УДК 510.52

АНАЛИТИЧЕСКАЯ ФУНКЦИЯ ТРУДОЕМКОСТИ В СРЕДНЕМ АЛГОРИТМА СОРТИРОВКИ ИНДЕКСАМИ НА ОСНОВЕ РАСПРЕДЕЛЕНИЯ РАЗМАХА ВАРЬИРОВАНИЯ

THE ANALYTICAL FUNCTION OF THE COMPLEXITY IN MEAN FOR THE INDEX-SORTING ALGORITHM BASED ON THE TOTAL VARIANCE DISTRIBUTION

[1,2]Головешкин В.А., д-р. техн. наук, проф.,

[2]Пономарёв А.В., кандидат физ.-мат. наук, доц.,

[1]Московский государственный университет приборостроения и информатики,

[2]Национальный исследовательский университет Высшая школа экономики

[2,3]Ульянов М.В., д-р. техн. наук, проф.

[3]Жукова Г.Н., кандидат физ.-мат. наук, доц.

[3]Московский государственный университет печати имени Ивана Федорова

Аннотация

В статье рассматривается алгоритм сортировки методом индексов в аспекте его трудоемкости в среднем. Для построения аналитической функции трудоемкости исследуется дискретное распределение вероятностей размаха варьирования в целочисленном массиве, имеющее самостоятельный теоретический интерес. Полученное распределение является базой для исследования трудоемкости алгоритма сортировки индексами, обладающего, в частных случаях, линейной сложностью по длине входа.

Abstract

The sorting algorithm based on the index method is being considered in the aspect of the complexity in mean. The discrete distribution of the total variance in the real number array is used to research the construction of the complexity analytical function. The distribution obtained is of independent theoretical interest as well as forms a basis for the analysis of the index-sorting algorithm complexity. In particular cases the complexity of the algorithm depends linearly on the input length.

1. Введение

Получение аналитической функции трудоемкости алгоритма в среднем — достаточно важный результат в анализе алгоритмов, на основе которого можно прогнозировать временные оценки программных реализаций [1]. Для получения функций трудоемкости в среднем широко используется аппарат теории вероятностей, в частности, метод моментов [2]. Однако достаточно часто при анализе алгоритмов класса *NPR* [3] для получения функции трудоёмкости или других ресурсных функций приходится, помимо основного аргумента — длины входа, вводить дополнительные аргументы (параметры), отражающие особенности задачи или данного алгоритма. В этом случае ресурсные функции уже являются функциями нескольких аргументов и получение классической трудоемкости в среднем, как функции длины входа требует усреднения по этим дополнительным параметрам. Рассматриваемая в статье задача определения трудоемкости в среднем для алгоритма сортировки методом индексов приводит к необходимости применения именно такого подхода к анализу.

2. Описание модели вычислений и базовые операции механизма реализации

В целях удобства анализа трудоемкости компьютерных алгоритмов, запись которых приближена к языку процедурного программирования высокого уровня, необходимо ввести соответствующую модель вычислений. Эта модель вводится как некоторое расширение классической модели вычислений, — машины с произвольным доступом к памяти, в части набора базовых операций, и фактически является новым алгоритмическим базисом для объектного базиса этой классической модели [3].

В дальнейшем будем считать, что объектами хранения в информационном носителе модели вычислений являются битовые слова фиксированной длины β, обращение к которым производится по их адресу в символической записи. Механизм реализации, аналогичный классическому последовательному процессору с фон-неймановской архитектурой, предполагает расположение программы и данных в информационном носителе. При этом считается, что базовыми (элементарными) операциями такого механизма реализации являются операции, коррелированные с основными операторами процедурного языка высокого уровня. Для используемой модели вычислений базовыми операциями будем считать следующие [4]:

— простое присваивание: $a \leftarrow b$;

— одномерная индексация: $A[i]$: (адрес $(A)+i*$ длина слова);

— арифметические операции: $\{ *, /, -, + \}$;

— операции сравнения: $a \{ <, >, =, \leq, \geq \} b$;

— логические операции: $(l_1)\{ or, and, not \}(l_2)$;

Алгоритмические конструкции сводимы к введенным базовым операциям, в частности цикл по счетчику требует три операции на один проход [3]. Трудоемкость алгоритма определяется при этом как число базовых операций, заданных алгоритмом на конкретном входе.

