КАЗАНСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ

ИНСТИТУТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ
И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Кафедра системного анализа и информационных технологий

Д.А. Дябилкин

СОРТИРУЮЩИЕ СЕТИ

Методическое пособие

Казань — 2015

УДК 004.42
ББК 22.12

Принято на заседании кафедры системного анализа
и информационных технологий
Протокол № 7 от 14 апреля 2015 года

Рецензенты:
доктор технических наук,
профессор кафедры САИТ КФУ **Е.Л. Столов;**
ст. преп. кафедры теоретической кибернетики КФУ **Р.М. Хадиев**

Дябилкин Д.А.
Сортирующие сети: Методическое пособие / Д.А. Дябилкин. —
Казань: Казан. ун-т, 2015. — 23 с.

В пособии рассмотрены алгоритмы сортировки, основанные на модели сортирующей сети. Достоинством сортирующих сетей является то, что они могут выполнять несколько операций одновременно. Это позволяет создавать масштабируемые параллельные алгоритмы.

Для студентов специальности «Фундаментальная информатика и информационные технологии».

Содержание

1 Введение

Алгоритмы сортировки, которые являются реализацией абстракции сортирующей сети, выполняют только две операции — сравнение и обмен местами двух элементов последовательности. Причём они выполняют одни и те же операции для всех входных последовательностей одинаковой длины. Такая независимость порядка последующих операций от результата предыдущих позволяет выполнять несколько операций одновременно или «параллельно». Известно, что последовательные алгоритмы сортировки, основанные на сравнениях, выполняют не менее чем $n \log n$ сравнений. Между тем существуют сортирующие сети, которые решают ту же задачу быстрее, чем за линейное время. В данном пособии рассматривается один из таких алгоритмов, доказывается его корректность. Сортирующие сети описаны в [1, 2]. В [2] рассматривается задача построения сети с минимальным числом сравнений.

На практике длина входной последовательности значительно превосходит число процессоров компьютера. В этом случае можно использовать сортирующую сеть, на входы которой передаются множества. Получившийся в результате алгоритм сортировки называют блочной сортировкой. В пособии рассмотрен пример такого подхода, доказана его корректность. Блочная сортировка описывается в [3, 4].

2 Сравнивающие сети

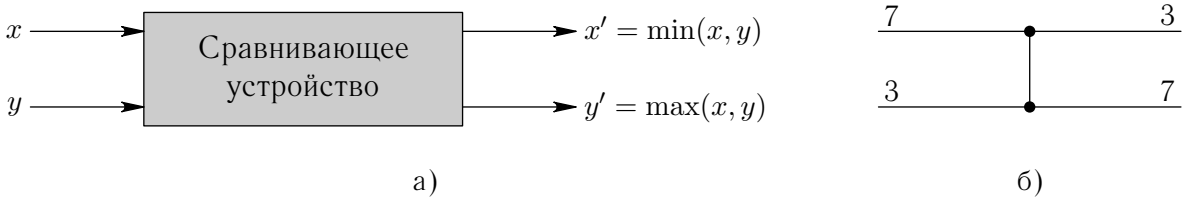
Сравнивающая сеть состоит из сравнивающих устройств и соединяющих их проводов. *Сравнивающее устройство* (comparator, compare-exchange operation) — это устройство с двумя входами, x и y, и двумя выходами, x' и y', выполняющее такую операцию:

$$x' = \min(x, y),$$
$$y' = \max(x, y).$$

Схематическое изображение сравнивающего устройства приведено на рис. 1,а. В дальнейшем условимся обозначать сравнивающее устройство в виде вертикальной черты с точками на концах, как показано на рис. 1,б. Входные величины будем изображать слева, а выходные — справа, при этом меньшая входная величина на выходе переходит вверх, а бóльшая — вниз.

В дальнейшем предполагается, что время, за которое сравнивающее устройство выполняет требуемое действие, равно константе.

Значение передаётся из одной точки в другую по *проводу* (wire). Провода соединяют выход одного сравнивающего устройства с входом другого; в противном случае они являются либо входами сети, либо выходами из неё. Пусть сравнивающая сеть содержит n *входных проводов* (input wires) $a_0, a_1, \ldots, a_{n-1}$, по которым в сеть поступают входные величины, и n *выходных проводов* (output wires) $b_0, b_1, \ldots, b_{n-1}$, по которым выдаются вычисленные сетью результаты. Кроме того, мы будем говорить о *входной последовательности* (input sequence)



а) б)

Рис. 1. а) Схематическое изображения сравнивающего устройства; б) Сравнивающее устройство, для которого входные величины есть $x = 7$ и $y = 3$, а выходные — $x' = 3$ и $y' = 7$

$\langle a_0, a_1, \ldots, a_{n-1}\rangle$ и *выходной последовательности* (output sequence) $\langle b_0, b_1, \ldots, b_{n-1}\rangle$, подразумевая под ними наборы величин во входных и выходных проводах. Таким образом, одно и то же название будет использоваться и для проводов, и для значений, которые по ним передаются. Что именно имеется в виду — всегда будет понятно из контекста.

На рис. 2 показана *сравнивающая сеть* (comparison network), представляющая собой набор сравнивающих устройств, соединённых между собой проводами. Будем изображать сравнивающую сеть на n входов в виде n горизонтальных *линий* (lines) с вертикальными отрезками между ними, представляющими компараторы. Заметим, что линия представляет собой не отдельный провод, а последовательность различных проводов, соединяющих разные сравнивающие устройства. Например, верхней линией на рис. 2 представлены три провода: входной провод a_0, который подсоединён к входу сравнивающего устройства A; провод, соединяющий верхний выход сравнивающего устройства A со входом сравнивающего устройства C; выходной провод b_0, присоединённый к верхнему выходу сравнивающего устройства C. Ко входу каждого сравнивающего устройства подсоединен провод, который либо является одним из n входных проводов $a_0, a_1, \ldots a_{n-1}$, либо присоединён к выходу другого сравнивающего устройства. Аналогично, к выходу каждого сравнивающего устройства подсоединен провод, который либо является одним из n выходных проводов $b_0, b_1, \ldots b_{n-1}$, либо подсоединен к входу другого сравнивающего устройства. Главное требование, которое должно быть удовлетворено при соединении сравнивающих устройств между собой, заключается в том, что получившийся в результате граф соединений должен быть ациклическим: если проследить путь, проходящий от выхода какого-нибудь сравнивающего устройства к входу другого сравнивающего устройства, затем от его выхода к входу следующего сравнивающего устройства и т.д., то этот путь никогда не должен зацикливаться, дважды проходя через одно и то же сравнивающие устройство. Таким образом, как видно из рис. 2, на схеме сравнивающей сети

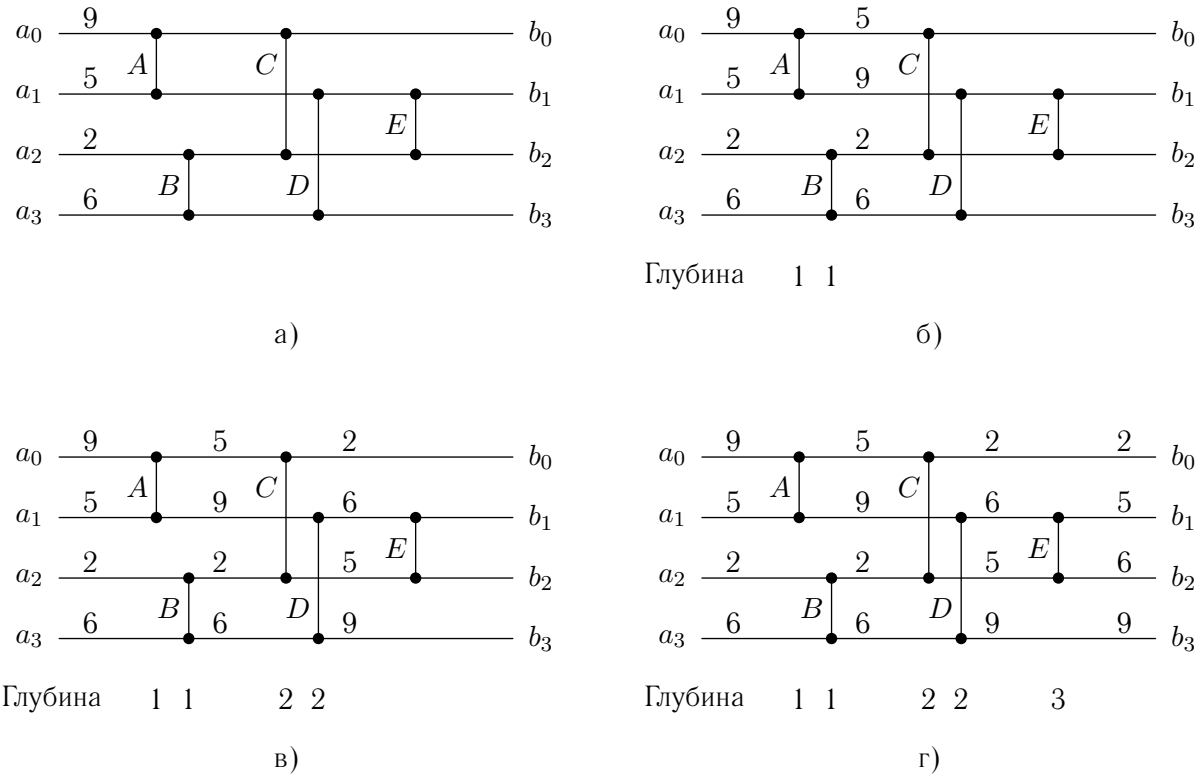


Рис. 2. Движение значений по сравнивающей сети

входы можно расположить слева, а выходы — справа; при этом данные в процессе обработки будут продвигаться слева направо.

