5.5. РЕЗЮМЕ. ИСТОРИЯ И БИБЛИОГРАФИЯ

ТЕПЕРЬ, КОГДА мы почти достигли конца этой очень длинной главы, "рассортируем" наиболее важные из рассмотренных выше сведений.

Алгоритм сортировки — это процедура, которая реорганизует файл записей таким образом, что ключи оказываются расположенными в порядке возрастания. Вследствие такого упорядочения группируются записи с равными ключами, становится возможной эффективная обработка множества файлов, рассортированных по одному ключу, создается информационная основа для эффективных алгоритмов выборки и более убедительно выглядят документы, подготовленные с помощью компьютера.

Внутренняя сортировка используется в тех случаях, когда все записи помещаются в быстродействующей внутренней памяти компьютера. Мы изучили с разной степенью детализации более двух десятков алгоритмов внутренней сортировки. Но, наверное, было бы куда спокойнее, если бы мы не знали столько различных подходов к решению данной задачи! Изучение всех этих методов было приятным времяпрепровождением. Но теперь перед нами безрадостная перспектива — предстоит на деле решить, какой метод следует использовать в той или иной конкретной ситуации.

Было бы прекрасно, если бы один или два метода сортировки превосходили все остальные безотносительно к приложению или используемой машине. На самом же деле каждый метод имеет свои собственные, одному ему присущие достоинства. Например, метод пузырька (алгоритм 5.2.2B) не имеет ярко выраженных преимуществ, так как всегда можно найти лучший способ сделать то, что он делает; но даже он после соответствующего обобщения оказывается полезным для сортировки с двумя лентами (см. раздел 5.4.8). Итак, приходим к заключению, что почти все алгоритмы заслуживают того, чтобы о них помнили, так как существуют приложения, в которых какой-либо из них оказывается самым подходящим.

В следующем кратком обзоре освещаются основные аспекты наиболее важных алгоритмов внутренней сортировки, с которыми мы встречались. (Как обычно, N означает число записей в файле.)

1. *Распределяющий подсчет.* Если диапазон ключей невелик, очень полезен алгоритм 5.2D. Метод устойчив (не изменяет порядок записей с равными ключами), но требуется память для счетчиков и $2N$ записей. Одна из модификаций, позволяющая сэкономить N из этих записей ценой устойчивости, встречается в упр. 5.2–13.

2. *Простая вставка.* Алгоритм 5.2.1S наиболее прост для программной реализации, не требует дополнительного объема памяти и довольно эффективен при малых N (скажем, при $N \leq 25$). При больших N он становится невыносимо медленным, если только исходные данные не окажутся сразу почти упорядоченными.

3. *Сортировка с убывающим смещением.* Алгоритм 5.2.1D (метод Шелла) также довольно просто программируется, использует минимальный объем памяти и весьма эффективен при умеренно больших N (скажем, при $N \leq 1000$).

4. *Вставка в список.* Алгоритм 5.2.1L основан на той же идее, что и алгоритм простой вставки, и поэтому годится только при небольших N. Как и в других методах сортировки списков, в нем благодаря операциям со ссылками экономятся затраты на пересылку длинных записей; это особенно удобно, когда записи имеют переменную длину или являются частью других структур данных.

5. *Сортировка с вычислением адреса* эффективна, если ключи подчиняются известному (обычно — равномерному) закону распределения; важнейшими вариантами этого подхода являются *вставки в несколько списков* (программа 5.2.1M) и комбинированная поразрядная сортировка со вставками Мак-Ларена (рассмотренная в конце раздела 5.2.5). Для последнего метода достаточно иметь $O(\sqrt{N})$ дополнительных ячеек памяти. Двухпроходный метод, который позволяет иметь дело с неравномерным распределением, анализируется в теореме 5.2.5T.

6. *Обменная сортировка со слиянием.* Алгоритм 5.2.2M (метод Бэтчера) и родственный ему алгоритм *битонной сортировки* (упр. 5.3.4–10) полезны, если можно одновременно выполнять большое число сравнений.

7. *Обменная сортировка с разделением* (метод Хоара, широко известный как быстрая сортировка). Алгоритм 5.2.2Q, вероятно, — самый полезный универсальный алгоритм внутренней сортировки, поскольку он требует очень мало памяти и опережает своих конкурентов по среднему времени выполнения на большинстве компьютеров. Однако в наихудшем случае он может работать *очень* медленно. Поэтому, если вероятность неслучайных данных достаточно велика, приходится тщательно выбирать разделяющие элементы. Если выбирается медиана из трех элементов (как предлагается в упр. 5.2.2–55), то такое поведение, как в наихудшем случае, становится крайне маловероятным и, кроме того, несколько уменьшается среднее время работы.

8. *Простой выбор.* Алгоритм 5.2.3S довольно прост и особенно подходит в случае, когда имеется специальное оборудование для быстрого поиска наименьшего элемента в списке.

9. *Пирамидальная сортировка.* Алгоритм 5.2.3H при минимальных требованиях к памяти обеспечивает достаточно высокую скорость сортировки; как среднее, так и максимальное время работы примерно вдвое больше среднего времени быстрой сортировки.

10. *Слияние списков.* Алгоритм 5.2.4L осуществляет сортировку списков, обеспечивая при этом, так же как и алгоритм пирамидальной сортировки, весьма высокую скорость даже в наихудшем случае. Кроме того, данный метод устойчив по отношению к равным ключам.

