Уральский федеральный университет

СТРУКТУРЫ ДАННЫХ

Лектор
Спиричева Наталия Рахматулловна

Ст. преподаватель каф. ИТ
Ауд. Р-246

Структуры данных

Составитель курса лекций:
Спиричева Наталия Рахматулловна,
ст. преподаватель каф. Информационных технологий

Структуры данных и алгоритмы

Целью лекции является приобретение студентами следующих компетенций:

- знать основные алгоритмы сортировок массивов и их характеристики
- знать основные операции логического уровня над статическими структурами данных

Структуры данных и алгоритмы

Основные темы лекции:

- Поиск
- Сортировка
- Прямой доступ и хеширование

Структуры данных и алгоритмы

- Операции логического уровня над статическими структурами.
- Поиск

Операции логического уровня над статическими структурами. Поиск

Порядком алгоритма называется функция O(N), позволяющая оценить зависимость времени выполнения алгоритма от объема перерабатываемых данных (N - количество элементов в массиве или таблице).

Большинство алгоритмов с точки зрения порядка сводятся к трем основным типам:

степенные - O(Na);

линейные - O(N);

логарифмические - O(loga(N)).

Поиск

Последовательный или линейный поиск

Простейший метод поиска элемента - последовательный просмотр каждого элемента набора, который продолжается до тех пор, пока не будет найден желаемый элемент.

Для последовательного поиска в среднем требуется (N+1)/2 сравнений. Таким образом, порядок алгоритма - линейный - O(N).

Поиск

Бинарный поиск

Метод бинарного поиска выполняется в заведомо упорядоченной последовательности элементов.

Записи в таблицу заносятся в лексикографическом (символьные ключи) или численно (числовые ключи) возрастающем порядке.

Для того чтобы найти нужную запись в таблице, в худшем случае требуется log2(N) сравнений.

Статические структуры данных

Сортировка

Операции логического уровня над статическими структурами. Сортировка

Классификация алгоритмов сортировки:

1). Стратегия выборки.
2). Стратегия включения.
3). Стратегия распределения.
4). Стратегия слияния.

Алгоритмы сортировки

Сортировка – процесс перегруппировки заданного множества объектов в некотором порядке.

Сортировка применяется во всех без исключения областях программирования, будь то базы данных или математические программы.

Практически каждый алгоритм сортировки можно разбить на три части:

- сравнение, определяющее упорядоченность пары элементов;
- перестановка, меняющая местами пару элементов;
- собственно сортирующий алгоритм, который осуществляет сравнение и перестановку элементов до тех пор, пока все элементы множества не будут упорядочены.

Сортировка

Методы сортировки разделяют на две категории: *сортировка массивов* и *сортировка файлов*. Эти два класса часто называют *внутренней* и *внешней* сортировкой, потому что массивы располагаются в оперативной памяти ЭВМ; для этой памяти характерен быстрый произвольный доступ, а файлы хранятся в более медленной, но более вместительной внешней памяти.

Пузырьковая сортировка вставками

При такой сортировке входное и выходное множества находятся в одной последовательности, причем выходное - в начальной ее части. В исходном состоянии можно считать, что первый элемент последовательности уже принадлежит упорядоченному выходному множеству, остальная часть последовательности - неупорядоченное входное. Первый элемент входного множества примыкает к концу выходного множества. На каждом шаге сортировки происходит перераспределение последовательности: выходное множество увеличивается на один элемент, а входное - уменьшается. Это происходит за счет того, что первый элемент входного множества теперь считается последним элементом выходного. Затем выполняется просмотр выходного множества от конца к началу с перестановкой соседних элементов, которые не соответствуют критерию упорядоченности.

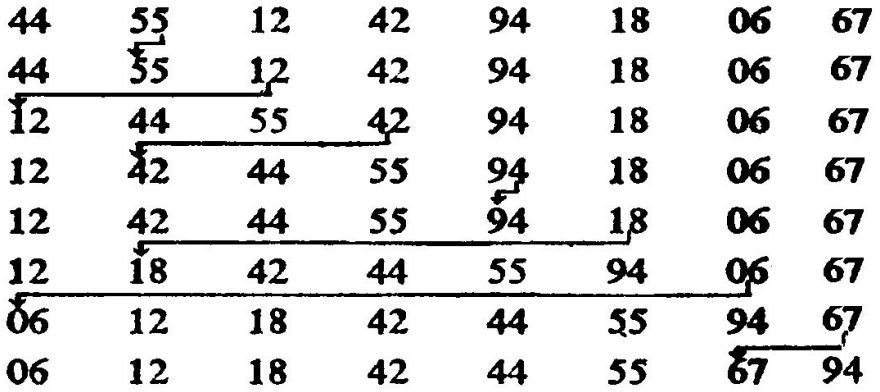
Просмотр прекращается, когда прекращаются перестановки. Это приводит к тому, что последний элемент выходного множества "выплывает" на свое место в множестве. Поскольку при этом перестановка приводит к сдвигу нового в выходном множестве элемента на одну позицию влево, нет смысла всякий раз производить полный обмен между соседними элементами - достаточно сдвигать старый элемент вправо, а новый элемент записать в выходное множество, когда его место будет установлено.

