**Лекция 7. Алгоритмы внешнего сортировка**

**Сортировка слиянием**

**Сортировка простым слиянием**

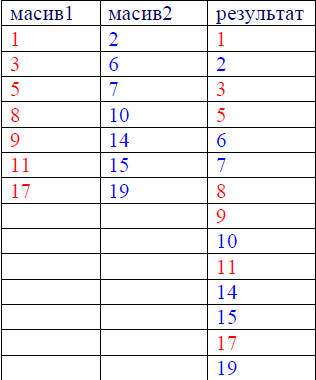
В основе данного метода внутренней сортировки лежит процедура слияния двух упорядоченных массивов. Эти массивы должны быть объединены таким образом, чтобы получился один упорядоченный массив.

Пусть имеется два массива А и В, упорядоченные по возрастанию ключей. Слияние заключается в поочередной пересылке элементов из массивов А и В в зону вывода С. Порядок пересылки в зону вывода зависит от результата попарного сравнения ключей массивов А и В.

Сортировка слиянием производится на основе использования рассмотренной выше процедуры. Сущность сортировки состоит в том, что массив, который мы пытаемся отсортировать, разделяется на равные группы элементов. Группы упорядочиваются, а затем попарно сливаются, образуя новые группы.

На первом этапе каждая группа содержит два соседних элемента исходного массива. Элементы внутри групп упорядочиваются (например, методом вставки или включения). Затем происходит попарное слияние групп. Количество групп в списке уменьшается до тех пор, пока не будет получена одна упорядоченная группа. Если число элементов нечетное, то вводится дополнительный элемент с максимальным значением. После сортировки он отбрасывается. Если число групп, сформированных на первом этапе, нечетное, то нечетное группа переписывается без слияния.

Пусть есть 2 массивы, каждый из которых уже отсортирован:



**Слияния и внешнее сортировка**

**слияние**

Внешнее сортировки сортирует файлы, которые не помещаются целиком в оперативную память.

Внешнее сортировки сильно отличается от внутреннего. Дело в том, что доступ к файлу является последовательным, а не параллельным как это было в массиве. И поэтому считывать файл можно только блоками и этот блок отсортировать в памяти и снова записать в файл.

Принципиальную возможность эффективно отсортировать файл, работая с его частями и не выходить за пределы части обеспечивает алгоритм слияния.

Под **слиянием** понимается объединение двух (или более) упорядоченных последовательностей в одну упорядоченную последовательность с помощью циклического выбора элементов, доступных в данный момент.

Слияние - гораздо более простая операция, чем сортировка.

Мы рассмотрим 2 алгоритма слияния:

1. Прямое слияния. Алгоритм Боуза - Нельсона

2. Естественное (неймановской) слияние.

**Прямое слияние. Алгоритм Боуза - Нельсона**

1. Последовательность, А разбивается на две половины b и с.

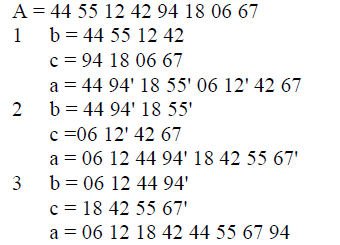
2. Последовательности b и с сливаются с помощью объединения отдельных элементов в упорядоченные пары.

3. Полученной последовательности присваивается имя а затем повторяются шаги 1 и 2; при этом упорядоченные пары сливаются в упорядоченные четверки.

4. Предыдущие шаги повторяются, при этом четверки сливаются в восьмерки и т.д., пока не будет упорядочена вся последовательность, потому что длины последовательностей каждый раз удваиваются.

пример

Выходная последовательность



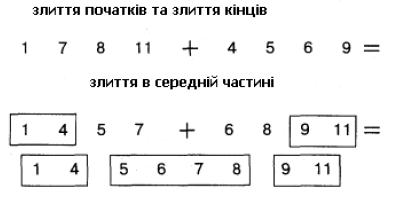
Операция, однократно обрабатывает все множество данных, называется фазой.

Наименьший подпроцесс, что, повторяясь, образует процесс сортировки, называется проходом или этапом.

В нашем примере сортировки осуществляется за три прохода. Каждый проход состоит из фазы разбиения и фазы слияния.

Главным минусом сортировки слиянием является удвоение размера памяти, первоначально занятой сортуемимы данными. Рассмотрим алгоритм с рекурсивным актом слияния, предложенный Боуз и Нельсоном и не требующим резерва памяти.

