DOI: 10.25586/RNU.V9187.21.02.P.020 УДК 621.396

А.Б. Артюшкин, В.П. Обрученков, М.А. Бельских

РЕАЛИЗАЦИЯ МЕТОДА ЭКОНОМНОГО КОДИРОВАНИЯ ХАФФМАНА В ИНФОРМАЦИОННО-ТЕЛЕМЕТРИЧЕСКИХ СИСТЕМАХ

Описывается подход к сокращению времени обработки данных при их кодировании экономным кодом Хаффмана, что позволяет значительно расширить возможности цифровых систем связи. Предлагаются два алгоритма – сортировки и поиска позиции числа в упорядоченном числовом массиве. Установлено, что экономия времени достигается сокращением количества требуемых элементарных операций за счет использования дополнительных объемов рабочей памяти кодера. Выявлено, что временная сложность алгоритма сортировки составляет $O(3N)$, а при совместном со вторым алгоритмом использовании в кодере Хаффмана – $O(2N)$. При этом временную сложность метода Хаффмана удается снизить с вида $O(Nk \log N)$ до вида $O(Nk)$.
Ключевые слова: метод Хаффмана, экономное кодирование, сортировка данных, сложность алгоритмов.

A.B. Artyushkin, V.P. Obrychenkov, M.A. Belskikh

IMPLEMENTATION OF THE HUFFMAN ECONOMIC CODING METHOD IN INFORMATION TELEMETRIC SYSTEMS

An approach is described that makes it possible to significantly reduce the processing time of data when they are encoded with an economical Huffman code, which significantly expands the capabilities of digital communication systems. Two algorithms are proposed: a sorting algorithm and an algorithm for finding the position of a number in an ordered numeric array. It was found that time saving is achieved by reducing the number of required elementary operations due to the use of additional volumes of the encoder's working memory. It was revealed that the time complexity of the sorting algorithm is $O(3N)$, and when it is used together with the second algorithm in the Huffman encoder, it is $O(2N)$. In this case, the time complexity of the Huffman method can be reduced from the form $O(N k \log N)$ to the form $O(Nk)$.
Keywords: Huffman method, economical coding, data sorting, complexity of algorithms.

Вводные замечания

Необходимость в контроле все более сложных технических средств, объединение их в системы, функционирование которых обеспечивается за счет обмена внутрисистемной информацией, все больше актуализирует вопросы телеметрического обеспечения. В той или иной постановке требуют решения вопросы увеличения количества и частоты опроса телеметрируемых параметров, миниатюризации телеметрических средств, минимизации удельного расхода энергии на единицу информации, увеличения скорости передачи информации [7].

Качественный рост современной электронной базы в той или иной мере способствует решению данных вопросов, однако часто решение задачи «в лоб» только за счет повышения скорости выполнения операций и увеличения доступных объемов памяти не является оптимальным, а иногда и вовсе не позволяет достичь желаемых результатов. В связи

Артюшкин Андрей Борисович
кандидат технических наук, доцент, доцент кафедры телеметрических систем, комплексной обработки и защиты информации Военно-космической академии им. А.Ф. Можайского. Сфера научных интересов: системы цифровой связи, кодирование данных. Автор более 30 опубликованных научных работ.
E-mail: vka@mil.ru

Обрученков Виктор Петрович
кандидат технических наук, доцент, доцент кафедры телеметрических систем, комплексной обработки и защиты информации Военно-космической академии им. А.Ф. Можайского. Сфера научных интересов: космические радиотехнические комплексы. Автор более 30 опубликованных научных работ.
E-mail: vka@mil.ru

Бельских Марк Андреевич
курсант кафедры телеметрических систем, комплексной обработки и защиты информации Военно-космической академии им. А.Ф. Можайского. Сфера научных интересов: телеметрические системы. Автор 1 опубликованной научной работы.
E-mail: Belskih.999@gmail.com

с этим наряду с использованием все более совершенных электронных компонентов в составе устройств обработки информации активно ведется совершенствование алгоритмов преобразования и обработки данных.

В частности, для повышения скорости передачи данных при доступной операционной частоте могут использоваться методы экономного кодирования. Данные методы позволяют сокращать передаваемый объем данных исходя из соображения, что более часто встречающиеся в сообщении информационные единицы должны передаваться кодовыми последовательностями меньшей длины, нежели единицы с меньшей частотой или вероятностью появления. В результате такого кодирования длина выходной кодовой последовательности становится короче относительно исходной [3, 10].

Однако использование таких методов перекодирования данных связано с необходимостью применения сложных алгоритмов обработки информации и большого количества вычислительных операций. В частности, один из наиболее эффективных методов – метод Хаффмана – предусматривает большой объем сортировок элементов массивов данных в процессе построения кодового дерева. Алгоритм Хаффмана математически гарантированно создает наименьший по размеру код для каждого из символов исходных данных и относится к классу алгоритмов оптимального префиксного кодирования. Классический алгоритм Хаффмана является статическим двухпроходным алгоритмом (первый цикл – накопление массива данных, вычисление вероятности или частоты появления каждого символа в массиве и сортировка данных, второй – построение кодового дерева и получение сокращенных кодов для каждого символа) [2, 4, 9]. С ростом количества элементов, входящих в обрабатываемый массив данных, в зависимости от используемого алгоритма сортировки время решения задачи растет с той или иной степенью нелинейности и в кон-

це концов становится неприемлемо большим для удовлетворительного решения задачи информационного обеспечения.

Таким образом, существует противоречие между возможностью использования метода Хаффмана для сокращения объемов передаваемых данных и ростом сложности его реализации при увеличении их разнообразия.

То есть задачей разработчика кодирующего устройства, реализующего преобразование данных методом Хаффмана (как и с помощью любого другого подобного метода экономного кодирования), является его модернизация с целью снижения влияния алгоритмов сортировки на время решения задачи при заданном количестве сортируемых элементов.

