Сортировки.
Внешние сортировки.

Понятие

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.

Понятие

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.

Эффективность алгоритма

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.

Эффективность алгоритма

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

Понятие серии

Серией длины K является последовательность записей A_i, A_{i+1},…,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).

Деление на серии

Пусть в некотором файле A хранится одномерный массив:
12 35 65 0 24 26 3 5 84 90 6 2 30

Поделим массив на серии:
12 35 65 | 0 24 36 | 3 5 84 90 | 6 | 2 30

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

Алгоритмы внешней 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.

Алгоритмы внешней сортировки

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.

Естественная сортировка

Исходный файл:

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

Делим на серии:

F: 20 50 | 7 30 80 | 40 60 | 50 | 35 | 25 70

Естественная сортировка

Первый проход:
Фаза распределения:

F1: 20 50 | 40 60 | 35

F2: 7 30 80 | 50 | 25 70

Фаза слияния:

F: 7 20 30 50 80 | 40 50 60 | 25 35 70

Естественная сортировка

Второй проход:

Фаза распределения:

F1: 7 20 30 50 80 | 25 35 70

F2: 40 50 60

Фаза слияния:

F: 7 20 30 40 50 50 60 80 | 25 35 70

Естественная сортировка

Третий проход:

Фаза распределения:

F1: 7 20 30 40 50 50 60 80

F2: 25 35 70

Фаза слияния:

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

Двухпутевое слияние

1. Сортировка методом двухпутевого сбалансированного слияния без использования оперативной памяти
2. Сбалансированная внешняя сортировка слиянием с использованием оперативной памяти

Двухпутевое слияние без использования оперативной памяти

1. Вся сортируемая последовательность данных разбивается на два файла f1 и f2. Желательно, чтобы количество записей в этих файлах было поровну.
2. Как и в алгоритме внутренней сортировки, считаем, что любой файл состоит из участков длиной 1.
3. Затем можно объединить участки длины 1 и распределить их по файлам g1 и g2 в виде участков длины 2.
4. 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.

Двухпутевое слияние без использования оперативной памяти

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

Пример

Исходные файлы:

f1: 28 3 93 10 54 65 30 90

f2: 31 5 96 40 85 9 39

Участки длинной 2:

g1: 28 31 | 93 96 | 54 85 | 30 39

g2: 3 5 | 10 40 | 9 65 | 90

Пример

Участки длинной 4:

f1: 3 5 28 31 | 9 54 65 85

f2: 10 40 93 96 | 30 39 90

Участки длинной 8:

g1: 3 5 10 28 31 40 93 96

g2: 9 30 39 54 65 85 90

Участки длинной 16:

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

Двухпутевое слияние 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.

Двухпутевое слияние с использованием оперативной памяти

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.

Двухпутевое слияние с использованием оперативной памяти

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.

Двухпутевое слияние с использованием оперативной памяти

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.

Анализ сортировки

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

При каждом проходе 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.

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 исходного файла распределяются по m вспомогательным файлам, а затем выполняется многопутевое слияние серий из m файлов.

Способы слияния

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

Пример (N=3)

Исходный файл:

F: 21 72 19 45 44 40 8 57 77 63 18 39 24 34 16

Первый проход:

F1: 21 | 45 | 8 | 63 | 24

F2: 72 | 44 | 57 | 18 | 34

F3: 19 | 40 | 77 | 39 | 16

Пример (N=3)

Второй проход:

F4: 19 21 72| 18 39 63

F5: 40 44 45 | 16 24 34

F6: 8 57 77

Третий проход:

F1: 8 19 21 40 44 45 57 72 77

F2: 16 18 24 34 39 63

F3:

Четвертый проход:

F4: 8 16 18 19 21 24 34 39 40 44 45 57 63 72 77

F5:

F6:

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.

Многофазная сортировка (Фибоначчиевая)

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.

Пример

Пусть используются три файла B1, B2 и B3.

B1	**B2**	**B3**
10	6	0
4	0	6
0	4	2
2	2	0
0	0	2

Многофазная сортировка (Фибоначчиевая)

Вопрос: Каким должно быть р

B2	B2	B3
1	0	0
0	1	1
1	0	2
3	2	0
0	5	3
5	0	8

Многофазная сортировка (Фибоначчиевая)

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.

Многофазная сортировка (Фибоначчиевая)

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, … .

$$F(n) = F(n-1) + F(n-2),\ n>1$$

$$F(0) = 0,\ F(1) = 1.$$

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.

