Информатика

Медведев Михаил Геннадиевич
Кандидат физико-математических наук, доцент факультета кибернетики Киевского национального университета имени Тараса Шевченко.

Расширенный алгоритм Евклида

Описывается расширенный алгоритм Евклида и рассматриваются его приложения к решению олимпиадных задач. Приводятся алгоритмы решения линейных сравнений и диофантовых уравнений.

Алгоритм вычисления наибольшего общего делителя (НОД) был открыт древнегреческими математиками и известен как алгоритм «взаимного вычитания». И хотя упоминание об этом алгоритме имеется ещё у Аристотеля, впоследствии его стали называть алгоритмом Евклида. Что такое наибольший общий делитель, его свойства и методы вычисления рассмотрены в [1].

Напомним, что НОД двух чисел можно вычислить согласно следующей рекуррентности:

$$\text{НОД}(a,b)=\begin{cases}a, b=0\\ \text{НОД}(b, a \bmod b), b\neq 0\end{cases} \quad (1)$$

На языке Си процедура вычисления НОД имеет вид:

```
int gcd(int a, int b)
{
  if (b == 0) return a;
  return gcd(b, a % b);
}
```

Алгоритм Евклида можно *расширить* для нахождения по заданным a и b таких целых x и y, что $ax+by=d$, где d – наибольший общий делитель a и b.

Лемма. Пусть для положительных целых чисел a и b $(a>b)$ известны

$$d=\text{НОД}(a,b)=$$
$$=\text{НОД}(b, a \bmod b),$$

а также числа x' и y', для которых

$$d=x'*b+y'*(a \bmod b).$$

Тогда значения x и y, являющиеся решениями уравнения

$$ax+by=d,$$

находятся из соотношений

$$x = y',\ y = x' - y' * \left\lfloor \frac{a}{b} \right\rfloor. \qquad (2)$$

Через $\lfloor x \rfloor$ здесь обозначена целая часть числа x.

Доказательство. Поскольку

$$a \bmod b = a - \left\lfloor \frac{a}{b} \right\rfloor * b, \text{ то}$$

$$d = x' * b + y' * (a - \left\lfloor \frac{a}{b} \right\rfloor * b) =$$

$$= y' * a + (x' - y' * \left\lfloor \frac{a}{b} \right\rfloor) * b =$$

$$= x * a + y * b,$$

где обозначено

$$x = y',\ y = x' - y' * \left\lfloor \frac{a}{b} \right\rfloor.$$

Функция **gcdext(int a,int b, int *d, int *x, int *y)**, приведённая ниже, по входным числам a и b находит $d = \text{НОД}(a, b)$ и такие x, y, что $d = a * x + b * y$. Для поиска неизвестных x и y необходимо рекурсивно запустить функцию **gcdext(b, a mod b, d, x, y)** и пересчитать значения x и y по выше приведённой формуле. Рекурсия заканчивается, когда $b = 0$.

При $b = 0$

$\text{НОД}(a, 0) = a$ и $a = a * 1 + 0 * 0$,

поэтому полагаем $x = 1, y = 0$.

```
void gcdext(int a,int b, int *d, int *x, int *y)
{
  int s;
  if (b == 0)
  {
    *d = a; *x = 1; *y = 0;
    return;
  }
  gcdext(b,a % b,d,x,y);
  s = *y;
  *y = *x - (a / b) * (*y);
  *x = s;
}
```

Для ручного выполнения расширенного алгоритма Евклида удобно воспользоваться таблицей с четырьмя столбцами (как показано ниже в примере), соответствующих значениям a, b, x, y. Процесс вычисления разделим на три этапа.

а) Сначала вычислим НОД (a, b), используя первые два столбца таблицы и соотношение (1). В последней строке в колонке a будет находиться значение $d = \text{НОД}(a, b)$.

б) Занесём значения 1 и 0 соответственно в колонки x и y последней строки.

