УДК 510.25+510.52+519.688

ЭФФЕКТИВНОЕ ПО ВРЕМЕНИ И ПАМЯТИ ВЫЧИСЛЕНИЕ ЛОГАРИФМИЧЕСКОЙ ФУНКЦИИ ВЕЩЕСТВЕННОГО АРГУМЕНТА НА МАШИНЕ ШЁНХАГЕ

С. В. Яхонтов

Санкт-Петербургский государственный университет, г. Санкт-Петербург, Россия

E-mail: SergeyV.Yakhontov@gmail.com

Строится алгоритм *FLE* быстрого вычисления логарифмической функции $\ln(1+x)$ вещественного аргумента на полуинтервале $\left[2^{-5}, 1-2^{-5}\right)$ на машине Шёнхаге с оракульной функцией и даётся верхняя оценка его временной и емкостной сложности. Алгоритм *FLE* строится на основе разложения в ряд Тейлора по аналогии с алгоритмом быстрого вычисления экспоненты *FEE*, при этом дополнительно строится модифицированный алгоритм двоичного деления *ModifBinSplit* для гипергеометрических рядов. Для алгоритмов *ModifBinSplit* и *FLE* показывается квазилинейность по времени и линейность по памяти при вычислении на машине Шёнхаге, то есть принадлежность классу Sch(FQLIN-TIME//LIN-SPACE). Для расчёта логарифмической функции на произвольном промежутке используется мультипликативная редукция интервала.

Ключевые слова: *логарифмическая функция, алгоритмические вещественные функции, квазилинейная временная сложность, линейная ёмкостная сложность.*

Введение

Данная работа является продолжением работ [1 – 3], посвящённых построению алгоритмических аналогов констант и элементарных функций с ограниченной сложностью вычисления двоично-рациональных приближений при вычислении на машине Шёнхаге [4].

Приводится мера временной и емкостной сложности [1 – 4] вычислений на машине Шёнхаге и даётся верхняя оценка временной и емкостной сложности предлагаемого алгоритма вычисления логарифмической функции $\ln(1+x)$ вещественного аргумента на полуинтервале $[2^{-5}, 1-2^{-5})$ на машине Шёнхаге.

Основные сведения о двоично-рациональных числах и алгоритмических числах и функциях можно найти в [5]. Посредством Sch(FQLIN-TIME//LINS-PACE) будем обозначать, как и в [1 – 3], класс алгоритмов, квазилинейных по времени и линейных по памяти при вычислении на машине Шёнхаге. Под квазилинейностью понимается ограниченность сверху функцией вида $\mathrm{O}(n\log(n)^k)$ при некотором k.

Основной предмет нашего интереса — это алгоритмы для расчёта элементарных функций, основанные на разложениях в ряды, так как такие алгоритмы важны в практической информатике в силу своей относительной простоты реализации.

Известно, что многие вещественные элементарные функции, с одной стороны, вычислимы с помощью разложения в ряд полиномиальными по времени и линейными по памяти алгоритмами [3] и, с другой — квазилинейными по времени и квазилинейными по памяти алгоритмами [6 – 8]. В данных работах рассматривается сложность

алгоритмов при реализации на машине Тьюринга [3] и в рамках модели битовых вычислений [6 – 8], но оценки вычислительной сложности, приведённые в них, верны и для машины Шёнхаге.

Возникает вопрос: нельзя ли рассчитывать элементарные функции алгоритмами, основанными на разложениях в ряды, одновременно квазилинейными по времени и линейными по памяти на машине Шёнхаге?

В [2, 3] даётся положительный ответ на данный вопрос для экспоненциальной функции комплексного аргумента и некоторых других элементарных функций, которые выражаются через экспоненциальную функцию комплексного аргумента с помощью простых соотношений. Квазилинейные по времени и линейные по памяти алгоритмы из [2, 3] для вычисления приближённых значений экспоненциальной функции на машине Шёнхаге основаны на модифицированном методе двоичного деления для гипергеометрических рядов [9] и модифицированном методе быстрого вычисления экспоненты [6, 7].

Метод двоичного деления (англ. binary splitting) [9], предназначенный для вычисления гипергеометрических рядов, является рекурсивным вариантом метода FEE [6 – 8, 10 – 13] вычисления гипергеометрических рядов: в методе из [9] дерево вычислений обходится сверху вниз с помощью рекурсии, в отличие от итеративного метода FEE, в котором, в частности, дерево вычислений обходится снизу вверх. Будем использовать рекурсивный метод из [9]; это связано с удобством его реализации на машине Шёнхаге, в которой есть рекурсивные вызовы. К тому же при использовании рекурсии как сам алгоритм, так и оценки его вычислительной сложности получаются проще.

В данной работе подход, использованный в [2, 3] для экспоненциальной функции комплексного аргумента, применяется для логарифмической функции вещественного аргумента: алгоритм класса Sch(FQLIN-TIME//LIN-SPACE) вычисления логарифмической функции на машине Шёнхаге основан на комбинации алгоритма расчёта гипергеометрических рядов из [1 – 3] и алгоритма быстрого вычисления логарифма, который строится для логарифмической функции по схеме метода быстрого вычисления экспоненты.

Везде далее под функцией $\log(k)$ будем понимать логарифм по основанию 2; через $M_{\mathrm{Sch}}(n)$ обозначим верхнюю оценку временной сложности алгоритма Шёнхаге — Штрассена [4] умножения целых чисел на машине Шёнхаге.

1. Описание вычислительной модели (машины Шёнхаге)

Данная машина, введенная в [4], оперирует символами из алфавита $\Sigma = \{0, 1, \ldots, 2^\lambda - 1\}$ и с последовательностями таких символов, где λ — некоторая константа (данную константу можно взять, например, равной 32). Машина состоит из одномерных массивов T_0, …, T_τ для чтения и записи символов из Σ, регистров A, B, C, M для арифметических операций над символами (которые интерпретируются в данном случае как натуральные числа) и управляющего устройства CPU. Массивы потенциально бесконечны в обе стороны; для каждого массива есть указатель p_i на текущий символ, записанный в массиве. Запись $\langle p + j \rangle$ означает символ, на который ссылается указатель $p + j$. Есть также дополнительный регистр Y — указатель на текущую выполняемую инструкцию, и дополнительный массив S, потенциально бесконечный в одну сторону, который служит в качестве стека рекурсивных вызовов. Битовый регистр E служит как регистр переполнения при арифметических операциях.

