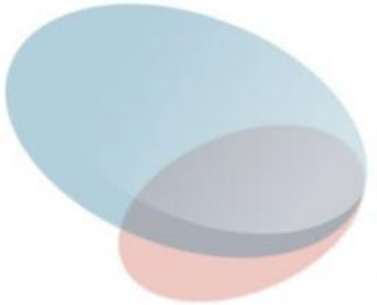


Алгоритмы внешней сортировки
Лекция 4

Алгоритмы внешней сортировки

Внешняя сортировка – это упорядочивание данных, которые хранятся на внешнем устройстве с медленным доступом (диск и т.п.) и не вмещаются в оперативную память. Используется, когда применить одну из внутренних сортировок невозможно.

При внешней сортировке прежде всего требуется **уменьшить число обращений к этому устройству**, т. е. число проходов через файл.

Внутренняя сортировка значительно эффективней внешней, так как на обращение к оперативной памяти затрачивается намного меньше времени, чем к магнитным дискам и т. п.

Наиболее часто внешняя сортировка используется в СУБД.

Алгоритмы внешней сортировки

Для выяснения эффективности алгоритмов внутренней сортировки подсчитывалось число выполняемых ими сравнений.

Объем работы по чтению или записи на диск блоков виртуальной памяти может значительно превышать трудоемкость логических и арифметических операций. Эта работа выполняется операционной системой, и у нас нет реальных средств воздействия на ее эффективность.

При другом подходе **можно воспользоваться файлами с прямым доступом** и заменить непосредственные обращения к массиву операциями поиска в файле для выхода в нужную позицию с последующим чтением блока. В результате размер используемой логической памяти уменьшается, а значит уменьшается неконтролируемая нагрузка на виртуальную память.

Объем операций ввода-вывода все равно остается значительным – управляем ли мы им сами или полагаемся на операционную систему.

Алгоритмы внешней сортировки

Основным понятием при использовании внешней сортировки является понятие **отрезка (серии)**.

Отрезком длины K является последовательность записей

$A_i, A_{i+1}, \ldots, A_{i+k-1}$ такая, что в ней все записи упорядочены по некоторому ключу.

Максимальное количество отрезков в файле равна N (все элементы не упорядочены).

Минимальное количество отрезков 1 (все элементы являются упорядоченными).

Алгоритмы внешней сортировки

Примеры серий (отрезков)

1) Пусть в некотором файле A хранится одномерный массив:

12 35 65 0 24 26 3 5 84 90 6 2 30

Поделим массив на отрезки:

12 35 65 | 0 24 36 | 3 5 84 90 | 6 | 2 30

Можно сказать, что массив в файле A состоит из 5 отрезков.

2) В файле B хранится одномерный массив:

1 2 3 4 5 6 7 8 9 10

Поделим массив на отрезки:

| 1 2 3 4 5 6 7 8 9 10 |

Можно сказать, что массив в файле B состоит из 1 отрезка.

3) В файле C хранится одномерный массив:

20 17 16 14 13 10 9 8 6 4 3 2 0

Поделим массив на отрезки:

| 20 | 17 | 16 | 14 | 13 | 10 | 9 | 8 | 6 | 4 | 3 | 2 | 0 |

Можно сказать, что массив в файле C состоит из 13 отрезков.

Алгоритмы внешней сортировки

1. Естественная сортировка (метод естественного слияния)

Несбалансированная двухфазная трехленточная сортировка слиянием

Пример

1) Пусть нужно отсортировать файл:

C: 20 50 7 30 80 40 60 50 35 25 70

Первый проход:

Фаза распределения:

A: 20 50 40 60 35

B: 7 30 80 50 25 70

Фаза слияния :

C: 7 20 30 50 80 40 50 60 25 35 40

Второй проход:

Фаза распределения:

A: 7 20 30 50 80 25 35 70

B: 40 50 60

Фаза слияния:

C: 7 20 30 40 50 50 60 80 25 35 70

Третий проход:

Фаза распределения:

A: 7 20 30 40 50 50 60 80 B: 25 35 70

Фаза слияния:

C: 7 20 25 30 35 40 50 50 60 70 80

Алгоритмы внешней сортировки

Несбалансированная двухфазная трехленточная сортировка слиянием

Пример

2) Этот пример показывает, что на фазе слияния несколько серий могут восприниматься как одна серия.