в) Будем заполнять последовательно строки для колонок x и y снизу вверх. Значения x и y в последнюю строку уже занесены на этапе *б*). Считая, что в текущей заполненной строке уже находятся значения (x', y'), мы при помощи формул (2) будем пересчитывать и записывать значения (x, y) в соответствующие ячейки выше стоящей строки.

1. Расширенный алгоритм Евклида. Найдём решение уравнения $5x + 3y = 1$. Вычисление НОД (5, 3) и нахождение соответствующих x, y произведём в таблице:

	a	b	x	y
а)	5	3	−1	2
	3	2	1	−1
	2	1	0	1
	1	0	1	0

в)

б)

Порядок и направление вычислений указаны стрелками и буквами.

Из таблицы находим, что

$$НОД(5, 3) = 5 * (-1) + 3 * 2 = 1,$$

то есть $x = -1, y = 2$.

Рассмотрим, например, как получены значения (x, y) для первой строки. Со второй строки берём значения $(x', y') = (1, -1)$. По формулам (2) получим:

$$x = y' = -1,$$

$$y = x' - y' * \left\lfloor \frac{a}{b} \right\rfloor = 1 - (-1) * \left\lfloor \frac{5}{3} \right\rfloor = 1 + 1 = 2.$$

2. Игра с округлением вниз и вверх [Вальядолид, 10673]. Для двух целых чисел x и k всегда существуют такие целые p и q, что

$$x = p * \left\lfloor \frac{x}{k} \right\rfloor + q * \left\lceil \frac{x}{k} \right\rceil.$$

По заданным x и k необходимо найти хотя бы одну такую пару p и q.

Вход. Первая строка содержит количество тестов t $(1 \le t \le 1000)$. Каждая следующая строка содержит два положительных целых числа x и k $(x, k < 10^8)$.

Через $\lceil x \rceil$ обозначено наименьшее целое, не меньшее x.

Выход. Для каждого теста вывести в отдельной строке два числа: p и q. Если таких пар существует несколько, то вывести одну из них. Значения $p\left\lfloor \frac{x}{k} \right\rfloor$ и $q\left\lceil \frac{x}{k} \right\rceil$ помещаются в 64-битовый целый тип.

Пример входа	Пример выхода
3	-5 5
5 2	0 2
40 2	0 4
46	

Решение. Если x нацело делится на k, то $\left\lfloor \frac{x}{k} \right\rfloor = \left\lceil \frac{x}{k} \right\rceil = x/k$. Выбрав $p = 0$, $q = k$, получим:

$$0 * (x/k) + k * (x/k) = x.$$

Пусть x не делится на k. Если

$$n=\left\lfloor \frac{x}{k} \right\rfloor, \text{ то } m=\left\lceil \frac{x}{k} \right\rceil = n+1$$

Поскольку

$$\text{НОД}(n,m)=\text{НОД}(n,n+1)=1,$$

то исходя из расширенного алгоритма Евклида, существуют такие целые t и u, что $1=tn+um$. Помножив равенство на x, получим $x=xtn+xum$, откуда $p=xt$, $q=xu$.

Пример. Для первого теста ($x=5$, $k=2$) имеет место соотношение:

$$5=(-5)^*\left\lfloor \frac{5}{2} \right\rfloor+5^*\left\lceil \frac{5}{2} \right\rceil=(-5)^*2+5^*3.$$

Реализация. При вычислении используем 64-битовый целый тип long long.


```
#include <stdio.h>
#include <math.h>
typedef long long i64;

i64 tests, x, k, g, t, u;
i64 n, m, p, q;

void gcdext(i64 a,i64 b, i64 *d, i64 *x, i64 *y) { ... }

int main(void)
{
  scanf("%d",&tests);
  while(tests--)
  {
    scanf("%lld %lld",&x,&k);
```

Если x делится на k, то устанавливаем $p=0, q=k$.

```
    if (x % k == 0) { p = 0; q = k;}
    else
    {
```