Программа для машины Шёнхаге состоит из нескольких модулей-процедур на языке TPAL, который аналогичен языку ассемблера для RISC-процессоров. В TPAL имеются команды загрузки в регистр символа, записанного в массиве, чтения из регистра и запись в массив, арифметические операции, команды увеличения/уменьшения содержимого регистров, команды сдвига, вызов и возврат из процедуры, переход по метке и условный переход. Целые числа, которыми оперирует машина, кодируются как последовательности символов в алфавите Σ:

$$a = a_0 + a_1 2^\lambda + a_2 (2^\lambda)^2 + \ldots + a_{k-1}(2^\lambda)^{k-1}, \quad 0 \leqslant a_i \leqslant 2^\lambda - 1.$$

Бит знака размещается в символе, следующем за старшим символом a_{k-1}.

При вызове процедур параметры записываются в массивах, локальные переменные процедур и возвращаемые значения — также в массивах. При возврате из процедуры память, занятая под параметры и локальные переменные, освобождается.

Временная вычислительная сложность алгоритма на машине Шёнхаге определяется как количество выполняемых на ней инструкций на языке TPAL. При этом затраты на арифметические операции над символами и на вызов процедуры учитываются как некоторое константное число шагов [4]. Память, используемую программой при вычислении в массиве, определим как максимум из количества битов по всем участвующим в вычислениях элементам массива.

Поскольку конструктивные функции — это функции, вычисляющие приближения своих значений по приближениям аргумента, определим оракульную машину Шёнхаге. Данная машина имеет оракульную функцию, которая вычисляет приближения аргумента; по таким приближениям машина вычисляет приближения функции. Условимся, что запрос к оракулу в виде записи точности вычисления записывается в массиве T_0, в котором также дается приближение аргумента. При оценке временной вычислительной сложности на оракульной машине Шёнхаге запрос к оракулу учитывается как одна операция.

Определение 1 [2, 3]. *Емкостную вычислительную сложность алгоритма* при расчёте на оракульной машине Шёнхаге определим как сумму длин используемой памяти по всем массивам, сложенную с максимумом объема памяти, занятой стеком.

Отметим, что машина Шёнхаге существенно отличается от машины Тьюринга, РАМ и РАСП [14] возможностью использования рекурсивных процедур. Поэтому из верхних оценок вычислительной сложности (временной и емкостной) для машины Шёнхаге непосредственно не следуют какие-либо верхние оценки сложности для реализации тех же алгоритмов на машинах Тьюринга, РАМ или РАСП.

В то же время машина Шёнхаге может рассматриваться как паскалевидная функция с теми же верхними оценками сложности. Из основного вывода в [15] и результатов данной работы получаем, что имеется алгоритм вычисления логарифмической функции вещественного аргумента, принадлежащий FP.

2. Алгоритмические вещественные числа и функции

В данной работе основу представления конструктивных объектов составляет понятие алгоритмической последовательности φ, сходящейся по Коши [5], при этом в качестве вычислительной модели берётся машина Шёнхаге. Такая последовательность определяется на множестве всех натуральных чисел $\mathbb{N}$, включая 0, а областью аппроксимирующих значений является всюду плотное в $\mathbb{R}$ естественное подмножество

множества рациональных чисел. Для последовательности, сходящейся по Коши и задающей вещественное число x, требуют, чтобы выполнялось $|\varphi(n) - x| \leqslant 2^{-n}$ для любого натурального n.

В качестве множества аппроксимирующих значений берётся множество двоично-рациональных чисел $\mathbb{D}$ [5]. Рациональное число d называется двоично-рациональным, если $d = m/2^n$ для некоторого целого m и натурального n. Двоично-рациональные числа имеют конечное двоичное представление: строка s, равная $\pm u_p u_{p-1} \dots u_0 . v_1 v_2 \dots v_r$, обозначает число

$$d = \pm \left(\sum_{i=0}^{p} u_i 2^i + m \sum_{j=1}^{r} v_j 2^{-j} \right).$$

Длина представления двоично-рационального числа определяется как количество символов в строке s, равное, с учётом знака и двоичной точки, $p+r+3$, и обозначается $l(s)$. Под точностью представления $\mathrm{prec}(s)$ понимается число битов справа от двоичной точки, то есть r. С точки зрения изучения вычислительной сложности двоично-рациональные числа удобны тем, что для любого n двоично-рациональные числа с точностью n равномерно распределены на вещественной прямой [5].

Последовательность $\varphi : \mathbb{N} \to \mathbb{D}$ двоично-рационально сходится к вещественному числу x, если для любого $n \in \mathbb{N}$ выполняется $\mathrm{prec}(\varphi(n)) = n + 1$ и $|\varphi(n) - x| \leqslant 2^{-n}$. Множество всех функций, двоично-рационально сходящихся к вещественному числу x, обозначается CF_x. Вещественное число x называется CF-алгоритмическим [5], если CF_x содержит вычислимую функцию φ.

Вещественная функция f, заданная на отрезке $[a, b]$ (или на любом другом промежутке), называется алгоритмической функцией [5] на этом отрезке, если существует машина Шёнхаге M с оракульной функцией, такая, что для любого $x \in [a, b]$ и любой вычислимой функции $\varphi \in CF_x$ функция ψ, вычисляемая M с оракульной функцией φ, принадлежит $CF_{f(x)}$. Фактически, это означает, что для любой вычислимой функции $\varphi \in CF_x$ и любого $n \in \mathbb{N}$ машина M последовательно вычисляет $m \in \mathbb{N}$ и $d \in \mathbb{D}$, такие, что $|\varphi(m) - x| \leqslant 2^{-m}$, $|d - f(x)| \leqslant 2^{-n}$.

Сложность расчёта двоично-рациональных приближений алгоритмических чисел и функций определяется в [5] на основе длины двоичного представления точности вычисления. Память ленты запроса и ленты ответа оракульной функции при оценке емкостной вычислительной сложности алгоритма не учитывается. Обращение к оракульной функции $\varphi \in CF_x$ аргумента x алгоритмической функции осуществляется следующим образом:

— на ленту запроса оракульной функции записывается точность вычисления аргумента 2^{-m} в виде 0^m (унарная запись);
— рассчитывается значение $\varphi(m)$ оракульной функции и результат записывается на ленту ответа;
— значение $\varphi(m)$ считывается с ленты ответа в промежуточную память.

