МОДУЛЬНАЯ АРИФМЕТИКА

В некоторых приложениях удобно выполнять арифметические операции над целыми числами, заданными в так называемом *модульном представлении.* Это представление предполагает, что целое число представлено вычетами (остатками) по модулям из множества попарно взаимно простых чисел. Вычет числа a по модулю m обозначают $a \bmod m$. Например, $34 \bmod 5 = 4$; $-8 \bmod 3 = -2 \bmod 3 = 1$.

В этом разделе под размерностью задачи мы будем понимать не количество чисел на входе алгоритма, как например, в задаче сортировки, а количество битов, использованных для записи числа. Алгоритм, получающий на вход целые числа $a_1, a_2, ..., a_k$ называется *полиномиальным*, если его время работы ограничено многочленом от $\log_2 a_1$, $\log_2 a_2$, …, $\log_2 a_k$, т.е. многочленом от длин исходных данных (в двоичной системе счисления).

Если p_0, p_1, ..., p_{k-1} – это k попарно взаимно простых чисел и $p = \prod_{i=0}^{k-1} p_i$, то любое целое число u, такое что $0 \le u \le p-1$, можно однозначно представить множеством его вычетов u_0, u_1, ..., u_{k-1}, где $u_i = u \bmod p_i$, $0 \le i \le k-1$. Обычно пишут $u \leftrightarrow (u_0, u_1, ..., u_{k-1})$.

Сложение, вычитание и умножение легко выполняются, если их результаты заключены между 0 и $p-1$, т.е. если их можно рассматривать как вычисления по модулю p. Пусть два целых числа u и v заданы их множествами вычетов, т.е.

$u \leftrightarrow (u_0, u_1, ..., u_{k-1})$ и $v \leftrightarrow (v_0, v_1, ..., v_{k-1})$.

Тогда операции сложения, умножения и вычитания определяются следующим образом:

$$u + v \leftrightarrow (w_0, w_1, ..., w_{k-1}), \text{ где } w_i = (u_i + v_i) \bmod p_i; \quad (1)$$

$$u - v \leftrightarrow (x_0, x_1, ..., x_{k-1}), \text{ где } x_i = (u_i - v_i) \bmod p_i; \quad (2)$$

$$uv \leftrightarrow (y_0, y_1, ..., y_{k-1}), \text{ где } y_i = u_i v_i \bmod p_i. \quad (3)$$

Рассмотрим пример. Пусть $p_0 = 2$, $p_1 = 5$, $p_2 = 7$. Получаем $p = 2 \cdot 5 \cdot 7 = 70$. Тогда $6 \leftrightarrow (0,1,6)$, так как $6 \bmod 2 = 0$, $6 \bmod 5 = 1$, $6 \bmod 7 = 6$. Аналогично, $9 \leftrightarrow (1,4,2)$, $54 \leftrightarrow (0,4,3)$. В силу (1) $6 + 9 \leftrightarrow (1,0,1)$, так как $(0+1) \bmod 2 = 1$, $(1+4) \bmod 5 = 0$, $(6+2) \bmod 7 = 1$ но нетрудно убедиться, что $15 \leftrightarrow (1,0,1)$. Согласно (2) и (3) получаем :

$6 - 9 \leftrightarrow (1, 2, 4)$ – это модульное представление числа –3.

$6 \cdot 9 \leftrightarrow (0,4,5)$ – это модульное представление числа 54.

Операция деления в модульной арифметике не определена.

Преимущество модульного представления состоит в том, что арифметические операции могут быть реализованы с меньшими вычислительными затратами, чем при обычном представлении, так как вычисления выполняются независимо для каждого модуля. Следующая теорема доказывает, что соответствие $u \leftrightarrow (u_0, u_1, ..., u_{k-1})$ является взаимно однозначным.

КИТАЙСКАЯ ТЕОРЕМА ОБ ОСТАТКАХ

Пусть $p_0, p_1, ..., p_{k-1}$ *попарно взаимно простые целые числа,* $p = \prod_{i=0}^{k-1} p_i$ *и* $u_i = u \bmod p_i$. *Тогда соответствие* $u \leftrightarrow (u_0, u_1, ..., u_{k-1})$ *между целыми числами и в*

интервале $[0, p)$ *и наборами вида* $(u_0, u_1, ..., u_{k-1})$, $0 \le u_i < p_i$ *при* $0 \le i < k$, *взаимно однозначно.*

Доказательство.