Постановка задачи модификации алгоритма реализации кодирования данных методом Хаффмана с целью сокращения количества выполняемых машинных операций

Метод Хаффмана относится к статистическим методам сжатия данных без потерь [4].

Подавляющий объем операций при реализации данного метода приходится на алгоритмы сортировки данных, которые, исходя из уровня сложности, можно разделить на два вида.

Первый вид – предварительная сортировка элементов исходного алфавита $A\{a_j\}$ на начальном этапе реализации метода Хаффмана. Для удобства назовем ее «Сортировка 1». Она выполняется один раз, являясь при этом наиболее сложной и длительной по времени выполнения совокупностью операций, так как затрагивает все элементы множества $A\{a_j\}$. Ее задачей является расстановка элементов a_j в порядке возрастания их значений.

Вторая группа – сортировки, выполняемые в процессе построения кодового дерева и формирования таблицы соответствия. Задачей таких сортировок является определение позиции вновь полученного элемента a_{j_r} в текущем множестве $A_{M_r}\{a_{j_r}\}$. Это более быстрый вид сортировок, так как изначально все элементы множества, кроме вновь полученного, уже упорядочены, решается задача поиска первого элемента, не превышающего заданный. Однако количество таких сортировок велико и имеет значение $N - 1$, где N – мощности исходного множества $A\{a_j\}$. Данный вид сортировок назовем «Сортировка 2».

Сложность алгоритма характеризуется двумя количественными показателями – емкостной и временной сложностью. Первый говорит о том, какой объем памяти, а второй – сколько времени необходимо для его выполнения [1, 8].

Говоря о емкостной сложности, имеют в виду количество памяти, необходимой для размещения всех данных, участвующих в вычислительном процессе: входные и выходные данные, промежуточные результаты. В большинстве случаев современная электронная база предоставляет разработчикам возможности, позволяющие считать память некритическим ресурсом, поэтому чаще всего под анализом сложности алгоритма понимают исследование его временно́й сложности.

Ясно, что одна и та же программа при одинаковых входных данных на разных вычислительных средствах будет в общем случае выполняться в течение разного времени [1, 8].

Поэтому под сравнительным анализом сложности алгоритмов будем понимать исследование того, как соотносится изменение времени работы программ при увеличении объема входных данных на гипотетическом вычислительном средстве с постоянными вычислительными характеристиками.

Временну́ю сложность алгоритма выражают функцией $T(n)$ зависимости времени работы от размера входа. В задачах обработки одномерных массивов, коими являются задачи сортировки в нашем случае, под размером входа принято считать количество элементов в массиве.

Особенностью метода Хаффмана является то, что его временная сложность определяется не столько объемом входных данных n, сколько их разнообразием, то есть N количеством букв, составляющих алфавит входного массива, и необходимостью их сортировки. Поэтому в дальнейшем условимся использовать обозначение n, если речь идет о размерах входа алгоритма «в общем», и N, если говорится об алгоритме сортировки данных.

Таким образом, время выполнения алгоритма кодирования данных по методу Хаффмана можно представить как $T_{\text{алгор}}(n, N)$. Оно определяется суммой времени выполнения «Сортировки 1» – $T_{\text{Сорт1}}(N)$, времени, необходимого на $N-1$ сортировку вида «Сортировка 2» – $(N-1)$ – $T_{\text{Сорт2}}(N)$, времени, затрачиваемого на выполнение операций сложения при формировании узлов кодового дерева $(N-1)$ – $T_{\text{сумм}}(N)$ – и времени $T_{\text{кодир}}(n)$, затрачиваемого на представление исходного массива данных кодовыми последовательностями экономного кода:

$$T_{\text{алгор}}(n,N) = T_{\text{Сорт1}}(N) + (N-1)T_{\text{Сорт2}}(N) + (N-1)T_{\text{сумм}}(N) + T_{\text{кодир}}(n). \qquad (1)$$

При этом $T_{\text{кодир}}(n)$ зависит от объема начальных данных n, представленных в равномерном коде, и для фиксированного объема $n_{\text{фикс}}$ не зависит от методов, непосредственно используемых для решения задачи экономного кодирования. Соответственно, сократить $T_{\text{алгор}}(n, N)$ при $n_{\text{фикс}} = \text{const}$ возможно только за счет уменьшения величины остальных элементов выражения (1).

Так как мы определили, что сравнительный анализ алгоритмов выполняется на одном и том же оборудовании, время выполнения можно измерять в обычных единицах, кратных секунде. Однако в общем случае физическое время выполнения алгоритма – это величина $\tau \cdot f$, где f – число элементарных операций, реализованных в ходе выполнения алгоритма, а τ – среднее время выполнения одного элементарного действия. Число элементарных операций определяется структурой алгоритма, языком программирования и листингом программы и не зависит от схемной реализации ЭВМ, а среднее время выполнения – от скорости обработки сигналов компьютером. То есть алгоритм выполняется за конкретное количество действий – шагов. Поэтому объективной математической характеристикой временно́й сложности алгоритма является число элементарных действий, выполняемых в ходе работы алгоритма [1, 8].

Скорость выполнения алгоритма может существенно зависеть от содержания набора входных данных: время работы алгоритма в худшем и в лучшем случаях может сильно отличаться. Например, быстрая «в среднем» программа способна давать сбои в отдельных «плохих» случаях.

Оценка временной сложности алгоритма выполняется с использованием символа «O» (O большое). Формально $O(f(n))$ означает, что время работы алгоритма растет в зависимости от объема входных данных не быстрее, чем некоторая константа, умноженная на $f(n)$ – функцию, дающую верхнюю границу максимального числа основных операций, используемых алгоритмом при размере входных данных равном n.

