

# Внешние сортировки

1. Особенности внешней сортировки.

2. Сортировки слиянием

3. Другие методы внешних сортировок

4. Сравнение и анализ эффективности внешних сортировок

# Особенности внешней сортировки

Специфика любого алгоритма внешней сортировки – как разделить последовательность на части и как их слить.

**Для выполнения внешней сортировки производятся следующие операции:**

- 1) Находят нужный блок во внешней памяти. Для этого перемещают считывающую головку.
- 2) Производят считывание и передачу данных по каналу ввода-вывода в оперативную память.
- 3) В оперативной памяти осуществляют операции сравнения и перемещения.
- 4) Выполняют запись во внешнюю память в соответствии с п.1 и 2.

# Выбор алгоритма внешней сортировки

**При выборе алгоритма внешней сортировки учитываются следующие параметры:**

- а) объем оперативной памяти, используемой в качестве рабочей области;
- б) тип, число, объем ЗУ внешней памяти, используемых в качестве рабочей области;
- в) число каналов ввода-вывода;
- г) число и длина записей;
- д) время выполнения операций в оперативной памяти.

# Основные определения внешних сортировок

**Упорядоченным отрезком или серией** называется последовательность элементов, для которых выполняется условие упорядоченности.

Количество элементов данной последовательности называется *длиной упорядоченного отрезка (серии)*.

Например, файл А содержит 12 элементов  
17 31; 05 59; 13 41 43 67; 11 23 29 47,  
которые образуют 4 серии (разделены «;»).

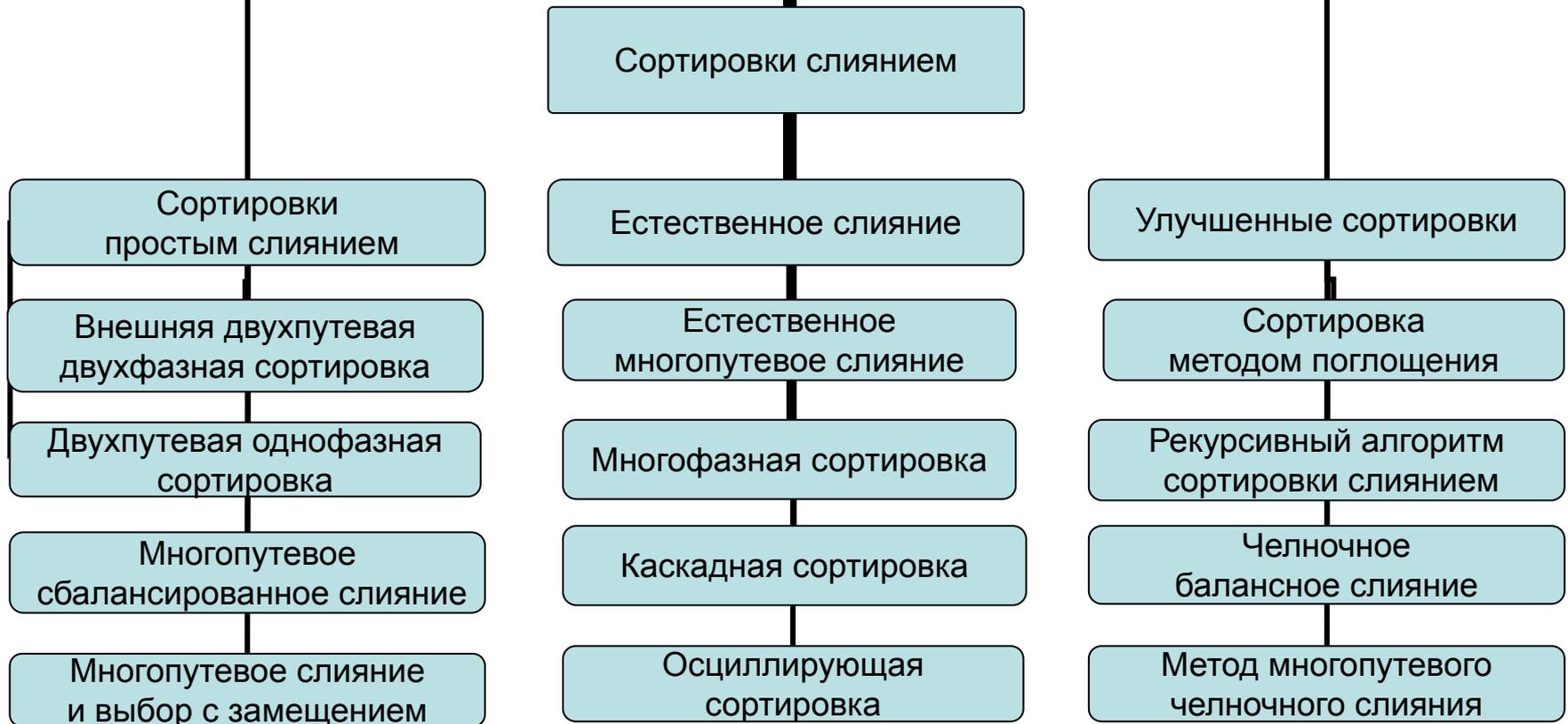
**Расстоянием между элементами** данных назовем разность номеров цилиндров, где они размещены.

