УДК 519.7

Р. Ю. Шлаустас [1]

[1] *Иркутский государственный университет путей сообщения, г. Иркутск, Российская федерация*

МЕТОД СВЕРХБЫСТРОЙ СОРТИРОВКИ МАССИВОВ

Аннотация. Рассматривается построение нового оригинального метода внутренней сортировки массивов, обладающего значительно более высокими качественными показателями даже в худшем случае по сравнению с известным методом быстрой сортировки Хоара.

Ключевые слова: сортировка массивов, внутренняя сортировка, быстрая сортировка.

R. Yu. Shlaustas [1]

[1] *Irkutsk State Transport University, Irkutsk, the Russian Federation*

METHOD SUPERFAST SORTING MASSIVE

Abstract. Considering the construction of a new original method of internal sorting of arrays, which has significantly higher quality indicators even in the worst case than the known method quick sorting of Hoare.

Keywords: sorting arrays, internal sorting, quick sorting.

Быстрая сортировка [1-3], разработанная Хоаром в 1960г., в наихудшем случае выполняет примерно $\frac{N^2}{2}$ операций сравнения. В наиболее благоприятном случае необходимо $N\log_2 N$ операций сравнения. В среднем требуется $2N\ln N$ таких операций. Обычная реализация быстрой сортировки оформляется в виде рекурсивной процедуры с присущими рекурсиям недостатками. Нерекурсивные версии также активно используют стек.

Пусть $S(N)$ — минимальное число сравнений N элементов. Тогда число перестановок из N элементов должно удовлетворять неравенству [2] $N! \leq 2^{S(N)}$.

Логарифмируя и применяя формулу Стирлинга для факториала получим, что

$$S(N) \geq N\log_2 N - N/\ln 2 + \frac{1}{2}\log_2 N + O(1) \quad (1)$$

В формуле (1) осуществляется округление к большему целому числу. Правая часть (1) — теоретико-информационная нижняя оценка числа сравнений. Видим, оценка (1) дает лучший результат, чем быстрая сортировка. Оценка (1) имеет место в случае отсутствия информации о начальной отсортированности массива. Поэтому имеет смысл задача построения более экономичных по числу сравнений методов сортировки. Этой задаче и посвящена данная статья.

Описание метода

Предлагаемый ниже алгоритм основан на методе простого слияния, предназначенного, в основном, для внутренней сортировки. Но нужно отметить, что начальная стадия существенно отличается. Классический алгоритм простого слияния по качественным показателям лежит между алгоритмами пирамидальной и быстрой сортировки [2, 3].

Для простоты построения оценок числа сравнений целого числа цепочек элементов положим, что число элементов массива имеет вид

$$N = 3 \cdot 2^k \quad (2)$$

На начальном шаге делим исходный массив на тройки чисел. Отсюда ясно почему принято представление (2). Каждую тройку элементов сортируем отдельно. Потребуется минимально два сравнения C и три сравнения в максимальном случае. Два сравнения нужны, если элементы изначально расположены в нужном порядке. В этом случае никаких перемещений элементов массива делать не нужно. Два сравнения также нужны, если первый и третий

элементы нужно поменять местами и осуществить два перемещения элементов. В других случаях — три сравнения и три перемещения M=3.

На каждом шаге выполняется простое слияние двух цепочек одинаковой длины, каждая из которых заранее отсортирована. На первом шаге сливаются две следующие друг за другом цепочки по 3 элемента. На втором шаге две цепочки из 6 элементов. На последнем k-м шаге сливаем две отсортированные на предыдущем шаге цепочки длины $3 \cdot 2^{k-1}$.

Необходимое число сравнений в максимальном случае не превышает уменьшенной на единицу суммы длин обеих цепочек. Минимальное число сравнений равно длине цепочки в паре. Число перемещений элементов на каждом шаге минимально равно нулю, а максимальное число копирований во вспомогательный массив совпадает с максимальным числом сравнений. Сказанное отражено в нижеследующей таблице 1

Таблица 1.

