УДК 681.3.06.(075.3)

АЛГОРИТМЫ СОРТИРОВКИ И ПОИСКА, ПРЕДНАЗНАЧЕННЫЕ ДЛЯ ОБРАБОТКИ ДАННЫХ ПРИ РЕШЕНИИ ЗАДАЧ НА КОМПЬЮТЕРЕ

Аязбаев Т.Л., Галагузова Т.А.

Таразский инновационно-гуманитарный университет, Тараз, e-mail: tamara5024@mail.ru

Данная статья посвящена составлению и анализу компьютерных алгоритмов. Тщательно подобранный материал, включает в себя основные фундаментальные классы алгоритмов, которые в том или ином виде наиболее часто встречаются в практике программирования. Точно также от порядка, в котором хранятся элементы в памяти компьютера, во многом зависит скорость выполнения и простота алгоритмов, предназначенных для их обработки. Сортировка позволяет использовать последовательный доступ к большим массивам в качестве приемлемой замены прямой адресации. Сортировка используется и при поиске; с её помощью можно сделать результаты обработки данных более удобными для восприятия человеком. Появление изощрённых алгоритмов сортировки говорит о том, что она и сама по себе интересна как объект исследования. Каждый метод сортировки имеет свои преимущества и недостатки, поэтому он оказывается эффективнее других при некоторых конфигурациях данных и аппаратуры.

Ключи: компьютерные алгоритмы, фундаментальные классы алгоритмов, программирование, скорость выполнения и простота алгоритмов, сортировка, последовательный доступ к большим массивам, методы сортировки

ALGORITHMS OF THE SORTING AND SEARCHING FOR, INTENDED FOR DATA PROCESSING AT SOLUTION OF THE PROBLEMS BY MEANS OF COMPUTER

Ayazbaev T.L., Galaguzova T.A.

Taraz Innovation and Humanitarian University, Taraz, e-mail: tamara5024@mail.ru

Given article is dedicated to formation and analysis computer algorithm. Carefully selected material, comprises of itself main fundamental classes algorithm, which in that or other type most often meet in practical person of the programming. Exactly also from order, in which kept elements in memories of the computer, in многом depends the velocity of the execution and simplicity algorithm, intended for their processing. Sorting allows to use the consequent access to big array as acceptable change to direct addressing. Sorting is used and at searching for; with her(its) help possible to do the results a data processing more suitable for perception of the person. The Appearance refined algorithm sorting speaks of that that she and itself interesting as object of the study. Each method of the sorting has their own advantage and defect so he turns out to be the эффективнее others under some desk side data and equipment's.

Keywords: computer algorithms, fundamental classes algorithm, programming, velocity of the execution and simplicity algorithm, sorting, consequent access to big array, methods of the sorting

Ученый Средней Азии Мухаммед ал-Хорезми, известный своими математическими, астрономическими и географическими трудами, в начале IX в. знакомит арабов с индийской нумерацией. Оригинал назывался «Арифметика в индийской нумерации». В Европе был издан латинский перевод его, сделанный в XII в. под названием «Алхорезми об индийском числе». Перевод этот начинался словами «Dixit Algorithmi, – сказал ал-Хорезми». Отсюда произошло очень широко употребляемое ныне в вычислительной математике слово «алгоритм», означающее всякий порядок действий или правило для получения того или иного результата [1].

Алгоритм решения математической задачи можно записать в виде *формул*. Наибольшее распространение получило *схемное* изображение. Оно является более наглядным. В этом случае алгоритм представлен в виде определённым образом *соединённых блоков*, внутри которых даётся поясняющая информация. Блок-схемой называется графическое изображение логической структуры алгоритма, в котором каждый процесс переработки данных представлен в виде геометрических фигур (блоков), имеющих определённую конфигурацию, в зависимости от выполняемых действий. На правила составления блок-схем (конфигурация, размер блоков и порядок построения схем) существует государственный стандарт ГОСТ 19.002-80 и ГОСТ 19.003-80 «Символы в схемах алгоритмов и программ». Двумерная структура блок-схем плохо приспособлена для ввода и обработки компьютерами, поэтому, как правило, исполнителями алгоритма, записанного в виде блок-схемы, является человек. Данная статья посвящена составлению и анализу компьютерных алгоритмов. Логика работы программ почти всегда поясняется простыми блок-схемами. Тщательно подобранный материал, включает в себя основные фундаментальные классы алгоритмов, которые в том или ином виде наиболее часто встречаются в практике программирования. Изучим вопрос,

который часто возникает в программировании: переразмещение элементов в порядке возрастания или убывания. Представьте, насколько трудно было бы пользоваться словарём, если бы слова в нём не располагались в алфавитном порядке. Точно также от порядка, в котором хранятся элементы в памяти компьютера, во многом зависит скорость выполнения и простота алгоритмов, предназначенных для их обработки [2].

