*Посвящается Ф.И.Карпелевичу, замечательному математику, собравшему в 70-х годах на кафедре высшей математики МИИТа славную плеяду молодых выпускников мехмата МГУ в надежде, что хоть кто-то из них решит хоть какую-нибудь железнодорожную проблему.*

# Сортировки, числа Фибоначчи, системы счисления и контекстно-свободные грамматики

**А.КУЛАКОВ**

*Вагоны шли привычной линией,*
*Подрагивали и скрипели,*
*Молчали желтые и синие,*
*В зеленых плакали и пели.*

А.Блок

## Введение и задачи

Кто из вас, дорогой читатель, не сталкивался с железной дорогой? Не стоял в очередях за билетами? Не ездил в грязных вагонах? Казалось бы, какая может быть математика на железных дорогах? Все простые задачи решены еще в прошлом веке. Остались очень сложные, скорее физические задачи, в которых большие сложные системы дифференциальных уравнений, описывающих движения больших составов, нужно численно решать на мощных компьютерах. Но, оказывается, и сейчас еще могут встретиться задачи, которые школьник не только может решить, но и помочь своими решениями железнодорожникам. Интересно то, что похожие задачи встречаются при проектировании компьютеров и при создании операционных систем к ним. Что же может быть общего между

Рис. 1

Рис. 2

Рис. 3

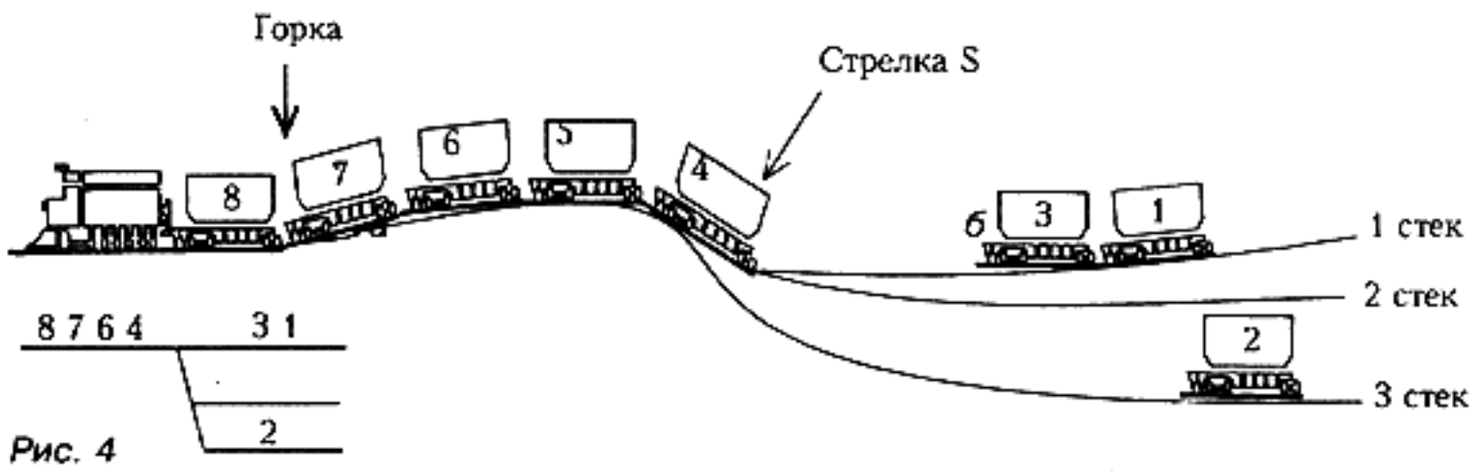


Рис. 4

компьютерами и железными дорогами?

Речь в статье пойдет о сортировках. Кто из вас наблюдал, как на сортировочной горке сортируются железнодорожные составы? *Горкой* груда путей называется потому, что в профиль действительно имеет вид горки.

У железнодорожников даже выработалась своя терминология: *надвигом* называется вталкивание состава на горку (рис.1), скатывание отцепленного вагона с горки называется *роспуском* (рис.2), а сам этот вагон называется *отцепом*, торможение отцепа внизу горки (а каждый вагон обязательно нужно тормозить — ведь при ударе тела испытывают огромные напряжения и разрушаются) называется *осаживанием*. Операция, изображенная на рисунке 3, называется *вытягиванием*.

На больших сортировочных узлах число тупиков очень велико, поэтому любой состав всегда можно отсортировать за один или два роспуска. А что дéлать на маленьких станциях, где есть два-три тупика, да и то один может быть занят частично?

Прежде чем идти дальше, давайте договоримся о терминологии. Будем операцию, изображенную на рисунках 1 и 2, называть, как и железнодорожники, роспуском состава или просто роспуском, а на рисунке 3 — вытягиванием. Роспуск и все следующие за ним вытягивания до следующего роспуска состава или его части назовем *шагом* сортировки. Тупики горки, используя компьютерную терминологию, назовем *стеками*. Стекам можно присваивать номера 0, 1, 2, ... и т.д., а саму горку будем условно изображать, как на рисунке 4.

Будем через $k$ обозначать число стеков, а через $p$ — число вытягиваний. Кроме того, договоримся считать состав отсортированным, если вагоны в нем идут в порядке возрастания нумерации слева направо (рис.5,*л*). Вагоны состава, поступающего на сортировку, уже имеют нумерацию. Ведь после сортировки состав следует по маршруту, встречая на своем пути станции-получатели вагонов в определенном порядке. Этот порядок и определяет номера вагонов. Мы будем считать, что вагоны состава, подаваемого на сортировку, пронумерованы так, чтобы отсортированный состав мог спокойно следовать по своему маршруту и на каждой станции от хвоста состава можно было отцепить направляемые на эту станцию вагоны.

**Задача 1.** Как бы вы предложили нумеровать вагоны, если получателю неважно, в каком порядке он получит свои вагоны?

**Пример 1.** На рисунке 5, *а*—*л*, приведен подробный пример сортировки состава из 5 вагонов, поданных на горку, имеющую 2 стека, в порядке, изображенном на рисунке 5, *а*. Сортировка состоит из двух шагов, первый шаг (рис.5,*б*—*е*) и второй шаг (рис.5,*ж*—*л*) включают в себя по два вытягивания. Итак, мы отсортировали состав из 5 вагонов за 2 шага, совершив 4 вытягивания.

Можно несколько формализовать наши определения. Представим себе, что элементы $\sigma_i$ лежат в стопке ($\sigma_1$ сверху, $\sigma_n$ снизу) на некотором выделенном месте, которое мы

