5.3. ОПТИМАЛЬНАЯ СОРТИРОВКА

ТЕПЕРЬ, КОГДА МЫ проанализировали множество методов внутренней сортировки, пришло время обратиться к более общему вопросу: *какой метод внутренней сортировки наилучший?* Существует ли такой верхний предел скорости сортировки, которого бы не мог достичь ни один программист, как бы искусен он ни был?

Разумеется, *не существует* наилучшего возможного способа сортировки; мы должны точно определить, что понимать под словом "наилучший", но не существует наилучшего возможного способа определения слова "наилучший". Аналогичную проблему, связанную с оптимальностью алгоритмов, мы обсуждали в разделах 4.3.3, 4.6.3 и 4.6.4, в которых рассматривались умножение с повышенной точностью и вычисление полиномов. В каждом случае, для того чтобы задача была поставлена в терминах конкретных математических структур, необходимо было сформулировать довольно простое определение "наилучшего из возможных" алгоритма. И в каждом случае перед нами вставали интереснейшие проблемы, настолько сложные, что они до сих пор полностью не решены. Так же обстоит дело и с сортировкой: были получены некоторые интересные результаты, но осталось еще много интригующих вопросов, на которые до сих пор нет ответов.

Изучение внутреннего механизма методов сортировки обычно было направлено на минимизацию числа сравнений ключей при сортировке n элементов или слиянии m элементов с n элементами, или выборе t-го наибольшего элемента из неупорядоченного набора n элементов. В разделах 5.3.1–5.3.3 эти вопросы обсуждаются для общего случая; в разделе 5.3.4 рассматриваются аналогичные вопросы с интересным ограничением: схема (структура) сравнений должна быть, по сути, заранее фиксированной. Другие интересные теоретические вопросы, связанные с оптимальной сортировкой, можно найти в упражнениях к разделу 5.3.4 и в разделах 5.4.4, 5.4.8 и 5.4.9, в которых анализируется внешняя сортировка.

> *Как только появится аналитическая машина, она, безусловно, определит дальнейший путь развития науки. Всякий раз, когда с ее помощью будет найден какой-либо результат, возникнет вопрос: "Существует ли метод вычислений, которым можно получить на этой машине тот же результат, но затратив минимум времени?".*
>
> — ЧАРЛЬЗ БЭББИДЖ (1864)

5.3.1. Сортировка с минимальным числом сравнений

Очевидно, минимальное число сравнений ключей, необходимое для сортировки n элементов, равно *нулю*, поскольку, как мы видели, существуют методы поразрядной сортировки, в которых сравнения вообще не выполняются. В самом деле, можно разработать MIX-программы, способные сортировать и, тем не менее, не содержащие ни одной команды условного перехода! (См. упр. 5–8 в начале этой главы.) Мы также встречались с несколькими методами сортировки, которые, по сути, были основаны на сравнении ключей, но время выполнения которых на деле определялось другими факторами, такими как перезапись данных, вспомогательные операции и т. д.

Поэтому ясно, что подсчет числа сравнений — не единственный способ измерения эффективности метода сортировки. Однако в любом случае небезынтерес-

но провести тщательное исследование числа сравнений, поскольку теоретический анализ этого вопроса позволит нам более глубоко проникнуть во внутреннюю природу процессов сортировки, а также поможет отточить мастерство для решения задач, более близких к практике, которые могут возникнуть перед нами в будущем. Исключим из рассмотрения метод поразрядной сортировки, в котором совсем не выполняется сравнений, и ограничимся обсуждением методов, основанных только на абстрактном линейном отношении порядка "$<$" между ключами, рассмотренном в начале этой главы. Для простоты мы также ограничимся случаем *различных ключей*, а это значит, что при любом сравнении ключей K_i и K_j возможны лишь два исхода: либо $K_i < K_j$, либо $K_i > K_j$. (Распространение результатов такого анализа на общий случай, когда допускаются равные ключи, приводится в упр. 3–12.) Ограничения на время выполнения в худшем случае рассматриваются в работах Fredman, Willard, *J. Computer и Syst. Sci.* **47** (1993), 424–436, Ben-Amram, Galil, *J. Comp. Syst. Sci.* **54** (1997), 345–370 и Thorup, *SODA* **9** (1998), 550–555.