Методы обработки данных

На рисунке 1 показаны графики роста O большого с увеличением количества информационных элементов на входе алгоритма в зависимости от вида функциональной зависимости O от *n*.

	S1	S2	S3	S4
O1		0,3		0,7
O2	0,2	0,5	0,3	
O3				1
O4	0,1		0,9	

	S1	S2	S3	S4
O1		0,3		0,7
O2	0,2	0,5	0,3	
O3	0,2			0,8
O4	0,1		0,9	

Рис. 1. Зависимость временной сложности алгоритмов от вида алгоритма сортировки данных

Известно достаточно большое количество методов сортировки. В таблице 1 приведены асимптотические оценки некоторых наиболее эффективных из них.

Таблица 1

Асимптотические оценки некоторых методов сортировки

Алгоритм	Временная сложность		
	Лучшее	В среднем	В худшем
Быстрая сортировка	$O(N\log(N))$	$O(N\log(N))$	$O(N^2)$
Сортировка слиянием	$O(N\log(N))$	$O(N\log(N))$	$O(N\log(N))$
Пирамидальная сортировка	$O(N\log(N))$	$O(N\log(N))$	$O(N\log(N))$
Блочная сортировка	$O(Nk)$	$O(Nk)$	$O(N^2)$
Поразрядная сортировка	$O(Nk)$	$O(Nk)$	$O(Nk)$

Из таблицы следует, что наилучшими временными характеристиками располагает поразрядная сортировка. То есть верхняя граница максимального числа основных операций, реализуемых алгоритмом при размере входных данных, равном *N*, для данной сортировки не должна превышать *Nk*, где *k* – константа, характеризующая конкретный алгоритм.

Зависимость количества операций от *N* для поразрядной сортировки описывается законом, графическое представление которого на рисунке 1 имеет вид прямого луча. Константа *k* на рисунке является коэффициентом, определяющим угол наклона графика относительно оси *N*. Ее значение для конкретного алгоритма определяется количеством и разнообразием используемых в нем видов элементарных операций, а также структурой алгоритма.

При реализации в одних и тех же условиях двух алгоритмов с разными значениями *k* (одинаковые массивы входных данных, одинаковая схемная реализация ЭВМ) алгоритм с меньшим *k* дает выигрыш в количестве необходимых для его реализации элементарных операций (см. рис. 1, *а*):

$$\begin{cases} k_1 < k_2, \\ O_1(Nk) < O_2(Nk), \\ n = \text{const}. \end{cases} \quad (2)$$

При этом с увеличением размера массива входных данных N выигрыш $\Delta O(Nk) = O_{ПР}(Nk) - O_1(Nk)$ линейно нарастает.

Для любого электронного вычислительного средства существует $O_{ПР}$ – максимальное количество элементарных операций, выполняемых данным средством в единицу времени.

На рисунке 1, *б* показано, что данному граничному значению для каждого выполняемого алгоритма соответствует свое предельное значение объема массива входных данных $N_{ПР}$, при этом с уменьшением характеризующего алгоритм значения k значение $N_{ПР}$ возрастает. Иными словами, при одном и том же $O_{ПР}$ предельное значение массива входных данных больше для алгоритма с меньшим значением k:

$$\begin{cases} k_1 < k_2, \\ N_{ПР1} > N_{ПР2}, \\ O = O_{ПР}. \end{cases} \quad (3)$$

Ясно, что если для решения какой-то задачи обработки данных необходимо использовать входной массив размером $N_{требуемое}$, который окажется больше $N_{ПР}$, используемого для этого алгоритма, получить полный выходной массив данных за заданный промежуток времени не удастся.

Кроме того, известно [6], что если в классическом алгоритме Хаффмана сортировать элементы после каждого суммирования или использовать приоритетную очередь, то алгоритм в целом будет работать за время $O(N k \log N)$. Вероятно, такую асимптотику можно улучшить до $O(Nk)$, используя усовершенствованные методы сортировки.

Таким образом, сформулируем задачи, решаемые в данной работе.

1. С целью повышения эффективности обработки данных в электронных вычислительных средствах, в частности для уменьшения времени выполнения сортировок потока данных при реализации кодирования методом Хаффмана, предложить алгоритм сортировки с характеристикой временной сложности вида $O(Nk)$ и коэффициентом k_A алгоритма, меньшим чем $k_{ПС}$ алгоритма поразрядной сортировки:

$$\begin{cases} k_A < k_{ПС}, \\ O < O_{ПС}. \end{cases} \quad (4)$$

2. Предложить механизм реализации метода Хаффмана, позволяющий снизить его временную сложность с вида $O(Nk \log N)$ до вида $O(Nk)$.

Способ реализации экономного кодирования методом Хаффмана с использованием модифицированных алгоритмов сортировки данных

Основным недостатком всех алгоритмов сортировки, включая поразрядную сортировку, является необходимость в выполнении большого количества операций сравнения элементов обрабатываемого числового массива, а также операций смещения подмножеств элементов между ячейками памяти при освобождении выделенной под отсортированный элемент позиции. Минимизировав количество таких операций, можно получить алгоритм, свойства которого описывает система условий (4).

Методы обработки данных

В отличие от существующих алгоритмов сортировки предлагаемый алгоритм использует операции сравнения лишь в некоторых возникающих при особых условиях ситуациях. Кроме того, в нем минимизировано количество операций сдвига.

Идея сортировки, реализуемая данным алгоритмом, заключается в однопроходной последовательной замене элементами исходного массива элементов переходного массива, находящихся на позициях, соответствующих численному значению очередного выбранного элемента исходного массива с дальнейшим уплотнением переходного массива посредством удаления оставшихся незамещенными элементов.

Реализация экономного кодирования методом Хаффмана включает в себя следующие шаги.