Он основан на очевидной идеи: слить две равные упорядоченные части можно слиянием первоначальных половин, слиянием конечных и слиянием 2-й половины 1-го результата с 1-й половиной 2-го результата, например



Если части не равны или не делятся точно пополам, процедуру уточняют должным образом. Аналогичное слияние "половинок" можно свести к слиянию "четвертушек", "восьмушок" и т.д .; имеет место рекурсия.

Const n = 200;

Type

tipkl = word;

tip = Record

kl: tipkl;

z: Array [1..50] of real

End;

Var

A: Array [1..n] of tip;

j: word;

Procedure Bose (Var AA; voz: Boolean);

Var

m, j: word; x: tip; {Tip - тип сортируемых записей}

A: Array [1..65520 div Sizeof (tip)] of tip Absolute AA;

Procedure Sli (j, r, m: word); {R - расстояние между началами

частей, сливаются, а m - их размер, j - наименьший номер записи

}

Begin if j + r <= n

Then If m = 1

Then

Begin If voz Xor (A [j] .kl <A [j + r] .kl) Then

Begin

x = A [j];

A [j]: = A [j + r];

A [j + r] = x

End

End

Else

Begin

m = m div 2;

Sli (j, r, m) {Слияние "начал"}

If j + r + m <= n Then

Sli (j + m, r, m) {Слияние "концов"}

Sli (j + m, r-m, m) End {Слияние в центральной части}

End {блока Sli};

Begin

m = 1;

Repeat

j = 1; {Цикл слияния списков равного размера:}

While j + m <= n do

Begin

Sli (j, m, m)

j = j + m + m

End;

m = m + m {Удвоение размера списка перед началом нового прохода}

Until m> = n {Конец цикла, реализующая все дерево слияний}

End {блока Bose};

BEGIN

Randomize;

For j = 1 to n do

begin

A [j] .kl = Random (65535)

Write (A [j] .kl: 8);

end;

Readln;

Bose (A, true);

For j = 1 to n do

Write (A [j] .kl: 8);

Readln

END.

**Естественное (Нейманивске) слияние**

Сочетаются упорядоченные части спонтанно возникли в исходном массиве; они могут быть также следствием предварительной обработки данных. Рассчитывать на одинаковый размер частей, сливаются, не приходится.

Записи, которые идут в порядке неубывания ключей, сцепляются, образуя подсписок. Минимальный подсписок одна запись.

пример:

Пусть даны ключи записей

5 7 8 3 9 4 1 7 6

ищем подсписки



В один общий список соединяются 1-й, 3-й, 5-й и т.д. подсписки, в другой - 2-й, 4-й и т.д. подсписки.

Сделаем слияния 1 подсписке 1 списка и 1 подсписке 2 списка, 2 подсписке 1 списка и 2 подсписке 2 списка и т.д.

Будут получены следующие цепи

3 ---> 5 ---> 7 ---> 8 ---> 9 и 1 ---> 4 ---> 7

Подсписок, состоящий из записи "6", пары нет и "принудительно" сочетается с последним цепью, принимает вид 1 ---> 4 ---> 6 ---> 7.

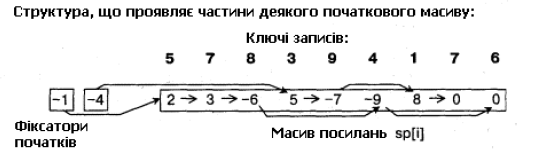
При нашем небольшом числе записей 2-й этап, на котором сливаются два

цепи, окажется последним.

В общем случае на каждом этапе подсписок - результат слияния начальных подсписков 1 и 2 списка становится началом нового 1-го списка, а результат слияния следующих двух подсписков - началом 2-го списка. Следующие образованные подсписки по очереди включаются в 1-й и во 2-й список.

Для программной реализации создают массив sp: элемент sp [i] - это номер записи, следующего за i-й.

Последняя запись одного подсписке ссылается на первую запись другого, а для различения концов подсписков эту ссылку дополняется знаком минус.



Repeat {Повторение актов слияния подсписков}

If A [j] .kl <A [i] .kl Then {Выбирается меньше запись}

Begin sp [k] = j; k = j; j = sp [j];

If j <= 0 Then {Сцепление с остатком "i" -пидсписка}

Begin sp [k] = i; Repeat m = i; i = sp [i] Until i <= 0 End

End

Else

Begin sp [k] = i; k = i; i = sp [i];

If i <= 0 Then {Сцепление с остатком "j" -пидсписка}

Begin sp [k] = j; Repeat m = j; j = sp [j] Until j <= 0 End

End;

If j <= 0 Then Begin sp [m] = 0; sp [p] = - sp [p]; i = - i;

j = - j; If j <> 0 Then p = r; k = r; r = m End

Until j = 0;

В конец сформированного подсписка всегда заносится нулевое ссылки (sp [m] = 0), так как он может оказаться последним.