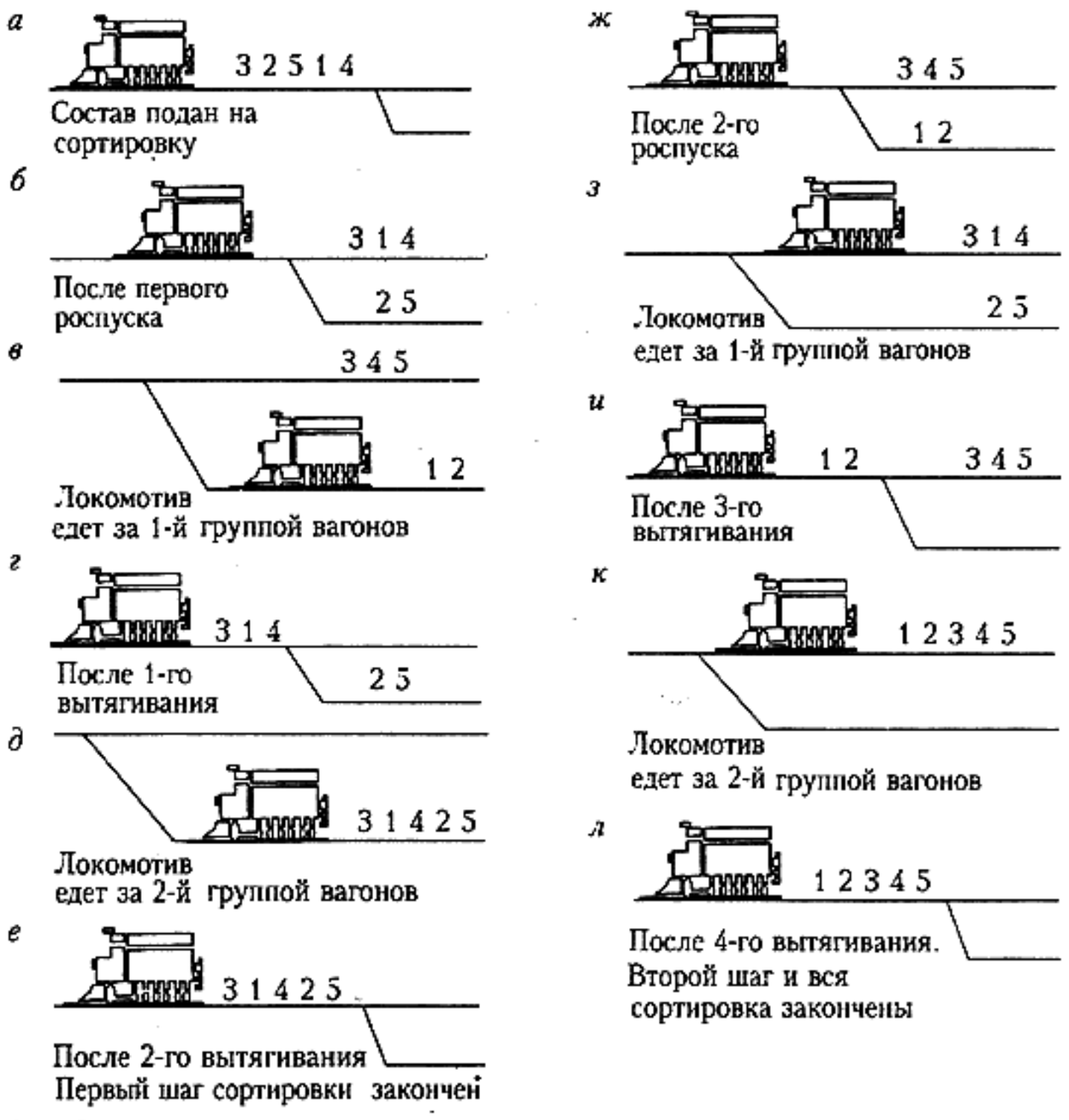


Рис. 5

будем называть *горкой*. Фиксировано число $k$. Разрешается, вынимая элементы снизу стопки, раскладывать их на $k$ стопок, т.е. элементы в стопках расположены в том же порядке, что и в стопке на горке. Эти новые стопки и есть стеки, а описанная операция — роспуск. Дальше все элементы из некоторого стека можно переносить на горку, не меняя порядок. Эта операция — вытягивание. Перемещать элементы на горку можно из нескольких стеков. Будем считать (хотя это не принципиально), что элементы из следующего стека подкладываются снизу под стопку, уже существующую на горке. В конце мы хотим получить упорядоченное множество 1, 2, 3, ..., $n$. Шагом называются все действия между двумя последовательными роспусками.

Прежде чем читать дальше, попробуйте самостоятельно решить несколько задач. Во всех задачах, если не оговорено противное, будем считать, что все стеки «бесконечные», т.е. могут вместить сколько угодно вагонов, хоть весь состав.

**Задачи**

**2.** Докажите, что каждый состав можно отсортировать за $n - 1$ шаг и $2n - 2$ вытягивания.

**3.** Отсортируйте состав, идущий в порядке 4, 3, 2, 1, на горке с двумя стеками ($k = 2$)

а) за наименьшее число шагов;

б) за наименьшее число вытягиваний.

**4.** Отсортируйте состав, вагоны которого идут в порядке 8, 7, 6, 5, 4, 3, 2, 1, на горке с двумя стеками ($k = 2$)

а) за наименьшее число шагов;

б) за наименьшее число вытягиваний.

**5.** Отсортируйте состав, вагоны которого идут в порядке 9, 8, 7, 6, 5, 4, 3, 2, 1, на горке с тремя стеками ($k = 3$)

а) за наименьшее число шагов;

б) за наименьшее число вытягиваний.

**6.** Отсортируйте состав, идущий в порядке 5, 7, 1, 3, 2, 6, 4, на горке с двумя стеками ($k = 2$)

а) за наименьшее число шагов;

б) за наименьшее число вытягиваний.

**7.** Отсортируйте состав, вагоны которого идут в порядке 15, 4, 12, 7, 11, 9, 8, 10, 13, 5, 1, 6, 3, 2, 14, на горке с двумя стеками ($k = 2$). Сколько шагов вам понадобится для этого? А сколько вытягиваний?

Как вы уже поняли, мы разделили каждую задачу о сортировке на две:

(1) первая — это задача о сортировке за наименьшее число шагов,

(2) вторая — это задача о сортировке за наименьшее число вытягиваний, ведь именно вытягивание — та реальная операция, которую приходится проделывать машинисту маневрового локомотива.

**Задачи**

**8.** Попробуйте отсортировать состав 1, 11, 12, 5, 13, 10, 2, 4, 9, 7, 6, 14, 3, 8, 15 на горке, имеющей 3 стека. В каком из ваших вариантов число вытягиваний минимально?

**9.** а) Придумайте алгоритм сортировки, который каждый состав из $n$ вагонов на горке с $k$ стеками сортирует за число шагов $s$, где $k^{s-1} < n \le k^s$.

б) Докажите, что ваш алгоритм правильно сортирует любой состав.

в) Докажите, что существует состав из $n$ вагонов, который нельзя отсортировать за меньшее число шагов.

Приступим к решению первой задачи.

## О системах счисления

Прежде чем описать алгоритмы сортировок составов, вспомним системы счисления. Мы с детства привыкаем к тому, что любое число можно записать с помощью девяти цифр:

0, 1, 2, 3, ..., 9.

Значение каждой цифры зависит от того места, на котором эта цифра стоит. Например, число 905, записанное цифрами 9, 0, 5, равно $9 \cdot 10^2 + 0 \cdot 10 + 5$, число $\overline{590} = 5 \cdot 10^2 + 9 \cdot 10 + 0$, а число $\overline{abcd} = a \cdot 10^3 + b \cdot 10^2 + c \cdot 10 + d$ (обычно над набором цифр сверху ставят черту, для того чтобы не спутать десятичную запись числа с произведением его цифр).

Число 10, лежащее в основании нашей системы счисления, можно заменить на любое другое число, например на 2. В этой системе счисления всего две цифры — 0 и 1. Для того чтобы записать число в двоичной системе, его нужно представить в виде суммы степеней двойки. Например, $1 = 2^0$, $2 = 2^1$, $3 = 2 + 1$, $4 = 2^2$, $5 = 4 + 1$, и т.д. Если $n = 2^{n_1} + 2^{n_2} + \ldots$, то его двоичной записью будет набор из нулей и единиц, в котором единицы стоят на $n_i$-х местах, считая справа, а на остальных местах стоят нули. Например, $13 = 8 + 4 + 1$, следовательно, $13 = 1101_2$.

Вместо числа 2 в основу системы счисления можно положить произвольное натуральное число $q$. Такая система называется $q$-ичной (2-ичная, 3-ичная, ..., 10-тичная, и т.д.). В ней для записи чисел используется $q$ цифр: 0, 1, 2, ..., $q - 2$, $q - 1$. $q$-ичной записью числа $m = a_n q^n + \ldots + a_1 q + a_0$, где все $a_i$ — цифры $q$-ичной системы $(0 \le a_i \le q - 1)$, служит запись $\overline{a_n a_{n-1} \ldots a_1 a_0}_q$ (индекс $q$ внизу показывает основание системы счисления; он не пишется, если понятно, о какой системе идет речь). Например, $25_{10} = 221_3$, или $1996_{10} = 5551_7$, или $111111_{10} = B207_{16}$.

Последняя, шестнадцатеричная система счисления широко распространена в программировании (запись адресов в современных компьютерах в 2-ичной системе слишком длинна). В ней используется 16 цифр: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, $A$(=10), $B$(=11), $C$(=12), $D$(=13), $E$(=14), $F$(=15).

Теперь сформулируем алгоритм поразрядной сортировки. Мы сформулируем его для сортировочной горки с двумя стеками, но он естественно обобщается на произвольные $k$.

*Замечание 1.* Число 0 в системах счисления во всех разрядах имеет одни нули. Будем поэтому нумерацию вагонов начинать с 0.

## Алгоритм поразрядной сортировки

(1) Напишем номера наших вагонов в 2-ичной системе счисления, начиная с 0. Нумерация разрядов: от младшего, нулевого разряда единиц к старшему.

(2) Дадим стекам номера 0 и 1.