Каждое сравнивающие устройство вычисляет выходные значения только после того, как получит оба входных значения. В качестве примера рассмотрим сравнивающую сеть, изображенную на рис. 2,а. Пусть последовательность $\langle 9, 5, 2, 6\rangle$ передаётся на входные провода сети в момент времени 0. В это время только сравнивающие устройства A и B получат по два входных значения. Примем продолжительность интервала времени, в течение которого сравнивающие устройство вычисляет выходные значения, за единицу. Тогда сравнивающие устройства A и B выдают выходные значения в момент времени 1, результат показан на рис. 2,б. Заметим, что сравнивающие устройства A и B выдают свои значения одновременно или «параллельно». Далее, в момент времени 1 сравнивающие устройства C и D, но не E, получили оба входных значения. По прошествии единичного интервала времени, в момент времени 2, они вычислят выходные значения, как показано на рис. 2,в. Сравнивающие устройства C и D также отработают параллельно. К верхнему выходу сравнивающего устройства C и нижнему выходу сравнивающего устройства D подсоединены выходные провода b_0 и b_3, соответственно, поэтому на эти выходы сети значения подаются в момент времени 2. Кроме того, в момент времени 2 сравнивающие устройство E получит свои входные значения. К моменту времени 3 оно вычислит свои выходные значения и отправит их в провода b_1 и b_2. Таким образом, в момент времени 3 выходная последовательность готова, она будет иметь вид $\langle 2, 5, 6, 9\rangle$, как показано на рис. 2,г.

Предполагая, что каждое сравнивающие устройство выполняет своё действие за единицу времени, можно определить «время работы» сравнивающей сети как время с момента подачи значений на входные провода сети до момента появления значений на всех её выходных проводах. Другими словами, это время есть наибольшее число сравнивающих устройств, через которые может пройти значение, двигаясь от входного провода сети к выходному. Определим понятие *глубины* (depth) провода. Входные провода сети имеют глубину 0. Если входные провода сравнивающего устройства имеют глубину d_x и d_y, тогда его выходные провода имеют глубину $\max(d_x, d_y) + 1$. Поскольку сравнивающая сеть не содержит циклов, глубина провода определяется однозначно, и можно определить глубину сравнивающего устройства как глубину его выходных проводов. На рис. 2 показаны глубины сравнивающих устройств сети. Определим глубину сравнивающей сети как максимальную глубину её выходных проводов, или эквивалентно, как максимальную глубину сравнивающих устройств. Сравнивающая сеть на рис. 2 имеет глубину 3, потому что сравнивающие устройство E имеет глубину 3. Если каждое сравнивающие устройство вычисляет свои выходные значения за единицу времени, и сеть получает входные значения в момент времени 0, то сравнивающие устройство глубины d выдаёт выходные значения в момент времени d. Таким образом, глубина сети равна времени, которое нужно сети, для того чтобы выдать значения на все свои выходные провода.

Сортирующая сеть (sorting network) — это сравнивающая сеть, выходная последовательность которой монотонно не убывает (т.е. $b_0 \leqslant b_1 \leqslant \ldots \leqslant b_{n-1}$) для любой входной последовательности. Конечно, не каждая сравнивающая сеть является сортирующей сетью, но сеть на рис. 2 будет такой. Чтобы убедиться в этом, заметим, что в момент времени 1 минимальное из четырёх входных значений окажется либо на верхнем выходе сравнивающего устройства A, либо на верхнем выходе сравнивающего устройства B. Следовательно, в момент времени 2 минимум окажется на верхнем выходе сравнивающего устройства C. Подобные рассуждения показывают, что в момент времени 2 максимум из четырёх входных значений окажется на нижнем выходе сравнивающего устройства D. В момент времени 3 сравнивающие устройство E упорядочит два средних значения, и выходная последовательность станет неубывающей.

Особенностью сравнивающих сетей является то, что последовательность выполняемых

ими операций сравнения не зависит от входных значений, а зависит только от количества входов сети. Поэтому в дальнейшем будут описываться «семейства» сравнивающих сетей. Каждая конкретная сеть из семейства будет задаваться именем семейства и количеством входов (которое равно количеству выходов). Например, если назвать семейство сортирующих сетей `SORTER`, то сортирующая сеть на n входов из этого семейства будет обозначаться `SORTER[n]`.

Алгоритмы сортировки, которые используют только две операции — сравнение и обмен местами двух элементов последовательности, называют обменные сортировки. Сортирующие сети принадлежат к классу обменных сортировок. Следующее утверждение даёт нижнюю оценку сложности обменной сортировки.

Предложение 1. *Сложность по числу сравнений алгоритма обменной сортировки есть величина* $\Omega(n \log n)$.

Доказательство. Выберем произвольный алгоритм обменной сортировки. Пусть при сортировке последовательности длины n этот алгоритм выполняет не более d сравнений. Возьмём входную последовательность длины n, все элементы которой попарно различны (т.е. $i \neq j$ влечёт $a_i \neq a_j$). Исполним алгоритм для каждой из $n!$ перестановок элементов выбранной последовательности; результаты сравнений, которые выполняет алгоритм, запишем в виде последовательности из нулей и единиц, длина такой последовательности будет не больше d. Если, обрабатывая какую-то перестановку, алгоритм выполнит меньше d сравнений, дополним получившуюся последовательность нулями, так чтобы её длина стала равной d. Таким образом, мы получим $n!$ последовательностей из d нулей и единиц. Все эти последовательности разные. Действительно, в противном случае алгоритм выполнил бы одни и те же же обмены для различных входных последовательностей, и один из двух полученных результатов был бы неверен. Поэтому справедливо $2^d \geqslant n!$.

Из полученного неравенства следует, что $d \geqslant \log_2 n!$. Используя свойство логарифмов, запишем

$$\log_2 n! = \log_2 1 + \log_2 2 + \ldots + \log_2 n.$$

Вторая половина слагаемых не меньше $\log_2(n/2) = \log_2 n - 1$ каждое. Следовательно, $\log_2 n! \geqslant \geqslant Cn \log_2 n$ для некоторой константы C. □

Количество сравнивающих устройств в сети называют её *размером* (size). Из предложения 1 вытекают оценки для размера и глубины сортирующей сети.

Следствие. *Размер сортирующей сети есть величина* $\Omega(n \log n)$. *Глубина сортирующей сети есть величина* $\Omega(\log n)$.

Доказательство. Оценка для размера справедлива, так как сравнивающее устройство выполняет только одно сравнение. Оценка для глубины следует из того, что сортирующая сеть на n входов может выполнять одновременно не более $\lfloor n/2 \rfloor$ сравнений. □

Существуют как программные, так и аппаратные реализации абстракции сортирующей сети. Рассмотрим возможные программные реализации, предполагая, что входная последовательность хранится в виде массива в оперативной памяти компьютера. Сравнивающее устройство может быть представлено функцией. Аргументами этой функции будут индексы двух элементов массива, которые передаются на входы сравнивающего устройства. Первый подход состоит в том, чтобы последовательно выполнить сравнивающие устройства глубины 1, затем — сравнивающие устройства глубины 2 и т.д. Получится некоторый список вызовов функции, моделирующей сравнивающее устройство. Сложность полученного алгоритма будет величиной одного порядка с размером сортирующей сети.

Второй подход предполагает, что у нас есть компьютер с несколькими независимыми процессорами, которые могут выполнять операции параллельно. Пусть все процессоры имеют доступ к общей памяти, где хранится входная последовательность, и число процессоров не

меньше $\lfloor n/2 \rfloor$. Тогда возможно исполнять все сравнивающие устройства одинаковой глубины одновременно. Говорят, что они выполняются за один *параллельный шаг* (parallel stage). Сложность полученного параллельного алгоритма будет величиной одного порядка с глубиной сортирующей сети.

3 Нуль-единичный принцип

Нуль-единичный принцип (zero-one principle) утверждает, что если сеть сортирует в порядке неубывания все последовательности из нулей и единиц, то она будет сортировать в том же порядке произвольную входную последовательность чисел (эти числа могут быть целыми, вещественными или, в общем случае, быть элементами некоторого линейно упорядоченного множества). Нуль-единичный принцип значительно упрощает задачу доказательства корректности работы сортирующей сети. Построив сортирующую сеть и доказав, что она сортирует все последовательности из нулей и единиц, можно обратиться к нуль-единичному принципу и заключить, что сеть будет правильно сортировать любую входную последовательность.

Функция f называется монотонно неубывающей, если для любых x, y из области определения $x \leqslant y$ влечёт $f(x) \leqslant f(y)$. Для доказательства нуль-единичного принципа потребуется следующая лемма.

Лемма 1. *Если сравнивающая сеть преобразует входную последовательность* $a = \langle a_0, a_1, \ldots, a_{n-1} \rangle$ *в выходную последовательность* $b = \langle b_0, b_1, \ldots, b_{n-1} \rangle$*, то для любой монотонно неубывающей функции* f*, эта сеть преобразует входную последовательность* $f(a) = \langle f(a_0), f(a_1), \ldots, f(a_{n-1}) \rangle$ *в выходную последовательность* $f(b) = \langle f(b_0), f(b_1), \ldots, f(b_{n-1}) \rangle$.