**Прогоном** назовем прохождение файла в одном направлении со считыванием данных в память и, возможно, их возвратом в файл.

# Факторы, учитываемые при выборе метода сортировки

- Размер сортируемого массива.
- Длина ключа.
- Вид ключей.
- Исходное распределение данных.
- Возможность дублирования ключей.
- Длина записей.
- Логический порядок следования записей

# классификация методов внешней сортировки



# Другие методы внешней сортировки

- Сортировка с использованием специальных структур
- Разделительная (поразрядная) внешняя сортировка
- Быстрая альтернатива внешней сортировке
- И другие ( solid-state drive, SSD - твёрдотельный накопитель)

## характеристики сортировок слиянием

*Количество вспомогательных файлов, на которые идет распределение серий. Если данные распределяются на два вспомогательных файла, то сортировка называется двухпутевым слиянием, если на  $N$  ( $N > 2$ ) вспомогательных файлов, то — многопутевым слиянием.*

*Количество фаз в реализации сортировки. Фазой называются действия по однократной обработке всего множества. Например, в двухфазной сортировке отдельно реализуются две фазы: распределение и слияние. После того как произошло распределение данных по вспомогательным файлам, они объединяются и записываются в исходный файл. В однофазной сортировке обе эти фазы объединены в одну.*

# Внешняя двухпутевая двухфазная сортировка прямым (простым) слиянием

Суть алгоритма в следующем:

1. Последовательность  $A$  разбивается на две половины  $B$  и  $C$ ;
2. Части  $B$  и  $C$  сливаются, при этом одиночные элементы этих частей образуют упорядоченные пары.
3. Полученная последовательность  $A$  опять обрабатывается в соответствии с пунктом 1 и 2, но сливаются упорядоченные пары и получаются упорядоченные четвёрки.
4. Повторяя предыдущие шаги, сливаем четвёрки в восьмёрки, и т д , пока не будет упорядочена вся последовательность.

Смысл алгоритма – сближение близких элементов в одну группу

# Пример:

исходный файл A 31 17 05 59 13 41 67 43 11 23 29 47

проход 1 файл B 31 05 13 67 11 29

- файл C 17 59 41 43 23 47

- файл A 17 31 05 59 13 41 43 67 11 23 29 47

проход 2 файл B 17 31 13 41 11 23

- файл C 05 59 43 67 29 47

- файл A 05 17 31 59 13 41 43 67 11 23 29 47

проход 3 файл B 05 17 31 59 11 23 29 47

файл C 13 41 43 67

- файл A 05 13 17 31 41 43 59 67 11 23 29 47

проход 4 файл B 05 13 17 31 41 43 59 67

- файл C 11 23 29 47

- файл A 05 11 13 17 23 29 31 41 43 47 59 67

# Двухпутевая однофазная (сбалансированная) сортировка

- Фаза разделения является вспомогательной. Если объединить разделение со слиянием, то избавляемся от лишних операций.
- вместо слияния в одну последовательность результаты слияния сразу же распределяются по двум файлам, которые станут исходными для следующего прохода. Но она требует два входных и два выходных файла при каждом проходе. Такую сортировку называют двухпутевой однофазной сбалансированной сортировкой.
- Общее число пересылок  $M = n \cdot \log_2 n$ . Число сравнений практического значения не имеет, т.к. время обмена с ВЗУ значительно больше.