(3) Сортировка начинается с нулевого шага. $k$-й шаг сортировки состоит в следующем: при роспуске состава вагоны, номера которых в $k$-м

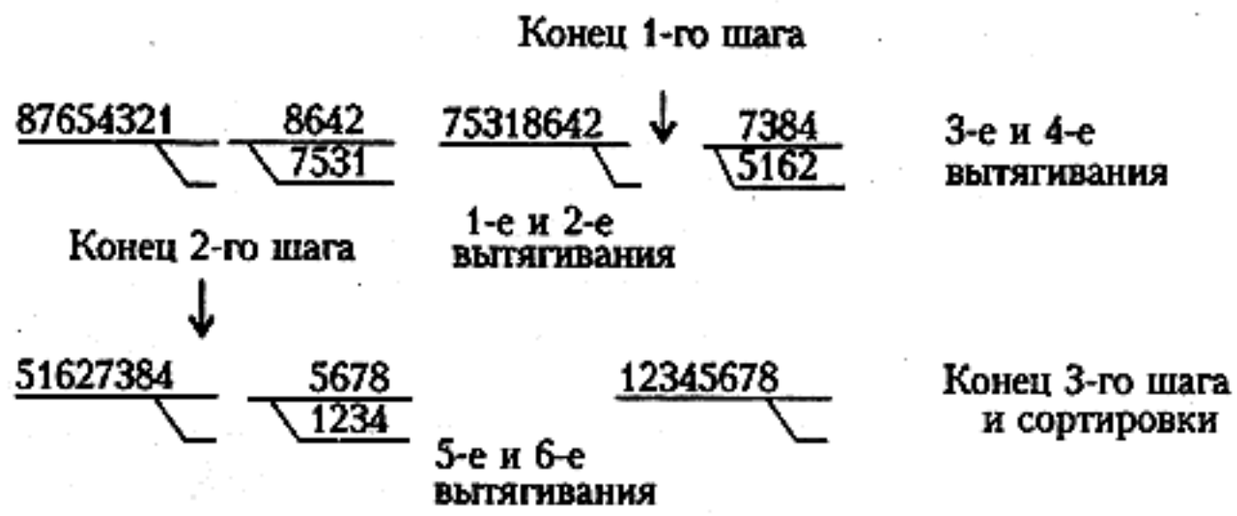


*Рис. 6.*

разряде имеют цифру 0, отправляются в 0-й стек, а номера которых в $k$-м разряде имеют цифру 1, отправляются в 1-й стек. После роспуска вытягиваются сначала вагоны из 0-го стека, а затем из 1-го. На этом $k$-й шаг сортировки заканчивается.

Посмотрим на примере решения задачи 4,а), как работает наш алгоритм (рис.6). Из рисунка видно, что для сортировки потребовалось сделать 3 шага ($8 = 2^3$) и $6 = 2\times3$ вытягиваний. Теперь понятно, что нужно делать, если у сортировочной горки 3 стека — нужно записывать номера вагонов в 3-ичной системе счисления и сортировать по разрядам 3-ичной системы.

Чтобы освоиться теперь с алгоритмом поразрядной сортировки, решите с его помощью задачи 3,а) — 6,а). Совпадают ли эти решения с вашими?

## Основные утверждения

Приведем одно очевидное утверждение, с которого начинаются решения наших задач.

**Лемма 1.** *Если два вагона попадают в один стек, то порядок между ними не меняется.*

**Теорема 1.** *Алгоритм поразрядной сортировки решает задачу 9.*

Вспомним, как мы сравниваем два числа $a = a_s a_{s-1}\ldots a_0$ и $b = b_s b_{s-1}\ldots b_0$. Мы всегда будем считать, что их записи выровнены по разрядам (в крайнем случае можно приписать нули). $a < b$ тогда и только тогда, когда $a_s = b_s$, $a_{s-1} = b_{s-1}$, ..., $a_{l+1} = b_{l+1}$, а $a_l < b_l$.

Посмотрим, что происходит с вагонами $a$ и $b$ при поразрядной сортировке. Первые $(l-1)$ шагов вагоны $a$, $b$ как-то шатаются по стекам. Абсолютно не важно, как. На $l$-м шаге вагон $a$ попадает в стек с номером $a_l$, а вагон $b$ попадает в стек с номером $b_l$. Так как $a_l < b_l$, то вагон $a$ будет вытянут на этом шаге перед вагоном $b$ и окажется левее $b$. Но во всех старших, чем $l$, разрядах у $a$ и $b$ стоят одинаковые цифры, поэтому во все последующие роспуски вагоны $a$ и $b$ будут попадать в одни и те же стеки. Значит, по лемме 1 их порядок не изменится. Следовательно, в конце сортировки вагон $a$ будет находиться левее $b$. А так как это верно для любой пары вагонов, то весь состав будет упорядочен.

Заметим, что точно так же упорядочены слова в словаре. Такой порядок называется *лексикографическим*.

Необходимое для записи числа $n$ число разрядов в $k$-ичной системе счисления ограничено сверху числом $\lceil \log_k n \rceil$, значит, и число шагов в поразрядной сортировке состава из $n$ вагонов ограничено сверху $\lceil \log_k n \rceil$, где $[x]$ — наименьшее целое число, не меньшее чем $x$.

**Определение 1.** Множество 1, 2, ..., $n-1$, $n$ в перепутанном порядке $(\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_n)$ называется *перестановкой*.

Вообще-то, перестановкой в математике называется взаимно-однозначное отображение $\sigma: \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$, но в нашей задаче можно не различать перестановку $\sigma$ и результат ее действия.

Назовем множество элементов $\{\sigma_{k_1}, \sigma_{k_2}, \sigma_{k_3}, \ldots, \sigma_{k_s}\}$ перестановки $\sigma = \{\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_n\}$ *обратной цепью*, если $\sigma_{k_i} > \sigma_{k_{i+1}}$, $k_i < k_{i+1}$. Обратная цепь, состоящая из наибольшего числа элементов, называется *максимальной*. Обозначим длину максимальной обратной цепи $f(\sigma)$.

Из принципа Дирихле следует, что длина максимальной обратной цепи $f(\sigma)$ за один шаг может уменьшиться не более чем в $k$ раз. Действительно, распуская состав в $k$ стеков, мы делим максимальную обратную цепь на $k$ обратных цепей, поэтому длина максимальной обратной цепи в получившейся перестановке не меньше $\frac{f(\sigma)}{k}$.

Это дает оценку снизу, а так как наибольшей максимальной обратной цепью обладает перестановка $(n, n-1, \ldots, 2, 1)$ и ее длина равна $n$, то эту перестановку нельзя отсортировать за меньшее, чем $\lceil \log_k n \rceil$, число шагов.

Теперь возникает вопрос, что делать с каждой конкретной перестановкой. Например, состав уже отсортирован, т.е. идет в порядке 1, 2, 3, ..., $n-1$, $n$. Было бы глупо его сортировать.

**Упражнение 1.** Посмотрите, что произойдет с этим составом, если к нему применить поразрядную сортировку, например, при $k = 2$?

Опять выручает лемма 1. Из нее следует уже менее очевидное утверждение.

**Лемма 2.** *Пусть в составе $(i+1)$-й вагон расположен правее $i$-го. Тогда для любого алгоритма $(A)$, сортирующего состав, существует алгоритм $(B)$, который не «хуже первого» (это означает, что новый алгоритм делает не больше шагов и не больше вытягиваний) и который при сортировке все время отправляет $i$-й и $(i+1)$-й вагоны в один стек.*

Действительно, изменим алгоритм $(A)$: пусть в алгоритме $(B)$ все вагоны попадают в те же стеки, что и в алгоритме $(A)$, за исключением $(i+1)$-го — $(i+1)$-й вагон всегда распускается в тот же самый стек, что и $i$-й вагон в алгоритме $(A)$. Ясно, что число вагонов между $i$-м и $(i+1)$-м вагонами не увеличивается. Пусть в самом начале между $i$-м и $(i+1)$-м вагонами стоял вагон с номером $j$. Если бы в алгоритме $(A)$ $j$-й вагон все время попадал в тот же стек, что и $i$-й, то в конце мы получили бы неотсортированный состав. А так как состав в конце работы алгоритма $(A)$ отсортирован, это означает, что в какой-то момент вагоны $i$ и $j$ попадают в разные стеки, и число вагонов между $i$-м и $(i+1)$-м уменьшается. А так как это верно для любого вагона, находящегося в начале между $i$-м и $(i+1)$-м вагонами, то в конце работы алгоритма $(B)$ они будут стоять рядом и в нужном порядке. Это завершает доказательство леммы 2.

Следовательно, если в составе $(i+1)$-й вагон расположен правее $i$-го, то оба эти вагона каждый раз можно отправлять в один стек. А это, в свою очередь, означает, что эти вагоны не нужно различать и, следовательно, им нужно дать одинаковые номера.