Шаг 1. Накопление информации в течение интервала наблюдения T_H с целью получения статистик элементов U_i, из которых формируется входной неупорядоченный массив данных $M\{m_r\}_{T_H}$ (рис. 4, *а*).

Шаг 2. Формирование алфавита $A\{a_i\}$ конечной мощности N, каждому элементу a_i которого соответствует единственное значение U_i.

Шаг 3. «Сортировка 1» – упорядочивание элементов алфавита в порядке возрастания.

Шаг 4. Построение кодового дерева и формирование таблицы соответствия исходного и экономного двоичного кода алфавита.

Шаг 5. Представление элементов массива $M\{m_r\}_{T_H}$ с помощью экономного кода.

Рассмотрим более подробно реализацию шага 3, предполагающую выполнение «Сортировки 1», – упорядочивание элементов полученного алфавита в порядке возрастания их значений. На данном этапе возможно значительное сокращение времени перекодирования за счет отказа от использования традиционных алгоритмов сортировки данных.

Вместо реализации того или иного порядка последовательного взаимного сравнения элементов сортируемого массива предлагается алгоритм упорядочивания данных посредством занесения информации об анализируемых элементах в ячейки памяти, адреса которых соответствуют их абсолютной величине (рис. 2, *а*). Назовем его модифицированным алгоритмом сортировки исходного массива.

При этом предполагается выполнение следующих этапов.

1. Среди элементов $A\{a_i\}$ определяются значения с максимальной и минимальной частотой появления на $T_{\text{Н}} - A_{\max}$ и $A_{\min}$, которые задают размер промежуточного массива:

$$|\text{N}| = \nu_{A_{\max}} - \nu_{A_{\min}}.$$

2. В памяти вычислительного устройства резервируется последовательно адресуемый ряд ячеек, при этом адреса первой и последней ячеек данного ряда имеют, соответственно, значения и $\nu_{A_{\min}} + \Delta№$ и $\nu_{A_{\max}} + \Delta№$, где $\Delta№$ – смещение относительно адресов ячеек, имеющих значения $\nu_{A_{\min}}$ и $\nu_{A_{\max}}$.

Последовательно от ν_1 до ν_N выполняется следующий проход по массиву $N\{\nu_i\}$. При этом, так как все значения ν_i лежат внутри интервала $[\nu_{A_{\min}}, \nu_{A_{\max}}]$, данные величины можно рассматривать как адреса ячеек выделенного участка памяти вычислительного устройства. То есть при наличии определенного числа в массиве $N\{\nu_i\}$ в соответствующую ячейку заносится «1». Если данное число встречается в массиве в нескольких позициях, то при

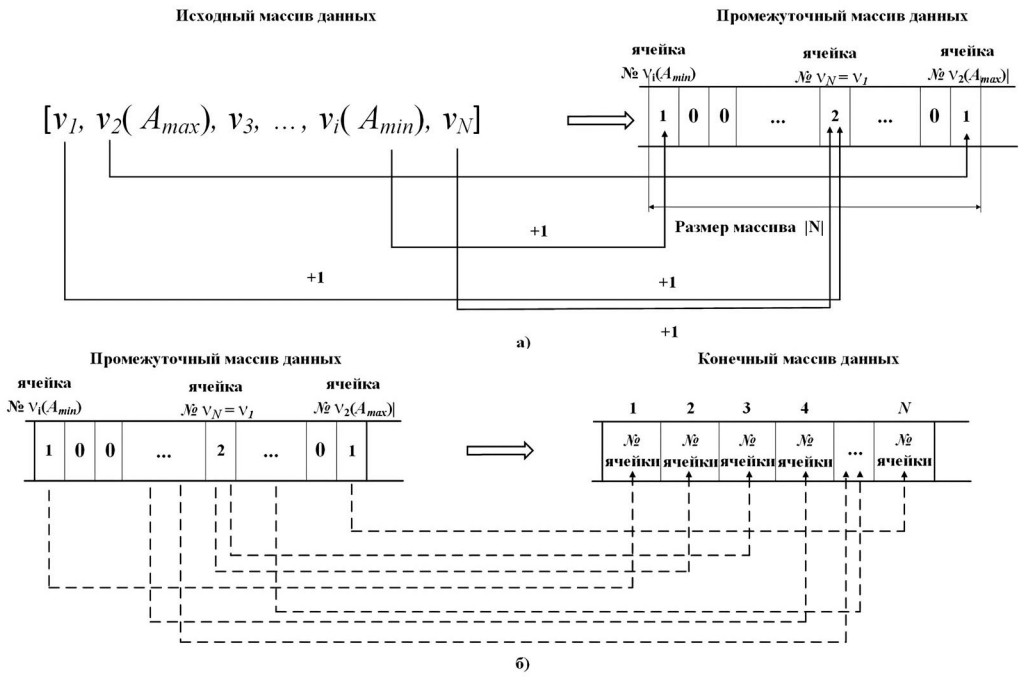

Рис. 2. Шаг 3: упорядочение исходного массива данных

прохождении каждой такой позиции значение величины, хранящейся по заданному адресу, увеличивается на единицу. Таким образом формируется промежуточный массив данных $N'\{v_j\}$, содержащий в определенных позициях информацию о количестве элементов массива $N\{v_i\}$ того или иного вида:

$$N\{v_i\} \to N'\{v_j\}.$$

При этом возможна ситуация, когда часть ячеек останется незаполненной в связи с отсутствием в $N\{v_i\}$ элементов с соответствующим значением.

3. На завершающем этапе выполняется проход по элементам промежуточного массива $N'\{v_j\}$. При этом формируется новый, конечный, массив данных $N_K\{v_i\}$:

$$N'\{v_j\} \to N_K\{v_i\}.$$

Формирование $N_K\{v_i\}$ происходит согласно следующему правилу (рис. 2, *б*).