Сортировка массивов

Сортировка простыми включениями

Элементы условно разделяют на готовую и входную последовательности. Перебирая элементы входной последовательности, передают этот элемент в готовую последовательность, вставляя его на подходящее место. Изначально готовая последовательность содержит первый элемент массива.

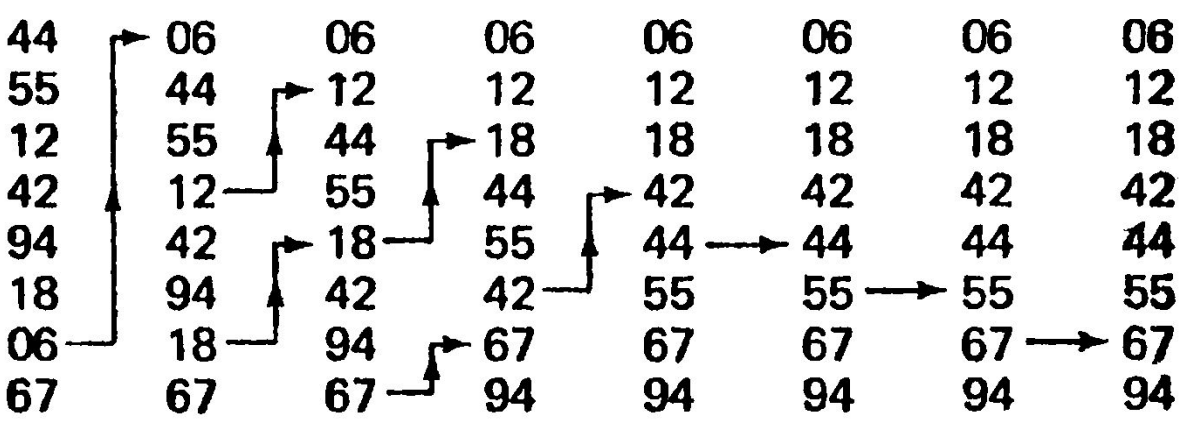
Процесс сортировки простыми включениями показан на примере восьми случайно взятых чисел



Количество сравнений элементов зависит от исходного порядка элементов массива. Недостаток метода - большое число пересылок элементов.

Сортировка простым обменом (метод пузырька)

Идея этого метода отражена в его названии. Самые легкие элементы массива "всплывают" наверх, самые "тяжелые" - тонут. Алгоритмически это можно реализовать следующим образом. Будем просматривать весь массив "снизу вверх" и менять стоящие рядом элементы в том случае, если "нижний" элемент меньше, чем "верхний". Таким образом, вытолкнем наверх самый "легкий" элемент всего массива. Теперь повторим всю операцию для оставшихся неотсортированных элементов (т.е. для тех

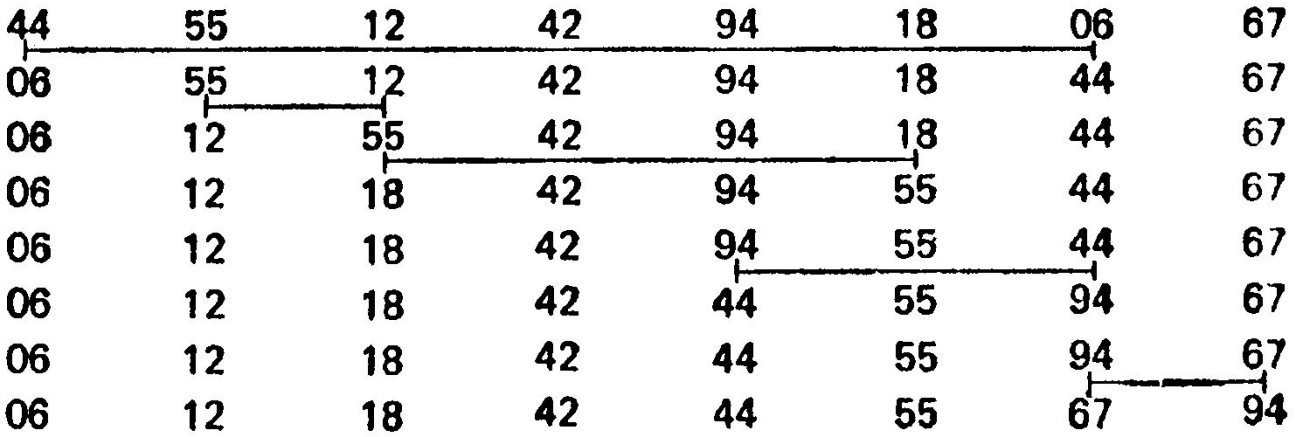


Как видно, алгоритм достаточно прост, но, как иногда замечают, он является непревзойденным в своей неэффективности.

Сортировка простым выбором

Суть метода: выбирается наименьший элемент массива и меняется местами с первым элементом. Эти операции повторяются с оставшимися элементами, кроме первого элемента, затем кроме первого и второго элементов массива, пока не останется только один элемент – наименьший.

Этот метод продемонстрирован на восьми элементах:



Число сравнений элементов не зависит от исходного порядка элементов массива.