Иначе вычисляем $n=\left\lfloor \frac{x}{k} \right\rfloor$, $m=\left\lceil \frac{x}{k} \right\rceil$ и запускаем на числах n и m расширенный алгоритм Евклида. Он находит такие t и u, что $1=tn+um$. Далее вычисляем $p=xt$, $q=xu$ и выводим результат.

```
      n = (int)floor(1.0 * x / k);
      m = (int)ceil(1.0 * x / k);
      gcdext(n,m,&g,&t,&u);
      p = t * x; q = u * x;
    }
    printf("%lld %lld\n",p,q);
  }
  return 0;
}
```

Линейным сравнением называется уравнение вида $ax = b \pmod n$. Оно имеет решение тогда и только тогда, когда b делится на $d = \text{НОД}(a, n)$. Если $d > 1$, то уравнение можно упростить, заменив его на $a'x = b' \pmod{n'}$, где $a' = a/d$, $b' = b/d$, $n' = n/d$. После такого преобразования числа a' и n' являются взаимно простыми.

Алгоритм решения уравнения $a'x = b' \pmod{n'}$ со взаимно простыми a' и n' состоит из двух частей:

1. Решаем уравнение $a'x = 1 \pmod{n'}$. Для этого при помощи расширенного алгоритма Евклида ищем решение (x_0, y_0) уравнения $a'x + n'y = 1$. Взяв по модулю n' последнее равенство, получим

$$a'x_0 = 1 \pmod{n'}.$$

2. Умножим на b' равенство $a'x_0 = 1 \pmod{n'}$. Получим $a'(b'x_0) = b' \pmod{n'}$, откуда решением исходного уравнения $a'x = b' \pmod{n'}$ будет $x = b'x_0 \pmod{n'}$.

Лемма. Если $\text{НОД}(k, m) = 1$, то равенство $ak = bk \pmod m$ эквивалентно $a = b \pmod m$.

Доказательство. Из $ak = bk \pmod m$ следует, что $(a - b) * k$ делится на m. Но поскольку k и m взаимно просты, то $a - b$ делится на m, то есть $a = b \pmod m$.

Пример. Решить уравнение $18x = 6 \pmod 8$.

Значение $d = \text{НОД}(18, 8) = 2$ является делителем 6, поэтому решение существует. После упрощения получим уравнение $9x = 3 \pmod 4$. Что согласно лемме, эквивалентно $3x = 1 \pmod 4$. Решая уравнение $3x + 4y = 1$ с помощью расширенного алгоритма Евклида, получим $x = -1$, $y = 1$. Откуда $x = -1 \pmod 4 = 3$. То есть $x = 3$ будет как решением уравнения $3x = 1 \pmod 4$, так и уравнения $18x = 6 \pmod 8$.

Диофантовые уравнения

Диофантовыми называются уравнения вида

$$P(x_1, x_2, \ldots, x_n) = 0,$$

где $P(x_1, \ldots, x_n)$ – многочлен с целыми коэффициентами.

В этой главе рассмотрим алгоритм нахождения решения линейного диофантового уравнения с двумя неизвестными вида $a * x + b * y = c$ (далее для простоты будем опускать знаки умножения и писать просто $ax + by = c$).

Теорема 1. Уравнение $ax + by = c$ имеет решение в целых числах тогда и только тогда, когда c делится на $\text{НОД}(a, b)$.

Теорема 2. Если пара (x_0, y_0) является решением уравнения $ax + by = c$, то всё множество его решений (x, y) описывается формулой:

$$x = x_0 + kb,$$
$$y = y_0 - ka,$$

где $k \in Z$. Очевидно, что если $ax_0 + by_0 = c$, то $a(x_0 + kb) + b(y_0 - ka) = c$ для любого целого k.