Пусть нужно отсортировать файл:

C: 1 2 1 30 20 40 2 10 15 2 10 20 40 30 50

Первый проход:

Фаза распределения:

A: 1 2 20 40 2 10 20 40

B: 1 30 2 10 15 30 50

Фаза слияния :

C: 1 1 2 20 30 40 2 2 10 10 15 20 30 40 50

Второй проход:

Фаза распределения:

A: 1 1 2 20 30 40 B: 2 2 10 10 15 20 30 40 50

Фаза слияния:

C: 1 1 2 2 2 10 10 15 20 20 30 30 40 40 50

Алгоритмы внешней сортировки

Как увеличить скорость сортировки

Идея большинства методов заключается в расчленении данных на ряд последовательностей, помещающихся в оперативную память.

Далее **применяется один из методов внутренней сортировки**, после чего **последовательности сливаются**. Чем больше объём оперативной памяти, тем длиннее могут быть последовательности и, следовательно, тем меньшим окажется их количество, что увеличит скорость сортировки.

Если объём оперативной памяти мал, то можно разделить исходные данные на несколько последовательностей, после чего непосредственно использовать процедуру слияния.

Основные методы внешних сортировок:

1. **Естественная сортировка** (метод естественного слияния)
2. **Сортировка методом двухпутевого сбалансированного слияния**
3. **Сортировка методом n-путевого слияния**.
4. **Многофазная сортировка** (Фибоначчиевая).
5. **Каскадное слияние**.

Внешняя сортировка слиянием

2а. Сортировка методом двухпутевого сбалансированного слияния без использования оперативной памяти

Вся сортируемая последовательность данных разбивается на два файла f1 и f2. Желательно, чтобы количество записей в этих файлах было поровну. Как и в алгоритме внутренней сортировки, **считаем, что любой файл состоит из участков длиной 1**.

Затем можно **объединить участки длины 1 и распределить их по файлам g1 и g2 в виде участков длины 2**.

После этого **делаем f1 и f2 пустыми и объединяем g1 и g2 в f1 и f2**, **которые затем можно организовать в виде участков длины 4** и т. д.

После выполнения **i проходов** получатся два файла, состоящие из участков длины 2^i. **Если 2^i ≥ n, то один из этих двух файлов будет пустым, а другой будет содержать единственный участок длиной n, т. е. будет отсортирован.** Так как 2^i ≥ n при i ≥ log n, то в этом случае **будет достаточно порядка O(log n) проходов по данным**.

При такой сортировке не требуется, чтобы отдельный участок полностью находился в оперативной памяти (при большой длине он может не поместиться в буфер). Участок считывается и записывается **последовательно** запись за записью. Именно **такой подход заставляет использовать два входных файла**. В противном случае можно было бы читать по два участка из одного файла одновременно.

Внешняя сортировка слиянием

Пример. Сортировка методом двухпутевого сбалансированного слияния без использования оперативной памяти

Исходные файлы

f1	28	3	93	10	54	65	30	90
f2	31	5	96	40	85	9	39	

Участки длиной 2

g1	28	31	93	96	54	85	30	39
g2	3	5	10	40	9	65	90	

Участки длиной 4

f1	3	5	28	31	9	54	65	85
f2	10	40	93	96	30	39	90	

Участки длиной 8

g1	3	5	10	28	31	40	93	96
g2	9	30	39	54	65	85	90	

Участки длиной 16

f1	3	5	9	10	28	30	31	39	40	54	65	85	90	93	96	
f2																

Внешняя сортировка слиянием

2б. Сбалансированная внешняя сортировка слиянием с использованием оперативной памяти

Пусть у нас **есть четыре файла** и инструмент их слияния.

Оценим сначала разумное число записей, которые можно хранить в оперативной памяти одновременно. Объявим массив, длина S которого равна этой величине; этот массив будет использоваться на двух этапах сортировки.

I этап сортировки – распределение

На первом шаге **прочитаем S записей** из входного файла **и отсортируем их с помощью подходящей внутренней сортировки**. Этот **набор уже отсортированных записей перепишем в файл A**.

Затем **прочитаем следующие S записей, отсортируем их и перепишем в файл B**.

Этот **процесс продолжается, причем отсортированные блоки записей пишутся попеременно то в файл A, то в файл B**, до тех пор, пока входной файл не будет исчерпан.