При описании алгоритмов данные шаги в явном виде здесь приводиться не будут; просто подразумеваем, что процесс вычислений содержит обращения к оракульной функции.

Определение 2 [1–3]. Число $x \in \mathbb{R}$ назовём Sch(FQLIN-TIME//LIN-SPACE)-*алгоритмическим вещественным числом*, если существует функция $\varphi \in CF_x$, вычислимая в пределах Sch(FQLIN-TIME//LIN-SPACE).

Определение 3 [2, 3]. Вещественную функцию f, заданную на отрезке $[a, b]$, назовём Sch(FQLIN-TIME//LIN-SPACE)-*алгоритмической вещественной функцией на отрезке* $[a, b]$, если для любого $x \in [a, b]$ существует Sch(FQLIN-TIME//LIN-SPACE)-вычислимая функция ψ из $CF_{f(x)}$.

Множества Sch(FQLIN-TIME//LIN-SPACE) алгоритмических вещественных чисел и функций будем обозначать Sch(FQLIN-TIME//LIN-SPACE)$_{CF}$ и Sch(FQ-LIN-TIME//LIN-SPACE)$_{C[a,b]}$ соответственно. Здесь индекс $C[a, b]$ обусловлен тем, что алгоритмические функции являются непрерывными на всей области определения [5]. Построение алгоритмического аналога вещественной функции f на отрезке $[a, b]$ означает описание алгоритма, вычисляющего двоично-рациональные приближения с произвольной точностью значений $f(x)$ для $x \in [a, b]$.

Аналогичные определения можно ввести и для любых промежутков из области определения функции.

Под приближением t с точностью 2^{-n} будем понимать двоично-рациональное число t^*, такое, что $|t^* - t| \leqslant 2^{-n}$. Для модуля t^* выполняются неравенства

$$|t^*| - |t| \leqslant |t^* - t| \leqslant 2^{-n}, \quad \text{то есть} \quad |t^*| \leqslant |t| + 2^{-n},$$
$$|t| - |t^*| \leqslant |t - t^*| \leqslant 2^{-n}, \quad \text{то есть} \quad |t^*| \geqslant |t| - 2^{-n}.$$

Далее, пусть двоично-рациональное число t^{**} таково, что $|t^{**} - t| \leqslant 2^{-(n+1)}$. Отбросим все биты t^{**} после двоичной точки, начиная с $(n+2)$-го, а новую величину обозначим t^*. Тогда $\mathrm{prec}(t^*) = n + 1$ и $|\varepsilon| = |t^* - t^{**}| < 2^{-(n+1)}$. Отсюда получаем

$$|t^* - t| \leqslant |t^* - t^{**}| + |t^{**} - t| < 2^{-(n+1)} + 2^{-(n+1)} = 2^{-n}.$$

То есть чтобы вычислить t с точностью 2^{-n}, можно рассчитать t с точностью $2^{-(n+1)}$, а затем отбросить биты полученной величины t^{**} после двоичной точки, начиная с $(n + 2)$-го.

3. Метод двоичного деления

Данный метод предназначен для вычисления значений гипергеометрических рядов частного вида с рациональными коэффициентами

$$S = \sum_{i=0}^{\infty} \frac{a(i)}{b(i)} \prod_{j=0}^{i} \frac{p(j)}{q(j)}, \tag{1}$$

где a, b, p, q — полиномы с целыми коэффициентами. Линейно сходящиеся гипергеометрические ряды используются для расчёта многих констант математического анализа и элементарных функций в рациональных точках. Ряд (1) линейно сходится, если его частичная сумма

$$S(\mu(k)) = \sum_{i=0}^{\mu(k)} \frac{a(i)}{b(i)} \prod_{j=0}^{i} \frac{p(j)}{q(j)}, \tag{2}$$

где $\mu(k)$ — линейная функция от k, отличается от точного значения не более чем на 2^{-k}:

$$|S - S(\mu(k))| \leqslant 2^{-k}.$$

В классическом варианте метод двоичного деления состоит в следующем. Обозначим $k_1 = \mu(k)$. Рассмотрим частичную сумму (2) для некоторых чисел i_1 и i_2,

$0 \leqslant i_1 \leqslant k_1$, $0 \leqslant i_2 \leqslant k_1$, $i_1 \leqslant i_2$:

$$S(i_1, i_2) = \sum_{i=i_1}^{i_2} \frac{a(i)p(i_1)\dots p(i)}{b(i)q(i_1)\dots q(i)}.$$

Будем определять величины

$$P(i_1, i_2) = p(i_1)\dots p(i_2), \quad Q(i_1, i_2) = q(i_1)\dots q(i_2), \quad B(i_1, i_2) = b(i_1)\dots b(i_2),$$
$$T(i_1, i_2) = B(i_1, i_2)Q(i_1, i_2)S(i_1, i_2).$$

Если $i_1 = i_2$, то они вычисляются напрямую. Иначе ряд делится на две части, левую и правую, и $P(i_1, i_2)$, $Q(i_1, i_2)$, $B(i_1, i_2)$ рассчитываются для каждой из частей рекурсивно. Затем полученные величины комбинируются:

$$P(i_1, i_2) = P_l P_r, \quad Q(i_1, i_2) = Q_l Q_r, \quad B(i_1, i_2) = B_l B_r, \quad T(i_1, i_2) = B_r Q_r T_l + B_l P_l T_r.$$

Алгоритм начинает свою работу с $i_1 = 0$, $i_2 = k_1$. После вычисления $T(0, k_1)$, $B(0, k_1)$, $Q(0, k_1)$ осуществляется деление $T(0, k_1)$ на $B(0, k_1)Q(0, k_1)$, чтобы получить результат с заданной точностью. Выпишем алгоритм двоичного деления в явном виде (алгоритм 1).

Алгоритм 1. *BinSplit*. Приближённое значение частичной суммы (2) с точностью 2^{-k}

Вход: Запись 0^k точности вычисления 2^{-k}
Выход: Кортеж $[P(i_1, i_2), Q(i_1, i_2), B(i_1, i_2), T(i_1, i_2)]$
1: $k_1 := \mu(k)$;
2: $[P, Q, B, T] := BinSplitRecurs(0, k_1)$;
3: осуществляем деление $r := \dfrac{T}{BQ}$ с точностью 2^{-k};
4: возвращаем результат r.

