УЧЕНЫЕ ЗАПИСКИ КАЗАНСКОГО УНИВЕРСИТЕТА.
СЕРИЯ ФИЗИКО-МАТЕМАТИЧЕСКИЕ НАУКИ

2019, Т. 161, кн. 2
С. 250–262

ISSN 2541-7746 (Print)
ISSN 2500-2198 (Online)

УДК 511.41:511.42

doi: 10.26907/2541-7746.2019.2.250-262

ПРИБЛИЖЕНИЕ ВЕЩЕСТВЕННОГО ЧИСЛА РАЦИОНАЛЬНЫМ В АППРОКСИМИРУЮЩЕМ k-АРНОМ АЛГОРИТМЕ

Р.Р. Еникеев

Казанский (Приволжский) федеральный университет, г. Казань, 420008, Россия

Аннотация

Исследована задача нахождения наилучшего приближения вещественного числа несократимой дробью, знаменатель которой не превосходит заданного значения n. Цель работы состояла в нахождении самого быстрого метода аппроксимации, что позволит ускорить сходимость аппроксимирующего k-арного алгоритма вычисления наибольшего общего делителя. Описано приближение с помощью рядов Фарея, рассмотрены методы ускорения этого алгоритма с использованием условия быстрого выхода из цикла, предвычисления начальных шагов алгоритма и поиска по заранее построенному ряду. Рассмотрена также аппроксимация цепными дробями и разработан метод, который использует их и предвычисленные начальные шаги, полученные с помощью рядов Фарея. Получены оценки сложности этих методов и проведено сравнение алгоритмов по количеству итераций и времени выполнения. В результате сравнения показано, что приближение с помощью рядов Фарея и предвычислением показывает лучшее время, а среди алгоритмов, не использующих дополнительную память, – метод, основанный на цепных дробях. Полученные результаты позволяют выбрать подходящий алгоритм в зависимости от требований, предъявляемых к программному продукту, реализующему приближение вещественного числа.

Ключевые слова: аппроксимирующий k-арный алгоритм, ряды Фарея, цепные дроби

Введение

Известно, что k-арный алгоритм для нахождения НОД чисел u и v ($u \geq v > 0$) вычисляет целочисленные коэффициенты a, b, которые удовлетворяют условиям $0 < a$, $|b| < \sqrt{k}$ и $ua + vb \equiv 0 \pmod{k}$ [1–3]. При этом вычисляется значение $t = |ua + vb|/k$ и с помощью k-арного алгоритма продолжается поиск НОД чисел v и t. Нахождение значений коэффициентов a, b, вычисление числа t и переход к новой паре v, t называется шагом редукции. В качестве k обычно выбирается степень двойки, то есть $k = 2^{2m}$, чтобы заменить операцию деления на битовый сдвиг. При данном значении k длина числа t меньше длины u почти на m бит, так как $|ua+vb|/k < (|u||a|+|v||b|)/k < (u\sqrt{k}+u\sqrt{k})/k < 2u\sqrt{k}/k = 2u/2^m$. Таким образом, k-арный алгоритм за два шага редукции позволяет сократить значения обоих чисел u, v на m бит.

Аппроксимирующий k-арный алгоритм на каждом шаге редукции для нахождения коэффициентов a, b использует аппроксимацию отношения u/v такую, что величина $|ua + vb|$ минимальна, что позволяет ускорить обычный k-арный алгоритм [4]. Для этого выбираем малое a и большее отрицательное значение $b \approx$

$\approx -ua/v$, что позволяет в двустороннем порядке уменьшить ua и vb. Таким образом, эффективное приближение вещественного числа приводит к ускорению аппроксимирующего k-арного алгоритма.

Дробь c/d является *наилучшим приближением* для вещественного числа r, если для любой дроби $a/b \neq c/d$ такой, что $0 < b \leq d$, выполняется неравенство $|r - a/b| > |r - c/d|$.

Задача состоит в том, что для заданного вещественного числа r в интервале $[0, 1]$ и целого числа n необходимо найти наилучшую аппроксимацию r несократимой дробью, знаменатель которой не превосходит n. Мы будем рассматривать приближение с помощью рядов Фарея и цепных дробей, метод же решения этой задачи с помощью геометрического подхода предложен в статье [5].

1. Аппроксимация с помощью рядов Фарея

Ряд (или *дроби*, или *последовательность*) *Фарея* порядка n, обозначаемый через F_n, — возрастающая последовательность несократимых дробей между 0 и 1, имеющих знаменатель, меньший или равный n [6]. Например, ряд Фарея при $n = 5$ имеет вид $F_5 = \left\{\frac{0}{1}, \frac{1}{5}, \frac{1}{4}, \frac{1}{3}, \frac{2}{5}, \frac{1}{2}, \frac{3}{5}, \frac{2}{3}, \frac{3}{4}, \frac{4}{5}, \frac{1}{1}\right\}$.

Количество дробей порядка n, $|F_n|$, имеет асимптотический предел $3n^2/\pi^2$ [7].

Две дроби a/b и c/d называются *близкими*, если $bc - ad = 1$. Если дроби близки, тогда из соотношения Безу следует, что они обе несократимы [8].

Рассмотрим важные свойства соседних элементов рядов Фарея. Если a/b и c/d – соседние дроби в ряде Фарея и $a/b < c/d$, то они являются близкими. Обратное тоже верно, то есть если $bc - ad = 1$, то дроби являются соседями в ряде Фарея порядка $\max(b, d)$. Разница между дробями a/b и c/d будет равна

$$c/d - a/b = 1/(bd). \tag{1}$$

Пусть a/b, p/q и c/d – три соседних дроби и выполняется $a/b < p/q < c/d$, то p/q является *медиантой*, то есть $\frac{p}{q} = \frac{a+c}{b+d}$. Таким образом, если a/b и c/d – соседние дроби, то медианта является первой дробью, которая появится в ряде Фарея порядка $b+d$, и она будет иметь наименьший знаменатель внутри интервала $[a/b, c/d]$. При этом медианта соседних дробей будет несократима.