	Шаг 0	Шаг 1	Шаг 2	Шаг 3		Шаг k
Длина цепочки (цепочек)	3	3 3	$6 = 3 \cdot 2$ $6 = 3 \cdot 2$	$12 = 3 \cdot 2^2$ $12 = 3 \cdot 2^2$		$3 \cdot 2^{k-1}$ $3 \cdot 2^{k-1}$
Количество (пар) цепочек	2^k	2^{k-1}	2^{k-2}	2^{k-3}		1
Минимальное число сравнений на цепочку в шаге	2	3	$3 \cdot 2$	$3 \cdot 2^2$		$3 \cdot 2^{k-1}$
Максимальное число сравнений на цепочку в шаге	3	$3 \cdot 2 - 1$	$3 \cdot 2^2 - 1$	$3 \cdot 2^3 - 1$		$3 \cdot 2^k - 1$

Почему число сравнений элементов на каждом шаге в последней строке на единицу меньше суммы числа элементов в обеих цепочках? Это становится ясным из того факта, что после каждого сравнения общее число сравниваемых элементов уменьшается на единицу. Минимальное число сравнений возможно только в случае расположения элементов в цепочках на каждом шаге таким образом, что значения элементов одной цепочки все меньше или все больше значений элементов другой цепочки.

В соответствии с таблицей легко получить минимальное и максимальное число сравнений элементов. Для минимального значения нужно составить сумму произведений значений ячеек третьей и четвертой строк

$$C_{min} = 2^k \cdot 2 + 2^{k-1} \cdot 3 + 2^{k-2} \cdot 3 \cdot 2 + 2^{k-3} \cdot 3 \cdot 2^2 + \cdots +$$

$$+3 \cdot 2^{k-1} = 3 \cdot 2^{k-1}(k+1) + 2^k \qquad (3)$$

Аналогично, максимальное число сравнений получается суммированием произведений значений ячеек третьей и пятой строк

$$C_{max} = 2^k \cdot 3 + 2^{k-1} \cdot (3 \cdot 2 - 1) + 2^{k-2} \cdot (3 \cdot 2^2 - 1) +$$

$$+2^{k-3} \cdot (3 \cdot 2^3 - 1) + \cdots + (3 \cdot 2^k - 1) = 3 \cdot 2^k(k+1) - 2^k + 1 \qquad (4)$$

Значение C_{min} из (3) незначительно больше (относительно) половины значения C_{max} из (4). Формулы (3) и (4) соответствуют оценке Седжвика [1], в соответствии с которой число сравнений имеет порядок

$\sim N\log_2 N \sim 3 \cdot 2^k(k + \log_2 3)$ с константой, меньшей единицы.

Проведем сравнение C_{max} из (4) со значением наилучшего для быстрой сортировки Хоара при N из (2)

$$C_{Xoap\min} = N\log_2 N = 3 \cdot 2^k(k + \log_2 3) \qquad (5)$$

Очевидно, что худшие показатели предлагаемого алгоритма (4) существенно превышают наилучшие для метода Хоара (5).

Наконец, вычислим разность значений формул (1) и (4).

$$C_{max} - S(N) \le [3 \cdot 2^k(k+1) - 2^k + 1] -$$

$$-\left[3 \cdot 2^k(k + \log_2 3) - 3 \cdot 2^k \log_2 e + \frac{1}{2}(k + \log_2 3) + O(1)\right] =$$

$$= 2^k[2 - 3(\log_2 3 - \log_2 e)] - \frac{k}{2} + O(1) \approx$$

$$\approx 1.57 \cdot 2^k - \frac{k}{2} + O(1) \qquad (6)$$

Таким образом, формула (6) показывает, что наш метод дает очень хороший результат — число сравнений отличается от нижней границы на число, примерно в два раза меньшее размера массива.