11. *Поразрядная сортировка* с использованием алгоритма 5.2.5R — это не что иное, как сортировка списков, которая приемлема для ключей либо очень коротких, либо имеющих необычный порядок лексикографического сравнения. Вместо ссылок можно применить распределяющий подсчет (п. 1 нашего обзора); такая процедура потребует пространства для $2N$ записей и для таблицы счетчиков, но благодаря простоте внутреннего цикла она особенно хороша для сверхбыстрых компьютеров — "пожирателей чисел", имеющих опережающее управление. (*Предостережение.* Поразрядную сортировку не следует использовать при малых N!)

12. *Сортировка методом вставок и слияния* (см. раздел 5.3.1) наиболее приемлема при очень малых N в "прямолинейных" программах. Например, этот метод оказался бы подходящим в тех приложениях, в которых требуется сортировать много групп из пяти или шести записей.

13. Могут существовать и гибридные методы, объединяющие один или более из приведенных выше. Например, короткие подфайлы, возникающие при быстрой сортировке, можно сортировать методом слияния и вставок.

14. И наконец, для реализации безымянного метода, встретившегося в ответе к упр. 5.2.1–3, требуется, по-видимому, кратчайшая из возможных программ сортировки. Но среднее время работы такой программы пропорционально N^3, т. е. это самая медленная программа сортировки из упомянутых в данной книге!

Пространственные и временные характеристики многих из этих методов, запрограммированных для компьютера MIX, сведены в табл. 1. Важно иметь в виду, что числа в данной таблице являются лишь грубыми оценками относительного времени сортировки. Они применимы только к одному компьютеру, и предположения, касающиеся исходных данных, далеко не для всех программ абсолютно правомерны. Сравнительные таблицы, подобные этой, приводились во многих работах, но не найдется таких двух авторов, которые пришли бы к одинаковым выводам! Тем не менее данные о времени работы позволяют оценить хотя бы порядок скорости, которую следует ожидать от каждого алгоритма при сортировке записей из одного слова, так как MIX — довольно типичный компьютер.

В столбце "Память" в табл. 1 содержится некоторая информация об объеме вспомогательной памяти, используемой каждым алгоритмом, в единицах длины записи. Здесь буквой ϵ обозначена доля записи, необходимая для одного поля связи; так, например, $N(1+\epsilon)$ означает, что методу требуется пространство для N записей и N полей связи.

В асимптотических оценках среднего и максимального времени, приведенных в табл. 1, учитываются только главные члены, доминирующие при больших N в предположении случайных исходных данных; c обозначает произвольную константу. Эти формулы могут иногда ввести в заблуждение, поэтому указано также фактическое время выполнения программы для двух конкретных последовательностей исходных данных. Случай $N = 16$ относится к шестнадцати ключам, так часто появлявшимся в примерах раздела 5.2, а случай $N = 1000$ относится к последовательности $K_1, K_2, \ldots, K_{1000}$, определенной формулами

$$K_{1001} = 0; \qquad K_{n-1} = (3141592621K_n + 2113148651) \bmod 10^{10}.$$

Для получения характеристик каждого алгоритма, представленного в таблице, использовалась достаточно совершенная программа для MIX, как правило, учитывающая усовершенствования, которые описаны в упражнениях. Размер байта при выполнении этих программ принят равным 100.

Для внешней сортировки необходимы методы, отличающиеся от методов внутренней сортировки, потому что предполагается использование сравнительно простых структур данных и большое внимание уделяется уменьшению времени ввода-вывода. В разделе 5.4.6 рассматриваются интересные методы, разработанные для сортировки данных на магнитных лентах, а в разделе 5.4.9 обсуждается использование дисков и барабанов.

Конечно, сортировка — не единственная тема этой главы. Попутно мы много узнали о том, как работать со структурами данных, обращаться с внешней памятью, анализировать алгоритмы, и о том, как изобретать... новые алгоритмы.