Внешняя сортировка слиянием

Алгоритм первого этапа – распределение

```
//Createfiles(S);
begin
//S     размер создаваемых отрезков
CurrentFile := A;
while конец входного файла не достигнут do
begin
      //read S записей из входного файла
      //sort S записей
      if CurrentFile := A then
            CurrentFile := B
      else
            CurrentFile := A;
      //записываем S записей в CurrentFile
end
end;
```

Внешняя сортировка слиянием

После того, как входной файл полностью разбит на два файла, содержащих отсортированные отрезки, мы готовы перейти ко второму шагу – слиянию этих отрезков.

Каждый из файлов A и B содержит некоторую последовательность отсортированных отрезков, однако, как и в случае сортировки слиянием, мы ничего не можем сказать о порядке записей в двух различных отрезках.

Внешняя сортировка слиянием

II этап сортировки – слияние отсортированных отрезков

- Начинаем с **чтения половинок первых отрезков из файлов A и B**. Читаем лишь по половине отрезков, поскольку в памяти может находиться одновременно лишь S записей, а нам нужны записи из обоих файлов. **Будем сливать эти половинки отрезков в один отрезок файла C**.
- **После того, как одна из половинок закончится, прочтем вторую половинку из того же файла**.
- **Когда обработка одного из отрезков будет завершена, конец второго отрезка будет переписан в файл C**.
- После того, как слияние первых двух отрезков из файлов A и B будет завершено, **следующие два отрезка сливаются в файл D**.
- Этот **процесс слияния отрезков продолжается с попеременной записью слитых отрезков в файлы C и D**.
- По завершении **получаем два файла, разбитых на отсортированные отрезки длины 2S**.

Далее описанный **процесс повторяется, при этом отрезки длины S/2 читаются из файлов C и D, а слитые отрезки длины 4S записываются в файлы A и B**.

В конце концов отрезки сольются в один отсортированный список в одном из файлов.

Внешняя сортировка слиянием

Алгоритм второго этапа - слияние

```
//PolyPhaseMerge(S);
begin
//S          размер исходных отрезков
Size:= S;
Input1 := A;
Input2 := B;
CurrentOutput := C;
while not done do begin
      while отрезки не кончились do
      begin
          //слить отрезок длины Size из файла Input1
          //с отрезком длины Size из файла Input2,
          //записав результат в CurrentOutput
        if (CurrentOutput = A) then CurrentOutput = B
           else if (CurrentOutput = B) then CurrentOutput = A
              else if (CurrentOutput = C) then CurrentOutput = D
       else if (CurrentOutput = D) then CurrentOutput = C;
       end;
```

Внешняя сортировка слиянием

Алгоритм второго этапа – слияние

```
    Size := Size*2;
  if (Input1 = A) then begin
      Input1 := C
      Input2 := D
      CurrentOutput := A
  end
  else begin
      Input1 = A
      Input2 = B
      CurrentOutput = C
  end
end ;
```

Внешняя сортировка слиянием

Анализ сортировки

Проанализируем, с каким количеством отрезков мы имеем дело, и как это количество влияет на число проходов.

Если в исходном файле N записей, и в память помещается одновременно S записей, то **после I этапа** – распределения, то есть после выполнения процедуры Createfiles, **получаем R = [N / S] отрезков, распределенных по двум файлам**.

При каждом проходе алгоритма PolyPhaseMerge **на II этапе – слиянии** – пары отрезков сливаются, поэтому **число отрезков уменьшается вдвое**. После первого прохода будет [R / 2] отрезков, после второго – [R / 4] и, **в общем случае, после j-го прохода будет [R / (2^j)] отрезков**.

Алгоритм завершается, если остается один отрезок, то есть когда [R / (2^D)] = 1, или после

D = [log R] = [log (N/S)] проходов процесса слияния.

Внешняя сортировка слиянием

3. Сортировка методом многопутевого слияния

При использовании метода многопутевой внешней сортировки на каждом шаге **примерно половина вспомогательных файлов используется для ввода данных и примерно столько же для вывода сливаемых серий**.

На шаге **распределения** возрастающие серии (отрезки) исходного файла распределяются по **m** вспомогательным файлам, а затем выполняется **многопутевое слияние** серий из **m** файлов. Способы слияния:

1 способ. 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 потребуется просмотреть только **m** ключей и выбрать из них наименьший.