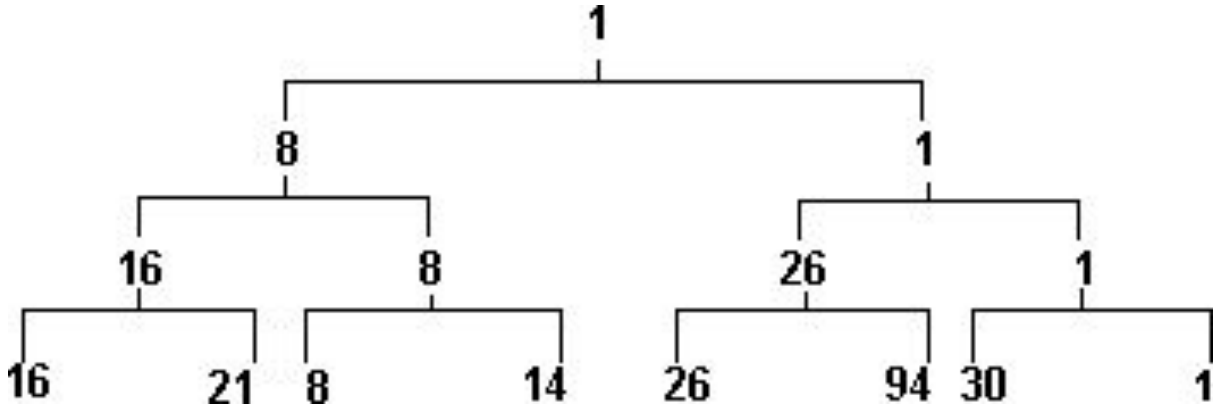
Сортировка упорядоченным двоичным деревом

Алгоритм складывается из построения упорядоченного двоичного дерева и последующего его обхода. Если нет необходимости в построении всего линейного упорядоченного списка значений, то нет необходимости и в обходе дерева, в этом случае применяется поиск в упорядоченном двоичном дереве.

Порядок алгоритма - O(N*log2N)), но в конкретных случаях все зависит от упорядоченности исходной последовательности, которая влияет на степень сбалансированности дерева и в конечном счете - на эффективность поиска.

Турнирная сортировка

Этот метод сортировки получил свое название из-за сходства с кубковой системой проведения спортивных соревнований: участники соревнований разбиваются на пары, в которых разыгрывается первый тур; из победителей первого тура составляются пары для розыгрыша второго тура и т.д. Алгоритм сортировки состоит из двух этапов. На первом этапе строится дерево, аналогичное схеме розыгрыша кубка.



Сортировки распределением

ПОРАЗРЯДНАЯ ЦИФРОВАЯ СОРТИРОВКА.

Алгоритм требует представления ключей сортируемой последовательности в виде чисел в некоторой системе счисления P. Число проходов сортировки равно максимальному числу значащих цифр в числе - D. В каждом проходе анализируется значащая цифра в очередном разряде ключа, начиная с младшего разряда. Все ключи с одинаковым значением этой цифры объединяются в одну группу. Ключи в группе располагаются в порядке их поступления. После того как вся исходная последовательность распределена по группам, группы располагаются в порядке возрастания связанных с группами цифр. Процесс повторяется для второй цифры и т.д., пока не будут исчерпаны значащие цифры в ключе. Основание системы счисления P может быть любым, в частном случае 2 или 10. Для системы счисления с основанием P требуется P групп.

Быстрая сортировка хоара

Данный алгоритм относится к распределительным и обеспечивает показатели эффективности O(N*log2(N)) даже при наихудшем исходном распределении.

Используются два индекса - *l* и r - с начальными значениями 1 и N соответственно. Ключ K[l] сравнивается с ключом K[r]. Если ключи удовлетворяют критерию упорядоченности, то индекс *r* уменьшается на 1 и производится следующее сравнение ключей. Если ключи не удовлетворяют критерию, то записи R[l] и R[r] меняются местами.

Сортировка слиянием

Алгоритмы сортировки слиянием, как правило, имеют порядок O(N*log2(N)), но отличаются от других алгоритмов большей сложностью и требуют большого числа пересылок. Алгоритмы слияния применяются в основном, как составная часть внешней сортировки.

Сортировка попарным слиянием

Входное множество рассматривается как последовательность подмножеств, каждое из которых состоит из единственного элемента и, следовательно, является уже упорядоченным. На первом проходе каждые два соседних одноэлементных множества сливаются в одно двухэлементное упорядоченное множество. На втором проходе двухэлементные множества сливаются в 4-элементные упорядоченные множества, и т.д. В конце концов получается одно большое упорядоченное множество.

Сортировка бинарными включениями (модифицированная сортировка простыми включениями)

Готовая последовательность, в которую нужно включить новый элемент массива, уже упорядочена. Поиск места включения: исследуем средний элемент готовой последовательности и продолжаем деление пополам, пока не будет найдено место включения. Такой поиск называется бинарным.

Достоинства: уменьшение числа сравнений элементов. Количество сравнений элементов не зависит от исходного порядка элементов массива.

Недостаток: большое число пересылок элементов, что значительно больше времени занимает, чем сравнение элементов.