Таблица 1

СРАВНЕНИЕ МЕТОДОВ ВНУТРЕННЕЙ СОРТИРОВКИ ПРИМЕНИТЕЛЬНО К РЕАЛИЗАЦИИ НА КОМПЬЮТЕРЕ MIX

Метод	Где упоминается	Устойчив?	Длина кода MIX	Память	Время выполнения: Среднее	Время выполнения: Макс.	Время выполнения: $N = 16$	Время выполнения: $N = 1000$	Прим.
Сравнение и подсчет	Упр. 5.2–5	Да	22	$N(1+\epsilon)$	$4N^2 + 10N$	$5.5N^2$	1065	3992432	c
Распределение и подсчет	Упр. 5.2–9	Да	26	$2N + 1000\epsilon$	$22N + 10010$	$22N$	10362	32010	a
Простая вставка	Упр. 5.2.1–33	Да	10	$N + 1$	$1.5N^2 + 9.5N$	$3N^2$	412	1491928	
Сортировка с убывающим смещением	Прог. 5.2.1D	Нет	21	$N + \epsilon \lg N$	$3.9N^{7/6} + 10N \lg N + 166N$	$cN^{4/3}$	567	128758	d, h
Вставка в список	Упр. 5.2.1–33	Да	19	$N(1+\epsilon)$	$1.25N^2 + 13.25N$	$2.5N^2$	433	1248615	b, c
Вставка в несколько списков	Прог. 5.2.1M	Нет	18	$N + \epsilon(N + 100)$	$.0175N^2 + 18N$	$3.5N^2$	645	35246	b, c, f, i
Обменная сортировка со слиянием	Упр. 5.2.2–12	Нет	35	N	$2.875N(\lg N)^2$	$4N(\lg N)^2$	939	284366	
Обменная сортировка с разделением	Прог. 5.2.2Q	Нет	63	$N + 2\epsilon \lg N$	$11.67N \ln N - 1.74N$	$\geq 2N^2$	470	81486	
Сортировка методом медианы из трех ключей	Упр. 5.2.2–55	Нет	100	$N + 2\epsilon \lg N$	$10.63N \ln N + 2.12N$	$\geq N^2$	487	74574	e
Обменная поразрядная сортировка	Прог. 5.2.2R	Нет	45	$N + 68\epsilon$	$14.43N \ln N + 23.9N$	$272N$	1135	137614	g, i, j
Простой выбор	Прог. 5.2.3S	Нет	15	N	$2.5N^2 + 3N \ln N$	$3.25N^2$	853	2525287	j
Пирамидальная сортировка	Прог. 5.2.3H	Нет	30	N	$23.08N \ln N + 0.01N$	$24.5N \ln N$	1068	159714	h, j
Слияние списков	Прог. 5.2.4L	Да	44	$N(1+\epsilon)$	$14.43N \ln N + 4.92N$	$14.4N \ln N$	761	104716	b, c, j
Поразрядная сортировка списков	Прог. 5.2.5R	Да	36	$N + \epsilon(N + 200)$	$32N + 4838$	$32N$	4250	36838	b, c

Примечания к табл. 1

a Ключи только из трех цифр

b Ключи только из шести цифр (т. е. из трех байтов)

c Вывод не перекомпонован; результирующая последовательность определяется неявно ссылками или счетчиками

d Смещение выбирается, как в 5.2.1–(11); несколько лучшая последовательность появляется в упр. 5.2.1–29

e $M = 9$, если использовать SRB; для варианта с DIV добавить к среднему времени выполнения $1.60N$

f $M = 100$ (размер байта)

g $M = 34$, так как $2^{34} > 10^{10} > 2^{33}$

h Поскольку теория неполная, данные о среднем времени получены эмпирически

i Оценка среднего времени базируется на предположении о равномерном распределении ключей

j Дальнейшее совершенствование программы, о котором упоминается в тексте раздела и упражнениях, относящихся к этой программе, могут уменьшить время выполнения

Ранние разработки. Поиск прототипов современных методов сортировки возвращает нас в 19 век, когда были изобретены первые машины для сортировки. В Соединенных Штатах Америки перепись всех граждан проводилась каждые 10 лет, и уже к 1880 году проблема, связанная с обработкой огромных по объему данных переписи стала очень острой. В самом деле, число одиноких (т. е. не состоящих в браке) граждан не подсчитывалось ежегодно, хотя вся необходимая информация собиралась. Герман Холлерит (Herman Hollerith), двадцатилетний служащий Бюро переписи, изобрел остроумный электрический табулятор, отвечающий нуждам сбора статистики, и около ста его машин успешно использовались при обработке данных переписи 1890 года.

На рис. 94 изображен первый аппарат Холлерита, приводимый в действие от аккумуляторных батарей. Для нас основной интерес представляет "сортировальный ящик" справа, который открыт для того, чтобы можно было увидеть половину из 26 внутренних отделений. Оператор вставлял перфокарту размером $6\frac{5}{8} \times 3\frac{1}{4}$ дюймов в "пресс" и опускал рукоятку; это приводило к тому, что закрепленные на пружинах штыри на верхней панели в тех местах, где на карте пробиты отверстия, входят в контакт с ртутью на нижней панели. В результате замыкания соответствующей электрической цепи показание связанного с ней циферблата изменялось на 1 и, кроме того, одна из 26 крышек сортировального ящика открывалась. В этот момент оператор отпускал пресс, клал карту в открытое отделение и закрывал крышку. Однажды через эту машину пропустили 19 071 карту за один $6\frac{1}{2}$-часовой рабочий день (в среднем около 49 карт в минуту!). (Средний оператор работал примерно втрое медленней.)

Население продолжало неуклонно расти, и первые табуляторы-сортировщики оказались недостаточно быстрыми, чтобы справиться с обработкой данных переписи 1900 года. Поэтому Холлерит изобрел еще одну машину, чтобы предотвратить еще один кризис в обработке данных. Его новое устройство (запатентованное в 1901 и 1904 годах) автоматически подавало карты и выглядело, в сущности, почти так