3. Алгоритм сортировки массива методом индексов

Этот оригинальный алгоритм [3] работает не для всех возможных массивов. Его применение изначально предполагает, что исходные числа являются целыми и положительными. Основная идея заключается в использовании дополнительного массива, количество элементов которого равно максимальному значению в исходном массиве. Далее значение каждого элемента исходного массива используется в качестве индекса элемента дополнительного массива. Этот элемент служит указателем на то, что в исходном массиве существует такое значение. Таким образом, если некоторый элемент исходного массива равен i, то i-ый элемент дополнительного массива устанавливается в единицу. После прохода по исходному массиву индексы ненулевых элементов дополнительного массива образуют исходный массив, отсортированный по возрастанию. Более детальное рассмотрение этой идеи позволяет уменьшить затраты дополнительной памяти. На самом деле, если минимальный элемент исходного массива равен 1000, то,

очевидно, что первые 999 элементов дополнительного массива не будут использоваться. Учитывая это, размер дополнительного массива можно уменьшить до величины размаха варьирования сортируемых чисел. Если при этом выполнить сдвиг элементов исходного массива на $\min - 1$, то реализуется возможность сортировки отрицательных чисел. Необходимость обработки повторяющихся элементов приводит к использованию значения элемента дополнительного массива как счетчика повторяющихся элементов. Предлагаемый ниже алгоритм работает с любыми допустимыми, в том числе и отрицательными, целыми числами, заданными словами битовой длины β, допуская повторение значений в элементах исходного массива.

Реализация такого алгоритма включает в себя следующие шаги:

— определение максимального и минимального значений в исходном массиве A с использованием быстрого алгоритма [3, с. 219] и вычисление размаха варьирования L и сдвига d:

$$L = \max_{i=1,n} A[i] - \min_{i=1,n} A[i] + 1, \quad d = \min_{i=1,n} A[i] - 1.$$

— создание и обнуление дополнительного массива необходимой длины;

— заполнение элементов дополнительного массива по методу индексов с учетом сдвига;

— получение отсортированного массива путем обратной сборки значений индексов ненулевых элементов дополнительного массива с учетом сдвига и кратности.

Запись этого алгоритма в принятом алгоритмическом базисе имеет вид (в столбце справа указано число заданных базовых операций):

```
Sort_Ind(A, n)
// Шаг 1: Поиск Max и Min в массиве A[1..n] - быстрый алгоритм
   d ← Min-1                                              2
   L ← Max-Min+1                                          3
// Шаг 2: Создание и обнуление дополнительного массива B[1..L]
   For j ← 1 to L                                      1+3L
      B[j] ← 0                                           2L
   end For j
// Шаг 3: Заполнение методом индексов
   For i ← 1 to n                                      1+3n
      k ← A[i]-d                                         3n
      B[k] ← B[k]+1                                      4n
   end For i
// Шаг 4: Обратная сборка
   i ← 1                                                  1
   For j ← 1 to L                                      1+3L
      If B[j]≠0                                          2L
         then
         // Обработка повторяющихся элементов исходного массива
            jd ← j+d                                     2x
            k ← B[j]                                     2x
            For m ← 1 to k                            1+3k
               A[i] ← jd                                 2k
               i ← i+1                                   2k
            end For m
   end For j
Return (A)
End.
```

4. Параметрический анализ трудоемкости алгоритма

Заметим, что трудоемкость этого алгоритма существенно зависит от размаха варьирования L, который и будет использован далее как дополнительный аргумент (параметр) функции трудоемкости $f_A(n)$, где n — длина входа. По сути, мы рассматриваем поведение трудоемкости алгоритма $f_A(D)$, где D — вход алгоритма (массив длины n), на $D \in D_n$ — множестве всех входов длины n, обладающих размахом варьирования L. С учетом введенной параметризации $f_A(n)$ может быть представлена на основе информации о числе базовых операций, указанных в строках записи алгоритма, в виде

