# Сложность сортировки и выбора. Материалы для лекций.

*И. С. Сергеев*[1]

предварительная версия 1 от 16 октября 2024 г.

[1]e-mail: isserg@gmail.com

# Оглавление

# Глава 1

# Минимизация числа сравнений в задаче сортировки

## 1.1 Введение

Рассмотрим классическую и традиционную (особенно для программирования) задачу сортировки. Обычно требуется отсортировать числовой набор, но в общем случае элементы набора могут принадлежать произвольному множеству, на котором задано отношение порядка.

*Частичный порядок* на множестве $\mathcal{M}$ — это бинарное отношение «$\leqslant$», удовлетворяющее свойствам: 1) рефлексивности: $a \leqslant a$; 2) транзитивности: из $b \leqslant a$ и $c \leqslant b$ следует $c \leqslant a$; 3) антисимметричности: из $b \leqslant a$ и $a \leqslant b$ следует $a = b$ для любых $a, b, c \in \mathcal{M}$. Вместо $b \leqslant a$ будем также использовать обозначение $a \geqslant b$.

Множество с заданным на нем частичным порядком $(\mathcal{M}, \leqslant)$ называется *частично упорядоченным множеством*, сокращенно ч.у.м. Элементы $a, b \in \mathcal{M}$, для которых выполнено либо $a \leqslant b$, либо $b \leqslant a$, называются *сравнимыми*. Частичный порядок, в котором любые два элемента из множества сравнимы, называется *линейным*. Легко проверяется, что любой частичный порядок может быть дополнен до линейного.

Задача сортировки корректно определена для набора элементов из множества с линейным порядком. Этот линейный порядок предполагается заранее неизвестным, и может быть выяснен при помощи операций сравнения. Результатом операции сравнения двух элементов $e_1, e_2$ является упорядоченная пара $(a,\, b)$, такая, что $\{a,\, b\} = \{e_1,\, e_2\}$ и $a \leqslant b$.

Алгоритмы, состоящие из операций сравнения, можно условно разделить на два класса: те, в которых выбор двух очередных элементов для сравнения зависит от результатов предыдущих сравнений, и те, в которых последовательность выполнения сравнений зафиксирована. Имеются и другие классификации, но мы их рассматривать не будем.

Алгоритм из первого класса удобно представить в виде дерева сравнений. Напомним, что *дерево* — это связный граф без циклов. *Корневое бинарное дерево с ориентацией в направлении от корня* — это дерево с ориентацией ребер, имеющее единственную вершину, из которой только выходят ребра (корень); из каждой внутренней вершины дерева выходят ровно два ребра, ориентированные в направлении от корня. *Дерево сравнений* — это корневое бинарное дерево с ориентацией в направлении от корня, в котором каждой внутренней вершине приписана пара элементов из входного набора, а два ребра, исходящие из внутренней вершины, отмечены символами «$\geqslant$» и «$\leqslant$».

По дереву сравнений алгоритм строится следующим образом: выполняется сравнение пары элементов, соответствующих корню; в зависимости от результата сравнения выполняется переход к следующей вершине по одному из ребер, выходящих из корня; для этой вершины выполняется та же процедура, и так далее, пока не дойдем до листа дерева.

Пример дерева сравнений для сортировки набора из трех элементов приведен на рис. 1.1. Листьям дерева приписаны упорядоченные перестановки элементов исходного набора.

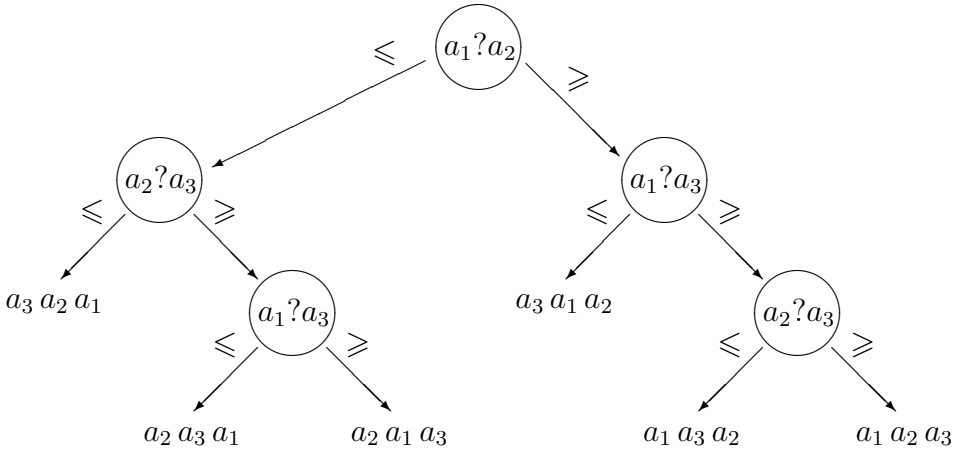



Рис. 1.1: Пример дерева сравнений

Ясно, что представление алгоритма деревом сравнений затруднительно с точки зрения наглядности. Уже описание алгоритма сортировки пяти элементов потребует изображения примерно сотни вершин. Поэтому дерево сравнений можно иметь в виду — оно далее пригодится при выводе нижних оценок — но для построения и анализа конкретных алгоритмов удобнее использовать другие средства, например, диаграммы Хассе.

Диаграммы Хассе используются для графического представления частично упорядоченных множеств. *Диаграмма* (или *диаграмма Хассе*) *ч.у.м.* — это ориентированный граф, в котором вершины соответствуют элементам ч.у.м., а ребро соединяет вершины $v_x$ и $v_y$, соответствующие элементам $x$ и $y$, и ориентировано в

направлении вершины $v_y$ тогда и только тогда, когда $x \leqslant y$ и нет такого элемента $z$, что $x \leqslant z \leqslant y$ (т.е. $x$ непосредственно предшествует $y$ в ч.у.м.). При изображении диаграммы ч.у.м. ориентация ребер часто опускается, при этом полагается, что из двух элементов, соединенных ребром, больше тот, который находится выше.

На рис. 2 диаграммы Хассе используются для описания алгоритма сортировки 4-х элементов. На каждом шаге изображается диаграмма множества с частичным порядком, который определяется выполненными сравнениями. Выделены пары элементов, которые сравниваются на очередном шаге.

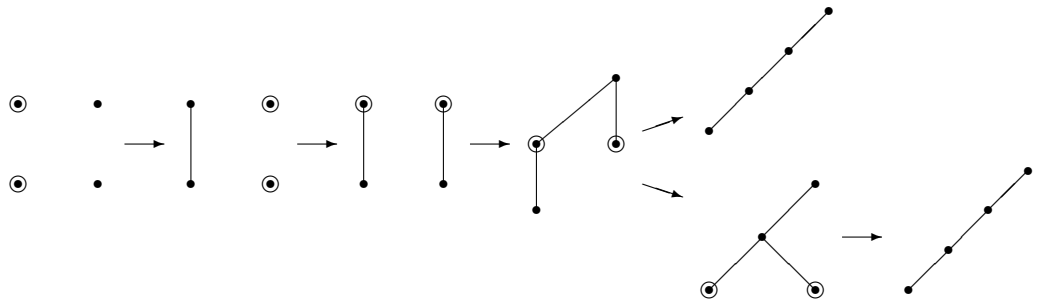


Рис. 2

## 1.2 Сортировка. Нижняя оценка числа сравнений

Пусть $(\mathcal{P}, \leqslant)$ — конечное ч.у.м.[1], состоящее из неравных элементов. Обозначим через $e(\mathcal{P})$ число возможных доопределений частичного порядка $\leqslant$ до линейного на множестве $\mathcal{P}$. Иначе говоря, $e(\mathcal{P})$ — это число перестановок элементов множества $\mathcal{P}$, согласованных с частичным порядком (т.е., в которых элементы перечисляются в порядке «неубывания»).

Поставим задачу *сортировки* в следующем общем виде. Дано ч.у.м. $\mathcal{P}$ из неравных элементов, подчиненных линейному порядку $\leqslant$, который неизвестен. Требуется упорядочить элементы $\mathcal{P}$ согласно порядку $\leqslant$, используя попарные сравнения элементов.

Без ограничения общности дальше будем считать, что сортируемое множество является подмножеством множества действительных чисел $\mathbb{R}$, на котором задан обычный линейный порядок $\leqslant$.

Сложностью алгоритма сортировки называется максимальное число сравнений, выполняемых алгоритмом, по всем возможным упорядочениям элементов входного набора. Сложность совпадает с глубиной дерева сравнений, изображающего алгоритм, т.е. числом ребер в самой длинной ориентированной цепочке от корня к листу дерева. Через $\mathsf{S}(\mathcal{P})$ обозначим минимум сложности алгоритмов, сортирующих ч.у.м. $\mathcal{P}$, а через $\mathsf{S}(n)$ — сложность сортировки никак не упорядоченных $n$ элементов (т.е. ч.у.м. с пустым частичным порядком).

[1] Далее символ операции в обозначении ч.у.м. мы будем опускать.

**Теорема 1.1.** $\mathsf{S}(\mathcal{P}) \geqslant \log_2 e(\mathcal{P})$.

► Для доказательства нам понадобится простой факт: корневое бинарное дерево глубины $h$ может иметь не более $2^h$ листьев. Следовательно, дерево с $N$ листьями имеет глубину не менее $\log_2 N$.

Лист дерева сравнений в алгоритме сортировки соответствует некоторой перестановке элементов входного набора; наоборот, каждой возможной перестановке соответствует некоторый лист (возможно, не один). Следовательно, дерево имеет не менее $e(\mathcal{P})$ листьев, откуда вытекает заявленная нижняя оценка. ■

**Следствие 1.1.** $\mathsf{S}(n) \geqslant \log_2(n!)$.

Напомним, что согласно известной формуле Стирлинга

$$\log_2(n!) = n\log_2 n - n/\ln 2 + 0.5\log_2 n + O(1).$$

Далее мы увидим, что нижняя оценка теоремы 1.1 и следствия 1.1, основанная на самых общих соображениях, оказывается не только асимптотически точной, но и по существу очень незначительно отличается от точных значений сложности.

## 1.3 Слияние упорядоченных наборов. Сортировка слиянием

Попутно рассмотрим еще одну часто встречающуюся задачу: слияние двух упорядоченных наборов. Пусть в первом наборе $m$ элементов, а во втором — $n$. Требуется выполнить сортировку указанного ч.у.м. $\mathcal{L}_{m,n}$ из $n+m$ элементов. Легко видеть, что $e(\mathcal{L}_{m,n}) = C_{n+m}^m$, поскольку существует $C_{n+m}^m$ способов объединить два данных упорядоченных набора в один.

Из теоремы 1.1 сразу вытекает $\mathsf{S}(\mathcal{L}_{m,n}) \geqslant \log_2 C_{n+m}^m$. В случае $m = 1$ получаем оценку $\mathsf{S}(\mathcal{L}_{n,1}) \geqslant \log_2(n+1)$, которая на самом деле является точной.

**Лемма 1.2.** $\mathsf{S}(\mathcal{L}_{n,1}) = \lceil \log_2(n+1) \rceil$.

▷ Осталось доказать верхнюю оценку. Доказательство проведем по индукции. В случае $n = 1$ утверждение очевидно верно. Рассмотрим индуктивный переход от $n$ к $n+1$. Сравним единственный элемент одного из массивов со средним элементом другого массива (с любым из двух средних, если $n$ — четно). В зависимости от результата сравнения, далее вставляем указанный элемент в одну из двух половин последнего массива. Получаем

$$\mathsf{S}(\mathcal{L}_{n,1}) \leqslant \mathsf{S}(\mathcal{L}_{\lceil (n-1)/2 \rceil,1}) + 1 \leqslant \log_2 \lceil (n+1)/2 \rceil + 1 = \lceil \log_2(n+1) \rceil.$$

□

Описанная в доказательстве леммы процедура называется бинарными вставками.

Интересен также случай близких по величине наборов $m = n + O(1)$, для которого согласно формуле Стирлинга выполняется $\mathsf{S}(\mathcal{L}_{m,n}) \geqslant m + n - O(\log n)$. Справедлива

**Лемма 1.3.** $\mathsf{S}(\mathcal{L}_{n,n}) = 2n - 1,\ \mathsf{S}(\mathcal{L}_{n+1,n}) = 2n.$

▷ По индукции докажем более общую верхнюю оценку $\mathsf{S}(\mathcal{L}_{m,n}) \leqslant m+n-1$. Она очевидна при $m + n \leqslant 2$. В случае $m + n > 2$ выполним сравнение максимальных элементов двух наборов — больший из них является максимальным элементом объединенного набора, удалим его из исходного набора. Останется выполнить слияние двух новых наборов с общим числом элементов $m + n - 1$. Применим индуктивное предположение, получим требуемую оценку.

Докажем нижнюю оценку для случая $m = n$. Пронумеруем элементы первого набора в порядке возрастания $a_i$, а элементы второго набора — $b_i$. В дереве сравнений рассмотрим путь, соответствующий упорядочению

$$b_1 \leqslant a_1 \leqslant b_2 \leqslant a_2 \leqslant \ldots \leqslant b_n \leqslant a_n.$$

На этом пути обязательно должны быть выполнены сравнения любых соседних элементов в данной перестановке, поскольку информация об отношении между соседними элементами изначально отсутствует, и не может быть получена даже если произвести все остальные сравнения. Таким образом, должно быть выполнено не менее $2n-1$ сравнений. Случай $m = n+1$ рассматривается аналогично. □

На базе алгоритма слияния можно построить простой и при этом асимптотически оптимальный по сложности алгоритм сортировки.

**Теорема 1.2.** $\mathsf{S}(n) \leqslant \log_2(n!) + O(n).$

► Метод, по существу, эксплуатирует популярную идею «деления пополам». Сортируемый набор делится на две части, эти части упорядочиваются по отдельности, выполняется слияние двух упорядоченных наборов. Используя доказанную лемму, для сложности $s(n)$ алгоритма можно выписать рекуррентные соотношения:

$$s(2n) = 2s(n) + 2n - 1, \qquad s(2n+1) = s(n) + s(n+1) + 2n \tag{1.1}$$

с начальным условием $s(1) = 0$.

**Утверждение 1.4.**

$$s(n) = n\lceil \log_2 n \rceil - 2^{\lceil \log_2 n \rceil} + 1. \tag{1.2}$$

▷ Введем функцию $q(n) = s(n) - s(n-1)$. Соотношения (1.1) удобно переписать как

$$q(2n) = q(2n-1) = q(n) + 1$$

и задать начальное условие $q(1) = 0$. Новые соотношения легко разрешаются в виде $q(n) = \lceil \log_2 n \rceil$. Остается воспользоваться формулой $s(n) = \sum_{i=1}^{n} q(n) = \sum_{i=1}^{n} \lceil \log_2 n \rceil$.

Теперь несложно проверить справедливость формулы (1.2) по индукции. Если (1.2) справедлива для $s(n)$, и $q(n+1) = q(n)$, то (1.2) очевидно справедлива и для $s(n+1)$. Иначе, если $q(n+1) = q(n) + 1$, т.е. $n = 2^k$, то

$$s(n) + q(n+1) = nk - n + 1 + (k+1) = (n+1)(k+1) - 2n + 1,$$

что и требовалось. □

Таким образом, $\mathsf{S}(n) \leqslant s(n) = n \log_2 n - O(n)$. ■

Сложность метода наиболее близка к нижней оценке сложности сортировки при $n = 2^k$, когда $s(n) = n \log_2 n - n + 1$.

## 1.4 Быстрая сортировка частично упорядоченного множества

Перейдем к более общей задаче сортировки произвольного ч.у.м. $\mathcal{P}$. Доказательство следующей теоремы опирается на метод бинарных вставок.

**Теорема 1.3** (М. Фредман)**.** $\mathsf{S}(\mathcal{P}) \leqslant \log_2 e(\mathcal{P}) + 2|\mathcal{P}|$.

► На самом деле, мы докажем чуть более общее утверждение, из которого сразу вытекает теорема.

**Лемма 1.5.** *Пусть $M$ — конечное подмножество $\mathbb{Z}^n$. Тогда можно определить заданный неизвестный вектор $b = (b_1, \ldots, b_n) \in M$ не более чем за $\log_2 |M| + 2n$ запросов вида $b_i \overset{?}{\leqslant} x$, выполняемых последовательно в порядке возрастания $i$.*

▷ Будем определять координаты вектора $b$ последовательно. Обозначим $M_k = \{c \in M \mid c_1 = k\}$ — множество векторов из $M$ с первой координатой $k$. Пусть $k_1 < k_2 < \ldots < k_t$ — индексы всех непустых множеств $M_k$. Без ограничения общности, можем полагать $k_i = i$ для всех $i = 1, \ldots, t$. Обозначим $m_k = |M_k|/|M|$. По построению, $m_1 + \ldots + m_t = 1$.

Разобьем отрезок $[0, 1]$ последовательно на интервалы длины $m_1, \ldots, m_t$. Пусть $f(x)$ означает число центров интервалов левее точки $x$.

Методом бинарного поиска мы можем определить номер $s$ интервала, для которого $b \in M_s$, последовательно выполняя запросы: $b_1 \overset{?}{\leqslant} f(1/2)$, затем, в зависимости от результата, $b_1 \overset{?}{\leqslant} f(1/4)$ или $b_1 \overset{?}{\leqslant} f(3/4)$ и т.д. Ввиду того, что заведомо $b_1 > f(0) = 0$ и $b_1 \leqslant f(1) = t$, после выполнения $j$ запросов имеем соотношение

$$\lambda_j = f(r_j/2^j) < b_1 \leqslant f((r_j+1)/2^j) = \rho_j$$

при некотором $r_j \in \mathbb{Z}$. Поиск завершается при условии $\lambda_j = \rho_j - 1$ — в этом случае искомый интервал найден: $b_1 = \rho_j$.

Положим $j = 1 - \lfloor \log_2 m_s \rfloor$, при этом $m_s \geqslant 2^{1-j}$. Тогда обе точки $r_j/2^j$ и $(r_j+1)/2^j$ принадлежат $s$-му подотрезку, поэтому условие завершения поиска заведомо выполнено после $j$ сравнений.

Поиск продолжается в множестве $M_s$ с $n-1$ неизвестными координатами. Оценка леммы доказывается по индукции.