2 способ. Если **m** велико, можно ускорить работу, строя дерево выбора (пирамиду) из **m** записей. Затем потребуется примерно **lg m** сравнений для выбора минимального ключа.

Внешняя сортировка слиянием

Пример. Сортировка методом 4-х путевого слияния (стадия слияния 4-х возрастающих серий)

Шаг 1

$$087 \left\{ \begin{array}{l} 087 \left\{ \begin{array}{l} 087\ 503\ \infty \\ 170\ 908\ \infty \end{array} \right. \\ 154 \left\{ \begin{array}{l} 154\ 426\ 653\ \infty \\ 612\ \infty \end{array} \right. \end{array} \right.$$

Шаг 2

$$087\ 154 \left\{ \begin{array}{l} 170 \left\{ \begin{array}{l} 503\ \infty \\ 170\ 908\ \infty \end{array} \right. \\ 154 \left\{ \begin{array}{l} 154\ 426\ 653\ \infty \\ 612\ \infty \end{array} \right. \end{array} \right.$$

Шаг 3

$$087\ 154\ 170 \left\{ \begin{array}{l} 170 \left\{ \begin{array}{l} 503\ \infty \\ 170\ 908\ \infty \end{array} \right. \\ 426 \left\{ \begin{array}{l} 426\ 653\ \infty \\ 612\ \infty \end{array} \right. \end{array} \right.$$

$\vdots$

Шаг 9

$$087\ 154\ 170\ 426\ 503\ 612\ 653\ 908\ \infty \left\{ \begin{array}{l} \infty \left\{ \begin{array}{l} \infty \\ \infty \end{array} \right. \\ \infty \left\{ \begin{array}{l} \infty \\ \infty \end{array} \right. \end{array} \right.$$

Внешняя многофазная сортировка слиянием

4. Многофазная сортировка (Фибоначчиевая)

Идея многофазной сортировки состоит в том, что **из имеющихся m вспомогательных файлов (m-1) файл служит для ввода сливаемых последовательностей, а один – для вывода образуемых серий**.

Как только один из файлов ввода становится пустым, его начинают использовать для вывода серий, получаемых при слиянии серий нового набора (m-1) файлов.

Первый шаг. Серии исходного файла **распределяются** по **m-1** вспомогательному файлу.

Второй шаг. Выполняется **многопутевое (m-1- путевое) слияние** серий из **(m-1)** файла, пока в одном из них не образуется одна серия.

На каждом шаге берем наименьший из начальных элементов входных серий и перемещаем в конец выходной серии.

При произвольном начальном распределении серий по вспомогательным файлам **алгоритм может не сойтись, поскольку в единственном непустом файле может существовать более, чем одна серия**.

Внешняя многофазная сортировка слиянием

Пример начального распределения серий, при котором трехфазная внешняя сортировка не приводит к нужному результату (алгоритм не сходится)

Пусть используются три файла B1, B2 и B3.

При начальном распределении в файл B1 помещены 10 серий, а в файл B2 - 6. При слиянии B1 и B2 к моменту, когда мы дойдем до конца B2, в B1 останутся 4 серии, а в B3 попадут 6 серий.

Продолжится слияние B1 и B3, и при завершении просмотра B1 в B2 будут содержаться 4 серии, а в B3 останутся 2 серии.

После слияния B2 и B3 в каждом из файлов B1 и B2 будет содержаться по 2 серии, которые будут слиты и образуют 2 серии в B3 при том, что B1 и B2 - пусты. Тем самым, алгоритм не сошелся.

Число серий в B1	Число серий в B2	Число серий в B3
10	6	0
4	0	6
0	4	2
2	2	0
0	0	2

Внешняя многофазная сортировка слиянием

Вопрос: каким должно быть начальное распределение серий, чтобы алгоритм трехфазной сортировки благополучно завершал работу и выполнялся максимально эффективно?

Рассмотрим работу алгоритма в обратном порядке, **начиная от желательного конечного состояния вспомогательных файлов**. Нас устраивает любая комбинация конечного числа серий в файлах B1, B2 и B3 из (**1,0,0**), (0,1,0) и (0,0,1). Для определенности выберем первую комбинацию. Для того, чтобы она сложилась, необходимо, чтобы на непосредственно предыдущем этапе слияний существовало распределение серий (**0,1,1**).

Чтобы получить такое распределение, необходимо, чтобы на непосредственно предыдущем этапе слияний распределение выглядело как (**1,0,2**) или (1,2,0).