Аналогичным образом

$$C_{min} - S(N) \le [3 \cdot 2^{k-1}(k+1) + 2^k] -$$

$$-\left[3 \cdot 2^k(k + \log_2 3) - 3 \cdot 2^k \log_2 e + \frac{1}{2}(k + \log_2 3) + O(1)\right] =$$

$$= -3 \cdot 2^k k + 2^{k-1}[5 - 6(\log_2 3 - \log_2 e)] - \frac{k}{2} + O(1) \approx$$

$$\approx -3 \cdot 2^k(k - 1{,}38) - \frac{k}{2} + O(1)$$

Отрицательное значение в предыдущей формуле возникает из-за «хорошей» начальной отсортированности массива.

Если число элементов не соответствует (2), то последняя цепочка элементов во все шагах $i \le k$ будет неполной, но, естественно, алгоритм остается работоспособным.

Достоинство предлагаемого метода в том, что он устойчив (сохраняет порядок сортируемых элементов, что бывает важным при сортировке текстовых данных или данных сложной структуры) в отличие от стандартного метода быстрой сортировки, а также легко допускает распараллеливание, может быть, кроме последнего шага.

Другое достоинство — нерекурсивная процедура сортировки легко строится, что позволяет избежать проблем со стеком программы.

Еще одно преимущество — можно применять как для внутренней сортировки, так и внешней. При этом для внешней сортировки можно организовать такие фрагменты, которые помещаем в оперативную память и сортируем. Только при увеличении длин цепочек выше некоторого предела, зависящего от наличия оперативной памяти, осуществить внешнюю сортировку с помощью файловых операций.

Из недостатков отметим необходимость в дополнительной оперативной памяти размером в сортируемый массив. Это такой же недостаток, как и у стандартного метода слияния.

Замечание 1. Первоначальное разбиение на тройки элементов существенно. При начальном разбиении на двойки элементов получим оценки, соответствующие стандартному методу слияния, т.е. хуже (3) и (4). При разбиении на нулевом шаге на четверки элементов также не получим лучший результат. Это следует из того, что, сортировка четверок чисел требует пяти сравнений элементов (см., например, [1]). Далее, для обоснования сказанного, для четверок чисел из (2) имеем

$$N = 4 \cdot 2^k - 2^k = 4 \cdot 2^k - 4 \cdot 2^{k-2} \qquad (7)$$

Для первого слагаемого (7), аналогично (4) получим

$$C_1 = 2^k \cdot 5 + 2^{k-1} \cdot (4 \cdot 2 - 1) + 2^{k-2} \cdot (4 \cdot 2^2 - 1) +$$

$$+2^{k-3} \cdot (4 \cdot 2^3 - 1) + \cdots + (4 \cdot 2^k - 1) = 4 \cdot 2^k(k+1) + 1 \quad . \qquad (8)$$

Из (8) легко получить оценку и для второго слагаемого заменив k на $k-2$. В итоге

$$C_{4\max} = 4 \cdot 2^k(k+1) - 2^k(k+1-2) = 3 \cdot 2^k(k+1) + 2^{k+1}. \quad (9)$$

Значение числа сравнений по формуле (9) явно хуже, чем из (4). Наблюдается ситуация, полностью аналогичная системам счисления. Наиболее выгодной из целочисленных является троичная система счисления, обладающая наибольшей плотностью записи информации [5].

Замечание 2. Недостатком метода является двойное потребление памяти. Но в настоящее время этот показатель не является критичным — теперь 64-х разрядные компьютеры обладают значительными объемами памяти как оперативной, так и дисковой.

Заключение

Предлагаемый метод несколько улучшает оценки по числу сравнений с методом Хоара, но формула (6) говорит о том, что можно пытаться строить еще более эффективные методы. И, второе, можно строить методы с минимальным потреблением дополнительной памяти. Правда, в данном случае ограничения в настоящий момент не столь существенны ─ оперативной памяти у нынешних компьютеров много и вполне достаточно для большинства практических задач. Более актуальной является проблема быстродействия.