1.1. Аппроксимация с помощью вычисления медианты. Чтобы найти аппроксимацию вещественного числа с использованием рядов Фарея [4], мы выбираем интервал, границами которого являются две близкие дроби a/b и c/d. Пусть аппроксимируемое число r находится внутри данного интервала. Вычисляем медианту дробей a/b и c/d, относительно которой можно разбить интервал на две части $\left[\frac{a}{b}, \frac{a+c}{b+d}\right]$, $\left[\frac{a+c}{b+d}, \frac{c}{d}\right]$. Затем из двух этих интервалов выбирается тот, внутри которого находится r, после чего описанная выше процедура продолжается для выбранного интервала. Эта процедура завершается, когда знаменатель медианты дробей a/b и c/d станет больше n, а в качестве результата аппроксимации r принимается ближайшая дробь к r из дробей a/b и c/d (если расстояния равны, тогда выбирается дробь с наименьшим знаменателем). В рассматриваемом методе используется свойство медианты, согласно которому она является дробью с наименьшим знаменателем в интервале $\left[\frac{a}{b}, \frac{c}{d}\right]$, что гарантирует нахождение наилучшего приближения. Так как в нашей задаче вещественное число лежит внутри $[0, 1]$, то в качестве начальных a/b и c/d выберем $0/1$ и $1/1$ (элементы F_1) соответственно. Данная процедура описана в алгоритме 1, который будем обозначать Ф1.

Алгоритм 1 Аппроксимация с помощью последовательного вычисления медианты

```
a := 0, b := 1          // задаем начальную левую границу
c := 1, d := 1          // задаем начальную правую границу
while b + d <= n do
    (p, q) := (a + c, b + d)
    if p == q * r then
        return (p, q)
    if p < q * r then
        (a, b) := (p, q)        // выбираем правую часть интервала
    else
        (c, d) := (p, q)        // выбираем левую часть интервала
return Выбрать ближайшую дробь к r из a/b и c/d
```

Алгоритм Ф1 можно улучшить с помощью следующих рассуждений (А и В).

А. В алгоритме Ф1 номер итерации, на которой вычисляется дробь $\dfrac{p}{q} = \dfrac{a+c}{b+d}$, зависит от самой дроби: чем больше знаменатель, тем больше итераций потребуется для ее нахождения. Проанализировав количество итераций, необходимых для вычисления всех дробей ряда Фарея, мы установили, что самое большое количество итераций алгоритма для ряда Фарея порядка n приходится на $n/2$ самых крайних дробей слева и справа. Они имеют вид $1/i$ слева, если находятся в интервале $e_l = \left[\dfrac{1}{n}, \ldots, \dfrac{1}{n/2}\right]$, и $(1 - 1/i)$ — справа, если содержатся в интервале $e_r = \left[\dfrac{n/2-1}{n/2}, \ldots, \dfrac{n-1}{n}\right]$. Специфический вид этих дробей позволяет упростить нахождение нужной дроби. Ее вычисление можно выполнить по простой формуле без цикла, например, если r попадает в e_l, тогда дробью, представляющей аппроксимацию r, будет та дробь $\dfrac{1}{\lfloor 1/r \rfloor}$ или $\dfrac{1}{\lceil 1/r \rceil}$, которая ближе к r. Это дает нам выигрыш по скорости. Таким образом, мы модифицируем алгоритм Ф1, сначала проверяя, какому из диапазонов e_l или e_r принадлежит аппроксимируемое вещественное число r. Установив это, мы без цикла можем легко вычислить подходящую дробь за $O(1)$ операций, так как не требуется выполнение цикла. Если r не принадлежит e_l и e_r, тогда запускаем метод Ф1. Обозначим данную модификацию алгоритма через Ф2.

В. Рассмотрим теперь следующую проблему. Пусть наилучшим приближением к r является $1/2$. Алгоритм Ф1 будет выполняться до тех пор, пока не будет найдена соседняя дробь $1/2$. Эти вычисления соседней дроби являются напрасными, так как аппроксимация r в качестве $1/2$ уже была найдена. При этом чем больше n, тем будет больше ненужных вычислений, необходимых для нахождения соседней дроби. Для решения этой проблемы определим условие выхода из цикла, если мы уже получили наилучшую аппроксимацию на текущем шаге итерации.

Пусть на определенном шаге Ф1 мы вычислили медианту a/b. Согласно формуле (1) расстояние между этой дробью и ее соседней c/d, значение которой мы еще не вычислили, равно $1/(bd)$. Отсюда и из того, что соседние дроби a/b не могут иметь знаменатель, больший, чем n, следует, что минимальное расстояние от дроби a/b до ее соседней дроби составляет $1/(bn)$. Таким образом, если r находится к a/b ближе, чем $1/(2bn)$ (половина расстояния до соседней дроби), то a/b является к r ближайшей дробью, чей знаменатель не превосходит n. Следовательно, при выполнении условия $|r - a/b| < 1/(2bn)$ необходимо прекратить алгоритм аппроксимации, результатом которого будет дробь a/b. В данном алгоритме также

используется быстрая аппроксимация крайних дробей (интервалы e_l и e_r), как в Ф2. Обозначим эту модификацию алгоритма Ф2 через Ф3.