Опять для определенности остановимся на первом варианте. Чтобы его получить, на предыдущем этапе годились бы следующие распределения: (**3,2,0**) и (0,3,1). Но второй вариант хуже, поскольку он приводит к слиянию только одной серии из файлов B2 и B3, в то время как при наличии первого варианта распределения будут слиты две серии из файлов B1 и B3.

Пожеланием к предыдущему этапу было бы наличие распределения (**0,5,3**), еще раньше - (**5,0,8**), еще раньше - (**13,8,0**) и т.д.

Внешняя многофазная сортировка слиянием

Метод трехфазной внешней сортировки дает желаемый результат и работает максимально эффективно (на каждом этапе сливается максимальное число серий), **если начальное распределение серий** между вспомогательными файлами описывается **соседними числами Фибоначчи**:

(**1,0**,0), (0,**1,1**), (**1**,0,**2**), (**3,2**,0), (0,**5,3**), (**5**,0,**8**), (**13,8**,0) и т.д.

Пример. При начальном распределении серий между тремя файлами 13, 8,0 метод сойдется:

B1	B2	B3
(**13,**	**8**,	0)
(**5,**	0,	**8**)
(0,	**5,**	**3**)
(**3,**	**2**,	0)
(**1,**	0,	**2**)
(0,	**1,**	**1**)
(**1,**	**0**,	0).

Последовательность чисел Фибоначчи начинается с 0, 1, а каждое следующее число образуется как сумма двух предыдущих: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...

Внешняя многофазная сортировка слиянием

Многофазная сортировка (Фибоначчиевая)

В общем случае **при использовании m вспомогательных файлов** аналогичным **условием успешного завершения и эффективной работы метода многофазной внешней сортировки** является то, чтобы **начальное распределение серий между (m-1) файлами** описывалось ***суммами*** **соседних (m-2), ..., 0 чисел Фибоначчи порядка m-1**.

Последовательность чисел Фибоначчи p-го порядка начинается с **(p-1)** нулей, затем идет **1**, и каждое следующее число является суммой **p** предыдущих чисел. При p=2 это обычная последовательность Фибоначчи.

Числа Фибоначчи **p-го** порядка определяются правилами:

$$F_n^{(p)} = F_{n-1}^{(p)} + F_{n-2}^{(p)} + \cdots + F_{n-p}^{(p)} \quad \text{при } n \geq p;$$

$$F_n^{(p)} = 0 \quad \text{при } 0 \leq n \leq p-2; \qquad F_{p-1}^{(p)} = 1.$$

Внешняя многофазная сортировка слиянием

Пример. Ниже показано начало последовательности чисел Фибоначчи порядка p=5:

0, 0, 0, 0, 1, 1, 2, 4, 8, 16, 31, 61, ...

Поскольку число серий в исходном файле может не обеспечивать возможность такого распределения серий, **применяется метод добавления пустых серий,** которые **в дальнейшем как можно более равномерно распределяются между промежуточными файлами и опознаются при последующих слияниях.**

Чем меньше таких пустых серий, т.е. чем ближе число начальных серий к требованиям Фибоначчи, тем более эффективно работает алгоритм.

Внешняя многофазная сортировка слиянием

Пример. Многофазное (6-и фазное) слияние

Далее 1^31 обозначает 31 серию относительной длины 1 и т.д.; везде изпользуется **пятипутевое слияние**.

Чтобы заставить механизм многофазного слияния работать, необходимо после каждой фазы иметь “точное фибоначчиево распределение” серий по файлам. Точные фибоначчиевы распределения можно получить, прокручивая приведенную схему в обратном направлении и циклически переставляя содержимое файлов.