**Определение 2.** Назовем каждую максимальную последовательность $i$, $i+1$, $i+2$, ..., $i+m$, в которой каждый следующий элемент стоит в перестановке правее предыдущего, *блоком*. Ясно, что элементам в блоке нужно давать один и тот же номер. Назовем эту операцию *перенумерацией*.

**Пример 2.** В составе из задачи 6 (5, 7, 1, 3, 2, 6, 4) 4 блока (1, 2; 3; 4; 5, 6 и 7), после перенумерации он будет выглядеть так: 2, 3, 0, 1, 0, 2, 1. А как выглядят после перенумерации составы из задач 7 и 8?

**Лемма 3.** *Пусть алгоритм $(A)$ сортирует состав из $n$ вагонов, идущих в обратном порядке. Тогда этот алгоритм так же (т.е. за такое же*

*число вытягиваний и шагов) сортирует состав, в котором после перенумерации n блоков. И наоборот.* (Под «так же» понимается, что на каждом шаге вагоны с тем же самым номером отправляются в тот же самый стек и вытягиваются вагоны из тех же стеков в том же порядке.)

Очевидно, что нужно наблюдать за крайним правым вагоном из $i$-го блока и крайним левым вагоном из $(i+1)$-го. Рассуждения такие же, как и в лемме 2.

Из леммы 3 мы получаем уже совершенно нетривиальное

**Следствие 1.** *Общую задачу решает алгоритм, оптимально сортирующий состав, вагоны в котором расположены в обратном порядке.*

Итак, доказана

**Теорема 2.** *Любую перестановку $\sigma$ можно отсортировать на горке с $k$ стеками за не более чем $s$ шагов, где $k^{s-1} < n \le k^s$, а $n$ — число блоков, и эта оценка неулучшаема.*

Итак, задача об алгоритме сортировки за наименьшее число шагов решена, причем не только в целом, но и для каждой конкретной перестановки.

*Замечание 2.* Кто из вас, дорогой читатель, не играл хоть раз в жизни в карты? Наверное, все когда-нибудь держали в руках колоду карт и тасовали ее. Существует несколько способов тасования колоды карт. Один из них называется врезка. Колоду карт делят на две, не обязательно равные, части. После этого одну из частей, не меняя в ней порядок, вставляют в другую так, что карты из разных частей перемежаются.

А задавал ли кто-нибудь из вас себе такой вопрос: за какое наименьшее число врезок колоду можно перевести из одного положения в другое?

Я думаю, вы уже догадались, что ответ и на этот вопрос содержится в этой статье. Ведь операция врезки — это операция обратная роспуску и последующему вытягиванию из обоих стеков на горке с двумя стеками.

**Задача 10.** Сформулируйте ответ на поставленный вопрос.

А как же наименьшее число вытягиваний? Сортируя состав из $n$ блоков по алгоритму поразрядной сортировки на горке с $k$ стеками, мы делаем $\lceil \log_k n \rceil$ шагов и $k \times \lceil \log_k n \rceil$ вытягиваний. Это число равно $k \times \left\lceil \frac{1}{\ln k} \ln n \right\rceil$ (вот зачем нужно уметь пользоваться свойствами логарифмов). Конечно, число вытягиваний — целое число, но мы рассмотрим функцию $\frac{k}{\ln k} \ln n$. Последний сомножитель $\ln n$ не зависит от $k$, поэтому посмотрим на коэффициент перед $\ln n$: $g(k) = \frac{k}{\ln k}$. Более того, рассмотрим эту функцию не только при целых $k$, но и при любых действительных $x$.

**Задача 11.** Докажите, что функция $g(x) = \frac{x}{\ln x}$ имеет минимум при $x = e$.

Ближайшие к $e$ целые числа — это 2 и 3. При этом $\frac{2}{\ln 2} \ge \frac{3}{\ln 3}$ (это упражнение). Поэтому для алгоритма поразрядной сортировки самое выгодное $k$ в смысле вытягиваний — это $k = 3$.

Но так уж устроен человек — ему мало знать, что при больших $n$ он может отсортировать состав примерно за $\frac{3}{\ln 3} \ln n$ вытягиваний. Хочется знать: а каково наименьшее число вытягиваний $p$ для данных конкретных $n$ и $k$?

Вспомним, что леммы 1, 2 и 3 были применимы не только к шагам сортировки, но и к вытягиваниям. Также для вытягиваний верно и следствие 1. Поэтому будем в дальнейшем предполагать, что в начале любой сортировки вагоны нашего состава расположены в обратном порядке.

Попробуем обратить наш вопрос: а каково наибольшее число вагонов, идущих в обратном порядке, которые можно отсортировать при данных $p$ и $k$?

Итак, введем еще одно обозначение.

**Определение 3.** Обозначим через $W(p, k)$ максимальное число вагонов $n$, идущих в последовательности $n$, $n-1$, ..., 2, 1, которые можно отсортировать за $p$ вытягиваний, имея $k$ стеков.

Вот некоторые очевидные равенства для $W(p, k)$:

$$W(p, 1) = 1,\ W(1, k) = 1,$$
$$W(0, k) = 1.$$

Первое равенство означает, что на одном стеке можно отсортировать только 1 вагон, второе — что за одно вытягивание можно отсортировать тоже только 1 вагон, и третье — что за 0 шагов можно отсортировать тоже только 1 вагон.

**Лемма 4.** *$W(2, 2) = 2$.*

Меньшим числом не обойтись, а за 2 вытягивания ясно как сделать.

**Лемма 5.** *Если $p < k$, то $W(p, k) = W(p, p)$.*

Действительно, если мы можем использовать $p$ вытягиваний, то и стеков можем использовать не больше $p$, ведь из каждого использованного стека вагоны когда-нибудь придется вытягивать.

Будем в процессе решения нашей задачи составлять таблицу, где в $p$-й строке и $k$-м столбце стоит $W(p, k)$.

Следующее утверждение является основанием для дальнейшей индукции.

**Теорема 3.**

$$W(k+1, k+1) = 2W(k, k).$$

Будем доказывать это утверждение по индукции.

1°. Для $k = 1$ это верно: $2 = W(2,2) = 2W(1,1) = 2 \times 1$.

2°. Предположим, мы умеем сортировать состав, идущий в обратном порядке, из $W(k, k)$ вагонов на горке из $k$ стеков за $k$ вытягиваний. Назовем этот алгоритм $(A(k))$. Рассмотрим теперь на горке с $(k+1)$ стеком состав из $2 \times W(k, k)$ вагонов. Напишем номера всех вагонов в двоичной системе, начиная с нуля (напомним, что нулевой вагон самый правый). Тогда алгоритм $(A(k+1))$ состоит в следующем:

(1) Все четные вагоны в первом роспуске помещаются в $(k+1)$-й стек.

(2) У нечетных вагонов отбрасывается последний разряд (т.е. номер $2m+1$ переходит в номер $m$), и они распускаются в стеки с 1-го по $k$-й в соответствии с алгоритмом $(A(k))$.

(3) Вагоны из $(k+1)$-го стека вытягиваются, у них отбрасывается последний разряд (т.е. номер $2m$ переходит в номер $m$), и они распускаются в стеки с 1-го по $k$-й в соответствии с алгоритмом $(A(k))$.

(4) Дальше вагоны сортируются в соответствии с алгоритмом $(A(k))$:

Предыдущее рассуждение доказывает, что $W(k+1, k+1) \ge 2 \times W(k, k)$.

3°. Докажем теперь обратное неравенство: $W(k+1, k+1) \le 2 \times W(k, k)$.

Действительно, пусть на горке стоит состав из $n > 2 \times W(k, k)$ вагонов. Тогда после первого роспуска и первого вытягивания либо на горке состав из $n_1 > W(k, k)$ вагонов, либо в

оставшихся $k$ стеках находится суммарно $n_2 > W(k, k)$. В каждом из этих случаев за оставшиеся $k$ вытягиваний не удастся отсортировать эти вагоны. Теорема 3 доказана.

**Следствие 2.** $W(k, k) = 2^{k-1}$.

Мы уже можем заполнить большую часть нашей таблицы. Вот как теперь она выглядит (рис.7).

| Число вытягиваний ($p$) \ Число стеков ($k$) | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 3 | 1 | | 4 | 4 | 4 | 4 | 4 | 4 |
| 4 | 1 | | | 8 | 8 | 8 | 8 | 8 |
| 5 | 1 | | | | 16 | 16 | 16 | 16 |
| 6 | 1 | | | | | 32 | 32 | 32 |
| 7 | 1 | | | | | | 64 | 64 |

*Рис. 7*

Попробуем применить идею предыдущего доказательства к случаю горки с двумя стеками.