Метод предсортировки и слияния

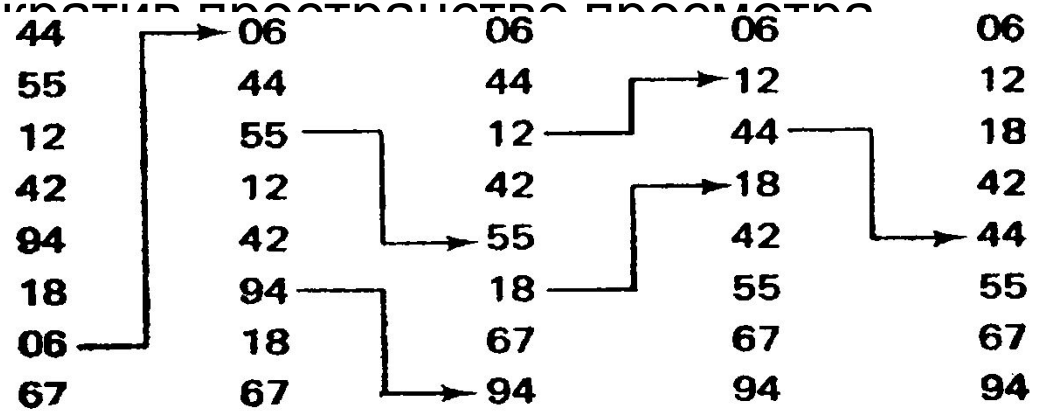
В основе данного метода лежит следующий алгоритм: основной массив разбивается на два дочерних, в отношении каждого из которых осуществляется первичная сортировка. Затем отсортированные массивы сливаются, и полученный массив подвергается вторичной сортировке - в результате получаем отсортированный массив.

Метод максимумов

- Просматривая массив от первого элемента, найти максимальный элемент и поместить его в последнюю ячейку промежуточного массива, а в основном массиве ячейка, содержащая найденный максимум, обнуляется.
- Просматривая массив от первого элемента, найти максимальный элемент и поместить его в предпоследнюю ячейку промежуточного массива, а в основном массиве ячейка, содержащая найденный максимум, обнуляется.
- И так далее.

Шейкер-сортировка

Модифицированный алгоритм сортировки простым обменом. Во-первых, т.к. метод «пузырька» асимметричен, то просмотр и обмены удобно осуществлять поочередно с разных концов. Во-вторых, можно запоминать не только сам факт наличия обменов, но и место (индекс) последнего обмена (все следующие элементы уже упорядочены), тем самым сокращая пространство просмотра.



Достоинства: уменьшение избыточных повторных проверок.

Недостаток: не изменилось число обменов элементов.

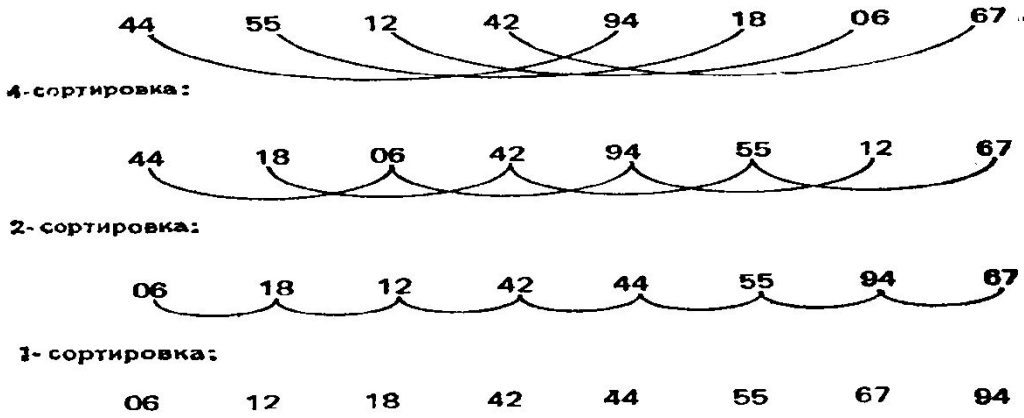
Сортировка включениями с убывающим приращением (сортировка Шелла)

Некоторое усовершенствование сортировки простыми включениями было предложено Д.Л. Шеллом в 1959 году.

Основная идея этого алгоритма заключается в том, чтобы в начале устранить массовый беспорядок в массиве, сравнивая далеко стоящие друг от друга элементы. Интервал между сравниваемыми элементами постепенно уменьшается до единицы. Это означает, что на поздних стадиях сортировка сводится просто к перестановкам соседних элементов (если, конечно, такие перестановки являются необходимыми).