Очевидно, что для каждого числа u найдется соответствующий k – членный набор, так как в интервале $[0, p)$ заключено ровно p значений переменной u и допустимых k – членных наборов также ровно p, так как $p = \prod_{i=0}^{k-1} p_i$. Достаточно показать, что каждый набор соответствует не более, чем одному числу u.

Допустим, что два числа u и v, $0 \le u < v < p$, соответствуют набору $(u_0, u_1, ..., u_{k-1})$. Тогда разность $u - v$ должна делиться на каждое из чисел p_i, так как все p_i взаимно простые, то разность $u - v$ должна делиться и на p. Но $u \ne v$ и $u - v$ делится на p, т.е. u и v должны отличаться не менее, чем на p, а значит не могут оба принадлежать интервалу $[0, p)$.

Результаты аналогичные результатам для целых чисел, справедливы и для полиномов. Пусть $p_0(x), ..., p_{k-1}(x)$ – это попарно взаимно простые полиномы, $p(x) = \prod_{i=0}^{k-1} p_i(x)$. Тогда любой полином $u(x)$ такой, что $\deg(u(x)) < \deg(p(x))$ можно однозначно представить последовательностью $(u_0(x), u_1(x), ..., u_{k-1}(x))$ остатков от деления $u(x)$ на каждый полином $p_i(x)$. Полином $u_i(x)$ – это тот единственный полином, для которого $\deg(u_i(x)) < \deg(p_i(x))$ и $u(x) = q_i(x) p_i(x) + u_i(x)$ для некоторого полинома $q_i(x)$. Мы в этом случае будем использовать обозначение $u_i(x) = u(x) \bmod p_i(x)$ аналогично модульной записи целых чисел.

По аналогии с целыми числами можно показать, что соответствие $u(x) \leftrightarrow (u_0(x), u_1(x), ..., u_{k-1}(x))$ – взаимно однозначное.

Пример. Пусть $p_0(x) = x - 3$, $p_1(x) = x^2 + x + 1$, $p_2(x) = x^2 - 4$, $p_3(x) = 2x + 2$. Рассмотрим $u(x) = x^5 + x^4 + x^3 + x^2 + x + 1$. Получаем

$u(x) \bmod p_0(x) = 364$,

$u(x) \bmod p_1(x) = 0$,

$u(x) \bmod p_2(x) = 21x + 21$,

$u(x) \bmod p_3(x) = 0$,

таким образом, $u(x) \leftrightarrow (364, 0, 21x + 21, 0)$.

Для того, чтобы можно было пользоваться модульной арифметикой, нужны алгоритмы, осуществляющие переход от позиционного представления к модульному и обратно. Один из методов перехода от позиционного представления к модульному состоит в том, чтобы разделить число u на каждый из модулей p_i, $0 \le i < k$.

Допустим, что каждое из чисел p_i содержит b разрядов в двоичном представлении. Тогда произведение модулей

$$p = \prod_{i=0}^{k-1} p_i$$

требует порядка bk двоичных разрядов для записи двоичного представления, а деление u на каждое из чисел p_i, где $0 \le u < p$, могло бы потребовать k делений kb-битового числа на b-битовое число. Разбив каждое деление на k делений $2b$-битовых чисел на b-битовые, можно перейти к модульному представлению за время $O(k^2 D(b))$, где $D(n)$ –

время деления $2n$-разрядного двоичного целого числа на n-разрядное, как показано в А. Ахо, Дж. Хопкрафт, Дж. Ульман "Построение и анализ вычислительных алгоритмов" $D(n) = O(n \log_2 n \log_2 \log_2 n)$.

Модульное представление числа можно вычислить гораздо быстрее, если использовать следующий метод. Очевидно, что, если нужно найти вычеты, например числа 80 по модулям 3, 5, 7, 11, то можно сначала найти $80 \bmod 15 = 5$ и $80 \bmod 77 = 3$, а затем $5 \bmod 3 = 2$, $5 \bmod 5 = 0$, $3 \bmod 7 = 3$ и $3 \bmod 11 = 3$ вместо вычисления $80 \bmod 3 = 2$, $80 \bmod 5 = 0$, $80 \bmod 7 = 3$ и $80 \bmod 11 = 3$.