При применении метода в практических расчетах можно применять указатели на исходный сортируемый массив и вспомогательный. После каждого шага сортировки указатели менять местами, либо на каждом четном шаге метода в качестве исходного брать вспомогательный массив, исходный как вспомогательный и затем опять менять их местами. При работе с указателями программный код будет меньше.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Седжвик Роберт. Фундаментальные алгоритмы на C++. Части 1-4 – М.: Diasoft, 2001. — 687 с.
2. Кнут Дональд. Искусство программирования. Т.3 ─. М.: Вильямс, 2002. ─ 800с.
3. Вирт Н. Алгоритмы и структуры данных. ─ М.: Мир, 1989. ─ 360с.
4. https://ru.wikibooks.org/wiki/%D0%A0%D0%B5%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%B8_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D0%BE%D0%B2/%D0%A1%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0/%D0%A1%D0%BB%D0%B8%D1%8F%D0%BD%D0%B8%D0%B5%D0%BC [Электронный ресурс, Доступ 30.07.2017].
5. https://ru.wikibooks.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B_%D1%81%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F#.D0.A2.D1.80.D0.BE.D0.B8.D1.87.D0.BD.D0.B0.D1.8F_.D1.81.D0.B8.D1.81.D1.82.D0.B5.D0.BC.D0.B0_.D1.81.D1.87.D0.B8.D1.81.D0.BB.D0.B5.D0.BD.D0.B8.D1.8F [Электронный ресурс, Доступ 20.08.2017].

REFERENCES

1. Robert Sedgwick. Fundamental algorithms in C++. Parts 1-4-M.: Diasoft, 2001. - 687 PP.
2. Whip Donald. The art of programming. Vol. Moscow: Williams, 2002. "800C.
3. Wirth N. Algorithms and data structures. - Moscow: Mir, 1989. - 360C.
4.https://ru.wikibooks.org/wiki/%D0%A0%D0%B5%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%B8_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D0%BE%D0%B2/%D0%A1%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0/%D0%A1%D0%BB%D0%B8%D1%8F%D0%BD%D0%B8%D0%B5%D0%BC [Электронный ресурс, Доступ 30.07.2017].
5.https://ru.wikibooks.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B_%D1%81%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F#.

D0.A2.D1.80.D0.BE.D0.B8.D1.87.D0.BD.D0.B0.D1.8F_.D1.81.D0.B8.D1.81.D1.82.D0.B5.D0.BC.D0.B0_.D1.81.D1.87.D0.B8.D1.81.D0.BB.D0.B5.D0.BD.D0.B8.D1.8F [Электронный ресурс, Доступ 20.08.2017].

Информация об авторе

Шлаустас Ромас Юргевич — к. ф.-м. н., доцент кафедры «Информационные системы и защита информации», Иркутский государственный университет путей сообщения, г. Иркутск, e-mail: shlaustas@gmail.com

Author

Shlaustas Romas Yurgevitch — Ph.D., in phisics and mathematics, Assistant Professor of “Information Systems and Information Protection”, Irkutsk State Transport University, Irkutsk, e-mail: shlaustas@gmail.com

Для цитирования

Шлаустас Р. Ю. Криптографическое применение трансцендентных функций. // «Информационные технологии и математическое моделирование в управлении сложными системами»: электрон. науч. журн. – 2019. – №3. – С. 56-60 – Режим доступа: http://ismm-irgups.ru/toma/34-2019, свободный. – Загл. с экрана. – Яз. рус., англ. (дата обращения: 20.11.2019)

For citation

Shlaustas R.Yu. Kriptograficheskoe primenenie transtsendentnykh funktsiy [Cryptographic application of transcendental functions] // Informacionnye tehnologii i matematicheskoe modelirovanie v upravlenii slozhnymi sistemami: ehlektronnyj nauchnyj zhurnal [Information technology and mathematical modeling in the management of complex systems: electronic scientific journal], 2019. No. 3. P. 56-60. [Accessed 20/11/19]