Рамиль Альварес Х.[1], Владимирова Ю.С.[2]

[1]МГУ им. М.В.Ломоносова, ф-т ВМК, ramil@cs.msu.ru

[2]МГУ им. М.В.Ломоносова, ф-т ВМК, vladimirova@cs.msu.ru

ОБ ИЗВЛЕЧЕНИИ КВАДРАТНОГО КОРНЯ В ТРОИЧНОЙ СИММЕТРИЧНОЙ СИСТЕМЕ

КЛЮЧЕВЫЕ СЛОВА

Троичная симметричная система, извлечение квадратного корня.

АННОТАЦИЯ

В статье [1] описаны алгоритмы деления и извлечения квадратного корня типа «цифра за цифрой» в троичной симметричной системе (ТСС). К сожалению, в алгоритме извлечения квадратного корня автором допущена ошибка, которая исправлена при описании алгоритма в [2], в работе [3] приведено описание алгоритма на языке С [4]. Однако эти работы малодоступны, а в интернете [5] приводятся ошибочные алгоритмы деления и извлечения квадратного корня в ТСС. Эти ошибки носят методологический характер и связаны с перенесением с некоторой корректировкой на алгоритмы в ТСС методов и приемов из двоичной или десятичной систем.

Общие правила извлечения квадратного корня

Как и при вычислении квадратного корня в десятичной системе, подкоренное число N разбивается на пары цифр (тритов) влево и вправо от точки. Обозначим через N_i целое число, получаемое из первых слева направо i пар тритов, A_i – целое значение квадратного корня из этого числа, D_i – i-ая пара тритов, d_i – значение i-го трита корня и n – число пар тритов. Если старшая пара состоит из одного трита, добавим к ней слева трит 0. Отметим, что действия производятся над целыми числами, а точка в результате будет после k-го трита, если в подкоренном числе она стояла после k-ой пары.

Если рассмотреть теперь подкоренное число как число с точкой перед первой парой, имеем число в интервале (.166… : .5), корень из которого будет в интервале (.408… : .707…). В ТСС числа, удовлетворяющие $0 < x < .5$ имеют нулевой трит, равный 0, числа удовлетворяющие $.5 < x < 1.5$ – нулевой трит, равный 1, откуда, нулевой трит корня может быть 0 или 1. Таким образом, в ТСС корень из числа, состоящего из n пар тритов, может состоять из $n + 1$ тритов. Например, корень из числа из одной пары тритов 11 (4 в десятичной системе) равен $1\,\bar{1}$ (2 в десятичной системе), числу из двух тритов ($\bar{1}$ в ТСС обозначается –1).

Схему алгоритма извлечения квадратного корня «в столбик» для ТСС можно описать следующими формулами:

$A_0 = d_0$, где $d_0 = 0$ или 1,

$B_0 = -d_0$,

$N_i = A_i^2 + B_i$,

$C_i = B_{i-1}3^2 + D_i$,

$B_i = C_i - (2A_{i-1}3 + d_i)d_i$,

$A_i = A_{i-1}3 + d_i$.

Заметим, что $B_{i-1}3^2 + D_i$ при ручном использовании алгоритма выполняется приписыванием справа к значению B_{i-1} двух тритов пары D_i, а $2A_{i-1}3 + d_i$ и $A_{i-1}3 + d_i$ – трита d_i соответственно к числам $2A_{i-1}3$ и $A_{i-1}3$.

Для вычисления значения квадратного корня приведенную схему необходимо дополнить правилом выбора очередной цифры d_i.

Определение d_0

Если подкоренное число, рассматриваемое как число с точкой перед первой парой, больше .25, корень больше, чем .5, т.е. нулевой трит корня d_0 будет 1. В противном случае корень меньше, чем .5, и нулевой трит корня d_0 будет 0. Значение B_0 в первом случае равно $\bar{1}$, во втором 0, согласно схеме алгоритма. Заметим, что .25 не представляется точно в ТСС и имеет представление

в виде бесконечной последовательности .(1 $\bar{1}$).