Для нахождения частичного решения (x_0, y_0) уравнения $ax + by = c$ следует сначала найти решение (x', y') уравнения $ax + by = d$ (d – наибольший общий делитель a и b) при помощи расширенного алгоритма Евклида, после чего умножить его на c/d. То есть

$$x_0 = x' * c / d,$$
$$y_0 = y' * c / d$$

Пример. Найти множество решений уравнения $5x + 3y = 7$.

1. Уравнение имеет решение, так как 7 делится на НОД$(5, 3) = 1$.

2. Находим решение уравнения $5x' + 3y' = 1$ при помощи расширенного алгоритма Евклида:

$$(x', y') = (-1, 2).$$

3. Находим решение (x_0, y_0) исходного диофантового уравнения:

$$x_0 = -1 * 7 / 1 = -7,$$
$$y_0 = 2 * 7 / 1 = 14.$$

Согласно теореме 2, множество решений исходного диофантового уравнения имеет вид:

$$(x, y) = (-7 + 3k, 14 - 5k)$$

3. Найти правильный размен [Вальядолид, 10548]. В древние времена люди вместо денег обменивались товарами. В наличии имеются два типа товаров: A и B. Покупатель желает приобрести товар на сумму $c > 0$. Стоимости товаров A и B соответственно равны a и b. Если одно из значений a или b отрицательно, то продавец может этим товаром давать сдачу. Одновременно отрицательными a и b быть не могут. Сколькими способами покупатель может приобрести товар

на сумму c, если это возможно?

Вход. Первая строка содержит количество тестов n ($0 < n < 1001$). Каждая следующая строка содержит три числа a, b и c ($0 < |a|, |b|, |c| < 2^{31}$).

Выход. Для каждого теста вывести количество комбинаций, которыми покупатель может приобрести товар ровно на сумму c. Если приобрести товар невозможно, то вывести сообщение «Impossible». Если число комбинаций бесконечно, то вывести «Infinitely many solutions».

Пример входа	Пример выхода
3	3
2 3 17	Infinitely many solutions
2 -3 5	Impossible
10 36 7	

Решение. Если покупатель приобретёт x штук товара A и y штук товара B, заплатив при этом сумму c, то получим уравнение $ax+by=c$. Уравнение имеет решения тогда и только тогда, когда c делится на НОД(a,b).

Если одно из значений a или b отрицательно, то уравнение $ax+by=c$ имеет бесконечно много решений. Действительно, если (x_0, y_0) – решение, то пара (x_0+kb, y_0-ka) будет также решением для любого целого отрицательного k, для которого $y_0-ka\ge 0$ (если $b<0$), или для любого натурального k, для которого $x_0+kb\ge 0$ (если $a<0$).

Если решение существует и оба значения a и b положительны, то сокращаем a, b, c на их общий делитель и при помощи расширенного алгоритма Евклида находим x', y', для которых $ax'+by'=1$. Помножив последнее равенство на c и положив $x_0=cx'$ и $y_0=cy'$, получим пару (x_0, y_0), являющуюся решением уравнения $ax+by=c$.

Из теоремы 2 следует, что все решения уравнения имеют вид (x_0+kb, y_0-ka), где k – целое число. Поскольку в задаче решения должны быть неотрицательными, то имеют место следующие неравенства:

$$x_0+kb\ge 0,\ \ y_0-ka\ge 0.$$

Откуда имеем следующие ограничения:

$$k\ge\lceil -x_0/b\rceil,\ \ k\le\lfloor y_0/a\rfloor.$$

Количество решений уравнения $ax+by=c$, для которых $x\ge 0$, $y\ge 0$, равно $\lfloor y_0/a\rfloor-\lceil -x_0/b\rceil+1$.