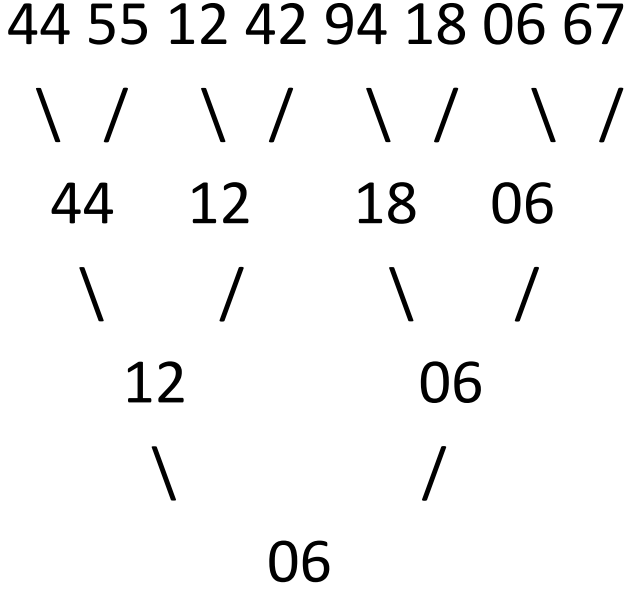
Рассмотрим пример, представленный на рисунке. На первом проходе отдельно группируются и сортируются все элементы, отстоящие друг от друга на четыре позиции. Этот процесс называется 4-сортировкой. В примере из восьми элементов каждая группа содержит ровно два элемента. После этого элементы вновь объединяются в группы с элементами, отстоящими друг от друга на две позиции, и сортируются заново. Этот процесс называется 2-сортировкой. Наконец, на третьем проходе все элементы сортируются обычной сортировкой, или 1-сортировкой.

Приращения не должны быть кратны друг другу. Это позволяет избежать явления, где каждый проход сортировки объединяет две цепочки, кото[illegible]ли

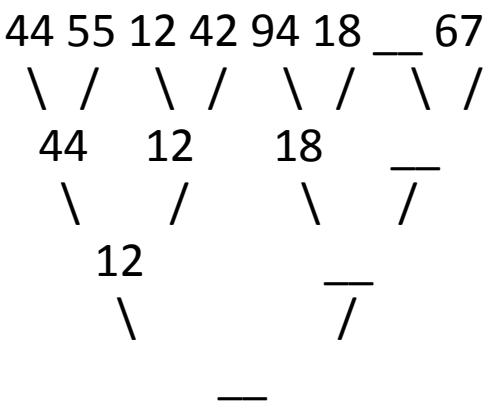


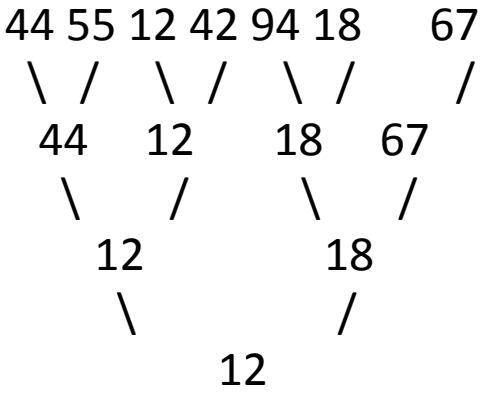
Сортировка с помощью дерева

Сортировка сводится к разбиению массива на пары и выбору меньшего среди них, последующему разбиению и выбору. Таким образом строим дерево выбора:



На следующем шаге мы поднимаемся по ветви наименьшего элемента, заменяя его на элемент из противоположного узла вышестоящего уровня, или удаляем его, если он крайний.



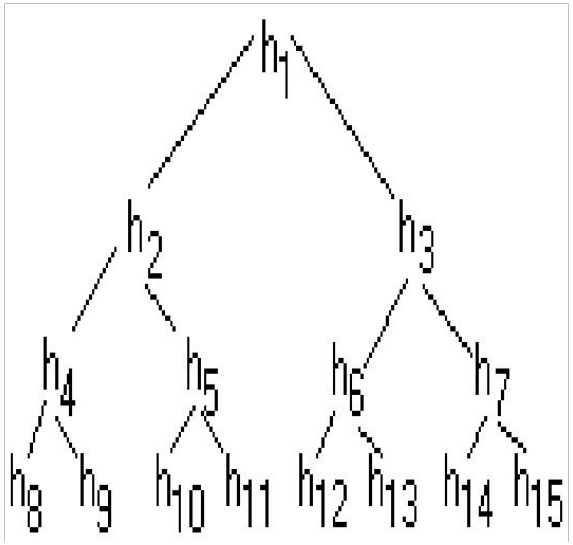


Повторив последний шаг несколько раз ,дерево пропадает.

Пирамидальная сортировка

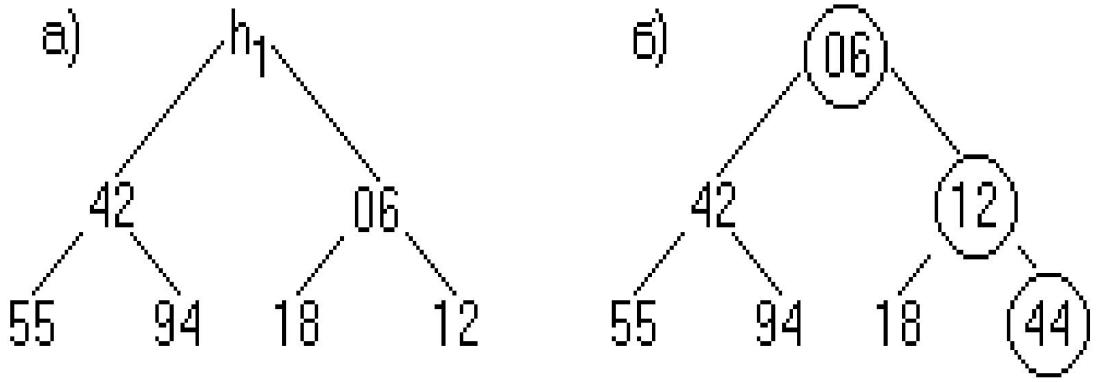
Пирамида – двоичное дерево с упорядоченными листьями (корень дерева – наименьший элемент). Пирамиду можно представить в виде массива. Первый элемент пирамиды является наименьшим.