Обоснование правила выбора d_i

Для получения правила определения d_i, i = 1, 2, ... покажем, что при d_i – i-ой цифре A_i значение $|B_i|$ минимально из трех возможных. Рассмотрим функцию дискретной трехзначной переменной

$$b_i(x) = C_i - (2\cdot 3A_{i-1} + x)x,$$

$x \in \{\bar{1}, 0, 1\}$. Отметим, что $b_i(d_i) = B_i$, где d_i – i-ая цифра $A_i = [\sqrt{N_i}]$.

Утверждение 1. В ТСС значение

$|b_i(x)| = |C_i - (2\cdot 3A_{i-1} + x)x|, x \in \{-1, 0, 1\}$

достигает минимума при $x = d_i$, i = 1, ..., n.

Доказательство. Отметим, что на i-м шаге ($i \geq 1$) $A_i \geq 1$. В самом деле, при i = 1 старшая пара цифр D_1 подкоренного числа по условию больше нуля, т.е. $1 \leq D_1 \leq 4$. В силу того, что $N_1 = D_1$ и $A_1 = [\sqrt{N_1}]$, $A_i \geq 1$. На следующих шагах, при i = 2, ..., n, $A_i = A_{i-1}3 + d_i$, откуда $A_i > 1$.

Оценим $|B_i|$. Из того, что

$$A_i = [\sqrt{N_i}],$$

следует:

$$B_i = N_i - A_i^2 = N_i - [\sqrt{N_i}]^2.$$

Для того чтобы оценить модуль разности $|N_i - [\sqrt{N_i}]^2|$, представим $\sqrt{N_i} = A_i + z$, где z – дробная часть $\sqrt{N_i}$:

$$N_i = (A_i + z)^2 = A_i^2 + 2A_iz + z^2,$$

откуда:

$$B_i = N_i - A_i^2 = 2A_iz + z^2 \quad (1)$$

Так как в ТСС $|z| < 0.5$,

$$|B_i| < A_i + 0.25. \quad (2)$$

Отметим также, что поскольку

$$A_i^2 = (3A_{i-1})^2 + (2\cdot 3A_{i-1} + d_i)d_i,$$

то B_i можно представить как:

$$B_i = N_i - (3A_{i-1})^2 - (2\cdot 3A_{i-1} + d_i)d_i.$$

Докажем, что $|B_i| = |b_i(d_i)|$ – минимально.

Предположим обратное: $|B_i| = |C_i - 2\cdot(3A_{i-1} + d_i)d_i|$ – не минимально, $d_i \in \{\bar{1}, 0, 1\}$ является i-ой цифрой A_i. Тогда существует число $t = \pm 1, \pm 2$, такое что $(d_i + t) \in \{\bar{1}, 0, 1\}$ и

$$|b_i(d_i + t)| < |B_i|.$$

Преобразуем:

$$\begin{aligned} b_i(d_i + t) &= N_i - (3A_{i-1})^2 - (2\cdot 3A_{i-1} + d_i + t)(d_i + t) = \\ &= N_i - (3A_{i-1})^2 - ((2\cdot 3A_{i-1} + d_i)d_i + (2\cdot 3A_{i-1} + t)t + 2d_it) = \\ &= B_i - ((2\cdot 3A_{i-1} + t)t + 2d_it) = \\ &= B_i - 2\cdot(3A_{i-1} + d_i)t - t^2 = B_i - (2A_it + t^2). \end{aligned}$$

Обозначив

$$R = 2A_it + t^2,$$

получим

$$b_i(d_i + t) = B_i - R.$$

По предположению, $|B_i|$ не минимально, т.е.

$$|b_i(d_i + t)| = |B_i - R| < |B_i|. \quad (3)$$

Нетрудно убедиться, что при $R > 0$ условие $|B_i - R| < |B_i|$ выполняется, если $R < 2B_i$, а при $R < 0$, если $R > 2B_i$.

При $A_i > 1$, если $t > 0$, то и $R > 0$, а если $t < 0$, то и $R < 0$. При $A_i = 1$ это верно не при всех значениях t. Рассмотрим в связи с этим раздельно случаи $A_i > 1$ и $A_i = 1$.

Рассмотрим случай $A_i > 1$.

Допустим, что $t > 0$, тогда и $R > 0$. Согласно предположению (3), при $R > 0$ имеет место $R < 2B_i$.

