // Сразу же вычесть из U
        temp2 = uShift[i] - temp1 + borrow;
        if (temp2 < 0) {
                uShift[i] = temp2 + BASE;
                borrow = -1;
        } else {
                uShift[i] = temp2;
                borrow = 0;
        }
}
```

Целочисленная арифметика

Оператор деления (общая схема) – 5 :

```
// Возможно, умноженое на qProbe число B удлинилось.
// Если это так, то после умножения остался
// неиспользованный перенос carry. Вычесть и его тоже.
temp2 = uShift[i] - carry + borrow;
if (temp2 < 0) {
        uShift[i] = temp2 + BASE;
        borrow = -1;
} else {
        uShift[i] = temp2;
        borrow = 0;
}
```

Целочисленная арифметика

Оператор деления (общая схема) – 6 :

```
// Прошло ли вычитание нормально ?
if (borrow == 0) {        // Да, частное угадано правильно
                q[vJ] = qProbe;
} else {        // Нет, последний перенос при вычитании borrow = -1,
                // значит, qProbe на единицу больше истинного частного
                q[vJ] = qProbe-1;
                // добавить одно, вычтенное сверх необходимого B к U
                carry = 0;
                for (i=0; i<n; i++) {
                                temp = uShift[i] + b[i] + carry;
                                if (temp >= BASE) {
                                                uShift[i] = temp - BASE;
                                                carry = 1;
                                } else {
                                                uShift[i] = temp;
                                                carry = 0;
                                }
                }
                uShift[i] = uShift[i] + carry - BASE;
}
```

Целочисленная арифметика

Оператор деления (общая схема) – 7 :

```
        // Обновим размер U, который после вычитания мог уменьшиться
        i = U.Size-1;
        while ( (i>0) && (u[i]==0) ) i--;
        U.Size = i+1;
    }

    // Деление завершено !

    // Размер частного равен m+1, но, возможно, первая цифра - ноль.
    while ( (m>0) && (q[m]==0) ) m--;
    Q.Size = m+1;

    // Если происходило домножение на нормализующий множитель –
    // разделить на него. То, что осталось от U – остаток.
    if (scale > 1) {
        short junk; // почему-то остаток junk всегда будет равен нулю…
        SDiv ( B, scale, B, junk);
        SDiv ( U, scale, R, junk);
    } else R=U;
}
```

Сложность операции деления :

Функция времени вычислений:

$T_{DIV}(A,B) = O [L(B) * \{L(A) - L(B) + 1\}]$,

где A – делимое и B – делитель (списки разрядов длинных целых чисел);
L(A), L(B) – размеры (длины) списков A и B.

Количество ячеек (разрядов) списков результата:

Частное: $L(Q) \le L(A) - L(B) + 1$ Остаток: $L(R) \le L(B)$

План лекции: тема подраздела

- Целочисленная арифметика
- **Полиномиальная арифметика**

Полиномиальная арифметика

Постулаты

1) Объекты вычислений в полиномиальной арифметике – полиномы степени n (где n – короткое целое число) от одной и нескольких переменных с целочисленными коэффициентами (причём коэффициенты – длинные целые числа).

2) Свойства целых чисел (СС, разрядность, знак, значение), применяемых в полиномиальной арифметике, и структуры данных для их представления аналогичны используемым в целочисленной арифметике.

3) Структура данных для представления объектов вычислений в полиномиальной арифметике – список (элементы списка – это переменные, степени и коэффициенты, а также, возможно, другие списки).

4) Структуры данных для представления полинома $P(x_1, x_2, \ldots x_N)$ и уравнения вида $P(x_1, x_2, \ldots x_N) = 0$ идентичны.

5) Алгоритмы выполнения операций полиномиальной арифметики являются обобщениями алгоритмов выполнения операций целочисленной арифметики и алгоритмов символьных вычислений.