**Теорема 4.** *$k = 2$. Пусть $F(p)$ — $p$-е число Фибоначчи, т.е. $p$-е число из последовательности $F(p)$, удовлетворяющей рекуррентному соотношению $F(p + 1) = F(p) + F(p - 1)$ и начальным данным $F(0)=1$, $F(1) = 1$, $F(2) = 2$. Тогда $W(p, 2) = F(p)$.*

Будем доказывать и это утверждение по индукции.

1°. $W(1, 2) = 1 = F(1)$, $W(2, 2) = 2 = F(2)$. Можно еще проверить, что $W(3, 2) = 3$.

Пусть теперь при всех $2 \le l < p$ за $l$ вытягиваний можно отсортировать состав из не более $F(l)$ вагонов, идущих в обратном порядке. Докажем, что то же верно и для $l = p$.

2°. Докажем, что $W(p, 2) \le F(p)$, т.е. состав из большего числа вагонов отсортировать нельзя. Пусть это не так. Пусть $n > F(p)$. Тогда после первого роспуска в стеках оказалось $n_1 \ge n_2$ вагонов. Так как $n > F(p)$, а $F(p) = F(p - 1) + F(p - 2)$, то хотя бы одно из чисел $n_i$ строго больше, чем соответствующее число $F(p - i)$, $i = 1,2$.

а) Если $n_1 > F(p - 1)$, то после первого же вытягивания у нас есть «состав» из $n_1$ вагонов, идущих в обратном порядке, который по предположению индукции нельзя отсортировать за $p - 1$ вытягивание.

б) Если $n_2 > F(p - 2)$, то после второго вытягивания у нас на горке окажется неотсортированный состав из не менее $n_2$ вагонов, идущих в обратном порядке, который мы не можем по предположению индукции отсортировать за $p - 2$ вытягивания.

Это же рассуждение позволяет рекурсивно построить алгоритм, с помощью которого за $p + 1$ вытягивание можно отсортировать состав из $F(p + 1)$ вагонов, идущих в обратом порядке.

3°. Пусть алгоритм $(B(p))$ сортирует состав из $F(p)$ вагонов, идущих в обратном порядке. Пусть после первого роспуска алгоритма $(B(p))$ в первом стеке окажется $F(p - 1)$ вагон, а во втором — $F(p - 2)$. Покрасим вагоны в первом стеке в синий цвет, а во втором — в желтый, и вернем ситуацию назад во времени. У нас на горке стоит $F(p)$ вагонов. Вставим после каждого синего вагона зеленый (т.е. зеленый вагон правее синего), и перенумеруем состав в убывающем порядке. Теперь у нас на горке стоит состав из $F(p + 1) = F(p) + F(p - 1)$ вагонов.

Алгоритм $(B(p + 1))$ будет работать так:

(1) во время первого роспуска синие вагоны отправляются в 1-й стек, а желтые и зеленые во второй;

(2) вагоны из второго стека вытягиваются (1-е вытягивание);

(3) зеленые вагоны распускаются в первый стек, а желтые во второй.

Теперь в 1-м стеке все зеленые вагоны стоят левее синих.

Перенумеровав вагоны в 1-м и 2-м стеках, мы получим ситуацию после первого роспуска в алгоритме $(B(p))$. Дальше сортируем с помощью алгоритма $(B(p))$.

Теорема 4 доказана.

Теперь уже ничего не стоит сформулировать аналог теоремы 4 для общего случая. Как вы уже догадались, общий алгоритм является обобщением сортировки Фибоначчи.

**Определение 4.** Введем последовательность $u_k(k) = 2^{k-1}$, задав ее рекуррентно:

$$u_k(p) = u_k(p-1) + u_k(p-2) + \ldots + u_k(p-k), \quad p > k,$$

$$u_k(1) = 1, \ \ldots, \ u_k(k) = 2^{k-1} \qquad (*)$$

**Теорема 5.** $W(p, k) = u_k(p)$.

Доказательство этого утверждения естественно обобщает доказательство теоремы 4. Его мы тоже проведем по индукции.

1°. При $p \le k$ наше утверждение следует из теоремы 3, леммы 5, следствия 2 и равенства $(*)$.

Рассмотрим случай $p > k$. Пусть при всех $k < l < p$ за $l$ вытягиваний можно отсортировать состав из не более $u_k(l)$ вагонов, идущих в обратном порядке. Докажем, что то же верно и для $l = p$.

2°. Докажем, что $W(p, k) \le u_k(p)$, т.е. состав из большего числа вагонов отсортировать нельзя. Пусть это не так. Пусть $n > u_k(p)$ и пусть после первого роспуска в стеках оказалось $n_1 \ge n_2 \ge n_3 \ge \ldots \ge n_k$ вагонов. Сравним эту последовательность с последовательностью $u_k(p-1) \ge u_k(p-2) \ge \ldots \ge u_k(p-k)$. Так как $n > u_k(p)$, то существует номер $m$ такой, что $n_m > u_k(p-m)$, а это означает, что среди чисел $n_1 \ge n_2 \ge n_3 \ge \ldots \ge n_k$ не меньше $m$ строго больших $u_k(p-m)$. Следовательно, после $m$ вытягиваний мы будем иметь или на горке, или в одном из стеков последовательность из не менее $u_k(p-m) + 1$ вагона, номера которых идут в убывающем порядке, и которую, по индукционному предположению, мы не можем отсортировать за оставшиеся $p - m$ вытягиваний.

Так же, как и в теореме 4, строится рекурсивный алгоритм.

3°. Пусть алгоритм $\left(C_k(p)\right)$, $p > k$, сортирует состав из $u_k(p)$ вагонов, идущих в обратном порядке. Пусть после первого роспуска алгоритма $\left(C_k(p)\right)$ в первом стеке окажется $u_k(p-1)$ вагон, во втором — $u_k(p-2)$ и т.д., в $k$-м — $u_k(p-k)$. Покрасим вагоны в первых $(k-1)$ стеках в синий цвет, а в последнем — в желтый, и вернем ситуацию назад во времени. У нас на горке стоит $u_k(p)$ вагонов. Вставим после каждого синего вагона зеленый (т.е. зеленый вагон правее синего), перенумеруем состав в убывающем порядке. Теперь у нас на горке стоит состав из

$$u_k(p+1) = u_k(p) + \left(u_k(p-1) + u_k(p-2) + \ldots + u_k(p-k+1)\right)$$

вагонов.

Алгоритм $\left(C_k(p+1)\right)$ работает так:

(1) во время первого роспуска синие вагоны отправляются в первые $(k - 1)$ стеки в соответствии с алгоритмом $\left(C_k(p)\right)$, а желтые и зеленые в последний;

(2) вагоны из последнего стека вытягиваются (1-е вытягивание);

| Число вытягиваний ($p$) \ Число стеков ($k$) | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 3 | 1 | 3 | 4 | 4 | 4 | 4 | 4 | 4 |
| 4 | 1 | 5 | 7 | 8 | 8 | 8 | 8 | 8 |
| 5 | 1 | 8 | 13 | 15 | 16 | 16 | 16 | 16 |
| 6 | 1 | 13 | 24 | 29 | 31 | 32 | 32 | 32 |
| 7 | 1 | 21 | 44 | 56 | 61 | 63 | 64 | 64 |
| 8 | 1 | 34 | 81 | 108 | 120 | 125 | 127 | 128 |

*Рис. 8*

(3) зеленые вагоны распускаются в первые $(k-1)$ стеки в соответствии с алгоритмом $(C_k(p))$ (они помнят, перед какими вагонами они стояли), а желтые в последний.

Теперь в первых $(k-1)$ стеках все зеленые вагоны стоят левее синих.

Вспоминая номера синих и желтых вагонов при сортировке алгоритмом $(C_k(p))$ (зеленые вагоны получают номера соответствующих синих), мы получим ситуацию после первого роспуска в алгоритме $(C_k(p))$. Дальше сортируем с помощью алгоритма $(C_k(p))$. Теорема 5 доказана.

Окончательный вид таблицы $W(p, k)$ показан на рисунке 8.

## О некоторых рекурсивных программах и избавлении от рекурсии

Этот параграф для тех, кто умеет программировать или любит решать олимпиадные задачи по программированию. Но в нем может разобраться и любой школьник, а может и безболезненно пропустить его.