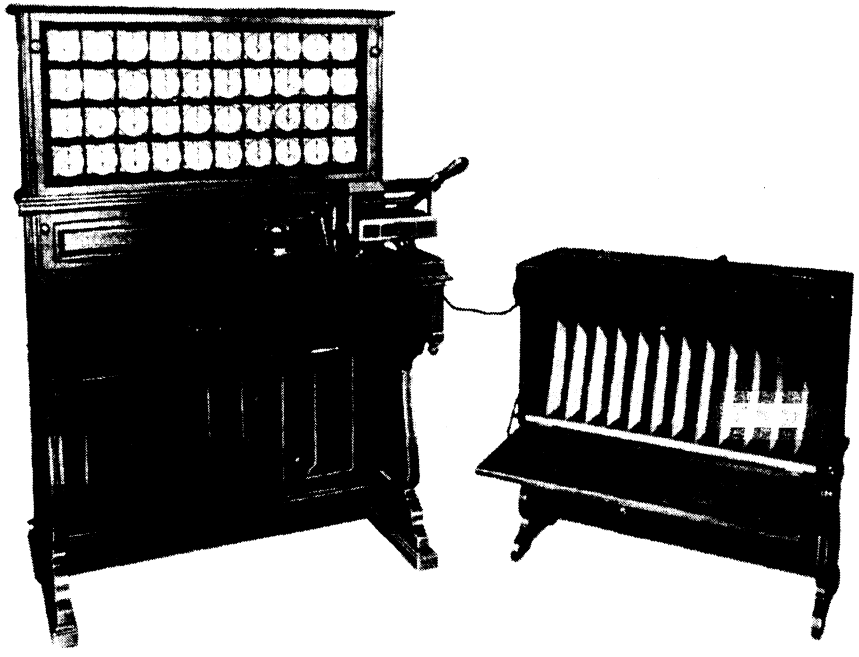

Рис. 94. Табулятор Холлерита. (Фотография любезно предоставлена архивом IBM.)

же, как современные карточные сортировальные машины. История ранних машин Холлерита с интересными подробностями изложена Леоном Э. Трусделлом (Leon E. Truesdell) в книге *The Development of Punch Card Tabulation* (Washington: U. S. Bureau of the Census, 1965); см. также сообщения современников Холлерита: *Columbia College School of Mines Quarterly* **10** (1889), 238–255; *J. Franklin Inst.* **129** (1890), 300–306; *The Electrical Engineer* **12** (November, 11, 1891), 521–530; *J. Amer. Statistical Assn.* **2** (1891), 330–341, **4** (1895), 365; *J. Royal Statistical Soc.* **55** (1892), 326–327; *Allgemeines statistisches Archiv* **2** (1892), 78–126; *J. Soc. Statistique de Paris* **33** (1892), 87–96; *U. S. Patents 395781* (1889), *685608* (1901), *777209* (1904). Холлерит и другой бывший служащий Бюро переписи Джеймс Пауэрс (James Powers) в дальнейшем основали конкурирующие компании, которые, в конце концов, вошли соответственно в корпорации IBM и Remington Rand corporation.

Сортировальная машина Холлерита базировалась, конечно, на методах поразрядной сортировки, используемых в цифровых ЭВМ. В его патенте упоминается, что числовые элементы, содержащие два столбца, должны сортироваться "по отдельности для каждого столбца", но он не говорит, какой столбец (единиц или десятков) должен рассматриваться первым. Патент за номером 518240, выданный Джону К. Гору (John K. Gore) в 1894 году, в котором описана другая машина для сортировки, предполагает начинать со столбца десятков. Далеко не очевидная идея сортировки сначала по столбцу единиц была, по-видимому, открыта каким-то неизвестным оператором и передана остальным (см. раздел 5.2.5); она имеется в самом раннем сохранившемся руководстве IBM по сортировке (1936 г.). Насколько мне известно, впервые метод "справа налево" упоминается в книге Robert Feindler,

Das Hollerith-Lochkarten-Verfahren (Berlin: Reimar Hobbing, 1929), 126–130; почти в это же время он упоминается в статье Л. Дж. Комри (L. J. Comrie), *Transactions of the Office Machinery Users' Association* (London, 1930), 25–37. Кстати, Комри оказался первым, кто сделал следующее важное наблюдение: табуляторы можно с успехом использовать для выполнения научных расчетов, хотя первоначально они создавались для статистических и бухгалтерских приложений. Его статья особенно интересна, поскольку содержит подробное описание табуляторов, имевшихся в Англии в 1930 году. Сортировальные машины в то время обрабатывали от 360 до 400 карт в минуту и сдавались в аренду за 9 фунтов стерлингов в месяц.

Идея слияния восходит к другому устройству для обработки карт — *подборочной машине* (collator), которая была изобретена значительно позднее, в 1938 году. Снабженная двумя подающими механизмами, она могла слить две рассортированные колоды карт в одну всего за один проход; метод выполнения этого слияния хорошо описан в первом руководстве IBM по методам подборки (апрель 1939 года). [См. James W. Bryce, *U. S. Patent 2189024* (1940).]

Затем на сцене появились ЭВМ и разработка методов сортировки тесно переплелась с их развитием. На самом деле имеются свидетельства того, что программа сортировки была первой из когда-либо написанных для вычислительных машин с запоминаемой программой. Конструкторы вычислительной машины EDVAC особенно интересовались сортировкой, поскольку она выступала как наиболее характерный представитель потенциальных нечисловых приложений ЭВМ. Они понимали, что удовлетворительная система команд должна годиться не только для составления программы решения разностных уравнений; она должна обладать достаточной гибкостью, чтобы справиться с комбинаторными аспектами в алгоритмах "выбора решений". Поэтому Джон фон Нейман (John von Neumann) подготовил в 1945 году программы для внутренней сортировки методом слияния, чтобы убедиться в необходимости некоторых типов команд, предложенных им для машины EDVAC. Существование эффективных специализированных сортировальных машин послужило тем естественным стандартом, в сопоставлении с которым можно было оценить достоинства предлагаемой организации электронной вычислительной машины. Подробно это интересное исследование описано в статье D. E. Knuth, *Computing Surveys* **2** (1970), 247–260; первая программа сортировки фон Неймана в окончательном, "отполированном" виде приводится в его сборнике *Collected Works* **5** (New York: Macmillan, 1963), 196–214.