При $n = 1$ в силу приведенного выше рассуждения ввиду $m_s = 1/|M|$ достаточно выполнить $1 + \lceil \log_2 |M| \rceil$ запросов. В общем случае (переход от $n-1$ к $n$) число запросов не превосходит

$$\log_2 |M_s| + 2(n-1) + 1 + \lceil \log_2 |M|/|M_s| \rceil \leqslant \log_2 |M| + 2n.$$

□

Теорема доказывается методом последовательных вставок $n = |\mathcal{P}|$ элементов $X_1, \ldots, X_n \in \mathcal{P}$ в упорядоченный массив. Вначале массив состоит из одного элемента $X_1$, затем к нему добавляется второй элемент $X_2$, потом вставляется третий и т.д. Такая процедура эквивалентна определению вектора $b = (b_1, \ldots, b_n)$, где $b_i$ — номер позиции для вставки $i$-го элемента. Запрос $b_i \overset{?}{\leqslant} k$ соответствует сравнению $X_i ? Y_k$, где $Y_k$ — элемент ранга $k$ в уже построенном массиве. Несложно видеть, что любое линейное расширение ч.у.м. $\mathcal{P}$ однозначно характеризуется вектором $b \in \mathbb{Z}^n$. Применим лемму 1.5. ■

Оценка теоремы 1.3 хороша для «разреженных» ч.у.м., с большим числом расширений. Но при малых $e(\mathcal{P})$ она не вполне эффективна. Далее мы покажем, что для сортировки ч.у.м. всегда достаточно $O(\log e(\mathcal{P}))$ сравнений.

Сначала заметим, что доказательство теоремы 1.3 влечет

**Следствие 1.6.** *Пусть ч.у.м. $\mathcal{P}$ содержит линейно упорядоченное подмножество мощности $l$. Тогда $\mathsf{S}(\mathcal{P}) \leqslant \log_2 e(\mathcal{P}) + 2(n-l)$.*

▷ Процедура вставок начинается с упорядоченного массива длины $l$. Можно полагать, что координаты $b_1, \ldots, b_l$ в условии леммы 1.5 фиксированы. □

Следующая лемма показывает, что число линейных расширений ч.у.м. $\mathcal{P}$ мало ровно в том случае, когда в $\mathcal{P}$ имеется большое линейно упорядоченное подмножество (это в точности ситуация, когда оценка теоремы 1.3 неэффективна).

**Лемма 1.7.** *Пусть максимальное линейно упорядоченное подмножество в ч.у.м. $\mathcal{P}$ имеет мощность $l$. Тогда $e(\mathcal{P}) \geqslant 2^{|P|-l}$.*

▷ Элементы ч.у.м. $\mathcal{P}$ можно разбить на $l$ слоев: $V_1, \ldots, V_l$. Множество $V_i$ содержит такие элементы, для которых длина максимальной упорядоченной по возрастанию цепочки в $\mathcal{P}$, оканчивающейся этим элементом, равна $i$.

По построению, элементы одного слоя попарно несравнимы и могут быть доупорядочены произвольно. Поэтому в силу простого неравенства $k! \geqslant 2^{k-1}$,

$$e(\mathcal{P}) \geqslant \prod_{i=1}^{l} |V_i|! \geqslant \prod_{i=1}^{l} 2^{|V_i|-1} = 2^{|\mathcal{P}|-l}.$$

□

**Теорема 1.4** (Дж. Кан, М. Сакс)**.** $\mathsf{S}(\mathcal{P}) = O(\log e(\mathcal{P}))$.

► Пусть $|\mathcal{P}| = n$. В графе (диаграмме Хассе) ч.у.м. $\mathcal{P}$ выделим максимальную цепочку $Z$; ее длину обозначим через $l$. Для каждого элемента $v$ вне $Z$ определим минимальный интервал $(a_v, b_v)$ в $Z$, куда может попасть $v$. Здесь $a_v$ — это максимальный элемент $Z$ с условием $a_v \leqslant v$ и $b_v$ — это минимальный элемент $Z$ с условием $v \leqslant b_v$, при условии, что такие элементы существуют. Интервал может быть неограниченным с одной или двух сторон.

Рассмотрим множество $\mathcal{P}'$, составленное из элементов вне $Z$ и из тех элементов $Z$, которые являются краями минимальных интервалов. Очевидно, $|\mathcal{P}'| \leqslant 3(n-l)$. Первый этап алгоритма состоит в сортировке $\mathcal{P}'$ методом теоремы 1.3 и использует не более $\log_2 e(\mathcal{P}') + 6(n-l)$ сравнений. Поскольку $\mathcal{P}' \subset \mathcal{P}$ и все отношения между элементами $\mathcal{P}'$ ровно такие же, как и в $\mathcal{P}$, выполнено $e(\mathcal{P}') \leqslant e(\mathcal{P})$. Другими словами, любое линейное расширение $\mathcal{P}'$ содержится в некотором расширении $\mathcal{P}$.

После сортировки $\mathcal{P}'$ ч.у.м. состоит из двух упорядоченных и, вообще говоря, пересекающихся цепочек, как показано на рис. 1.2 (цепочки элементов изображены жирными линиями). Сортировка ч.у.м. сводится к независимым слияниям пар фрагментов цепочек. При этом вторые фрагменты в парах содержат исключительно элементы вне $Z$, и их суммарная длина не превышает $n-l$.

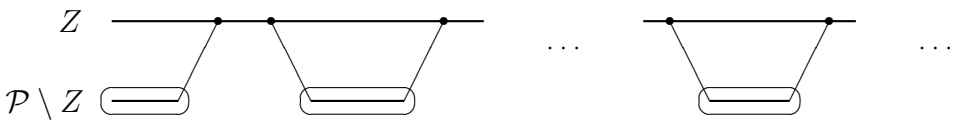



Рис. 1.2: Вид ч.у.м. после первого этапа

Заметим, что если обозначить через $T_i$ число возможных исходов выполнения слияния в $i$-й паре, то $\prod_i T_i \leqslant e(\mathcal{P})$, поскольку любая комбинация результатов

слияний в разных фрагментах является легальным расширением $\mathcal{P}$ (это следует из минимальности интервалов для вершин вне $Z$ — никаких дополнительных отношений между вершинами в $Z$ и вне $Z$ по отношению к показанным на рис. 1.2 в $\mathcal{P}$ нет).

Используя следствие 1.6, находим, что для выполнения слияний достаточно $\sum_i \log_2 T_i + 2(n-l)$ сравнений. Объединяя оценки сложности обоих этапов, получаем метод сортировки множества $\mathcal{P}$ за

$$\log_2 e(\mathcal{P}') + 6(n-l) + \sum_i \log_2 T_i + 2(n-l) \leqslant 10 \log_2 e(\mathcal{P})$$

сравнений с учетом леммы 1.7. ■

# Упражнения

**Упр. 1.1.** Покажите, что метод сортировки бинарными вставками также имеет сложность $\log_2(n!) + O(n)$. Метод заключается в разбиении множества элементов на пары, выполнении сравнений в парах, затем в упорядочении старших элементов пар с последующей вставкой младших элементов пар в упорядоченное множество.

**Упр. 1.2.** Определите величину $\mathsf{S}(\mathcal{L}_{2,n})$ (решение есть в [1]). Результат показывает, что вставку двух элементов в упорядоченный массив часто выгоднее выполнять совместно, нежели по отдельности. [*Р. Грэхем, Ф. Хуан, Ш. Линь*]

**Упр. 1.3.** Покажите, что даже $1.44 \log_2 e(\mathcal{P})$ сравнений может быть недостаточно для сортировки ч.у.м. $\mathcal{P}$ в общем случае. Для этого рассмотрите множество $x_1, \ldots, x_n$ с заданным частичным порядком:

$$x_i \leqslant x_{i+2}, \quad i = 1, \ldots, n-2, \qquad x_i \leqslant x_{i+3}, \quad i = 1, \ldots, n-3.$$

(См. доказательство леммы 1.3.) [*Н. Линьял*]

**Комментарии.** Материал разделов 1.1–1.3 стандартен и очень подробно рассматривается в [1]. Теорема 1.3 доказана М. Фредманом в [4]. Доказательство теоремы 1.4 в основном следует схеме, предложенной в [5].

Еще несколько методов сортировки, помимо сортировки слияниями, позволяют достичь асимптотически точной оценки сложности $\log_2(n!)$ с превышением на $O(n)$. Наиболее близко к ней подходит метод бинарных вставок в версии Форда—Джонсона, см. [1]. Автор доказал [2], что $\mathsf{S}(n) = \log_2(n!) + o(n)$ (метод групповых вставок).

Результат теоремы 1.4 о сложности сортировки произвольного ч.у.м. сначала был получен в [7] как следствие наличия в любом не полностью упорядоченном ч.у.м. $\mathcal{P}$ сбалансированной пары элементов $x, y$, т.е. такой, что определение порядка в этой паре

кратно уменьшает число расширений, а именно, $c \leqslant \frac{e(\mathcal{P}, x \leqslant y)}{e(\mathcal{P})} \leqslant 1 - c$, где $c$ — универсальная константа. Согласно известной гипотезе, должно быть $c = 1/3$, но пока доказано только, что $c \geqslant \frac{5-\sqrt{5}}{10} \approx 0.28$ [3]. Методы доказательства оценок на константу сбалансированности [7, 3] используют весьма нетривиальный аппарат выпуклой геометрии. Другой подход [6] к построению алгоритма сортировки связан с анализом энтропии графов ч.у.м. и тоже не вполне прост. Предложенный в [5] метод элементарен, хотя и приводит к большей мультипликативной постоянной в оценке сложности.

## Литература


[1] Кнут Д. Э. *Искусство программирования. Т.* 3. *Сортировка и поиск.* М.: Вильямс, 2007.

[2] Сергеев И. С. *О верхней границе сложности сортировки.* Журнал вычислительной математики и математической физики. 2021. **61**(2), 345–362.

[3] Brightwell G. R., Felsner S., Trotter W. T. *Balancing pairs and the cross product conjecture.* Order. 1995. **12**, 327–349.

[4] Fredman M. L. *How good is the information theory bound in sorting?* Theor. Comput. Sci. 1976. **1**, 355–361.

[5] Haeupler B., Hladík R., Iacono J., Rozhon V., Tarjan R. E., Tětek J. *Fast and simple sorting using partial information.* 2024. arXiv:2404.04552v1.

[6] Kahn J., Kim J. H. *Entropy and sorting.* Proc. STOC (Victoria, Canada, 1992). NY: ACM, 1992, 178–187.

[7] Kahn J., Saks M. *Balancing poset extensions.* Order. 1984. **1**, 113–126.

# Глава 2

# Сложность задачи выбора

## 2.1 Введение

Задача *выбора* элементов ранга $r_1, \ldots, r_k$ в множестве мощности $n$, где $r_i \leqslant n$, с неизвестным линейным порядком $\leqslant$ ставится аналогично задаче сортировки. Эта задача является частным случаем задачи генерации ч.у.м., в которой требуется, чтобы путем ряда сравнений получить заданное ч.у.м. или его расширение. Обозначим через $\mathsf{C}(\mathcal{P})$ сложность генерации ч.у.м. $\mathcal{P}$. Из теоремы 1.1 вытекает

**Следствие 2.1.** $\mathsf{C}(\mathcal{P}) \geqslant \log_2(|\mathcal{P}|!/e(\mathcal{P}))$.

Выбор элементов ранга $r_1, \ldots, r_k$ в $n$-элементном множестве состоит в построении ч.у.м. со структурой

$$V_1 \leqslant x_1 \leqslant V_2 \leqslant x_2 \leqslant \ldots \leqslant V_k \leqslant x_k \leqslant V_{k+1}, \tag{2.1}$$

где $x_i$ — элемент ранга $r_i$, а $V_i$ — множество из $r_i - r_{i-1} - 1$ элементов (здесь полагаем $r_0 = 0$ и $r_{k+1} = n + 1$). Проверим, что это действительно так.

**Лемма 2.2.** *Если в ч.у.м. известен элемент ранга $m$, то также известно множество из $m - 1$ элементов, больших его.*

▷ Предположим противное. Пусть $e$ — рассматриваемый элемент. Обозначим через $M_1$, $M_2$ и $M_3$ множество элементов, больших $e$, меньших $e$ и множество элементов, не сравнимых с $e$, соответственно. По предположению, $M_3 \neq \varnothing$.

По построению, элемент из $M_3$ не может быть больше никакого элемента из $M_1$ и не может быть меньше никакого элемента из $M_2$. Следовательно, существует линейное расширение ч.у.м., в котором множество $M_3 \cup \{e\}$ расположено строго между множествами $M_1$ и $M_2$. Элемент $e$ может занимать произвольное положение в множестве $M_3 \cup \{e\}$, следовательно, его порядок в совокупном множестве не определен однозначно. □

Сложность задачи выбора в общей постановке обозначим через $\mathsf{C}_{r_1,\ldots,r_k}(n)$.

## 2.2 Выбор крайних элементов

Легко показать, что сложность выбора максимального (минимального) элемента множества равна $n-1$.

**Теорема 2.1.** $\mathsf{C}_1(n) = n-1$.

► Верхняя оценка $\mathsf{C}_1(n) \leqslant n-1$ следует из очевидного алгоритма, в котором на каждом шаге текущий максимум (из множества рассмотренных элементов) сравнивается с новым элементом, выбирается максимальный и т.д.

Нижняя оценка вытекает из тривиального наблюдения: каждый элемент, кроме максимального, должен проиграть хотя бы в одном сравнении. ■

Менее очевидным является ответ на вопрос, сколько потребуется сравнений для определения максимального и минимального элементов множества. Изящное решение этой задачи было найдено Полом в 1972 г.

**Теорема 2.2** (А. Пол)**.** $\mathsf{C}_{1,n}(n) = \lceil 3n/2 \rceil - 2$.

► Верхняя оценка доказывается следующим простым алгоритмом. Разобьем все элементы по парам, выполним сравнения в парах. Затем во множестве максимальных элементов пар, включив в нечетном случае в него непарный элемент, определим максимум. Аналогично, определим минимум среди минимальных элементов пар и, при необходимости, непарного элемента. Всего будет выполнено

$$\lfloor n/2 \rfloor + 2(\lceil n/2 \rceil - 1) = \lceil 3n/2 \rceil - 2$$

сравнений.

Докажем нижнюю оценку. На каждом шаге алгоритма будем обозначать четверкой $(a, b, c, d)$ соответственно число элементов, которые не участвовали в сравнениях, число элементов, которые только выигрывали, число элементов, которые только проигрывали, число элементов, которые как выигрывали, так и проигрывали.

Перед началом алгоритма $(a, b, c, d) = (n, 0, 0, 0)$, в конце алгоритма $(a, b, c, d) = (0, 1, 1, n-2)$. В дереве сравнений рассмотрим цепочку, в каждой вершине которой выбирается такая ветвь, что если в паре оба элемента — из одного подмножества (из четырех упомянутых выше), то исход сравнения выбирается произвольно; иначе, элемент, который до того только выигрывал (если он есть) выигрывает, а элемент, который только проигрывал (если такой есть), проигрывает; иначе допускается любой исход.

Тогда в ходе алгоритма никакое сравнение не изменяет одновременно числа $a$ и $d$. Таким образом, для того, чтобы «опустошить» первое подмножество, требуется не менее $\lceil n/2 \rceil$ сравнений, еще не менее $n-2$ сравнений требуется, чтобы «заполнить» четвертое подмножество. ■

Вопрос о сложности выбора второго элемента не настолько очевиден — дважды публиковались неправильные доказательства (нижней оценки), пока в 1962 г. С. С. Кислицын не предложил корректное, хотя довольно громоздкое доказательство. Впоследствии оно было существенно упрощено.

**Теорема 2.3** (С. С. Кислицын)**.** $\mathsf{C}_2(n) = n + \lceil \log_2 n \rceil - 2$.

► Для доказательства верхней оценки построим турнир с выбыванием по кубковой системе для определения максимального элемента. В таком турнире победитель участвует не более чем в $\lceil \log_2 n \rceil$ сравнениях. Любой элемент, который в ходе алгоритма не сравнивался с максимальным, уступает еще какому-либо элементу, следовательно, не может быть вторым. Остается определить наибольший элемент среди выбывших в непосредственных сравнениях с победителем. Таким образом, для выбора второго элемента достаточно выполнить не более $(n-1)+(\lceil \log_2 n \rceil - 1)$ сравнений.

Из леммы 2.2, в частности, следует, что любой алгоритм выбора второго элемента находит также первый элемент. Теперь покажем, что существует такое упорядочение элементов входного набора, что максимальный элемент участвует не менее, чем в $\lceil \log_2 n \rceil$ сравнениях.

В любой момент алгоритма элемент, который ни разу не проигрывал будем называть «лидером». Введем отношение «подчинения», согласно которому каждый элемент в любой момент времени подчинен некоторому лидеру, причем только одному; лидер подчинен сам себе. Перед началом работы алгоритма лидерами являются все элементы, подчиненные сами себе, а в конце остается только один лидер (максимальный элемент), которому подчинены все элементы.

Теперь в дереве сравнений рассмотрим такой путь, что в любом сравнении (не считая сравнений с предопределенным исходом) с участием лидера и не-лидера победителем признается лидер, с участием двух лидеров — лидер, участвовавший в большем числе сравнений, иначе — произвольно. Если лидер проигрывает очередное сравнение, то он сам и подчиненные ему элементы переходят в подчинение к победителю сравнения; результаты других сравнений не изменяют отношение подчинения.

По индукции проверим, что лидер, участвовавший в $r$ сравнениях, имеет в подчинении не более $2^r$ элементов (включая себя). При $r = 0$ это очевидно. Иначе, если лидер выигрывает $r$-е сравнение, то он в дополнение к не более чем $2^{r-1}$ подчиненных элементов (по предположению индукции) получает также не более $2^{r-1}$, потому что его соперником не мог быть лидер, выигравший более $r - 1$ сравнения.

Таким образом, максимальный из $n$ элементов будет участвовать не менее, чем в $\lceil \log_2 n \rceil$ сравнениях. Любой из множества элементов, непосредственно сравниваемых с победителем, кроме второго по порядку во всем множестве, должен

проиграть, по меньшей мере, еще в одном сравнении. Таким образом, $n-1$ элемент проигрывает хотя бы в одном сравнении и не менее $\lceil \log_2 n \rceil - 1$ элементов проигрывают еще по разу. ■

## 2.3 Выбор среднего элемента

Введем стандартное обозначение $H(x) = -x\log_2 x - (1-x)\log_2(1-x)$ для функции двоичной энтропии, определенной на отрезке $[0, 1]$. На концах отрезка полагаем $H(0) = H(1) = 0$ по непрерывности.