Фаза	T1	T2	T3	T4	T5	T6	Число обрабатываемых начальных серий
1	1^{31}	1^{30}	1^{28}	1^{24}	1^{16}	—	$31 + 30 + 28 + 24 + 16 = 129$
2	1^{15}	1^{14}	1^{12}	1^{8}	—	5^{16}	$16 \times 5 = 80$
3	1^{7}	1^{6}	1^{4}	—	9^{8}	5^{8}	$8 \times 9 = 72$
4	1^{3}	1^{2}	—	17^{4}	9^{4}	5^{4}	$4 \times 17 = 68$
5	1^{1}	—	33^{2}	17^{2}	9^{2}	5^{2}	$2 \times 33 = 66$
6	—	65^{1}	33^{1}	17^{1}	9^{1}	5^{1}	$1 \times 65 = 65$
7	129^{1}	—	—	—	—	—	$1 \times 129 = 129$

Внешняя многофазная сортировка слиянием

Начало последовательности чисел Фибоначчи порядка p=5
0, 0, 0, 0, 1, 1, 2, 4, 8, 16, 31, 61, ... представляют числа a_n.
Читая приведенную выше таблицу снизу вверх, можно заметить, что первые семь точных фибоначчиевых распределений при количестве файлов =6 суть (1,0,0,0,0), (1,1,1,1,1), (2,2,2,2,1), (4,4,4,3,2), (8,8,7,6,4), (16,15,14,12,8) и (31,30,28,24,16) (p=5). Это числа a_n, b_n, c_n, d_n, e_n.

Уровень	T1	T2	T3	T4	T5	Сумма	Окончательный результат будет на
0	1	0	0	0	0	1	T1
1	1	1	1	1	1	5	T6
2	2	2	2	2	1	9	T5
3	4	4	4	3	2	17	T4
4	8	8	7	6	4	33	T3
5	16	15	14	12	8	65	T2
6	31	30	28	24	16	129	T1
7	61	59	55	47	31	253	T6
8	120	116	108	92	61	497	T5
. . . . .							
n	a_n	b_n	c_n	d_n	e_n	t_n	T(k)
$n+1$	$a_n + b_n$	$a_n + c_n$	$a_n + d_n$	$a_n + e_n$	a_n	$t_n + 4a_n$	T($k-1$)
. . . . .							

Внешняя многофазная сортировка слиянием

Пример

Из правила перехода от n-го уровня к n+1 следуют неравенства:

an>=bn>=cn>=dn>=en для любого уровня.

$$e_n = a_{n-1},$$

$$d_n = a_{n-1} + e_{n-1} = a_{n-1} + a_{n-2},$$

$$c_n = a_{n-1} + d_{n-1} = a_{n-1} + a_{n-2} + a_{n-3},$$

$$b_n = a_{n-1} + c_{n-1} = a_{n-1} + a_{n-2} + a_{n-3} + a_{n-4},$$

$$a_n = a_{n-1} + b_{n-1} = a_{n-1} + a_{n-2} + a_{n-3} + a_{n-4} + a_{n-5},$$

где $a_0 = 1$ и где мы полагаем, что $a_n = 0$ при $n = -1, -2, -3, -4$.

Приведенные соотношения показывают, что число серий в файле T1 в процессе шестиленточного многофазного слияния является числом Фибоначчи пятого порядка: $a_n = F^{(5)}_n$, n=-4,-3,-2,-1,0,1,2,3,…

Внешняя сортировка каскадным слиянием

5. Каскадное слияние

Каскадное слияние начинается с точного распределения серий по файлам, хотя правила точного распределения отличны от правил для многофазной сортировки.

Пусть используются 6 файлов.

Проход 1. Серии распределяются по первым пяти файлам, пусть файл f6 – пустой.

Проход 2. Получается посредством выполнения

пятипутевого слияния из файлов f1, f2, f3, f4, f5 в файл f6, пока один из файлов, например, f5 не станет пустым;

затем – четырехпутевого слияния из f1, f2, f3, f4 в f5, f4 – пустой;

затем – трехпутевого слияния из f1, f2, f3 в f4, f3 – пустой;

двухпутевого слияния из f1, f2 в f3, f2 – пустой; и, наконец,

однопутевого слияния (операция копирования) из f1в f2, f1 – пустой.

Проход 3. Получается таким же образом, путем выполнения сначала пятипутевого слияния, например, из файлов f2, f3, f4, f5, f6 в f1, пока один файл не станет пустым, затем четырехпутевого, трехпутевого, двухпутевого и однопутевого.

Для шести и более файлов каскадная сортировка лучше, чем многофазная.

Внешняя сортировка каскадным слиянием

Пример. Каскадное слияние

Каждая строка таблицы представляет полный проход **по всем** данным.

Проход 2, например, получается посредством выполнения пятипутевого слияния с {T1, T2, T3, T4, T5} на T6, пока T5 не станет пустым (при этом на T6 помещаются 15 серий относительной длиной 5), затем четырехпутевого слияния с {T1, T2, T3, T4} на T5, затем трехпутевого слияния на T4, двухпутевого слияния на T3 и, наконец, однопутевого слияния (операции копирования) с T1 на T2.