В алгоритме 1 используется подалгоритм *BinSplitRecurs* (алгоритм 2), рассчитывающий рекурсивно P, Q, B, T.

Алгоритм 2. *BinSplitRecurs*. Вычисление величин P, Q, B, T

Вход: Границы отрезка i_1, i_2
Выход: Кортеж $[P(i_1, i_2), Q(i_1, i_2), B(i_1, i_2), T(i_1, i_2)]$
1: **Если** $i_1 = i_2$, **то**
2: $\quad T := a(i_1)p(i_1)$ и возвращаем кортеж $[p(i_1), q(i_1), b(i_1), T]$;
3: $i_{\text{mid}} := \left\lfloor \dfrac{i_1 + i_2}{2} \right\rfloor$;
4: вычисляем $[P_l, Q_l, B_l, T_l] := BinSplitRecurs(i_1, i_{\text{mid}})$;
5: вычисляем $[P_r, Q_r, B_r, T_r] := BinSplitRecurs(i_{\text{mid}}, i_2)$;
6: $T := B_r Q_r T_l + B_l P_l T_r$ и возвращаем кортеж $[P_l P_r, Q_l Q_r, B_l B_r, T]$.

Длины чисел $T(0, k_1)$ и $B(0, k_1)Q(0, k_1)$ пропорциональны $k \log(k)$, то есть алгоритм двоичного деления является квазилинейным по памяти; его временная сложность — $\mathrm{O}(M_{\text{Sch}}(k) \log(k)^2)$ [6].

4. Метод быстрого вычисления $\exp(x)$

Рассмотрим ряд Тейлора вещественной экспоненциальной функции [16]

$$\exp(x) = \sum_{i=0}^{\infty} \frac{x^i}{i!} = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \ldots + \frac{x^n}{n!} + r_n(x). \tag{3}$$

Будем вычислять значение этого ряда с точностью 2^{-n} в двоично-рациональной точке x_m, $|x_m - x_0| \leqslant 2^{-m}$, $-1/4 < x_m < 1/4$. Пусть k — наименьшее число, такое, что $n+1 \leqslant 2^k$, и $m = 2^{k+1}$. Представим $x_m = \pm 0{,}00\alpha_3\alpha_4 \ldots \alpha_m\alpha_{m+1}$ в виде

$$\begin{aligned} x_m &= \pm 0{,}00\alpha_3\alpha_4 + \pm 0{,}0000\alpha_5\alpha_6\alpha_7\alpha_8 + \ldots + \\ &\quad + \pm 0{,}00\ldots 0a_{m-2^k+1}a_{m-2^k+2}\ldots\alpha_m\alpha_{m+1} = \\ &= \frac{\beta_2}{2^4} + \frac{\beta_3}{2^8} + \frac{\beta_4}{2^{16}} + \ldots + \frac{\beta_{k+1}}{2^m} = \gamma_2 + \gamma_3 + \ldots + \gamma_{k+1}, \end{aligned}$$

где $\beta_2 = \pm\alpha_3\alpha_4$; $\beta_3 = \pm\alpha_5\alpha_6\alpha_7\alpha_8$; …; $\beta_{k+1} = \pm a_{m-2^k+1}a_{m-2^k+2}\ldots\alpha_m\alpha_{m+1}$; $\gamma_\varsigma = \beta_\varsigma 2^{-2^\varsigma}$, $2 \leqslant \varsigma \leqslant k+1$; β_ς — $2^{\varsigma-1}$-значное целое число. Значение $\exp(x_m)$ запишем как произведение

$$\exp(x_m) = \exp(\gamma_2)\exp(\gamma_3)\cdot\ldots\cdot\exp(\gamma_{k+1}).$$

В методе быстрого вычисления экспоненты (методе Карацубы; англ. FEE [7]) величины $\exp(\gamma_\varsigma)$ далее рассчитываются с помощью ряда Тейлора (3):

$$\exp(\gamma_\varsigma) = 1 + \frac{\beta_\varsigma}{1!2^{2^\varsigma}} + \frac{\beta_\varsigma^2}{2!2^{2^\varsigma 2}} + \ldots + \frac{\beta_\varsigma^r}{r!2^{2^\varsigma r}} + R_\varsigma(r) = \xi_\varsigma + R_\varsigma(r). \tag{4}$$

Здесь $r = m2^{-\varsigma+1}$. Так как для остаточного члена выполняется неравенство [7]

$$|R_\varsigma(r)| < 2\frac{|\beta_\varsigma|^{r+1}}{(r+1)!2^{2^\varsigma(r+1)}},$$

то $|R_\varsigma(r)| < 2^{-m}$. Величины ξ_ς получаются из формул

$$\xi_\varsigma = \frac{a_\varsigma}{b_\varsigma}, \quad a_\varsigma = \xi_\varsigma b_\varsigma, \quad b_\varsigma = r!2^{2^\varsigma r},$$

в которых целые a_ς вычисляются с помощью некоторого последовательного процесса группировки членов ряда (4). Отметим, что в формуле для b_ς присутствует факториал r, а r меняется от $m2^{-1}$ до $m2^{-k}$. Следовательно, здесь фигурирует величина, пропорциональная $n!$, и поэтому длина промежуточных данных вычислений имеет порядок $n\log(n)$.

5. Модификация метода двоичного деления для ряда Тейлора логарифмической функции

Рассмотрим ряд Тейлора логарифмической функции [16]:

$$\ln(1+x) = \sum_{i=1}^{\infty} (-1)^{(i-1)}\frac{x^i}{i} = x - \frac{x^2}{2} + \frac{x^3}{3} - \ldots + (-1)^{(n-1)}\frac{x^n}{n} + r_n(x). \tag{5}$$

Этот ряд сходится для значений x на полуинтервале $(-1, 1]$. Остаточный член ряда (5) для $|x| < 1$ в форме Коши имеет вид [16]

$$|r_n(x)| \leqslant \frac{|x|^{n+1}}{1-|x|}\left(\frac{1-\theta}{1+\theta x}\right)^n, \text{ где } 0 < \theta < 1.$$