Доказательство. Сначала докажем, что если функция f монотонно неубывающая, то сравнивающее устройство, получая на входы $f(x)$ и $f(y)$, выдаёт на выходы $f(\min(x, y))$ и $f(\max(x, y))$.

Чтобы доказать сформулированное утверждение, рассмотрим сравнивающее устройство, чьи входные значения есть x и y. Тогда на верхнем выходе окажется $\min(x, y)$, а на нижнем — $\max(x, y)$. Если же передать на входы сравнивающего устройства $f(x)$ и $f(y)$, то на верхнем выходе окажется $\min(f(x), f(y))$, а на нижнем — $\max(f(x), f(y))$ (рис. 3). Из монотонного неубывания функции f следуют тождества

$$\min(f(x), f(y)) = f(\min(x, y)),$$
$$\max(f(x), f(y)) = f(\max(x, y)).$$

Таким образом, сравнивающее устройство выдаёт значения $f(\min(x, y))$ и $f(\max(x, y))$, если получает на входы $f(x)$ и $f(y)$, доказательство утверждения закончено.

Используя индукцию по глубине провода, можно доказать утверждение более сильное, чем утверждение леммы: если некоторый провод получает значение a_i, когда входной последовательностью сети является a, то он получит значение $f(a_i)$, когда входной последовательностью сети будет $f(a)$. Поскольку это утверждение справедливо для всех проводов сети, в том числе и для выходных, его доказательство станет доказательством леммы.

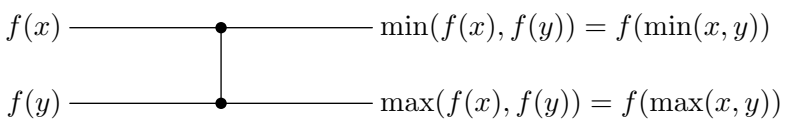


Рис. 3. Изображение сравнивающего устройства из доказательства леммы 1

Для доказательства базиса индукции выберем провод глубины 0, т.е. некоторый входной провод a_i. В этом случае утверждение очевидно: при подаче входной последовательности $f(a)$ выбранный провод получит значение $f(a_i)$. Для доказательства шага индукции выберем провод глубины d, где $d > 0$. Выбранный провод присоединён к выходу сравнивающего устройства глубины d, входные провода которого имеют глубину строго меньшую d. Из предположения индукции следует, что если это сравнивающее устройство получает на входы значения a_i и a_j, когда входной последовательностью сети является a, то оно получит значения $f(a_i)$ и $f(a_j)$, когда входной последовательностью будет $f(a)$. Согласно доказанному ранее утверждению, на выходы рассматриваемого сравнивающего устройства поступят значения $f(\min(a_i, a_j))$ и $f(\max(a_i, a_j))$. Это же сравнивающее устройство выдаст $\min(a_i, a_j)$ и $\max(a_i, a_j)$, когда входной последовательностью будет a, лемма доказана. □

В качестве примера, иллюстрирующего утверждение леммы 1, возьмём сеть изображенную на рис. 2 (повторно изображена на рис. 4,а) и применим к входной последовательности монотонно неубывающую функцию $f(x) = \lceil x \rceil$. Результат изображен на рис. 4,б, можно видеть, что значение, передаваемое каждым проводом сети, получается применением функции f к значению, передаваемому тем же самым проводом на рис. 4,а.

Если сравнивающая сеть является сортирующей, то лемма 1 позволяет доказать следующее утверждение.

Теорема 1 (Нуль-единичный принцип). *Если сравнивающая сеть на n входов правильно сортирует все 2^n последовательностей из нулей и единиц, то она будет правильно сортировать любую последовательность n чисел.*

Доказательство. Предположим противное: некоторая сеть сортирует все последовательности из нулей и единиц, но существует последовательность чисел, которую сеть не может отсортировать правильно. А именно, существует последовательность $\langle a_0, a_1, \ldots, a_{n-1} \rangle$, содержащая элементы a_i и a_j, такие что $a_i < a_j$, но сеть размещает a_j перед a_i в выходной последовательности. Определим монотонно неубывающую функцию f следующим образом

$$f(x) = \begin{cases} 0 & \text{при } x \leqslant a_i, \\ 1 & \text{при } x > a_i. \end{cases}$$

Поскольку сеть размещает a_j перед a_i в выходной последовательности, когда на вход подаётся последовательность $\langle a_0, a_1, \ldots, a_{n-1} \rangle$, из леммы 1 следует, что она разместит $f(a_j)$ перед $f(a_i)$ в выходной последовательности, когда на вход будет подаваться последовательность $\langle f(a_0), f(a_1), \ldots, f(a_{n-1}) \rangle$. Но $f(a_j) = 1$ и $f(a_i) = 0$, а входная последовательность $\langle f(a_0), f(a_1), \ldots, f(a_{n-1}) \rangle$ состоит из нулей и единиц. Получено противоречие. □

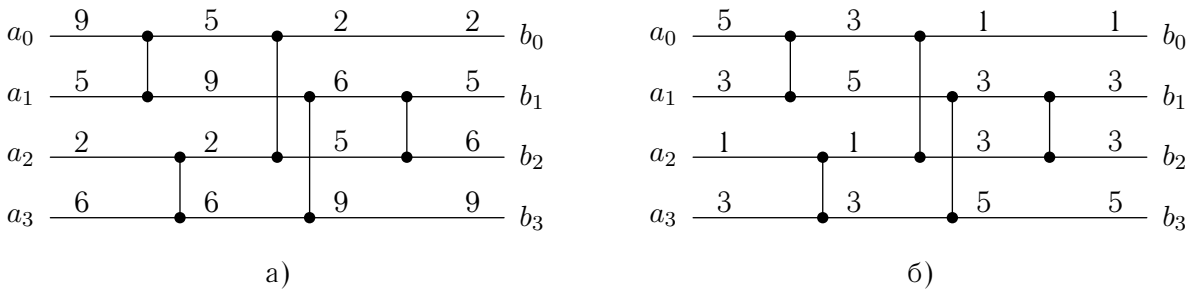


Рис. 4. а) Сортирующая сеть с входной последовательностью $\langle 9, 5, 2, 6 \rangle$; б) Та же сеть, к входной последовательности которой применена функция $f(x) = \lceil x \rceil$

4 Чётно-нечётная сортировка с обменами

Первой из рассмотренных нами сортирующих сетей будет *сеть чётно-нечётной сортировки с обменами* (odd-even transposition sort). Сформулируем правило, по которому строится сортирующая сеть.

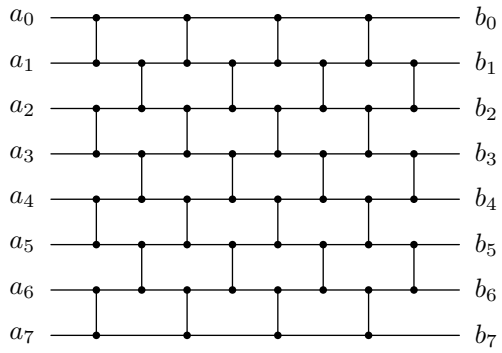
Алгоритм 1. *Сеть на n входов выполняет n параллельных шагов. Если $i = 0, 1, \ldots, n-1$ — это номер линии, а $d = 1, 2, \ldots, n$ — глубина, то линия i на глубине d соединяется сравнивающим устройством с линией $j = i - (-1)^{i+d}$, когда $0 \leqslant j < n$.*

Назовем семейство сортирующих сетей, построенных алгоритмом 1, `ODD_EVEN_SORTER`. На рис. 5,а изображена сеть на 8 входов, принадлежащая этому семейству. На рис. 5,б показана обработка сетью входной последовательности, на каждом шаге дугой соединены элементы, которые сравниваются между собой.

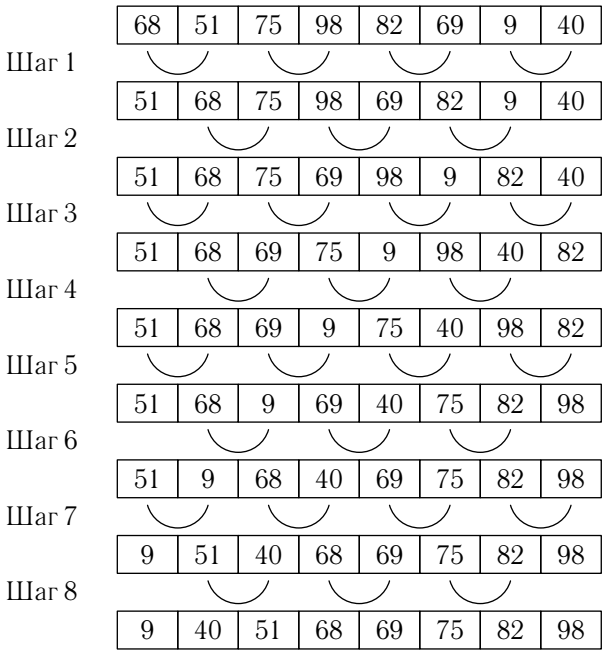
Можно видеть, что на нечётном шаге сеть сравнивает каждый чётный элемент последовательности с нечётным, стоящим после него. В то же время на чётном шаге каждый нечётный элемент сравнивается со стоящим после него чётным элементом. Таким образом, алгоритм использует только два действия, которые выполняются поочередно. Это обуславливает главное достоинство описанной сортировки — исключительную простоту аппаратной реализации.