# Пример

исходный файл A	31 17 05 59 13 41 67 43 11 23 29 47
проход 1 файл F1	31 05 13 67 11 29
файл F2	17 59 41 43 23 47
проход 2 файл F3	17 31 13 41 11 23
файл F4	05 59 43 67 29 47
проход 3 файл F1	05 17 31 59 11 23 29 47
файл F2	13 41 43 67
проход 4 файл F3	05 13 17 31 41 43 59 67
файл F4	11 23 29 47
файл A	05 11 13 17 23 29 31 41 43 47 59 67

# Количество проходов

Если  $k$  — это номер прохода, а  $N$  — количество вспомогательных файлов, то длина серии определяется формулой  $N^k$ .

*Если после внутренней сортировки было получено  $S$  серий, причем  $2^{k-1} < S \leq 2^k$ , то процедура сбалансированного двухпутевого слияния произведёт ровно  $k = \lceil \lg S \rceil$  проходов*

## Признаками конца сортировки простым слиянием являются следующие условия

- длина серии не меньше количества элементов в файле (определяется после фазы слияния);
- количество серий - одна (определяется на фазе слияния);
- второй по счету вспомогательный файл при однофазной сортировке после распределения серий остался пустым.

# Многопутевое сбалансированное слияние

- Затраты на сортировку последовательностей пропорциональны числу проходов, так как при каждом проходе пересылаются все данные, в двухфазных сортировках – дважды, в однофазных – один раз. Один из способов сократить число пересылок – распределять серии в больше чем в два файла.
- Общее число проходов, необходимое для сортировки  $n$  элементов с помощью  $N$ -путевого слияния, равно  $k = \log_N n$ , а число пересылок при объединении фаз слияния и разделения в самом худшем случае будет  $M = n \log_N n$ .

# Пример многопутевого двухфазного слияния (N = 3)

исходный файл A    31 17 05 59 13 41 67 43 11 23 29 47  
проход 1    файл F1    31 59 67 23  
              файл F2    17 13 43 29  
              файл F3    05 41 11 47  
              файл A    05 17 31 13 41 59 11 43 67 23 29 47  
проход 2    файл F1    05 17 31 23 29 47  
              файл F2    13 41 59  
              файл F3    11 43 67  
              файл A    05 11 13 17 31 41 43 59 67 23 29 47  
проход 3    файл F1    05 11 13 17 31 41 43 59 67  
              файл F2    23 29 47  
              файл F3  
              файл A    05 11 13 17 23 29 31 41 43 47 59 67