В программистском практикуме встречается задача о вычислении $s$-го числа Фибоначчи:

$$F(0) = 1,\ F(1) = 1,\ F(s) = F(s-1) + F(s-2),\ s > 1.$$

Вот рекурсивная программа вычисления $F(s)$:

```
functionF(s : integer) : integer;
    |{s >=0}
begin
    |if(s = 0)or(s = 1)thenbegin
    ||F :=1;
    |endelsebegin1 < s
   ||F :=F(s - 1) + F(s - 2);
    |end;
end
```

Что можно сказать о времени работы этой программы и необходимой для нее памяти?

Используемая программой память пропорциональна $s$. Глубина рекурсии ($s$) умножается на количество памяти, необходимое для одного экземпляра процедуры, т.е. на константу. А вот время растет экспоненциально, так как вычисление $F(s)$ сводится к двум вызовам $F(s-1)$ и $F(s-2)$, те — к четырем вызовам $F(s-2)$, $F(s-3)$, $F(s-3)$ и $F(s-4)$ и так далее. Поэтому время растет экспоненциально, правда, не как $2^s$, а как $F(s) \simeq \lambda_1^s$, где $\lambda_1 = \frac{1+\sqrt{5}}{2}$ — наибольший корень уравнения $\lambda^2 = \lambda + 1$.

Хорошо известно, как избежать рекурсии (если вы случайно не знаете, как это сделать, то попробуйте, прежде чем читать дальше, придумать самостоятельно нерекурсивную программу вычисления $F(s)$).

Нужно вместо одного значения $F(s)$ вычислить одновременно пару (вектор) $(F(s), F(s-1))$, начиная с пары (1, 1). Вот кусок этой программы:

```
begin
    |c := b;
    |b := a;
    |a := a + c;
end
```

Повторив это вычисление $s$ раз, начиная со значений $a := 1$; $b := 1$;, вы получите в $a$ значение $F(s)$. Время работы такого алгоритма порядка $s$, а памяти он требует порядка константы.

**Упражнение 2.** Допишите аккуратно вторую программу и сравните ее работу с работой первой.

Прием, который был применен, обычно называется *индуктивным расширением по Кушниренко*. Прочитать об этом можно в совершенно замечательной книге А.Х.Шеня «Программирование: Теоремы и задачи».

В нашем случае ситуация аналогична и работа рекурсивной процедуры усугубляется еще тем, что каждый рекурсивный вызов обращается не к простой функции, а к сложной программе.

Избавление от рекурсии в нашем случае заключается в обращении к алгоритму поразрядной сортировки. Только система счисления нам понадобится не 2-ичная или основанная на других геометрических прогрессиях, а основанная на последовательности Фибоначчи.

## Еще раз о системах счисления

Более общий способ построения систем счисления таков. Задается некоторый «базис» системы — набор целых чисел $1 = q_0 < q_1 < q_2 < \ldots$ Для того чтобы записать число $n$ в этой системе счисления, нужно выбрать наибольшее из чисел $q_s \le n$ и разделить $n$ на $q_s$ с остатком: $n = a_s q_s + n_{s-1}$. Затем нужно разделить $n_{s-1}$ на $q_{s-1}$ с остатком: $n_{s-1} = a_{s-1}q_{s-1} + n_{s-2}$ и т.д. В результате получатся такие равенства:

$$n = a_s q_s + n_{s-1},$$
$$n_{s-1} = a_{s-1}q_{s-1} + n_{s-2},$$
$$\ldots$$
$$n_0 = a_0 q_0 \left(n_0 = a_0\right),$$

или

$$n = a_s q_s + a_{s-1}q_{s-1} + \ldots + a_0 q_0.$$

Число $n$ записывается в виде выражения $\overline{a_s a_{s-1} \ldots a_0}$, и эта запись называется записью числа в системе счисления с базисом $1 = q_0 < q_1 < q_2 < \ldots$ Числа $a_i$ играют роль цифр и удовлетворяют неравенствам $a_i < \frac{q_{i+1}}{q_i}$. Заметим, что в таких системах счисления в разных разрядах могут стоять, вообще говоря, разные наборы цифр. Десятичная система обладает базисом из степеней десятки, двоичная из степеней двойки, $q$-ичная из степеней числа $q$.

Для следующего алгоритма нам понадобится система счисления с базисом 1, 2, 3, 5, 8, 13, ... — последовательностью чисел Фибоначчи. Такая система так и называется *фибоначчиевой*. Вот пример записи первых девяти чисел в фибоначчиевой системе счисления:

$$0 = 0_f,\quad 1 = 1_f,\quad 2 = 10_f,\quad 3 = 100_f,$$
$$4 = 101_f,\quad 5 = 1000_f,\quad 6 = 1001_f,$$
$$7 = 1010_f,\quad 8 = 10000_f.$$

С этой системой можно познакомиться по книге Н.Н.Воробьева «Числа Фибоначчи».

**Задача 12.** Докажите, что в системе счисления Фибоначчи, как и в двоичной, используются всего две цифры 0 и 1 и запись любого числа в фибоначчиевой системе счисления не содержит двух рядом стоящих единиц.

Посмотрим теперь, что будет происходить на горке с двумя стеками, если мы будем сортировать состав по

алгоритму двоичной сортировки, а номера вагонов запишем в фибоначчиевой системе счисления. Вагоны, которые на $k$-м шаге попали в стек с номером 1 (т.е. у которых в $k$-м разряде стоит 1), будут вытянуты в хвосте состава и после следующего роспуска все вместе отправятся в 0-й стек (ведь перед 1 обязательно стоит 01).

Возникает идея: не будем вытягивать эти вагоны из стека, а на $k$+1-м шаге перенумеруем стеки.

## Алгоритм ($F$) поразрядной сортировки Фибоначчи

(1) Нумеруем вагоны в фибоначчиевой системе.

(2) На $k$-м шаге распускаем состав в соответствии с поразрядной сортировкой.

(3) Затем вытягиваем вагоны из 0-го стека.

(4) Перенумеровываем стеки для $k$ + 1-го шага.

**Упражнение 3.** Разберитесь на примерах решения задач 6,б) и 7,б), как работает алгоритм ($F$).

**Задача 13.** Попробуйте, прежде чем читать дальше, обобщить этот алгоритм для горки с большим, чем 2, числом стеков.

## Обобщение алгоритма ($F$) на несколько стеков

Прежде чем обобщать алгоритм сортировки Фибоначчи для горки с числом стеков большим двух, попробуем переформулировать свойство фибоначчиевой системы из задачи 12.

Запишем его так:

ф2.0) перед цифрой 0 может стоять любая цифра,

ф2.1) перед цифрой 1 может стоять только цифра 0.

Обобщим эти свойства для создания «системы счисления» для любого $k$, например для $k = 3$.

Построим «систему счисления», использующую три цифры 0, 1, 2 и удовлетворяющую следующим свойствам:

ф3.0) перед цифрой 0 может стоять любая цифра,

ф3.1) перед цифрой 1 может стоять только цифра 0,

ф3.2) перед цифрой 2 может стоять только цифра 1.

Вот набор одноразрядных чисел этой системы: 0, 1, 2.

Множество двузначных чисел этой системы: 00, 01, 10, 12, 20.

Множество трехзначных чисел включает в себя 9 чисел:

000, 001, 010, 012, 100, 101, 120, 200, 201.

Уже по набору этих «чисел» видно, что $3 < W(3, 3) = 4$; $5 < W(4, 3) = 7$; $9 < W(5, 3) = 13$. Т.е. мы получили что-то не то.

Кроме того, то, что мы сейчас определили, не является системой счисления. Ведь в любой системе счисления, если существует число 2, то должно существовать и число 02, и число 002 и т.д. А в нашей «системе» есть число 2, но нет числа 02. Наши числа — это, скорее, «слова в некотором языке с алфавитом и с некоторыми правилами». Эти правила называются *контекстно-свободной грамматикой.*

Постараемся объяснить, что это такое. Если вас заинтересует, как применяются КС-грамматики в программировании, то и об этом можно прочитать в упоминавшейся уже книге А.Х.Шеня.

## Контекстно-свободные грамматики

Рассмотрим конечный алфавит, обозначим его буквы $a$, $b$, $c$, ... Множество слов в этом алфавите мы будем называть *языком*. Но в качестве слов мы будем рассматривать не произвольные наборы букв, а только те, которые можно получить по некоторым правилам, называемым *грамматическими*. Грамматические правила устроены так: некоторые слова или подслова можно заменять на другие.