Вместо того, чтобы делить число u на каждый из k модулей p_0, $p_1, \ldots, p_{k-1}$, сначала вычисляем произведения модулей: $p_0 p_1$, $p_2 p_3, \ldots, p_{k-2} p_{k-1}$, затем $p_0 p_1 p_2 p_3$, $p_4 p_5 p_6 p_7$, … и т.д. Далее вычисляем вычеты u_1 и u_2 числа u по модулям $p_0 p_1 \ldots p_{k/2-1}$ и $p_{k/2} p_{k/2+1} \ldots p_{k-1}$, соответственно. Теперь задача вычисления $u \bmod p_i$, сведена к двум задачам половинного размера, а именно, $u \bmod p_i = u_1 \bmod p_i$ для $0 \le i < k/2$, и $u \bmod p_i = u_2 \bmod p_i$ для $k/2 \le i < k$. Далее вычисляем вычеты u_{11} и u_{12} числа u_1 по модулям $p_0 p_1 \ldots p_{k/4-1}$ и $p_{k/4} p_{k/4+1} \ldots p_{k/2-1}$ и вычеты u_{21}, u_{22} числа u_2 по модулям $p_{k/2} p_{k/2+1} \ldots p_{3k/4-1}$ и $p_{3k/4} p_{3k/4+1} \ldots p_{k-1}$. Таким образом, в свою очередь каждая из подзадач может быть сведена к двум задачам половинного размера, а именно, $u_1 \bmod p_i = u_{11} \bmod p_i$ для $0 \le i < k/4$, и $u_1 \bmod p_i = u_{12} \bmod p_i$ для $k/4 \le i < k/2$, $u_2 \bmod p_i = u_{21} \bmod p_i$ для $k/2 \le i < 3k/4$ и $u_2 \bmod p_i = u_{22} \bmod p_i$ для $3k/4 \le i < k$. Время выполнения этого алгоритма можно оценить следующим образом:

$$2D(kb/2) + 4D(kb/4) + \ldots + kD(b) \approx$$

$$\approx (2kb/2)\log_2(kb/2)\log_2\log_2(kb/2) + (4kb/4)\log_2(kb/4)\log_2\log_2(kb/4) + \ldots +$$

$$+ (kb)\log_2 b \log_2\log_2 b \le (kb\log_2 k \log_2(kb) - kb(\log_2 k)^2/2)\log_2\log_2(kb) \approx kb\log_2 k \log_2 b$$

Выигрыш по времени по сравнению с делением на каждый модуль приблизительно равен

$$\frac{k^2 b \log_2 b}{kb \log_2 k \log_2 b} \approx \frac{k}{\log_2 k}.$$

Блок-схема быстрого алгоритма нахождения модульного представления числа приведена на Рис. 1. Вход: Модули $p(I)$, $I = 0,1,\ldots,k-1$ и целое число u, выход: вычеты $UM(I)$, $I = 0,1,\ldots,k-1$ целого числа u по модулям $p(I)$. Заметим, что $k = 2^t$.

Пример работы алгоритма.

Пусть $k = 4$, т.е. $t = 2$. Заданы модули p_0, p_1, p_2, p_3.

Вычисляем произведения модулей и запоминаем их в массиве q. Индекс I – номер произведения, индекс J – номер шага.

$q_{00} = p_0$

$q_{10} = p_1$

$q_{20} = p_2$

$q_{30} = p_3$

$q_{01} = q_{00} \cdot q_{10} = p_0 p_1$

$q_{21} = q_{20} \cdot q_{30} = p_2 p_3$

$U_{02} = u$

$U_{01} = U_{02} \bmod q_{01} = u \bmod q_{01}$

$$U_{21} = U_{02} \bmod q_{21} = u \bmod q_{21}$$

$$U_{00} = U_{01} \bmod q_{00}$$
$$U_{10} = U_{01} \bmod q_{10}$$
$$U_{20} = U_{21} \bmod q_{20}$$
$$U_{30} = U_{21} \bmod q_{30}$$