**Теорема 2.4** (С. Бент, Дж. Джон). *Пусть $t = \alpha n$. Тогда*

$$\mathsf{C}_t(n) \geqslant (1 + H(\alpha))n - O(\sqrt{n}).$$

► Рассмотрим произвольный алгоритм, решающий задачу выбора элемента ранга $t$ в множестве $M$ мощности $n$. Зафиксируем некоторое множество $T$ мощности $t$. Пусть в нем содержатся $t-1$ максимальных элементов множества $M$. Обозначим $U = M \setminus T$ и положим $U_0 = U$, $T_0 = T$. Далее множество $T_0 \cup U_0$ играет роль множества предентентов на позицию элемента ранга $t$. Пусть $r > 1$ и $q \geqslant r$ — параметры, которые будут определены позже.

а) В соответствующем алгоритму дереве сравнений пойдем по следующему пути. Пока $|T_0| > r$, исход сравнения элементов $x$ и $y$ определяется как $x \leqslant y$, если $x \in U$ и $y \in T$ и произвольным образом, если $x, y \in T$ или $x, y \in U$. При этом в предпоследнем случае больший элемент удаляется из $T_0$, а в последнем случае меньший элемент удаляется из $U_0$.

Остановим этот процесс в момент, когда $|T_0| = r$. Это обязательно случится, потому что на основании выполненных сравнений никакой из элементов множества $T_0$ нельзя исключить из числа претендентов. В конце работы алгоритма, если его не останавливать, было бы $|T_0| = 1$.

б) Для $y \in T_0$ обозначим через $W_y$ множество элементов из $U$, которые непосредственно сравнивались с $y$. Пусть $y^* \in T_0$ — такой элемент, что $|W_{y^*}|$ минимально. Положим $Q = (U_0 \setminus W_{y^*}) \cup \{y^*\}$.

Будем полагать, что в линейном порядке множество $Q$ расположено строго между множествами $(U \setminus U_0) \cup W_{y^*}$ (маленьких элементов) и $T \setminus \{y^*\}$ (больших элементов). Тогда элемент ранга $t$ множества $M$ является максимальным в $Q$. Заметим также, что никакие два элемента из $Q$ еще не сравнивались друг с другом.

в) Если $|W_{y^*}| > q - r$, то в алгоритме выполнено:

– не менее $(r-1)(q-r+1)$ сравнений элементов из $T_0$ и $U$;

– $n - |U_0| - r$ сравнений внутри множеств $U$ и $T$;

– $|W_{y^*}|$ сравнений элемента $y^*$ с элементами из $U$.

Кроме того, понадобится выполнить еще не менее $|U_0|-|W_{y^*}|$ сравнений для определения максимального элемента в $Q$. Следовательно, общая сложность алгоритма не меньше, чем

$$(r-1)(q-r+1)+(n-|U_0|-r)+|W_{y^*}|+(|U_0|-|W_{y^*}|)=n-1+(r-1)(q-r).$$

г) В случае $|W_{y^*}|\leqslant q-r$ докажем, что любому выбору множества $T$ в дереве сравнений соответствует не менее $2^{n-q}$ листьев. Для этого покажем, что суммарно не менее $n-q$ сравнений выполняется внутри множеств $T$, $U$ и затем $Q$.

Действительно, не менее $n-|U_0|-r$ сравнений выполняется внутри множеств $T$ и $U$ и не менее $|U_0|-|W_{y^*}|\geqslant|U_0|+r-q$ сравнений — внутри множества $Q$.

д) Множество $T$ можно выбрать $C_n^t$ способами, при этом одному исходу (листу дерева сравнений) отвечает $n+1-t$ множеств $T$ ($t-1$ максимальных элементов множества $T$ фиксированы, $t$-й элемент может быть выбран произвольно).

е) Таким образом, глубину дерева сравнений и сложность алгоритма можно оценить как $n-1+(r-1)(q-r)$ (ситуация п. в) или двоичный логарифм числа листьев (ситуация п. г), которых в дереве не меньше, чем $2^{n-q}C_n^t/(n+1-t)$. Таким образом,

$$\mathsf{C}_t(n)\geqslant\min\left\{n-1+(r-1)(q-r),\,n-q+\log_2\frac{C_n^t}{n+1-t}\right\}.$$

При выборе параметров $r\sim\sqrt{n}$ и $q\sim2\sqrt{n}$ с учетом

$$\log_2 C_n^t=\log_2 C_n^{\alpha n}=nH(\alpha)-O(\log n)$$

(следствие из формулы Стирлинга) приходим к утверждению теоремы. ■

Заметим, что при $\alpha=1/2$, т.е. для задачи выбора среднего элемента, оценка теоремы является максимальной и составляет асимптотически $2n$.

Наиболее эффектная верхняя оценка сложности задачи выбора, до сих пор улучшенная лишь незначительно, появилась в 1976 г. Для простоты изложения ограничимся выбором среднего элемента (медианы); общий случай принципиально не отличается от рассматриваемого.

**Теорема 2.5** (А. Шёнхаге, М. Патерсон, Н. Пиппенджер)**.** $\mathsf{C}_{\lfloor n/2\rfloor}(n)\leqslant3.5n+o(n)$.

► В основе метода лежит идея массового производства (частично упорядоченных множеств). Строится большое число ч.у.м. $\mathcal{X}_k$ мощности $2k+1$ с центральным элементом (медианой). По ходу алгоритма центральные элементы множеств упорядочиваются. Общий вид получаемого ч.у.м. изображен на рис. 2.1.

Заметим, что если число множеств $\mathcal{X}_k$ в цепочке $Z$ достаточно велико — общее число элементов в них близко к $n$ —, то бóльшая половина множества с максимальным центральным элементом и меньшая половина множества с минимальным центральным элементом (на рис. 2.1 они выделены) могут быть исключены из дальнейшего рассмотрения, как заведомо не содержащие медиану.

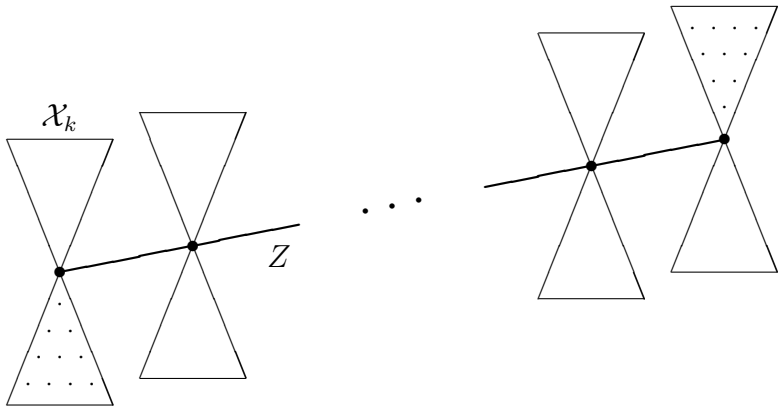



Рис. 2.1: Общий вид ч.у.м. в процессе вычислений

Действительно, обозначим через $s$ число ч.у.м. $\mathcal{X}_k$ в цепочке $Z$, а через $r$ — число элементов, не вошедших в ч.у.м. Ясно, что $s = (n-r)/(2k+1)$. Максимальный центральный элемент в цепочке согласно диаграмме рис. 2.1 превосходит еще $s(k+1)-1$ элементов.

Решая неравенство $s(k+1)-1 \geqslant n/2+1$, получаем условие $r \leqslant \frac{n+4}{2k+2} - 4$, при котором указанные элементы не претендуют на позицию среднего элемента. Далее средний элемент ищется среди $n-2k-2$ оставшихся.

Те элементы, которые не принадлежат готовым ч.у.м. $\mathcal{X}_k$ и не исключены из рассмотрения, считаются находящимися на фабрике (производящей новые ч.у.м. $\mathcal{X}_k$).

Фабрика характеризуется следующими показателями: $J_k$ — минимальное число элементов, достаточное для производства ч.у.м., $I_k$ — число сравнений, необходимое для запуска массового производства, $C_k$ — число сравнений, достаточное для построения одного ч.у.м. $\mathcal{X}_k$.

Обозначим через $S_n$ — общее число ч.у.м., изготовленных на фабрике, а через $R_n$ — число элементов в конце работы алгоритма. Последнее можно выбрать из условия $\frac{R_n+4}{2k+2} - 4 \geqslant J_k$, а первое легко определить как $S_n = (n-R_n)/(k+1)$.

Сложность алгоритма $T(n)$ теперь можно оценить как

$$T(n) = I_k + C_k \cdot S_n + S_n \log_2 n + O(R_n \log R_n), \tag{2.2}$$

где третье слагаемое отвечает сложности вставки центрального элемента ч.у.м. $\mathcal{X}_k$ в упорядоченный список, а последнее — поиску медианы остаточного множества из $R_n$ элементов, что можно выполнить обычной сортировкой.

Параметры выбираются следующим образом: $k \asymp \sqrt[4]{n}$ и $R_n \asymp n^{3/4}$. Для завершения доказательства теоремы остается построить фабрику с параметрами $I_k = O(k^2)$, $J_k = O(k^2)$ и $C_k \sim 3.5k$.

Определим индуктивно ч.у.м. $\mathcal{H}_p(v)$ — *гиперпару* порядка $p$ с центром $v$. Гиперпара $\mathcal{H}_0(v)$ состоит из единственного элемента $v$. Гиперпара $\mathcal{H}_p(v)$ образуется

из двух гиперпар $\mathcal{H}_{p-1}(v_1)$ и $\mathcal{H}_{p-1}(v_2)$ добавлением сравнения $v_1$ и $v_2$. В качестве $v$ выбирается победитель сравнения, если $p = 2$ или $p \geqslant 3$ — нечетно, и проигравший, если $p = 1$ или $p \geqslant 4$ — четно. Младшие представители семейства $\mathcal{H}_p$ изображены на рис. 2.2.

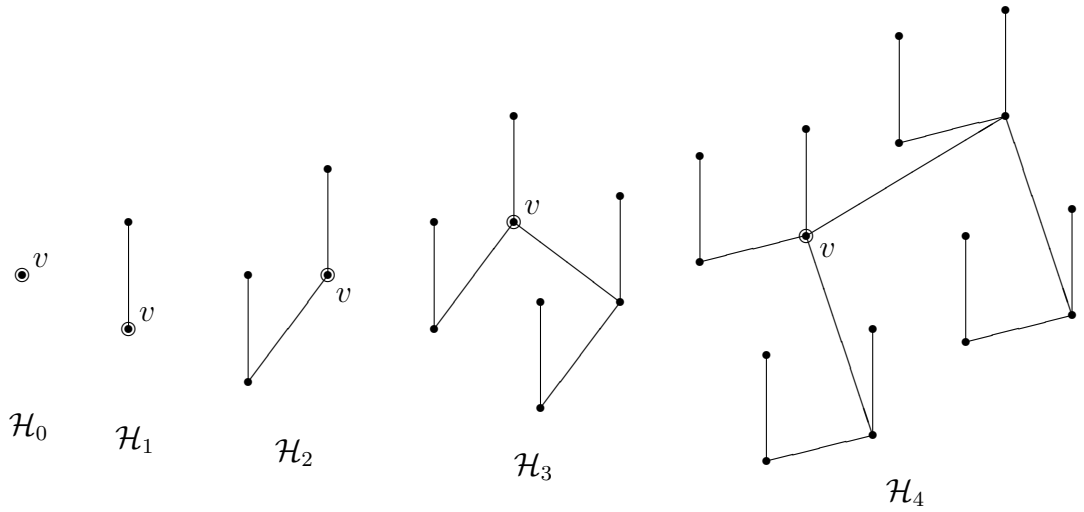



Рис. 2.2: Семейство гиперпар

Из определения вытекают следующие элементарно проверяемые по индукции свойства гиперпар.

**Лемма 2.3.** (*i*) *Гиперпара* $\mathcal{H}_p(v)$ *состоит из* $2^p$ *элементов.*

(*ii*) *При* $p \geqslant 2$ *гиперпара* $\mathcal{H}_p(v)$ *содержит* $2^{\lfloor p/2 \rfloor} - 1$ *элементов, бо́льших* $v$*, и* $2^{\lceil p/2 \rceil} - 1$ *элементов, меньших* $v$*.*

Таким образом, из гиперпары порядка $2p$ можно выделить ч.у.м. $\mathcal{X}_{2^p-1}$.

Отметим, что имеется взаимно однозначное соответствие ребер в диаграмме гиперпары и выполняемых при построении гиперпары сравнений. Также заметим, что при удалении центра $v$ гиперпары $\mathcal{H}_p(v)$ она распадается на семейство изолированных гиперпар $\mathcal{H}_0 \cup \{\mathcal{H}_{2i+1} \mid 1 \leqslant i < (p-1)/2\}$, центры которых выше $v$, и семейство гиперпар $\mathcal{H}_1 \cup \{\mathcal{H}_{2i} \mid 1 \leqslant i < p/2\}$, центры которых ниже $v$.

Далее, легко видеть, что множество элементов, больших (меньших) $v$ в гиперпаре $\mathcal{H}_p(v)$, является объединением множеств элементов, не меньших (не больших) центра в первом (во втором) семействе гиперпар.

В данном случае важно то, что удаление центра гиперпары приводит к образованию гиперпар меньшего порядка, которые могут быть использованы для «реконструкции» гиперпары. Более того, процедуру выделения ч.у.м., состоящего из центра гиперпары и некоторого числа элементов, меньших и больших центра, можно рассматривать как последовательность удалений центров определенных гиперпар, поэтому в результате также останется набор гиперпар меньших порядков.

Ввиду упомянутого соответствия между сравнениями и ребрами диаграммы, число сравнений, необходимых для восстановления гиперпары после удаления ч.у.м., равно числу удаленных из диаграммы ребер.

Обозначим через $V(p)$ и $N(p)$ число ребер, удаляемых при удалении центра гиперпары порядка $p$ и всех элементов, больших центра и соответственно меньших центра. Отдельно вычислим начальные значения этих функций: $V(0) = N(0) = 0, V(1) = N(1) = 1, V(2) = 2, N(2) = 3$.

**Лемма 2.4.** *При $p \geqslant 2$ справедливы соотношения:*

$$N(2p) = N(2p-1) + 1 = 5 \cdot 2^{p-1} - 2, \quad V(2p) = V(2p+1) - 1 = 5 \cdot 2^{p-1} - 3.$$

$\triangleright$ Центр гиперпары $\mathcal{H}_p$ в диаграмме имеет степень $p$, поэтому получаем следующие рекуррентные соотношения:

$$N(2p) = 2p + N(1) + N(2) + \ldots + N(2p-2),$$

$$V(2p+1) = 2p + 1 + V(0) + V(3) + \ldots + V(2p-1),$$

$$N(2p-1) = N(2p) - 1, \quad V(2p) = V(2p+1) - 1.$$

Подставляя начальные значения для $N(1), N(2), V(0)$, первые два (основных) соотношения решаются по индукции как $N(2p) = V(2p+1) = 5 \cdot 2^{p-1} - 2$. $\square$

Как следствие, мы можем выделить из гиперпары $\mathcal{H}_{2p}$ ч.у.м. $\mathcal{X}_{2^p-1}$ со сложностью восстановления не выше $5 \cdot 2^p$, что позволяет построить фабрику с характеристикой $C_k \sim 5k$, а это в силу (2.2) и выбора параметров ведет к оценке сложности алгоритма $5n + o(n)$.

Для того, чтобы получить более сильный результат, заметим (это следует из вышеприведенных рассуждений), что из гиперпары $\mathcal{H}_{2p}$ можно также выделить ч.у.м., состоящее из центра и $l \leqslant 2^p - 1$ элементов, меньших (больших) его, удалив не более $2p + 2.5l$ ребер.

Теперь, прежде чем выделять из гиперпары ч.у.м. будем выполнять сравнения центра с элементами, не принадлежащими гиперпаре, до тех пор, пока не наберется $k = 2^p - 1$ таких элементов, больших или меньших центра.

В ч.у.м. $\mathcal{X}_k$, выделяемое из гиперпары, включим все добавленные последним способом элементы, остальные выберем непосредственно из гиперпары. Число сравнений, приходящихся на изготовление одного такого ч.у.м., можно оценить как $k + l$ (число элементов, добавленных вторым способом) плюс $2p + 2.5(k - l)$ (число сравнений, требуемых для восстановления гиперпары — формально, они относятся к изготовлению следующего ч.у.м., но при суммировании это несущественно), т.е. всего не более $3.5k + o(k)$.

С учетом $I_k = (k+1)^2 - 1$, $J_k = (k+1)^2 + 2k$ и $C_k \sim 3.5k$ устанавливаем справедливость теоремы. $\blacksquare$

## 2.4 Быстрый алгоритм множественного выбора

Вернемся к общей задаче выбора элементов ранга $r_1, \ldots, r_k$ в $n$-элементном множестве, сформулированной во введении. Полагаем $1 \leqslant r_1 < r_2 < \ldots < r_k \leqslant n$.

Обозначим $\Delta_i = r_i - r_{i-1} - 1$. Число линейших расширений соответствующего решению задаче выбора ч.у.м. $\mathcal{P}$ равно $e(\mathcal{P}) = \prod_{i=1}^{k+1} \Delta_i!$, см. (2.1). Поэтому

$$\mathsf{C}_{r_1,\ldots,r_k}(n) \geqslant B_{r_1,\ldots,r_k}(n) = \log_2(n!) - \sum_{i=1}^{k+1} \log_2(\Delta_i!) = n\log_2 n - \sum_{i=1}^{k+1} \Delta_i \log_2 \Delta_i - O(n) \tag{2.3}$$

согласно следствию 2.1.

Очевидно, $\mathsf{C}_{r_1,\ldots,r_k}(n) = \mathsf{C}_{n+1-r_1,\ldots,n+1-r_k}(n)$, потому что любому дереву сравнений, осуществляющему выбор элементов ранга $r_i$, можно сопоставить двойственное дерево, выполняющее выбор элементов порядка $n+1-r_i$ (оно выбирает элементы ранга $r_i$ с точки зрения отношения порядка $\geqslant$).

Пример выбора среднего элемента показывает, что информационно-теоретическая нижняя оценка, в данном случае равная $B_{n/2}(n) \sim n$, не обязательно дает правильную асимптотику сложности задачи. Однако при больших значениях $B$ оценка (2.3) достижима в асимптотическом смысле. Справедлива

**Теорема 2.6** (К. Калигоси, К. Мельхорн, Дж. Мунро, П. Сандерс)**.**

$$\mathsf{C}_{r_1,\ldots,r_k}(n) \leqslant (1 + o(1))B_{r_1,\ldots,r_k}(n) + O(n).$$

Ослабленный вариант оценки теоремы $O(B + n)$ предлагается доказать в упр. 2.3.

▶ Положим $l = \max\{2, \lceil B/n \rceil\}$. Рассмотрим следующую рекурсивную процедуру.

Вход: семейство $C$ упорядоченных цепочек длины $< 2l$ каждая и множество рангов $R = \{r_1, \ldots, r_k\}$.