Теорема 2. *Сеть, построенная алгоритмом 1, является сортирующей сетью.*

Доказательство. В силу нуль-единичного принципа достаточно показать, что сеть сортирует все последовательности из нулей и единиц. Докажем утверждение индукцией по числу входов сети. Возьмём $n = 1$, сеть с 1 входом, построенная алгоритмом 1, не содержит сравнивающих устройств. Она состоит из одного входного провода, который является и выходным. Так как последовательность длины 1 уже упорядочена, утверждение теоремы верно. Пусть теперь $n > 1$. Возможны два случая.



а)



б)

Рис. 5. а) Сеть `ODD_EVEN_SORTER[8]`; б) Исполнение алгоритма чётно-нечётной сортировки с обменами

Случай 1: $a_{n-1} = 1$. В этом случае сравнивающие устройства, связывающие линии $n-2$ и $n-1$, никогда не выполняют обменов, поэтому единица пройдет по линии $n-1$ и окажется на нижнем выходе сети. Сравнивающие устройства глубины $1, 2, \ldots, n-1$, связывающие линии $0, 1, \ldots, n-2$, образуют сеть чётно-нечётной сортировки с обменами на $n-1$ вход. По предположению индукции эта сеть является сортирующей. Она отсортирует последовательность $\langle a_0, a_1, \ldots, a_{n-2}\rangle$ за $n-1$ параллельный шаг. Так как на линии $n-1$ находится 1, вся входная последовательность длины n будет отсортирована через $n-1$ шаг. Сравнивающие устройства глубины n никогда не будут выполнять обмены. Рис. 6 иллюстрирует рассмотренный случай, избыточные сравнивающие устройства (т.е. те, которые никогда не выполняют обменов) обведены по контуру.

Случай 2: $a_{n-1} = 0$. Все сравнивающие устройства на вход которых передаётся значение a_{n-1} выполняют обмен (если на входы сравнивающего устройства передаются два нуля и обмен не выполняется, можно считать, что обмен выполняется, так как выходные значения будут те же самые). На рис. 7,а указанные сравнивающие устройства обведены по контуру. Можно построить эквивалентную сравнивающую сеть, заменив их одним проводом, который передаёт значение $a_{n-1} = 0$ на верхний выход сети (рис. 7,б). Если исключить из рассмотрения добавленный провод, то останется сеть на $n-1$ вход глубины $n-1$. Эта сеть будет чётно-нечётной сортировкой с обменами (рис. 7,в). По предположению индукции она отсортирует входную последовательность $\langle a_0, a_1, \ldots, a_{n-2}\rangle$. Таким образом, на выходе получим нуль и отсортированную последовательность длины $n-1$ после него, т.е. вся выходная последовательность будет отсортирована. □

Следующее утверждение даёт оценку размера и глубины сети чётно-нечётной сортировки с обменами.

Предложение 2. *Размер сети, построенной алгоритмом 1, есть величина* $\Theta(n^2)$*, её глубина есть величина* $\Theta(n)$.

Доказательство. Выведем формулу для размера $S(n)$ сети `ODD_EVEN_SORTER[n]`. Если n чётное, то сеть выполнит $n/2$ нечётных шагов, $n/2$ сравнений в каждом, и $n/2$ чётных шагов, $(n-2)/2$ сравнений в каждом. Общее число сравнений будет

$$S(n) = \frac{n}{2} \cdot \frac{n}{2} + \frac{n}{2} \cdot \frac{n-2}{2} = \frac{n(n-1)}{2}.$$

Если n нечётное, то сеть выполнит $(n+1)/2$ нечётных шагов, $(n-1)/2$ сравнений в каждом, и $(n-1)/2$ чётных шагов, $(n-1)/2$ сравнений в каждом. Общее число сравнений будет

$$S(n) = \frac{n+1}{2} \cdot \frac{n-1}{2} + \frac{n-1}{2} \cdot \frac{n-1}{2} = \frac{n(n-1)}{2}.$$

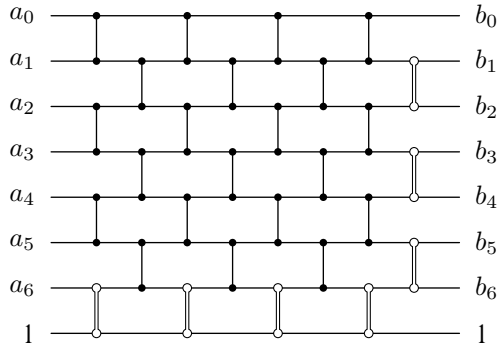


Рис. 6. Иллюстрация случая 1

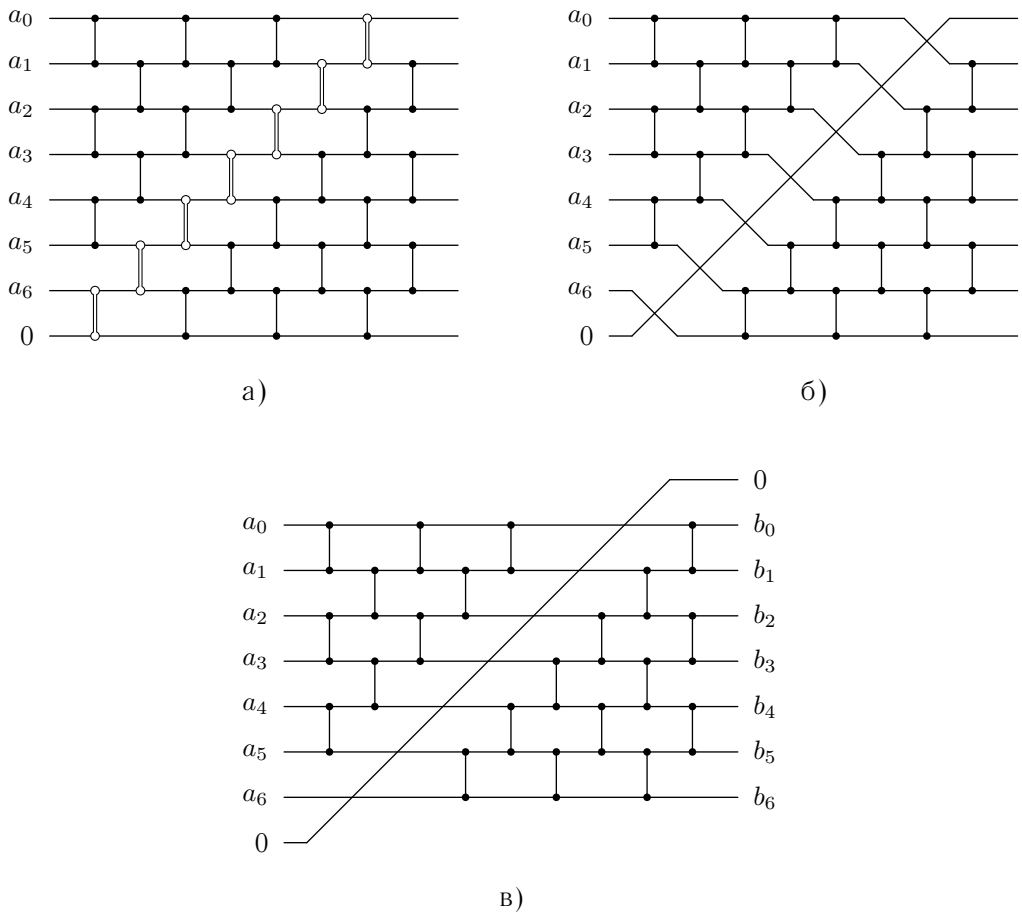


Рис. 7. Иллюстрация случая 2

Получаем, что в обоих случаях $S(n) = n(n-1)/2 \in \Theta(n^2)$.

Глубина сети $D(n)$ равна 0 при $n = 1$, $D(2) = 1$, $D(n) = n$ при $n > 2$. Следовательно, $D(n) \in \Theta(n)$. □

Сортирующая сеть, каждое сравнивающее устройство которой соединяет соседние линии, называется *простейшей*. Сеть чётно-нечётной сортировки с обменами будет простейшей сетью. Следующее утверждение даёт нижнюю оценку для размера простейшей сети.

Предложение 3. *Размер простейшей сети есть величина* $\Omega(n^2)$.

Доказательство. Говорят, что элементы последовательности a_i и a_j образуют *инверсию*, если $i < j$, но $a_i > a_j$. Отсортированная последовательность не содержит инверсий. Обменивая два соседних элемента последовательности, можно уничтожить не более одной инверсии. Следовательно, число обменов простейшей сети не меньше числа инверсий входной последовательности. Для монотонно убывающей входной последовательности (т.е. $a_0 > a_1 > \ldots > a_{n-1}$) число инверсий будет $C_n^2 = n(n-1)/2$. □

Из предложения 3 следует, что невозможно построить простейшую сеть, оценка числа сравнений которой лучше, чем у алгоритма 1.

5 Чётно-нечётная сортировка слиянием

Сеть чётно-нечётной сортировки слиянием (odd-even mergesort) основана на алгоритме слияния. Сравнивающая сеть называется *сливающей сетью* (merging network), ес-

ли, получая на вход последовательность, две половины которой отсортированы, она выдаёт отсортированную выходную последовательность. На протяжении этого раздела будем предполагать, что число входов сети n является степенью двойки (тогда $n/2$ всегда будет целым числом). Данное предположение существенно упрощает рассуждения. Обобщение алгоритма на случай произвольного n приводится в разделе 7. Сформулируем правило, по которому строится сеть чётно-нечётного слияния.