Действие sp [p] = -sp [p] обозначает минусом конец ранее построенного подсписка.

В переменных i, j ссылки на начала новых пидсписков сливающихся - со знаком минус; его снимаем. Переход к новым подсписков требует восстановления переменных p, k, r}

**Внешнее сортировка**

Из мира "электронных" скоростей переместимся в мир меньших скоростей, мир с механическими перемещениями и влиянием инерции. Хотя диск и делает сотни оборотов в секунду, проходят миллисекунды до момента, когда требуется запоминающее элемент окажется под головкой чтения / записи. Поэтому если любой известный нам способ упорядочения в

памяти применить "механически" к данным на диске, затраты времени окажутся, скорее всего, неприемлемыми.

Сортировка данных на магнитных дисках называют внешним.

Сложности при сортировке на диске.

Исследовать, а тем более конструировать внешнее сортировки невозможно, не представляя, мало-мальски, специфики дисков.

Диски имеют запоминающие магнитные поверхности и вращаются с большой постоянной скоростью под головками чтения / записи. Головка обслуживает одну поверхность.

При фиксированном положении головок доступ происходит с данными треков (окружностей), находящихся под головками. Совокупность этих треков назовем текущим цилиндром.

Головки крепятся на штанге, подобной "коромысла" проигрывателя. После перемещения блока головок (поворота штанги) текущим становится следующий цилиндр. Цилиндры нумеруются в порядке их прохождения блоком головок при движении штанги в одну сторону. Расстоянием между элементами данных назовем разницу номеров цилиндров, где они размещены.

Адресация (нумерация) ячеек памяти на диске идет в пределах цилиндра с переходом по исчерпании его объема на следующий, потом на третий и т.д. Файл данных записывается по порядку адресов, но если свободного участка не хватило, продолжается в другой области и, возможно, на другом цилиндре (случай фрагментации файла).

Изложенная достаточно упрощенная модель диска, но ее достаточно.

Хотя диски называют памятью прямого доступа, время обращения к данным гораздо большего порядка, чем при чтении / записи в основной памяти, через

а) задержки, связанные с ожиданием момента, когда требуется элемент цилиндра пройдет под головкой;

б) перемещение головок, когда нужны данные не с текущего цилиндра.

Прогон - это прохождение файла в одном направлении со считыванием данных в память и, возможно, их возвращением в файл.

{По возможности большую часть работы выполняйте данным, вызванными в память, не обращаясь к диску до конца обработки этой части.}

Применительно к внешней сортировки это означает упорядочение возможно больших частей файла внутренним сортировкой: мы заменяем "быстрыми" перемещениями в памяти перемещения данных на диске. Уменьшая число актов их чтения и перезаписи, мы одновременно исключаем многочисленные задержки "а", "б".

Увеличение физического размера считывающей (записываемой) порции данных сокращает задержку и число актов чтения (записи), а вместе с ними и число промежуточных движений блока головок, которые назовем ходами штанги или просто ходами.

Время внешнего сортировки зависит от

а) внутренней сортировки частей файла;

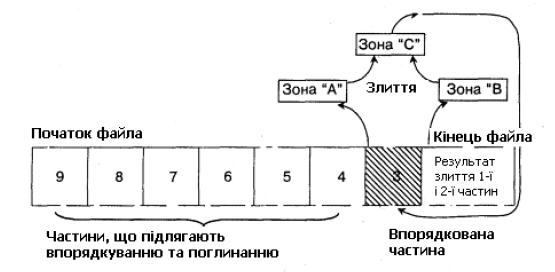
б) многократного считывания и записи данных на диск;

в) ходов головки между актами чтения / записи;

г) действий в памяти при слиянии упорядоченных частей.

**Сортировка методом поглощения.**

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



Перед поглощением очередная часть файла считывается в зону "А" памяти, там упорядочивается и остается. Начало ранее упорядоченной части считывается в зону "В" и начинается слияние, прерываемый считыванием в зону "В", когда она опустошается. По мере заполнения зоны "С" записями акта слияния, содержимое ее переписывается в файл (на место поглощаемы части и дальше в сторону конца файла). Если при слиянии взяты все записи поглощаемы части, поглощение завершается передачей в файл зоны "С" остатка результата. Слияние также завершается, если исчерпаны раньше упорядоченную часть. Поглощения ею очередной части произошло.