Используя оценку (2), получим

$$R < 2B_i < 2A_i + 0.5. \quad (4)$$

С другой стороны, т.к. $R = 2A_it + t^2$, наименьшее значение $R > 0$

$R = 2A_i + 1 > 2A_i + 0.5$,

что несовместимо с (4).

Допустим, что $t < 0$, тогда и $R < 0$, и согласно предположению (3),

$$R > 2B_i. \qquad (5)$$

Отсюда следует, что $B_i < 0$. Отметим также, что при $t < 0$, т.е. $t = -1$ или $t = -2$, i-ая цифра результата $d_i \in \{0, 1\}$.

Из (2) следует оценка для $2B_i$:

$$2B_i > -2A_i - 0.5.$$

В силу того, что B_i и A_i – целые числа, полученная оценка сводится к:

$$2B_i \geq -2A_i. \qquad (6)$$

Выясним, при каких условиях неравенство (5) и (6) совместимы, т.е. возможно

$$-2A_i \leq 2B_i < R.$$

R может принимать одно из двух значений: $R = -4A_i + 4$ (при $t = -2$) или $R = -2A_i + 1$ (при $t = -1$).

При $A_i > 1$, $t = -2$ выполнено $-4A_i + 4 \leq -2A_i$, т.е.

$R \leq -2A_i \leq 2B_i$,

что несовместимо с (5).

При $t = -1$ значение $R = -2A_i + 1$, откуда (5) выполнено только в случае, если $2B_i = -2A_i$, т.е. $B_i = -A_i$. В силу (1)

$$B_i = 2A_iz + z^2.$$

В рассматриваемом случае

$$2A_iz + z^2 = -A_i,$$

откуда

$$A_i = \frac{-z^2}{2z+1}. \qquad (7)$$

По предположению, $A_i > 1$, откуда число в правой части (7) положительно, если $2z + 1 < 0$ или

$$z < -.5,$$

что невозможно в силу того, что z – дробная часть числа в ТСС, и $|z| < .5$. Таким образом, при $A_i > 1$ и $t < 0$ невозможно $B_i = -A_i$, и предположение, что $R > 2B_i$ неверно.

Рассмотрим случай $A_i = 1$, $i = 1, \ldots, n$. Очевидно, что $A_i = 1$ может быть только при $i = 1$ и $A_0 = 0$. Тогда $d_1 = 1$, и

$$B_1 = C_1 - 2A_0 3d_1 - d_1^2 = D_1 - 1.$$

Поскольку

$$b_1(x) = C_1 - (2 \cdot 3A_0 + x)x = D_1 - x^2, x \in \{-1, 0, 1\},$$

значение $D_1 - 1$ минимально для $b_1(x)$. Утверждение 1 доказано.

Утверждение 2. Докажем, что

1) $|B_i|$ единственный минимум $|b_i(x)|$ при $A_i \geq 1$ ($i = 2, \ldots, n$) и при $A_0 = 1$ ($i = 1$);

2) $|b_i(x)|$ может иметь два минимума при $A_0 = 0$ ($i = 1$), только один из которых соответствует старшему триту результата, а он положительный.

Доказательство. 1) Предположим, что для некоторой цифры $d_i' \neq d_i$, $d_i' \in \{\overline{1}, 0, 1\}$ соответствующее ей значение

$$|b_i(d_i')| = |N_i - (3A_{i-1})^2 - (2 \cdot 3A_{i-1} + d_i')d_i'|$$

также минимально, т.е. $|b_i(d_i')| = |B_i|$. Рассмотрим функцию вещественной переменной

$$f(x) = |b_i(x)| = |N_i - (3A_{i-1})^2 - (2 \cdot 3A_{i-1} + x)x|$$

и исследуем ее на сегменте $[-1; 1]$.