Вычислим временну́ю сложность описанных выше алгоритмов. Знаменатель в них вычисляется по формуле $q = b + d$. Оценим сверху это значение: $q = b + d < 2\max(b, d) < \cdots < 2^k$, где k — количество итераций. Таким образом, оценка снизу для количества итераций в Ф1 составляет $O(\log n)$. Худший случай будет при приближении крайних дробей, и в таком случае сложность будет составлять $O(n)$. Алгоритм Ф2 работает за $O(1)$ итераций при аппроксимации крайних дробей. Как было указано выше, было проведено исследование, которое находит количество итераций, требуемое для вычисления каждой дроби ряда Фарея. В результате такого исследования получено, что в случае приближения вещественного числа, не лежащего в интервалах e_l и e_r, худшей ситуацией является та, при которой аппроксимируемое число r находится рядом с $1/2$. В этом случае потребуется $O(n/2) = O(n)$ итераций. Количество итераций при использовании Ф3 будет полностью зависеть от числа, которое мы аппроксимируем, но в худшем случае, когда быстрое условие выхода из цикла не выполняется, сложность данного метода будет такая же, как и у Ф2. В алгоритмах Ф1–Ф3 количество операций на каждой итерации фиксированно, поэтому сложность этих алгоритмов составляет $O(n)$.

Во всех вышеперечисленных алгоритмах используется несколько чисел для представления дробей a/b, c/d и их медианты, p/q. Числитель данных дробей меньше знаменателя, который не превосходит n. Следовательно, для хранения каждого числа необходимо $O(\log n)$ бит памяти. Количество чисел, которое должно храниться в памяти, постоянное, поэтому в Ф1–Ф3 требуется $O(\log n)$ бит памяти.

1.2. Аппроксимация с предвычислением начальных границ. В этом и последующих методах мы пытаемся ускорить аппроксимацию с помощью рядов Фарея, используя дополнительную память и предвычисления.

Предлагаемый метод является улучшением предыдущего. В этом алгоритме сначала выполняем шаг предвычислений, для этого производим разбивку диапазона $[0, 1]$ на m равных отрезков. Пусть $s_i = [(i-1)/m, i/m)$ – i-й отрезок. Далее для каждого s_i запускаем модифицированный алгоритм Ф1, чтобы он вернул нам дроби a_i/b_i и c_i/d_i такие, что $a_i/b_i \le (i-1)/m$, $i/m \le c_i/d_i$, а медианта $(a_i + c_i)/(b_i + d_i)$ находится внутри s_i. Дроби a_i/b_i и c_i/d_i будут левой и правой границами интервала, начиная с которого будет проводиться аппроксимация числа r, если оно принадлежит s_i. Дроби a_i/b_i и c_i/d_i сохраняются в массив. Таким образом, мы с помощью предвычислений выполняем начальную часть алгоритма Ф1, заранее приближая границы начального интервала к вещественному числу настолько близко, насколько это возможно для заданного m.

Сам алгоритм приближения заключается в следующем. Находится номер i отрезка, в который попадает аппроксимируемое вещественное число r. Далее извлекаем из массива заранее вычисленные границы a_i/b_i и c_i/d_i начального интервала аппроксимации r. Затем запускается алгоритм Ф3, только вместо первоначальных границ $0/1$ и $1/1$ (первый и второй шаг Ф1) используем дроби a_i/b_i и c_i/d_i соответственно. Обозначим данный метод через Ф4.

В алгоритме Ф4 чем больше m, тем меньше размер отрезков и тем ближе начальные границы интервала a_i/b_i и c_i/d_i к r, следовательно, чем больше m, тем меньше количество итераций в данном алгоритме будет проводиться и тем быстрее будет он выполняться.

В рассматриваемом алгоритме используется m отрезков, каждый из которых содержит четыре дроби, у которых размер числителя и знаменателя не превосходит n. Следовательно, Ф4 использует $O(m \log n)$ бит дополнительной памяти.

Значение m можно задать, следовательно, можно контролировать как размер дополнительной памяти, так и количество итераций в алгоритме аппроксимации.

1.3. Аппроксимация с помощью поиска по ряду Фарея. В следующих двух методах строим весь ряд Фарея заранее. Поскольку в нашей задаче r необходимо аппроксимировать дробью, знаменатель которой не превосходит n, строим ряд F_n. Для построения ряда Фарея был использован метод генерации последующего члена с помощью двух предыдущих, начиная с $0/1$ и $1/n$. Данный метод описан в [9]. В результате выполнения указанной процедуры будет получен массив, содержащий все дроби ряда Фарея порядка n. Как было показано выше, количество дробей в массиве будет порядка $n^2/3$, а размер каждого элемента (числитель и знаменатель) не превосходит $2\log n$. Поэтому в двух алгоритмах, описанных ниже, для хранения ряда Фарея необходимо $O(n^2 \log n)$ бит памяти.

1.3.1. Аппроксимация с помощью бинарного поиска. Дроби в ряде Фарея по определению являются возрастающими, поэтому можно использовать бинарный поиск для аппроксимации вещественного числа. В этом алгоритме поиск начинается с массива, который содержит весь ряд Фарея. Далее на каждой итерации делим массив пополам и выбираем ту его половину, внутрь которой попадает аппроксимируемое вещественное число, уменьшая размер массива в два раза. Эта процедура продолжается для выбранной половины до тех пор, пока не будет найдена необходимая аппроксимация r. Временна́я сложность бинарного поиска по всему ряду будет равна $O(\log n^2) = O(2\log n) = O(\log n)$. Обозначим рассматриваемый алгоритм через Ф5.

1.3.2. Аппроксимация с помощью интерполяционного поиска. Интерполяционный поиск имитирует поиск имени в телефонной книге [10]. Например, если фамилия начинается на «В», мы откроем телефонную книгу ближе к началу, а не в середине или в конце. Как описано выше, при бинарном поиске всегда происходит сравнение со значением, находящимся в середине интервала, при этом игнорируется искомое значение. При интерполяционном поиске, с другой стороны, искомое значение учитывается.

Использование интерполяционного поиска для аппроксимации r по аналогии с поиском в телефонной книге будет строиться на следующем соображении: если, например, $r = 0.235$, то число r будет находиться ближе к началу построенного ряда Фарея.