Возможны и другие эквивалентные варианты постановки задачи сортировки посредством сравнений. Если есть n грузов и весы с двумя чашами, то каково минимальное число взвешиваний, необходимое для того, чтобы расположить грузы по порядку в соответствии с весом, если в каждой чаше весов помещается только один груз? Или же, если в некотором турнире участвуют n игроков, то каково наименьшее число парных встреч, достаточное для того, чтобы распределить места между соревнующимися в предположении, что силы игроков можно линейно упорядочить (ничейные результаты не допускаются).

Методы сортировки n элементов, удовлетворяющие указанным ограничениям, можно представить с помощью структуры расширенного бинарного дерева, такого, которое показано на рис. 34. Каждый *внутренний узел* (изображенный в виде кружочка) содержит два индекса "$i:j$" и означает сравнение ключей K_i и K_j. Левое поддерево этого узла соответствует последующим сравнениям, которые необходимо выполнить, если $K_i < K_j$, а правое поддерево — тем действиям, которые необходимо предпринять, если $K_i > K_j$. Каждый *внешний узел дерева* (изображенный в виде прямоугольника) содержит перестановку $a_1\, a_2 \ldots a_n$ множества $\{1, 2, \ldots, n\}$, а это обозначает, что было установлено упорядочение

$$K_{a_1} < K_{a_2} < \cdots < K_{a_n}.$$

(Если взглянуть на путь от корня к этому внешнему узлу, то каждое из $n-1$ соотношений $K_{a_i} < K_{a_{i+1}}$, где $1 \le i < n$, — результат некоторого сравнения $a_i:a_{i+1}$ или $a_{i+1}:a_i$ на этом пути.)

Так, на рис. 34 представлен метод сортировки, суть которого состоит в том, чтобы сначала сравнить K_1 с K_2; если $K_1 > K_2$, то продолжать (двигаясь по правому поддереву) сравнивать K_2 с K_3, а затем, если $K_2 < K_3$, сравнить K_1 с K_3; наконец, если $K_1 > K_3$, становится ясно, что $K_2 < K_3 < K_1$. Реальный алгоритм сортировки будет также перезаписывать ключи в массиве, но нас интересуют только сравнения, поэтому мы игнорируем все перезаписи данных. При сравнении K_i с K_j в этом дереве всегда имеются в виду *исходные ключи* K_i и K_j, а не те ключи, которые могли занять i- и j-ю позиции в массиве в результате перемещения записей.

Возможны и избыточные сравнения; например, на рис. 35 нет необходимости сравнивать 3:1, поскольку из неравенств $K_1 < K_2$ и $K_2 < K_3$ следует $K_1 < K_3$. Ни-

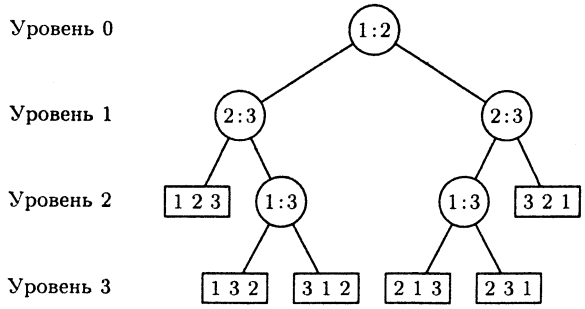


Рис. 34. Дерево сравнений для сортировки трех элементов.

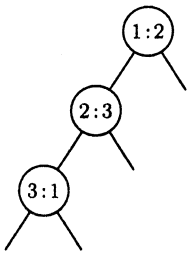


Рис. 35. Пример избыточного сравнения.

какая перестановка не может соответствовать левому поддереву узла $3:1$ на рис. 35, так что эта часть алгоритма никогда не будет выполняться! Поскольку нас интересует минимальное число сравнений, можно считать, что избыточные сравнения не выполняются. С этого момента мы будем иметь дело со структурой расширенного бинарного дерева, в котором каждому внешнему узлу соответствует некоторая перестановка. Все перестановки исходных ключей возможны, и каждая перестановка определяет единственный путь от корня к внешнему узлу; отсюда вытекает, что *в дереве сравнений для сортировки n элементов без избыточных сравнений имеется ровно $n!$ внешних узлов.*

Оптимизация в наихудшем случае. Первая естественным образом возникающая задача — найти деревья сравнений, минимизирующие *максимальное* число выполняемых сравнений. (Позже мы рассмотрим *среднее* число сравнений.)