Полиномиальная арифметика

Списочные представления полиномов - 1

Полином от одной переменной P (x) степени n:

$P(x) = (x, c_r, e_r, c_{r-1}, e_{r-1}, \ldots c_1, e_1), \qquad r \geq 1$

Целые ненулевые ($c_i \neq 0$) коэффициенты представлены списками:

$c_i = (c_{i1}, c_{i2}, \ldots c_{imi})$, $m_i \geq 1$

Показатели e_i располагаются в порядке убывания:

$e_r > e_{r-1} > \ldots > e_1$

Степень полинома P (x) равна $n = e_r$.

Знак полинома P (x) совпадает со знаком c_r.

Полиномиальная арифметика

Списочные представления полиномов - 2

Замечание 1. Полином от одной переменной P (x) степени n ≥ 0 можно представить несколькими способами.

Например, следующим образом:

$P(x) = (x, n, c_n, c_{n-1}, \ldots c_0)$

В таком представлении указываются все коэффициенты (в том числе, нулевые).

Замечание 2. Полином P (x) = 0 представляется пустым списком.

Полиномиальная арифметика

Списочные представления полиномов - 3

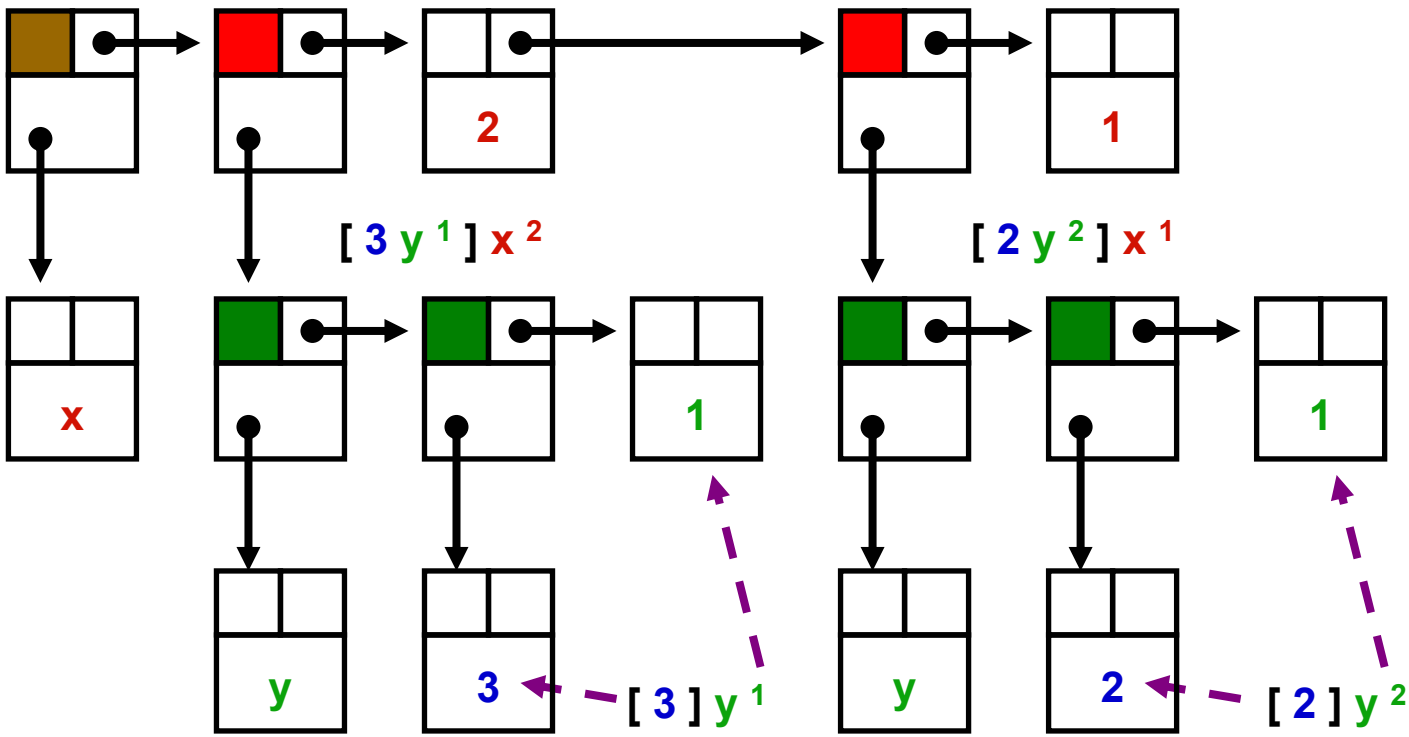
Полином от нескольких переменных $P(x_1, x_2, \ldots x_q)$ представляется в рекурсивной канонической форме.