В рассматриваемом алгоритме поиск по массиву, содержащему построенный заранее ряд Фарея, начинается с интервала, чей левый и правый индексы равны первому и последнему индексам массива соответственно. Пусть на текущей итерации левый и правый индексы интервала равны i_l и i_r соответственно. Тогда индекс i, на основе которого проводится разбиение на две части текущего интервала, вычисляется по формуле [10]

$$i = i_l + \left\lfloor \frac{(r - F_n[i_l])(i_r - i_l)}{F_n[i_r] - F_n[i_l]} \right\rfloor.$$

На основе этой формулы в предвычисленном массиве прогнозируется индекс дроби i, который является аппроксимацией r, по расстоянию между r и текущим значением дроби в ряде Фарея. После сравнения r с $F_n[i] = a_i/b_i$ мы либо прекращаем алгоритм, если r и a_i/b_i достаточно близки (при выполнении условия $|r - F_n[i]| < 1/(2b_i n)$ из Ф3), либо выбираем интервал, внутри которого находится r, продолжая процедуру интерполяционного поиска.

Преимущество данного метода по сравнению с бинарным состоит в уменьшении числа итераций, что влечет за собой сокращение количества запросов на чтение медленной памяти (ОЗУ), если запросы происходят часто.

В этом алгоритме количество операций зависит от равномерности распределения дробей, но в среднем сложность по времени составляет $O(\log\log n^2) = O(\log(2\log n)) = O(\log\log n)$ [10]. Обозначим его через Ф6.

2. Аппроксимация с помощью цепных дробей

В книге [11] подробно описаны цепные дроби и доказательства всех утверждений из этого раздела. Мы же лишь кратко рассмотрим теорию непрерывных дробей, необходимую для аппроксимации.

Простейшая цепная дробь (*ЦД*) (или *непрерывная дробь*) – это конечное или бесконечное математическое выражение вида

$$[a_0; a_1, a_2, a_3, \dots] = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \dots}}}$$

Конечная ЦД записывается как $[a_0; a_1, a_2, a_3, \dots, a_m]$, где a_i называется *элементом* цепной дроби. Мы рассматриваем непрерывные дроби, у которых a_0 является целым числом, а все остальные элементы — натуральные числа.

Любое вещественное число представимо в виде цепной дроби, при этом рациональное число – с помощью конечной ЦД, а иррациональное число можно представить посредством бесконечной непрерывной дроби, причем единственным способом [11].

2.1. Разложение вещественного числа в цепную дробь. Для нахождения представления вещественного числа r цепной дробью $[a_0; a_1, a_2, a_3, \dots]$ воспользуемся следующим алгоритмом. Сначала вычислим значения $a_0 = \lfloor r \rfloor$ и $r_0 = r - a_0$. Пусть на предыдущем шаге были вычислены значения a_{i-1} и r_{i-1}, тогда для нахождения значений на текущем шаге воспользуемся формулами $a_i = \left\lfloor \dfrac{1}{r_{i-1}} \right\rfloor$ и $r_i = \dfrac{1}{r_{i-1}} - a_i$. Данный алгоритм завершается, если $r_i = 0$, то есть цепная дробь конечна, так как количество элементов a_i конечно. Если цепная дробь бесконечная, то процедура будет выполняться бесконечно, вследствие чего может потребоваться выход из алгоритма после определенного количества итераций.

В данном методе $\lfloor r \rfloor$ означает целую часть r. Можно заметить, что a_0 является целой частью вещественного числа, а так как мы аппроксимируем вещественные числа внутри $[0, 1]$, то $a_0 = 0$.

2.2. Подходящие дроби. *Подходящая дробь порядка* k (convergent) p_k/q_k вычисляется по следующей рекуррентной формуле: $p_{-1} = 1$, $q_{-1} = 0$, $p_0 = a_0$, $q_0 = 1$, $p_k = a_k p_{k-1} + p_{k-2}$, $q_k = a_k q_{k-1} + q_{k-2}$, $k = 1, 2, \dots, n$.

Поскольку a_i, $i > 0$, — натуральное число, последовательности числителей $\{p_i\}$ и знаменателей $\{q_i\}$ являются возрастающими, при этом $p_i < q_i$ при $i > 0$.

Для рационального числа $p/q = [a_0; a_1, a_2, a_3, \dots, a_m]$, количество подходящих дробей конечно и равняется $(m + 1)$, а вычисление m-й подходящей дроби даст само рациональное число, то есть $p/q = p_m/q_m$.

Иррациональному числу $\alpha = [a_0; a_1, a_2, a_3, \dots]$ соответствует бесконечное число подходящих дробей, при этом последовательность $\left\{\dfrac{p_i}{q_i}\right\}_{i\geq 0}$ сходится к α. Таким образом, каждая следующая подходящая дробь дает лучшую аппроксимацию, чем предыдущая.

Подходящие дроби обладают тем свойством, что любая подходящая дробь p_i/q_i находится ближе к r (наилучшее приближение) среди всех дробей, знаменатель которых меньше или равен q_i.

Каждая подходящая дробь нечетного порядка больше каждой дроби четного порядка, при этом $p_i/q_i \geq r$, если значение i — нечетно, и $p_i/q_i \leq r$, если значение i — четно [11]. Подходящие дроби четных порядков образуют возрастающую последовательность, а нечетные — убывающую. Таким образом, эти две последовательности идут справа и слева и стремятся к r.

Для соседних подходящих дробей выполняется следующее равенство: $q_k p_{k-1} - p_k q_{k-1} = (-1)^k$ при $k \geq 0$. Из него следует, что подходящие дроби являются несократимыми, что является одним из требований нашей задачи.

2.3. Промежуточные дроби. Рассмотрим при $k \geq 2$ последовательность

$$\frac{p_{k-2}}{q_{k-2}}, \frac{p_{k-2}+p_{k-1}}{q_{k-2}+q_{k-1}}, \frac{p_{k-2}+2p_{k-1}}{q_{k-2}+2q_{k-1}}, \dots, \frac{p_{k-2}+a_k p_{k-1}}{q_{k-2}+a_k q_{k-1}} = \frac{p_k}{q_k}.$$

Крайними элементами этой последовательности являются подходящие дроби, а элементы между ними (если $a_k > 1$) называются *промежуточными дробями* (semiconvergent). Соседние члены этой последовательности являются близкими дробями, то есть последовательные дроби a/b и c/d обладают свойством $ad - bc = \pm 1$, из которого следует, что промежуточные дроби также несократимы.