Пусть $S(n)$ — минимальное число сравнений, достаточное для сортировки n элементов. Если все внутренние узлы в дереве сравнений располагаются на уровнях $< k$, то очевидно, что в дереве не может быть более 2^k внешних узлов. Следовательно, полагая $k = S(n)$, имеем

$$n! \le 2^{S(n)}.$$

Поскольку $S(n)$ — целое число, можно записать эту формулу иначе, получив нижнюю оценку:

$$S(n) \ge \lceil \lg n! \rceil. \tag{1}$$

Заметим, что по формуле Стирлинга

$$\lceil \lg n! \rceil = n \lg n - n/\ln 2 + \tfrac{1}{2} \lg n + O(1), \tag{2}$$

следовательно, необходимо выполнить около $n \lg n$ сравнений. Соотношение (1) часто называют *теоретико-информационной нижней оценкой*, поскольку специалист в области теории информации сказал бы, что в процессе сортировки проверяется $\lg n!$ "бит информации"; каждое сравнение дает не более одного бита информации. Такие деревья, как на рис. 34, называют также "вопросниками" (questionnaires); их математические свойства исследованы в книге Claude Picard, *Théorie des Questionnaires* (Paris: Gauthier-Villars, 1965).

Из всех рассмотренных нами методов сортировки три метода требуют меньше всего сравнений: бинарные вставки (см. раздел 5.2.1), выбор из дерева (см. раздел 5.2.3) и простое двухпутевое слияние, описанное в алгоритме 5.2.4L). Нетрудно видеть, что максимальное число сравнений для метода бинарных вставок равно

$$B(n) = \sum_{k=1}^{n} \lceil \lg k \rceil = n\lceil \lg n \rceil - 2^{\lceil \lg n \rceil} + 1 \tag{3}$$

(см. упр. 1.2.4–42), а максимальное число сравнений для двухпутевого слияния определено в упр. 5.2.4–14. Оказывается (см. раздел 5.3.3), что для метода выбора из дерева верхняя оценка числа сравнений либо такая же, как для метода бинарных вставок, либо такая же, как для двухпутевого слияния, в зависимости от вида дерева. Во всех трех случаях имеем асимптотическое значение $n \lg n$; объединяя верхнюю и нижнюю оценки для $S(n)$, получим

$$\lim_{n\to\infty} \frac{S(n)}{n \lg n} = 1. \tag{4}$$

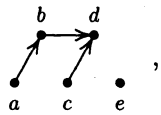
Таким образом, мы получили приближенную формулу для $S(n)$, однако желательно иметь более точную оценку. В следующей таблице приведены значения верхней и нижней границ при малых n.

$n =$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
$\lceil \lg n! \rceil =$	0	1	3	5	7	10	13	16	19	22	26	29	33	37	41	45	49
$B(n) =$	0	1	3	5	8	11	14	17	21	25	29	33	37	41	45	49	54
$L(n) =$	0	1	3	5	9	11	14	17	25	27	30	33	38	41	45	49	65

Здесь $B(n)$ и $L(n)$ относятся соответственно к методам бинарных вставок и слияния списков. Можно показать, что $B(n) \le L(n)$ при любом n (см. упр. 2).

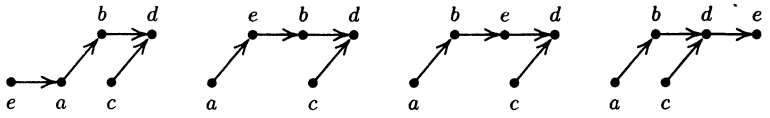
Как видно из приведенной выше таблицы, $S(4) = 5$, но $S(5)$ может равняться либо 7, либо 8. В результате снова приходим к задаче, поставленной в начале раздела 5.2. Каков наилучший способ сортировки пяти элементов? Возможна ли сортировка пяти элементов при помощи всего семи сравнений?