Более формально,

**Определение 5.** Для того чтобы задать контекстно-свободную грамматику (КС-грамматику), нужно

(1) задать множество символов алфавита, эти символы называются *терминальными* (окончательными);

(2) задать некоторое другое множество символов, эти символы называются *нетерминальными* (промежуточными);

(3) выбрать среди нетерминалов один символ, называемый *начальным*;

(4) задать конечное число правил, имеющих вид $S \to X$, где $S$ — некоторый нетерминальный символ, а $X$ — слово, в которое могут входить как терминальные, так и нетерминальные символы.

*Выводом* в этой грамматике называется последовательность слов $X_0$, $X_1$, ..., $X_n$, в которой $X_0$ — начальный символ, а $X_{k+1}$ получается из $X_k$ заменой некоторого нетерминального символа $S$ на слово $X$ по одному из правил грамматики. Слово, составленное из терминальных символов, называется *выводимым*, если существует вывод, который этим словом кончается. Множество всех выводимых слов, состоящих из терминальных символов, называется *языком, порождаемым данной грамматикой.*

Обычно интересуются вопросом: «выводимо ли данное слово в данной грамматике?» Алгоритмы, которые для любого слова отвечают на этот вопрос, составляют существенную часть современных трансляторов, которая называется «грамматическим разбором».

Наша задача скромнее: «построить КС-грамматику, поразрядная сортировка $p$-буквенных слов языка которой эквивалентна алгоритму $\left(C_k(p+1)\right)$».

**Пример 3.** Рассмотрим алфавит, состоящий из одной буквы $a$ и правил

1) $S \to$,

2) $S \to aS$.

$S$ — начальный символ. Обычно, если не оговорено противное, начальным символом считается символ, стоящий в левой части первого правила. Первое правило означает, что $S$ можно заменять на пустое слово или, попросту говоря, вычеркивать из слова. Второе правило позволяет порождать новые слова, приписывая к ним букву $a$. Например, если нам нужно породить слово $aaa$, то, применяя второе правило, мы можем написать $S \to aS \to aaS \to aaaS$, а применив первое правило — $aaa \to Saaa$. Этот процесс, использующий грамматические правила языка, и есть вывод.

**Упражнение 4.** Сколько слов длины $n$ (такие слова мы будем еще называть $n$-буквенными) в этом языке?

**Пример 4.** Рассмотрим язык всех слов в алфавите из двух букв $a$ и $b$. Правила грамматики здесь такие:

1) $S \to$,

2) $S \to aS$,

3) $S \to bS$.

**Упражнение 5.** Выведите слова $aababba$.

**Пример 5.** Следующий язык описывает все регулярные скобочные структуры. Регулярные скобочные

структуры порождаются скобками в арифметических выражениях.

Например, структура (()())() может быть представлена выражением (7 − (5 + 2) − (3 − 4)) + (3 − 8), а структура ()(()) — выражением (2 + 3) − ((5 − 3)−9).

Вот пример всех скобочных структур, порожденных не более чем тремя парами скобок:

(), ()(), (())(), (()), ()(()), (())(), (()()), ((())).

Число регулярных скобочных структур из $n$ пар скобок задает последовательность

1, 1, 2, 5, 14, 42, 132, ...

известную, как *числа Каталана*.

Правила вывода этого языка задаются двумя правилами:

1) $S \to$,

2) $S \to S(S)$.

Вот вывод слова (()())():

$S \to S(S) \to S(S)(S) \to S(S(S))(S) \to$
$\to S(S(S)(S))(S) \to (()())()$ .

**Упражнение 6.** Выведите слово ((()())()).

**Задачи**

**14.** Напишите программу, выводящую регулярные скобочные структуры.

**15.** Найдите где-нибудь определение чисел Каталана и докажите по индукции, используя правила вывода, что число скобочных структур из $n$ пар скобок задается числами Каталана.

**16.** Рассмотрим язык, содержащий только конечное число слов. Напишите грамматические правила такого языка.

**17.** Придумайте правила вывода в языке, порожденном двумя символами 0 и 1, во всех словах которого символы 1 стоят перед символами 0. Например, 111100 и т.д.

**18.** Рассмотрим язык регулярных скобочных структур, содержащих две пары скобок () и []. Напишите грамматические правила для этого языка.

Как мы уже говорили, основное свойство системы Фибоначчи — это отсутствие двух единиц подряд в записи любого числа. Тогда язык Фибоначчи — это язык всех слов в алфавите из двух букв 0 и 1, не содержащих подслова 11.

Вот все фибоначчиевы слова, содержащие не более четырех букв:

0, 1,
00, 01, 10,
000, 001, 010, 100, 101,
0000, 0001, 0010, 0100, 0101, 1000, 1001, 1010.

Это хорошо знакомые вам 1-, 2-, 3- и 4-значные числа системы счисления Фибоначчи. Попробуем сформулировать грамматические правила. Но мы уже формулировали что-то похожее на них:

ф2.0) перед цифрой 0 может стоять любая цифра,

ф2.1) перед цифрой 1 может стоять только цифра 0.

Действительно, эти правила позволяли по числу длины $n$ строить число длины $n + 1$. Осталось их только записать в виде КС-грамматики. Заметим, что слова в примерах 3, 4, и 5 порождаются по-разному: в примерах 3 и 4 — слева направо, а в примере 5 — справа налево. Правила из примеров 3 и 4 называются праворекурсивными, а из примера 5 — леворекурсивными. Правила для чисел Фибоначчи «ф2.0» и «ф2.1» леворекурсивны, а алгоритмы $(A)$, $(B)$ и $(C)$ — праворекурсивны! Попробуем переписать наши правила на праворекурсивные. Получаем:

f2.0) после цифры 0 может стоять любая цифра,

f2.1) после цифры 1 может стоять только цифра 0.

Чтобы построить КС-грамматику Фибоначчи $(\Phi)$, введем 2 нетерминальных символа $L_0$ и $L_1$. Первый символ будет «помнить», что перед ним стояла цифра 0, а второй — что стояла цифра 1. Тогда грамматические правила запишутся так:

(1) $L_0 \to 0L_0$,

(2) $L_0 \to 1L_1$,

(3) $L_1 \to 0L_0$.

Осталось только дописать правила исчезновения нетерминальных символов. Окончательно все правила выглядят так:

(1) $L_0 \to$,

(2) $L_1 \to$,

(3) $L_0 \to 0L_0$,

(4) $L_0 \to 1L_1$,

(5) $L_1 \to 0L_0$.

Начальный символ $L_0$.

**Упражнения**

**7.** Выведите слова 0100, 0101.

**8.** Докажите, что любое слово фибоначчиева языка может быть получено из этих грамматических правил.

**Задача 19.** Придумайте какие-нибудь другие грамматические правила для фибоначчиева языка.

**Упражнение 9.** Придумайте для языков примеров 1 и 2 леворекурсивные грамматические правила.

Попробуем теперь также сформулировать алгоритм $(C)$ (прочитайте еще раз внимательно, как делается индукционный шаг в теореме 5). Посмотрим на зеленый вагон $a$, который мы вставили правее синего $b$. Во время второго роспуска (и всех последующих) $a$ попадает в тот же стек, что и $b$. Это означает, что предпоследние символы (да и во всех следующих разрядах) у $a$ и $b$ совпадают. Пусть предпоследний разряд у них $l$. Но при первом роспуске зеленый вагон попал в стек, из которого вагоны вытягиваются. Это означает, что последний символ в «номере» зеленого вагона — ноль (договоримся, что вытягиваются вагоны из нулевого стека). Получается первое правило:

c1) «после цифры $l$ может стоять цифра 0».

Рассмотрим теперь желтые вагоны. Ясно, что последний символ в их номере 0, а предпоследний $k - 1$. Получаем следующее правило:

c2) «после цифры $k - 1$ может стоять цифра 0».

И наконец, рассмотрим синий вагон. Предпоследний символ в его номере — $l$. А вот последний — $l + 1$, ведь вытянув первый раз из 0-го стека, мы сдвинули нумерацию стеков на единицу! Последнее правило:

c3) «после цифры $l$ может стоять цифра $l + 1$».