# Пример многопутевого однофазного слияния (N = 3)

исходный файл A 17 31 05 59 13 41 43 67 11 23 29 47

проход 1 файл F1 17 59 43 23  
файл F2 31 13 67 29  
файл F3 05 41 11 47

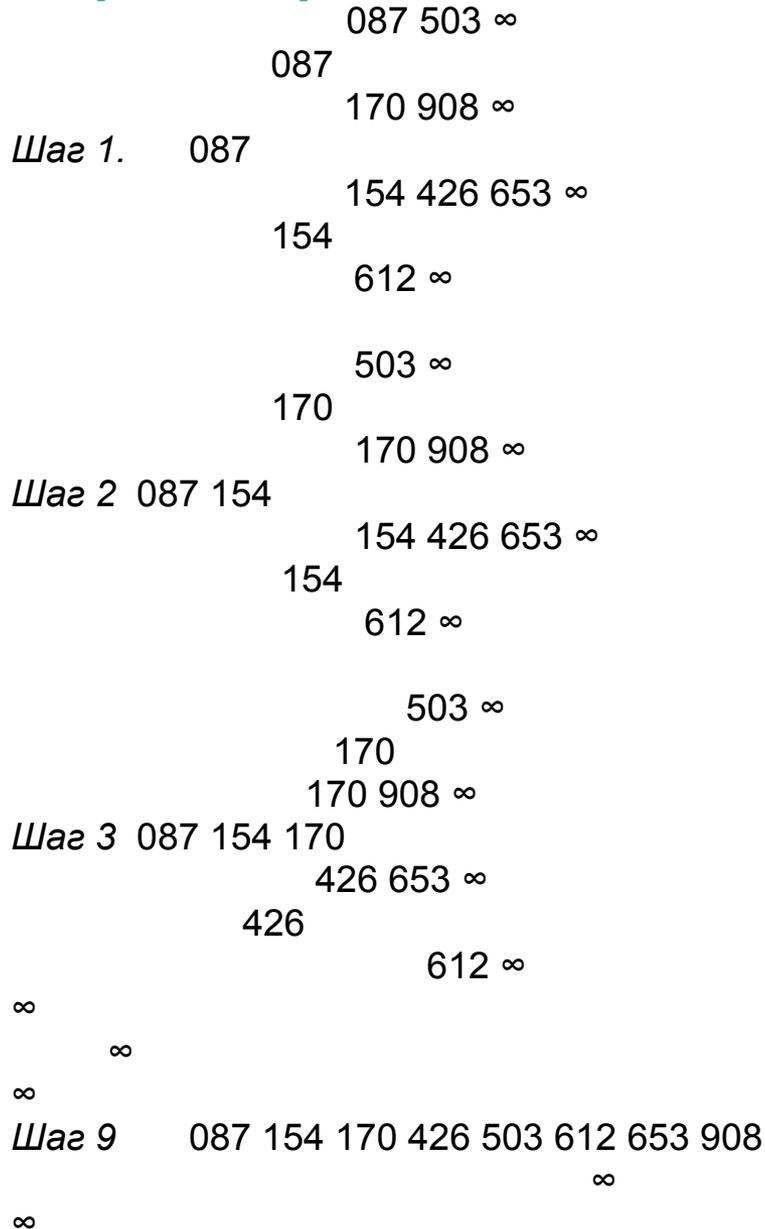
проход 2 файл F4 05 17 31 23 29 47  
файл F5 13 41 59  
файл F6 11 43 67

проход 3 файл F1 05 11 13 17 31 41 43 59 67  
файл F2 23 29 47  
файл F3  
файл A 05 11 13 17 23 29 31 41 43 47 59 67

# Многопутевое слияние и выбор с замещением

- Очевидным способом слияния  $P$  возрастающих серий будет следующий:
  - просмотреть первые записи каждой серии и выбрать из них ту, которая имеет минимальный ключ;
  - эта запись передается на выход и исключается из входных данных, затем процесс повторяется. В любой момент времени потребуются просмотреть только  $P$  ключей (один на каждую серию) и выбрать из них наименьший.
- Пока  $P$  не слишком велико, этот выбор удобно осуществлять, просто выполняя  $P-1$  сравнений для нахождения наименьшего из текущих ключей. Но если  $P$  равно, скажем, 8, то можно ускорить работу, используя *дерево выбора*, каждый раз потребуются примерно  $\lg P$  сравнений.

# пример 087 503 170 908 154 426 653 612



## Формирование и распределение начальных серий

- Если использовать свободные области оперативной памяти для внутренней сортировки, то можно увеличить длины серий.
- Если на этапе распределения начальных серий по файлам длины серий будут большими, то число проходов для сортировки исходного файла значительно сократится.
- Очевидный способ:
- из файла ввода в оперативную память переносят как можно больше записей, сортируют их одним из методов внутренней сортировки, а затем выводят. Преимуществом этого способа является то, что независимо от размера заключительного отрезка, начальные отрезки будут иметь одинаковую длину. Это свойство позволяет несколько упростить программу сортировки.

# Альтернативный алгоритм формирования начальных отрезков

1. Построить древовидную структуру с меньшим значением в корне.
2. Вывести корень.
3. Ввести следующую запись и выбрать значение ключа.
4. Если оно больше последнего выведенного значения,
  - 4.1. поместить его в корень.
  - 4.2. В противном случае:
    - 4.2.1. если древовидная структура не пуста, то последний элемент поместить в корень, а введенный элемент поместить в последнюю позицию; диапазон адресов древовидной структуры уменьшить на 1.
    - 4.2.2. Если древовидная структура пуста, то завершить обработку отрезка и возвратиться к шагу 1.
5. Восстановить древовидную структуру и перейти к шагу 2.