Массив, расположенный в виде бинарного дерева



Просеивание – построение новой пирамиды: помещение нового элемента в вершину дерева, далее он перемещается («просеивается») по пути, на котором находятся меньшие по сравнению с ним элементы, которые одновременно поднимаются вверх.

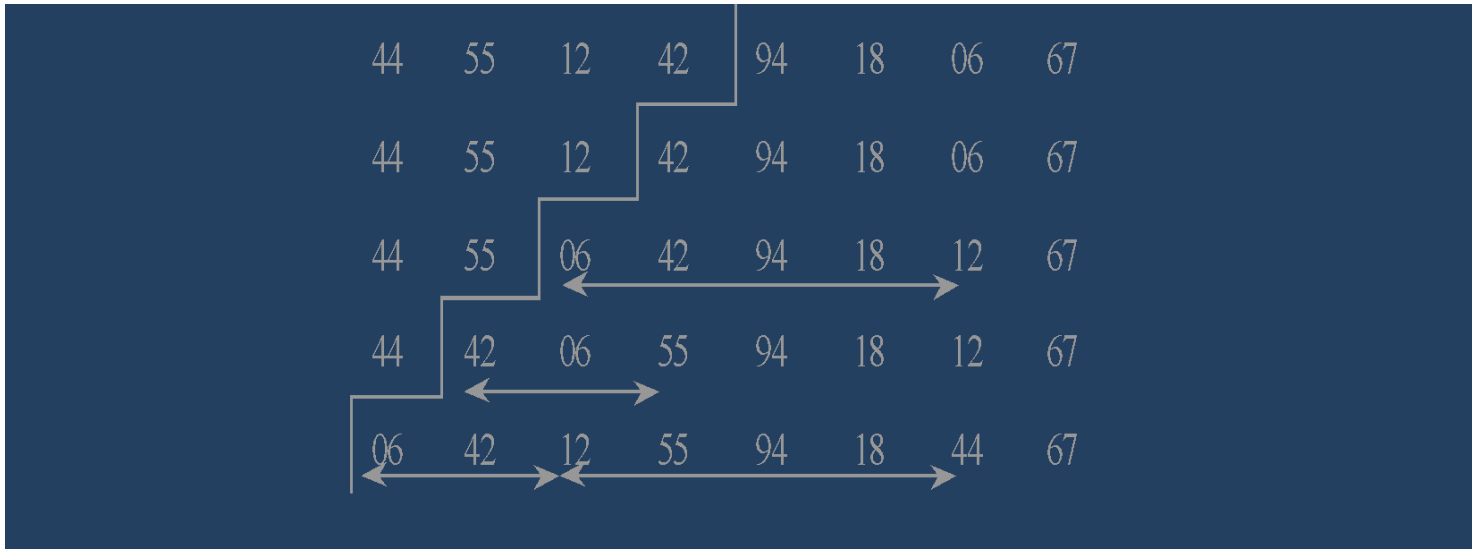
Например, возьмём исходную пирамиду, показанную на рисунке «а», и расширим эту пирамиду «влево», добавив элемент h_1=44. Значение 44 сначала меняется местами с 06, затем с 12, и так формируется дерево, показанное на рисунке «б».

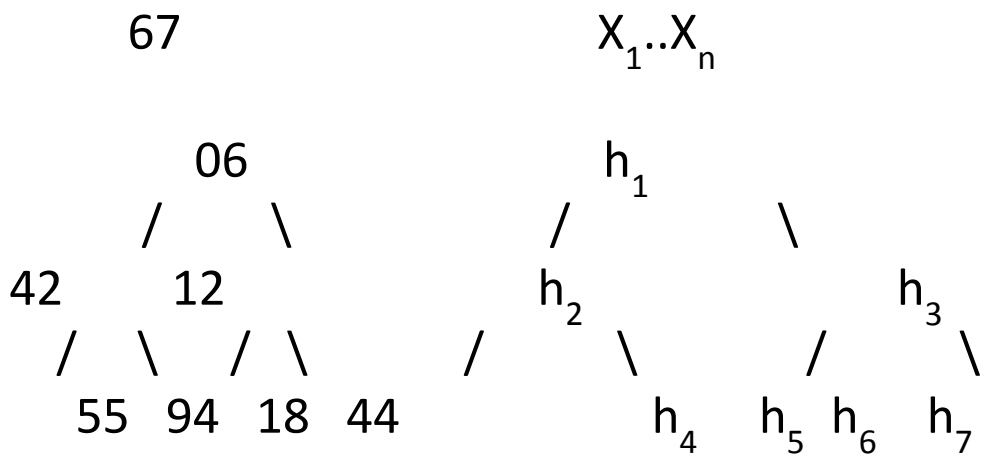


а) пирамида из семи элементов;

б) просеивание элемента 44 через пирамиду.