Независимо от них в 1945 году К. Цузе (K. Zuse) в Германии разработал программу для сортировки методом простой вставки. Это был один из самых простых примеров операций с линейными списками в разработанном им языке "Plankalkül". (Эта пионерская работа ожидала публикации почти 30 лет; см. *Berichte der Gesellschaft für Mathematik und Datenverarbeitung* **63** (Bonn, 1972), part 4, 84–85.)

Из-за ограниченного объема памяти в ранних машинах приходилось думать о внешней сортировке наравне с внутренней, и в докладе "Progress Report on the EDVAC", подготовленном Дж. П. Эккертом (J. P. Eckert) и Дж. У. Мочли (J. W. Mauchly) для Школы Мура по электротехнике [Moore School of Electrical Engineering (30 September, 1945)], указывалось, что ЭВМ, оснащенная устройством с магнитной проволокой или лентой, могла бы моделировать действия перфокарточного оборудования, достигая при этом большей скорости сортировки. В этом докладе были

описаны методы сбалансированной двухпутевой поразрядной сортировки и сбалансированного двухпутевого слияния (там он был назван "подборкой" (collating)) с использованием четырех устройств внешней памяти на магнитной проволоке или ленте, читающих или записывающих "не менее 5 000 импульсов в секунду".

Джон Мочли выступил с лекцией о "сортировке и слиянии" на специальной сессии по вычислениям, созывавшейся в Школе Мура в 1946 году. В конспекте его лекции содержится первое опубликованное обсуждение сортировки с помощью вычислительных машин [*Theory and Techniques for the Design of Electronic Digital Computers*, edited by G. W. Patterson, **3** (1946), 22.1–22.20]. Мочли начал свое выступление с интересного замечания: "Требование, чтобы одна машина объединяла возможности вычислений и сортировки, кое-кому может показаться требованием, чтобы один прибор использовался как ключ для консервов и как авторучка". Затем он заметил, что машины, способные выполнять сложные математические процедуры, должны также иметь возможность сортировать и классифицировать данные; он показал, что сортировка может быть полезна даже в связи с численными расчетами. Мочли описал методы простой вставки и бинарных вставок, упомянув, что в первом методе в среднем необходимо около $N^2/4$ сравнений, в то время как в последнем их никогда не требуется более $N \lg N$. Однако для бинарных вставок нужна весьма сложная структура данных и Мочли затем показал, что при двухпутевом слиянии достигается столь же малое число сравнений, но используется только последовательный просмотр списков. Последняя часть конспекта его лекции посвящена методам поразрядной сортировки с частичными проходами, которые моделируют цифровую карточную сортировку на четырех лентах, затрачивая менее четырех проходов на цифру (см. раздел 5.4.7).

Вскоре после этого Эккерт и Мочли организовали компанию, выпустившую некоторые из самых ранних электронных вычислительных машин BINAC (для военных приложений) и UNIVAC (для коммерческих приложений). Вновь Бюро переписи США сыграло в этом свою роль, приобретя первый UNIVAC. В это время далеко не всем было ясно, что ЭВМ станут экономически выгодными: вычислительные машины могли сортировать быстрее, но они и стоили дороже. Поэтому программисты UNIVAC под руководством Фрэнсис Э. Гольбертон (Frances E. Holberton) приложили значительные усилия к созданию программ внешней сортировки, работающих с высокой скоростью; их первые программы повлияли также на разработку оборудования. По их оценкам 100 млн записей по 10 слов могли быть рассортированы на UNIVAC за 9 000 ч (т. е. за 375 дней).

Вычислительная машина UNIVAC I, официально запущенная в эксплуатацию в июле 1951 года, имела оперативную память объемом 1 000 12-буквенных (72-битовых) слов. В ней предусматривались чтение и запись на ленту блоков по 60 слов со скоростью 500 слов в секунду; чтение могло быть прямым или обратным, допускалось совмещение операций чтения, записи и вычислений. В 1948 году миссис Гольбертон придумала интересный способ выполнения двухпутевого слияния с полным совмещением чтения, записи и вычислений с использованием шести буферов ввода. Для каждого вводного файла имелись один "текущий" буфер и два "вспомогательных" буфера; она предложила так организовать слияние, что всякий раз, когда приходило время вывода одного блока, два текущих буфера ввода содержали вместе один блок необработанных данных. Следовательно, за время формирования

каждого выводного блока ровно один буфер ввода становился пустым и можно было организовать работу так, чтобы три вспомогательных буфера из четырех оказывались заполненными всякий раз, когда данные читались в оставшийся буфер. Этот метод чуть быстрее метода прогнозирования (см. алгоритм 5.4.6F), так как нет необходимости проверять результат одного ввода перед началом следующего. [См. *Collation Methods for the UNIVAC System* (Eckert-Mauchly Computer Corp., 1950), 2 volumes.]