Просмотр элементов массива $N'\{v_j\}$ – ячеек выделенной памяти – выполняется от значения ячейки с адресом $v_{A\min}$ и далее – в сторону ячейки с адресом $v_{A\max}$. При этом в качестве очередного элемента $N_K\{v_i\}$ принимается значение адреса ячейки памяти, к которой выполняется обращение (с учетом смещения Δ№), при условии, что значение числа, хранящегося там, не равно нулю. Такие адреса игнорируются. Если содержимое ячейки превосходит по величине единицу, значение ее адреса последовательно заносится в $N_K\{v_i\}$ количество раз, равное значению содержимого данной ячейки. Таким образом, после опроса всех ячеек памяти, соответствующих $N'\{v_j\}$, будет окончательно сформирован массив $N_K\{v_i\}$ мощностью N, идентичный по составу входящих в него элементов массиву

$N_K\{v_i\}$, но при этом структурно представляющий собою их неубывающую по величине последовательность.

Таким образом, для сортировки данных с помощью предложенного алгоритма достаточно выполнить три полных прохода по последовательности элементов сортируемого массива.

Шаг 4 включает построение кодового дерева и формирование таблицы соответствия исходного и экономного двоичного кода алфавита. При этом переход к нему возможен сразу после формирования промежуточного массива $N'\{v_j\}$, так как дальнейшая часть общего алгоритма предполагает работу с входными данными именно такого формата. То есть количество проходов массива при выполнении «Сортировки 1» возможно уменьшить до двух.

Шаг 4 предполагает наличие «Сортировок 2». При этом вместо бинарного поиска предлагается использовать модернизированный алгоритм, позволяющий уменьшить количество элементарных операций, приходящихся на данный шаг.

Как и в классическом методе Хаффмана, построение узлов кодового дерева начинается с суммирования частот первых двух элементов массива. Для этого в $N'\{v_j\}$ определяется значение числа, находящегося в первой позиции, – ячейке с адресом $v_i(A_{\min})$, так как данная ячейка всегда заполнена. Если данное значение равно единице, то ищется следующая ячейка – v_{i+k}, в которой содержится величина, отличная от нуля (рис. 3, *а*). На основе суммы значений адресов данных ячеек формируется новый адрес ячейки, содержимое которой необходимо увеличить на единицу:

$$v_j = v_i + v_{i+k}.$$

Это будет означать, что в новом массиве, который назовем трансформируемым массивом $N'\{v_j\}$, вновь сформированный элемент помещается в соответствующей его частоте v_i позиции. Верхняя граница данного массива ограничивается положением ячейки, адрес которой равен сумме значений всех входящих в него элементов Σ_N.

Случай, когда ячейка содержит число, значение которого превышает единицу, эквивалентен наличию соответствующего количества одинаковых членов ряда. При этом построение очередного узла дерева выполняется суммированием двух значений адреса данной ячейки с уменьшением хранящегося в ней числа на две единицы (рис. 3, *б*, *в*):

$$v_{g+l} = v_g + v_g.$$

Процесс построения узлов кодового дерева сопровождается построением двоичных кодовых последовательностей для участвующих в нем элементов исходного массива $N'_T\{v_j\}$с последующим формированием таблицы соответствия кодовых алфавитов.

На последнем этапе кодирования данных (шаг 5) выполняется представление элементов массива $M\{m_r\}_{T_H}$ с помощью полученных двоичных последовательностей экономного кода и формирование выходного информационного потока.