Выход: элементы ранга $r_1, \ldots, r_k$.

Алгоритм.

$(i)$. Если общее число элементов в $C$ не превосходит $8l^2$, то выполняется полная сортировка, при этом определяются все искомые элементы.

$(ii)$. Назовем упорядоченную цепочку *короткой*, если ее длина меньше $l$. Если в семействе $C$ найдутся две коротких цепочки одинаковой длины, то выполним их слияние. Будем выполнять такие слияния до тех пор, пока длины всех коротких цепочек не станут различными. Построенное семейство обозначим через $D$.

$(iii)$. Определим медиану (средний элемент) медиан цепочек из $D$. Обозначим ее через $m$.

$(iv)$. Каждую цепочку из $D$ разобьем на две «точкой» $m$. В одну часть (младшую) определим элементы, не превосходящие $m$, а в другую (старшую) — остальные элементы. Обозначим через $r$ ранг элемента $m$ — он становится известным после выполнения разбиений.

$(v)$. Выполним рекурсивный вызов процедуры для семейства $C'$ младших цепочек и множества рангов $R' = R \cap [1,\, r]$ (если $R'$ не пусто), а также для семейства $C''$ старших цепочек и множества рангов $R'' = (R \setminus R') - r$ (если $R''$ не пусто).

При решении общей задачи первоначальный вызов алгоритма выполняется с множеством из $n$ отдельных элементов. Корректность алгоритма очевидна: при каждом рекурсивном вызове число элементов сокращается.

Обозначим через $I(C) = \sum_{c \in C} |c| \log_2 |c|$ информационную емкость семейства $C$, где $|c|$ обозначает длину цепочки $c$. Пусть $T$ — дерево рекурсии алгоритма, и $v$ — произвольная его вершина. Введем дополнительные обозначения, привязанные к вершине дерева:

$I_v = I(C)$ — информационная емкость входного семейства;

$p_v = |C|$ — число цепочек на входе;

$n_v$ — общее число элементов в $C$;

$J_v = I(D)$ — информационная емкость после слияний;

$q_v = |D|$ — число цепочек в семействе $D$;

$I'_v = I(C')$, $I''_v = I(C'')$, — информационная емкость выходных семейств;

$p'_v = |C'|$, $p''_v = |C''|$ — число цепочек на выходах;

$\alpha_v n_v$ — общее число элементов в семействе $C'$.

Оценим сложность алгоритма по шагам. Шаг $(i)$ выполняется в концевых вершинах дерева для непересекающихся множеств элементов, поэтому его общая сложность по всем вызовам равна $O(n \log l)$.

**Утверждение 2.5.** *Число сравнений, выполняемых на этапе слияний в вершине $v$, не превосходит $J_v - I_v + q_v - p_v$.*

▷ Одно слияние цепочек длины $s$ требует не более

$$2s - 1 = 2s \log_2(2s) - 2s \log_2 s + 1 - 2$$

сравнений согласно лемме 1.3. □

Напомним, что $H(x) = -x \log_2 x - (1 - x) \log_2(1 - x)$ — это функция двоичной энтропии, определенная на отрезке $[0,\, 1]$. Легко видеть, что функция $H(x)$ симметрична относительно точки $x = 1/2$, в которой принимает максимальное значение 1. Кроме того, функция является выпуклой (вверх): $H''(x) < 0$.

**Утверждение 2.6.** $J_v \leqslant I'_v + I''_v + n_v H(\alpha_v)$.

▷ Пусть цепочка $d_i$ семейства $D = \{d_1, \ldots, d_q\}$ распадается на цепочки $d'_i \in C'$ и

$d''_i \in C''$. Обозначим $\alpha_i = |d'_i|/|d_i|$. Получаем[1]

$$\begin{aligned} J_v - I'_v - I''_v &= \sum_{i=1}^{q} (|d_i| \log_2 |d_i| - |d'_i| \log_2 |d'_i| - |d''_i| \log_2 |d''_i|) \\ &= \sum |d_i| \left(\log_2 |d_i| - \alpha_i \log_2(\alpha_i |d_i|) - (1-\alpha_i) \log_2((1-\alpha_i)|d_i|)\right) \\ &= \sum |d_i| H(\alpha_i) \leqslant n_v H\left(\sum \alpha_i |d_i|/n_v\right) = n_v H(\alpha_v). \end{aligned}$$

Последнее неравенство справедливо в силу выпуклости функции энтропии. □

Проверим, что рекурсивное разбиение на две подзадачи достаточно сбалансировано.

**Утверждение 2.7.** *При $n_v \geqslant 8l^2$ выполнено $\frac{1}{16} \leqslant \alpha_v \leqslant \frac{15}{16}$.*

▷ Оценим снизу число элементов заведомо выше медианы $m$. По построению, $q_v > n_v/(2l) \geqslant 4l$. В наихудшем случае выше $m$ находятся медианы самых коротких цепочек, среди которых могут быть по одной цепочке длины $1, \ldots, l-1$. Остальные цепочки содержат не менее

$$(q_v/2 - l) \cdot l/2 > (n_v - 4l^2)/8 > n_v/16$$

элементов. Аналогично оценивается число элементов снизу от $m$. □

**Утверждение 2.8.** *Для непустого множества рангов $R = \{r_1, \ldots, r_k\}$ справедливо*

$$\tau(n, R) = \sum_{v \in T;\ \deg v > 1} n_v H(\alpha_v) \leqslant n \log_2 n - \sum_{i=1}^{k+1} \Delta_i \log_2 \Delta_i + r_k - r_1 + 16n$$

*(суммирование в левой части выполняется по внутренним вершинам дерева рекурсии $T$).*

▷ Неравенство докажем индукцией по глубине дерева. При $n < 8l^2$ имеем $\tau(n, R) = 0 \leqslant 16n$, и доказывать нечего.

Докажем индуктивный переход. Пусть задача с параметрами $n, k, R$ разбивается на подзадачи с параметрами $n', k', R'$ и $n'', k'', R''$ согласно рассматриваемому алгоритму, где

$$n' = \alpha n, \quad n'' = (1-\alpha)n, \quad k = k' + k'', \quad R' = R \cap [1,\, n'], \quad R'' = (R \setminus R') - n'.$$

Поскольку заведомо $k > 0$, не ограничивая общности, можно полагать $k' > 0$.

[1] Как обычно, полагая $0 \log 0 = 0$.

**1.** Рассмотрим случай $k'' > 0$. Обозначим $\Delta' = n' - r_{k'}$ и $\Delta'' = r_{k'+1} - n' - 1$ — это длины частей, на которые разбивается интервал $(r_{k'}, r_{k'+1})$. Используя предположение индукции, получаем

$$\begin{aligned}\tau(n, R) &= nH(\alpha) + \tau(n', R') + \tau(n'', R'') \\ &\leqslant nH(\alpha) + n' \log_2 n' + n'' \log_2 n'' - \sum_{i=1}^{k+1} \Delta_i \log_2 \Delta_i \\ &+ \Delta_{k'+1} \log_2 \Delta_{k'+1} - \Delta' \log_2 \Delta' - \Delta'' \log_2 \Delta'' + r_k - r_{k'+1} + r_{k'} - r_1 + 16n \\ &\leqslant n \log_2 n - \sum_{i=1}^{k+1} \Delta_i \log_2 \Delta_i + r_k - r_1 + 16n\end{aligned}$$

в силу

$$H(\alpha) + \alpha \log_2(\alpha n) + (1-\alpha) \log_2((1-\alpha)n) = \log_2 n \tag{2.4}$$

и неравенства (следствия выпуклости вниз функции $x \log_2 x$)

$$(x+y) \log_2(x+y) - x \log_2 x - y \log_2 y \leqslant x + y, \tag{2.5}$$

применяемого с подстановкой $x = \Delta'$, $y = \Delta''$, $x + y = \Delta_{k'+1}$.

**2.** В случае $k'' = 0$, сохраняя обозначение $\Delta' = n' - r_k$, имеем

$$\begin{aligned}\tau(n, R) &= nH(\alpha) + \tau(n', R') \\ &\leqslant nH(\alpha) + n' \log_2 n' + n'' \log_2 n'' - \sum_{i=1}^{k+1} \Delta_i \log_2 \Delta_i \\ &+ \Delta_{k+1} \log_2 \Delta_{k+1} - \Delta' \log_2 \Delta' - n'' \log_2 n'' + r_k - r_1 + 16n' \\ &\leqslant n \log_2 n - \sum_{i=1}^{k+1} \Delta_i \log_2 \Delta_i + r_k - r_1 + 16n' + \Delta_{k+1}\end{aligned}$$

опять ввиду (2.4) и неравенства (2.5) при $x = \Delta'$, $y = n''$, $x + y = \Delta_{k+1}$. Осталось заметить, что $\Delta_{k+1} < n \leqslant 16n''$ в силу леммы 2.7. □

**Утверждение 2.9.**

$$\sum_{v \in T} (I'_v + I''_v - I_v + q_v - p_v) \leqslant n \log_2(2l). \tag{2.6}$$

▷ С учетом $q_v \leqslant p'_v + p''_v$ сумма в левой части (2.6) оценивается сверху как

$$-I_u - p_u + \sum_{v \in T;\ \deg(v)=1} (I_v + p_v).$$

Здесь суммирование выполняется по листьям дерева $T$, а $u$ — корневая вершина. По условию, $I_u = 0$ и $p_u = n$. Семейства, относящиеся к различным листьям дерева, попарно не пересекаются, поэтому $\sum p_v = n$. Наконец, поскольку все цепочки имеют длину менее $2l$, имеем $\sum I_v \leqslant n \log_2(2l)$. □

Совместно утверждения 2.8 и 2.9 дают оценку сложности слияний — шага $(ii)$ алгоритма. Перейдем к шагам $(iii)$ и $(iv)$.

После этапа слияний в вершине $v$ имеется не более $l$ коротких и $n_v/l$ длинных цепочек. Пусть выбор среднего из $k$ элементов выполняется за $\beta k$ сравнений. Вставка элемента (медианы) в цепочку требует не более $\log_2 l$ сравнений, т.к. длина половины любой цепочки меньше $l$. Поэтому с учетом $n_v \geqslant 8l^2$ сложность шагов $(iii), (iv)$ по всем внутренним вершинам дерева оценивается как

$$\sum_{v\in T;\ \deg v>1} (n_v/l + l)(\log_2 l + \beta) \asymp \frac{\log l}{l} \sum_{v\in T;\ \deg v>1} n_v.$$

Сумму в правой части можно оценить при помощи утверждения 2.8, т.к. $n_v \leqslant 3n_v H(\alpha_v)$ ввиду $H(\alpha_v) \geqslant H(1/16) > 1/3$ согласно утверждению 2.7.

Суммируя сложность всех шагов, окончательно получаем

$$\mathsf{C}_{r_1,\ldots,r_k}(n) \leqslant \left(1 + O\left(\frac{\log l}{l}\right)\right) B_{r_1,\ldots,r_k}(n) + O(n \log l).$$

Осталось заметить, что $n \log l = O(n + (B/l)\log l)$. ■

# Упражнения

**Упр. 2.1.** Сконструируйте алгоритм выбора элемента порядка $n/4$ из $n$-элементного множества, имеющий сложность асимптотически меньше $3n$. [*Д. Дор, У. Цвик*]

**Упр. 2.2.** Покажите, что если верна гипотеза Яо о том, что сложность выбора подмножества с частичным порядком $\mathcal{X}_{k,l}$ ($k$ элементов больше, а $l$ элементов меньше некоторого элемента) в $m$-элементном множестве одна и та же при любом $m \geqslant k + l + 1$, то $\mathsf{C}_{n/2}(n) \leqslant 2.5n + o(n)$. [*А. Шёнхаге, М. Патерсон, Н. Пиппенджер*]

**Упр. 2.3.** Докажите ослабленный вариант теоремы 2.6. Оцените сложность алгоритма, построенного рекурсивно путем выбора медианы и разбиения задачи пополам. [*Д. Добкин, Дж. Мунро*]

**Комментарии.** Материал разделов 2.1–2.2 представлен в [2], частично в виде упражнений (в том числе, приведено более простое доказательство теоремы 2.3 Кислицына, нежели в оригинальной работе [1]). Теорема 2.4 доказана в [3]. Теорема 2.5 доказана в [7] с лучшей оценкой $3n + o(n)$. Результат теоремы 2.6 получен в [6].

Чтобы усилить оценку теоремы 2.5 до $3n + o(n)$, на втором этапе построения ч.у.м. вместо сравнения центра с одиночными элементами следует выполнять сравнения с элементами упорядоченных пар, а также учитывать сравнения между элементами фрагментов ч.у.м., которые возвращаются на фабрику. При помощи весьма изощренной модификации метода Д. Дор и У. Цвик [5] усилили оценку далее до $2.95n + o(n)$. Они же в работе [4] незначительно уточнили нижнюю оценку теоремы 2.4 асимптотически до $(2 + \varepsilon)n$, где $\varepsilon \approx 2^{-70}$.

## Литература

[1] Кислицын С. С. *О выделении k-го элемента упорядоченной совокупности путем попарных сравнений.* Сиб. матем. журн. 1964. **5**(3), 557–564.

[2] Кнут Д. Э. *Искусство программирования. Т. 3. Сортировка и поиск.* М.: Вильямс, 2007.

[3] Bent S. W., John J. W. *Finding the median requires* $2n$ *comparisons.* Proc. STOC (Providence, USA, 1985). NY: ACM, 1985, 213–216.

[4] Dor D., Zwick U. *Median selection requires* $(2 + \epsilon)n$ *comparisons.* Proc. FOCS (Burlington, USA, 1996). Los-Alamitos: IEEE, 1996, 125–134.

[5] Dor D., Zwick U. *Selecting the median.* SIAM J. Comput. 1999. **28**(5), 1722–1758.

[6] Kaligosi K., Mehlhorn K., Munro J. I., Sanders P. *Towards optimal multiple selection.* Proc. ICALP (Lisbon, 2005). Berlin, NY: Springer, 2005, 103–114.

[7] Schönhage A., Paterson M., Pippenger N. *Finding the median.* J. Comp. Sys. Sci. 1976. **13**, 184–199.

# Глава 3

# Схемы компараторов

## 3.1 Введение

В этой главе рассматривается популярная модель *схем компараторов*[1]. Компаратором называется схема с двумя входами $x$, $y$ и двумя выходами, на которых вычисляются функции $\min(x, y)$ и $\max(x, y)$. Схема компараторов — это частный случай схемы из функциональных элементов. Ее элементами являются компараторы, при этом ветвление входов схемы и выходов компараторов запрещается (каждый вход схемы или выход компаратора используются только по разу). Ввиду этого ограничения число выходов схемы совпадает с числом входов. В результате вычисления на выходе схемы реализуется некоторое переупорядочение входного набора. Важнейшее свойство схем компараторов: выбор элементов для очередного сравнения не зависит от результатов предыдущих сравнений. Схема компараторов соответствует частному случаю дерева сравнений, но в отличие от дерева сравнений общего вида ее удобно реализовать в электронной схеме.

Схемы компараторов удобно изображать в виде горизонтальных линий, слева на которые поступают входы, а справа возникают выходные значения. Компараторы изображаются в виде перемычек, соединяющих пару линий. По перемычке наверх отправляется меньший из поступивших на вход компаратора элементов, а вниз — больший. Пример схемы показан на рис. 3.1.

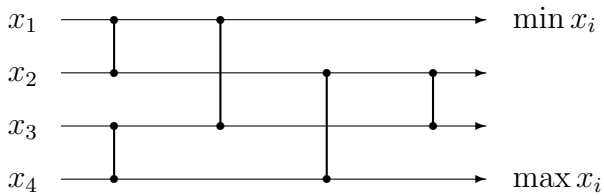



Рис. 3.1: Схема компараторов для сортировки 4 элементов

[1] В русском переводе книги Д. Кнута такие схемы называются сетями сортировки.

Сложность схемы $\Sigma$ — число компараторов в ней — будем обозначать через $\mathsf{S}^*(\Sigma)$. Через $\mathsf{S}^*(n)$ обозначим минимальную сложность схемы сортировки $n$ элементов. По определению, $\mathsf{S}^*(n) \geqslant \mathsf{S}(n)$.

Другой важный показатель эффективности схемы — ее *глубина*. Это число слоев из параллельно расположенных (т.е. независимых по входам) компараторов. Альтернативным образом, как в случае обычных схем из функциональных элементов, глубину можно определить как максимальное число компараторов в цепочке, ведущей от входа к выходу схемы. Для обозначения глубины схемы будем использовать символ $\mathsf{D}$. Схема на рис. 3.1 имеет сложность 5 и глубину 3.

Для анализа схем сортировки (в общем случае, ч.у.м.) полезен известный *принцип нулей и единиц*.

**Лемма 3.1** (Д. Кнут). *Схема компараторов правильно упорядочивает ч.у.м. на любом входном наборе, если она правильно работает на любом наборе из нулей и единиц.*

▷ Поскольку компаратор переводит вектор входов $(x, y)$ в вектор $(\min\{x, y\}, \max\{x, y\})$, то для любой монотонной функции $f$ он переводит вектор $(f(x), f(y))$ в $(f(\min\{x, y\}), f(\max\{x, y\}))$. Следовательно, если схема компараторов переводит вектор входов $(x_1, \ldots, x_n)$ в вектор $(y_1, \ldots, y_n)$, то вектор $(f(x_1), \ldots, f(x_n))$ она переводит в $(f(y_1), \ldots, f(y_n))$.

Пусть схема компараторов переводит некоторый входной набор $(x_1, \ldots, x_n)$ в выходной набор $(y_1, \ldots, y_n)$, в котором порядок нарушен ввиду, скажем, $y_i > y_{i+1}$. Определим функцию $f : \mathbb{R} \to \{0, 1\}$ как $f(x) = (x \geqslant y_i)$. Тогда получаем, что схема не упорядочивает ч.у.м. при входном булевом векторе $(f(x_1), \ldots, f(x_n))$. □

## 3.2 Схемы слияния

Ограничения модели схем компараторов проявляются при попытках построить эффективные схемы сортировки. Простые подходы, на которых основаны быстрые алгоритмы сортировки, не приводят к оптимальным решениям для схем компараторов. Это можно наглядно продемонстрировать на примере задачи слияния упорядоченных наборов. Напомним, что в модели без ограничений слияние имеет линейную сложность (см. лемму 1.3).

**Теорема 3.1** (П. Милтерсен, М. Патерсон, Ю. Таруи). $\mathsf{S}^*(\mathcal{L}_{n,n}) > n \log_2 n - O(n)$.

► Пусть входами схемы являются упорядоченные наборы $x_1 \leqslant \ldots \leqslant x_n$ и $y_1 \leqslant \ldots \leqslant y_n$, а выходами — $z_1 \leqslant \ldots \leqslant z_{2n}$.