При $x \geqslant 0$ имеем $1 + \theta x > 1 - \theta$, и последний множитель меньше единицы. Поэтому для остаточного члена ряда на полуинтервале $[0, 1)$ получаем следующую оценку:

$$|r_n(x)| \leqslant \frac{|x|^{n+1}}{1 - |x|}.$$

Пусть $m = 2^{k+1}$, $m \geqslant 2^5$, где k — натуральное число. Будем вычислять ряд (5) с точностью 2^{-m} в точке $\gamma^{(\varsigma)} = \beta^{(\varsigma)} 2^{-\xi}$, где $\xi = 2^\varsigma - 1$:

$$\ln\left(1 + \gamma^{(\varsigma)}\right) = \frac{\beta^{(\varsigma)}}{2^\xi} - \frac{\left(\beta^{(\varsigma)}\right)^2}{2 \cdot 2^{\xi 2}} + \ldots \pm \frac{\left(\beta^{(\varsigma)}\right)^r}{r \cdot 2^{\xi r}} + R^{(\varsigma)}(r), \tag{6}$$

где ς и r — натуральные числа, такие, что $\varsigma \geqslant 1$, $r = m2^{(-\varsigma+2)}$, $\beta^{(\varsigma)}$ — $2^{(\varsigma-1)}$-значное целое число. Ряд (6) линейно сходится относительно m, так как для остаточного члена на полуинтервале $[0, 1)$ выполняется следующая оценка:

$$|R^{(\varsigma)}(r)| \leqslant 2\frac{\left|\beta^{(\varsigma)}\right|^{r+1}}{2^{\xi(r+1)}} < 2\frac{2^{2^{(\varsigma-1)}(r+1)}}{2^{\xi(r+1)}} = \frac{2}{2^{\xi m 2^{(-\varsigma+2)}}} = \frac{2}{2^{2m}} \leqslant 2^{-(m+1)},$$

то есть сходимость ряда (6) не зависит от ς.

Модифицируем метод двоичного деления для частичной суммы ряда (6) так, чтобы вычисления находились в пределах класса Sch(FQLIN-TIME//LIN-SPACE).

Возьмём $k_1 = \log(r)$, $r_1 = \lceil r/k_1 \rceil$ (k_1 — натуральное число) и запишем частичную сумму ряда (6) в виде

$$P\left(\gamma^{(\varsigma)}\right) = \sigma_1 + \tau_2[\sigma_2 + \tau_3[\sigma_3 + \ldots + \tau_{k_1-1}[\sigma_{k_1-1} + \tau_{k_1}\sigma_{k_1}]]], \tag{7}$$

где

$$\begin{aligned}
\sigma_1 &= \frac{\beta^{(\varsigma)}}{2^\xi} - \frac{\left(\beta^{(\varsigma)}\right)^2}{2 \cdot 2^{\xi 2}} + \ldots \pm \frac{\left(\beta^{(\varsigma)}\right)^{r_1-1}}{(r_1-1)2^{\xi(r_1-1)}},\\
\tau_2 &= \frac{\left(\beta^{(\varsigma)}\right)^{r_1}}{2^{\xi r_1}},\\
\sigma_2 &= \pm\frac{1}{r_1} \mp \frac{\beta^{(\varsigma)}}{(r_1+1)2^\xi} \pm \frac{\left(\beta^{(\varsigma)}\right)^2}{(r_1+2)2^{\xi 2}} \mp \ldots \pm \frac{\left(\beta^{(\varsigma)}\right)^{r_1-1}}{(2r_1-1)2^{\xi(r_1-1)}},\\
\tau_3 &= \tau_2 = \frac{\left(\beta^{(\varsigma)}\right)^{r_1}}{2^{\xi r_1}},\\
\sigma_3 &= \pm\frac{1}{2r_1} \mp \frac{\beta^{(\varsigma)}}{(2r_1+1)2^\xi} \pm \frac{\left(\beta^{(\varsigma)}\right)^2}{(2r_1+2)2^{\xi 2}} \mp \ldots \pm \frac{\left(\beta^{(\varsigma)}\right)^{r_1-1}}{(3r_1-1)2^{\xi(r_1-1)}},\\
&\ldots
\end{aligned}$$

Величины σ_t будем вычислять классическим методом двоичного деления для суммы (2), где

$$\begin{aligned}
&\mu(k) = r_1 - 1;\\
&a(i) = \pm 1, \quad b(i) = \begin{cases} (1+i), & t = 0,\\ ((t-1)r_1 + i), & t > 0; \end{cases}\\
&p(j) = \begin{cases} 1, & j = 0,\\ \beta^{(\varsigma)}, & j \neq 0; \end{cases} \quad q(j) = \begin{cases} 1, & j = 0,\\ 2^\xi, & j > 0. \end{cases}
\end{aligned} \tag{8}$$

Сформулируем утверждения об оценке вычислительной сложности реализации расчёта σ_t и τ_t в виде двух лемм.

Лемма 1. Временная сложность алгоритма двоичного деления для вычисления σ_t на машине Шёнхаге ограничена сверху $\mathrm{O}(M_{\mathrm{Sch}}(m)\log(m))$; емкостная сложность — $\mathrm{O}(m)$.

Лемма 2. Временная сложность алгоритма двоичного деления для вычисления τ_t на машине Шёнхаге ограничена сверху $\mathrm{O}(M_{\mathrm{Sch}}(m)\log(m))$; емкостная сложность — $\mathrm{O}(m)$.

Доказательство лемм 1 и 2 аналогично доказательству подобных лемм из [2, 3].

Рассчитывать приближённые значения $P\left(\gamma^{(\varsigma)}\right)^*$ с точностью $2^{-(m+1)}$ по формуле (7) будем в соответствии со следующим итеративным процессом:

$$\begin{aligned} &h_1(m_1) = \sigma^*_{k_1},\\ &\widehat{h}_i(m_1) = \sigma^*_{k_1-i+1} + \tau^*_{k_1-i+2}h_{i-1}, \quad i = 1,\ldots,k_1,\\ &h_i(m_1) = \widehat{h}_i(m_1) + \varepsilon_i; \end{aligned} \tag{9}$$

при $i = k_1$ полагаем $P\left(\gamma^{(\varsigma)}\right)^* = h_{k_1}(m_1)$. Здесь $m_1 \geqslant m$, σ^*_i — приближения σ_i с точностью 2^{-m_1}, τ^*_i — приближения τ_i с точностью 2^{-m_1}. Величины $h_i(m_1)$ получаются отбрасыванием битов $q_{m_1+1}q_{m_1+2}\ldots q_{m_1+j}$ чисел $\widehat{h}_i(m_1)$ после двоичной точки, начиная с (m_1+1)-го, то есть

$$|\varepsilon_i| = |h_i(m_1) - \widehat{h}_i(m_1)| = 0{,}0\ldots0q_{m_1+1}q_{m_1+2}\ldots q_{m_1+j}, \tag{10}$$

а знак ε_i совпадает со знаком $\widehat{h}_i(m_1)$ (ясно, что $|\varepsilon_i| < 2^{-m_1}$).