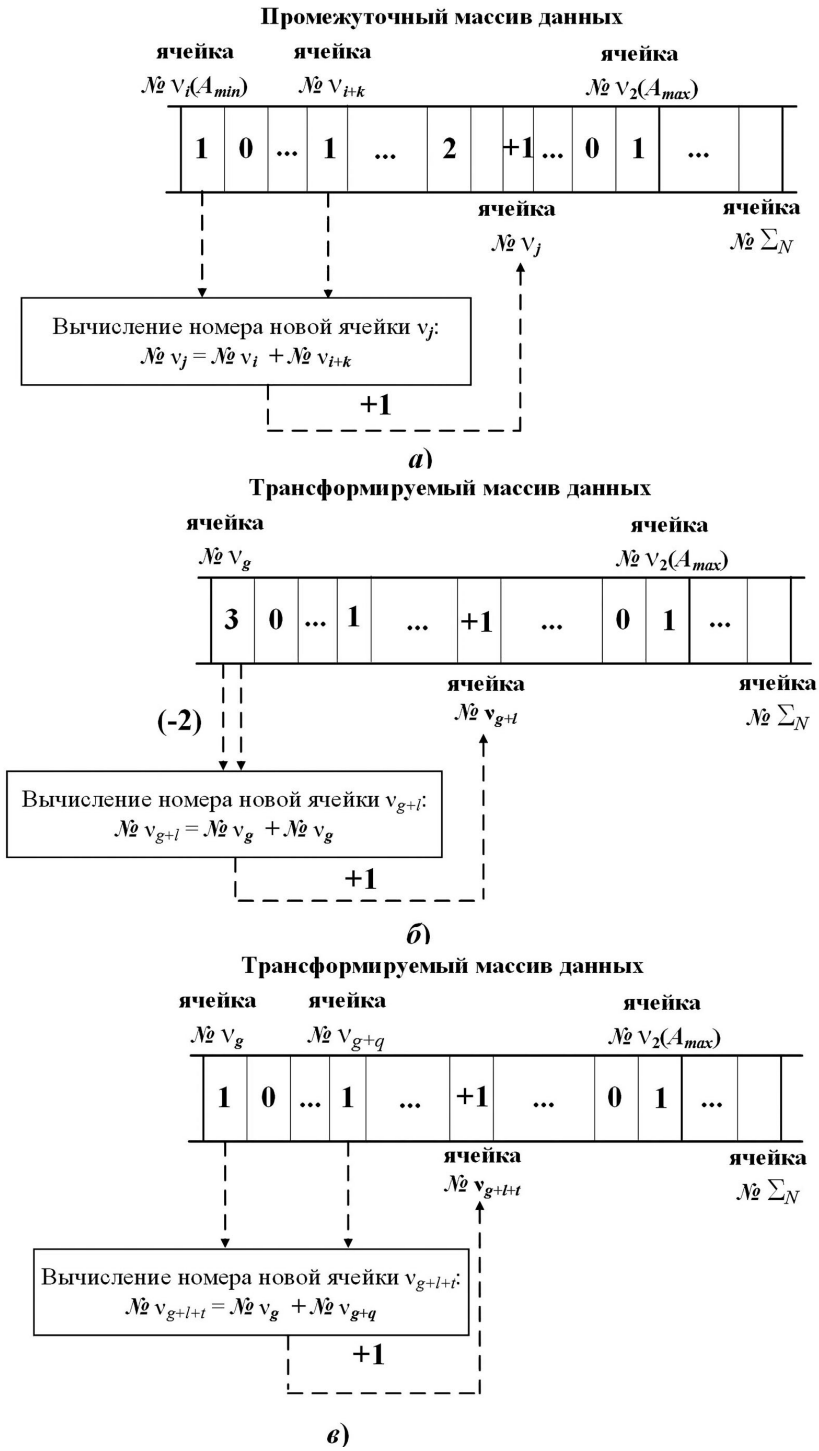

Рис. 3. Шаг 4: построение кодового дерева

Методы обработки данных

Оценка эффективности модифицированных алгоритмов сортировки данных

Реализация метода Хаффмана состоит из нескольких этапов, содержащих сортировки вида «Сортировка 1» и «Сортировка 2». При этом наиболее подходящим известным алгоритмом для реализации первого из них является поразрядная сортировка, обеспечивающая наибольшую эффективность при практической реализации метода, для второго – бинарный поиск.

Анализ временной сложности предложенного алгоритма реализации этапа «Сортировка 1» и алгоритма поразрядной сортировки показывает, что оба они выполняются за линейное время и имеют нотацию О большое вида $O(Nk)$. Понятно, что относительная сложность алгоритмов определяется соотношением соответствующих значений k.

В основе поразрядной сортировки лежит принцип разделения чисел упорядочиваемого ряда на массивы, содержащие числа с одинаковыми значениями в разрядах с совпадающими весами [5]. Обычно сортировку начинают с проверки крайних – младших или старших – разрядов и движутся затем, соответственно, в направлении возрастания или уменьшения их номеров. При этом на упорядочивание ряда по текущему разряду достаточно одного цикла-прохода по его элементам – они распределяются по группам, в которые входят числа с одинаковыми значениями в рассматриваемом в данном цикле разряде. Соответственно, для полной сортировки необходимое количество циклов совпадает с разрядностью сортируемых чисел p: ряд, состоящий из одноразрядных чисел, будет разделен на соответствующие подмножества за один проход, состоящий из двухразрядных – за два, и так далее. При этом значение $O(Nk)$ растет как с увеличением N, так и пропорционально росту количества разрядов p. То есть по сути в данном случае $p = k$.

Модифицированный алгоритм сортировки исходного массива выполняется за три прохода независимо от того, числа какой разрядности сортируются, а при реализации метода Хаффмана с использованием модернизированного алгоритма поиска, как было показано, необходимость в третьем цикле вообще отпадает. Уже при пороговом значении $k_{ПС} = 3$ предлагаемый алгоритм начинает выигрывать у алгоритма поразрядной сортировки в количестве необходимых элементарных операций, и с ростом p эта разница приобретает кратные значения (рис. 4 и табл. 2).

Таблица 2

Кратность значений с ростом *p*

p	1	2	3	8	16
$k_{ПС}/k_{А}$	1/3(2)	2/3(2)	3/3(2)	8/3(2)	16/3(2)

Данные о времени, затрачиваемом предлагаемым алгоритмом на решение задачи сортировки в зависимости от объема сортируемого массива, приведены в таблице 3 и на рисунке 5. Видно, что связь между количеством элементов в массиве и временем вычислений практически линейна (нелинейность графику придает необходимость сокращения длины горизонтальной оси), а его наклон имеет незначительный угол, что указывает на малую скорость нарастания времени обработки с ростом объема сортируемых данных.

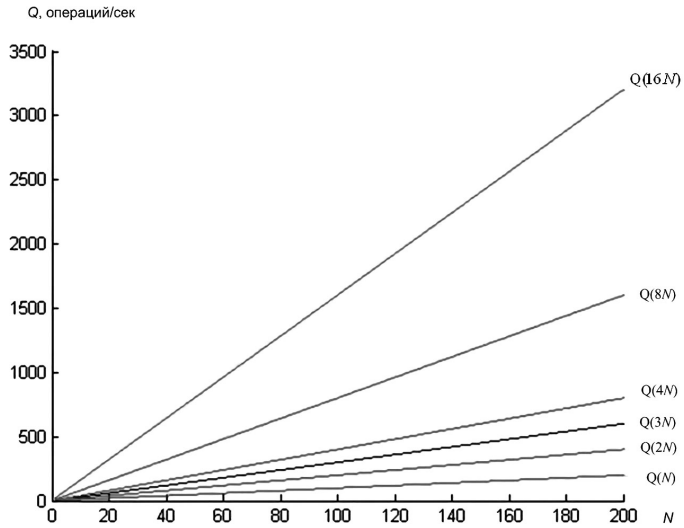

Рис. 4. Зависимость количества элементарных операций от *N* и *p*, необходимых для реализации алгоритмов сортировки