Кульминацией этой работы стало создание генератора программ сортировки, который был первой крупной программой, разработанной для автоматического программирования. Пользователь указывал размер записи, позиции ключей (до пяти) в частичных полях каждой записи и "концевые" ключи, отмечающие конец файла, и после этого генератор сортировки порождал требуемую программу сортировки для файлов на одной бобине. На первом проходе этой программы выполнялась внутренняя сортировка блоков по 60 слов с использованием метода сравнения и подсчета (алгоритм 5.2C); затем выполнялось несколько сбалансированных двухпутевых проходов слияния с обратным чтением, исключающих сцепление лент, как описано выше. [См. *Master Generating Routine for 2-way Sorting* (Eckert-Mauchly Division of Remington Rand, 1952). Первый набросок этого доклада был озаглавлен "Основная составляющая программа двухпутевого слияния" (*Master Prefabrication Routine for 2-way Collation*)! См. также F. E. Holberton, *Symposium on Automatic Programming* (Office of Naval Research, 1954), 34–39.]

К 1952 году многие методы внутренней сортировки прочно вошли в программистский фольклор, но теория была развита сравнительно слабо. Даниэль Гольденберг (Daniel Goldenberg) [*Time analyses of various methods of sorting data*, Digital Computer Laboratory memo M-1680 (Mass. Inst. of Tech., 17 October, 1952)] запрограммировал для машины Whirlwind computer пять различных методов и выполнил анализ наилучшего и наихудшего случаев для каждой программы. Он нашел, что для сортировки сотни 15-битовых записей по 8-битовому ключу наилучшие по скорости результаты получаются в том случае, если используется таблица из 256 слов и каждая запись помещается в единственную соответствующую ее ключу позицию, а затем эта таблица сжимается. Однако данный метод имел очевидный недостаток: запись уничтожалась, если последующая имела тот же ключ. Остальные четыре проанализированных метода были ранжированы следующим образом: прямое двухпутевое слияние лучше поразрядной сортировки с основанием 2, которая лучше простого выбора, который, в свою очередь, лучше метода пузырька.

Эти результаты получили дальнейшее развитие в диссертации Гарольда Х. Сьюворда (Harold H. Seward) в 1954 году [*Information sorting in the application of electronic digital computers to business operations*, Digital Computer Lab. report R-232 (Mass. Inst. of Tech., 24 May, 1954; 60 pages)]. Сьюворд высказал идеи распределяющего подсчета и выбора с замещением. Он показал, что первый отрезок случайной перестановки имеет среднюю длину $e - 1$, и проанализировал наряду с методами внутренней сортировки методы внешней сортировки, причем не только на магнитных лентах, но и на устройствах внешней памяти других типов.

Еще более достойная внимания диссертация — на этот раз докторская — была написана Говардом Б. Демутом (Howard B. Demuth) в 1956 году [*Electronic Data Sorting* (Stanford University, October, 1956), 92 pages; *IEEE Trans.* **C-34** (1985),

296–310]. Эта работа помогла заложить основы теории сложности вычислений. В ней рассматривались три абстрактные модели задачи сортировки: с использованием циклической памяти, линейной памяти и памяти с произвольным доступом; для каждой модели были разработаны оптимальные или почти оптимальные методы. (См. упр. 5.3.4–68.) Хотя непосредственно из диссертации Демута не вытекает никаких практических следствий, в ней содержатся важные идеи о том, как связать теорию с практикой.

Таким образом, история решения проблемы сортировки тесно связана с большинством этапов развития вычислительной техники: с первыми машинами для обработки данных, первыми хранимыми программами, первым программным продуктом, первыми методами буферизации, первой работой по анализу алгоритмов и сложности вычислений.

Ни один из документов, касающихся ЭВМ и упомянутых выше, не появлялся в "открытой литературе". Так уж случилось, что почти вся ранняя история вычислительных машин отражена в сравнительно труднодоступных докладах, поскольку относительно немного людей было в то время связано с вычислительной техникой. Наконец, в 1955 и 1956 годах литература о сортировке проникает в печать в виде трех больших обзорных статей. Первая статья была подготовлена Дж. К. Хоскеном (J. C. Hosken) [*Proc. Eastern Joint Computer Conference* **8** (1955), 39–55]. Он начинает с тонкого наблюдения: "Чтобы снизить стоимость единицы результата, люди обычно укрупняют операции. Но при этом стоимость единицы сортировки не уменьшается, а возрастает". Хоскен описал все оборудование специального назначения, имевшееся в продаже, а также методы сортировки с использованием существовавших на то время ЭВМ. Его библиография включает 54 ссылки и основана большей частью на брошюрах фирм-изготовителей.

Подробная статья Э. Г. Френда (E. H. Friend) *Sorting on Electronic Computer Systems* [*JACM* **3** (1956), 134–168] явилась важной вехой в развитии технологии сортировки. Хотя за время, прошедшее с 1956 года, было разработано множество методов, эта статья все еще необычно современна во многих отношениях. Фрэнд дал тщательное описание весьма большого числа алгоритмов внутренней и внешней сортировки и обратил особое внимание на методы буферизации и характеристики накопителей на магнитных лентах. Он предложил некоторые новые методы (например, выбор из дерева, метод двуглавого змия и прогнозирования) и проанализировал некоторые математические свойства старых методов.