$$UM_0 = U_{00}$$
$$UM_1 = U_{10}$$
$$UM_2 = U_{20}$$
$$UM_3 = U_{30}$$

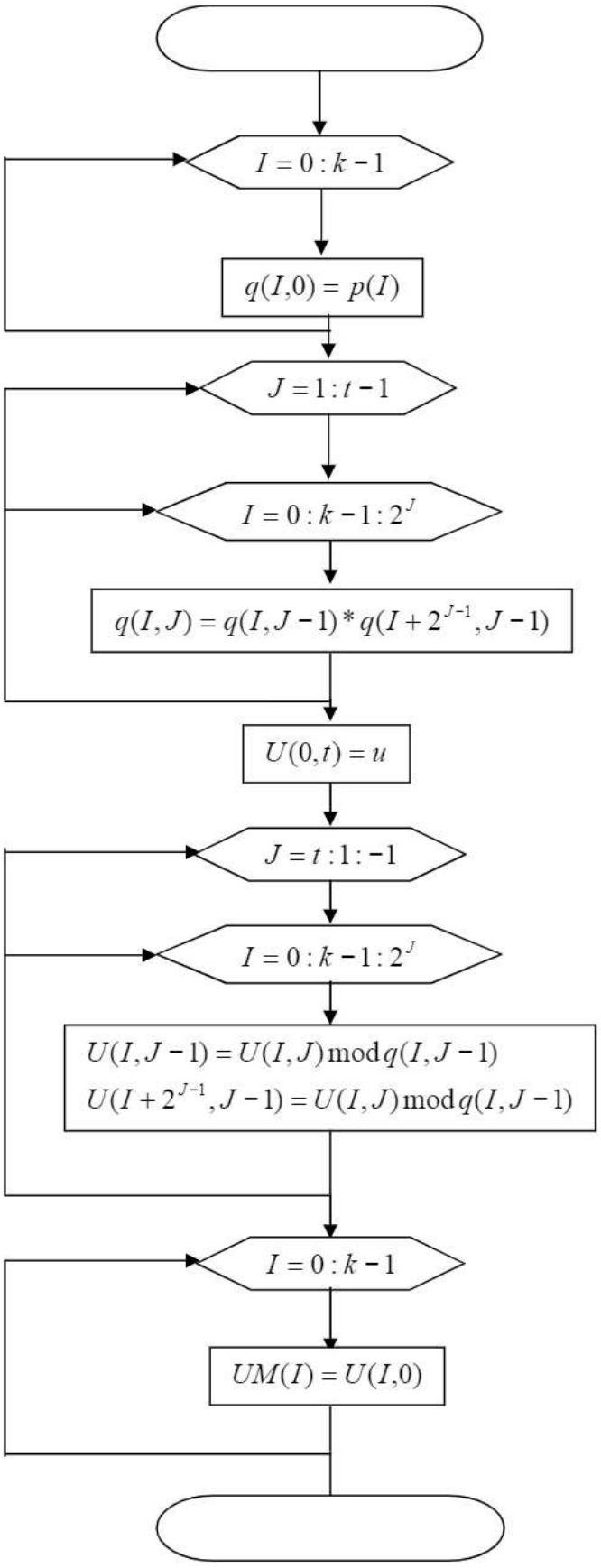

Рис.1 Быстрый алгоритм нахождения модульного представления числа

ПРИМЕНЕНИЕ КИТАЙСКОЙ ТЕОРЕМЫ ОБ ОСТАТКАХ

Рассмотрим задачу преобразования модульного представления целого числа в его позиционное представление. Процесс восстановления числа по его остаткам был известен китайцам более 2000 лет назад, и потому соответствующую теорему называют *китайской теоремой об остатках*. Пусть даны попарно взаимно простые модули p_0, p_1, …, p_{k-1} и вычеты u_0, u_1, …, u_{k-1}, где $k = 2^t$, надо найти такое целое число u, что $u \leftrightarrow (u_0, u_1, ..., u_{k-1})$.

Лемма. Пусть c_i – произведение всех p_j, кроме p_i, т.е. $c_i = p / p_i$, где $p = \prod_{j=0}^{k-1} p_j$.

Пусть $d_i = c_i^{-1} \bmod p_i$, т.е. $d_i c_i = 1 \mod p_i$ и $0 \le d_i < p_i$. Тогда

$$u = \sum_{i=0}^{k-1} c_i d_i u_i \bmod p .$$

Доказательство.