Алгоритм 2. *Если $n = 2$, разместить одно сравнивающее устройство между линиями 0 и 1. Иначе взять последовательности $\langle a_0, a_2, \ldots, a_{n-2}\rangle$ (передаётся по чётным линиям) и $\langle a_1, a_3, \ldots, a_{n-1}\rangle$ (передаётся по нечётным линиям). Для каждой из них применить алгоритм рекурсивно. После этого разместить сравнивающие устройства между линиями i и $i + 1$ для всех $i = 1, 3, \ldots, n - 3$.*

Назовём семейство сетей, построенных алгоритмом 2, MERGER. На рис. 8 изображены сети из этого семейства на 4, 8 и 16 входов. С левой стороны показано как строится сеть. Сети справа получены из сетей, изображённых слева, путём группирования сравнивающих устройств одинаковой глубины.

Докажем утверждение подобное нуль-единичному принципу для сливающих сетей.

Лемма 2. *Если сравнивающая сеть правильно выполняет слияние любых двух отсортированных последовательностей из нулей и единиц, то она правильно выполнит слияние любых двух отсортированных последовательностей чисел.*

Доказательство. Повторяет доказательство теоремы 1. Предположим противное. Возьмём последовательность чисел, которая нарушает утверждение леммы. Применим для её членов функцию f, определённую в доказательстве теоремы 1. Получим последовательность из нулей и единиц. В силу монотонности f две половины полученной последовательности отсортированы. Сеть не может обработать её правильно. Получено противоречие. □

Лемма 3. *Сеть, построенная алгоритмом 2, является сливающей сетью.*

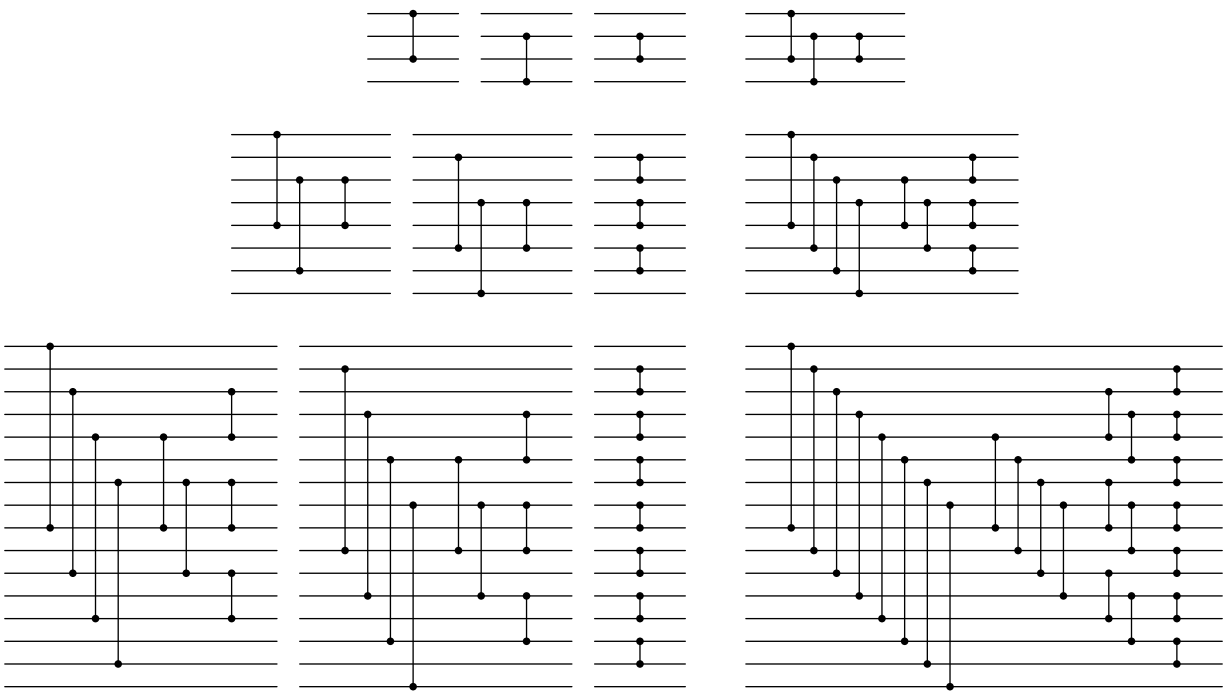


Рис. 8. Рекурсивное построение сетей семейства MERGER

Доказательство. В силу леммы 2 достаточно проверить утверждение для последовательностей из нулей и единиц. Докажем утверждение индукцией по числу входов сети. Для $n = 2$ утверждение очевидно. Возьмём $n > 2$. Пусть последовательность $\langle a_0, a_1, \ldots, a_{n/2-1}\rangle$ состоит из k нулей, за которыми следуют $n/2 - k$ единиц, а последовательность $\langle a_{n/2}, a_{n/2-1}, \ldots, a_{n-1}\rangle$ — из l нулей с последующими $n/2 - l$ единицами. Обозначим последовательность, которая получается после рекурсивного применения двух копий сети на $n/2$ входов, $\langle s_0, s_1, \ldots, s_{n-1}\rangle$. По предположению индукции сеть на $n/2$ входов правильно выполняет слияние. Тогда последовательность $\langle s_0, s_2, \ldots, s_{n-2}\rangle$ будет состоять из $\lceil k/2\rceil + \lceil l/2\rceil$ нулей с последующими единицами, а $\langle s_1, s_3, \ldots, s_{n-1}\rangle$ — из $\lfloor k/2\rfloor + \lfloor l/2\rfloor$ нулей с последующими единицами. Ключевым моментом доказательства является то, что разность

$$(\lceil k/2\rceil + \lceil l/2\rceil) - (\lfloor k/2\rfloor + \lfloor l/2\rfloor)$$

может принимать значения 0, 1 или 2. Если она равна 0 или 1, то последовательность $\langle s_0, s_1, \ldots, s_{n-1}\rangle$ уже отсортирована. Если же она равна 2, то одно из сравнивающих устройств на последнем шаге выполнит обмен, который сделает последовательность отсортированной. □

Следующее утверждение даёт оценку размера и глубины сливающей сети.

Лемма 4. *Размер сети, построенной алгоритмом 2, есть величина* $\Theta(n \log n)$*, её глубина есть величина* $\Theta(\log n)$.

Доказательство. Из рекурсивного правила построения сети следует, что размер $S(n)$ сети `MERGER[n]` равен сумме удвоенного размера сети `MERGER[n/2]` и числа сравнений на последнем шаге. Поэтому $S(n)$ выражается рекуррентным уравнением

$$S(n) = \begin{cases} 1 & \text{при } n = 2, \\ 2S(n/2) + n/2 - 1 & \text{при } n = 2^k, k > 1, \end{cases}$$

решение которого имеет вид $S(n) = (n \log_2 n)/2 - n/2 + 1 \in \Theta(n \log n)$. Проверим справедливость формулы по индукции. Для $n = 2$ формула верна. Возьмём $n > 2$. По предположению индукции формула будет верна для $n/2$. Отсюда получаем

$$S(n) = 2S(n/2) + n/2 - 1 = 2\left(\frac{n}{2} \cdot \frac{\log_2 n - 1}{2} - \frac{n}{4} + 1\right) + \frac{n}{2} - 1 = \frac{n \log_2 n}{2} - \frac{n}{2} + 1.$$

Две копии сети `MERGER[n/2]` выполняются параллельно, поэтому глубина $D(n)$ сети `MERGER[n]` на единицу больше глубины сети `MERGER[n/2]`. Для $D(n)$ получаем рекуррентное уравнение

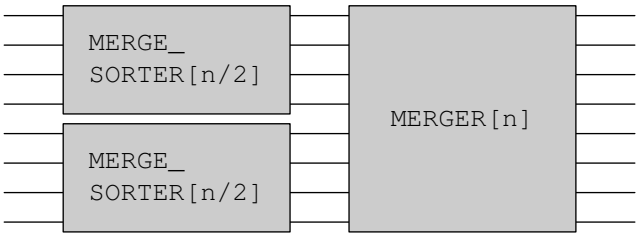
$$D(n) = \begin{cases} 1 & \text{при } n = 2, \\ D(n/2) + 1 & \text{при } n = 2^k, k > 1, \end{cases}$$

решение которого имеет вид $D(n) = \log_2 n$. Справедливость формулы проверяется по индукции. □

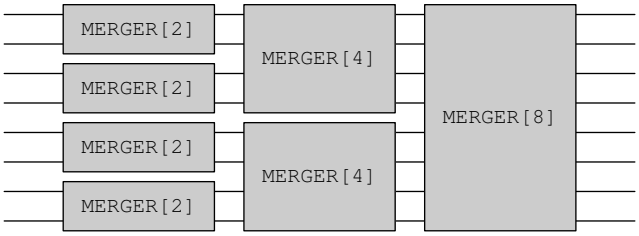
Теперь можно построить сортирующую сеть с помощью слияния.