Многофазная сортировка (Фибоначчиевая)

Числа Фибоначчи p-го порядка определяются правилами:

$$F(n) = F(n-1)+F(n-2)+ \ldots +F(n-p) \text{ при } n \geq p$$

$$F(n) = 0,\ 0 \leq n \leq p-2,\ F(p-1) = 1.$$

Фибоначчи порядка p=5:

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

Многофазная сортировка (Фибоначчиевая)

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.

Пример (p=2)

Исходный файл:
F: 31 72 19 45 44 40 8 57 77 63 13

Количество элементов **n = 11**.
Ближайшее число Фибоначчи **n ≤ f = 13 = 5 + 8**

Значит будем разбивать на 2 файла с количеством серий 8 и 5 по 1 элементу
Так как у нас только 11 элементов, то необходимо еще добавить пустых 2 серии

Пример (p=2)

Начальное распределение:

F1: ø| ø | 31 | 72 | 19 | 45 | 44 | 40

F2: 8 | 57 | 77 | 63 | 13

F3:

Первый проход:

F1: 45 | 44 | 40

F2:

F3: 8 | 57 | 31 77 | 63 72 | 13 19

Пример (p=2)

Второй проход:

F1:

F2: 8 45 | 44 57 | 31 40 77

F3: 63 72 | 13 19

Третий проход:

F1: 8 45 63 72 | 13 19 44 57

F2: 31 40 77

F3:

Пример (p=2)

Четвертый проход:

F1: 13 19 44 57

F2:

F3: 8 31 40 45 63 72 77

Пятый проход:

F1:

F2: 8 13 19 31 40 44 45 57 63 72 77

F3:

Каскадное слияние

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

Рассуждая в обратном направлении от конечного состояния (1, 0,…,0), так же для и многофазной сортировки, Д. Кнут вывел точное распределение для каскадного слияния.

Оптимальное количество уровней для N=4

Уровень	F1	F2	F3	F4
0	1	0	0	0
1	1	1	1	1
2	4	3	2	1
3	10	9	7	4
4	30	26	19	10
5	85	75	56	30

Уровень	F1	F2	F3	F4
n	a_n	b_n	c_n	d_n
n+1	$a_n+b_n+c_n+d_n$	$a_n+b_n+c_n$	a_n+b_n	a_n

Каскадное слияние

Числа a_n, b_n, c_n, d_n имеют интересное свойство – их относительные величины являются также и длинами
диагоналей (2N-1)-угольника (N – число используемых
файлов). Например, для 4 диагонали 9-угольника имеют относительные длины, очень близкие к 85, 75, 56 и 30.
В книге Кнута приведено доказательство того факта, что значения относительного времени, затрачиваемого на (N-1), (N-2), …, 1-путевое слияние, приблизительно пропорциональны квадратам длин этих диагоналей.

Каскадное слияние - этапы

1. Распределение.
Распределяем элементы по N файлам, согласно оптимальному приведенному выше алгоритму. Один файл будем использовать дополнительно. 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.

Каскадное слияние - этапы

2. Слияние - прямое

- На первом шаге сливаем файлы F_1, F_2, …, F_n в файл F_{n+1} с помощью n-путевого слияния, пока один из файлов (например, F_n) не станет пустым.
- На втором шаге сливаем файлы F_1, F_2, …, F_{n-1} в файл F_n с помощью (n-1)-путевого слияния, по один из файлов (например, F_{n-1}) не станет пустым
- Продолжаем эту процедуру до шага 1-путевого слияния (копирование) из файла F_1 в файл F_2.

Каскадное слияние - этапы

3. Слияние – обратное

Повторяем процедуру прямого слияния, только в обратном порядке

- На первом шаге сливаем файлы F_2, F_3, …, F_{n+1} в файл F_1 с помощью n-путевого слияния, пока один из файлов (например, F_2) не станет пустым.
- На втором шаге сливаем файлы F_3, F_4, …, F_{n+1} в файл F_2 с помощью (n-1)-путевого слияния, по один из файлов (например, F_3) не станет пустым
- Продолжаем эту процедуру до шага 1-путевого слияния (копирование) из файла F_{n+1} в файл F_n.

Каскадное слияние - этапы

4. Цикл

Повторяем этапы 2 и 3 до тех пор, пока все файлы кроме 1 не станут пустыми.

Пример (N=4)

# прохода	F_1	F_2	F_3	F_4	F_5
1	1x85	1x75	1x56	1x30	-
2	-	1x10	2x19	3x26	4x30
3	10x10	9x9	7x7	4x4	-
4	-	10x1	19x2	26x3	30x4
5	85x1	75x1	56x1	30x1	-
6	-	-	-	-	246x1

Общее количество начальных элементов – 246
19x2 означает 2 серии по 19 элементов

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 N, и Дэвидом Фергюсоном (1964 г.).