Число c_i делится на p_j при $i \ne j$, так что $c_i d_i u_i = 0 \bmod p_j$, $i \ne j$. Следовательно,

$$\sum_{i=0}^{k-1} c_i d_i u_i = c_j d_j u_j \mod p_j .$$

Так как $c_j d_j = 1 \bmod p_j$, то

$$\sum_{i=0}^{k-1} c_i d_i u_i = u_j \mod p_j .$$

Так как p_i делит p, то эти соотношения выполняются и тогда, когда все арифметические операции производятся по модулю p. Таким образом, Лемма доказана.

Пример. Пусть $p_0 = 3$, $p_1 = 5$, $p_2 = 7$, $p = \prod_{i=0}^{2} p_i = 105$ и модульное представление числа имеет вид $(2,4,1)$.

Восстановим число по его модульному представлению. Для этого вычислим $c_0 = 35$, $c_1 = 21$ и $c_2 = 15$. Нетрудно проверить, что $d_0 = 2$ так как $35 \cdot 2 \bmod 3 = 1$, $d_1 = 1$ так как $21 \bmod 5 = 1$ и $d_2 = 1$, $15 \bmod 7 = 1$. Тогда по теореме получаем

$$u = \sum_{i=0}^{2} c_i d_i u_i \mod p = 35 \cdot 2 \cdot 2 + 21 \cdot 1 \cdot 4 + 15 \cdot 1 \cdot 1 = 239 \bmod 105 = 29.$$

Задача состоит в эффективном вычислении $\sum_{i=0}^{k-1} c_i d_i u_i \bmod p$. В рассмотренном примере d_i мы вычисляли перебором. Ниже будет показано, как это можно сделать непереборным методом.

НАИБОЛЬШИЕ ОБЩИЕ ДЕЛИТЕЛИ И АЛГОРИТМ ЕВКЛИДА

Пусть a_0 и a_1 – положительные целые числа. Положительное целое число g называется *наибольшим общим делителем* чисел a_0 и a_1, обозначается $HOD(a_0, a_1)$, если:

1. g делит a_0 и a_1,
2. Всякий общий делитель a_0 и a_1 делит g.

Можно показать, что для положительных целых a_0 и a_1 такое число g единственно. Например, $HOD(57,33) = 3$.

Алгоритм Евклида для вычисления $HOD(a_0, a_1)$ состоит в вычислении последовательности остатков $a_0, a_1, \ldots, a_k$, где a_i, $2 \le i \le k$ представляет собой ненулевой остаток от деления a_{i-2} на a_{i-1} и a_k нацело делит a_{k-1}, т.е. $a_{k+1} = 0$. Тогда $HOD(a_0, a_1) = a_k$.

Другими словами этот алгоритм вычисляет

$$a_{i+1} = a_{i-1} - q_i a_i \text{ для } 1 \le i < k, \text{ где } q_i = \lfloor a_{i-1} / a_i \rfloor.$$

Рассмотрим пример. Пусть $a_0 = 57$, а $a_1 = 33$

$a_2 = 57 - 33 \cdot 1 = 24$

$a_3 = 33 - 24 \cdot 1 = 9$

$a_4 = 24 - 9 \cdot 2 = 6$

$a_5 = 9 - 6 \cdot 1 = 3$

$a_6 = 6 - 3 \cdot 2 = 0$,

следовательно $HOD(57,33) = 3$.

Блок-схема алгоритма Евклида приведена на Рис. 2.

ТЕОРЕМА. Алгоритм Евклида правильно находит значение $HOD(a_0, a_1)$.

Доказательство. Алгоритм вычисляет $a_{i+1} = a_{i-1} - q_i a_i$ для $1 \le i < k$, где $q_i = \lfloor a_{i-1} / a_i \rfloor$. Так как $a_{i+1} < a_i$ при $i \ge 1$, то алгоритм сходится, т.е. заканчивает работу. Из формулы $a_{i+1} = a_{i-1} - q_i a_i$ следует, что $HOD(a_{i-1}, a_i)$ делит a_{i+1}, но так как $a_{i-1} = a_{i+1} + q_i a_i$, то $HOD(a_{i+1}, a_i)$ делит a_{i-1}. Таким образом, получаем, что $HOD(a_0, a_1) = .$ $HOD(a_1, a_2) = \ldots = HOD(a_{k-1}, a_k) = a_k$.