Такая сортировка возможна, но сам способ найти не так просто. Начинаем так же, как при сортировке четырех элементов посредством слияний, сравнивая сначала $K_1 : K_2$, затем — $K_3 : K_4$, а затем — наибольшие элементы обеих пар. Эти сравнения порождают конфигурацию, которую можно изобразить диаграммой



, (5)

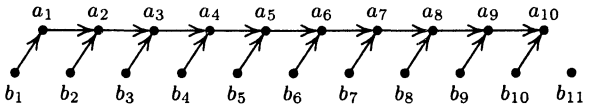
показывающей, что $a < b < d$ и $c < d$. (Для представления известных отношений порядка между элементами удобно воспользоваться ориентированными графами, такими, что x считается меньше y тогда и только тогда, когда на графе есть путь от x к y.) Теперь вставляем пятый элемент $K_5 = e$ в соответствующее место среди $\{a, b, d\}$; для этого требуются всего два сравнения, поскольку можно сравнить его сначала с b, а затем — с a или d. Таким образом, остается один из четырех возможных вариантов



(6)

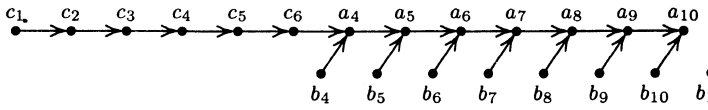
и в каждом случае достаточно еще двух сравнений, чтобы вставить c в цепочку остальных элементов, меньших d. Такой способ сортировки пяти элементов впервые обнаружил Г. Б. Демут (H. B. Demuth) [Ph. D. thesis, Stanford University (1956), 41–43].

Сортировка посредством вставок и слияния. Изящное обобщение изложенного выше метода принадлежит Лестеру Форду (мл.) (Lester Ford, Jr.) и Селмеру М. Джонсону (Selmer M. Johnson). Поскольку оно объединяет некоторые особенности двух способов сортировки (посредством слияний и посредством вставок), мы назовем этот метод *сортировкой посредством вставок и слияния*. Рассмотрим, например, сортировку 21 элемента. Начать можно со сравнений десяти пар ключей $K_1 : K_2, K_3 : K_4, \ldots, K_{19} : K_{20}$; затем следует рассортировать посредством вставок и слияния бóльшие элементы пар. В результате получим конфигурацию



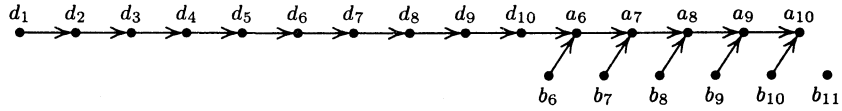
, (7)

аналогичную (5). Следующий шаг — вставить элемент b_3 в последовательность $\{b_1, a_1, a_2\}$, а затем — b_2 в последовательность остальных элементов, меньших a_2. В результате приходим к конфигурации



. (8)

Назовем верхние элементы *главной цепочкой*. Элемент b_5 можно вставить в главную цепочку за три сравнения (сравнив его сначала с c_4, затем с c_2 или c_6 и т. д.). После этого еще за три сравнения можно переместить в главную цепочку b_4 и получить



. (9)

"Следующий шаг решающий; ясно ли вам, что делать дальше?" При помощи всего четырех сравнений вставляем b_{11} (но *не* b_7) в главную цепочку. Далее элементы b_{10}, b_9, b_8, b_7, b_6 (именно в таком порядке) можно вставить в нужное место в главной цепочке не более чем за четыре сравнения каждый.

Аккуратный подсчет числа требуемых сравнений показывает, что 21 элемент можно рассортировать не более чем за $10+S(10)+2+2+3+3+4+4+4+4+4+4=66$ шагов. Поскольку

$$2^{65} < 21! < 2^{66},$$

ясно также, что и в любом другом случае необходимо не менее 66 сравнений; следовательно,

$$S(21) = 66. \tag{10}$$

(При сортировке с помощью бинарных вставок понадобилось бы 74 сравнения.)

В общем случае сортировка посредством вставок и слияния для n элементов выглядит следующим образом.

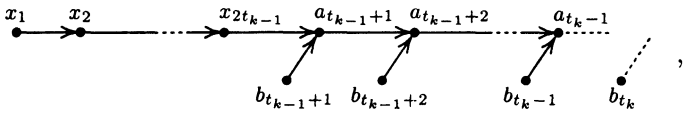
i) Сравнить $\lfloor n/2 \rfloor$ непересекающихся пар элементов. (Если n нечетно, то один элемент не участвует в сравнениях.)

ii) Рассортировать $\lfloor n/2 \rfloor$ бóльших элементов пар, найденных на шаге (i), посредством вставок и слияния.