Хотя в словарях слово «сортировка» (sorting) определяется как процесс разделения объектов по виду или сорту, программисты традиционно используют его в гораздо узком смысле, обозначая им такую перестановку предметов, при которой они располагаются в порядке возрастания или убывания. Такой процесс, пожалуй, следовало бы назвать не сортировкой, а упорядочением (ordering), но использование этого слова привело бы к путанице из-за перегруженности значениями слова «порядок». Рассмотрим, например, следующее предложение: «Так как только два из имеющихся у нас лентопротяжных механизмов в порядке, меня призвали к порядку и обязали в срочном порядке заказать ещё несколько устройств, чтобы можно было упорядочивать данные разного порядка на несколько порядков быстрее». В математической терминологии это слово также изобилует значениями (порядок группы, порядок перестановки, порядок точки ветвления, отношения порядка и т.п.). Итак, слово «порядок» приводит к хаосу. Конечно, слово «сортировка» имеет довольно много значений, но оно прочно вошло в программистский жаргон. В дальнейшем будем использовать слово «сортировка» в узком смысле: «размещение по порядку». Вот некоторые из наиболее важных областей применения сортировки.

a) *Решение задачи группирования*, когда нужно собрать вместе все элементы с одинаковыми значениями некоторого признака. Допустим, имеется 10 000 элементов, расположенных в случайном порядке, причём значения многих из них равны и нужно переупорядочить массив так, чтобы элементы с равными значениями занимали соседние позиции в массиве. Это, по существу, тоже задача «сортировки», но в более широком смысле, и она легко может быть решена путём сортировки массива в указанном выше узком смысле, а именно – в результате расположения элементов в порядке неубывания $\upsilon 1 \le \upsilon 2 \le \cdots \le \upsilon 10000$. Эффект, который может быть достигнут после выполнения этой процедуры, и объясняет изменение первоначального смысла слова «сортировка».

b) Поиск общих элементов в двух или более массивах. Если два или более массивов рассортировать в одном и том же порядке, то можно отыскать в них все общие элементы за один последовательный просмотр всех массивов без возвратов. Именно этим принципом и воспользовался Перри Мейсон, чтобы раскрыть дело об убийстве.

Из книги о детективе Перри Мейсоне. «Но мы не успели просмотреть все номера автомобилей», – возразил Дрейк. «А нам и не нужно этого делать, Пол. Мы просто расположим их по порядку и поищем одинаковые», – сказал Перри Мейсон [3].

Оказывается, что, как правило, гораздо удобнее просматривать список последовательно, а не перескакивая с места на место случайным образом, если только список не настолько мал, что он целиком помещается в оперативной памяти. Сортировка позволяет использовать последовательный доступ к большим массивам в качестве приемлемой замены прямой адресации.

c) *Поиск информации по значениям ключей*. Сортировка используется и при поиске; с её помощью можно сделать результаты обработки данных более удобными для восприятия человеком. В самом деле, подготовленный компьютером список, рассортированный в алфавитном порядке, зачастую выглядит весьма внушительно, даже если содержащиеся в нём числовые данные были рассчитаны неверно.

Хотя сортировка традиционно и большей частью использовалась для обработки коммерческих данных, в действительности она является инструментом, полезным в самых разных ситуациях, и поэтому о ней не следует забывать. Можно применить сортировку для упрощения алгебраических формул. Сортировка заслуживает серьёзного изучения с точки зрения её практического использования. Но даже если бы сортировка была почти бесполезна, нашлась бы масса других причин заняться ею! Появление изощрённых алгоритмов сортировки говорит о том, что она и сама по себе интересна как объект исследования. В этой области существует множество увлекательных нерешенных задач наряду с весьма немногими уже решенными.

Рассматривая вопрос в более широком плане, мы обнаружим, что алгоритмы сортировки представляют собой интересный *частный пример* того, как следует подходить к решению проблем программирования вообще. Обобщение этих частных методов позволит нам в значительной степени овладеть теми подходами, которые помогут создавать качественные алгоритмы для решения других проблем, связанных с использованием компьютеров.