Преобразуем $f(x)$:

$$f(x) = |-x^2 - 2 \cdot 3A_{i-1}x + N_i - (3A_{i-1})^2| = |x^2 + 2 \cdot 3A_{i-1}x - N_i + (3A_{i-1})^2|.$$

Функция $f(x)$ имеет два минимума, равных нулю, в точках

$$x_{1,2} = -3A_{i-1} \pm \sqrt{N_i}.$$

В силу того, что на i-ом шаге при $i = 2, \ldots, n$, $A_{i-1} > 0$ или при $i = 1$, $A_0 = 1$,

$$x_1 = -3A_{i-1} - \sqrt{N_i} < -3 < -1,$$

т.е. точка x_1 лежит вне сегмента $[-1; 1]$. Откуда функция $f(x)$ имеет на сегменте $[-1, 1]$ единственный минимум.

2) Рассмотрим случай $i = 1$, $A_0 = 0$. Функция $f(x) = |x^2 - N_1|$ достигает минимума на концах сегмента $[-1; 1]$. Значение $f(x)$ в точке $x_2 = 1$ соответствует положительному старшему триту

результата $d_i = 1$. Утверждение 2 доказано.

Определение d_i

Согласно доказанным утверждениям, в качестве очередного d_i выбирается троичная цифра, для которой $|b_i(x)| = /C_i - 2\cdot(3A_{i-1} + x)x|$ минимально, т.е. минимальное из трех значений:

$$|b_i(-1)| = |C_i + (2\cdot 3A_{i-1} - 1)|, \quad (8)$$

$$|b_i(0)| = /C_i|, \quad (9)$$

$$|b_i(1)| = |C_i - (2\cdot 3A_{i-1} + 1)|. \quad (10)$$

Знак в выражениях под модулем в (8) и (10) зависит от того $A_{i-1} \geq 1$ или $A_{i-1} = 0$, поэтому рассмотрим эти случаи раздельно.

Отметим, что при $C_i \geq 0$ в случае $A_{i-1} \geq 1$, ($i = 1, 2, \ldots, n$).

$$|C_i + (2\cdot 3A_{i-1} - 1)| > C_i,$$

$$|C_i + (2\cdot 3A_{i-1} - 1)| > |C_i - (2\cdot 3A_{i-1} + 1)|,$$

т.е. (8) не может быть минимальным, и $d_i \neq \overline{1}$.

Аналогично, при $C_i < 0$:

$$|C_i - (2\cdot 3A_{i-1} + 1)| > |C_i|,$$

$$|C_i - (2\cdot 3A_{i-1} + 1)| > |C_i + (2\cdot 3A_{i-1} - 1)|,$$

т.е. (10) не может быть минимальным, $d_i \neq 1$. Таким образом, если d_i отлично от 0, его знак совпадает со знаком C_i.

При $C_i \geq 0$ для определения d_i необходимо сравнивать (9) и (10). Раскрывая модули в этих выражениях, получим, что если $C_i < 3A_{i-1}$, то минимально (9), т.е. $d_i = 0$, а если $C_i > 3A_{i-1}$, то минимально (10), т.е. $d_i = 1$.

Аналогично, при $C_i < 0$ для определения d_i необходимо сравнивать (8) и (9), в результате чего получаем, что если $C_i < -3A_{i-1}$, то минимально (9), и $d_i = 0$, а при $C_i > -3A_{i-1}$ то минимально (8), и $d_i = \overline{1}$.

Рассмотрим случай $i = 1$, $A_0 = 0$. При этом

$$|b_1(0)| = |C_1|,$$

$$|b_1(-1)| = |b_1(1)| = |C_1 - 1|.$$

Согласно схеме алгоритма,

$$C_1 = B_0 3^2 + D_1,$$

где $B_0 = -d_0 = -A_0 = 0$, D_1 – старшая пара подкоренного числа, для которой верно $1 \leq D_1 \leq 4$. Таким образом, $C_1 = D_1$, $1 \leq C_1 \leq 4$ и

$$|C_1| < |C_1 - 1|,$$

откуда $d_i = 1$.

Обобщая, получим следующий критерий выбора очередного d_i: $d_i = 0$ при $|C_i| < 3A_{i-1}$, и $d_i = sign(C_i)$ при $|C_i| > 3A_{i-1}$.