# пример

<i>Файл ввода</i>	78, 45, 72, 59, 20, 43, 85, 33, 92, 81, 34, 85, 16, 49, 61, ...
<i>Изменение в оперативной памяти</i>	78 <u>45</u> 72
<i>M=3. Подчеркнутое значение вы-</i>	78 <u>59</u> 72
<i>водится, на его место вводится</i>	78 20 <u>72</u>
<i>другое</i>	<u>78</u> 20 43
	<u>85</u> 20 43
	<hr/>
	33 <u>20</u> 43
	<u>33</u> 92 43
	81 92 <u>43</u>
	<u>81</u> 92 34
	<u>85</u> 92 34
	16 <u>92</u> 34
	<hr/>
	<u>16</u> 49 34
<i>Файл вывода</i>	45, 59, 72, 78, 85   20, 33, 43, 81, 85, 92   16, ...

*Конец первого отрезка*

*Конец второго отрезка*

Для формирования начальной серии можно предложить так же другой, более простой и эффективный способ. В оперативной памяти из считываемых записей входного файла строится древовидная таблица поиска. Записи в таблице размещаются в порядке их поступления из файла, никаких перестановок записей в таблице не производится. Дерево же организуется за счет образования соответствующих ссылок. Левый смешанный обход дерева дает возможность извлекать записи из дерева в порядке возрастания ключей.

Извлеченные записи пересылаются в очередной выходной файл. В таблицу считывается очередная порция записей файла для формирования следующей серии. Начальное распределение серий по файлам в случае естественного слияния и многопутевого сбалансированного слияния не вызывает затруднений, так как все файлы имеют одинаковое число серий.

# Естественное слияние

Сортировка простым слиянием не учитывает частичную упорядоченность данных. Поэтому размер сливаемых подпоследовательностей на  $k$ -м проходе  $\leq 2^k$ . В то же время при очередном слиянии две упорядоченные подпоследовательности длиной  $m$  и  $n$  можно было бы слить в одну длиной  $m+n$ .

Сортировка, при которой всегда сливаются две очередные серии, называется естественным слиянием.

# Естественное двухпутевое слияние

- Представим, что исходный файл разделён на два файла, каждый из которых содержит по  $n$  серий (или по  $n-1$ ). Тогда при слиянии образуется файл, содержащий также  $n$  серий. При каждом проходе число серий уменьшается вдвое и общее число пересылок в худшем случае равно  $n \log_2 n$ , а в среднем меньше.
- Процесс заканчивается, если при очередном слиянии в выходной файл будет записана только одна единственная серия

## Пример двухпутевого двухфазного естественного слияния

исходный файл А 17 31; 05 59; 13 41 43 67; 11 23 29 47

Т.е. имеем 4 серии, разделённых знаком «;».

проход 1 файл В 17 31; 13 41 43 67

файл С 05 59; 11 23 29 47

файл А 05 17 31 59; 11 13 23 29 41 43 47 67

проход 2 файл В 05 17 31 59

файл С 11 13 23 29 41 43 47 67

файл А 05 11 13 17 23 29 31 41 43 47 59 67

- Если использовать четыре файла, то фазы слияния и разделения можно объединить в одну фазу. Для этого достаточно полученные при слиянии файлов В и С серии поочерёдно направлять в файлы Д и Е, и наоборот
- Сортировка заканчивается, если при очередной фазе слияния – разделения образуется только одна серия и один из выходных файлов остаётся пустым, т.е. туда не будет записана ни одна серия.

- Так же как и простое слияние, сортировка естественным слиянием может быть двухпутевой или многопутевой, двухфазной или однофазной.
- Поскольку признаком конца серии в естественном слиянии является результат сравнения двух соседних элементов, две или несколько серий, распределенных друг за другом во вспомогательный файл, могут объединиться в одну

Например

FO: 1 2 9 3 39 11 4 18 13 5 16 24 15 4 25 17 317

После фазы распределения вспомогательные файлы будут иметь следующий вид:

F1: 1 2 9 11 13 15 17 317

F2: 3 39 ' 4 18 ' 5 16 24 ' 4 25

# *сбалансированное слияние*

Обратим внимание на то, что сортировка проходит эффективно, если количество серий, распределенных на вспомогательные файлы, приблизительно одинаковое.