Третий обзор по сортировке, который появился в то время, был подготовлен Д. У. Дэвисом (D. W. Davies) [*Proc. Inst. Elect. Engineers* **103B**, Supplement 1 (1956), 87–93]. В последующие годы было опубликовано еще несколько прекрасных обзоров: D. A. Bell [*Comp. J.* **1** (1958), 71–77], A. S. Douglas [*Comp. J.* **2** (1959), 1–9]; D. D. McCracken, H. Weiss, T. Lee [*Programming Business Computers* (New York: Wiley, 1959), Chapter 15, pages 298–332], I. Flores [*JACM* **8** (1961), 41–80], K. E. Iverson [*A Programming Language* (New York: Wiley, 1962), Chapter 6, 176–245], C. C. Gotlieb [*CACM* **6** (1963), 194–201], T. N. Hibbard [*CACM* **6** (1963), 206–213], M. A. Goetz [*Digital Computer User's Handbook*, edited by M. Klerer and G. A. Korn (New York: McGraw-Hill, 1967), Chapter 1.10, pages 1.292–1.320]. В ноябре 1962 года ACM организовала симпозиум по сортировке (бóльшая часть работ, представленных на этом симпозиуме, опубликована в мае 1963 года в выпуске *CACM*). Они

дают хорошее представление о состоянии работ в этой области на то время. Обзор К. К. Готлиба (C. C. Gotlieb) о современных генераторах сортировки, обзор Т. Н. Хиббарда (T. N. Hibbard) о внутренней сортировке с минимальной памятью и раннее исследование Дж. А. Хаббарда (G. U. Hubbard) о сортировке файлов на дисках — статьи из этого сборника, заслуживающие наибольшего внимания.

За прошедший период были открыты новые методы сортировки: вычисление адреса (1956), слияние с вставкой (1959), обменная поразрядная сортировка (1959), каскадное слияние (1959), метод Шелла с убывающим смещением (1959), многофазное слияние (1960), вставки в дерево (1960), осциллирующая сортировка (1962), быстрая сортировка Хоара (1962), пирамидальная сортировка Уильямса (1964), обменная сортировка со слиянием Бэтчера (1964). История каждого отдельного алгоритма прослеживается в тех разделах настоящей главы, в которых этот метод описывается. Конец 60-х годов нашего столетия ознаменовался интенсивным развитием соответствующей теории.

Полная библиография всех работ, изученных автором при написании и подготовке первого издания этой главы и составленная с помощью Р. Л. Ривест (R. L. Rivest), приводится в *Computing Reviews* **13** (1972), 283–289.

Последние достижения. Со времени выхода из печати первого издания этой книги (1970 г.) появилось много сообщений об изобретении новых алгоритмов сортировки, однако почти все они являются вариациями уже известных алгоритмов. *Быстрая сортировка на множестве ключей*, которая обсуждалась в упр. 5.2.2–30, представляет собой прекрасный пример таких более новых методов.

Другая тенденция в этой области, представляющая, скорее, теоретический интерес, — изучение *адаптивных* схем сортировки. Такие схемы по замыслу разработчиков должны гарантировать более быстрое выполнение сортировки в случаях, когда входная последовательность удовлетворяет какому-либо из заранее установленных критериев. [См., например, H. Mannila, *IEEE Transactions* **C-34** (1985), 318–325; V. Estivill-Castro, D. Wood, *Computing Surveys* **24** (1992), 441–476; C. Levcopoulos, O. Petersson, *Journal of Algorithms* **14** (1993), 395–413; A. Moffat, G. Eddy, O. Petersson, *Software Practice & Experience* **26** (1996), 781–797.]

Разительные изменения в характеристиках аппаратного обеспечения современных компьютерных систем стимулировали интерес к исследованию проблемы эффективности алгоритмов сортировки при совершенно новых критериях стоимости затрат. [См., например, обсуждение применения виртуальной памяти в упр. 5.4.9–20.] Эффект от применения аппаратной кэш-памяти при внутренней сортировке анализировался в работе A. LaMarca, R. E. Ladner, *J. Algorithms* **31** (1999), 66–104. Ее авторы пришли к выводу, что не следует включать шаг Q9 в алгоритм 5.2.2Q, если для сортировки используется компьютер с современной архитектурой, хотя для компьютеров традиционной структуры, каковым является наш MIX, алгоритм в прежнем виде вполне работоспособен. Вместо того чтобы завершать быструю сортировку с помощью метода прямых вставок, предлагается, что, по мнению авторов, значительно лучше, заранее сортировать короткие подмассивы, сохраняя их ключи в кэш-памяти.

А что можно сказать о нынешнем состоянии в области сортировки больших массивов данных? Одним из распространенных тестов для сравнительной оценки

различных средств решения такого рода задач с 1985 года стала сортировка миллиона 100-символьных записей с равномерно распределенными случайными 10-символьными ключами. Предполагается, что исходная последовательность и результат должны храниться на диске, а целью является минимизация времени обработки, включая время загрузки и запуска программы. В работе R. C. Agarwal, *SIGMOD Record* **25**, 2 (June, 1996), 240–246, описан эксперимент, проведенный на компьютере IBM RS/6000, модель 39H, в котором используется процессор с RISC-архитектурой. Методом поразрядной сортировки обрабатывались файлы, распределенные между восемью дисками, и задача была решена за 5.1 с. В подобных экспериментах узким местом является скорость ввода-вывода. Процессору для решения этой задачи понадобилось всего 0.6 с! Можно получить еще более высокую скорость, если параллельно использовать несколько процессоров. Сеть из 32 рабочих станций UltraSPARC I, каждая из которых была оснащена двумя собственными дисками, может сортировать миллион записей за 2.41 с, используя гибридный метод, названный NOW-Sort [A. C. Arpaci-Dusseau, R. H. Arpaci-Dusseau, D. E. Culler, J. M. Hellerstein, D. A. Patterson, *SIGMOD Record* **26**, (June, 1997), 243–254].