Рассмотрим случай $|C_i| = 3A_{i-1}$, тогда B_i можно представить в виде:

$B_i = C_i - 2A_{i-1}3d + (.5)^2 - .25$,

где $d = .5$ для $C_i > 0$ и $-.5$ для $C_i < 0$. Значение d_i зависит от значения n_i – необработанной части числа N, рассматриваемой как число с точкой перед ее первым тритом. То, что функция квадратного корня возрастающая, позволяет определить значение d_i. При $C_i > 0$ и $n_i > .25$ невычисленная часть корня больше .5, т.е. d_i будет 1. При $C_i < 0$ и $n_i < .25$ невычисленная часть корня меньше $-.5$, т.е. d_i будет $\overline{1}$.

Описание алгоритма

Целое троичное число представляется в виде структуры

struct *trin* {**int** *n*; **char** *trit* [*K*]};

где n – число тритов, а массив *trit* – значение числа: младший трит хранится в элементе с индексом 0, элементы массива принимают значения –1, 0 и 1. Для числа нуль n равно нулю. Параметр алгоритма K – максимальная длина числа.

Вспомогательные функции

В виду простоты реализации функций *sgn*(*w*), *red*(&*res*), *shftb*(&*bi*,*x*,*i*) и *shiftc*(&*y*) их определение на языке С не приводится, даются только словесные описания.

Результатом функции **int** *sgn*(**struct** *trin w*) является знак числа *w*. Если n равно нулю, то результат будет нуль, в противном случае, в соответствии со свойством ТСС и принятым представлением числа, результатом будет значение элемента массива *trit* с индексом $n-1$.

Функция *void red*(**struct** *trin* **res*) приводит число *res* к принятому представлению, т.е. удаляет старшие нулевые триты структуры *res*. Она используется в функциях *shftb*(&*bi*,*x*,*i*) и *sum*3(*x*, *y*, &*res*).

Функция *void shftb*(**struct** *trin *bi*, **struct** *trin x*, **int** *i*) определяет новое значение *bi*. Ненулевое значение *bi* сдвигается влево на два трита. Тритам с индексами один и нуль присваиваются значения тритов структуры *x* с индексами, соответственно, *i* и *i*–1, после чего производится приведение числа к принятому представлению, т.е. обращение к функции *red*().

Функция *void shiftc*(**struct** *trin *y*) сдвигает влево на один трит значение структуры *y* и обнуляет трит с индексом нуль.

Функция *acmp*(*x*, &*y*) сравнивает модуль *x* и *y* (*y* > 0). Если длина *x* больше длины *y*, результат будет 1, а если длина *x* меньше длины *y*, результат будет –1. При равных длинах *x* и *y* сравниваются триты чисел *x* и *y* с учетом знака *x*, начиная со старших, до первого несовпадения, которое определяет результат. Если все триты совпали, то результат – нуль.

```
int acmp(struct trin x, struct trin * y) {
  int i = x.n, sx = sgn(x);
          if (i > y->n) return 1;
          if (i-- == y->n) {
            while (sx * x.trit[i] == y->trit[i] && i-- >= 0);
            if (i<0) return 0;
            if (sx * x.trit[i] > y->trit[i]) return 1;
  }
          return -1;
}
```

Функция *cmp*(*x*, *k*) производит сравнение необработанной части структуры *x* с индексами с *k* по нуль (рассматриваемой как число с точкой перед *k*-ым тритом) с числом .25. Поэтому сравниваются последовательно пары тритов структуры x и пара $1\ \bar{1}$.

```
int cmp(struct trin x, int k)
{
          int i = k;
 while ((i > 0) && (x.trit[i] == 1) && (x.trit[i - 1] == -1))
    i -= 2;
 if ((i > 0)  &&  (x.trit[i] == 1) &&  (x.trit[i - 1] > -1))
    return 1;
 else return -1;
}
```

Функция *sum*3(*x*, *y*, &*res*) производит вычисление по формуле *res* = *x* – 1 – (*y* + *y*)*sign*(*x*). Практически это одновременное сложение трех слагаемых, заметим, что в этом случае, также как и при обычном сложении, перенос возможен только одной единицы (положительной или отрицательной). Максимальное значение суммы тритов будет при совпадении знаков суммируемых тритов и трита переноса и равно ±4, и это дает перенос в следующий трит единицы того же знака (1 или –1). Отметим, что при обычном сложении трех слагаемых начальное значение этого трита (переменная *c*1) равно нулю, в данном случае в связи с формулой вычисления *res* начальное значение трита равно –1.