Проход 3 получается таким же образом, путем выполнения сначала пятипутевого слияния, пока один файл не станет пустым, затем – четырехпутевого и т.д.

	T1	T2	T3	T4	T5	T6	Обработанные начальные серии
Проход 1	1^{55}	1^{50}	1^{41}	1^{29}	1^{15}	—	190
Проход 2	—	$*1^{5}$	2^{9}	3^{12}	4^{14}	5^{15}	190
Проход 3	15^{5}	14^{4}	12^{3}	9^{2}	$*5^{1}$	—	190
Проход 4	—	$*15^{1}$	29^{1}	41^{1}	50^{1}	55^{1}	190
Проход 5	190^{1}	—	—	—	—	—	190

Внешняя сортировка каскадным слиянием

Каскадное слияние

Рассуждая в обратном направлении от конечного состояния

(1, 0,…,0), Д. Кнут вывел точное распределение для каскадного слияния. В случае шести файлов (лент) оно следующее (пустые файлы в распределении не присутствуют).

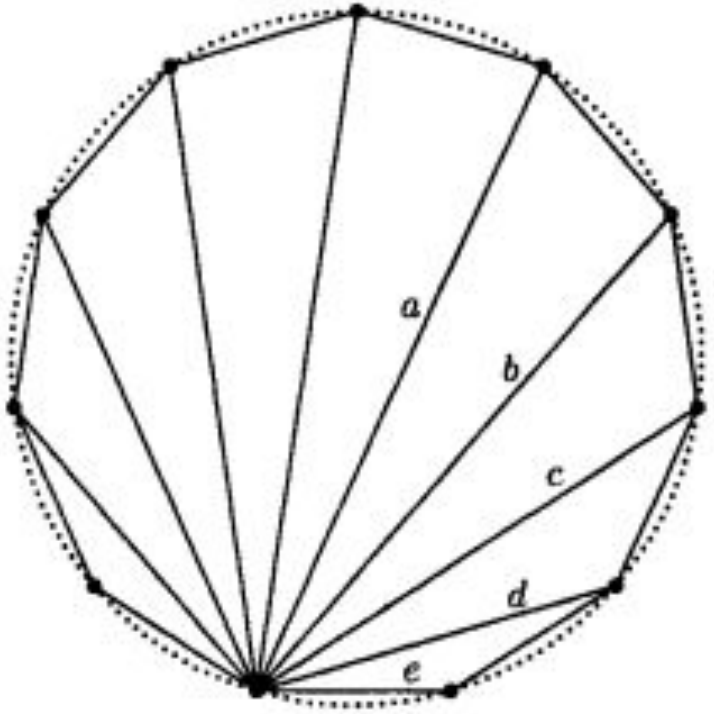
Распределение серий в обратном порядке:

Уровень	T1	T2	T3	T4	T5
0	1	0	0	0	0
1	1	1	1	1	1
2	5	4	3	2	1
3	15	14	12	9	5
4	55	50	41	29	15
5	190	175	146	105	55
. . . . .					
n	a_n	b_n	c_n	d_n	e_n
$n+1$	$a_n+b_n+c_n+d_n+e_n$	$a_n+b_n+c_n+d_n$	$a_n+b_n+c_n$	a_n+b_n	a_n

Внешняя сортировка каскадным слиянием

Числа a_n, b_n, c_n, d_n, e_n имеют интересное свойство – их относительные величины являются также и длинами диагоналей (2T-1)-угольника (T – число используемых файлов). Например, пять диагоналей одиннадцатиугольника имеют относительные длины, очень близкие к 190, 175, 146, 105 и 55.

В книге Кнута приведено доказательство того факта, что **значения относительного времени, затрачиваемого на (T-1), (T-2), …,1-путевое слияние, приблизительно пропорциональны квадратам длин этих диагоналей**.



Геометрическая интерпретация каскадных чисел.

Внешняя сортировка каскадным слиянием

Каскадное слияние

Для 6 и более файлов каскадная схема лучше, чем многофазная.

Каскадная сортировка впервые была исследована У.К.Картером (1962 г.), который получил численные результаты для небольших T, и Дэвидом Фергюсоном (1964 г.).