Алгоритм Евклида можно расширить так, чтобы он находил не только $HOD(a_0, a_1)$, но и целые числа x и y, такие, что $a_0 x + a_1 y = HOD(a_0, a_1)$.

Блок-схема расширенного алгоритма Евклида показана на Рис. 3.

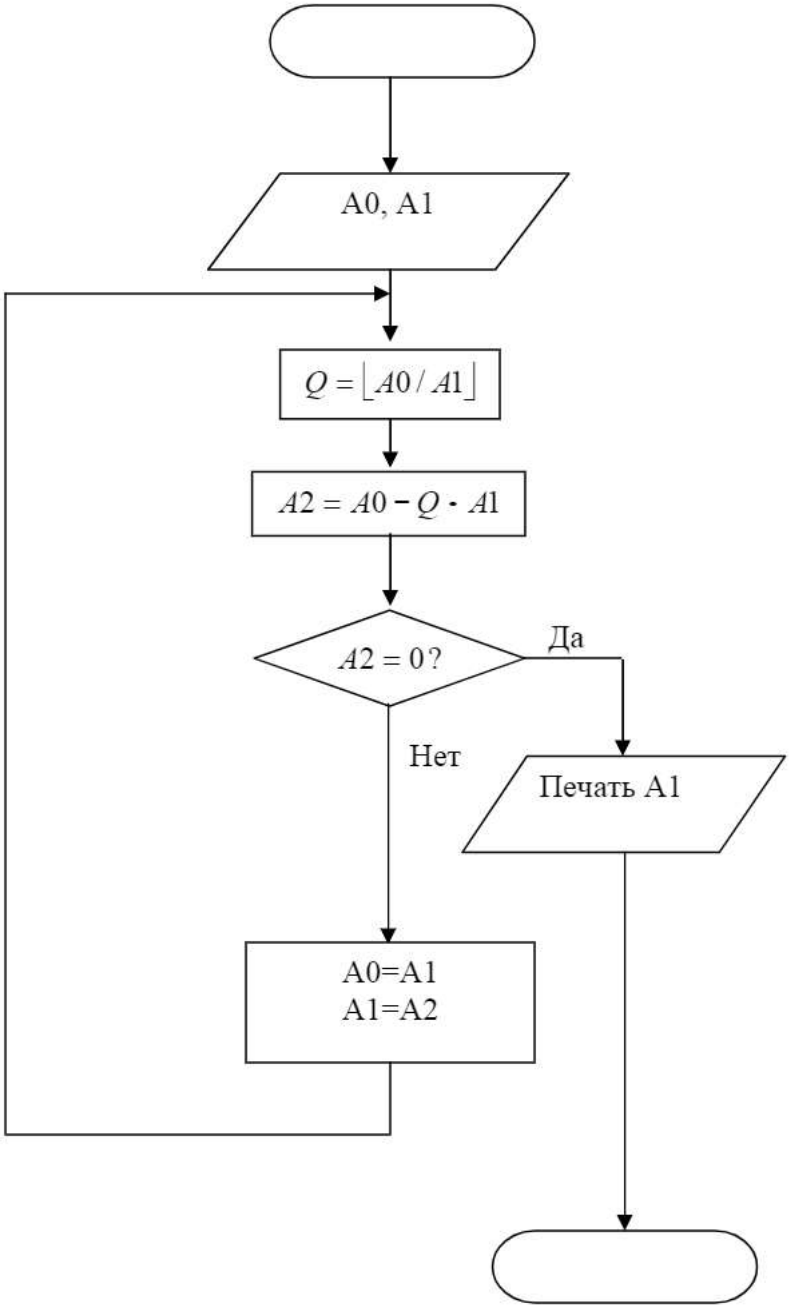

Рис. 2 Алгоритм Евклида

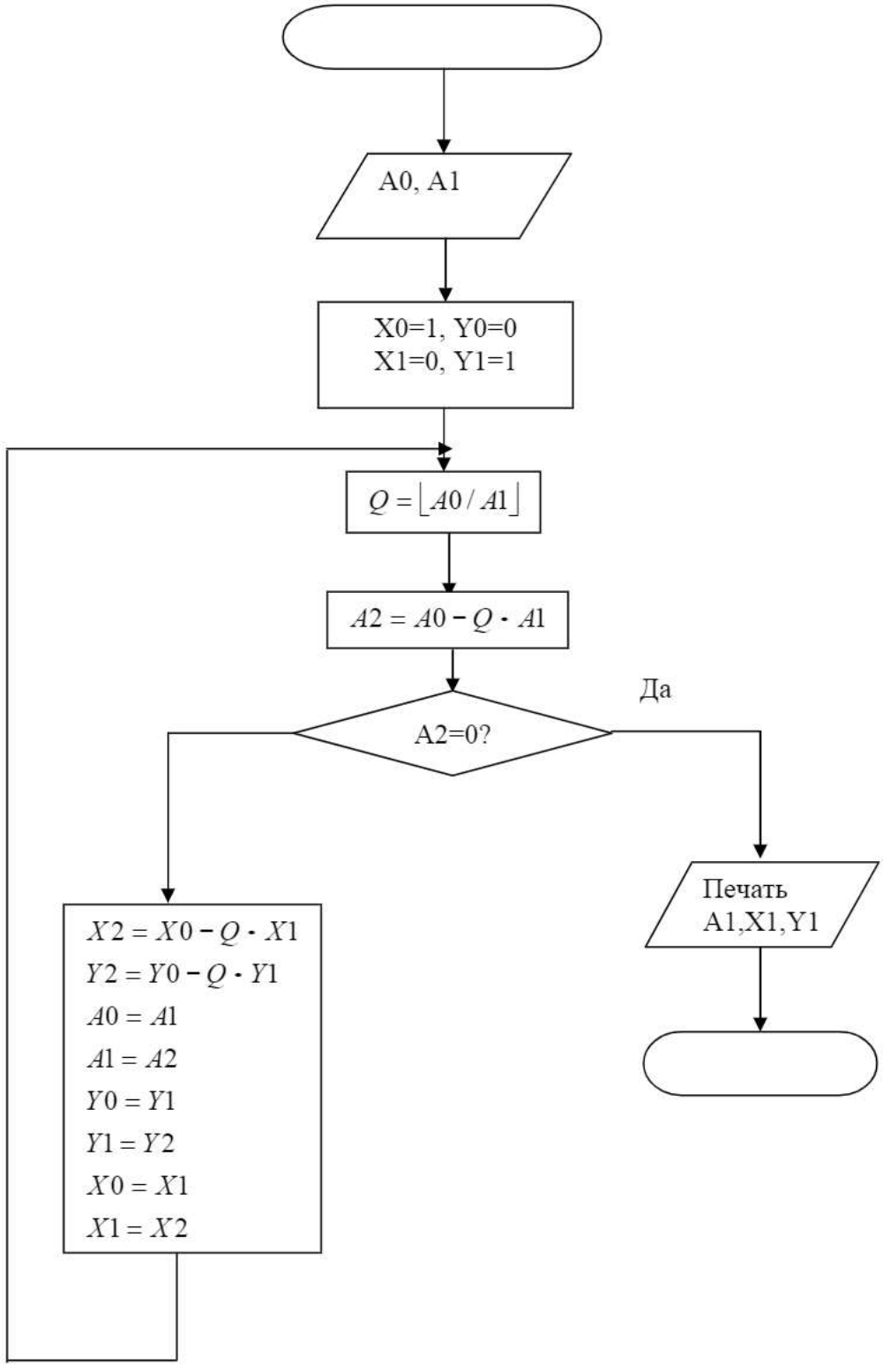

Рис. 3 Расширенный алгоритм Евклида

Пример. Для $a_0 = 57$ и $a_1 = 33$ получаем

$a_2 = a_0 - a_1 q = 57 - 33 \cdot 1 = 24$

$x_2 = x_0 - x_1 q = 1 - 0 \cdot 1 = 1$

$y_2 = y_0 - y_1 q = 0 - 1 \cdot 1 = -1$

$a_3 = 33 - 24 \cdot 1 = 9$

$x_3 = 0 - 1 \cdot 1 = -1$

$y_3 = 1 - (-1) \cdot 1 = 2$

$a_4 = 24 - 9 \cdot 2 = 6$

$x_4 = 1 - (-1) \cdot 2 = 3$

$y_4 = -1 - 2 \cdot 2 = -5$

$a_5 = 9 - 6 \cdot 1 = 3$

$x_5 = -1 - 3 \cdot 1 = -4$

$y_5 = 2 - (-5) \cdot 1 = 7$

$a_6 = 6 - 3 \cdot 2 = 0$.