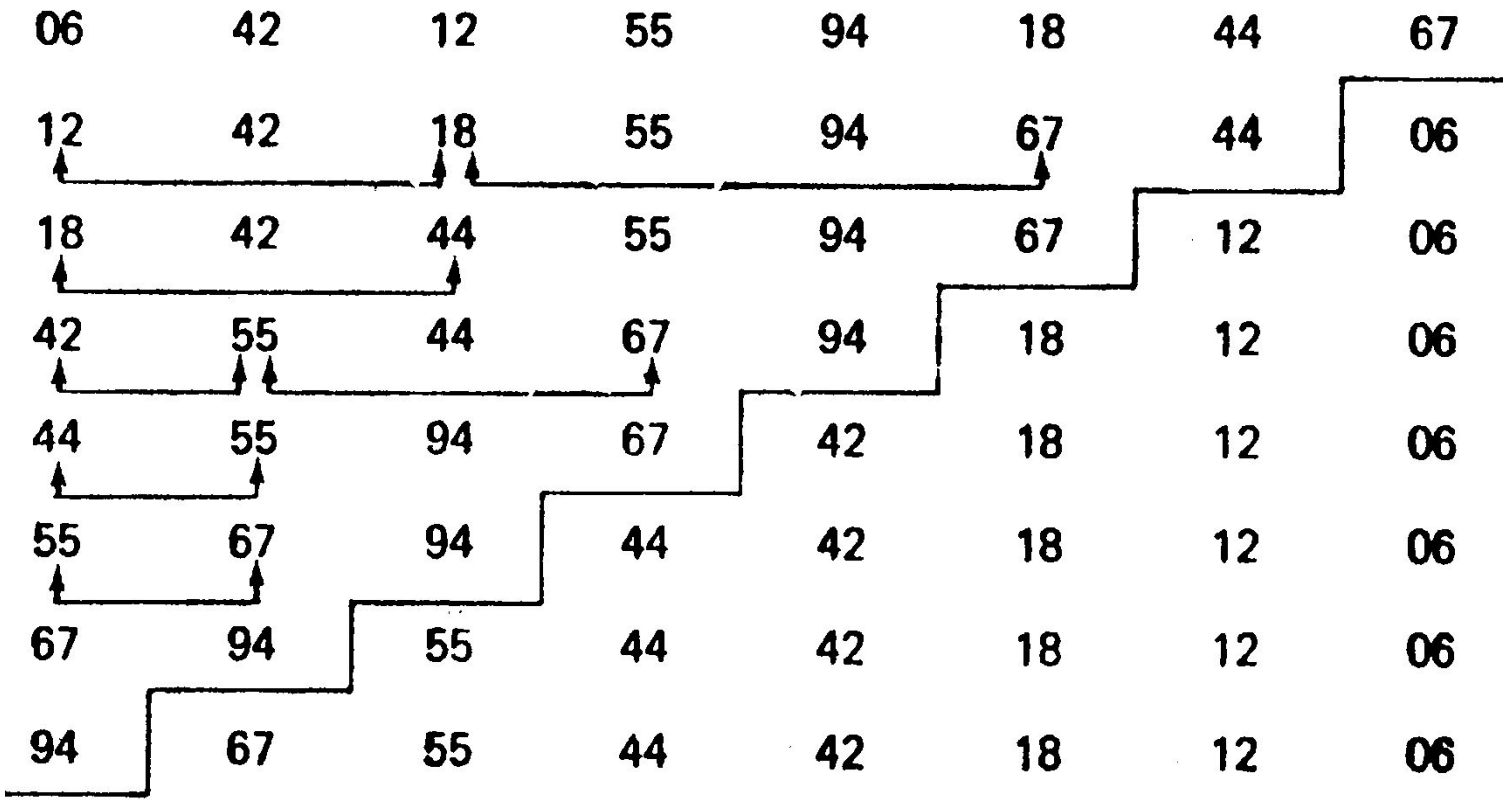
Способ построения пирамиды, предложенный Р.У. Флойдом: пусть дан массив $h_1,\ldots h_n$, тогда элементы $h_{n/2+1}\ldots h_n$ уже образуют пирамиду. Эти элементы составляют последовательность, которую можно рассматривать как нижний ряд соответствующего двоичного дерева. Пирамида расширяется влево: на каждом шаге добавляется новый элемент и при помощи просеивания помещается на соответствующее место.





h_1 (наименьшее число) записывается в отсортированный массив, X записывается в h_1 и просеивается. Это повторяется, пока не закончатся свободные элементы. После этого h1 записывается в отсортированный массив, а на его место записывается меньший элемент из следующего уровня, на место которого в свою очередь записывается меньший элемент следующего уровня этой ветви, и т.д. На место элемента из нижнего уровня записывается последний по счету элемент.