Методы сортировки служат великолепной иллюстрацией базовых концепций *анализа алгоритмов*, т. е. оценки качества алгоритмов, что, в свою очередь, позволяет разумно делать выбор среди, казалось бы, равноценных методов. Прежде чем двигаться дальше, необходимо более четко сформулировать задачу и ввести соответствующую терминологию. Пусть надо упорядочить N элементов

$$R_1, R_2, \ldots, R_N.$$

Назовём их *записями*, а всю совокупность N записей назовём *файлом*. Каждая запись R_j имеет *ключ*, K_j, который и управляет процессом сортировки. Помимо ключа, запись может содержать дополнительную «сопутствующую информацию», которая не влияет на сортировку, но всегда остаётся в этой записи.

Отношение порядка «<» на множестве ключей вводится таким образом, чтобы для любых трёх значений ключей *a, b, c* выполнялись следующие условия:

i) справедливо одно и только одно из соотношений $a < b$, $a = b$, $b < a$ (закон трихотомии);

ii) если $a < b$ и $b < c$, то $a < c$ (закон транзитивности).

Эти два свойства определяют математическое понятие *линейного упорядочения*, называемого также *совершенным упорядочением*. Любое множество с отношением «<», удовлетворяющим обоим этим свойствам, поддаётся сортировке большинством методов, описанных в этой статье, хотя некоторые из методов годятся только для числовых и буквенных ключей с общепринятым отношением порядка.

Задача сортировки – найти такую перестановку записей $p(1)\, p(2) \ldots p(N)$ с индексами $\{1,2,\ldots,N\}$, после которой ключи расположились бы в порядке неубывания:

$$K_{p(1)} \le K_{p(2)} \le \ldots \le K_{p(N)} \qquad (1)$$

Сортировка называется *устойчивой*, если она удовлетворяет такому дополнительному условию, что записи с одинаковыми ключами остаются в прежнем порядке, т.е., другими словами,

$$p(i) < p(j) \text{ для любых } K_{p(i)} = K_{p(j)} \text{ и } i < j. \qquad (2)$$

В одних случаях придётся физически перемещать записи в памяти так, чтобы их ключи были упорядочены; в других случаях достаточно создать вспомогательную таблицу, которая некоторым образом описывает перестановку и обеспечивает доступ к записям в соответствии с порядком их ключей.

Некоторые методы сортировки предполагают существование величин «∞» и «– ∞» или одной из них. Величина «∞» считается больше, а величина «– ∞» меньше любого ключа:

$$-\infty < K_j < \infty \text{ для } 1 \le j \le N. \qquad (3)$$

Эти величины используются в качестве искусственных ключей, а также граничных признаков. Равенство в (3), вообще говоря, исключено. Если же оно, тем не менее, допускается, алгоритмы можно модифицировать так, чтобы они всё-таки работали, хотя нередко при этом их изящество и эффективность отчасти утрачиваются.

Обычно методы сортировки подразделяют на два класса: внутренние, когда все записи хранятся в быстрой оперативной памяти, и внешние, когда все записи в ней не помещаются. Методы внутренней сортировки обеспечивают большую гибкость при построении структур данных и доступа к ним, внешние же методы обеспечивают достижения нужного результата в «спартанских» условиях ограниченных ресурсов.

Достаточно хороший общий алгоритм затрачивает на сортировку N записей время пропорционально $N \log N$; при этом требуется около $\log N$ «проходов» по данным. Это минимальное время, если записи расположены в произвольном порядке и сортировка выполняется попарным сравнением ключей. Если же удвоить число записей, то и время при прочих равных условиях возрастает немногим больше чем вдвое. (На самом деле, когда N неограниченно возрастает, время сортировки растет как $N(\log N)^2$, если все ключи различны, поскольку и размеры ключей увеличиваются, как минимум, пропорционально $\log N$ с ростом N; но практически N всегда остаётся ограниченным).

С другой стороны, если известно, что ключи являются случайными величинами с некоторым непрерывным распределением, то, как мы увидим ниже, сортировка может быть выполнена в среднем за $O(N)$ шагов.

Комбинаторные свойства перестановок. Перестановкой конечного множества называется некоторое расположение его элементов в ряд. Перестановки особенно важны при изучении алгоритмов сортировки, так как они служат для представления неупорядоченных исходных данных. Чтобы исследовать эффективность различных методов сортировки, нужно уметь подсчитывать количество перестановок, которые вынуждают повторять некоторый шаг алгоритма определённое число раз.

Внутренняя сортировка. Начнём обсуждение «хорошего» процесса сортировки с маленького эксперимента. Как бы вы решили следующую задачу программирования?