Лемма 3. Для любого $i \in \{1,\ldots,k_1\}$ справедлива оценка

$$|h_i(m_1)| < 4. \tag{11}$$

Доказательство. Применим математическую индукцию по j для $h_j(m_1)$, используя оценки

$$|\sigma_i| < 1, \quad |\tau_i| = \left(\gamma^{(\nu)}\right)^{r_1} < \frac{1}{2}.$$

База индукции при j, равном 1: $|h_1(m_1)| \leqslant \sigma_{k_1} + 2^{-m_1} < 2$. Индукционный переход для $(j+1) \geqslant 2$:

$$|h_{j+1}(m_1)| = |\sigma^*_{k_1-(j+1)+1} + \tau^*_{k_1-(j+1)+2}h_j + \varepsilon_{j+1}| < 1 + \left[\frac{1}{2} + 2^{-m_1}\right]4 + 2^{-m_1} < 4.$$

Лемма доказана. ■

Лемма 4. Погрешность вычисления $h_{k_1}(m_1)$ по схеме (9) оценивается как

$$\Delta(k_1, m_1) < 2^{-m_1+k_1+3}.$$

Доказательство. Обозначим

$$H_1 = \sigma_{k_1}, \quad H_i = \sigma_{k_1-i+1} + \tau_{k_1-i+2}H_{i-1}, \quad \eta(i, m_1) = |h_i(m_1) - H_i|.$$

Воспользуемся методом математической индукции для $\eta(j, m_1)$ по j. База индукции при j, равном 1:

$$\eta(1, m_1) = |h_1(m_1) - H_1| = |\sigma^*_{k_1} - \sigma_{k_1}| < 2^{-m_1+4}.$$

Индукционный переход для $(j+1) \geqslant 2$:

$$\eta(j+1, m_1) = |\sigma^*_{k_1-(j+)+1} + \tau^*_{k_1-(j+1)+2} h_j(m_1) + \varepsilon_{j+1} - \sigma_{k_1-(j+1)+1} - \tau_{k_1-(j+1)+2} H_j| <$$
$$< |\tau^*_v h_j(m_1) - \tau_v h_j(m_1) + \tau_v h_j(m_1) - \tau_v H_j| + 2 \cdot 2^{-m_1} \leqslant 2^{-m_1} h_j(m_1) + 2^{-1}\eta(j, m_1) + 2 \cdot 2^{-m_1}.$$

Так как из (11) $|h_j(m_1)| < 4$ и, по индукционному предположению, $\eta(j, m_1) < 2^{-m_1+j+3}$, то

$$\eta(j+1, m_1) < 4 \cdot 2^{-m_1} + 2^{-1} 2^{-m_1+j+3} + 2 \cdot 2^{-m_1} < 2^{-m_1+(j+1)+3}.$$

Из $\Delta(k_1, m_1) = \eta(k_1, m_1)$ получаем искомое неравенство. ■

Из леммы 4 следует, что достаточно взять $m_1 = 2m + 3$, чтобы вычислять $P\left(\gamma^{(\varsigma)}\right)$ с точностью $2^{-(m+1)}$.

Обозначим алгоритм расчета частичной суммы ряда (6), использующий схему (9), через *ModifBinSplit* (modified binary splitting, алгоритм 3).

Алгоритм 3. *ModifBinSplit*. Приближенное значение ряда Тейлора логарифмической функции

Вход: Запись 0^m точности вычисления 2^{-m}
Выход: Приближённое значение ряда (6) с точностью 2^{-m}

1: $m_1 := 2m + 3$
2: $h := \sigma^*_{k_1}$ (с помощью обычного алгоритма двоичного деления с точностью 2^{-m_1})
3: **Для всех** $i = 2, 3, \ldots, k_1$
4: рассчитываем $a := \sigma^*_{k_1-i+1}$ с точностью 2^{-m_1} с помощью обычного алгоритма двоичного деления и $b := \tau^*_{k_1-i+2}$ с точностью 2^{-m_1};
5: вычисляем выражение $\widehat{h} := a + b \cdot h$;
6: h присваиваем величину $\widehat{h}$, округленную в соответствии с (10);
7: на выход записываем h.

Оценим временную вычислительную сложность алгоритма 3 при расчёте на машине Шёнхаге:

— $\mathrm{O}(\log(m))$ вычислений σ_t дают $\mathrm{O}(M_{\mathrm{Sch}}(m)\log(m)^2)$;
— $\mathrm{O}(\log(m))$ вычислений τ_t дают $\mathrm{O}(M_{\mathrm{Sch}}(m)\log(m)^2)$;
— $\mathrm{O}(\log(m))$ умножений чисел длины $\mathrm{O}(m)$ дают $\mathrm{O}(M_{\mathrm{Sch}}(m)\log(m))$;

итого получаем $\mathrm{O}(M_{\mathrm{Sch}}(m)\log(m)^2)$. Емкостная сложность модифицированного алгоритма двоичного деления *ModifBinSplit* — это $\mathrm{O}(m)$, так как во всех вычислениях в данном алгоритме фигурируют числа длины $\mathrm{O}(m)$.

Утверждение 1. Модифицированный алгоритм *ModifBinSplit* двоичного деления для вычисления ряда Тейлора логарифмической функции принадлежит классу алгоритмов Sch(FQLIN-TIME//LIN-SPACE).

6. Быстрое вычисление функции $\ln(1+x)$

Пусть $u = 1 + x$, $\sigma^{(x)}(5)$ и $\sigma^{(u)}(5)$ — полуинтервалы $[2^{-5}, 1 - 2^{-5})$ и $[1 + 2^{-5}, 2 - 2^{-5})$ на вещественной прямой соответственно.

Построим метод вычисления функции $\ln(1 + x)$ на полуинтервале $\sigma^{(x)}(5)$, аналогичный быстрому методу вычисления экспоненциальной функции, и назовём его, по аналогии с методом быстрого вычисления экспоненты, быстрым методом вычисления логарифмической функции.

