Тема 5. Сортировка

5.6. Сортировка подсчетом

Данная сортировка известна также как *сортировка перечислением*. Метод основан на том, что j-е имя в окончательно упорядоченной таблице превышает точно $j - 1$ остальных имен. Тогда если известно, что некоторое имя больше $j - 1$ других имен, то в отсортированной таблице оно займет j-ю позицию. Таким образом, идея сортировки заключается в попарном сравнении всех имен и подсчете, сколько из них меньше каждого отдельного имени. Очевидно, что нет необходимости сравнивать имя само с собой и после сравнения x_i с x_j не нужно сравнивать x_j с x_i.

Для реализации сортировки подсчетом необходимо каждому имени x_i исходной таблицы сопоставить элемент (счетчик) c_i, т. е. всего требуется n таких элементов. Если $x_i < x_j$, то увеличивается на единицу значение элемента c_j, в противном случае – элемента c_i. После завершения всех сравнений каждый элемент c_i будет содержать число имен, меньших имени x_i. Чтобы окончательно выполнить сортировку, достаточно поместить каждое имя x_i в позицию $c_i + 1$ (если начальное значение $c_i = 0$) выходной таблицы. Следует отметить, что при правильной реализации сортировка подсчетом обладает свойством устойчивости.

Пример

Исходная таблица X								
i	1	2	3	4	5	6	7	8
x_i	22	54	07	42	14	27	17	49
Счетчики								
	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
	0	0	0	0	0	0	0	0
$i = 1$	3	1	0	1	0	1	0	1
$i = 2$	3	7	0	1	0	1	0	1
$i = 3$	3	7	0	2	1	2	1	2
$i = 4$	3	7	0	5	1	2	1	3
$i = 5$	3	7	0	5	1	3	2	4
$i = 6$	3	7	0	5	1	4	2	5
$i = 7$	3	7	0	5	1	4	2	6
Выходная таблица								
Результат	07	14	17	22	27	42	49	54

В выходную таблицу имя x_i записывается в позицию c_i +1.
Число сравнений: 7+6+5+4+3+2+1=28.

В общем случае число сравнений: $\dfrac{(n-1)n}{2} = O(n^2)$

Таким образом, сортировка подсчетом кроме исходной таблицы требует вспомогательный массив из n элементов для хранения счетчиков c_i и дополнительную выходную таблицу для формирования результатов сортировки. В целях экономии памяти сортировку подсчетом можно выполнить на месте, т. е. переразместить имена внутри исходной таблицы, используя только вспомогательный массив счетчиков. Ясно, что это приведет к некоторому увеличению времени сортировки.

Время работы сортировки подсчетом (независимо от того, используется дополнительная выходная таблица или сортировка выполняется на месте) составляет $O(n^2)$.

Разновидностью сортировки подсчетом является *сортировка распределяющим подсчетом*. Она применима в основном в тех случаях, когда исходная таблица может содержать много равных имен, причем каждое имя является целым положительным числом в диапазоне от a до b.

Сортировка выполняется следующим образом. Каждому имени i из диапазона (не из таблицы) сопоставляется элемент c_i, т. е. требуется вспомогательный массив C из $k = b – a + 1$ элементов. Сначала элементу c_i присваивается количество имен в исходной таблице, равных i. Затем находятся частичные суммы последовательности $c_a, \ldots, c_b$, т. е. для всех i от $a + 1$ до b элементу c_i присваивается $c_i + c_{i–1}$. В результате значение c_i будет показывать количество имен, не превосходящих i, т. е. позицию имени i в отсортированной таблице. Для завершения сортировки имя i помещается в позицию c_i выходной таблицы. При этом необходимо учитывать следующее обстоятельство. Если все n имен в исходной таблице различны, то в отсортированной таблице имя i должно стоять в позиции c_i, так как именно столько имен в таблице не превосходит имя i. Если же встречаются равные имена, то после каждой записи имени i в выходную таблицу значение c_i должно уменьшаться на единицу, поэтому при следующей встрече с именем, равным i, оно будет записано на одну позицию левее. Чтобы сортировка была устойчивой, запись имен в выходную таблицу следует производить, просматривая исходную таблицу справа налево, начиная с имени x_n и завершая x_1.

Пример. Диапазон имен [5, 7], т. е. имеем 3 счетчика: c_5, c_6, c_7.

Исходная таблица X								
i	1	2	3	4	5	6	7	8
x_i	7_1	6_1	7_2	5_1	5_2	7_3	6_2	5_3

Результат прохода по таблице: $c_5 = 3$, $c_6 = 2$, $c_7 = 3$, число операций $O(n)$.

Результат подсчета частичных сумм: $c_5 = 3$, $c_6 = 5$, $c_7 = 8$, число операций $O(k)$, где k – число счетчиков.

Заполнение выходной таблицы (справа налево):

$i = 8$: имя 5_3 в позицию $c_5 = 3$, устанавливаем $c_5 = 2$.

$i = 7$: имя 6_2 в позицию $c_6 = 5$, устанавливаем $c_6 = 4$.

$i = 6$: имя 7_3 в позицию $c_7 = 8$, устанавливаем $c_7 = 7$.

$i = 5$: имя 5_2 в позицию $c_5 = 2$, устанавливаем $c_5 = 1$.

$i = 4$: имя 5_1 в позицию $c_5 = 1$, устанавливаем $c_5 = 0$.

$i = 3$: имя 7_2 в позицию $c_7 = 7$, устанавливаем $c_7 = 6$.

$i = 2$: имя 6_1 в позицию $c_6 = 4$, устанавливаем $c_6 = 3$.

$i = 1$: имя 7_1 в позицию $c_7 = 6$, устанавливаем $c_7 = 5$.

Число операций $O(n)$.

Выходная таблица:

i	1	2	3	4	5	6	7	8
x_i	5_1	5_2	5_3	6_1	6_2	7_1	7_2	7_3

Таким образом, сортировка распределяющим подсчетом дополнительно к исходной таблице требует вспомогательный массив из k элементов для хранения счетчиков и выходную таблицу для записи результатов сортировки. В целях экономии памяти сортировку распределяющим подсчетом можно выполнить на месте внутри исходной таблицы, что несколько усложнит алгоритм и приведет к дополнительным затратам времени.

Подсчет числа имен, равных i, и запись имен в выходную таблицу требуют времени $O(n)$, предварительная инициализация счетчиков и нахождение частичных сумм – времени $O(k)$. Таким образом, сортировка выполняется за время $O(n + k)$. Если $k = O(n)$, то время работы есть $O(n)$. Полученная оценка не противоречит нижней оценке эффективности алгоритмов сортировки, рассмотренной в разд. 5.1. Это связано с тем, что нижние оценки определялись для алгоритмов сортировки, основанных на сравнении имен, а сортировка распределяющим подсчетом не сравнивает имена между собой, а использует их в качестве индексов.