Ходов в данном методе будет мало и при сортировке вы не услышите характерного поскрипывания механизма головок. Это дает экономию времени и при небольшом размере файла сортировки проходит быстро.

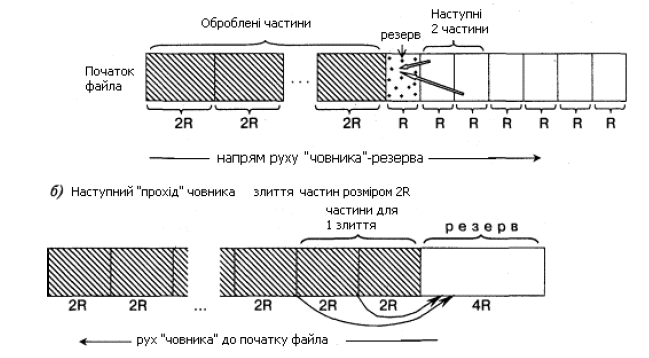
**Челночное балансовое слияния**

На 1 этапе внутренней сортировки частей в файле создают М упорядоченных частей, по возможности большего размера R. К ним применяют прямое слияние.

Создают резервный дисковое пространство файла и располагают его непосредственно до или после частей, сливающихся и перемещается к следующим частей по завершении слияния. Его размер не менее размера R части. Резерв и пространство ближайшей части будут заполнены результатом слияния двух первых частей, при этом пространство 2-й части освободится и станет резервом, необходимым для слияния следующей пары частей и т.д.

По мере слияния частей резерв перемещается от начала файла до конца, потом обратно и т.д., как челнок. Поскольку место результата не удалено от частей, сливаются, ходы небольшие, пока сами части невелики. В процессе сортировки "челнок" увеличивают, так как растут части:

а) слияние частей размером R (промежуточная ситуация)



Стрелками показано перемещение данных при слиянии.

Заметим, что резерв можно увеличивать, когда он в конце файла; это происходит один раз за 2 прогоны челнока.

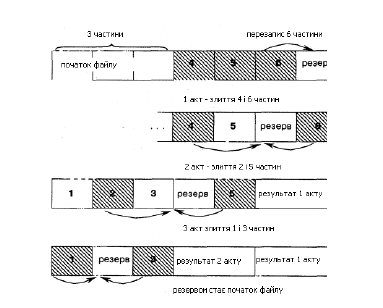
Все бы хорошо, однако, челнок величиной половины файла может занимать его начало.

Хорошо бы ограничить рост размера "челнока", потому что этот рост увеличивает размер ходов.

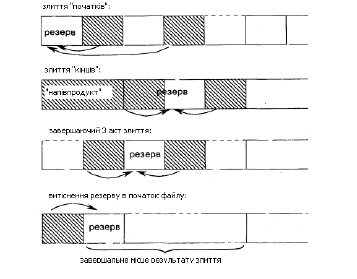
Путем модификации конечных этапов слияния добьемся, чтобы этот рост остановился на размере D = 1/6 размера файла.

Для этого все равно надо, чтобы программа так определила выходные размер и положение резерва (в начале или в конце файла), чтобы в момент, когда останутся всего 3 большие части, резерв, что размер D, стоял в начале файла.

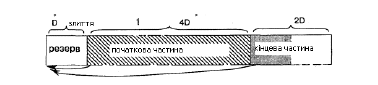
в) этап слияния (с подэтапа), когда в файле 6 частей.



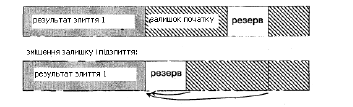
г) этап слияния "по половинах" (естественное слияние), когда частей 3. Конечная часть файла на этом этапе не используется.



д) заключительный этап слияния; сначала берем для слияния первую половину конечной части и всю начальную часть:



По окончании слияния 1 выясняется, надо сместить остаток начальной части, чтобы освободить место для конца результата и продолжить слияния ( "пидзлиття»). Пидзлиття не требуется, если в ходе слияния начальная часть исчерпана, надо только вытеснить резерв в конец файла, переписав окончания.



И при слиянии 1, и при пидзлитти размер резерва не менее минимально необходимого.