Алгоритм 3. *Если* $n = 1$ *сеть не содержит сравнивающих устройств. Иначе применить алгоритм рекурсивно для последовательностей* $\langle a_0, a_1, \ldots, a_{n/2-1}\rangle$ *и* $\langle a_{n/2}, a_{n/2+1}, \ldots, a_{n-1}\rangle$*. После этого разместить сеть чётно-нечётного слияния на* n *входов.*

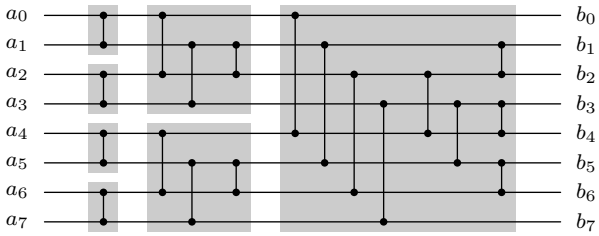
Назовём семейство сетей, построенных алгоритмом 3, `MERGE_SORTER`. На рис. 9,а показано рекурсивное построение сети `MERGE_SORTER[n]`. Две половины входной последовательности длины n сортируются (параллельно) копиями сети `MERGE_SORTER[n/2]`. Затем отсортированные половины сливаются сетью `MERGER[n]`. Выход из рекурсии произойдёт при $n = 1$. В этом случае достаточно одного провода, так как последовательность длины 1 уже отсортирована. На рис. 9,б показан порядок рекурсивных вызовов для $n = 8$. На



а)



б)



в)

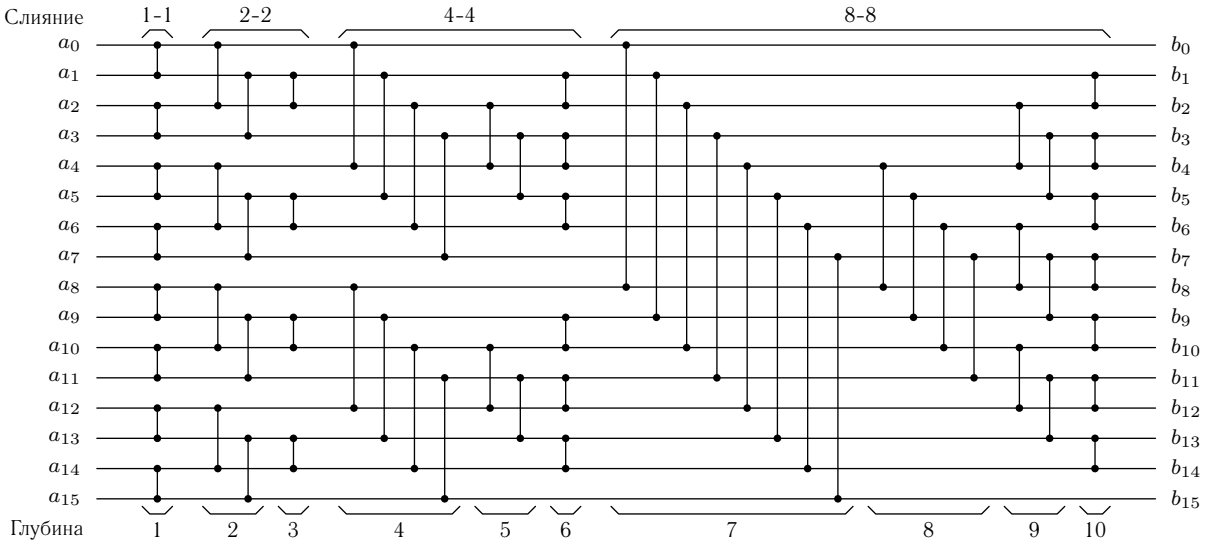
Рис. 9. Построение сети MERGE_SORTER

рис. 9,в условные обозначения сети MERGER заменены настоящими схемами, построена сеть MERGE_SORTER[8].

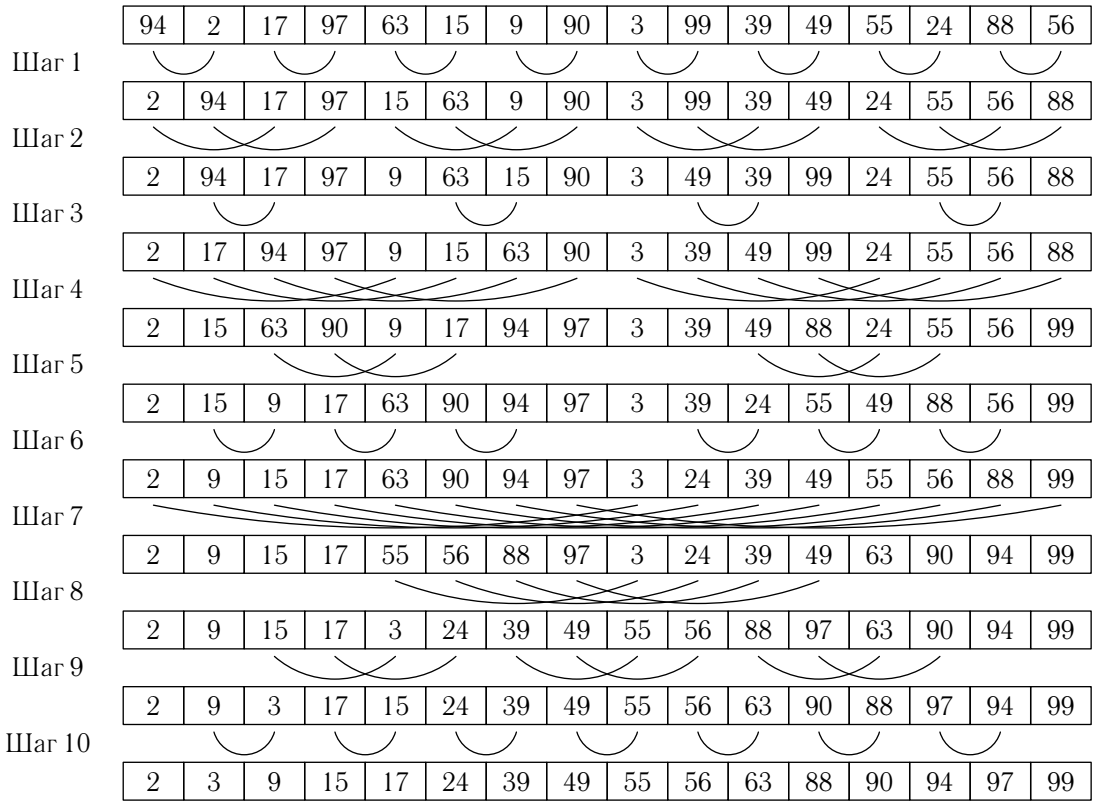
В сети MERGE_SORTER[n] данные проходят $\log_2 n$ этапов, которые будем называть *фазы слияния*. Можно считать, что входная последовательность состоит из n отсортированных последовательностей длины 1. На первом этапе данные, двигаясь по сети, проходят через $n/2$ копий сети MERGER[2], которые работают параллельно. После этого данные представляют из себя отсортированные последовательности длины 2. Будем говорить, что на данном этапе выполнено слияние 1-1. На втором этапе данные проходят через $n/4$ копий сети MERGER[4], которые сливают отсортированные последовательности длины 2 в отсортированные последовательности длины 4 (слияние 2-2). В общем случае на этапе k, где $k = 1, 2, \ldots, \log_2 n$, $n/2^k$ копий сети MERGER[2^k] выполняют слияние отсортированных последовательностей длины 2^{k-1} в отсортированные последовательности длины 2^k. На последнем этапе получается отсортированная последовательность длины n. На рис. 10,а изображена сеть MERGE_SORTER на 16 входов, указана глубина сравнивающих устройств сети, приведено разбиение на фазы слияния. На рис. 10,б показана обработка сетью входной последовательности.

Теорема 3. *Сеть, построенная алгоритмом 3, является сортирующей сетью.*

Доказательство. Докажем утверждение индукцией по числу входов сети. Для $n = 1$ утверждение очевидно. Возьмём $n > 2$. По предположению индукции сеть правильно отсортирует две половины входной последовательности. Тогда утверждение теоремы следует из леммы 3. □



а)



б)

Рис. 10. а) Сеть `MERGE_SORTER[16]`; б) Исполнение алгоритма чётно-нечётной сортировки слиянием

Следующее утверждение даёт оценку размера и глубины сети чётно-нечётной сортировки слиянием.

Предложение 4. *Размер сети, построенной алгоритмом 3, есть величина* $\Theta(n \log^2 n)$, *её глубина есть величина* $\Theta(\log^2 n)$.

Доказательство. Размер $S(n)$ сети `MERGE_SORTER[n]` равен сумме удвоенного размера сети `MERGE_SORTER[n/2]` и размера сети `MERGER[n]`. Поэтому $S(n)$ выражается рекуррентным уравнением

$$S(n) = \begin{cases} 0 & \text{при } n = 1, \\ 2S(n/2) + (n \log_2 n)/2 - n/2 + 1 & \text{при } n = 2^k, k \geqslant 1, \end{cases}$$

решение которого имеет вид $S(n) = n \log_2 n(\log_2 n - 1)/4 + n - 1 \in \Theta(n \log^2 n)$.