На множестве $\Pi_{n,n}$ согласованных с частичным порядком перестановок $\pi : \{x_1, \ldots, x_n, y_1, \ldots, y_n\} \to \{z_1, \ldots, z_{2n}\}$ рассмотрим некоторое вероятностное распределение. Тогда $\sigma_k = \pi^{-1}(z_k)$ — случайная величина, принимающая значения в множестве входных переменных.

Предварительно напомним известный факт из теории информации. Пусть $\sigma$ — случайная величина, принимающая значения в конечном множестве $A = \{a_i\}$, причем $\mathbf{P}[\sigma = a_i] = p_i$. *Энтропия* (информационная энтропия, энтропия по Шеннону) величины $\sigma$ определяется как $H(\sigma) = -\sum p_i \log_2 p_i$. *Префиксный код* — это такое отображение множества $A$ в множество слов некоторого алфавита, что ни одно кодовое слово не является началом другого.

**Лемма 3.2** (К. Шеннон)**.** *При любом префиксном кодировании элементов множества $A$ двоичными словами, $a_i \to c(a_i)$, справедливо $\mathbf{E}[|c(\sigma)|] \geqslant H(\sigma)$, где $|b|$ — длина слова $b$.*

▷ Рассмотрим оптимальное кодирование $c$, доставляющее минимум средней длины символа $\mathbf{E}[|c(\sigma)|]$. Применим индукцию по величине множества $A$.

В случае $|A| = 2$ всегда оптимален однобитный код, для которого очевидно выполнено неравенство $p_1 + p_2 = 1 \geqslant H(\sigma) = H(p_1)$: здесь энтропия случайной величины совпадает с функцией двоичной энтропии одной переменной.

Теперь заметим, что при оптимальном кодировании найдутся два символа, коды которых отличаются только в последней позиции. В противном случае из самого длинного кодового слова последнюю цифру можно было бы удалить.

При $|A| = n > 2$ будем считать, что коды символов $a_{n-1}$ и $a_n$ отличаются ровно в последней позиции. Отождествляя эти символы ($a_n := a_{n-1}$), перейдем к алфавиту $A'$ мощности $n-1$, случайной величине $\sigma'$, отличающейся от $\sigma$ тем, что принимает значение $a_{n-1}$ с вероятностью $p_{n-1} + p_n$, и коду $c'$, в котором для кодирования символа $a_{n-1}$ будем использовать общую часть кода двух отождествляемых символов. Тогда в силу индуктивного предположения

$$\mathbf{E}[|c(\sigma)|] = p_{n-1} + p_n + \mathbf{E}[|c'(\sigma')|] \geqslant p_{n-1} + p_n + H(\sigma') =$$
$$H(\sigma) + p_{n-1} + p_n + (p_{n-1} + p_n)\log_2(p_{n-1} + p_n) - p_{n-1}\log_2(p_{n-1}) - p_n \log_2(p_n).$$

Вследствие неравенства (2.5), выражение в правой части не меньше $H(\sigma)$. □

Наша ближайшая цель — оценить число сегментов $R$, на которые разбиваются горизонтальные линии компараторами в минимальной схеме слияния $S$. Каждый сегмент проводит некоторую входную (или выходную, смотря с какой стороны смотреть) переменную. Обозначим через $P_i^\pi$ путь (множество сегментов), который проходит входная переменная до выхода $z_i$ при перестановке $\pi$. Каждый такой путь можно задать двоичным кодом: просматривая путь справа налево, всякий раз при достижении компаратора нулем определять продолжение через верхний вход компаратора, а единицей — через нижний.

Тогда каждому возможному значению $x$ случайной величины $\sigma_k$ можно поставить в соответствие код $C_k(x)$ кратчайшего пути, проходящего переменной $x$ до выхода $z_k$ при некоторой подходящей перестановке из $\Pi_{n,n}$. При любом $k$ множество слов $C_k(x)$ будет префиксным кодом, так как пути имеют разные начальные

(конечные с точки зрения кода) точки. Теперь, используя лемму 3.2, для случайной перестановки $\pi$ получаем

$$R = \sum_{k=1}^{2n} \mathbf{E}[|P_k^\pi|] \geqslant 2n + \sum_{k=1}^{2n} \mathbf{E}[|C_k(\sigma_k)|] \geqslant 2n + \sum_{k=1}^{2n} H(\sigma_k). \tag{3.1}$$

Осталось построить вероятностное распределение на множестве $\Pi_{n,n}$, обеспечивающее высокую нижнюю оценку.

Удобно заметить, что возможные перестановки в задаче слияния взаимно однозначно соответствуют монотонным путям в целочисленной решетке на вершинах с координатами от 0 до $n$, см. рис. 3.2. Пути начинаются в левом нижнем углу и заканчиваются в верхнем правом. Если очередной элемент по порядку относится к группе $\{x_i\}$, то совершаем перемещение вправо, если к группе $\{y_j\}$, то вверх[2]. Наоборот, по пути легко восстановить перестановку.

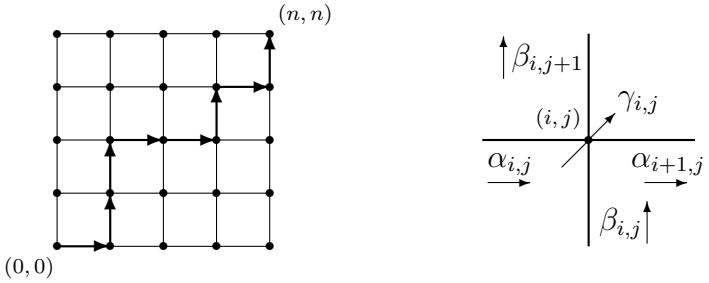



Рис. 3.2: Перестановки и монотонные пути на решетке

Далее, через $\alpha_{i,j}$ и $\beta_{i,j}$ обозначим вероятности того, что путь проходит через ребро $((i-1,\, j),\, (i,\, j))$ и соответственно через $((i,\, j-1),\, (i,\, j))$, см. рис. 3.2. Положим $\gamma_{i,j} = \alpha_{i,j} + \beta_{i,j}$ — вероятность посещения путем вершины $(i,\, j)$.

По построению, $\alpha_{i,j} = \mathbf{P}[z_{i+j} = x_i]$ и $\beta_{i,j} = \mathbf{P}[z_{i+j} = y_j]$. Во всех вершинах $(i,\, j)$, кроме начальной и конечной, выполнено

$$\alpha_{i,j} + \beta_{i,j} = \alpha_{i+1,j} + \beta_{i,j+1} = \gamma_{i,j}. \tag{3.2}$$

Для крайних вершин выполняется

$$1 = \gamma_{0,0} = \alpha_{1,0} + \beta_{0,1} = \alpha_{n,n} + \beta_{n,n} = \gamma_{n,n}. \tag{3.3}$$

Условия (3.2), (3.3) задают *единичный поток* (через решетку). Таким образом, вероятностное распределение на $\Pi_{n,n}$ определяет поток. Покажем, что соответствие между потоками и распределениями взаимно-однозначно.

**Утверждение 3.3.** *Любой единичный поток, т.е. система неотрицательных величин $\alpha_{i,j}$, $\beta_{i,j}$, удовлетворяющая условиям* (3.2), (3.3), *соответствует некоторому вероятностному распределению на множестве $\Pi_{n,n}$.*

[2]Изображенный на рис. 3.2 путь соответствует перестановке $(x_1, y_1, y_2, x_2, x_3, y_3, x_4, y_4)$.

▷ Случайный путь строится по правилу: из вершины $(i,\,j)$ выполняется перемещение направо с вероятностью $\alpha_{i+1,j}/\gamma_{i,j}$, и вверх — с вероятностью $\beta_{i,j+1}/\gamma_{i,j}$. Теперь индукцией по $i+j$ легко проверяется, что вероятность посещения вершины $(i,\,j)$ равна $\gamma_{i,j}$, откуда следует, что предложенное вероятностное распределение определяет заданный поток. □

Определим поток следующими правилами: для $0 < i + j \leqslant n$ положим

$$\alpha_{i,j} = \beta_{j,i} = \alpha_{n+1-i,\,n-j} = \beta_{n-j,\,n+1-i} = \frac{i}{(i+j)(i+j+1)}$$

(коэффициенты симметричны относительно диагоналей $x = y$ и $x + y = n$). Легко проверить, что условия (3.2), (3.3) при этом выполнены[3].

Замечая, что $H(\sigma_k) = -\sum_{i=1}^{k} \alpha_{i,k-i} \log_2 \alpha_{i,k-i} - \sum_{i=1}^{k} \beta_{k-i,i} \log_2 \beta_{k-i,i}$, с учетом симметрии коэффициентов получаем

$$\begin{aligned}
\sum_{k=1}^{2n} H(\sigma_k) &= -4 \sum_{k=1}^{n} \sum_{i=1}^{k} \alpha_{i,k-i} \log_2 \alpha_{i,k-i} = -4 \sum_{k=1}^{n} \frac{1}{k(k+1)} \sum_{i=1}^{k} i \log_2 \frac{i}{k(k+1)} \\
&= 4 \sum_{k=1}^{n} \frac{\log_2(k(k+1))}{k(k+1)} \sum_{i=1}^{k} i - 4 \sum_{i=1}^{n} i \log_2 i \cdot \sum_{k=i}^{n} \frac{1}{k(k+1)} \\
&= 2 \sum_{k=1}^{n} \log_2(k(k+1)) - 4 \sum_{i=1}^{n} \left(1 - \frac{i}{n+1}\right) \log_2 i \\
&= 4 \log_2(n!) + 2 \log_2(n+1) - 4 \log_2(n!) + \frac{4}{n+1} \sum_{i=1}^{n} i \log_2 i \\
&\geqslant 2 \log_2(n+1) + \frac{4}{n+1} \int_1^n x \log_2 x \, dx \\
&= 2 \log_2(n+1) + \frac{1}{n+1} \cdot x^2 (2 \log_2 x - \log_2 e) \Big|_{x=1}^{n} \\
&= 2 \log_2(n+1) + 2 \frac{n^2}{n+1} \log_2 n - (n-1) \log_2 e > 2n \log_2 n - n \log_2 e.
\end{aligned}$$

Поскольку каждый компаратор добавляет в схему два сегмента, из (3.1) окончательно выводим

$$\mathsf{S}^*(\mathcal{L}_{n,n}) = (R - 2n)/2 > n \log_2 n - 0.5n \log_2 e.$$

■

Асимптотическую точность полученной оценки демонстрирует простой метод Бэтчера.

[3]При равномерном распредении почти наверное случайный путь пройдет близко вдоль диагонали $x = y$. Предложенное распределение повышает вероятность удаления пути от диагонали и равномерно распределяет пути вдоль диагоналей $x + y = k$. Вероятность посещения любой точки решетки на диагонали $x + y = k$ одинакова и равна $1/(\min\{k, 2n - k\} + 1)$.

**Теорема 3.2** (К. Бэтчер). $\mathsf{S}^*(\mathcal{L}_{n,n}) \leqslant n\lceil\log_2 n\rceil + n$.

► Построим рекурсивно схемы слияния $M_n$ упорядоченных наборов длины $n$. На входы схемы $M_n$ подаются наборы $x_1 \leqslant x_2 \leqslant \ldots \leqslant x_n$ и $y_1 \leqslant y_2 \leqslant \ldots \leqslant y_n$. Схема устроена следующим образом. Подсхемы $M_{\lceil n/2\rceil}$ и $M_{\lfloor n/2\rfloor}$ сортируют группы элементов соответственно с нечетными и четными индексами, т.е. $x_1, y_1, x_3, y_3, \ldots$ и $x_2, y_2, x_4, y_4, \ldots$ Результат их работы: упорядоченные наборы $u_1 \leqslant u_2 \leqslant \ldots \leqslant u_{2\lceil n/2\rceil}$ и соответственно $v_1 \leqslant v_2 \leqslant \ldots \leqslant v_{2\lfloor n/2\rfloor}$. Заметим, что $u_i \leqslant v_i$, поскольку каждому элементу с четным индексом можно сопоставить предшествующий элемент с нечетным индексом. Таким образом,

$$u_1 \leqslant v_1, u_2 \leqslant v_2, u_3 \leqslant \ldots$$

Поэтому для завершения сортировки достаточно выполнить сравнения в парах $v_i, u_{i+1}$. Получаем соотношение

$$\mathsf{S}^*(\mathcal{L}_{n,n}) \leqslant \mathsf{S}^*(\mathcal{L}_{\lceil n/2\rceil,\lceil n/2\rceil}) + \mathsf{S}^*(\mathcal{L}_{\lfloor n/2\rfloor,\lfloor n/2\rfloor}) + n - 1,$$

которое с учетом $\mathsf{S}^*(\mathcal{L}_{1,1}) = 1$ разрешается так, как заявлено в утверждении теоремы. ■

Попытка с помощью схем слияния построить схему сортировки, как в теореме 1.2, приводит лишь к схемам сложности $O(n\log^2 n)$, хотя они вполне практичны. Построение схем сложности $O(n\log n)$ потребовало привлечения нетривиального математического аппарата.

## 3.3 Нижняя оценка сложности сортировки

Теоретико-информационная нижняя оценка $\mathsf{S}(n) \geqslant \log_2 n! \sim n\log_2 n$, конечно, справедлива и для схем компараторов. Но с учетом ограничений вычислительной модели схем, она может быть усилена.

*Подходящей потенциальной функцией* будем называть непрерывную функцию $f(x) : [0,\, 1] \to \mathbb{R}$, для которой выполняются условия:

(1) $f(0) = f(1) = 0$;

(2) При любых $p, q, r$, таких что $0 \leqslant pq \leqslant r \leqslant p \leqslant q \leqslant 1$, справедливо

$$f(p) + f(q) - f(r) - f(p + q - r) \leqslant p + q - 2r. \tag{3.4}$$

**Теорема 3.3** (Н. Кахале, Т. Лейтон, Ю. Ма, Г. Плакстон, Т. Сьюэл, Э. Семереди). *Если $f(x)$ — подходящая потенциальная функция, то $\mathsf{S}^*(n) \geqslant (f(1/2) - o(1))n\log_2 n$.*

► Рассмотрим произвольную схему $\Sigma$ из множества $\mathbf{\Sigma_n}$ $n$-входовых сортирующих схем и некоторый входной набор $\alpha \in \{0,\, 1\}^n$. Обозначим через $a(\Sigma, \alpha)$, $b(\Sigma, \alpha)$ и

$c(\Sigma, \alpha)$ число компараторов в схеме, значения входов которых: оба равны 1, оба равны 0 и различны соответственно. Такие компараторы будем далее называть 11-компараторами, 00-компараторами и 01-компараторами. Далее $|\alpha|$ обозначает вес булева вектора $\alpha$.

**Лемма 3.4.** *Пусть* $\alpha \in \{0, 1\}^n$ . *Тогда если* $\Sigma \in \mathbf{\Sigma_n}$, *то* $a(\Sigma, \alpha) \geqslant \mathsf{S}^*(|\alpha|)$ *и* $b(\Sigma, \alpha) \geqslant \mathsf{S}^*(n - |\alpha|)$.

$\triangleright$ Докажем первое неравенство (второе доказывается аналогично). Для этого покажем, что из множества 11-компараторов схемы $\Sigma$ можно составить схему сортировки $k$ элементов.

Удалим из схемы входы, соответствующие нулям набора $\alpha$. Далее последовательно, передвигаясь от входов к выходам, удалим 00-компараторы вместе с ребрами, выходящими из них, а также 01-компараторы, соединяя «единичный» вход компаратора с выходом, на котором вычисляется максимум и удаляя ребро, ведущее из выхода, вычисляющего минимум. В итоге будет построена схема компараторов $\Sigma_1$ с $k$ входами и сложности в точности $a(\Sigma, \alpha)$.

При этом получается естественное взаимно-однозначное соответствие между:

– входами схемы $\Sigma_1$ и подмножеством входов (отвечающих единицам набора $\alpha$) схемы $\Sigma$;

– компараторами схемы $\Sigma_1$ и 11-компараторами схемы $\Sigma$;

– выходами схемы $\Sigma_1$ и подмножеством выходов, вычисляющих максимальные $k$ значений, в схеме $\Sigma$.

Теперь рассмотрим произвольный набор $\beta$, в котором максимальные $k$ элементов находятся на позициях единиц набора $\alpha$. Подавая на соответствующие входы схем $\Sigma$ и $\Sigma_1$ соответствующие элементы набора $\beta$, получим, что входы и выходы соответствующих компараторов в обеих схемах принимают одни и те же значения, а поэтому одни и те же значения принимают соответствующие выходы обеих схем. Поскольку соответствующее подмножество выходов схемы $\Sigma$ содержит упорядочение соответствующих элементов набора $\beta$, то же верно и в отношении выходов схемы $\Sigma_1$. Поэтому в силу произвольности выбора $\beta$ заключаем, что $\Sigma_1$ является схемой сортировки. $\square$

Обозначим

$$C(n) = \min_{\Sigma \in \mathbf{\Sigma_n}} \max_{\alpha \in \{0,1\}^n} c(\Sigma, \alpha)$$

**Лемма 3.5.**

$$\mathsf{S}^*(n) \geqslant \min_{0<i<n} (\mathsf{S}^*(i) + \mathsf{S}^*(n-i)) + C(n).$$

$\triangleright$ Выберем в качестве $\Sigma$ схему, доставляющую минимум $\mathsf{S}^*(n)$. По определению $C(n)$ существует вектор $\alpha \in \{0, 1\}^n$, такой, что $c(\Sigma, \alpha) \geqslant C(n)$. Пусть $k = |\alpha|$.

Тогда по лемме 3.4

$$\begin{aligned}\mathsf{S}^*(n) &= a(\Sigma,\alpha)+b(\Sigma,\alpha)+c(\Sigma,\alpha)\\ &\geqslant \mathsf{S}^*(k)+\mathsf{S}^*(n-k)+C(n) \geqslant \min_{0<i<n}(\mathsf{S}^*(i)+\mathsf{S}^*(n-i))+C(n).\end{aligned}$$

□

**Утверждение 3.6.** *Если* $C(n) \geqslant (\gamma - o(1))n$, *то* $\mathsf{S}^*(n) \geqslant (\gamma - o(1))n\log_2 n$.

▷ Покажем, что при любом $\varepsilon > 0$ выполнено $\mathsf{S}^*(n) \geqslant (\gamma-\varepsilon)n\log_2 n$ при всех $n \geqslant n_0(\varepsilon)$.

