Тема 5. Сортировка

Сортировкой множества элементов называется расположение этих элементов по возрастанию или убыванию в соответствии с определенным отношением линейного порядка. Обычно рассматривается отношение естественного порядка, когда элементы располагаются по возрастанию. Формально задача сортировки определяется следующим образом. Дано конечное множество $T = \{x_1, x_2, \ldots, x_n\}$, которое будем называть *таблицей*. Требуется найти перестановку $\Pi = \{\pi_1, \pi_2, \ldots, \pi_n\}$ этих n элементов, которая отобразит данную таблицу (множество T) в неубывающую последовательность $x_{\pi_1} \le x_{\pi_2} \le \ldots \le x_{\pi_n}$. Как правило, алгоритмы сортировки вырабатывают саму упорядоченную последовательность, а не упорядочивающую перестановку П.

В общем случае элементы таблицы могут иметь сложную структуру, но с каждым элементом ассоциирован некоторый *ключ* (*имя*), используемый для того, чтобы отличить один элемент от другого. Поскольку рассматривается прежде всего процесс сортировки ключей, идентифицирующих каждый элемент, а остальные компоненты не влияют на упорядочивающую функцию, будем считать, что элементами таблицы являются имена (ключи).

Алгоритм сортировки называется *устойчивым*, если он сохраняет исходный порядок равных имен. Данное свойство важно в тех случаях, когда элементы уже упорядочены по какому-то вторичному ключу, и необходимо провести сортировку по первичному ключу (не зависящему от вторичного) так, чтобы внутри групп с одинаковыми первичными ключами сохранялся порядок, определяемый вторичным ключом. Если алгоритм сортировки не обладает свойством устойчивости, то эту задачу придется решать, сортируя элементы по составному ключу, являющемуся объединением первичного и вторичного ключей.

Для упрощения анализа алгоритмов сортировки будем считать, что никакие два имени не имеют одинаковых значений, т. е. любые $x_i, x_j \in T$ обладают тем свойством, что если $i \neq j$, то либо $x_i < x_j$, либо $x_i > x_j$. В этом случае сортировка должна найти перестановку П, отображающую исходную таблицу T в возрастающую последовательность $x_{\pi_1} < x_{\pi_2} < \ldots < x_{\pi_n}$. Ограничение $x_i \neq x_j$ при $i \neq j$ упрощает анализ алгоритмов сортировки без потери общности, поскольку и при наличии равных имен корректность идей и алгоритмов не нарушается.

Выделяют два вида сортировки: внутреннюю и внешнюю. *Внутренняя сортировка* решает задачу полной сортировки для случая достаточно малой таблицы, умещающейся непосредственно в оперативной памяти. *Внешняя сортировка* решает задачу полной сортировки для случая такой большой таблицы, не умещающейся в оперативной памяти, что доступ к ней организован по частям, расположенным на внешних запоминающих устройствах.

Время работы многих алгоритмов сортировки зависит от содержимого исходной таблицы, поэтому для анализа эффективности ряда алгоритмов необходим некоторый критерий, позволяющий оценить степень неотсортированности входной таблицы. В качестве такого критерия удобно использовать понятие инверсии перестановки.

Пусть $X = (x_1, x_2, \ldots, x_n)$ есть некоторая перестановка. Пара (x_i, x_j) называется *инверсией* перестановки X, если $i < j$, а $x_i > x_j$. *Вектором инверсий* перестановки X называется последовательность целых чисел $D = (d_1, d_2, \ldots, d_n)$, где d_j – число элементов x_i, таких, что пара (x_i, x_j) является инверсией. Другими словами, элемент d_j – это число элементов, больших x_j и стоящих слева от него в перестановке X, т. е. $0 \le d_j < j$. Вектор инверсий однозначно определяет перестановку. Например, вектором инверсий перестановки

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 27 & 07 & 54 & 42 & 14 & 10 & 25 & 17 \end{pmatrix}$$

будет

$$\begin{array}{cccccccc} j & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ d_j & 0 & 1 & 0 & 1 & 3 & 4 & 3 & 4 \end{array}.$$

Предположим, что исходное множество имен представляется с помощью массива, хотя возможны и другие способы представления, например, связный список.