Поскольку $p_i > 0$ и $q_i > 0$, последовательности числителей $\{p_{k-2} + ip_{k-1}\}_{k\geq 2}$ и знаменателей $\{q_{k-2} + iq_{k-1}\}_{k\geq 2}$ промежуточных дробей, как и у подходящих, являются возрастающими.

Нетрудно видеть, что i-я промежуточная дробь $\dfrac{p_{k-2}+ip_{k-1}}{q_{k-2}+iq_{k-1}}$ является медиантой $(i-1)$-й промежуточной дроби и подходящей дроби $\dfrac{p_{k-1}}{q_{k-1}}$. Медианта двух дробей всегда находится между ними, а поскольку порядки дробей $\dfrac{p_{k-2}}{q_{k-2}}$ и $\dfrac{p_{k-2}+a_k p_{k-1}}{q_{k-2}+a_k q_{k-1}} = \dfrac{p_k}{q_k}$ имеют одинаковую четность, то они находятся по одну сторону от r, следовательно, и все промежуточные дроби будут находиться с той же стороны. При этом дробь $\dfrac{p_{k-1}}{q_{k-1}}$ располагается по другую сторону от r, так как порядок данной дроби имеет другую четность.

Как было отмечено выше, если k четно, то выполняется неравенство $\dfrac{p_{k-2}}{q_{k-2}} < \dfrac{p_{k-1}}{q_{k-1}}$. Из свойства медианты вытекают неравенства $\dfrac{p_{k-2}}{q_{k-2}} < \dfrac{p_{k-2}+p_{k-1}}{q_{k-2}+q_{k-1}} < \dfrac{p_{k-1}}{q_{k-1}}$. Продолжая вычислять медианты, получим неравенства

$$\frac{p_{k-2}}{q_{k-2}} < \frac{p_{k-2}+p_{k-1}}{q_{k-2}+q_{k-1}} < \frac{p_{k-2}+2p_{k-1}}{q_{k-2}+2q_{k-1}} < \dots < \frac{p_{k-2}+a_k p_{k-1}}{q_{k-2}+a_k q_{k-1}} = \frac{p_k}{q_k} < \frac{p_{k-1}}{q_{k-1}}.$$

Таким образом, промежуточные дроби при четном k образуют возрастающую последовательность от дроби p_{k-2}/q_{k-2} к дроби p_k/q_k. А поскольку дробь p_k/q_k ближе к r, чем дробь p_{k-2}/q_{k-2}, то последовательность промежуточных дробей приближается к r. Используя аналогичные рассуждения, можно показать, что при нечетном k последовательность промежуточных дробей убывает, приближаясь к r.

2.4. Аппроксимация с помощью вычисления подходящих и промежуточных дробей. Если дробь является наилучшим приближением, то она является либо подходящей, либо промежуточной. В п. 2.2 было установлено, что, если дробь подходящая, то она будет наилучшим приближением к r. Однако для промежуточной дроби обратное неверно, то есть если дробь является промежуточной, то из этого не следует, что она есть наилучшее приближение.

На основе приведенных соображений можно построить следующий алгоритм для аппроксимации r. Находим подходящую дробь с максимальным знаменателем, не превосходящим n. Эта дробь будет наилучшим приближением среди подходящих дробей. Промежуточная дробь может быть ближе к r, чем найденная подходящая дробь, поэтому далее, используя найденную подходящую дробь, вычисляем промежуточную дробь с максимальным знаменателем, не превосходящим n, если такая промежуточная дробь существует. В конце из полученных дробей выбираем ту дробь, которая находится ближе к числу r. Если расстояния равны, то выбираем подходящую дробь, так как ее знаменатель меньше. Если промежуточной дроби со знаменателем, не превосходящим n, не существует, то возвращаем подходящую дробь.

Пусть мы нашли подходящие дроби p_{k-1}/q_{k-1}, p_k/q_k и p_{k+1}/q_{k+1}, для которых выполнено неравенство $q_k \le n < q_{k+1}$. Тогда вычислить промежуточную дробь с максимальным знаменателем, не превосходящим n, можно найдя максимальное j, удовлетворяющее неравенству $q_{k-1} + jq_k \le n$. Из этого неравенства находим $j = \lfloor (n - q_{k-1})/q_k \rfloor$. Далее, подставив найденное j в $\dfrac{p_{k-2} + jp_{k-1}}{q_{k-2} + jq_{k-1}}$, получим необходимое значение промежуточной дроби. Алгоритм 2 позволяет провести аппроксимацию с помощью цепных дробей; операция "/" возвращает вещественное число в результате деления. Как было описано в п. 2.2., строка "$p_{i+1}/q_{i+1} == r$" в алгоритме 2 истинна тогда и только тогда, когда r является рациональным числом.

В [11] дана следующая оценка для подходящих дробей для точности приближения, полученного с помощью цепных дробей: $|r - p_k/q_k| < 1/q_k^2$. Оценим время выполнения алгоритма 2. Обозначим подходящую дробь с максимальным знаменателем, не превосходящим n, через p_m/q_m. Эта дробь вычисляется в цикле алгоритма 2. Знаменатель подходящей дроби удовлетворяет неравенству $q_k \ge 2^{(k-1)/2}$ при $k \ge 2$. Используя это неравенство и тот факт, что знаменатель результирующей дроби не должен превосходить n, то есть $q_m \le n$, получим верхнюю оценку для количества итераций, необходимых для вычисления подходящей дроби, $m \le 2\log n + 1$. Нахождение промежуточной дроби в алгоритме 2 после цикла требует $O(1)$ операций. Таким образом, верхняя оценка сложности всего алгоритма составляет $O(\log n)$. Значение q_n целиком и полностью зависит от величины коэффициентов a_i, так как $q_n > a_n q_{n-1} > \cdots > a_n \cdot \ldots \cdot a_1$. Таким образом, количество итераций для конкретного аппроксимируемого числа r зависит от коэффициентов его разложения в цепную дробь, и чем больше значения коэффициентов, тем быстрее завершится алгоритм. Наихудший случай возникает, если при разложении некоторого r получим $a_i = 1$ (напомним, что $a_i \ne 0$ для $i > 0$), что соответствует золотому сечению. В этом алгоритме используется $O(\log n)$ бит памяти, как и в Ф1. Обозначим данный метод как Ц1.