Изложенное выше подводит нас к выводу, что предлагаемый тест сортировки миллиона записей является, скорее, оценкой времени ввода-вывода, чем эффективности метода собственно сортировки; входные последовательности бóльшей длины потребуют других, более "значимых", тестов. Например, последовательность поистине глобального масштаба для *терабайтовой сортировки* — 10^{10} записей по 100 символов — была рассортирована за 2.5 ч. Этот результат получен в сентябре 1997 года на системе Silicon Graphics Origin2000, включающей 32 процессора, 8 Гбайт оперативной памяти и 559 дисков по 4 Гбайт. Массив был рассортирован с помощью доступной на рынке программного обеспечения программы Nsort™, разработанной К. Нибергом (C. Nyberg), Ч. Койстером (C. Koester) и Дж. Греем (J. Gray), в которой используется неопубликованный до сих пор метод обработки.

Но даже тест сортировки терабайтовых массивов может по нынешним временам оказаться недостаточно емким показателем. Наилучший из имеющихся на сегодняшний день претендентов на звание "универсального" теста эффективности сортировки, который обещает жить вечно (во всяком случае, так нам хотелось бы), — это так называемый критерий *Минут-Сорт*. Суть его состоит в выяснении, сколько 100-символьных записей можно рассортировать за 60 с. Когда данная книга была на последней стадии готовности к печати, рекордсменом по этому критерию был метод NOW-Sort; 30 марта 1997 года 95 рабочим станциям, объединенным в сеть, понадобилось всего 59.21 с на сортировку 90.25 млн записей. Но и результаты, полученные на самых современных системах, не опровергли ни одну из основополагающих оценок, полученных теоретически.

Подводя итог, можно с уверенностью заявить, что проблема эффективности сортировки остается сегодня такой же злободневной, как и ранее.

УПРАЖНЕНИЯ

1. [*05*] Подведите итог этой главе; сформулируйте обобщение теоремы 5.4.6A.

2. [*20*] Взяв за основу табл. 1, скажите, какой из методов сортировки списков с шестиразрядными ключами будет наилучшим для машины MIX.

3. [*37*] (*Устойчивая сортировка с минимальным объемом памяти.*) Считается, что алгоритм сортировки требует минимальной памяти, если он использует для своих переменных только $O((\log N)^2)$ бит памяти сверх пространства, необходимого для размещения N записей. Алгоритм должен быть общим в том смысле, что он должен работать при любых N, а не только при определенном значении N, если, конечно, предполагается, что при вызове алгоритма для сортировки обеспечивается достаточное количество памяти с произвольным доступом. Во многих изученных нами алгоритмах сортировки это требование минимальной памяти нарушается; в частности, запрещено использование N полей связи. Быстрая сортировка (алгоритм 5.2.2Q) удовлетворяет требованию минимальной памяти, но время выполнения в наихудшем случае пропорционально N^2. Пирамидальная сортировка (алгоритм 5.2.3H) является единственным среди изученных нами алгоритмом типа $O(N \log N)$, который использует минимальный объем памяти, хотя можно сформулировать еще один подобный алгоритм, если использовать идею из упр. 5.2.4–18.

Самым быстрым общим алгоритмом из изученных нами, который *устойчиво* сортирует ключи, является метод слияния списков (алгоритм 5.2.4L), однако он использует не минимальную память. Фактически единственными устойчивыми алгоритмами сортировки с минимальной памятью, которые мы анализировали, были методы типа $\Omega(N^2)$ (простые вставки, метод пузырька и вариации на тему простого выбора).

Разработайте устойчивый алгоритм сортировки с минимальной памятью, требующий менее $O(N(\log N)^2)$ машинных циклов в наихудшем случае. [*Указание.* Можно создать устойчивый метод слияния, удовлетворяющий критерию минимальной памяти, который затрачивает на сортировку $O(N \log N)$ машинных циклов.]

▶ **4.** [*28*] Алгоритм сортировки называется *бережливым*, если принимает решение только на основе сравнения ключей и никогда не выполняет тех сравнений, результат которых может быть предсказан на основе сравнений, выполненных ранее. Какие из перечисленных в табл. 1 методов являются бережливыми?

5. [*46*] Довольно сложно сравнивать неслучайные данные, в которых присутствует множество равных ключей. Придумайте тест для определения эффективности сортировки, который (i) был бы актуальным как сегодня, так и через 100 лет, (ii) не включал бы в рассмотрение случай равномерно распределенных случайных ключей и (iii) не использовал бы входных последовательностей данных, которые изменились бы со временем.

Я посчитал бы цель достигнутой, если бы мне удалось рассортировать и расположить в логическом порядке основную часть того огромного материала, касающегося сортировки, который появился за несколько последних лет.

— ДЖ. К. ХОСКЕН (J. C. HOSKEN) (1955)