Пусть при $n \geqslant n_1$ справедливо $C(n) \geqslant (\gamma - \varepsilon/2)n$, а константа $c_\varepsilon$ подобрана так, что для $n \leqslant n_1$ выполнено

$$\mathsf{S}^*(n) \geqslant (\gamma-\varepsilon/2)n\log_2 n - c_\varepsilon n. \tag{3.5}$$

Докажем (3.5) по индукции для любого $n > n_1$.

Пусть $k$ доставляет минимум величины $\mathsf{S}^*(k)+\mathsf{S}^*(n-k)$. Обозначая $\gamma' = \gamma-\varepsilon/2$, согласно лемме 3.5 и индуктивному предположению получаем

$$\begin{aligned}\mathsf{S}^*(n) &\geqslant \mathsf{S}^*(k)+\mathsf{S}^*(n-k)+\gamma' n\\ &\geqslant \gamma' k\log_2 k+\gamma'(n-k)\log_2(n-k)+\gamma' n - c_\varepsilon k - c_\varepsilon(n-k) \geqslant \gamma' n\log_2 n - c_\varepsilon n,\end{aligned}$$

где в последнем переходе мы воспользовались неравенством (2.5).

Теперь можно выбрать $n_0$, исходя из условия $(\varepsilon/2)n_0\log_2 n_0 \geqslant c_\varepsilon n_0$. □

В силу доказанного нам остается получить хорошую нижнюю оценку для $C(n)$. Вместо $C(n)$ удобнее рассматривать усредненную величину

$$\bar{C}(n) = \min_{\Sigma\in\mathbf{\Sigma_n}}\left(\frac{1}{2^n}\sum_{\alpha\in\{0,1\}^n} c(\Sigma,\alpha)\right).$$

Очевидно, $C(n) \geqslant \bar{C}(n)$.

Пусть $h(x_1,\ldots,x_n)$ — булева функция. Будем обозначать через $|h|$ число наборов, на которых функция равна 1. Воспользуемся следующим известным результатом из области комбинаторики булева куба:

**Лемма 3.7.** *Пусть* $u, v$ — *монотонные булевы функции* $n$ *переменных. Тогда* $2^n|uv| \geqslant |u||v|$.

Это утверждение вытекает из более общего факта. Для функции $\varphi(x) : \{0,1\}^n \to \mathbb{R}_{\geqslant 0}$ введем обозначение $|\varphi| = \sum_{x\in\{0,1\}^n}\varphi(x)$. Справедлива

**Теорема 3.4** (Р. Альсведе, Д. Дайкин)**.** *Пусть $\alpha, \beta, \gamma$ — такие неотрицательные функции, что при любых $x, y \in \{0,1\}^n$ выполняется*

$$\alpha(x)\beta(y) \leqslant \gamma(x \vee y),$$

*где $\vee$ обозначает поразрядную дизъюнкцию двоиных векторов. Тогда $|\alpha||\beta| \leqslant 2^n|\gamma|$.*

Заметим, что лемма 3.7 следует из этой теоремы при подстановке $\alpha = u$, $\beta = v$, $\gamma = uv$, т.к. в силу монотонности $u(x \vee y)v(x \vee y) \geqslant u(x)v(y)$, а два определения для $|\varphi|$ совпадают в случае монотонной булевой функции.

► Теорему докажем индукцией по $n$. Пусть $n = 1$. Для любой функции $\varphi \in \{\alpha, \beta, \gamma\}$ обозначим $\varphi_x = \varphi(x)$, $x \in \{0,1\}$. По условию теоремы, имеем

$$\alpha_0\beta_0 \leqslant \gamma_0, \qquad \alpha_0\beta_1, \alpha_1\beta_0, \alpha_1\beta_1 \leqslant \gamma_1.$$

Доказываемое соотношение в этом случае имеет вид

$$(\alpha_0 + \alpha_1)(\beta_0 + \beta_1) \leqslant 2(\gamma_0 + \gamma_1).$$

Если $\gamma_1 = 0$, то оно очевидно. Иначе, из

$$(\gamma_1 - \alpha_0\beta_1)(\gamma_1 - \alpha_1\beta_0) \geqslant 0$$

следует

$$(\alpha_0 + \alpha_1)(\beta_0 + \beta_1)\gamma_1 \leqslant (\gamma_1 + \alpha_0\beta_0)(\gamma_1 + \alpha_1\beta_1).$$

Деля обе части на $\gamma_1 > 0$, получаем

$$(\alpha_0 + \alpha_1)(\beta_0 + \beta_1) \leqslant (\gamma_1 + \alpha_0\beta_0)\left(1 + \frac{\alpha_1\beta_1}{\gamma_1}\right) \leqslant (\gamma_0 + \gamma_1)(1+1),$$

что и требовалось.

Рассмотрим индуктивный переход от $n-1$ к $n$. Вектор $x \in \{0,1\}^n$ будем записывать в виде $\bar{x}, x^*$, где $\bar{x} \in \{0,1\}^{n-1}$, а $x^* \in \{0,1\}$, т.е. выделяя один из разрядов $x^*$ вектора $x$. По функции $\varphi \in \{\alpha, \beta, \gamma\}$ определим функцию $\varphi'(y) = \varphi(y,0) + \varphi(y,1)$ при любом $y \in \{0,1\}^{n-1}$.

**1.** Покажем, что $\alpha'(y)\beta'(z) \leqslant 2\gamma'(y \vee z)$ при любых $y, z \in \{0,1\}^{n-1}$. Для этого, зафиксировав $y$ и $z$, рассмотрим функции $\bar{\alpha}(x) = \alpha(y,x)$, $\bar{\beta}(x) = \beta(z,x)$, $\bar{\gamma}(x) = \gamma(y \vee z, x)$, где $x \in \{0,1\}$. Для функций $\bar{\alpha}, \bar{\beta}, \bar{\gamma}$ выполнены условия теоремы в случае $n = 1$, для которого она уже доказана, поэтому

$$\alpha'(y)\beta'(z) = (\bar{\alpha}(0) + \bar{\alpha}(1))(\bar{\beta}(0) + \bar{\beta}(1)) \leqslant 2(\bar{\gamma}(0) + \bar{\gamma}(1)) = 2\gamma'(y \vee z).$$

**2.** Теперь можно применить предположение индукции к набору функций $\alpha', \beta', 2\gamma'$ и получить $|\alpha'||\beta'| \leqslant 2^{n-1}|2\gamma'|$, а в силу $|\varphi'| = |\varphi|$ — в качестве следствия требуемое соотношение $|\alpha||\beta| \leqslant 2^n|\gamma|$. ■

Напомним, что выход любой схемы компараторов реализует монотонную функцию относительно значений входов, в частности, монотонную булеву функцию, если значения входов выбираются из множества $\{0, 1\}$. Это следует из того, что функция, реализуемая на выходе, является композицией монотонных функций min и max, реализуемых отдельными компараторами.

Обозначим через $x_i$ и соответственно через $y_i$ вероятность того, что $i$-й вход (соответственно выход) схемы компараторов $\Sigma$ принимает значение 1 в предположении, что на входы схемы подаются равномерно распределенные на множестве $\{0, 1\}$ случайные величины. В этом случае вероятность события $X$ совпадает с отношением числа наборов, для которых $X$ выполняется, к числу всех наборов, $2^n$ для $n$ входов. Положим $f_I(\Sigma) = \sum_{i=1}^{n} f(x_i)$ и $f_O(\Sigma) = \sum_{i=1}^{n} f(y_i)$, где $f$ — подходящая потенциальная функция.

**Лемма 3.8.** *Пусть $\Sigma'$ — схема компараторов, получающаяся присоединением компаратора $e$ к выходам схемы $\Sigma$. Тогда, если $f$ — подходящая потенциальная функция, то вероятность того, что $e$ является 01-компаратором, не меньше, чем $f_O(\Sigma) - f_O(\Sigma')$.*

$\triangleright$ Пусть $p$ и $q$ — вероятности равенства единице каждого из входов компаратора $e$, а $r$ — вероятность того, что оба входа равны единице. Тогда вероятность того, что меньший выход $e$ равен единице, равна $r$, а того, что больший выход равен единице, равна $p + q - r$. Поскольку выходы реализуют монотонные функции, то $r \geqslant pq$ согласно лемме 3.7. С другой стороны, $r \leqslant \min\{p, q\}$. Поэтому получаем

$$f_O(\Sigma) - f_O(\Sigma') = f(p) + f(q) - f(r) - f(p + q - r) \leqslant p + q - 2r,$$

где $p+q-2r$ — это в точности вероятность того, что $e$ является 01-компаратором. $\square$

**Следствие 3.9.** *В произвольной схеме $\Sigma$ среднее число 01-компараторов не меньше, чем $f_I(\Sigma) - f_O(\Sigma) = nf(1/2) - f_O(\Sigma)$.*

**Лемма 3.10.** *Если $f$ — подходящая потенциальная функция, то $\bar{C}(n) \geqslant (f(1/2) - o(1))n$.*

Для целей получения еще и хорошей оценки глубины докажем эту лемму в более сильной формулировке, применительно к задаче приближенной сортировки. Пусть $\mathbf{\Sigma_{n,r}}$ состоит из схем, получаемых из схем сортировки удалением последних $r$ слоев компараторов, а $\bar{C}_r(n)$ означает минимальное среднее число 01-компараторов в таких схемах. Сперва докажем вспомогательную лемму.

**Лемма 3.11.** *Если на выходе схемы компараторов могут появляться элементы рангов $r$ и $s$, то (при подходящих входных наборах) могут появляться элементы любого промежуточного ранга.*

▷ Считая, что на вход схемы подается набор чисел от 1 до $n$, с каждым входным набором $v$ свяжем перестановку $\pi([n]) = (\pi(1), \pi(2), \ldots, \pi(n))$, где $\pi(k)$ — номер линии, на которую подается число $k$.

Пусть $\pi_1$ и $\pi_2$ — перестановки, соответствующие входным наборам, при которых на интересующем нас выходе появляются числа $r$ и $s$. Хорошо известно, что одну перестановку можно преобразовать в другую при помощи транспозиций вида $\pi(i) \leftrightarrow \pi(i+1)$.

В результате применения транспозиции к входному набору значения на выходах компараторов изменяются не более чем на 1: на самом деле, все возможные изменения сводятся к превращениям чисел $i$ и $i+1$ одно в другое. Следовательно, на рассматриваемом выходе схемы в процессе преобразования $\pi_1$ в $\pi_2$ посредством транспозиций встретятся все числа в интервале между $r$ и $s$. □

**Лемма 3.12.** *Если $f$ — подходящая потенциальная функция, и $r \leqslant \log_2 n - \omega(n)$, то $\bar{C}_r(n) \geqslant (f(1/2) - o(1))n$.*

▷ Как вытекает из следствия 3.9, достаточно показать, что для произвольной схемы $\Sigma \in \mathbf{\Sigma_{n,r}}$ и подходящей потенциальной функции $f$ справедливо $f_O(\Sigma) = o(n)$. Пусть схема сортировки $\Sigma' \in \mathbf{\Sigma_n}$ получается из $\Sigma$ добавлением $r$ слоев компараторов.

Положим $t = \max\{2^r, \sqrt{n \log_2 n}\}$. В множестве выходов схемы $\Sigma'$ выделим три группы: группа $G_1$ выходов, производящих $n/2 - 2t$ максимальных элементов, группа $G_2$ выходов, производящих $n/2 - 2t$ минимальных элементов, и группа $G_3$ выходов, производящих средние $2t$ элементов.

Поскольку выход схемы $\Sigma$ связан ориентированными путями не более чем с $2^r$ выходами схемы $\Sigma'$, то в силу леммы 3.11 никакой выход схемы $\Sigma$ не может быть связан с выходами из двух разных групп $G_1, G_2, G_3$. Через $H_1$ и $H_2$ обозначим множества выходов схемы $\Sigma$, связанных с группами выходов $G_1$ и соответственно $G_2$. По построению, $|H_i| \geqslant n/2 - 2t$.

Напомним стандартную оценку суммы биномиальных коэффициентов (вариант неравенства Чернова):

$$\sum_{k=0}^{n/2-t} C_n^k \leqslant 2^n e^{-2t^2/n}.$$

Из нее вытекает, что доля наборов длины $n$, в которых менее $n/2 - t$ единиц или менее $n/2 - t$ нулей составляет $o(1)$. Поэтому для любого выхода из $H_1$ вероятность того, что он принимает значение 1, равна $1 - o(1)$, а для выхода из $H_2$ аналогичная вероятность равна $o(1)$.

Теперь величину $f_O(\Sigma)$ можно оценить как

$$f_O(\Sigma) \leqslant (n/2 - t)\,(f(o(1)) + f(1 - o(1))) + 4t \max_{x \in [0,\,1]} f(x) = o(n),$$

так как $f(o(1)), f(1-o(1)) \in o(1)$ ввиду непрерывности $f$, а $\max_{x\in[0,1]} f(x) = 1$. □

Теперь теорема 3.3 вытекает из доказанной леммы и утверждения 3.6. ■

Для того чтобы получить нетривиальную нижнюю оценку при помощи теоремы 3.3, остается построить подходящую потенциальную функцию с высоким значением в точке 1/2. Простые примеры $\min\{x, 1-x\}$ и $4(x-x^2)$ не подходят для этой цели. Поэтому мы обращаемся к многочленам более высоких степеней.

**Следствие 3.13.** *$\mathsf{S}^*(n) \geqslant (C - o(1))n\log_2 n$, где $C = 25/22 \approx 1.136$.*

▷ Сначала заметим, что если $f(0) = f(1) = 0$ и $f''(x) \leqslant 0$, то $f$ является подходящей потенциальной функцией, если условие (2) выполняется при $r = pq$. Действительно, разность между правой и левой частью (3.4) как функция переменной $r$ является выпуклой вверх: ее вторая производная равна $f''(r) + f''(p+q-r) \leqslant 0$, а выпуклая вверх функция минимальные значения на отрезке может принимать только на концах отрезка. Поскольку неравенство (3.4) заведомо выполнено при $r = p$, то останется проверить его при $r = pq$.

**1.** Будем искать $f$ в виде $\gamma\cdot\hat{f}(x-1/2)$, где $\hat{f}(y) = \frac{1}{16} + \frac{a}{4} - ay^2 - y^4$. Вид функции обусловлен естественными требованиями: $\hat{f}$ симметрична относительно точки $y = 0$, где имеет локальный максимум, и равна нулю в точках $y = \pm\frac{1}{2}$. Параметр $a \geqslant 0$ мы определим позднее. Константа $\gamma$ подбирается так, чтобы неравенство (3.4) обращалось в равенство при $p = q = 1/2$ и $r = pq = 1/4$ (предполагаем этот случай экстремальным). Таким образом, $\gamma = \frac{64}{16a+1}$.

По построению, условие (1) выполнено. Также легко видеть, что $\hat{f}''(y) = -2a - 12y^2 \leqslant 0$ при всех $y$. Осталось проверить условие (2) при $r = pq$.

**2.** В обозначениях $y = (p+q-1)/2$ и $x = (q-p)/2$ неравенство (3.4) при $r = pq$ переписывается как

$$\hat{f}(y-x) + \hat{f}(y+x) - \hat{f}((y+1/2)^2 - x^2 - 1/2) - \hat{f}(2y - (y+1/2)^2 + x^2 + 1/2)$$
$$= D_y(x) - D_y(1/4 - y^2 + x^2) \leqslant \frac{16a+1}{32}(1/4 - y^2 + x^2), \quad (3.6)$$

где $D_y(x) = \hat{f}(y-x) + \hat{f}(y+x) = \frac{1}{8} + \frac{a}{2} - 2ay^2 - 2ax^2 - 2y^4 - 2x^4 - 12x^2y^2$.

**3.** Сначала добьемся того, чтобы разность между левой и правой частью в (3.6) была максимальна при $x = 0$ (т.е. когда $p = q$). Для этого достаточно показать, что

$$D_y(0) - D_y(1/4 - y^2) - D_y(x) + D_y(1/4 - y^2 + x^2) \geqslant \frac{16a+1}{32}x^2. \quad (3.7)$$

Получаем

$$D_y(0) - D_y(x) = (2a + 12y^2)x^2 + 2x^4.$$

Обозначим $Y = 1/4 - y^2$. В силу $Y \leqslant 1/4$ и выпуклости вниз функций $x^2$ и $x^4$ имеем

$$\begin{aligned} D_y(Y) - D_y(Y + x^2) &= (2a + 12y^2)((Y + x^2)^2 - Y^2) + 2((Y + x^2)^4 - Y^4) \\ &\leqslant (2a + 12y^2)(x^4 + x^2/2) + 2(x^8 + x^6 + 3x^4/8 + x^2/16) \\ &= (a + 1/8 + 6y^2)x^2 + (2a + 3/4 + 12y^2)x^4 + 2x^6 + 2x^8. \end{aligned}$$

Теперь левая часть в (3.7) оценивается снизу как

$$(a - 1/8 + 6y^2)x^2 - (2a - 5/4 + 12y^2)x^4 - 2x^6 - 2x^8.$$

Ввиду $x \leqslant 1/2$ справедливо $y^2x^2 \geqslant 4y^2x^4$, поэтому указанное выражение минимально при $y = 0$. Теперь справедливость (3.7) обеспечивается неравенством

$$\left(\frac{a}{2} - \frac{5}{32}\right)X - \left(2a - \frac{5}{4}\right)X^2 - 2X^3 - 2X^4 = X\left(\frac{1}{4} - X\right)\left(2X^2 + \frac{5}{2}X + 2a - \frac{5}{8}\right) \geqslant 0$$

при $X = x^2 \in [0,\, 1/4]$ и дополнительном ограничении $a \geqslant 5/16$.

**4.** Осталось доказать (3.6) при $x = 0$. Справедливо

$$\begin{aligned} &D_y(0) - D_y(Y) - \frac{16a + 1}{32}Y = (2a + 12y^2)Y^2 + 2Y^4 - \left(\frac{a}{2} + \frac{1}{32}\right)Y \\ &= (2a + 3 - 12Y)Y^2 + 2Y^4 - \left(\frac{a}{2} + \frac{1}{32}\right)Y = Y\left(Y - \frac{1}{4}\right)\left(2Y^2 - \frac{23}{2}Y + 2a + \frac{1}{8}\right) \leqslant 0 \end{aligned}$$

при $Y \in [0,\, 1/4]$, если только последний квадратичный множитель $P(Y)$ неотрицателен. Так как минимальное значение он принимает на правом конце отрезка, при $Y = 1/4$, то достаточно потребовать $P(1/4) \geqslant 0$. Получаем $a \geqslant 21/16$. Выполнение условия (2) обеспечено.