$$f_A(n,L) = f_{A\max}(n) + f_1(n,L) + f_2(x,k).$$

Трудоемкость в среднем для первого шага алгоритма — этапа поиска минимума и максимума, известна [3]:

$$f_{A\max}(n) = 5{,}5n + 5\ln(n) + 2{,}5.$$

Компонент $f_1(n,L)$ описывает трудоемкость шагов 2, 3 и 4, не связанную с числом совпадений элементов, и зависящую только от длины и размаха варьирования исходного массива. Это значение легко определяется по тексту алгоритма

$$f_1(n,L) = 1 + 5L + 1 + 10n + 1 + 1 + 5L = 10L + 10n + 4.$$

Компонент $f_2(x,k)$ внутри шага 4 зависит от числа совпадений элементов исходного массива. Лучший по трудоемкости случай — это ситуация, когда все элементы одинаковы $(x = 1,\ k = n)$, тогда

$$f_2^{\vee}(n) = 2 + 2 + 1 + 7n = 7n + 5,$$

а в худшем случае все элементы различны $(x = n,\ k = 1)$ и

$$f_2^{\wedge}(n) = 2n + 2n + 1n + 7n = 12n.$$

В случае если есть только парные совпадения, трудоемкость составляет $9{,}5n$ базовых операций.

Очевидно, среднее значение для $f_2(x,k)$ определяется распределением совпадений в исходном массиве, которое связано со способом его формирования. В предположении, что максимальное значение элемента массива велико (например, при $\beta = 32$ бита, значение $\max A[i] = 2^{32} - 1$), а длина массива n существенно меньше максимального значения ($n << \max A[i]$), вероятность совпадений при равновероятной выборке мала. Например, при $n = 1000,\ \beta = 32$ вероятность того, что массив не содержит совпадающих элементов, равна 0,999884. Поэтому в дальнейшем будем рассматривать $f_2(x,k) = f_2^{\wedge}(n) = 12n$. Объединяя полученные результаты, окончательно получаем:

$$f_A(n,L) = 5{,}5n + 5\ln(n) + 2{,}5 + 10L + 10n + 4 + 12n = 10L + 27{,}5n + 5\ln(n) + 6{,}5. \qquad (1)$$

Тем самым мы получили трудоемкость в терминах метода классов входных данных [5] — разбиение D_n на подмножества входов, в каждом из которых алгоритм задает одинаковую трудоемкость, поскольку при фиксированном значении L, в соответствии с (1), $f_A(n,L)$ постоянна.

Отметим, что при большом размахе варьирования объем дополнительной памяти, требуемой данным алгоритмом, может быть значительным (при $\beta = 32$ бита, $\max L = 2^{32}$). Интересно отметить также, что при значениях $L \approx c \cdot n$ данный алгоритм, за счет дополнительных затрат памяти, сортирует массив за линейное время.

Для данного алгоритма ресурсная сложность имеет вид [3]

$$\overline{\Re}_c(A) = \langle \Theta(n+L), \Theta(n+L) \rangle,$$

где первый компонент отражает оценку трудоемкости, а второй — суммарные затраты памяти, определяемые объемом исходного и дополнительного массивов. Дополнительные затраты памяти алгоритма составляют $\Theta(L)$ и при больших β могут быть значительны.

4. Постановка задачи усреднения по размаху варьирования

Содержательно представляет интерес получение аналитического вида функции трудоемкости данного алгоритма в среднем только как функции длины входа. Мы хотим усреднить трудоемкость (1) по размаху варьирования L (при фиксированной длине входа n). Будем рассматривать L как случайную величину на вероятностном пространстве входов алгоритма. Обозначим через $E(\cdot)$ математическое ожидание, тогда

$$\overline{f}_A(n) = E(f_A(n,L)) = 27{,}5n + 5\ln(n) + 6{,}5 + 10E(L). \qquad (2)$$

Для получения $E(L)$ построим вероятностную модель, — вероятностное пространство Ω_L и распределение вероятностей для случайной величины L, а затем вычислим $E(L)$.

Ограничивая битовую длину слова в информационном носителе модели вычислений β битами, мы получаем возможные значения чисел от 0 до максимального значения $m = 2^{\beta} - 1$, и, следовательно, диапазон возможных значений случайной величины L от 1 до $m+1$, тем самым $\Omega_L = \{1,...,m+1\}$. Это, очевидно, приводит к зависимости L от m.