Заметим, что трехвходовые сумматоры использовались в ЭВМ «Сетунь» [6] в множительном устройстве. Для умножения 18-тритных чисел на первом уровне использовалось 6 трехвходовых сумматоров, на втором – 2 трехвходовых сумматора для суммирования результатов первого уровня, а на третьем уровне – один обычный (двухвходовый) сумматор для суммирования результатов второго уровня. Это позволило для последовательности выполнения операций выполнять умножения за 320 машинных тактов, притом, что сложение выполнялось за 180 тактов.

```
void sum3 (struct trin x, struct trin * y, struct trin * res)
{
          int i; s; sx = sgn(x); c1 = -1; j = max(x.n; y-> n);
          res-> n = j ;
          for (i = 0; i < j; i ++)
 {
          s = c1;
          if (i < x.n) { s += x.trit[i]; }
          if (i < y-> n) { s -= (y-> trit[i] + y-> trit[i]) * sx; }
          if (s < -1) { res-> trit[i] = s + 3; c1 = -1; }
          else
```

```
            if (s > 1) { res-> trit[i] = s - 3; c1 = 1; }
            else {  res-> trit[i] = s; c1 = 0; }
    }
   if (c1 != 0)  { res-> trit[j] = c1; res-> n = j + 1; }
   else red(res);
}
```

Алгоритм извлечения квадратного корня

Функция *sqrt*(*x*, &*y*) производит вычисление квадратного корня числа *x*.

```
void sqrt (struct trin x, struct trin * y)
   int i = x.n; j, sbi, k, cmpbc;
   struct trin bi, cx = x;
   if (i == 0) y->n = 0; else  {
      if (x.trit[i - 1] < 0) printf ("Number less 0\n");
      else {
               if ((i&1) == 1) {
        // первая пара состоит из одного трита
          cx.n++; cx.trit[i] = 0;
               }
     }
   }
   j = cx.n >> 1; // j – индекс старшего трита корня
   if (cmp(cx, j + j - 1) == 1) // cx больше .25
     {bi.n = 1; bi.trit[0] = -1; y-> n = 1; y-> trit[0] = 1;} // bi=-1, ai=1
   else { bi.n = 0; y-> n = 0; } // bi=0 и ai=0
   for (j--; j >= 0; j--)
  { // определение j-го трита корня
    shiftb(&bi, x, i); shiftc(y);
    sbi = sgn(bi); cmpba = acmp(bi, y);
    if ((cmpba == 1) || (cmpba == 0) && (cmp1(cx, j + j - 1) == sbi))
    { // j –й трит корня не нуль
      sum3(bi, y, &bi);
      if (y-> n == 0) { y->n = 1; }
      y-> trit[0] = sbi;
      } // if
    } // for
  } // else
} // конец функции
```

Литература

1. Рамиль Альварес Х. Алгоритмы деления и извлечения квадратного корня в троичной симметричной системе // Вестн. Моск. Ун-та. Серия 15. Вычислительная математика и кибернетика. № 2, 2008. С. 42–45.
2. Рамиль Альварес Х. Троичный алгоритм извлечения квадратного корня // Программные системы и инструменты. №. 11. – М.: МАКС Пресс, Москва, 2010. С. 98–106.
3. Рамиль Альварес Х. Извлечение квадратного корня в троичной симметричной системе // Рамиль Альварес Х. Алгоритмы троичной арифметики. М: Фонд «Новое тысячелетие», 2012. С. 14-19.
4. Березин Б.И., Березин С.Б. Начальный курс С и С++. М: ДИАЛОГ МИФИ, 2005.
5. Balanced ternary [Электронный ресурс] Электрон. дан. – URL: https://en.wikipedia.org/wiki/Balanced_ternary (дата обращения 12.10.2015)
6. Брусенцов Н.П., Маслов С.П. и др. Малая цифровая вычислительная машина «Сетунь». М.: Изд-во МГУ, 1965.

Опубликовано: Современные информационные технологии и ИТ-образование, издательство Фонд содействия развитию интернет-медиа, ИТ-образования, человеческого потенциала Лига интернет-медиа (Москва), том 11, № 2, с. 233-238.