Рекурсивная каноническая форма – это представление полинома от q переменных в виде полинома от одной переменной x_q с коэффициентами c_i, являющимися полиномами от (q-1) переменных: $x_1, x_2, \ldots x_{q-1}$.

Полиномиальная арифметика

Списочные представления полиномов - 4

Пример. $P(x, y) = 3x^2y + 2xy^2 = + 3\ x^2\ y^1 + 2\ x^1\ y^2$



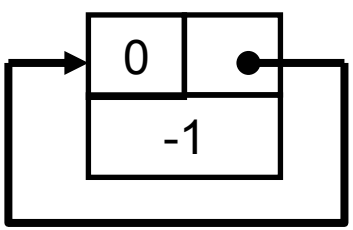
Полиномиальная арифметика

Списочные представления полиномов - 5

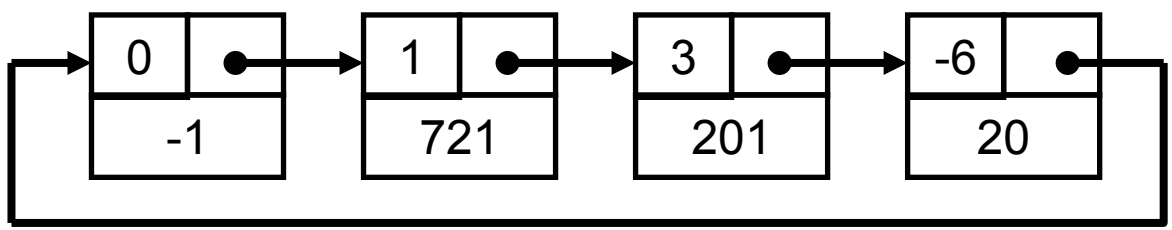
Замечание 3. Полином от нескольких переменных $P(x_1, x_2, \ldots x_q)$ можно представить несколькими способами.

Например, в виде циклического списка:

Нулевой полином:



Полином $P(x, y, z) = x^7y^2z + 3x^2z - 6y^2$:



Полиномиальная арифметика

Списочные представления полиномов - 6

Правила формирования вышеуказанного представления:

- Полиномы состоят из мономов и зависят от переменных.
- Фиксируется старшинство переменных, т.е. порядок их указания в мономах.
- Для каждого монома вычисляется его индекс («свёрнутая степень»). Например, для монома $x^A y^B z^C$ индекс равен

  $A*10^2 + B*10^1 + C*10^0$.

  Индекс монома определяет его старшинство.
- Мономы в полиноме упорядочиваются по убыванию старшинства.
- Полином представляется циклическим списком. Каждое звено списка – моном с ненулевым коэффициентом. Звено состоит из полей коэффициента монома, индекса монома и указателя на звено следующего по порядку монома.

Полиномиальная арифметика

Сложность выполнения операций - 1

Временная сложность выполнения операций над полиномами оценивается функцией от степеней и длин норм полиномов.

Определение. Пусть имеем полиномиальное уравнение от одной переменной с целыми коэффициентами следующего вида:

$$P(x) = \sum_{0 \le i \le n} c_i \; x^i$$

Тогда определены следующие нормы:

- максимальная (норма с нижним индексом ∞);
- суммарная (норма с нижним индексом 1);
- евклидова.