Поскольку во многих алгоритмах сортировки используется операция обмена значений имен, для более компактной записи алгоритмов операцию обмена будем записывать в виде $x_i \leftrightarrow x_j$ (значения имен x_i и x_j меняются местами).

5.1. Оценки эффективности алгоритмов сортировки

Предварительно рассмотрим задачу сортировки с теоретической точки зрения, чтобы получить некоторое представление об ожидаемой эффективности. Для многих алгоритмов сортировки хорошей мерой производимой работы является число сравнений имен. Эта характеристика не всегда является определяющей для эффективности алгоритмов сортировки, поскольку существуют алгоритмы, в которых число обменов преобладает над числом сравнений имен; существуют также алгоритмы сортировки, в которых отсутствует прямое сравнение имен. Тем не менее для большинства алгоритмов сортировки указанная характеристика является определяющей. Поэтому представляет интерес минимальное число сравнений, необходимых для сортировки n имен.

Выполнение любого алгоритма сортировки, основанного на сравнении имен, можно представить в виде расширенного бинарного дерева решений, в котором каждая внутренняя вершина соответствует сравнению имен x_i и x_j (обозначим через $x_i : x_j$), а каждая внешняя вершина (лист) – исходу алгоритма. Два сына внутренней вершины показывают два возможных исхода сравнения. Бинарное дерево решений для сортировки трех имен представлено на рис. 5.1.

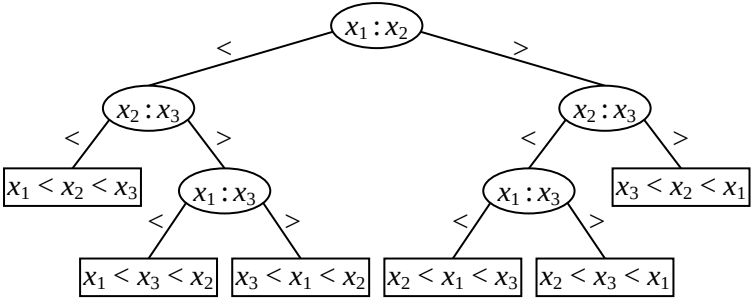

Рис.5.1. Бинарное дерево решений для сортировки трех имен

В любом таком дереве решений каждая перестановка определяет единственный путь от корня к листу. Поскольку алгоритмы сортировки должны правильно работать на всех $n!$ перестановках имен, листья, соответствующие разным перестановкам, должны быть разными. Следовательно, в дереве решений для сортировки n имен должно быть по крайней мере $n!$ листьев. Очевидно, что высота дерева решений равна числу сравнений, требующихся алгоритму в наихудшем случае. Обозначим через $C(n)$ минимальное число сравнений, выполняемых любым алгоритмом сортировки в худшем случае. Поскольку бинарное дерево высоты h может иметь не более 2^h листьев, должно выполняться условие $2^{C(n)} \geq n!$. Используя формулу Стирлинга (формулу приближенного вычисления факториалов), получаем

$$C(n) \geq \log n! \approx n \log n.$$

Таким образом, любой алгоритм сортировки, основанный на сравнении имен, в наихудшем случае потребует не меньше $n \log n$ сравнений.

Минимальное среднее число сравнений имен $C_{ave}(n)$, выполняемых алгоритмом, который правильно сортирует все $n!$ перестановок при условии, что все они равновероятны, также легко определяется с помощью бинарного дерева решений. Известно, что длина внешних путей дерева решений равна сумме всех расстояний от корня до листьев. Деление длины внешних путей на число листьев, дает среднюю длину внешних путей, т. е. среднее число сравнений для соответствующего алгоритма. Известно, что минимальную длину внешних путей (соответственно и минимальное среднее число сравнений) имеют полностью сбалансированные деревья. Для расширенного бинарного дерева с N внешними вершинами минимальная длина внешних путей равна $N \log N + O(N)$. Положив $N = n!$ и разделив на число листьев $n!$, получим соотношение

$$C_{ave}(n) \geq \log n! \approx n \log n.$$

Таким образом, любой алгоритм сортировки, основанный на сравнении имен, потребует в среднем не меньше $n \log n$ сравнений имен.

Полученные результаты для $C(n)$ и $C_{ave}(n)$ дают некоторый эталон для сравнения эффективности многих алгоритмов сортировки, основанных на сравнении имен.