Таблица 3

Решение задачи сортировки в зависимости от объема материала

Кол-во элементов в массиве	500	1000	2000	4000	8000	16000	32000
Время вычисления, с	0,0195	0,51	0,57	0,61	0,75	1,09	2,06

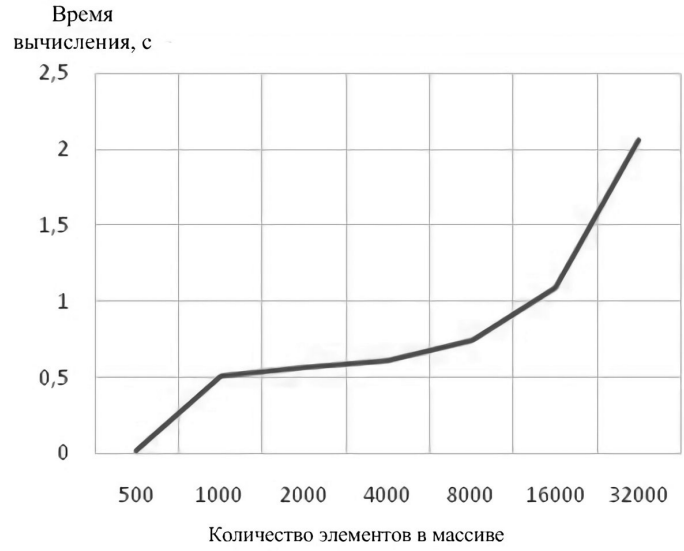

Рис. 5. Зависимость времени выполнения сортировки от объема массива

Для сравнения в таблице 4 приведены результаты моделирования для метода сортировки «пузырьками».

Методы обработки данных

Таблица 4

Метод сортировки «пузырьками»

Кол-во элементов в массиве	10	50	100	150	200	250	350	500	1000	2000
Время вычисления, с	0,0033	0,044	0,277	0,765	1,69	3,20	8,24	22,9	172,0	1348,2

В данном случае количество элементов массива менялось внутри небольшого диапазона – от 10 до 2000 чисел, что связано с экспоненциальным характером анализируемой зависимости: если при минимальном количестве элементов A время вычисления составляло тысячные доли секунды, то при 200 элементах на вычисление затрачивалось в среднем около полутора секунд, при 2000 элементах на упорядочивание массива требовалось более 22 минут.

Результаты работы алгоритма ускоренной сортировки отличаются от вышеприведенных кардинально: при объеме входных данных в 500 элементов они сравнимы с результатами выполнения сортировки «пузырьками» лишь для 50 входных значений, а с ростом размера входного массива относительный рост эффективности предлагаемого метода только увеличивается. Так, 2000 элементов с его помощью обрабатываются в среднем за 0,57–0,6 секунды (для метода «пузырьков» этому объему соответствует значение в 22 минуты), а на 32000 элементов необходимо всего 2 секунды.

Таким образом, эффективность алгоритма ускоренной сортировки можно было бы считать доказанной, однако разработанный алгоритм имеет некоторые особенности, ограничивающие его применение при определенных условиях.

В таблице 5 и на рисунке 6 приведены результаты эксперимента, в котором исследовалась зависимость $t_{\text{вычисл}}(A)$ с учетом значений абсолютной величины m наибольшего члена входного массива.

Таблица 5

Результаты эксперимента

Кол-во элементов в массиве		500	1000	2000	4000
Величина наибольшего элемента в массиве / время вычисления, с	6000	0,0282	0,037	0,062	0,123
	10000	0,053	0,065	0,089	0,145
	20000	0,174	0,182	0,215	0,295
	30000	0,425	0,43	0,475	0,531

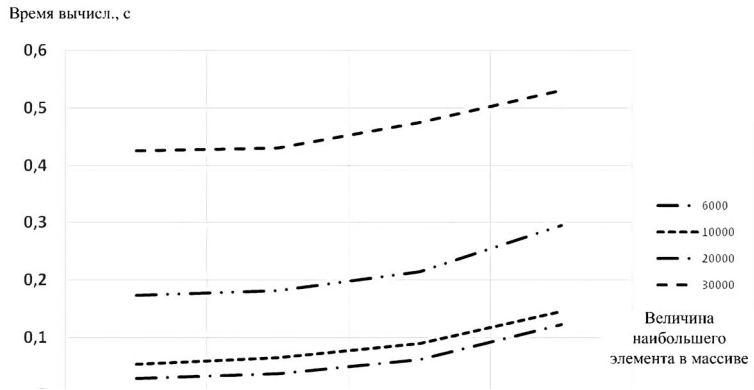

Рис. 6. Зависимость времени выполнения сортировки от объема массива и значения его наибольшего элемента

Очевидно, что с ростом m время выполнения сортировки растет, причем динамика процесса незначительно отличается от линейной. Это связано с тем, что с ростом m растет диапазон просматриваемой алгоритмом памяти, ограниченный значениями максимального и минимального элементов A, несмотря на независимость количества прохождений по массиву при сортировке от значений A и m.

В заключение отметим особенности реализации этапа «Сортировка 2». При использовании бинарного поиска, сложность которого имеет вид $O(\log N)$, часть алгоритма Хаффмана, реализующая построение кодового дерева, имеет сложность $O(N\log N)$. В то же время модернизированный алгоритм поиска позволяет решить задачу кодирования всего лишь за один проход, то есть имеет сложность $O(N)$.

Заключение

Таким образом, предложенный подход к решению задачи кодирования позволяет значительно снизить уровень противоречий между эффективностью метода Хаффмана по сокращению больших объемов данных и ростом сложности его реализации при увеличении их разнообразия, снижая влияние алгоритмов сортировки и поиска на время решения задачи при заданном количестве сортируемых элементов. При этом временная сложность модифицированного алгоритмом сортировки исходного массива составляет $O(3N)$, что при количестве разрядов в сортируемых числах более четырех позволяет считать его наиболее быстрым из известных алгоритмов сортировки. При совместном его использовании с модифицированным алгоритмом поиска в кодере Хаффмана его временная сложность составит $O(2N)$. При этом временную сложность алгоритма самого метода Хаффмана удается снизить с вида $O(Nk\log N)$ до вида $O(Nk)$.