Пример. Рассмотрим первый тест, где следует найти количество решений уравнения $2x+3y=17$ в целых неотрицательных числах. Числа 2 и 3 взаимно простые, находим x' и y', для которых $2x'+3y'=1$. Из расширенного алгоритма Евклида имеем: $x'=-1$, $y'=1$. Помножив их на 17, получим $x_0=17x'=-17$, $y_0=17y'=17$. Имея частичное решение $(x_0, y_0)=(-17,17)$, можно описать множество всех решений исходного уравнения согласно теореме 2: $x=-17+3k$, $y=17-2k$. Они должны быть неотрицательными, следовательно, $-17+3k\ge 0$, $y=17-2k\ge 0$. Или то же самое, что $3k\ge 17$, $17\ge 2k$. Откуда $k\ge\lceil 17/3\rceil=6$, $k\le\lfloor 17/2\rfloor=8$.

Объединяя ограничения на k, получим: $6 \le k \le 8$. То есть существуют 3 варианта размена, которые можно получить, подставив в общее решение значения $k = 6, 7, 8$. Например, при $k = 6$ получим решение (1, 5), при $k = 7$ решение (4, 3), а при $k = 8$ решением будет пара (7, 1).

Для второго теста имеем уравнение $2x - 3y = 5$. Одним из его решений будет $x_0 = 1$, $y_0 = -1$. Всё множество решений описывается формулой: $x = 1 - 3k$, $y = -1 - 2k$. Для всех отрицательных k значения x и y будут положительными и удовлетворять условию задачи.

Для третьего теста решений не существует, так как для любых натуральных x и y значение $10x + 36y$ всегда чётно и не может равняться 7 (7 не делится на НОД $(10, 36) = 2$).

Реализация. При вычислении используем 64-битовый целый тип long long. Для простоты использования переопределим тип long long на i64.

```
#include <stdio.h>
#include <math.h>

typedef long long i64;
i64 i, n, a, b, c;
i64 temp, d, x, y;
i64 kmin, kmax, res;

i64 gcd(i64 a, i64 b)
{
  return (!a) ? b : gcd(b % a, a);
}

void gcdext(i64 a,i64 b, i64 *d, i64 *x, i64 *y) { ... }

int main(void)
{
  scanf("%lld",&n);
  for(i = 0; i < n; i++)
  {
    scanf("%lld %lld %lld",&a,&b,&c);
```

Вычисляем наибольший общий делитель d чисел a и b. Если c не делится на d, то выводим сообщение о невозможности осуществления размена.

```
    d = gcd(a,b);
    if (c % d > 0)
    {
      printf("Impossible\n"); continue;
    }
```

Если a или b отрицательны, то существует бесконечно много вариантов выполнения размена.

```
    if ((a < 0) || (b < 0))
    {
      printf("Infinitely many solutions\n"); continue;
    }
```

Сокращаем входные значения a, b, c на их общий делитель d. Запускаем процедуру расширенного алгоритма Евклида и находим пару (x, y) – решение уравнения $ax + by = c$.

```
    a /= d; b /= d; c /= c;
    gcdext(a,b,&d,&x,&y);
    x *= c; y *= c;
```

Ищем нижнюю $kmin$ и верхнюю $kmax$ границы для переменной k, откуда и получаем количество решений уравнения $ax + by = c$.

Поскольку a, b, x, y – целые, то чтобы не потерять точность вычислений, помножим дроби $-x/b$ и y/a на действительное число 1.0.

```
    kmin = (i64)ceil(-1.0 * x / b); kmax = (i64)floor(1.0 * y / a);
    res = (i64)kmax - kmin + 1;
```

Если значение res не равно нулю, то выводим его. Иначе печатаем сообщение о том, что произвести размен невозможно.

```
    if (res) printf("%lld\n",res); else printf("Impossible\n");
  }
  return 0;
}
```

В статье рассмотрено решение ряда олимпиадных задач, требующих реализации расширенного алгоритма Евклида. Условия задач взяты со страницы университета Вальядолид **http://acm.uva.es/problemset**, посвящённой олимпиадному программированию.

Список литературы

1. *Медведев М.Г.* Наибольший общий делитель //Потенциал. – 2008. – №10.

2. *Винокуров Н.А., Ворожцов А.В.* Практика и теория программирования, книга 2. – М: Физматкнига, 2008 - 288 с.