iii) Для элементов введем обозначения $a_1, a_2, \ldots, a_{\lfloor n/2 \rfloor}$, $b_1, b_2, \ldots, b_{\lceil n/2 \rceil}$, как в (7), где $a_1 \le a_2 \le \cdots \le a_{\lfloor n/2 \rfloor}$ и $b_i \le a_i$ при $1 \le i \le \lfloor n/2 \rfloor$; назовем b_1 и все элементы a главной цепочкой. Не трогая элементов b_j при $j > \lceil n/2 \rceil$, вставим посредством бинарных вставок в главную цепочку остальные элементы b в следующем порядке:

$$b_3, b_2;\ b_5, b_4;\ b_{11}, b_{10}, \ldots, b_6;\ \ldots;\ b_{t_k}, b_{t_k-1}, \ldots, b_{t_{k-1}+1};\ \ldots. \tag{11}$$

Наша цель — сформировать последовательность $(t_1, t_2, t_3, t_4, \ldots) = (1, 3, 5, 11, \ldots)$, присутствующую в (11), таким образом, чтобы каждый из элементов $b_{t_k}, b_{t_k-1}, \ldots, b_{t_{k-1}+1}$ можно было вставить в главную цепочку не более чем за k сравнений. Обобщая (7)–(9), получим диаграмму



,

на которой главная цепочка до a_{t_k-1} включительно содержит $2t_{k-1} + (t_k - t_{k-1} - 1)$ элементов. Это число должно быть меньше 2^k; для нас лучше всего положить его равным $2^k - 1$, и тогда

$$t_{k-1} + t_k = 2^k. \tag{12}$$

Поскольку $t_1 = 1$, для удобства можно положить $t_0 = 1$. В итоге, суммируя члены геометрической прогрессии, найдем

$$\begin{aligned} t_k = 2^k - t_{k-1} = 2^k - 2^{k-1} + t_{k-2} = \cdots &= 2^k - 2^{k-1} + \cdots + (-1)^k 2^0 \\ &= \bigl(2^{k+1} + (-1)^k\bigr)/3\,. \end{aligned} \tag{13}$$

(Любопытно, что точно такая же последовательность возникла при изучении алгоритма вычисления наибольшего общего делителя двух целых чисел; см. упр. 4.5.2–36.)

Пусть $F(n)$ — число сравнений, необходимых для сортировки n элементов посредством вставок и слияния. Ясно, что

$$F(n) = \lfloor n/2 \rfloor + F(\lfloor n/2 \rfloor) + G\bigl(\lceil n/2 \rceil\bigr), \tag{14}$$

где функция G описывает объем работы, выполняемой на шаге (iii). Если $t_{k-1} \le m \le t_k$, то, суммируя по частям, получаем

$$G(m) = \sum_{j=1}^{k-1} j(t_j - t_{j-1}) + k(m - t_{k-1}) = km - (t_0 + t_1 + \cdots + t_{k-1}). \tag{15}$$

Положим

$$w_k = t_0 + t_1 + \cdots + t_{k-1} = \lfloor 2^{k+1}/3 \rfloor, \tag{16}$$

и тогда $(w_0, w_1, w_2, w_3, w_4, \dots) = (0, 1, 2, 5, 10, 21, \dots)$. В упр. 13 показано, что

$$F(n) - F(n-1) = k \qquad \text{тогда и только тогда, когда} \qquad w_k < n \le w_{k+1}, \tag{17}$$

а последнее условие эквивалентно неравенствам

$$\frac{2^{k+1}}{3} < n \le \frac{2^{k+2}}{3}$$

или $k + 1 < \lg 3n \le k + 2$; следовательно,

$$F(n) - F(n-1) = \lceil \lg \tfrac{3}{4} n \rceil. \tag{18}$$

(Эта формула получена А. Хадьяном (A. Hadian) [Ph. D. thesis, Univ. of Minnesota (1969), 38–42].) Отсюда вытекает, что функция $F(n)$ выражается на удивление простой формулой

$$F(n) = \sum_{k=1}^{n} \lceil \lg \tfrac{3}{4} k \rceil, \tag{19}$$

которая очень похожа на формулу (3) для метода бинарных вставок. В замкнутом виде эту сумму можно найти в упр. 14.