Будем вычислять приближённое значение $\ln(u_0)^*$ с точностью 2^{-n}, где $u_0 = 1 + x_0$, $u_0 \in \sigma^{(u)}(5)$, $n \geqslant 2^4$. Возьмём натуральные k и m, такие, что

$$k \geqslant \min\{j : n \leqslant 2^j\},\ m = 2^{k+1}. \tag{12}$$

Пусть u_m — двоично-рациональное приближение аргумента u_0 с точностью 2^{-m}, то есть $|u_m - u_0| \leqslant 2^{-m}$. Так как $u_m \in [1, 2)$, то двоично-рациональная запись числа u_m есть $u_m = 1{,}\alpha_1^{(1)}\alpha_2^{(1)} \ldots \alpha_m^{(1)}\alpha_{m+1}^{(1)}$.

Обозначим u_m через $u^{(1)}$; возьмём $v^{(1)} = 1{,}\alpha_1^{(1)}$ и вычислим $u^{(2)} = \dfrac{u^{(1)}}{v^{(1)}}$ с точностью 2^{-m}, то есть $u^{(2)} = \dfrac{u^{(1)}}{v^{(1)}} + \xi^{(2)}$, где $|\xi^{(2)}| \leqslant 2^{-m}$. Легко проверить, что первый символ в двоичной записи величины $u^{(2)}$ — это 0, то есть $u^{(2)} = 1{,}0\alpha_2^{(2)}\alpha_3^{(2)} \ldots \alpha_m^{(2)}\alpha_{m+1}^{(2)}$. В силу соотношения $\ln(u) = \ln\left(\dfrac{u}{v} \cdot v\right) = \ln\left(\dfrac{u}{v}\right) + \ln(v)$ имеем следующее равенство:

$$\ln\left(u^{(1)}\right) = \ln\left(\frac{u^{(1)}}{v^{(1)}}v^{(1)}\right) = \ln\left(u^{(2)} - \xi^{(2)}\right) + \ln\left(v^{(1)}\right) = \ln\left(1 - \frac{\xi^{(2)}}{u^{(2)}}\right) + \ln\left(u^{(2)}\right) + \ln\left(v^{(1)}\right).$$

Далее, возьмём $v^{(2)} = 1{,}0\alpha_2^{(2)}\alpha_3^{(2)}$ и поделим $u^{(2)}$ на $v^{(2)}$ с точностью 2^{-m}. В результате получим соотношение

$$\ln\left(u^{(1)}\right) = \ln\left(1 - \frac{\xi^{(2)}}{u^{(2)}}\right) + \ln\left(1 - \frac{\xi^{(3)}}{u^{(3)}}\right) + \ln\left(u^{(3)}\right) + \ln\left(v^{(1)}\right) + \ln\left(v^{(2)}\right).$$

Затем берём $v^{(3)} = 1{,}000\alpha_4^{(3)}\alpha_5^{(3)}\alpha_6^{(3)}\alpha_7^{(3)}$ и выводим аналогичную формулу для $\ln\left(u^{(1)}\right)$. Возьмём натуральное число $p = k+2$. На p-м шаге данного процесса, начиная с шага 2, получим следующую формулу для $\ln\left(u^{(1)}\right)$:

$$\ln\left(u^{(1)}\right) = \sum_{i=2}^{p} \ln\left(1 + \frac{\xi^{(i)}}{u^{(i)}}\right) + \ln\left(u^{(p)}\right) + \sum_{i=1}^{p-1} \ln\left(v^{(i)}\right) = \sum_{i=2}^{p} F_i + G + \sum_{i=1}^{p-1} H_i.$$

Оценим сверху величины F_i и G, используя неравенство $|\ln(1 + x)| < |x|$ (данное неравенство следует из свойств знакопеременных рядов [10]). Так как $|u^{(i)}| \geqslant 1$, то $\dfrac{\xi^{(i)}}{u^{(i)}} \leqslant 2^{-m}$ и $F_i < 2^{-m}$. Далее, так как $u^{(p)} = 1 + \zeta^{(p)}$, где $|\zeta^{(p)}| \leqslant 2^{-m}$, то $G < 2^{-m}$.

Получаем, что если вычислять величины H_i с точностью 2^{-m}, то погрешность вычисления $\ln\left(u^{(1)}\right)$ оценивается следующим образом:

$$\varepsilon_1(m) = \left|\ln\left(u^{(1)}\right)^* - \ln\left(u^{(1)}\right)\right| < 4p \cdot 2^{-m} = (\log(m) + 2)2^{-m+2}.$$

Если взять $m \geqslant 2n$ (что согласуется с (12)), то $\varepsilon_1(m) < 2^{-(n+2)}$. Далее, так как

$$\varepsilon_2 = \left|\ln\left(u^{(1)}\right) - \ln(u_0)\right| = \left|\ln\left(u_0 + \zeta^{(1)}\right) - \ln(u_0)\right| = \left|\ln\left(1 + \frac{\zeta^{(1)}}{u_0}\right)\right| < 2^{-m},$$

где $|\zeta^{(1)}| \leqslant 2^{-m}$, то получаем оценку погрешности вычисления функции $\ln(u)$ в точке u_0:

$$\left|\ln\left(u^{(1)}\right)^* - \ln\left(u_0\right)\right| \leqslant \varepsilon_1 + \varepsilon_2 < 2^{-(n+1)}.$$

Вычислять приближения величин H_ς, то есть величины $\ln\left(v^{(\varsigma)}\right)^*$, будем с помощью алгоритма *ModifBinSplit* для ряда (6); для этого запишем $v^{(\varsigma)}$ для $1 \leqslant \varsigma \leqslant (p-1)$ в виде $1+\gamma^{(\varsigma)}$, где

$$\begin{aligned}
&\gamma^{(\varsigma)} = \beta^{(\varsigma)} 2^{-(2^\varsigma - 1)},\\
&\beta^{(1)} = \alpha_1^{(1)},\\
&\beta^{(2)} = \alpha_2^{(2)}\alpha_3^{(2)},\\
&\beta^{(3)} = \alpha_4^{(3)}\alpha_5^{(3)}\alpha_6^{(3)}\alpha_7^{(3)},\\
&\dots
\end{aligned}$$

Здесь $\beta^{(\varsigma)}$ — $2^{(\varsigma-1)}$-значное целое число.