Пусть значения $a_i, i = \overline{1,n}$ выбираются в массив A равномерно случайно из множества $Z_m = \{z \mid z \in \overline{0,m}\}$, а случайная величина L определяется как размах варьирования в массиве A:

$$L = \max(a_i) - \min(a_i) + 1,\ i = \overline{1,n}.$$

Отметим, что тем самым величина L зависит также и от длины входа n, т.е. $L = L(n,m)$.

В этих предположениях и обозначениях необходимо:

1. Построить распределение $g(n,m,k)$ случайной величины $L(n,m)$

$$\forall k \in \Omega_L \quad g(n,m,k) = P(L(n,m) = k),\ \ n = const, m = const.$$

2. В соответствии с этим распределением вычислить математическое ожидание

$$E\big(L(n,m)\big)=\sum_{k=1}^{m+1}k\cdot P\big(L(n,m)=k\big). \tag{3}$$

5. Распределение размаха варьирования

Принимая гипотезу о равномерной случайной выборке n элементов из множества Z в исходный массив A, подлежащий сортировке, мы тем самым принимаем гипотезу о том, что все возможные массивы длины n на входе алгоритма равновероятны. Таким образом, исходная вероятностная модель имеет вид:

$$M=\langle\Omega,\,P(\cdot)\rangle,\ \Omega=\{D\mid D\in D_n\},\ P(D)=\frac{1}{|\Omega|}.$$

Мощность вероятностного пространства Ω, очевидно, определяется числом способов заполнения n элементов массива A числами из множества Z, причем $|Z|=m+1$. Поскольку выборка из Z в A производится с возвратом (повторения значений допустимы), то:

$$|\Omega|=(m+1)^n.$$

Обозначим через $N_k(n,m)$ число массивов, имеющих размах варьирования k, тогда распределение вероятностей размаха варьирования принимает вид

$$g(n,m,k)=P\big(L(n,m)=k\big)=\frac{N_k(n,m)}{(m+1)^n}. \tag{4}$$

Рассмотрим $N_k(n,m)$ при различных значениях k.

При $k=1$ весь массив состоит, очевидно, из одинаковых чисел. Поскольку множество Z содержит всего $m+1$ различных элементов, то

$$N_1(n,m)=(m+1).$$

При $k=2$ исходный массив формируется только из двух соседних чисел $z_i, z_{i+1}=z_i+1,\ i=\overline{0,m-1}$, причем всего существует m соседних пар. Поскольку длина массива равна n, и выборка производится с возвратом, то мы имеем всего 2^n вариантов, однако существует два варианта заполнения, не удовлетворяющие условию $k=2$ — это вариант заполнения только числом z_i и только числом z_{i+1}, которые и необходимо исключить, таким образом

$$N_2(n,m)=\big(2^n-2\big)\cdot m.$$

В общем случае, при $k\ge 2$ массив формируется из k соседних чисел в Z_m, причем в выборке обязательно должны присутствовать как наименьшее из них, обозначим его через $z_{\min}$, так и наибольшее $z_{\max}=z_{\min}+k-1$.

Всего в Z_m существует $(m-k+2)$ неупорядоченных наборов из k подряд идущих положительных целых чисел, не превосходящих m, из которых можно сформировать k^n различных массивов. Из этого множества наборов необходимо исключить массивы, размах варьирования которых будет меньше k. Если в массиве нет наименьшего ($z_{\min}$) из наших k подряд идущих чисел, то размах варьирования в таком массиве будет гарантированно меньше k, а число таких массивов — $(k-1)^n$. Столько же существует массивов с размахом варьирования меньше k, не содержащих $z_{\max}$ (наибольшее из наших k подряд идущих чисел).