Глубина $D(n)$ сети `MERGE_SORTER[n]` равна сумме глубины сети `MERGE_SORTER[n/2]` и глубины сети `MERGER[n]`. Для $D(n)$ получаем рекуррентное уравнение

$$D(n) = \begin{cases} 0 & \text{при } n = 1, \\ D(n/2) + \log_2 n & \text{при } n = 2^k, k \geqslant 1, \end{cases}$$

решение которого имеет вид $D(n) = \log_2 n(\log_2 n + 1)/2 \in \Theta(\log^2 n)$. Справедливость обеих формул можно проверить по индукции. □

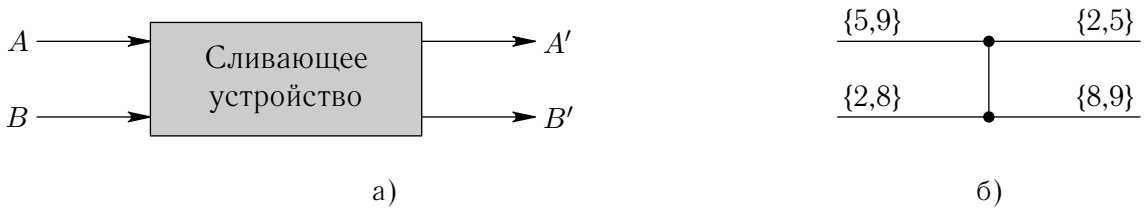
Сеть чётно-нечётной сортировки слиянием была разработана Бэтчером (K.E. Batcher) [5] в 1960-х. Чётно-нечётное слияние оптимально по числу сравнений, и некоторое время считалось, что основанная на нём сортировка тоже является оптимальной. В 1983 году Айтаи (Ajtai), Комлош (Comlós) и Шемериди (Szemerédi) [6] разработали сеть, которая выполняет сортировку за $O(n \log n)$ сравнений. Но константы скрытые в O-обозначении настолько велики (несколько тысяч), что эта сеть не имеет практического применения. Сеть чётно-нечётной сортировки слиянием не оптимальна, но она является одним из самых эффективных практических алгоритмов.

6 Блочная сортировка

Возьмём две отсортированные последовательности из m чисел. Обозначим их A и B. Последовательности могут содержать повторяющиеся элементы. Множества, которые содержат повторяющиеся элементы называю *мультимножества*. Операция сложения для мультимножеств определяется следующим образом: если какой-то элемент входит в одно множество k раз, а в другое l раз, то в сумму он будет входить $k + l$ раз. Обозначается эта операция $\uplus$. Например, если $A = \{1, 3\}$, а $B = \{1, 7\}$, то $A \uplus B = \{1, 1, 3, 7\}$. Будем писать $A \leqslant B$, если любой элемент последовательности A меньше или равен любого элемента последовательности B.

Определим понятие сливающего устройства. *Сливающее устройство* (merging comparator, compare-split operation) — это устройство, которое получает на входе две отсортированные последовательности из m чисел, A и B, и выдаёт на выходе две отсортированные последовательности из m чисел, A' и B', причём $A \uplus B = A' \uplus B'$ и $A' \leqslant B'$. Схематическое изображение сливающего устройства приведено на рис. 11,а. Таким образом, на верхнем выходе окажутся наименьшие m, а на нижнем выходе наибольшие m из $2m$ входных элементов. Будем изображать сливающее устройство также как сравнивающее устройство — вертикальной чертой с точками на концах (рис. 11,б). Можно реализовать сливающее устройство, выполнив слияние переданных на его входы последовательностей, а затем передав первую половину результата на верхний выход, вторую — на нижний.

Соединяя сливающие устройства проводами, можно строить сети, так же как строились сети из сравнивающих устройств. По линиям таких сетей будут передаваться мультимножества. Пример для $m = 2$ изображён на рис. 12.



а) б)

Рис. 11. а) Схематическое изображение сливающего устройства; б) Сливающее устройство, для которого входные последовательности есть $A = \{5, 9\}$ и $B = \{2, 8\}$, а выходные — $A' = \{2, 5\}$ и $B' = \{8, 9\}$

Пусть у нас есть компьютер с p независимыми процессорами, которые могут выполнять операции параллельно. Нужно отсортировать входную последовательность длины n, которая хранится в общей памяти компьютера. Обычно n значительно больше p. В этом случае можно использовать алгоритм *блочной сортировки* (block sort). Далее внутри этого раздела будем предполагать, что n кратно p. Обобщение алгоритма на случай последовательности произвольной длины проводится в разделе 7.

Алгоритм 4. *Разделить входную последовательность на p блоков длины n/p, отсортировать каждый блок, после этого применить сортирующую сеть на p входов, составленную из сливающих устройств.*

На рис. 13 показана обработка алгоритмом блочной сортировки той же входной последовательности, что и в примере на рис. 10,б. При этом $p = 8$, а в качестве сортирующей сети выбрана сеть чётно-нечётной сортировки слиянием.

Обоснуем корректность алгоритма. Для этого потребуется ввести некоторые обозначения. Будем обозначать сравнивающие сети греческими буквами. Например, если сеть на n входов, названная α, принимает на вход последовательность a, то выходную последовательность будем обозначать $\alpha(a)$. Сравнивающее (или сливающее) устройство, связывающее линии i и j, будем обозначать $[i, j]$, где $0 \leqslant i < n$, $0 \leqslant j < n$ и $i < j$. Если α имеет размер r, то сеть можно определить, указав порядок выполнения сравнивающих устройств $\alpha = [i_1, j_1], [i_2, j_2], \ldots, [i_r, j_r]$ (устройства одинаковой глубины можно записывать в произвольном порядке). Например, сеть, изображённую на рис. 12, можно записать $[0, 1], [2, 3], [0, 2], [1, 3], [1, 2]$. Справедлива следующая лемма.

Лемма 5. *Пусть α — сортирующая сеть на n входов. Построим сеть, заменив сравнивающие устройства сливающими. Тогда для любой последовательности $a = \langle A_0, A_1, \ldots, A_{n-1}\rangle$, каждый элемент которой — мультимножество из m элементов, выполняется $\alpha(a)_0 \leqslant \alpha(a)_1 \leqslant \ldots \leqslant \alpha(a)_{n-1}$.*

Доказательство. Сформулируем и докажем утверждение обратное противоположному: если существует последовательность мультимножеств x длины n, такая что для некоторых i, j выполняется $i < j$, но нарушено условие $\alpha(x)_i \leqslant \alpha(x)_j$, то найдется последовательность y из нулей и единиц, такая что $\alpha(y)_i = 1$, $\alpha(y)_j = 0$.

Пусть u — наименьший элемент среди $\alpha(x)_j$. Будем сопоставлять некоторой последовательности мультимножеств последовательность из нулей и единиц так, чтобы соблюдалось условие: если элементу сопоставлен нуль, то он содержит значение меньшее или равное u, а если сопоставлена единица, то он содержит значение большее u. Обозначим это условие $(*)$. Сначала построим последовательность из нулей и единиц, удовлетворяющую условию $(*)$, для выходной последовательности $\alpha(x)$. Обозначим её $y^{(0)}$. Можно выбрать $y_i^{(0)} = 1$ и $y_j^{(0)} = 0$, не нарушив $(*)$.

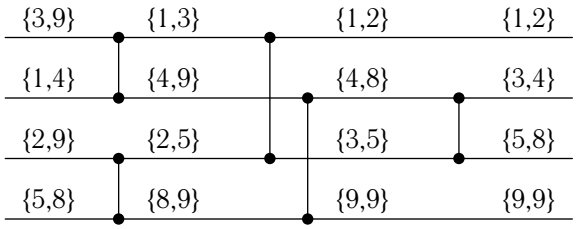


Рис. 12. Сеть, построенная из сливающих устройств

Пусть размер сети α есть r, тогда обозначим через β сеть размера $r-1$, полученную из α удалением последнего сливающего устройства, т.е. $\alpha = \beta, [i_r, j_r]$. Выходной последовательности $\beta(x)$ можно сопоставить нуль-единичную последовательность $y^{(1)}$, так чтобы выполнялось условие $(*)$ и выполнялось дополнительное условие: применение $[i_r, j_r]$ к $y^{(1)}$ давало $y^{(0)}$. Покажем, как построить последовательность $y^{(1)}$. Элементы $y^{(1)}_k = y^{(0)}_k$ при k отличном от i_r и j_r. Теперь выберем значения $y^{(1)}_{i_r}$ и $y^{(1)}_{j_r}$. Пусть сливающее устройство $[i_r, j_r]$ получает на вход последовательности S и T, а выдаёт S' и T'. Для элементов $y^{(0)}_{i_r}$ и $y^{(0)}_{j_r}$ возможны три комбинации значений. Комбинация $y^{(0)}_{i_r} = 1$ и $y^{(0)}_{j_r} = 0$ невозможна. Действительно, тогда S' содержит элемент больший u, а T' содержит элемент меньший или равный u, но $S' \leqslant T'$. Рассмотрим три возможных случая.

Случай 1: $y^{(0)}_{i_r} = 0$ и $y^{(0)}_{j_r} = 0$. В силу того что $S' \leqslant T'$ все элементы S' меньше или равны u, и есть по крайней мере один элемент в T' меньший или равный u. Значит, в $S \uplus T$ не менее $m+1$ элементов меньших или равных u. Значит, по крайней мере один такой элемент входит как в S, так и в T. Следовательно, последовательностям S и T можно сопоставить нуль в соответствии с условием $(*)$.

Случай 2: $y^{(0)}_{i_r} = 0$ и $y^{(0)}_{j_r} = 1$. Тогда в S' есть элемент меньший или равный u, обозначим его q, а в T' есть элемент больший u, обозначим его r. Если q и r были в разных входных последовательностях, то сопоставим нуль той последовательности, где находится q, и единицу той последовательности, где находится r. Если q и r были в одной входной последовательности, то этой последовательности согласно условию $(*)$ можно сопоставить как нуль, так и единицу. Пусть q и r попали в последовательность S. Тогда, если в T есть элемент меньший или равный u, сопоставим T нуль, а S — единицу. Иначе сопоставим T единицу, а S — нуль.