Задание: В ячейках памяти R + 1, R + 2, R + 3, R + 4, R + 5 содержится пять чисел. Напишите программу, которая переразмещает, если нужно, эти числа так, чтобы они расположились в порядке возрастания. Если вы уже знакомы с методами сортировки, постарайтесь на минуту забыть о них; вообразите, что вы решаете такую задачу впервые, не имея никакого представления о том, как к ней подступиться. Время, затраченное на решение приведённой выше задачи, окупится с лихвой. Возможно, ваше решение окажется одним из следующих:

a) *сортировка методом вставок*. Элементы просматриваются по одному, и каждый новый элемент вставляется в подходящее место среди ранее упорядоченных элементов;

b) *обменная сортировка*. Если два элемента расположены не по порядку, то они меняются местами. Этот процесс повторяется до тех пор, пока элементы не будут упорядочены;

c) *сортировка посредством выбора*. Сначала выделяется наименьший (или наибольший) элемент и каким-либо образом отделяется от остальных, затем выбирается наименьший (наибольший) из оставшихся и т.д.;

d) *сортировка путём подсчёта*. Каждый элемент сравнивается со всеми остальными; окончательное положение элемента определяется после подсчёта числа меньших ключей;

e) *специальная сортировка*. Она хороша для пяти элементов, указанных в задаче, но не поддаётся простому обобщению, если элементов больше;

f) *новая суперметодика сортировки*. Это существенно усовершенствованные известные методы [3].

Приведем примеры простых алгоритмов сортировки одномерного массива.

Сортировка выбором. Начинаем с первого элемента массива. Ищем в массиве наименьший элемент m и меняем его местами с первым. Поскольку наименьший элемент массива после перестановки гарантированно размещается на первом месте, дальнейший поиск начинаем со второго элемента и меняем местами второй с наименьшим из оставшихся. Таким образом, получаем двойной цикл сортировки.

Сортировка обменами (метод пузырька). В этом методе организуется последовательный перебор элементов массива A_1, A_2, ... A_N и сравнение значений 2-х соседних элементов. Если впереди находится элемент с большим значением, выполняется перестановка (при сортировке по возрастанию). Сортировка по убыванию будет отличаться только знаком операции (<, а не >) в операторе сравнения соседних элементов. Переменная m теперь используется как буфер для перестановки 2-х соседних элементов в массиве. Этот метод называют также методом пузырька, так как при его реализации наибольшие или наименьшие текущие элементы как пузырьки «поднимаются» к началу массива (или «опускаются» к его концу).

Сортировка простыми вставками. Последовательно просматриваем элементы массива A_1, A_2, ... A_N и каждый просматриваемый элемент этой последовательности вставляем на подходящее место в уже упорядоченную последовательность A_1, ... A_i. Место вставки определяется последовательным сравнением значения A_1 с предварительно упорядоченными значениями A_1, ... A_{i-1} . Затем в найденном месте соседние элементы массива «раздвигаются», освобождая место под вставляемый элемент. Для сортировки по убыванию по-прежнему достаточно заменить знак в операции сравнения – «<» на «>».

Существуют и значительно более мощные алгоритмы сортировки, однако, их применение имеет смысл для действительно больших объемов данных.

Заключение

Было изобретено множество различных алгоритмов сортировки, может быть 25 или более того. Такое пугающее количество методов на самом деле – лишь малая толика всех алгоритмов, придуманных на сегодняшний день. Почему же существует так много методов сортировки? Ответ если и неочевиден, то вполне понятен – каждый метод имеет свои преимущества и недостатки, поэтому он оказывается эффективнее других при некоторых конфигурациях данных и аппаратуры. К сожалению, неизвестен наилучший способ сортировки (если он вообще существует); имеется *много* наилучших методов, но только в случае, когда известно, *что* сортируется, на *каком* компьютере и с какой *целью*.

Список литературы

1. Депман И.Я. История арифметики. Пособие для учителей. 2-е изд. испр. – М.: Просвещение. 1965. – С. 416.

2. Галагузова Т.А. Алгоритмизация и программирование. Учебное пособие для студентов специальности «Информатика», «Вычислительная техника и программное обеспечение» и экономических специальностей. – Тараз.: ТОО «Рысбаева и К», 2008. – 200 с.

3. Дональд Э. Кнут. Искусство программирования, том 3. Сортировка и поиск, 2-е изд.: Пер. с англ. – М.: Издательский дом «Вильямс», 2001. – 832 с.