Естественное слияние, у которого после фазы распределения количество серий во вспомогательных файлах отличается друг от друга не более чем на единицу, называется ***сбалансированным***, в противном случае речь идет о *несбалансированном* слиянии.

В первом вспомогательном файле оказалась одна упорядоченная серия, хотя записывалось туда пять серий.

Чтобы в сбалансированном слиянии серии распределялись корректно, необходимо во время записи очередной серии в файл выполнять следующую проверку: если серия является продолжением предыдущей, то записать в этот файл не одну, а две серии. Для сбалансированного слияния в приведенном выше примере данные после распределения будут иметь следующий вид:

FO: 1 2 9 3 39 11 4 18 13 5 16 24 15 4 25 17 317

F1: 1 2 9 11 ' 4 18 ' 5 16 24 ' 17

F2: 3 39 ' 13 15 ' 4 25 317

Признаками конца сортировки естественным слиянием являются следующие условия:

количество серий — одна (определяется на фазе слияния);

второй по счету вспомогательный файл для однофазной сортировки после распределения серий остался пустым.

# Многофазная сортировка

- Особенность этой сортировки – алгоритм слияния
- Метод многофазной сортировки основан на том, что имеется несколько входных файлов с разным числом серий и только один выходной файл. При каждом проходе серии из входных файлов сливаются до тех пор, пока входной файл с наименьшим числом серий не будет исчерпан. Тогда освободившийся файл становится выходным, начинается следующий проход, серии из всех остальных файлов сливаются в этот выходной файл. Процесс сортировки завершается, когда все серии будут объединены в одну серию.

## Пример из трех файлов

$f_1$  содержит 13 серий,  $f_2$  — 8 серий и  $f_3$  — выходной файл

Файлы	$f_1$	$f_2$	$f_3$
Исходное состояние	13	8	0
Первый проход	5	0	8
Второй проход	0	5	3
Третий проход	3	2	0
Четвертый проход	1	0	2
Пятый проход	0	1	1
Шестой проход	1	0	0

Сортировка завершена, отсортированные записи в файле  $f_1$ .

- Число проходов при многофазной сортировке будет зависеть от начального распределения серий по входным файлам. Кнут показал, что для хорошей работы многофазной сортировки с тремя файлами необходимо, чтобы числа начальных серий в двух выходных файлах были двумя соседними числами Фибоначчи.
- При сортировке на 6 файлах (5 входных) числа серий во входных файлах должны быть числами Фибоначчи четвертого порядка.
  - Напомним, что ряд, в котором каждый элемент является суммой двух предыдущих элементов, называется последовательностью Фибоначчи:
  - 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, .....
- В общем случае эта схема Фибоначчи требует приблизительно  $1.04 \log S + 0.99$  проходов, что делает ее сравнимой с четырехленточным сбалансированным слиянием, хотя она использует только три ленты. ( $S$  – количество серий)
- Эту идею можно обобщить на случай  $f$  лент при любом  $f \geq 3$ , используя  $(f - 1)$  ленточное слияние. Мы увидим, например, что в случае четырех лент требуется только около  $0.703 \log S + 0.96$  проходов по данным. Обобщенная схема использует обобщенные числа Фибоначчи. Рассмотрим следующий пример с шестью лентами

Точные фибоначчиевы распределения можно получить, "прокручивая" рассмотренную схему в обратную сторону, циклически переставляя содержимое лент. Например, при  $f = 6$  имеем следующее распределение серий:

Уровень,	f1	f2	f3	f4	f5	Сумма	Файл с окончательным результатом
0	1	0	0	0	0	1	f1
1	1	1	1	1	1	5	f 6
2	2	2	2	2	1	9	f 5
3	4	4	4	3	2	17	f 4
4	8	8	7	6	4	33	f 3
5	16	15	14	12	8	65	f 2
6	31	30	28	24	16	129	f1
.....	.....	.....	.....	.....	.....	.....	.....
n	$a^n$	$b^n$	$c^n$	$d^n$	$e^n$	$t^n$	f (k)
n+1	$a^n+b^n$	$a^n+c^n$	$a^n+d^n$	$a^n+e^n$	$a^n$	$4a^n+t^n$	f (k-1)

# Числа Фибоначчи

$$e^n = a^{n-1}$$

$$d^n = a^{n-1} + e^{n-1} = a^{n-1} + a_{n-2}$$

$$c^n = a^{n-1} + d^{n-1} = a^{n-1} + a_{n-2} + a_{n-3}$$

$$b^n = a^{n-1} + c^{n-1} = a^{n-1} + a_{n-2} + a_{n-3} + a_{n-4}$$

$$a^n = a^{n-1} + b^{n-1} = a^{n-1} + a_{n-2} + a_{n-3} + a_{n-4} + a_{n-5}$$

где  $a_0 = 1$ , и где полагаем  $a^n = 0$  при

$n = -1, -2, -3, -4$ .