2.5. Аппроксимация с помощью цепных дробей и предвычислением. Можно заметить связь между вычислением дробей ряда Фарея с помощью медианты и нахождением подходящих и промежуточных дробей, особенно если установить начальную правую границу в рядах Фарея на $1/0$ вместо $1/1$. Тогда вычисленные медианты будут полностью совпадать либо с подходящими, либо с промежуточными дробями. Таким образом, можно сказать, что при вычислениях подходящих дробей пропускаются некоторые дроби, которые были бы вычислены

Алгоритм 2 Аппроксимация с помощью цепных дробей

```
p_i := 1, q_i := 0                    // p_{-1} и q_{-1}
p_{i+1} := 0, q_{i+1} := 1            // p_0 и q_0
x := r
while true do // Вычисляем подходящую дробь
        x := 1/x
        a_i := ⌊x⌋
        x := x − a_i
        if a_i * q_{i+1} + q_i > n then
                break // Если знаменатель след-ей подходящей дроби > n
        (p_i, p_{i+1}) := (p_{i+1}, a_i * p_{i+1} + p_i)
        (q_i, q_{i+1}) := (q_{i+1}, a_i * q_{i+1} + q_i)
        if p_{i+1}/q_{i+1} == r then
                return (p_{i+1}, q_{i+1})
// в p_{i+1}, q_{i+1} хранится подходящая дробь с максимальным знаменателем
j = ⌊(n − q_i)/q_{i+1}⌋
if j == 0 then
        return (p_{i+1}, q_{i+1}) // Не существует промежуточной дроби
(p', q') = (p_i + j * p_{i+1}, q_i + j * q_{i+1}) // Вычисляем промежуточную дробь
if |r − p_{i+1}/q_{i+1}| ≤ |r − p'/q'| then
        return (p_{i+1}, q_{i+1})
else
        return (p', q')
```

с помощью медианты и рядов Фарея. Следовательно, можно использовать метод, аналогичный алгоритму Ф4, то есть заранее предвычислить границы начального интервала аппроксимации, насколько это возможно.

Алгоритм предвычисления остается таким же, как и в Ф4. Но с самой аппроксимацией возникают трудности. В алгоритме 2 во время аппроксимации происходит разложение вещественного числа в цепную дробь, на каждой итерации используется значение x, вычисленное на предыдущих этапах приближения. Но значение x зависит от величины конкретного вещественного числа, поэтому алгоритм 2 нельзя будет применить для интервала вещественных чисел s_i. Однако Хинчин в [11] описал алгоритм нахождения подходящих дробей через промежуточные без знания элементов a_i цепной дроби.

Пусть мы вычислили p_{k-1}/q_{k-1} и p_k/q_k. Имеем, что $(k-1)$-я и $(k+1)$-я подходящие дроби лежат по одну сторону от r, а k-я – с другой. Тогда, чтобы найти p_{k+1}/q_{k+1}, последовательно находим промежуточные дроби $\dfrac{p_{k-1}+ip_k}{q_{k-1}+iq_k}$ до тех пор, пока они располагаются с той же стороны r, что и $(k-1)$-я подходящая дробь. Отсюда вытекает новый способ вычисления a_{k+1}: необходимо найти максимальное j, удовлетворяющее неравенствам

$$\frac{p_{k-1}+jp_k}{q_{k-1}+jq_k}\begin{cases}\leq r, & \text{если } k-\text{четное},\\ \geq r, & \text{если } k-\text{нечетное}.\end{cases}$$

Заметим, что $p_k - rq_k$ отрицательно при четном k, так как $p_k/q_k < r$, поэтому для нахождения j остается одно условие $j(p_k - rq_k) \leq rq_{k-1} - p_{k-1}$, выполняющееся при любом k. Из последнего неравенства находим $j = \left\lfloor \dfrac{rq_{k-1}-p_{k-1}}{p_k - rq_k} \right\rfloor$. Будем обращаться к этому методу как к Ц2.

Правильность результата, полученного алгоритмом Ц2, следует из того, что наилучшим приближением на каждой итерации будет одна из двух охватывающих дробей.

В Ц2 величина используемой памяти составляет $O(m \log n)$ бит, как и в Ф4.

3. Реализация алгоритмов

В этом разделе рассматриваются основные аспекты реализации на компьютере описанных выше алгоритмов. В начале статьи была поставлена математическая задача аппроксимации вещественного числа, в которой предполагалось, что это аппроксимируемое число неограничено. Однако при реализации вещественное число представляется в памяти компьютера конечным количеством бит. Например, мантисса у числа двойной точности с плавающей запятой (double) составляет 53 бита (52 бита хранятся явно). Следовательно, формат "double" позволяет хранить лишь 2^{53} различных вещественных числа из $[0, 1]$. При $n = 2^{27}$ существует 2^{54} различных дробей, тогда из принципа Дирихле следует, что одному вещественному числу будут соответствовать несколько дробей. В связи с этим, возможно, понадобится модификация наших алгоритмов, чтобы соответствовать данному ограничению.

Алгоритм аппроксимации, использующий ряды Фарея для вычисления медианты, остается неизменным, так как если вещественное число находится внутри определенного интервала, то медианта будет обладать наименьшим знаменателем среди всех дробей из заданного интервала.