Массивов, не содержащих как $z_{\min}$, так и $z_{\max}$ ровно $(k-2)^n$, так что для подсчета числа массивов с размахом варьирования k достаточно вычесть из общего количества различных массивов число массивов без $z_{\min}$, число массивов без $z_{\max}$ и прибавить число массивов, не содержащие как $z_{\min}$, так и $z_{\max}$ (поскольку мы их вычитаем дважды). Эти рассуждения приводят к общей формуле, справедливой при $k \geq 2$:

$$N_k(n,m) = \left(k^n - 2(k-1)^n + (k-2)^n\right) \cdot (m-k+2),\ k \geq 2. \qquad (5)$$

Числовой пример, иллюстрирующий распределение, заданное формулой (5) при $n=5$ и $m=15$ приведен в таблице 1. Отметим, что уже при этих небольших значениях распределение вероятностей имеет ярко выраженную правую асимметрию.

Табл. 1. Распределение для случайной величины L при $n=5$ и $m=15$.

k	$N_k(n,m)$	$P(L(n,m)=k)$
1	16	0,00002
2	450	0,00043
3	2 520	0,00240
4	7 410	0,00707
5	15 840	0,01511
6	28 050	0,02675
7	43 800	0,04177
8	62 370	0,05948
9	82 560	0,07874
10	102 690	0,09793
11	120 600	0,11501
12	133 650	0,12746
13	138 720	0,13229
14	132 210	0,12609
15	110 040	0,10494
16	67 650	0,06452
Σ	1 048 576	1

Соответствующий график распределения случайной величины $L(5,15)$ приведен на рисунке 1.

Рис. 1. График распределения случайной величины $L(5,15)$.

Остается показать, что формула (5) действительно задает вероятностную меру $P(\Omega)=1$. Мы докажем эквивалентное утверждение:

$$\sum_{k=1}^{m+1} N_k(n,m) = (m+1)^n . \qquad (6)$$

Поскольку $N_1(n,m)=(m+1)$, а остальные значения $N_k(n,m)$ задаются формулой (5), то формула (6) преобразуется к виду

$$\sum_{k=1}^{m+1} N_k(n,m) = (m+1) + \sum_{k=2}^{m+1} \left(k^n - 2(k-1)^n + (k-2)^n\right)\cdot(m-k+2) =$$

$$= (m+1) + \sum_{k=2}^{m+1} \left(k^n - 2(k-1)^n + (k-2)^n\right)\cdot(m-k+2) =$$

$$= (m+1) + \sum_{k=2}^{m+1} \left(k^n - 2(k-1)^n + (k-2)^n\right)\cdot(m+2) - \sum_{k=2}^{m+1} k^{n+1} + 2\sum_{k=2}^{m+1} k(k-1)^n - \sum_{k=2}^{m+1} k(k-2)^n =$$

$$= (m+1) + (m+2)\cdot\left(\sum_{k=0}^{m+1} k^n - 1 - 2\sum_{k=0}^{m} k^n + \sum_{k=0}^{m-1} k^n\right) - \sum_{k=0}^{m+1} k^{n+1} + 1 + 2\sum_{k=0}^{m} k^n(k+1) - \sum_{k=0}^{m-1} k^n(k+2) =$$

$$= (m+1) - (m+2) + 1 + (m+2)\cdot\left((m+1)^n - m^n\right) - \sum_{k=0}^{m+1} k^{n+1} + 2\sum_{k=0}^{m} k^{n+1} + 2\sum_{k=0}^{m} k^n - \sum_{k=0}^{m-1} k^{n+1} - 2\sum_{k=0}^{m-1} k^n =$$

$$= (m+2)\cdot\left((m+1)^n - m^n\right) - \left(\sum_{k=0}^{m+1} k^{n+1} - 2\sum_{k=0}^{m} k^{n+1} + \sum_{k=0}^{m-1} k^{n+1}\right) + 2\left(\sum_{k=0}^{m} k^n - \sum_{k=0}^{m-1} k^n\right) =$$

$$= (m+2)\cdot(m+1)^n - (m+2)\cdot m^n - \left((m+1)^{n+1} - m^{n+1}\right) + 2m^n =$$

$$= (m+1)^n(m+2-m-1) - m^n(m+2-m-2) = (m+1)^n,$$

что собственно и доказывает утверждение (6).