Многофазовая сортировка требует оптимального распределения серий по файлам.

Для этого, во-первых, надо знать число серий в исходном файле, а оно становится известным только в процессе формирования серий.

Во-вторых, числа серий в файлах при оптимальном распределении по  $(f - 1)$  файлам должны быть числами Фибоначчи порядка  $(f - 2)$ , что возможно только при определенных значениях общего числа серий.

Недостающее число серий можно дополнить пустыми сериями, причем пустые серии нужно как можно более равномерно распределить по  $(f - 1)$  файлам.

# Каскадная сортировка

- *Каскадная сортировка* похожа на многофазную сортировку. Отличие заключается в самом процессе слияния:

сначала проводится  $(f-1)$  — путевое слияние в  $f$ -ый файл до тех пор, пока файл с номером  $f-1$  не опустеет;

затем ( $f$ -ый файл уже не затрагивается) проводится  $(f-2)$  — путевое слияние в  $(f-1)$ -ый файл;

затем  $(f-3)$  — путевое в  $(f-2)$ -ый файл; ...; потом двухпутевое слияние в третий файл,

а в конце копирование из файла номером 2 в первый файл.

Следующий проход работает по аналогичной схеме, только в обратном направлении.

# Распределение серий в каскадной сортировке

- Поскольку слияние в каскадной сортировке отличается от слияния в многофазной сортировке, то и начальное распределение данных осуществляется по-другому — количество серий в каждом из вспомогательных файлов должно быть другим. Оно определяется в соответствии с формулами:
- $a_1^0 = 1$      $a_i^0 = 0, i = 2, 3, \dots, f$
- $a_1^N = a_{i+1}^L + a_{f-i}^{L-1}, i = f-1, f-2, \dots, 1$
- или  $a_1^0 = 1$      $a_i^0 = 0, i = 2, 3, \dots, f$

$$a_1^L = \sum_{i=1}^{f-1} a_i^{L-1}; \quad a_{i+1}^L = a_i^L - a_{f-i}^{L-1}, \quad i = 1, 2, \dots, f-2$$

# Анализ

- Н. Вирт пишет: "Хотя такая схема выглядит хуже многофазного слияния, поскольку некоторые последовательности "простаивают", и есть просто операции копирования, тем не менее, она, что удивительно, оказывается **не хуже многофазной сортировки при очень больших файлах и шести или более последовательностях**". Заметим, что при хорошей реализации алгоритма каскадной сортировки, от копирования можно избавиться, используя карты индексов. Более подробное объяснение алгоритмов, а также их реализацию можно увидеть в книгах Н. Вирта и Д. Кнута.

# **Улучшенные методы внешних сортировок слиянием**

# Рекурсивный алгоритм сортировки слиянием

- Принципиальную возможность эффективно сортировать файл, работая с его частями и не выходя за его пределы, обеспечивает алгоритм Боуза и Нельсона, который применялся лишь к внутренней сортировке.
- Он основан на очевидной идее:
  - слить две равные упорядоченные части можно слиянием их начальных половин,
  - слиянием конечных половин
  - слиянием 2-й половины 1-го результата с 1-й половиной 2-го результата,

# пример

слияние начал и слияние концов

$$1 \quad 7 \quad 8 \quad 11 \quad + \quad 4 \quad 5 \quad 6 \quad 9 =$$

слияние в средней части

$$\boxed{1 \quad 4} \quad 5 \quad 7 \quad + \quad 6 \quad 8 \quad \boxed{9 \quad 11} =$$

$$\boxed{1 \quad 4} \quad \boxed{5 \quad 6 \quad 7 \quad 8} \quad \boxed{9 \quad 11}$$