Воспользовавшись (19), нетрудно построить таблицу значений функции $F(n)$; имеем

$n =$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
$\lceil \lg n! \rceil =$	0	1	3	5	7	10	13	16	19	22	26	29	33	37	41	45	49
$F(n) =$	0	1	3	5	7	10	13	16	19	22	26	30	34	38	42	46	50

$n =$	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
$\lceil \lg n! \rceil =$	53	57	62	66	70	75	80	84	89	94	98	103	108	113	118	123
$F(n) =$	54	58	62	66	71	76	81	86	91	96	101	106	111	116	121	126

Обратите внимание на то, что $F(n) = \lceil \lg n! \rceil$ при $1 \le n \le 11$ и при $20 \le n \le 21$; таким образом, при этих значениях n сортировка посредством вставок и слияния оптимальна:

$$S(n) = \lceil \lg n! \rceil = F(n) \qquad \text{при } n = 1, \dots, 11, 20 \text{ и } 21. \tag{20}$$

Задачу нахождения функции $S(n)$ поставил Гуго Штейнгауз (Hugo Steinhaus) во втором издании своей классической книги *Mathematical Snapshots* (Oxford University Press, 1950, 38–39)*. Он описал метод бинарных вставок, который является наилучшим способом сортировки n элементов при условии, что n-й элемент не рассматривается до тех пор, пока не рассортированы первые $n - 1$ элементов; он

* Есть русский перевод первого издания этой книги: Штейнгауз Г. Математический калейдоскоп. — М.: Гостехиздат, 1949. — *Прим. перев.*

Таблица 1

ЗНАЧЕНИЯ ФАКТОРИАЛОВ В ДВОИЧНОЙ СИСТЕМЕ СЧИСЛЕНИЯ

$$(1)_2 = 1!$$
$$(10)_2 = 2!$$
$$(110)_2 = 3!$$
$$(11000)_2 = 4!$$
$$(1111000)_2 = 5!$$
$$(1011010000)_2 = 6!$$
$$(1001110110000)_2 = 7!$$
$$(1001110110000000)_2 = 8!$$
$$(1011000100110000000)_2 = 9!$$
$$(11011101011111000000000)_2 = 10!$$
$$(10011000010001010100000000)_2 = 11!$$
$$(11100100011001111110000000000)_2 = 12!$$
$$(10111001100101000110011000000000)_2 = 13!$$
$$(10100010011000011101100101000000000000)_2 = 14!$$
$$(10011000001110111011101110101100000000000)_2 = 15!$$
$$(100110000011101110111011101011000000000000000)_2 = 16!$$
$$(101000011011111101110111011001101100000000000000)_2 = 17!$$
$$(101101011111011101100110010100111001100000000000000000)_2 = 18!$$
$$(110110000001010111001001100000110100010010000000000000000)_2 = 19!$$
$$(100001110000110110011101111100100000101011010000000000000000)_2 = 20!$$

же сделал предположение о том, что метод бинарных вставок оптимален и в общем случае. Несколько лет спустя Штейнгауз сообщил [*Calcutta Math. Soc. Golden Jubilee Commemoration* **2** (1959), 323–327], что двое его коллег, С. Трибула (S. Trybuła) и Чжен Пинг (P. Czen), "недавно" опровергли его предположение и нашли значения $S(n)$ при $n \leq 11$. Вероятно, Трибула и Чжен Пинг независимо пришли к сортировке посредством вставок и слияния, о чем вскоре появилась публикация Ford, Johnson, *AMM* **66** (1959), 387–389.

После открытия сортировки посредством вставок и слияния первым неизвестным значением функции $S(n)$ оставалось $S(12)$. Из табл. 1 видно, что число 12! довольно близко к 2^{29}, поэтому существование 29-шаговой процедуры сортировки 12 элементов весьма маловероятно. Для решения этого вопроса Марком Уэлсом (Mark Wells) был предпринят продолжительный вычислительный эксперимент (продолжавшийся на компьютере Maniac II около 60 ч), который показал, что $S(12) = 30$ [*Proc. IFIP Congress 65* **2** (1965), 497–498]. Итак, процедура вставок и слияний оказывается оптимальной и при $n = 12$.

***Более глубокий анализ.** Чтобы более тщательно исследовать функцию $S(n)$, внимательно изучим диаграммы частичного упорядочения, подобные (5). Полученные после нескольких сравнений сведения можно представить в виде ориентированного графа. Этот граф в силу транзитивности отношения "$<$" не содержит циклов. Следовательно, его всегда можно изобразить таким образом, чтобы все дуги были ориентированы слева направо; поэтому удобнее удалить из диаграммы стрелки. В результате диаграмма (5) преобразуется в