6. Математическое ожидание размаха варьирования и трудоемкость в среднем алгоритма сортировки индексами

Для завершения анализа алгоритма осталось вычислить математическое ожидание случайной величины $L(n,m)$. В соответствии с (3), (4) и (5) имеем

$$E(L(n,m)) = \sum_{k=1}^{m+1} k \cdot P(L(n,m)=k) = \sum_{k=1}^{m+1} k \cdot \frac{N_k(n,m)}{(m+1)^n}.$$

Для упрощения выкладок мы опустим множитель $(m+1)^{-n}$, который учтем позже, тем самым задача сводится к вычислению суммы

$$S(n,m) = E(L(n,m)) \cdot (m+1)^n = 1 \cdot (m+1) + \sum_{k=2}^{m+1} k \cdot \left(k^n - 2(k-1)^n + (k-2)^n\right) \cdot (m-k+2). \qquad (7)$$

На основе алгебраических преобразований формулы (7) получаем

$$S(n,m) = (m+1) + \sum_{k=2}^{m+1} k \cdot (m-k+2) \cdot \left(k^n - 2(k-1)^n + (k-2)^n\right) =$$

$$= (m+1) + \sum_{k=2}^{m+1} (m+2) \cdot \left(k^{n+1} - 2k(k-1)^n + k(k-2)^n\right) - \sum_{k=2}^{m+1} \left(k^{n+2} - 2k^2(k-1)^n + k^2(k-2)^n\right) =$$

$$= (m+1) + (m+2) \cdot \left(\sum_{k=0}^{m+1} k^{n+1} - 1 - 2\sum_{k=0}^{m} (k+1)k^n + \sum_{k=0}^{m-1} (k+2)k^n \right) -$$

$$- \sum_{k=0}^{m+1} k^{n+2} + 1 + 2\sum_{k=0}^{m} (k+1)^2 k^n - \sum_{k=0}^{m-1} (k+2)^2 k^n =$$

$$= (m+1) - (m+2) + 1 + (m+2) \cdot \left(\sum_{k=0}^{m+1} k^{n+1} - 2\sum_{k=0}^{m} k^{n+1} + \sum_{k=0}^{m-1} k^{n+1} - 2\sum_{k=0}^{m} k^n + 2\sum_{k=0}^{m-1} k^n \right) -$$

$$- \sum_{k=0}^{m+1} k^{n+2} + 2\sum_{k=0}^{m} k^{n+2} - \sum_{k=0}^{m-1} k^{n+2} + 4\sum_{k=0}^{m} k^{n+1} - 4\sum_{k=0}^{m-1} k^{n+1} + 2\sum_{k=0}^{m} k^n - 4\sum_{k=0}^{m-1} k^n =$$

$$= (m+2) \cdot \left((m+1)^{(n+1)} + m^{(n+1)} - 2m^{(n+1)} - 2m^n\right) - \left((m+1)^{(n+2)} + m^{(n+2)} - 2m^{(n+2)}\right) +$$

$$+ 4m^{(n+1)} - 2\sum_{k=0}^{m-1} k^n + 2m^n =$$

$$= (m+1)^{(n+1)} + m^{(n+1)} \cdot (-m-2+m+4) + m^n \cdot (-2m-4+2) - 2\sum_{k=0}^{m-1} k^n =$$

$$= (m+1)^{(n+1)} + 2m^{(n+1)} + m^n \cdot (-2m-2) - 2\sum_{k=0}^{m-1} k^n =$$

$$= (m+1)^{(n+1)} + m^n \cdot (2m-2m-2) - 2\sum_{k=0}^{m-1} k^n =$$

$$= (m+1)^{(n+1)} - 2m^n - 2\sum_{k=0}^{m-1} k^n = (m+1)^{(n+1)} - 2\sum_{k=0}^{m} k^n.$$

Учитывая множитель $(m+1)^{-n}$, окончательно получаем:

$$E(L(n,m)) = \frac{S(n,m)}{(m+1)^n} = (m+1) - \frac{2}{(m+1)^n} \sum_{k=0}^{m} k^n \ . \tag{8}$$