Случай 3: $y^{(0)}_{i_r} = 1$ и $y^{(0)}_{j_r} = 1$. В силу того что $S' \leqslant T'$ все элементы T' больше u, и есть по крайней мере один элемент в S' больший u. Значит, в $S \uplus T$ не менее $m+1$ элементов больших u. Значит, по крайней мере один такой элемент входит как в S, так и в T. Следовательно последовательностям S и T можно сопоставить единицы в соответствии с условием $(*)$.

Продолжая этот процесс, будем убирать из сети по одному сливающему устройству. В итоге получим последовательность $y^{(r)}$, которую можем принять за y. □

Используя лемму 5, можно обосновать корректность предложенного алгоритма.

Теорема 4. *Алгоритм 4 сортирует любую входную последовательность чисел.*

Доказательство. Предположим противное: найдётся входная последовательность, которую алгоритм не может обработать правильно, т.е. в выходной последовательности больший элемент стоит раньше меньшего. Сначала алгоритм сортирует все блоки по отдельности. Во время работы сортирующей сети сливающие устройства записывают в блоки только отсортированные последовательности. Следовательно, элементы, нарушающие порядок, не могут находится внутри одного блока. Если же они попадают в разные блоки, в силу леммы 5 это противоречит тому, что сеть является сортирующей. □

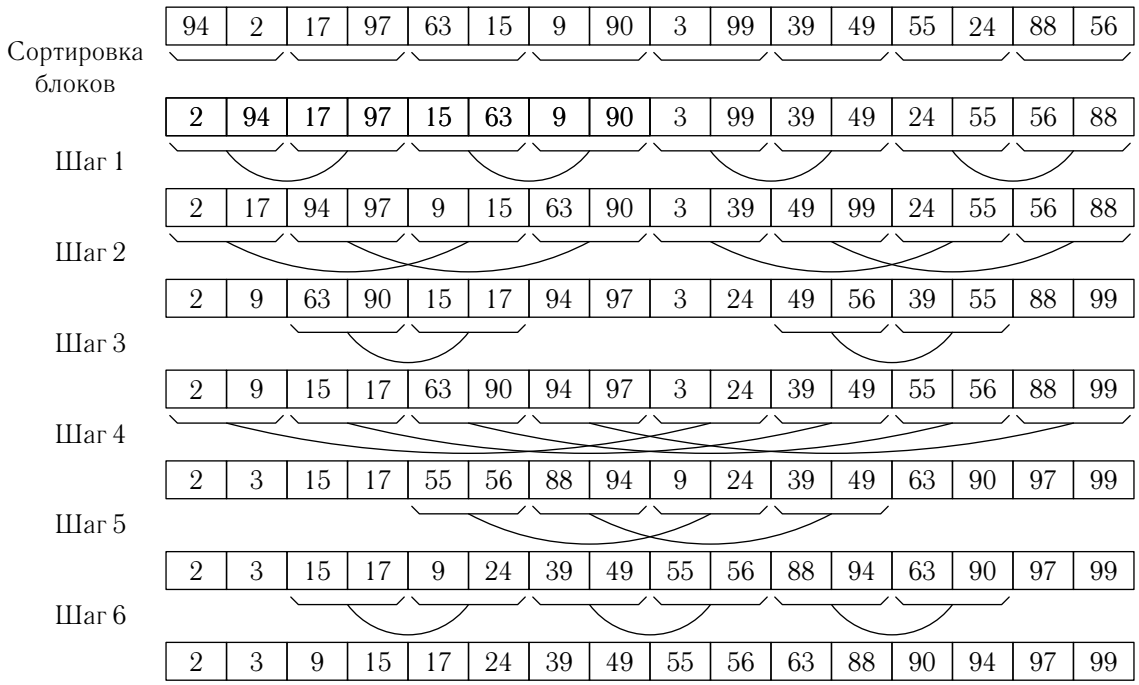


Рис. 13. Исполнение алгоритма блочной сортировки

В алгоритме 4 сортировка всех блоков выполняется параллельно. Существуют последовательные алгоритмы сортировки, которые имеют сложность $\Theta(n \log n)$. Задачу слияния отсортированных последовательностей длины n можно решить за $\Theta(n)$ действий. Следовательно, такую же сложность будет иметь функция, моделирующая сливающее устройство. Сортирующая сеть все сливающие устройства одинаковой глубины выполняет параллельно. Если воспользоваться сетью чётно-нечётной сортировки слиянием, сложность блочной сортировки будет

$$\Theta((n/p)\log(n/p)) + \Theta((n/p)\log^2 p).$$

Первое слагаемое — это затраты на сортировку блоков, второе — на работу сети из сливающих устройств.

7 Обобщение на случай последовательности произвольной длины

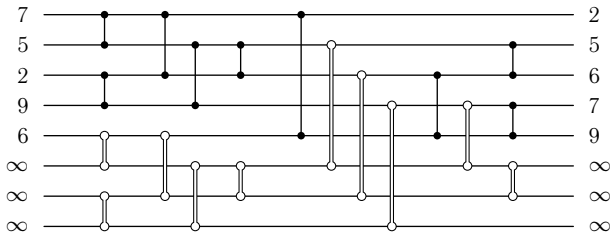
В разделе 5 было показано как построить сеть чётно-нечётной сортировки слиянием, если число входов n является степенью двойки. Если это не так, найдем k, такое что $2^{k-1} < n < 2^k$. Построим сеть на 2^k входов. Будем считать, что на линии, номера которых больше или равны n, передаётся бесконечно большое число ∞, которое больше любого элемента входной последовательности. Так как сеть на 2^k входов сортирующая, выходная последовательность будет состоять из отсортированной последовательности длины n, за которой идёт последовательность из ∞ длины $2^k - n$ (рис. 14,а).

На вход сравнивающего устройства ∞ попадает в двух случаях. Первый: на верхнем входе конечное число, на нижнем ∞. Второй: на обоих входах ∞. В обоих случаях сравнивающее устройство не выполняет обмена. Следовательно, сравнивающие устройства, соединённые с линиями, номера которых больше или равны n, избыточны. На рис. 14,а избыточные устрой-

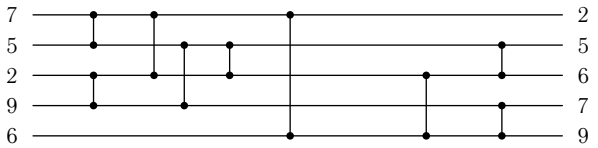
ства обведены по контуру. Удалив избыточные устройства из сети, получим сортирующую сеть на n входов (рис. 14,б).

Рассмотрим блочную сортировку в случае, когда n не кратно p. Пусть $n = qp + r$, где $r \neq 0$. Тогда $n = s(q+1)+t$ и $s < p$. Если $t = 0$, разобьём входную последовательность на s блоков по $q + 1$ элементов в каждом. Иначе разобьём на $s + 1$ блоков длины $q + 1$, дополнив последний блок элементами равными ∞. Так как в процессе работы алгоритма ∞ всегда остаются в конце последнего блока, можно считать, что последний блок имеет длину t.

Сортирующая сеть на n входов выполняет одновременно не более $\lfloor n/2 \rfloor$ сравнений. Поэтому, если на компьютере с p процессорами исполнять сеть на p входов, половина процессоров всегда будет простаивать. В алгоритме блочной сортировки выгоднее разбить входную последовательность на $2p$ блоков. Сначала каждый процессор сортирует по два блока. Затем исполняется сортирующая сеть на $2p$ входов, составленная из сливающих устройств. Если n не кратно $2p$, можно воспользоваться описанным выше приёмом, заменив p на $2p$.



а)



б)

Рис. 14. Построение сети `MERGE_SORTER[5]`

Основные обозначения

$\lfloor x \rfloor$ — наибольшее целое, меньшее или равное вещественного x;

$\lceil x \rceil$ — наименьшее целое, большее или равное вещественного x;

$$O(g(n)) = \left\{ \begin{array}{l} f(n) \mid \text{существуют положительные константы } C, N \\ \text{такие что } 0 \leqslant f(n) \leqslant Cg(n) \text{ для всех } n \geqslant N \end{array} \right\};$$

$$\Omega(g(n)) = \left\{ \begin{array}{l} f(n) \mid \text{существуют положительные константы } C, N \\ \text{такие что } 0 \leqslant Cg(n) \leqslant f(n) \text{ для всех } n \geqslant N \end{array} \right\};$$

$$\Theta(g(n)) = \left\{ \begin{array}{l} f(n) \mid \text{существуют положительные константы } C_1, C_2 \text{ и } N \\ \text{такие что } 0 \leqslant C_1 g(n) \leqslant f(n) \leqslant C_2 g(n) \text{ для всех } n \geqslant N \end{array} \right\};$$

$\square$ — конец доказательства.

Список литературы

[1] *Кормен Т.Х., Лейзерсон Ч.И., Ривест Р.Л., Штайн К.* Алгоритмы: построение и анализ, 2-е издание. М.: ИД «Вильямс», 2012.

[2] *Кнут Д.Э.* Искусство программирования, том 3. Сортировка и поиск, 2-е издание. М.: ИД «Вильямс», 2007.

[3] *Седжвик Р.* Алгоритмы на C++. М.: ИД «Вильямс», 2011.

[4] *Grama A., Gupta A., Karypis G., Kumar V.* Introduction to Parallel Computing, 2nd edition. Addison-Wesley, 2003.

[5] *Batcher K.E.* Sorting Networks and their Applications // Proc. AFIPS Spring Joint Comput. Conf. 1968. Vol. 32. P. 307–314.

[6] *Ajtai M., Komlós J., Szemerédi E.* Sorting in $c \log n$ parallel steps // Combinatorica. 1983. Vol. 3. P. 1–19.