Таким образом, $HOD(57,33) = 3 = 57 \cdot (-4) + 33 \cdot 7$.

ЛЕММА. В расширенном алгоритме Евклида

$$a_0 x_i + a_1 y_i = a_i \text{ при } i \geq 0. \qquad (4)$$

Доказательство. При $i = 0$

$$a_0 \cdot 1 + a_1 \cdot 0 = a_0,$$

При $i = 1$

$$a_0 \cdot 0 + a_1 \cdot 1 = a_1.$$

Пусть равенство (4) справедливо для $i-1$ и i. Покажем, что (4) справедливо для $i+1$. В соответствии с расширенным алгоритмом Евклида

$$x_{i+1} = x_{i-1} - q x_i, \quad y_{i+1} = y_{i-1} - q y_i.$$

Отсюда следует, что

$$a_0 x_{i+1} + a_1 y_{i+1} = a_0 x_{i-1} + a_1 y_{i-1} - q(a_0 x_i + a_1 y_i) \qquad (5)$$

По предположению $a_0 x_{i+1} + a_1 y_{i+1} = a_{i-1} - q a_i$, но в соответствии с алгоритмом Евклида $a_{i-1} - q a_i = a_{i+1}$, т.е. имеем $a_0 x_{i+1} + a_1 y_{i+1} = a_{i+1}$, лемма доказана.

Выше при рассмотрении вопроса о восстановлении числа по его модульному представлению мы находили числа $d_i = c_i^{-1} \bmod p_i$, $i = 0,...,k-1$. До сих пор мы делали это перебором. Рассмотрим, как расширенный алгоритм Евклида может быть применен для нахождения числа обратного к данному по некоторому модулю.

Уравнение вида $ax = b \bmod p$, где a, b и p – целые числа называют *линейным диофантовым уравнением.* Очевидно, что в нашем случае это уравнение имеет вид

$$ax = 1 \bmod p. \qquad (6)$$

Известно, что в этом частном случае, если $HOD(a, p) = 1$, то уравнение имеет единственное решение, и это решение представляет собой элемент обратный к a по модулю n. Уравнение (6) можно переписать в виде

$$ax + py = 1, \qquad (7)$$

где y – некоторое целое число.

Нетрудно видеть, что если $HOD(a,p)=1$, то (7) представляет собой разложение $HOD(a,p)=1$, которое может быть найдено с помощью расширенного алгоритма Евклида. Таким образом, элемент x обратный к a по модулю p может быть найден с помощью расширенного алгоритма Евклида.

Пример. Пусть $p_0=3$, $p_1=5$, $p_2=7$, $p=\prod_{i=0}^{2} p_i=105$ и модульное представление числа имеет вид $(1,1,2)$.

Восстановим число по его модульному представлению. Для этого вычислим $c_0=35$, $c_1=21$ и $c_2=15$.

Обратные к c_0 элемент d_0 найдем с помощью расширенного алгоритма Евклида.

Положим $a_0=c_0$ и $a_1=p_0$ (примечание: всегда должно выполняться $a_0>a_1$).

$a_2=35-3\cdot 11=2$

$x_2=1-11\cdot 0=1$

$y_2=0-11\cdot 1=-11$

$a_3=3-2\cdot 1=1$

$x_3=0-1\cdot 1=-1$

$y_3=1-(-11)\cdot 1=12$

$a_4=2-1\cdot 2=0$

Отсюда получаем $35\cdot(-1)+11\cdot 12=1$, а искомое $d_0=-1=2 \bmod 3$.

Аналогично находим, $d_1=1$ и $d_2=1$. Окончательно получаем,

$$u=\sum_{i=0}^{2} c_i d_i u_i \bmod p=35\cdot 2\cdot 1+21\cdot 1\cdot 1+15\cdot 1\cdot 2=121 \bmod 105=16.$$