Алгоритм приближения с помощью цепных дробей, наоборот, нужно будет изменить из-за того, что, как отмечалось ранее, во время вычисления подходящих дробей пропускаются некоторые медианты. Таким образом, при реализации вычисления подходящих дробей на компьютере они не всегда являются наилучшим приближением. На основе проведенных расчетов было установлено, что ошибки в реализации, связанные с ограниченностью представления вещественного числа, при аппроксимации с помощью цепных дробей начинают проявляться при $n \geq 2^{23}$. Для решения проблемы, вызванной ограниченным количеством бит для представления вещественного числа с помощью формата "double", используем следующий метод. Для простоты описания модифицированного алгоритма будем считать, что промежуточные дроби включают в себя подходящие дроби. Изменение алгоритма приближения с помощью цепных дробей заключается в том, что для найденного решения с помощью первоначального алгоритма необходимо затем просмотреть предыдущие промежуточные дроби. Если они находятся на таком же расстоянии или даже ближе, то принимаем их в качестве результата.

4. Результаты экспериментов

В табл. 1 приводится количество итераций различных аппроксимирующих алгоритмов в зависимости от n. При вычислении крайних дробей в рядах Фарея мы считали, что количество итераций равно 0. Обозначение $\Ф4_n$ и $\Ц2_n$ указывает на то, что при этом использовалось n дополнительной памяти. При использовании алгоритмов Ц1 и Ц2 для вычисления промежуточных дробей после выполнения цикла потребовалась лишь 1 итерация. Символ "–" означает, что мы не проводили расчетов, так как при этом требовалось слишком много дополнительной памяти: порядка сотен мегабайт. Во время расчетов в качестве аппроксимируемого числа r мы выбирали числа вида $2^s - 1$, чтобы ограничить бинарную длину знаменателя s битами.

Самый интересный результат получается для алгоритма $\Ф4_{n^2/3}$: среднее количество итераций меньше 1 (мы полагали, что аппроксимация крайних дробей выполняется за 0 итераций, так как в этом случае не происходит выполнения

Табл. 1

Среднее количество итераций аппроксимирующих алгоритмов

n	Ф1	Ф2	Ф3	$Ф4_n$	$Ф4_{n^2/3}$	Ф5	Ф6	Ц1	$Ц2_n$	$Ц2_{n^2/3}$
2^4–1	7.63	4.57	3.78	1.93	0.65	4.66	1.35	4.35	2.38	1.98
2^8–1	25.7	22.3	19.9	9.27	0.92	14.12	2.69	6.72	4.39	2.64
2^{12}–1	52.7	49.5	45.4	20.9	0.93	22.33	3.30	9.06	5.68	2.68
2^{16}–1	89.7	86.2	80.4	37.4	–	–	–	11.40	6.88	–
2^{24}–1	189	189	179	84	–	–	–	16	9.2	–
2^{32}–1	294	294	294	–	–	–	–	16.65	–	–

Табл. 2

Относительное время работы аппроксимирующих алгоритмов

n	Ф1	Ф2	Ф3	$Ф4_n$	$Ф4_{n^2/3}$	Ф5	Ф6	Ц1	$Ц2_n$	$Ц2_{n^2/3}$
2^4–1	2.57	1.81	1.78	1.27	1	2.12	1.6	1.88	1.98	1.7
2^8–1	7.32	6.45	6.45	2.71	1	6.01	2.81	2.7	2.75	1.92
2^{12}–1	5.11	4.8	5.0	1.87	1	5.68	1.99	1.26	1.23	1.33
2^{16}–1	5.79	5.58	5.95	2.17	–	–	–	1.06	1	–
2^{24}–1	6.92	6.93	7.55	2.98	–	–	–	1.22	1	–
2^{32}–1	9.45	9.44	10.88	-	–	–	–	1	–	–

цикла). При увеличении n в 2 раза число итераций в алгоритмах Ф1–Ф3, $Ф4_n$, Ф5 увеличивается более чем в 2 раза, а для алгоритмов, основанных на цепных дробях, и $Ф4_{n^2/3}$ – менее чем в два раза.

В табл. 2 приводится время выполнения алгоритмов. Значения в ней представлены относительно наиболее быстрого метода. При этом мы, естественно, не учитывали время, необходимое на предвычисления в Ф4–Ф6 и Ц2.

Как видно из табл. 2, Ц1 является самым быстрым среди алгоритмов, не требующих дополнительной памяти, при этом получаются результаты, близкие или даже лучшие, чем $Ф4_n$ и $Ц2_n$ с n дополнительной памятью. Среди алгоритмов с $n^2/3$ дополнительной памятью ($Ф4_{n^2/3}$, Ф3, Ф4, $Ц2_{n^2/3}$) метод $Ф4_{n^2/3}$ является быстрейшим, что неудивительно, если учитывать малое количество итераций в данном алгоритме. Если сравнивать Ф5 (бинарный поиск) и Ф1, можно заметить, что хотя Ф5 требуется меньшее количество операций, но по времени получаются схожие результаты, потому что Ф5 обращается гораздо чаще к оперативной памяти. Интересный результат также получается относительно времени выполнения Ф3: оно больше чем у Ф1, начиная с некоторого n, хотя количество итераций меньше. Это является следствием того, что в Ф3 в условии быстрого выхода используется более «дорогостоящие» операции.

Литература

1. *Sorenson J.* Two fast GCD algorithms // J. Algorithms. – 1994. – V. 16, No 1. – P. 110–144. – doi: 10.1006/jagm.1994.1006.

2. *Jebelean T.* A generalization of the binary GCD algorithm // Proc. 1993 Int. Symp. on Symbolic and Algebraic Computation. – N. Y.: ACM, 1993. – P. 111–116. – doi: 10.1145/164081.164102.