Окончательно для максимизации значения $f(1/2) = \gamma\hat{f}(0) = 1 + \frac{3}{16a+1}$ выбираем $a = 21/16$. □

Лемма 3.12 и следствие 3.13 влекут оценку глубины схем сортировки (заметим, что не более $n/2$ компараторов можно разместить на одном слое).

**Теорема 3.5.** $\mathsf{D}(n) \geqslant (36/11 - o(1)) \log_2 n$.

## 3.4 Нижние оценки сложности и глубины схемы выбора

Хорошие оценки сложности и глубины выбора дает следующий метод. Пусть слои компараторов делят горизонтальные линии схемы на сегменты. Свяжем с сегментами линий весá: пусть $w_{l,j}$ относится к $l$-му сегменту линии $j$, см. рис. 3.3. Через $[i : j]$ далее обозначаем сравнение элементов на $i$-й и $j$-й линиях.

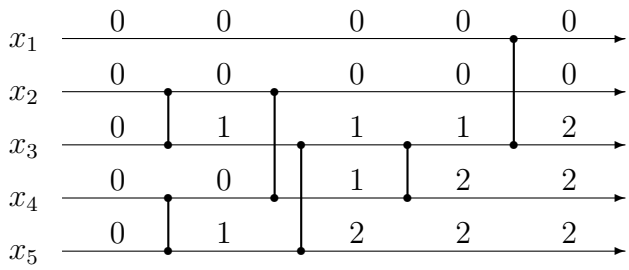



Рис. 3.3: Схема компараторов с приписанными сегментам весами

1) Положим $w_{0,j} = 0$ для всех $j$.

2) Если в $l$-м слое нет компараторов, присоединенных к линии $j$, то положим $w_{l+1,j} = w_{l,j}$.

3) Если в $l$-м слое имеется компаратор $[i : j]$, где $i < j$, то положим $w_{l+1,i} = \min\{w_{l,i},\, w_{l,j}\}$ и $w_{l+1,j} = \max\{w_{l,i},\, w_{l,j}\} + 1$.

**Лемма 3.14.** *Для того, чтобы $j$-й выход схемы компараторов гарантированно возвращал элемент ранга $\geqslant r$, необходимо, чтобы вес выходного сегмента линии $j$ был не меньше* $\log_2 r$.

▷ Обозначим через $v_{l,j}$ нижнюю границу ранга элемента на $l$-м сегменте линии $j$. Очевидно, $v_{0,j} = 1$. Пусть компаратор $[i : j]$, где $i < j$, находится в $l$-м слое схемы. Тогда

$$v_{l+1,i} = \min\{v_{l,i},\, v_{l,j}\}, \qquad v_{l+1,j} \leqslant v_{l,i} + v_{l,j}.$$

Первое соотношение очевидно, второе вытекает из оценки мощности объединения множеств элементов, подчиненных двум сравниваемым.

Остается заметить, что в силу определения веса, $2^{w_{l,j}} \geqslant v_{l,j}$. □

Далее сложность и глубину выбора элемента ранга $t$ обозначаем через $\mathsf{C}^*_t(n)$ и $\mathsf{D}_t(n)$. Поскольку сумма весов выходных сегментов равна сложности схемы, из леммы 3.14 немедленно вытекает

**Теорема 3.6** (В. Е. Алексеев)**.** $\mathsf{C}^*_t(n) \geqslant (n - t)\log_2 t$.

Максимальная оценка, асимптотически $n \log_2 n$, получается для выбора элемента ранга $t \asymp n/\log n$.

Для хорошей оценки глубины требуется чуть более аккуратное рассуждение. Покажем, что веса сегментов не могут в совокупности расти слишком быстро от слоя к слою.

Рассмотрим произвольную схему компараторов на $n$ входах. Положим $x_{l,k} = |\{j \mid w_{l,j} = k\}|$ — число линий с весами $l$-х сегментов, равными $k$. Также введем вспомогательную величину

$$y_{l,k} = \sum_{i=0}^{k} (k + 1 - i)x_{l,i}.$$

Она имеет смысл суммарной «недостачи» весов $l$-го сегмента до порогового значения $k+1$. По построению, $y_{0,k} \geqslant y_{1,k} \geqslant y_{2,k} \geqslant \ldots$ Стратегия доказательства требует оценить $y_{l,k}$ снизу.

**Утверждение 3.15.**

$$y_{l,k} \geqslant \frac{n}{2^l} \sum_{j=0}^{k} (k+1-j) C_l^j.$$

▷ Сначала заметим, что

$$y_{l,k} - y_{l+1,k} \leqslant (x_{l,0} + x_{l,1} + \ldots + x_{l,k})/2. \tag{3.8}$$

Это так, поскольку к $s$ линиям на одном слое присоединяется не более $s/2$ компараторов.

Далее заметим, что по определению $y_{l,k}$

$$y_{l,0} = x_{l,0}, \qquad y_{l,k} - y_{l,k-1} = x_{l,0} + x_{l,1} + \ldots + x_{l,k}. \tag{3.9}$$

Отсюда находим, что $y_{l+1,0} \geqslant y_{l,0}$ и ввиду (3.8), (3.9),

$$y_{l+1,k} \geqslant y_{l,k} - (y_{l,k} - y_{l,k-1})/2 = (y_{l,k} + y_{l,k-1})/2.$$

Таким образом, для нормированных величин $y'_{l,k} = 2^l y_{l,k}/n$ выполнено

$$y'_{0,k} = k+1, \qquad y'_{l+1,k} \geqslant y'_{l,k} + y'_{l,k-1}.$$

Здесь первое соотношение обеспечивает базу индукции $l = 0$, а второе — индуктивный переход от $l$ к $l+1$:

$$y'_{l+1,k} \geqslant \sum_{j=0}^{k} (k+1-j) C_l^j + \sum_{j=1}^{k} (k+1-j) C_l^{j-1} = \sum_{j=0}^{k} (k+1-j) C_{l+1}^j.$$

□

**Следствие 3.16.** *Для схемы выбора элемента ранга $t$ в $n$-элементном множестве глубины $d$ выполнено*

$$t(\lfloor \log_2 t \rfloor + 1) \geqslant n 2^{-d} \left( C_d^0 + C_d^1 + \ldots + C_d^{\lfloor \log_2 t \rfloor} \right). \tag{3.10}$$

▷ Оценим $y_{d,\lfloor \log_2 t \rfloor + 1}$ при помощи леммы 3.14 с одной стороны, а при помощи утверждения 3.15 — с другой. □

При фиксированных $n, t$ правая часть неравенства (3.10) является убывающей функцией относительно $d$. Поэтому максимальное $d$, при котором неравенство не выполняется, служит нижней оценкой глубины. Ограничимся экстремальным случаем выбора среднего элемента.

Пусть $\mathsf{D}(n,t)$ означает глубину выбора элемента ранга $t$ в $n$-элементном множестве.

**Теорема 3.7** (Э. Яо)**.** $\mathsf{D}(n, n/2) \geqslant (a - o(1)) \log_2 n$, *где* $a = 1/(2 - \log_2 3) \approx 2.41$.

► Для оценки глубины схемы выбора полагаем $m = t = n^\alpha$. Положим $d = \beta \log_2 n$. Логарифмируя (3.10) по основанию $n$, получаем условие, при котором $d$ является нижней оценкой глубины $(n, m, t)$-сепаратора для достаточно больших $n$:[4]

$$\alpha < 1 - \beta + \beta H(\alpha/\beta).$$

При подстановке $\beta = 3\alpha$ указанное условие превращается в $\alpha < 1/(4 - 3H(1/3)) = a/3$. Поэтому при некотором $\alpha = a/3 - o(1)$ нарушается неравенство (3.10).

Тем самым требуемая оценка доказана для глубины выбора элемента ранга $t = n^\alpha$. Осталось заметить, что $\mathsf{D}(n, t) \leqslant \mathsf{D}(2n - 2t, n - t)$. Действительно, если в схеме выбора медианы $2n - 2t$ элементов зафиксировать линии наименьших $n - 2t$ элементов и удалить связанные с ними компараторы, то остальная часть схемы будет выполнять выбор элемента ранга $t$. ■

## 3.5 Быстрые схемы сортировки (AKS-схемы)

В этом разделе описывается конструкция схем компараторов, известная теперь как AKS-схемы[5]. Метод эксплуатирует сразу несколько идей, центральной из которых следует признать идею приближенных вычислений. Схемы сортировки оказалось выгодным строить из подсхем, выполняющих сортировку приближенно, с контролируемым числом ошибок.

Введем концепцию приближенной сортировки. Пусть $0 < \varepsilon \leqslant 1$ и $0 < \lambda \leqslant 1/2$. Схема компараторов на $n$ входах называется $(\lambda, \varepsilon)$-*сепаратором*, если при любом $m \leqslant \lambda n$ схема размещает не менее $(1 - \varepsilon)m$ из $m$ наибольших элементов среди $\lambda n$ нижних выходов и не менее $(1 - \varepsilon)m$ из $m$ наименьших элементов — среди $\lambda n$ верхних выходов. Следующие две леммы устанавливают существование сепараторов постоянной глубины (и, следовательно, линейной сложности).

**Лемма 3.17.** *При любом* $\varepsilon > 0$ *и любом* $n$ *существует* $(1/2, \varepsilon)$-*сепаратор на* $2n$ *входах глубины* $O(1/\varepsilon^3)$ *при* $\varepsilon \to 0$.

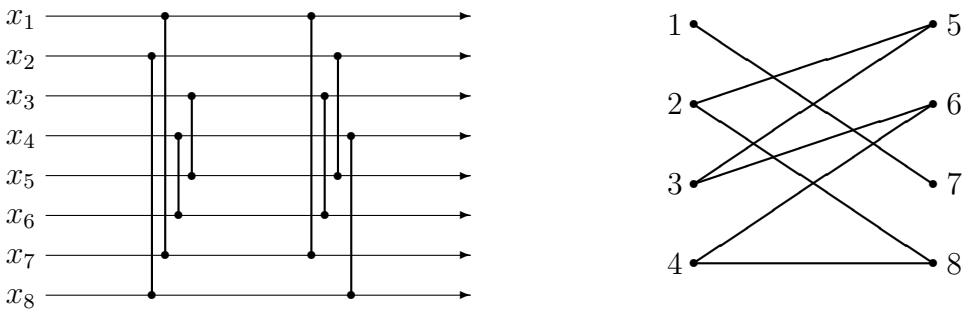
▷ Если $n$ мало, скажем, $n < 64/\varepsilon^3$, то воспользуемся любой схемой сортировки. Поэтому далее полагаем $n \geqslant 64/\varepsilon^3$.

Покажем, что требуемую схему можно построить из нескольких слоев компараторов, где на каждом слое сравниваются между собой элементы из верхней и нижней половин согласно случайно выбранному паросочетанию. Пример схемы показан на рис. 3.4а.

[4] Напомним, что $H(x) = -x \log_2 x - (1 - x) \log_2 (1 - x)$ — функция двоичной энтропии. Далее мы используем известный факт: $C_n^{\alpha n} = 2^{n(H(\alpha) - o(1))}$ при $\alpha > 0$.

[5] По фамилиям авторов метода: М. Айтаи, Я. Комлош, Э. Семереди.

Сопоставим такой схеме двудольный $(n, n)$-граф: вершины одной доли соответствуют позициям элементов из верхней половины списка, а другой — из нижней. Две вершины графа соединяются ребром, если на каком-то слое схемы выполнялось сравнение соответствующих элементов, см. рис. 3.4б. Другими словами, граф является композицией паросочетаний, задающих слои схемы.




а) б)

Рис. 3.4: Схема компараторов и ее граф

*I*. Сначала проверим, что схема является $(1/2, \varepsilon)$-сепаратором, если граф является $(\varepsilon, \alpha)$-*расширителем*, $\alpha = 1/\varepsilon - 1/2$, что значит: для каждого $m \leqslant \varepsilon n$ любое множество из $m$ вершин одной доли соединено ребрами не менее чем с $\alpha m$ вершинами другой доли.

Прежде заметим, что если две вершины соединены ребром в графе, то элементы на соответствующих выходах схемы связаны отношением $\leqslant$ (одна доля собирает только минимумы упорядоченных пар, а другая — только максимумы).

Теперь предположим, что схема не является сепаратором, т. е. при некотором входном наборе, скажем, среди $k \leqslant n$ наибольших элементов $p > \varepsilon k$ оказываются в младшей половине. Рассмотрим соответствующее этим элементам множество вершин в графе. В этом множестве $m = \min\{p, \lfloor \varepsilon n \rfloor\}$ вершин соединены ребрами по меньшей мере с $\alpha m$ вершинами другой доли. Это значит, что минимальный из $p$ элементов уступает не менее чем $p - 1 + \alpha m$ другим элементам. Но при $m = p$ выполнено

$$p - 1 + \alpha m = p - 1 + m/\varepsilon - m/2 > p/\varepsilon - 1 > k - 1,$$

а при $m = \lfloor \varepsilon n \rfloor < p$:

$$p-1+m/\varepsilon-m/2 > (p-1)/2+m/\varepsilon > (\varepsilon n-1)(1/\varepsilon+1/2) > (1+\varepsilon/2)n-1/\varepsilon-1 \geqslant n-1.$$

Это противоречит тому, что выбранный элемент находится среди $k$ наибольших.

*II*. Осталось доказать, что двудольный граф, составленный из подходящего числа $r = r(\varepsilon)$ случайных паросочетаний[6], с ненулевой вероятностью является

[6]Распределение равномерное: все сочетания равновероятны.

$(\varepsilon, \alpha)$-расширителем.

Заметим, что граф является $(\varepsilon, \alpha)$-расширителем, если он не содержит пустых (т.е. безреберных) $(k, n-\alpha k)$-подграфов при любом $k \leqslant \varepsilon n$. Вероятность того, что случайное паросочетание не пересекает (по ребрам) данный $(k, n-\alpha k)$-подграф, равна $C_{\alpha k}^k / C_n^k$. Тогда вероятность $P_k$ того, что $r$ случайных паросочетаний не пересекают хотя бы один из $(k, n-\alpha k)$-подграфов, при помощи простых соотношений $\frac{1}{4\sqrt{k}}(\frac{en}{k})^k \leqslant C_n^k \leqslant (\frac{en}{k})^k$ оценивается как

$$P_k \leqslant 2C_n^k C_n^{\alpha k} \left(\frac{C_{\alpha k}^k}{C_n^k}\right)^r \leqslant 2(4\sqrt{k})^r \left(\frac{e^{1+\alpha} n^{1+\alpha} (\alpha k)^r}{k^{1+\alpha} \alpha^\alpha n^r}\right)^k = 2(4\sqrt{k})^r \left(\alpha e^{1+\alpha} \left(\frac{\alpha k}{n}\right)^{r-\alpha-1}\right)^k \leqslant \left(c_2 \left(\frac{c_1 \alpha k}{n}\right)^{r-\alpha-1}\right)^k,$$

где $c_1 = (4\sqrt{k})^{1/k} \leqslant 4$ и $c_2 = 2^{1/k} \alpha (c_1 e)^{1+\alpha} \leqslant (8e)^{1+\alpha}$.

При $k \leqslant \varepsilon n/4$ выполнено $c_1 \alpha k \leqslant (1-\varepsilon/2)n$. Иначе $k \geqslant \varepsilon n/4 \geqslant 16/\varepsilon^2$, поэтому $c_1 = e^{\ln(16k)/2k} < 1 + \ln(16k)/k \leqslant 1 + \frac{\varepsilon^2}{8} \ln \frac{16}{\varepsilon} \leqslant 1 + \varepsilon/2$. Следовательно, $c_1 \alpha k < (1-\varepsilon^2/4)n$. В любом из двух случаев, если $r$ выбрано несколько большим, чем $\alpha + 4\ln(2c_2)/\varepsilon^2$, получаем $P_k < 2^{-k}$. Тогда вероятность того, что рассматриваемый граф не является $(\varepsilon, \alpha)$-расширителем, не превосходит $\sum_k P_k < 1$. □

Легко проверить, что любой $r$-регулярный (все вершины имеют степень $r$) двудольный граф является объединением $r$ паросочетаний, поэтому схему можно построить из графа.

**Лемма 3.18.** *При любом постоянном $\varepsilon > 0$, любых $\lambda < 1/2$ и $n$ существует $(\lambda, \varepsilon)$-сепаратор на $2n$ входах глубины $O(\log^4(1/\lambda))$ при $\lambda \to 0$.*

▷ Положим $s = \lfloor 2\lambda n \rfloor$ и $k = \lceil \log_2(1/\lambda) \rceil$. Схема строится из $k+1$ слоя $(1/2, \varepsilon_0)$-сепараторов (параметр $\varepsilon_0$ будет выбран позднее): на первом слое — сепаратор на $2n$ входах, на следующих двух — два сепаратора слева и справа для $2^{k-1}s$ крайних элементов[7], на следующем слое — сепараторы для $2^{k-2}s$ элементов с каждого края и т.д. вплоть до последнего слоя из двух сепараторов на $2s$ крайних элементах, см. рис. 3.5 (на рис. схема ориентирована сверху вниз).

Из $m \leqslant s$ наибольших (наименьших) элементов сепаратор первого слоя определяет в «неправильную» половину не более $\varepsilon_0 m$ штук. В следующей паре сепараторов независимо от порядка их расположения: сепаратор правых (левых) $2^{k-1}s$ элементов оставляет за пределами крайнего интервала еще максимум $\varepsilon_0 m$ элементов, а сепаратор с другой стороны не ухудшает характеристики отсечения наибольших (наименьших) элементов[8]. На каждом из последующих слоев сепа-

[7] Если $2^{k-1}s = n$, то эти два сепаратора можно расположить параллельно на одном слое.

[8] В любой схеме компараторов множество $m$ наибольших (наименьших) элементов перемещается строго вправо (влево).

ратор с соответствующей стороны ошибочно выбрасывает из крайнего интервала еще не более $\varepsilon_0 m$ элементов.