Рассмотрим отношение математического ожидания размаха варьирования к максимальному возможному значению размаха, равному $(m+1)$:

$$\frac{E(L(n,m))}{m+1} = 1 - \frac{2}{(m+1)^{n+1}} \sum_{k=0}^{m} k^n .$$

Обозначим через $\Delta(n,m)$ отклонение от единицы этого отношения, тогда

$$\Delta(n,m) = 1 - \frac{E(L(n,m))}{m+1} = \frac{2}{(m+1)^{n+1}} \sum_{k=0}^{m} k^n . \tag{9}$$

приближенно оценивая сумму в (9) соответствующим интегралом, получим

$$\Delta(n,m) = \frac{2}{(m+1)^{n+1}} \sum_{k=0}^{m} k^n \le \frac{2}{(m+1)^{n+1}} \int_0^{m+1} x^n dx = \frac{2(m+1)^{n+1}}{(n+1)(m+1)^{n+1}} = \frac{2}{n+1} .$$

При фиксированном m последовательность положительных чисел $\Delta(n,m)$ ограничена сверху сходящейся к нулю последовательностью, поэтому

$$\lim_{n\to\infty} \Delta(n,m) = 0 .$$

Итак, для математического ожидания размаха варьирования выполняется неравенство:

$$1 - \frac{E(L(n,m))}{m+1} \le \frac{2}{n+1} ,$$

поэтому

$$(m+1)\left(1 - \frac{2}{n+1}\right) \le E(L(n,m)) \le m+1 . \tag{10}$$

На основании формулы (2) с учетом (10) получаем окончательный результат для практических вычислений:

$$\overline{f}_A(n,m) = 27{,}5n + 5\ln(n) + 6{,}5 + 10(m+1) = \Theta(n+m) . \tag{11}$$

Заметим, что в итоге трудоемкость в среднем зависит не только от длины входа n, но и от m — максимального значения числа в информационном носителе модели вычислений.

7. Заключение

В статье получена аналитическая функция трудоемкости алгоритма сортировки индексами в среднем для модели вычислений с информационным носителем, элементы которого ограничены значением $m = 2^{\beta} - 1$, т.е. имеют фиксированную битовую длину β. Математическое ожидание размаха варьирования, к сожалению, оказалось близким к максимальному возможному значению размаха, так что трудоемкость алгоритма в среднем существенно определяется

максимальным значением числа в информационном носителе модели вычислений. Ярко выраженная правая асимметрия распределения при условиях равномерной случайной выборки чисел позволяет говорить о том, что вероятность входов, приводящих к линейной трудоемкости, пренебрежимо мала.

В связи с тем, что математическое ожидание размаха варьирования оказалось близко к наибольшему возможному значению размаха $m+1$, по-видимому, имеет смысл использовать статический дополнительный массив длины $m+1$ для любых входных массивов, размах варьирования значений в которых ограничен значением $m+1$. Исследование показывает, что индексный метод сортировки имеет смысл применять при относительно небольших значениях элементов сортируемого массива. Дело в том, что при больших m получается огромный дополнительный массив, сложность работы с которым сводит на нет достоинства метода. Решение задачи рационального выбора между сортировкой индексами и вставками приведено в [3].

Авторы надеются, что самостоятельный интерес имеет полученное в статье распределение, а также формула математического ожидания размаха варьирования значений в массиве из n элементов, формируемого равномерно случайно из целых неотрицательных чисел, ограниченных значением m.

Полученные результаты могут быть использованы для прогнозирования временных оценок программных реализаций алгоритма сортировки методом индексов при известных ограничениях на значения чисел, составляющих исходный массив.

Библиографический список

1. Ульянов М.В. Метод прогнозирования временных оценок программных реализаций алгоритмов на основе функции трудоемкости // Информационные технологии. 2004. № 5. С. 54–62.

2. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: построение и анализ. – М.: МЦНМО, 1999. – 960 с., 263 ил.

3. Ульянов М.В. Ресурсно-эффективные компьютерные алгоритмы. Разработка и анализ. — М.: ФИЗМАТЛИТ, 2008. — 304 с.

4. Ульянов М.В. Теоретико-множественный подход к определению функций трудоемкости алгоритма // Информация и безопасность. 2004. № 2. С. 53–58.

5. Петрушин В.Н, Ульянов М.В. Информационная чувствительность компьютерных алгоритмов. — М.: ФИЗМАТЛИТ, 2010. — 224 с.