3. *Weber K.* The accelerated integer GCD algorithm // ACM Transact. Math. Software. – 1995. – V. 21, No 1. – P. 111–122. – doi: 10.1145/200979.201042.

4. *Ishmukhametov S.* An approximating k-ary GCD algorithm // Lobachevskii J. Math. – 2016. – V. 37, No 6 – P. 723–729. – doi: 10.1134/S1995080216060147.

5. *Charrier E., Buzer L.* Approximating a real number by a rational number with a limited denominator: A geometric approach // Discrete Appl. Math. – 2009. – V. 157, No 16 – P. 3473–3484. – doi: 10.1016/j.dam.2009.03.005.

6. *Hardy G.H., Wright E.M.* An Introduction to the Theory of Numbers. – Oxford: Oxford Univ. Press, 1975. – 433 p.

7. *Vardi I.* Computational Recreations in Mathematica. – Boston: Addison-Wesley, 1991. – 304 p.

8. *Фаддеев Д.К.* Лекции по алгебре. – М.: Наука, 1984. – 416 с.

9. *Routledge N.* Computing Farey Series // Math. Gazette. – 2008. – V. 92, No 523. – P. 55–62. – doi: 10.1017/S002555720018252X.

10. *Левитин А.В.* Алгоритмы. Введение в разработку и анализ. – М.: Вильямс, 2006. – 576 с.

11. *Хинчин А.Я.* Цепные дроби. – М.: ГИФМЛ, 1961. – 112 с.

Поступила в редакцию
07.06.18

Еникеев Разиль Радикович, ассистент кафедры системного анализа и информационных технологий

Казанский (Приволжский) федеральный университет
ул. Кремлевская, д. 18, г. Казань, 420008, Россия
E-mail: *renikeev@kpfu.ru*

ISSN 2541-7746 (Print)
ISSN 2500-2198 (Online)

**UCHENYE ZAPISKI KAZANSKOGO UNIVERSITETA.
SERIYA FIZIKO-MATEMATICHESKIE NAUKI
(Proceedings of Kazan University. Physics and Mathematics Series)**

2019, vol. 161, no. 2, pp. 250–262

doi: 10.26907/2541-7746.2019.2.250-262

Real Number Approximation by a Rational Number in the Approximating k-ary Algorithm

R.R. Enikeev

Kazan Federal University, Kazan, 420008 Russia
E-mail: *renikeev@kpfu.ru*

Received June 7, 2018

Abstract

The best approximation by the irreducible fraction with the denominator not exceeding some specified value n was investigated. The aim of this study was to find the fastest approximation algorithm that enables to accelerate the convergence of the k-ary algorithm for computing the greatest common divisor. The approximation based on Farey sequences was described; this method was considered using a quick exit condition, precomputation of the initial steps, and search in the Farey sequence built in advance. We also discussed the approximation with continued fractions and proposed an algorithm using a precomputation

and continued fractions. The time and memory complexity of these algorithms were shown; these methods were compared with respect to running time and iterations amount. It was concluded that the approximation with the help of Farey sequences and precomputation is the fastest method, but the approximation with continued fractions has no additional memory demands.

Keywords: approximating k-ary algorithm, Farey sequence, continued fractions

References

1. Sorenson J. Two fast GCD algorithms. *J. Algorithms*, 1994, vol. 16, no. 1, pp. 110–144. doi: 10.1006/jagm.1994.1006.
2. Jebelean T. A generalization of the binary GCD algorithm. *Proc. 1993 Int. Symp. on Symbolic and Algebraic Computation*. New York, ACM, 1993, pp. 111–116. doi: 10.1145/164081.164102.
3. Weber K. The accelerated integer GCD algorithm. *ACM Trans. Math. Software*, 1995., vol. 21, no. 1, pp. 111–122. doi: 10.1145/200979.201042.
4. Ishmukhametov S. An approximating k-ary GCD algorithm. *Lobachevskii J. Math.*, 2016, vol. 37, no. 6, pp. 723–729. doi: 10.1134/S1995080216060147.
5. Charrier E., Buzer L. Approximating a real number by a rational number with a limited denominator: A geometric approach. *Discrete Appl. Math.*, 2009, vol. 157, no. 16, pp. 3473–3484. doi: 10.1016/j.dam.2009.03.005.
6. Hardy G.H., Wright E.M. *An Introduction to the Theory of Numbers*. Oxford, Oxford Univ. Press, 1975. 433 p.
7. Vardi I. *Computational Recreations in Mathematica*. Boston, Addison-Wesley, 1991. 304 p.
8. Faddeev D.K. *Lektsii po algebre* [Lectures on Algebra]. Moscow, Nauka, 1984. 416 p. (In Russian)
9. Routledge N. Computing Farey Series. *Math. Gazette*, 2008, vol. 92, no. 523, pp. 55–62. doi: 10.1017/S002555720018252X.
10. Levitin A.V. *Algoritmy. Vvedenie v razrabotku i analiz* [Algorithms. Introduction into the Development and Analysis]. Moscow, Vil'yams, 2006. 576 p. (In Russian)
11. Khinchin A.Ya. *Tsepnye drobi* [Continued Fractions]. Moscow, GIFML, 1961. 112 p. (In Russian)

⟨ ***Для цитирования:*** *Еникеев Р.Р.* Приближение вещественного числа рациональным в аппроксимирующем k-арном алгоритме // Учен. зап. Казан. ун-та. Сер. Физ.-матем. науки. – 2019. – Т. 161, кн. 2. – С. 250–262. – doi: 10.26907/2541-7746.2019.2.250-262. ⟩

⟨ ***For citation:*** Enikeev R.R. Real number approximation by a rational number in the approximating k-ary algorithm. *Uchenye Zapiski Kazanskogo Universiteta. Seriya Fiziko-Matematicheskie Nauki*, 2019, vol. 161, no. 2, pp. 250–262. doi: 10.26907/2541-7746.2019.2.250-262. (In Russian) ⟩