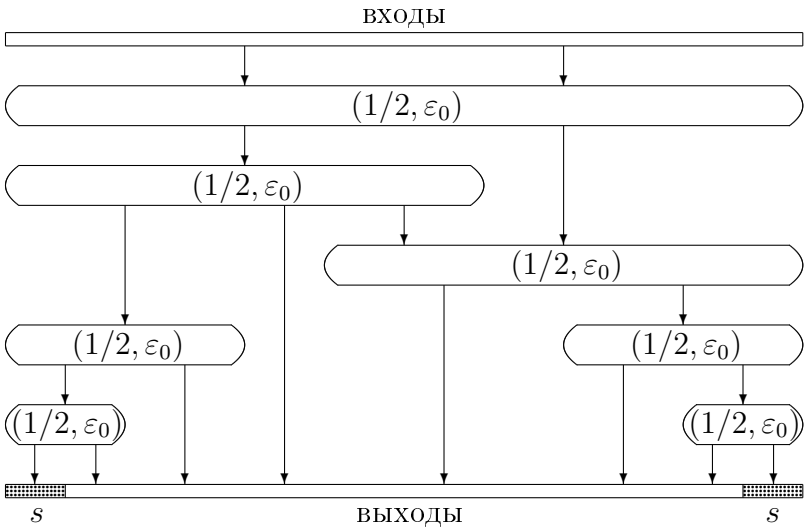



Рис. 3.5: Конструкция $(\lambda, \varepsilon)$-сепаратора

Таким образом, всего не более $k\varepsilon_0 m$ наименьших (наибольших) элементов могут оказаться вне $s$ крайних правых (левых) выходов схемы. Поэтому достаточно выбрать $\varepsilon_0 = \varepsilon/k$. Теперь оценка глубины схемы следует из леммы 3.17. □

**Теорема 3.8** (М. Айтаи, Я. Комлош, Э. Семереди)**.** $\mathsf{D}(n) = O(\log n)$.

► Для простоты рассуждений будем полагать $n = 2^k$. Пусть $\mu, \varepsilon > 0$ — параметры, которые будут выбраны позже. Схема сортировки строится из слоев параллельно расположенных $(\lambda, \varepsilon)$-сепараторов, где $\lambda$ определяется индивидуально для каждого слоя, при этом $\lambda \geqslant \mu$. Удобно представить схему функционирующей во времени: преобразования одного слоя выполняются за одну единицу времени. Действие слоя сепараторов мы рассматриваем как перегруппировку элементов в структуре, ассоциированной с полным двоичным деревом глубины $k$. При каждой вершине дерева имеется контейнер для хранения элементов.

В начальный момент времени $t = 0$ все $n$ элементов находятся в контейнере при корневой вершине. Далее, в любой момент времени элементы каждого непустого контейнера подвергаются приближенной сортировке при помощи сепаратора: элементы из крайних интервалов отправляются на уровень выше к корню, оставшиеся распределяются поровну между контейнерами дочерних вершин, в направлении листьев.

Определим *емкость* контейнера на глубине $d$ в момент времени $t$ как $B_{d,t} = na^d \nu^t$, где постоянные параметры $a > 1$ и $\nu < 1$ будут определены позже. Если

в контейнере находится $b$ элементов, то в случае $\mu B_{d,t} < b/2$ содержимое контейнера, за исключением, быть может, нечетного элемента, упорядочивается при помощи композиции $(1/2, \varepsilon)$- и $(\lambda, \varepsilon)$-сепараторов, $\lambda = \mu B_{d,t}/b$, после чего по $\lfloor \mu B_{d,t} \rfloor$ элементов с каждого края, а также нечетный элемент (при наличии), отправляются на уровень выше. Остальные элементы опускаются на уровень ниже: левая половина — в контейнер левой дочерней вершины, правая — в контейнер правой. Иначе, в случае $\mu B_{d,t} \geqslant b/2$, сепарация не производится — все элементы возвращаются в родительский контейнер. Исключение составляет корневой контейнер — его элементы просеиваются при помощи $(1/2, \varepsilon)$-сепаратора, делятся поровну на две части (число элементов в контейнере обязательно четно), которые отправляются в соответствующие контейнеры дочерних вершин. Процесс продолжается до тех пор, пока все элементы не окажутся на нижних $O(1)$ уровнях дерева; точное условие завершения формулируется как $B_{k,t} < 1/\mu$. После этого применяются схемы сортировки в каждом поддереве глубины $O(1)$. Построенная схема содержит $k \log_{1/\nu}(2a) + O(1) \asymp \log n$ слоев сепараторов и поэтому имеет заявленную глубину. Осталось проверить корректность ее функционирования.

Прежде всего заметим, что в любой момент времени контейнеры одного уровня заполнены одинаково. В частности, поэтому в корневом контейнере всегда четное число элементов. Кроме того, в контейнерах листьев не может оказаться более одного элемента. Следовательно, описанная процедура не выводит элементы за пределы дерева.

Также из построения ясно, что в любой момент все элементы сосредоточены либо на четных, либо на нечетных уровнях дерева.

*I*. Индукцией по $t$ докажем, что при определенных условиях на параметры $a, \varepsilon, \mu, \nu$ число элементов в любом контейнере никогда не превосходит его емкости. Утверждение очевидно при $t = 0$.

Рассмотрим произвольный контейнер на глубине $d$, пустой в момент времени $t - 1$. Число элементов в следующий момент времени в нем в случае $B_{d,t} \geqslant a\nu$ не превосходит

$$2(2\mu B_{d+1,t-1} + 1) + B_{d-1,t-1}/2 = B_{d,t}(4\mu a + 1/(2a))/\nu + 2 < B_{d,t}(4\mu a + 3/a)/\nu,$$

что меньше $B_{d,t}$ при условии

$$\underline{\nu \geqslant 4\mu a + 3/a}. \tag{3.11}$$

В оставшемся случае $B_{d,t} < a\nu$ в момент времени $t - 1$ все контейнеры на вышестоящих уровнях $d' < d$ пусты, т.к. $B_{d',t-1} < 1$. Это значит, что все элементы находятся на уровнях $d+1$ и ниже. При этом $B_{d+1,t-1} < a^2$, и при дополнительном предположении

$$\underline{a^2 = 1/\mu} \tag{3.12}$$

заключаем, что $d+1 \neq k$ (иначе процесс построения схемы был бы уже закончен). Значит, в каждом поддереве с корнем в вершине глубины $d + 1$ в момент $t - 1$

находится четное число элементов, следовательно, четное число элементов находится в корне поддерева (дочерней вершине по отношению к рассматриваемой). Таким образом, в контейнер глубины $d$ в момент времени $t$ отправляется не более $4\mu B_{d+1,t-1} = 4\mu a B_{d,t}/\nu < B_{d,t}$ элементов согласно (3.11).

*II*. Доказательство корректности алгоритма основано на оценке числа посторонних элементов в каждом контейнере. Мы полагаем, что при размещении в листьях дерева элементы должны быть упорядочены в порядке возрастания слева направо. Для любого элемента *родными* вершинами мы считаем лист дерева, в котором элемент должен находиться после упорядочения, а также все вершины на пути от корня дерева к этому листу. В ходе выполнения алгоритма элемент некоторого контейнера будем называть $r$-*чужаком*, если он находится на расстоянии $\geqslant r$ по ребрам дерева от ближайшей родной вершины.

Индукцией по $t$ проверим, что в момент времени $t$ число $r$-чужаков ($r \geqslant 1$) в контейнере глубины $d$ не превосходит $\varepsilon^{r-1}\mu B_{d,t}$ при подходящем выборе параметров. База индукции тривиальна, т.к. в момент $t = 0$ все элементы находятся в корневом, родном для них, контейнере. Докажем индуктивный переход.

*III*. Сначала рассмотрим более простой случай $r \geqslant 2$ (тогда можно полагать $d \geqslant 2$). В разряд $r$-чужаков контейнера глубины $d$ могут попасть $(r+1)$-чужаки из контейнеров дочерних вершин и $(r-1)$-чужаки из родительского контейнера в предыдущий момент времени. С учетом фильтрации их число оценивается сверху как

$$2\varepsilon^r \mu B_{d+1,t-1} + \varepsilon(\varepsilon^{r-2}\mu B_{d-1,t-1}) = \varepsilon^{r-1}\mu B_{d,t}(2\varepsilon a + 1/a)/\nu,$$

т.е. не превосходит $\varepsilon^{r-1}\mu B_{d,t}$ при условии

$$\underline{\nu \geqslant 2\varepsilon a + 1/a}. \tag{3.13}$$

*IV*. Теперь рассмотрим случай $r = 1$. Чужаками в контейнере глубины $d$ при вершине $v$ могут стать 2-чужаки от дочерних вершин, а из родительской вершины — как чужаки, так и элементы, родные для другой ее дочерней вершины $v'$, в предыдущий момент времени. Число чужаков от дочерних вершин легко оценить как $2\varepsilon\mu B_{d+1,t-1} = 2\varepsilon\mu a B_{d,t}/\nu$. Более внимательного рассмотрения требует родительский контейнер.

Пусть родительский контейнер в момент $t-1$ содержит $q$ элементов, из них $q_0$ родных для $v$ элементов, $q_1$ — родных для $v'$, а также $q_2$ чужаков. Если $q_0 \geqslant q/2$, то число чужаков для вершины $v$, отправляемых в ее контейнер, оценивается как $\varepsilon(q_1 + q_2) \leqslant \varepsilon q/2$, т.е. как сумма ошибок $(\lambda, \varepsilon)$-сепаратора, не отправившего чужаков наверх, и $(1/2, \varepsilon)$-сепаратора, отправившего родные для $v'$ элементы в неправильную половину. Иначе, если $q_0 < q/2$, то это число приходится оценивать уже как $\varepsilon q/2 + (q/2 - q_0)$, где второе слагаемое учитывает родные для $v'$ элементы, которые оказываются в половине вершины $v$ даже при правильной сортировке. Оценим величину $q/2 - q_0$.

Рассмотрим специальное гипотетическое распределение элементов по контейнерам в момент времени $t-1$ с тем же числом элементов в каждом контейнере, что и в реальном распределении. Рассортируем и равномерно распределим все элементы между вершинами на уровне $d$ (вершин $v$ и $v'$), а затем произвольно переместим элементы вверх и вниз по дереву для правильного заполнения всех контейнеров, но так, чтобы в родительскую по отношению к $v$ и $v'$ вершину попало соответственно $\lceil q/2 \rceil$ и $\lfloor q/2 \rfloor$ элементов от этих дочерних вершин.

В построенном распределении поддерево с корнем в вершине $v$ содержит только родные для нее элементы, а в контейнере родительской к $v$ вершины находится $\geqslant q/2$ родных для $v$ элементов. Оценим максимальное число родных для $v$ элементов, которые можно переместить из этого контейнера в какие-либо другие. Так получим оценку для $q/2 - q_0$.

Применительно к контейнерам того же уровня $d-1$: для одного контейнера указанные элементы будут 1-чужаками, для двух — 2-чужаками, для четырех — 3-чужаками и т.д. Общее число доступных для размещения позиций на этом уровне оценивается как

$$\mu \left(1 + 2\varepsilon + (2\varepsilon)^2 + \ldots\right) B_{d-1,t-1} < \mu B_{d-1,t-1}/(1-2\varepsilon). \tag{3.14}$$

На произвольном вышестоящем уровне $d-h$ для одного контейнера рассматриваемые элементы будут родными, для другого — 1-чужаками, еще для двух — 2-чужаками, для четырех — 3-чужаками и т.д. Общее число доступных позиций на этих уровнях (при нечетных $h \geqslant 3$) оценивается как

$$\left(1 + \mu(1 + 2\varepsilon + (2\varepsilon)^2 + \ldots)\right) \sum_{i=1}^{d/2} B_{d-2i-1,t-1} < \left(1 + \frac{\mu}{1-2\varepsilon}\right) \sum_{i \geqslant 1} a^{-2i} B_{d-1,t-1} = \left(1 + \frac{\mu}{1-2\varepsilon}\right) \frac{B_{d-1,t-1}}{a^2 - 1}. \tag{3.15}$$

Поскольку в поддереве вершины $v$ свободных для заполнения позиций уже нет, на произвольном нижестоящем уровне $d+h$ доступно $2^h$ контейнеров[9], для которых указанные элементы будут $(h+1)$-чужаками, $2^{h+1}$ контейнеров, для которых эти элементы будут $(h+2)$-чужаками, и т.д. Общее число возможных позиций не превосходит

$$\sum_{i=1}^{(k-d)/2} \mu \left((2\varepsilon)^{2i-1} + (2\varepsilon)^{2i} + \ldots\right) B_{d+2i-1,t-1} < \mu \sum_{i \geqslant 1} \frac{(2\varepsilon)^{2i-1}}{1-2\varepsilon}\, a^{2i-1} B_{d,t-1} = \frac{2\varepsilon a \mu B_{d,t-1}}{(1-2\varepsilon)(1-(2a\varepsilon)^2)}. \tag{3.16}$$

[9] В поддереве с корнем $v'$.

Суммируя (3.14), (3.15), (3.16), получаем

$$q/2 - q_0 < \left(\frac{1}{a^2-1} + \frac{\mu a^2}{(1-2\varepsilon)(a^2-1)} + \frac{2\varepsilon a^2 \mu}{(1-2\varepsilon)(1-(2a\varepsilon)^2)}\right) B_{d-1,t-1}.$$

Как следствие, общее число чужаков в контейнере вершины $v$ в момент $t$ с учетом поступающих из дочерних вершин оценивается как

$$\left(2\varepsilon a\mu + \frac{1}{a^3-a} + \frac{\mu a}{(1-2\varepsilon)(a^2-1)} + \frac{2\varepsilon a\mu}{(1-2\varepsilon)(1-(2a\varepsilon)^2)}\right) B_{d,t}/\nu,$$

что не превосходит $\mu B_{d,t}$ при условии

$$\underline{\nu \geqslant 2\varepsilon a + \frac{1}{\mu(a^3-a)} + \frac{a}{(1-2\varepsilon)(a^2-1)} + \frac{2\varepsilon a}{(1-2\varepsilon)(1-(2a\varepsilon)^2)}}. \qquad (3.17)$$

*V*. Теперь легко видеть, что в момент $t$ окончания процедуры построения схемы ни в одном контейнере нет чужаков (в предположении, что параметры выбраны верно). Действительно, в произвольном контейнере глубины $d$ согласно доказанному выше — не более $\mu B_{d,t} \leqslant \mu B_{k,t} < 1$ чужаков.

При этом в силу (3.12) $B_{d,t} = a^{d-k} B_{k,t} < a^{d-k}/\mu \leqslant 1$ при $d \leqslant k-2$, значит, все элементы находятся в нижних двух слоях дерева.

Осталось указать выбор параметров, удовлетворяющий всем необходимым условиям (3.11), (3.12), (3.13), (3.17). Например, подходит $\varepsilon = \mu = 1/100$, $a = 10$, $\nu = 0.7$. ■

**Следствие 3.19.** $\mathsf{S}^*(n) = O(n \log n)$.

# Упражнения

**Упр. 3.1.** Покажите, что результат теоремы 3.1 можно обобщить до $\mathsf{S}^*(\mathcal{L}_{m,n}) \geqslant 0.5(m+n)\log_2 m - O(m)$ при $m \leqslant n$. [*П. Милтерсен, М. Патерсон, Ю. Таруи*]

**Упр. 3.2.** Оцените предел возможностей метода теоремы 3.3. Покажите, что если подходящая потенциальная функция $f(x)$ монотонно возрастает на отрезке $[0, 1/2]$ и убывает на отрезке $[1/2, 1]$, то, скажем, $f(1/2) \leqslant 1.75$.

**Упр. 3.3.** Используя следствие 3.16, покажите, что глубина выбора элемента ранга 2 равна $\log_2 n + \log_2 \log n \pm O(1)$. [*Э. Яо*]

**Комментарии.** Теорема 3.1 доказана в [7] с чуть более точной оценкой $n \log_2 n - 0.66n$, а также в более общей форме $\mathsf{S}^*(\mathcal{L}_{m,n}) \geqslant 0.5(m+n)\log_2 m - 0.73m$ при $m \leqslant n$. Несколько более простое доказательство асимптотически такой же, но уступающей в

точности остаточного члена оценки приведено в [6]; простая и точная по порядку оценку доказывается в [3]. Верхняя оценка получена в [4], см. также [3].

Теорема 3.3 и следствие 3.13 с константой $C = 1.12$ доказаны в работе [5]. Теорема 3.4 является частным случаем теоремы о четырех функциях Р. Альсведе и Д. Дайкина. Результаты раздела 3.4 получены В. Е. Алексеевым [1] (см. также [3]) и Э. Яо [9].

Доказательство теоремы 3.8 следует адаптированному Ж. Сейферасом [8] варианту метода сортировки М. Айтаи, Я. Комлоша и Э. Семереди [2]. Мультипликативная константа в оценке глубины теоремы 3.8 безумно велика. В ряде работ предпринимались попытки уменьшить ее. Опубликованные с доказательствами оценки (для разнообразных модификаций алгоритма) имеют величину порядка нескольких тысяч. Некоторые прикидки допускают существование схем с глубиной в районе $100 \log_2 n$. На практике лучшие результаты дают варианты схем Бэтчера [4] теоретической глубины порядка $\log^2 n$.

Выбор многочлена $\hat{f}(y) = \frac{1}{64} + \frac{a}{4} - ay^2 - y^6$ при $a = 0.268$ позволяет улучшить константу в следствии 3.13 до $C > 1.215$, а оценку теоремы 3.5 — до $(3.43 - o(1)) \log_2 n$.

## Литература


[1] Алексеев В. Е. *О некоторых алгоритмах сортировки с минимальной памятью.* Кибернетика. 1969. **5**(5), 99–103.

[2] Ajtai M., Komlós J., Szemerédi E. *Sorting in $c \log n$ parallel steps.* Combinatorica. 1983. **3**(1), 1–19.

[3] Кнут Д. Э. *Искусство программирования. Т. 3. Сортировка и поиск.* М.: Вильямс, 2007.

[4] Batcher K. E. *Sorting networks and their applications.* Proc. AFIPS spring joint comput. conf. (Atlantic City, 1968). **32**. NY: AFIPS, 1968, 307–314.

[5] Kahale N., Leighton T., Ma Y., Plaxton C. G., Suel T., Szemerédi E. *Lower bounds for sorting networks.* Proc. STOC (Las Vegas, 1995). NY: ACM, 1995, 437–446.

[6] Leighton T., Ma Y., Suel T. *On probabilistic networks for selection, merging, and sorting.* Proc. ACM Symp. Parall. Alg. and Architect. (Santa Barbara, 1995). NY: ACM, 1995, 106–118.

[7] Miltersen P. B., Paterson M., Tarui Y. *The asymptotic complexity of merging networks.* J. ACM. 1996. **43**(1), 147–165.

[8] Seiferas J. *Sorting networks of logarithmic depth, further simplified.* Algorithmica. 2009. **53**(3), 374–384. [см. также: Seiferas J. *AKS sorting networks.* Manuscript, 2017. доступно на http://www.researchgate.net/profile/Joel-Seiferas]

[9] Yao A. *Bounds on selection networks.* SIAM J. Comput. 1980. **9**(3), 566–582.