Литература

1. *Абрамов С.А.* Лекции о сложности алгоритмов. М.: Изд-во Московского центра непрерывного математического образования, 2012. 246 с.
2. *Александров О.Е.* Компрессия данных или измерение и избыточность информации. Метод Хаффмана: Методические указания к лабораторной работе. Екатеринбург: Изд-во Ухтинского государственного технического ун-та, 2000. 52 с.
3. *Артюшкин А.Б., Куксенко М.А., Пантенков А.П.* Экономное кодирование как метод повышения скорости передачи информации в телеметрических системах // Вестник Российского нового университета. Серия «Сложные системы: модели, анализ, управление». 2020. Вып. 1. С. 43–54. DOI: 10.25586/RNU.V9187.20.01.P.043
4. *Горячкин О.В.* Теория информации и кодирования: учеб. пособие. Часть 2. Самара: Изд-во Поволжского государственного ун-та телекоммуникаций и информатики, 2017. 138 с.
5. *Кнут Д.Э.* Искусство программирования: пер. с англ. 2-е изд. Т. 3. Сортировки и поиск. М.: Вильямс, 2007. 832 с.
6. *Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К.* Алгоритмы. Построение и анализ / пер. с англ. И.В. Красикова, Н.А. Ореховой, В.Н. Романова; под ред. И.В. Красикова. 2-е изд. М.: Вильямс, 2005. 1296 с.
7. *Лоскутов А.И., Бянкин А.А., Козырев Г.И., Сакулин А.Н. и др.* Телеметрия: учебник / под общ. ред. А.И. Лоскутова. СПб.: Изд-во Военно-космической академии им. А.Ф. Можайского, 2017. 342 с.

8. *Разова Е.В.* Дополнительная подготовка школьников по дисциплине «Информационные технологии». Учебный модуль «Сложность алгоритмов». Владимир: Изд-во Владимирского государственного гуманитарного ун-та, 2011. 30 с.
9. *Семенюк В.В.* Экономное кодирование дискретной информации. СПб.: Изд-во Ун-та ИТМО, 2001. 115 с.
10. *Эльшафеи М.А., Сидякин И.М., Харитонов С.В., Ворнычев Д.С.* Исследование методов обратимого сжатия ТМИ // Вестник МГТУ им. Н.Э. Баумана. Серия «Приборостроение». 2014. № 3. С. 92–104.

References

1. Abramov S.A.(2012) *Lektsii o slozhnosti algoritmov* [Lectures on the Complexity of Algorithms]. Moscow, Moscow Centerfor Continuous Mathematical Education Publishing. 246 p. (in Russian).
2. Aleksandrov O.E. (2000) *Kompressiya dannykh ili izmerenie i izbytochnost' informatsii. Metod Khaffmana: Metodicheskie ukazaniya k laboratornoj rabote* [Data Compression or Measurement and Redundancy of Information. Huffman Method: Methodical Instructions for Laboratory Work]. Ekaterinburg, Ukhta State Technical University Publishing. 52 p. (in Russian).
3. Artyshkin A.B., Kyksenko M.A., Pantenkov A.P. (2020) *Ekonomnoe kodirovanie kak metod povysheniya skorosti peredachi informatsii v telemetricheskikh sistemakh* [Economical Coding as a Method of Increasing the Speed of Information Transmission in Telemetry Systems]. *Vestnik Rossijskogo novogo universiteta. Series "Complex Systems: Models, Analysis, Management"*, iss. 1, pp. 43–54 (in Russian). DOI: 10.25586/RNU.V9187.20.01.P.043
4. Goryachkin O.V. (2017) *Teoriya informatsii i kodirovaniya: ucheb. Posobie. Chast'2.* [Information and Coding Theory. Part 2]. Samara, Povolzhskiy State University of Telecommunications and Informatics Publishing. 138 p. (in Russian).
5. Knuth D.E. (1998) The Art of Computer Programming. 2nd ed. Vol. 3: Sorting and Searching. Boston, MA, Addison-Wesley. 796 p.
6. Cormen Th. H., Leiserson Ch.E., Rivest R.L., Stein C. (2001) Introduction to Algorithms. 2nd ed. Cambridge, MA, The MIT Press. 1184 p.
7. Loskutov A.I., Byankin A.A., Kozyrev G.I., Sakulin A.N. et al. (2017) *Telemetriya: uchebnik* [Telemetry]. Saint Petersburg, Mozhaisky Military Space Academy Publishing. 342 p. (in Russian).
8. Razova E.V. (2011) *Dopolnitel'naya podgotovka shkol'nikov po distsipline "Informatsionnye tekhnologii". Uchebnyj modul'"Slozhnost'algoritmov"* [Additional Training of Schoolchildren in the Discipline "Information Technology". Training Module "Complexity of Algorithms"]. Vladimir, Vladimir State University for the Humanities Publishing. 30 p. (in Russian).
9. Semenyuk V.V. (2001) *Ekonomnoe kodirovanie diskretnoj informatsii* [Economical Coding of Discrete Information]. Saint Petersburg, ITMO University Publishing. 115 p. (in Russian).
10. El'shafei M.A., Sidyakin I.M., Kharitonov S.V., Vornychev D.S. (2014) *Issledovanie metodov obratimogo szhatiya TMI* [Study of Methods for Lossless Compression of the Telemetry Information Stream]. *Bulletin of Bauman MSTU. Series "Instrument Making"*, no. 3, pp. 92–104 (in Russian).