Полиномиальная арифметика

Сложность выполнения операций - 2

Максимальная норма: $\left| P(x) \right|_{\infty} = \max_{0 \le i \le n} \left(\left| c_i \right| \right)$

Суммарная норма: $\left| P(x) \right|_{1} = \sum_{0 \le i \le n} \left| c_i \right|$

Евклидова норма: $\left| P(x) \right|_{2} = \left(\sum_{0 \le i \le n} \left| c_i \right|^{2} \right)^{1/2}$

Полиномиальная арифметика

Сложность выполнения операций - 3

Замечание 1.

Из определения норм следует:

$$\left| P(x) \right|_{\infty} \le \left| P(x) \right|_{1} \le (n+1) \left| P(x) \right|_{\infty}$$

где n – степень полинома P (x).

Поэтому:

$$L\left[\left| P(x) \right|_{\infty} \right] \sim L\left[\left| P(x) \right|_{1} \right]$$

Полиномиальная арифметика

Сложность выполнения операций - 4

Пусть имеем два полинома $P_1(x)$ и $P_2(x)$ с целыми коэффициентами c_i и d_j степеней m и n соответственно.

Необходимо получить ограничения в виде функций степеней m и n и длин максимальных норм времён, требуемых для вычисления следующих выражений:

- $P_1(x) \pm P_2(x)$ (операции сложения и вычитания полиномов);
- $P_1(x) * P_2(x)$ (операция умножения полиномов);
- Q (x) и R (x), (где степень полинома $R(x) < n$ и $m \geq n$), обладающих свойством делимости: $P_1(x) = P_2(x) * Q(x) + R(x)$ (операция деления полиномов).

Полиномиальная арифметика

Сложность выполнения операций - 5

Пример. Расчёт функции времени вычислений $T_{PADD}(P_1(x), P_2(x))$ для программы сложения двух полиномов $P_1(x)$ и $P_2(x)$.

Слагаемые $P_1(x)$ и $P_2(x)$
– это списки длин m и n соответственно
(учитываем только списки коэффициентов, остальное игнорируем).
Результат сложения – сумма ($P_1(x) + P_2(x)$)
– это список длины $L_{PADD} = 2 * [\max (m, n) + 1]$.
Количество операций сложения коэффициентов – не более max (m, n) + 1.
Напомним, что для любых двух длинных целых чисел A и B имеем:

(1) $\mathbf{T_{ADD}(A, B) = O\{\max[L(A), L(B)]\}}$

В наихудшем случае все коэффициенты полиномов $P_1(x)$ и $P_2(x)$ равны максимальным коэффициентам, т.е. $|P_1(x)|_\infty$ и $|P_2(x)|_\infty$ соответственно.
Таким образом, одно сложение коэффициентов выполняется за время:

(2) $\mathbf{T_{ADD}(c_i, d_j) = O\{\max[L|P_1(x)|_\infty, L|P_2(x)|_\infty]\}}$

Очевидно, что произведение величин **(1)** и **(2)** – это искомая оценка T_{PADD}

Полиномиальная арифметика

Сложность выполнения операций – 6

Выводы :

Функция времени вычислений
аддитивных операций над полиномами:

$$T_{PADD}\left[P_1(x), P_2(x)\right] = O\left(\left[\max(m,n)+1\right] * \max\left\{L\left[\left|P_1(x)\right|_\infty\right], L\left[\left|P_2(x)\right|_\infty\right]\right\}\right)$$

Функция времени вычислений
мультипликативных операций над полиномами:

$$T_{PMUL}\left[P_1(x), P_2(x)\right] = O\left\{(m+1) * (n+1) * L\left[\left|P_1(x)\right|_\infty\right] * L\left[\left|P_2(x)\right|_\infty\right]\right\}$$

Спасибо за внимание !

Вопросы ?