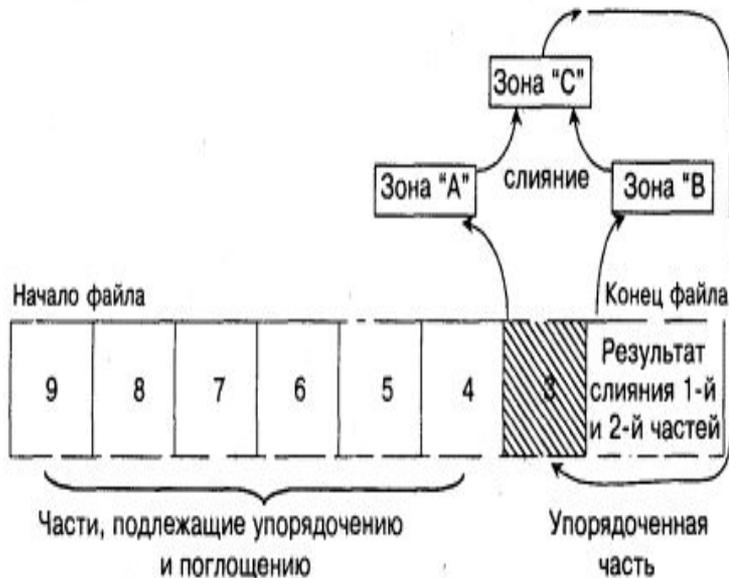
- Эта процедура отвечает стремлению вести основную работу в памяти.
- Как только в ходе рекурсии получаем части, вписывающиеся в память, выполняем их слияние в памяти, но уже другим, быстрым алгоритмом.
- Общий эффект слияний всех частей, образующихся в рекурсии, тождественен эффекту слияния исходных частей, сколь бы велики они ни были.

# Оценка сложности

- Пусть исходные упорядоченные нами части файла имеют размер 64 К, их число  $M$ , а для слияния частей используются в качестве буферов 3 сегмента памяти размером 64 К. Время 1-го этапа пропорционально  $M$  и в первом приближении асимптотическая сложность  $O(M * 1.5^{\log M})$
- При больших значениях  $Rf$  (размер файла) и  $M$  ( $Rf = 2 \text{ Мб} \dots 40 \text{ Мб}$ ,  $M = 40 \dots 300$  частей) эта зависимость практически подтверждается.
- В данном анализе не учтены ходы штанги.

# Сортировка методом поглощения

Имея несколько частей файла и начав со слияния двух из них, будем сливать все следующие с большей (растущей) упорядоченной частью. Она как бы поглощает часть за частью



Перед поглощением очередная часть файла считывается в зону "А" ОП, там упорядочивается и остается. Начало ранее упорядоченной части считывается в зону "В" и начинается слияние, прерываемое считыванием в зону "В", когда она опустошается. По мере заполнения зоны "С" записями акта слияния, содержимое ее переписывается в файл (на место поглощаемой части и далее в сторону конца файла).

# Анализ

Если при слиянии взяты все записи поглощаемой части, поглощение завершается передачей в файл из зоны "С" остатка результата.

Слияние также завершается, если исчерпана ранее упорядоченная часть. Поглощение ею очередной части произошло.

- Ходов в данном методе будет мало. Это дает экономию времени и при небольшом размере файла сортировка проходит быстро.

# Челночное балансное слияние

- На 1 этапе внутренней сортировки частей в файле создают  $M$  упорядоченных частей, по возможности большего размера  $R$ . К ним применяют прямое слияние.
- Создают резервное дисковое пространство файла и располагают его непосредственно до или после сливаемых частей и перемещается к следующим частям по завершении слияния. Его размер не меньше размера  $R$  части.
- Резерв и пространство ближайшей части будут заполнено результатом слияния двух первых частей, при этом пространство 2-й части освободится и станет резервом, необходимым для слияния следующей пары частей и т. д.

# Схема челночного балансного слияния

а) слияние частей размером  $R$  (промежуточная ситуация)



б) следующий "прогон" челнока слияние частей размером  $2R$



# Пути повышения эффективности внешних сортировок

- Применение методов внешней сортировки в зависимости от поставленной задачи сортировки
- Многопроцессорная обработка данных
- Использование буферной памяти
- Использование виртуальной памяти