Алгоритм 4. *FLE* (fast logarithm evaluation). Быстрое вычисление логарифмической функции вещественного аргумента

Вход: Запись 0^n точности вычисления 2^{-n}. **Оракульные функции:** φ_x аргумента x
Выход: Приближённое значение $\ln(1+x)$ с точностью 2^{-n} на полуинтервале $[2^{-5}, 1-2^{-5})$

1: Возьмём k и m так, чтобы выполнялось (12);
2: $p := k+2$;
3: $u^{(1)} := \varphi_x(m)$;
4: $S := 0$ (S — двоично-рациональное число).
5: **Для всех** $i = 2, 3, \dots, p$
6: вычисляем $u^{(i)} = \dfrac{u^{(i-1)}}{v^{(i-1)}}$;
7: рассчитываем $H^{(i)} := \ln\left(1+\gamma^{(i)}\right)^*$ с помощью алгоритма *ModifBinSplit* с точностью 2^{-m};
8: $S := S + H^{(i)}$;
9: на выход записываем значение S, округленное до точности 2^{-n}.

Вычислительная сложность алгоритма 4 при расчёте на машине Шёнхаге следующая: временная сложность — $\mathrm{O}(M_{\mathrm{Sch}}(n)\log(n)^3)$, так как алгоритм расчёта гипергеометрических рядов *ModifBinSplit* использует $\mathrm{O}(M_{\mathrm{Sch}}(n)\log(n)^2)$ операций, а в алгоритме *FLE* используется $\mathrm{O}(\log(n))$ таких вычислений и $\mathrm{O}(\log(n))$ сложений чисел длины $\mathrm{O}(n)$; емкостная сложность — $\mathrm{O}(n)$, так как во всех вычислениях фигурируют числа такой длины.

Утверждение 2. Алгоритм *FLE* быстрого вычисления логарифмической функции вещественного аргумента принадлежит классу вычислительной сложности на машине Шёнхаге Sch(FQLIN-TIME//LIN-SPACE).

Заключение

Алгоритм *FLE* расчета логарифмической функции можно применять в информатике как основу Sch(FQLIN-TIME//LIN-SPACE)-алгоритмической функции $\ln(1+x)$,

заданной на подмножестве множества Sch(FQLIN-TIME//LIN-SPACE) алгоритмических вещественных чисел.

Отметим, что если для умножения использовать рекурсивный метод Карацубы [17] с временной сложностью $\mathrm{O}(n^{\log 3})$, то временная сложность алгоритма расчёта логарифмической функции *FLE* равна $\mathrm{O}(n^{\log 3}\log^3 n)$, то есть она полиномиальна степени меньше чем 2.

Для построения алгоритма расчёта логарифмической функции на произвольном промежутке можно использовать мультипликативную редукцию интервала [18] по аналогии с тем, как осуществляется мультипликативная редукция интервала для логарифмической функции в [3].

ЛИТЕРАТУРА

1. *Яхонтов С. В.* Вычисление гипергеометрических рядов с квазилинейной временной и линейной емкостной сложностью // Вестник Самарского государственного технического университета. Сер. Физико-математические науки. 2011. Вып. 2 (17). С. 239–249.
2. *Яхонтов С. В.* Эффективное по времени и по памяти вычисление экспоненциальной функции комплексного аргумента на машине Шёнхаге // Вестник С.-Петербург. ун-та. Сер. 10. Прикладная математика. Информатика. Процессы управления. 2011. Вып. 4. С. 97–110.
3. *Яхонтов С. В., Косовский Н. К., Косовская Т. М.* Эффективные по времени и памяти алгоритмические приближения чисел и функций: учеб. пособие. СПб., 2012. 256 с.
4. *Schonhage A., Grotefeld A. F. W, and Vetter E.* Fast algorithms. A multitape Turing machine implementation. Leipzig: Brockhaus AG, 1994. 298 p.
5. *Ko K.* Complexity theory of real functions. Boston: Birkhauser, 1991. 310 p.
6. *Карацуба Е. А.* Быстрое вычисление $\exp(x)$ // 17-я Всесоюз. школа по теории информации и её приложениям. Проблемы передачи информации. 1990. Т. 26. № 3. С. 109.
7. *Карацуба Е. А.* Быстрые вычисления трансцендентных функций // Проблемы передачи информации. 1991. Т. 27. Вып. 4. С. 76–99.
8. *Карацуба Е. А.* Fast evaluation of hypergeometric function by FEE // Proc. 3rd CMFT conference on computational methods and function theory, Nicosia, Cyprus, October 13–17, 1997. Singapore: World Scientific, 1999. Ser. Approx. Decompos. V. 11. P. 303–314.
9. *Haible B. and Papanikolaou T.* Fast multiple-presicion evaluation of series of rational numbers // Proc. of the Third Intern. Symposium on Algorithmic Number Theory, Portland, Oregon, USA. June 21–25, 1998. P. 338–350.
10. *Карацуба Е. А.* Быстрое вычисление $\zeta(3)$ // Проблемы передачи информации. 1993. Т. 29. № 1. С. 68–73.
11. *Karatsuba C. A.* Fast evaluation of Bessel functions // Integral Transforms and Special Functions. 1993. V. 1. No. 4. P. 269–276.
12. *Карацуба Е. А.* Быстрое вычисление дзета-функции Римана $\zeta(s)$ для целых значений аргумента s // Проблемы передачи информации. 1995. Т. 31. No. 4. С. 69–80.
13. *Карацуба Е. А.* Быстрое вычисление дзета-функции Гурвица и L-рядов Дирихле // Проблемы передачи информации. 1998. Т. 34. № 4. С. 342–353.
14. *Ахо А., Хопкрофт Дж., Ульман Дж.* Построение и анализ вычислительных алгоритмов. М.: Мир, 1979. 536 с.
15. *Косовская Т. М., Косовский Н. К.* Принадлежность классу FP дважды полиномиальных паскалевидных функций над подпрограммами из FP // Компьютерные инструменты в образовании. 2010. № 3. С. 3–7.

16. *Фихтенгольц Г. М.* Курс дифференциального и интегрального исчисления. М.: Физматлит, 2003. Т. 2.
17. *Карацуба А. А., Офман Ю. П.* Умножение многозначных чисел на автоматах // Доклады АН СССР. 1962. Т. 145. № 2. С. 293–294.
18. *Muller J.-M.* Elementary functions. Algorithms and implementation. Boston: Birkhauser, 1997. 204 p.