Теперь так же, как и в случае грамматики $(\Phi)$, рассмотрим грамматику $(Q_k)$ с $k$ терминальными символами

0, 1, 2, ..., $k - 1$,

$k$ нетерминальными символами

$L_0, L_1, L_2, \ldots, L_{k-1}$

и правилами вывода:

$L_{i-1} \to 0L_0$,

$L_{i-1} \to iL_i \quad (1 \le i < k-1)$,

$L_{k-1} \to 0L_0$,

$L_i \to \quad (1 \le i < k-1)$,

где $L_0$ — главный нетерминальный символ. Ясно, что при $k = 2$ грамматика $(Q_2)$ совпадает с $(\Phi)$ и, следовательно, совпадают языки, ими порожденные. Так же, как и в случае поразрядной сортировки Фибоначчи $(F)$, можно сформулировать

## Алгоритм $(K)$ обобщенной поразрядной сортировки Фибоначчи

(1) Породим в языке $Q_k$ все слова длины $p - 1$, где $W(p - 1, k) < n \le W(p, k)$, лексикографически

их упорядочим (естественно, $0 < 1 < \ldots < k-1$) и присвоим соответствующим вагонам.

(2) Алгоритм начинается с нулевого шага.

(3) На $l$-м шаге распускаем состав в соответствии с символами в $l$-м разряде, как в поразрядной сортировке.

(4) Затем вытягиваем вагоны из 0-го стека (т.е. того стека, где собрались вагоны, у которых в $l$-м разряде стоит 0).

(5) Перенумеровываем стеки для $(l+1)$-го шага в соответствии с символами в $(l+1)$-м разряде, пустой стек получает номер $k-1$.

**Следствие 3.** *Число $(p-1)$-буквенных слов в языке, порожденном грамматикой $Q_k$, равно $u_k(p)$.*

**Следствие 4.** *Если $p \le k$, то алгоритм (K) совпадает с алгоритмом (A).*

**Следствие 5.** *Если $p \le k$, то число $(p-1)$-буквенных слов в языке, порожденном грамматикой $Q_k$, равно $u_k(p) = 2^{p-1}$.*

В одной из первых задач спрашивалось, как бы вы предложили нумеровать вагоны, если получателю неважно, в каком порядке он получит свои вагоны?

Обычно все вагоны для одного получателя занумеровывают одним номером. Такой объект, где есть несколько первых номеров, несколько

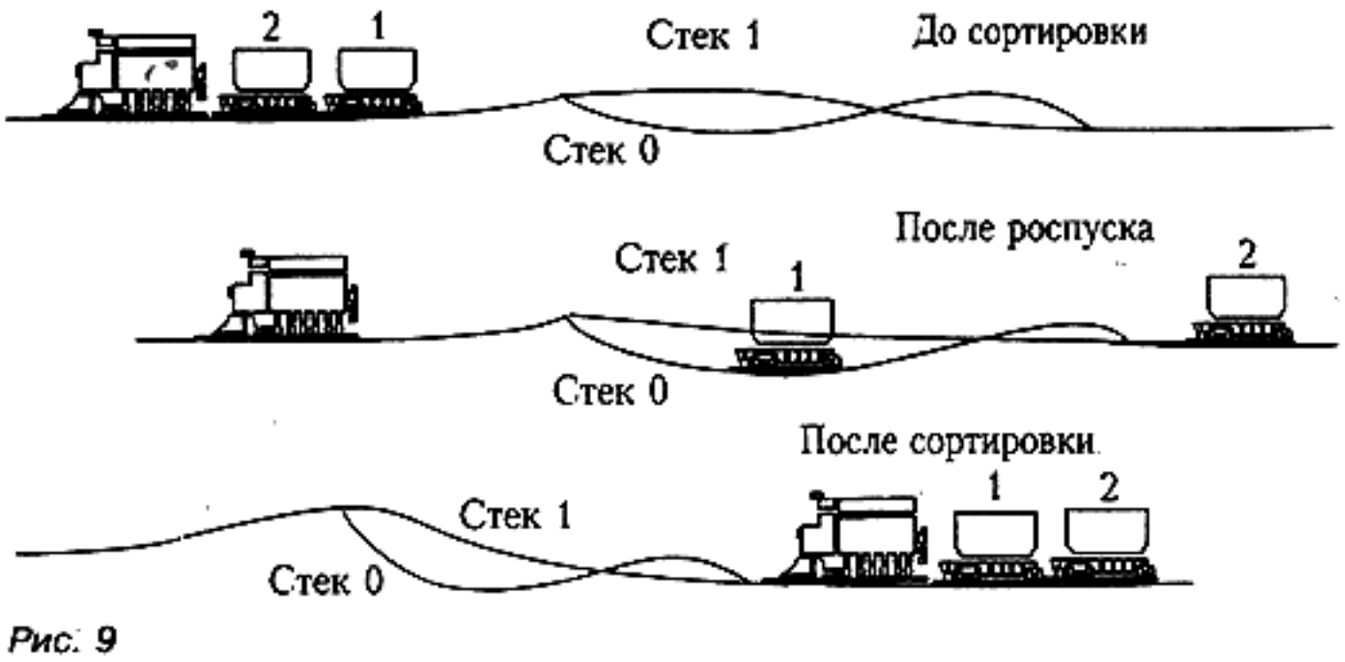


*Рис. 9*

вторых, ..., несколько $n$-х называют *мультимножеством*. Можно рассматривать перестановки мультимножества

$$S = \{\underbrace{1\ldots1}_{m_1}\underbrace{2\ldots2}_{m_2}\ldots\underbrace{s\ldots s}_{m_s}\}.$$

**Задача 20.** Введите для мультимножества понятие блока.

Алгоритм $(K)$ вместе с приведенными выше результатами полностью завершает решение задачи о **минимальном числе вытягиваний при сортировке железнодорожных составов.**

## Заключение

Итак, задача решена полностью. Теперь хочется остановиться и оглядеться. Давайте посмотрим, что мы получили. Основное — это таблица $W(p, k)$. Если вы в детстве считали вагоны проходящих мимо вас составов, то могли бы заметить, что редко их бывает больше 60, а число получателей редко бывает больше 20. Рассмотрев окончательную таблицу чисел $W(p, k)$ (рис.8), можно увидеть, что разумно делать горки с 3–4 стеками, но даже горка с 2 стеками не сильно увеличивает число вытягиваний. Но повысит ли наш алгоритм существенно производительность? Ведь когда состав сортируется на горке, она занята для других составов и они простаивают. А нельзя ли интенсифицировать процесс сортировки как-нибудь по-другому? Что мешает сортировать сразу несколько составов? Это принцип устройства горки — в стек можно класть с одной стороны и вынимать из него тоже с той же стороны. А не изменить ли устройство горки, сделав ее похожей на «очередь», в которую с одной стороны можно «класть» вагоны, а «вынимать» можно с другой? Но большинство составов не отсортируешь за один шаг. А не сделать ли несколько горок, соединенных последовательно, да еще и замкнуть всю эту систему в кольцо! Как, например, на рисунках 9 и 10.

Правда, такое кольцо должно занимать много места, гораздо больше, чем нынешние горки. И вот тут возникает последняя идея: а не убрать ли все это под землю! Ведь существование окончательного алгоритма означает, что составы можно сортировать без какого-либо присутствия человека. А вместо железнодорожных путей можно будет разбить прекрасные парки.

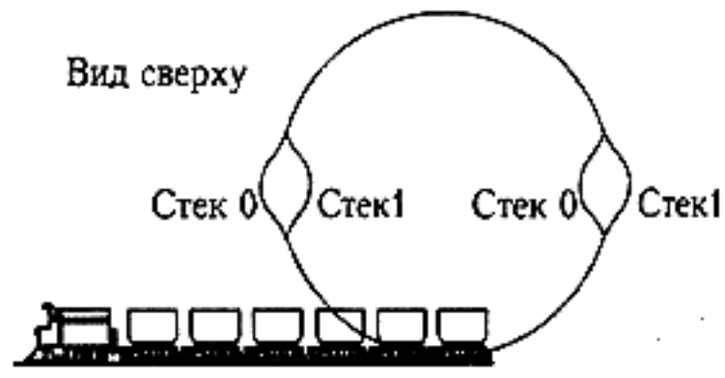


*Рис. 10*

## Задачи для самостоятельного исследования

**Задача 1°.** Оцените, каков порядок числа вагонов $W$, прошедших через стрелку $S$, в алгоритмах, приведенных в этой статье.

**Задача 2°.** Придумайте алгоритм сортировки, который каждую индивидуальную перестановку сортирует за наименьшее число

а) шагов,

б) вытягиваний,

если в каждом стеке помещается ограниченное число вагонов $n_1, n_2, \ldots, n_k$.

**Задача 3°°.** Пусть теперь разрешается распускать не весь состав и вытягивать из стеков не обязательно всю группу вагонов. Придумайте алгоритм сортировки, в котором каждая индивидуальная перестановка сортируется за наименьшее число $W$ вагонов, прошедших через стрелку $S$.