Пример пирамидальной сортировки

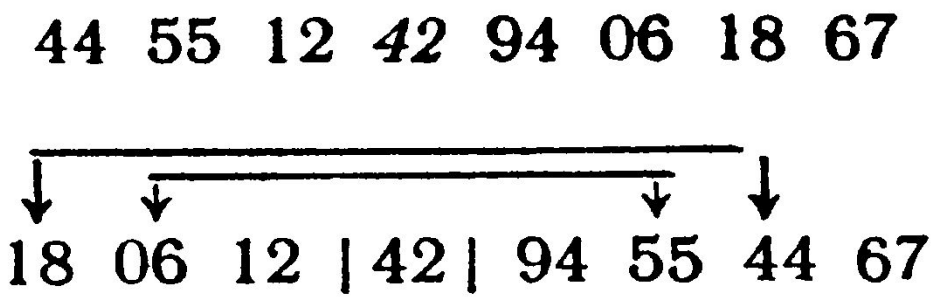


Быстрая сортировка (метод хоора)

Этот метод разработан К. Хоором в 1962 году.

Суть метода заключается в том, чтобы выбрать случайным образом какой-то элемент, просмотреть массив, двигаясь слева направо, пока не будет найден элемент, больший выбранного. Затем просмотреть его справа налево, пока не будет найден элемент, меньший выбранного. После этого поменять эти элементы местами и продолжить «просмотр с обменом», пока два просмотра не встретятся где-то в середине массива. В результате массив разделится на две части: левую – с меньшими элементами, чем выбранный, и правую – с большими элементами. Разделив массив, нужно сделать то же самое с обеими полученными частями, затем с частями этих частей и т.д., пока каждая часть не будет содержать только один элемент.

Пример первого разделения массива:



Преимущество: для сортировки уже разделенных небольших подмассивов легко применить какой-либо простой метод.

Недостаток: эффективность алгоритма быстрой сортировки определяется выбором случайного начального элемента.

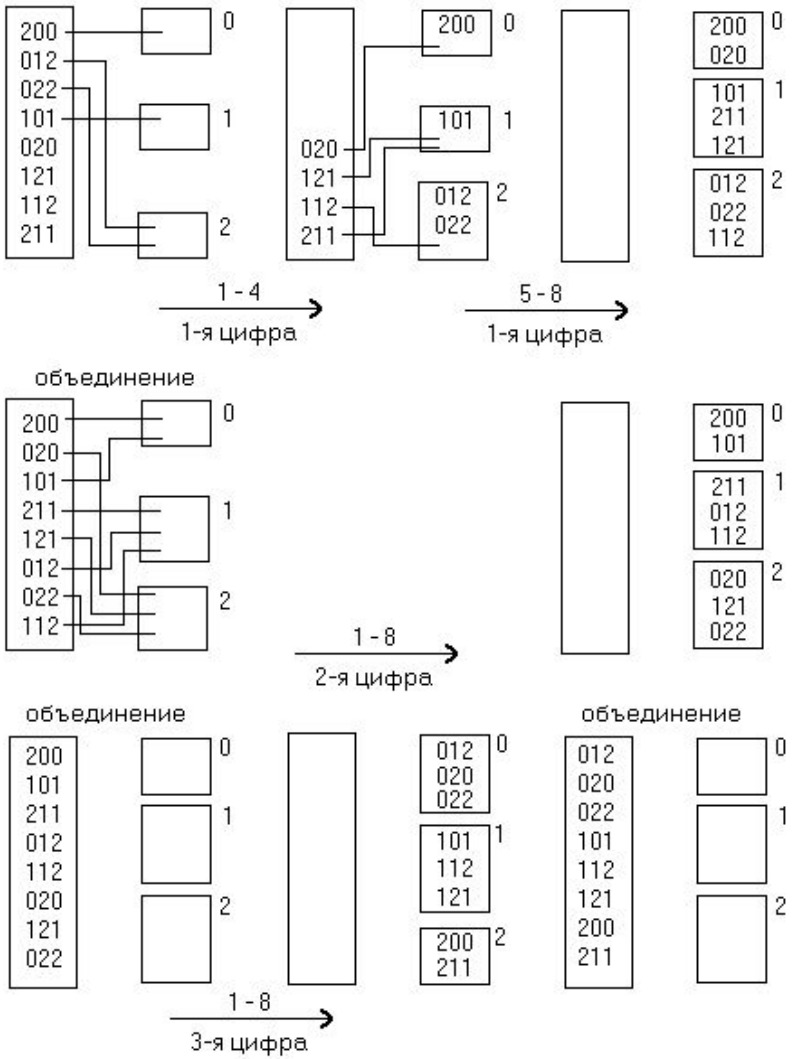
ЦИФРОВАЯ РАСПРЕДЕЛЯЮЩАЯ СОРТИРОВКА

Применяется для упорядочивания n-ричных чисел.

Идея: просматривая записи последовательно, распределяем их на группы с одинаковой последней цифрой. После этого группы снова объединяются в один массив в порядке возрастания последних цифр. Затем это же проделывается с получившимся массивом для второй с конца цифры, и так столько раз, какова разрядность чисел. При этом очередной проход не нарушает упорядоченности по уже обработанным разрядам.

Данный метод более сложный, но в месте с тем более экономичный.

На рисунке показано, как осуществляется такая сортировка для трехзначных чисел.



Сортировка последовательных файлов

Простое слияние

Слияние означает объединение двух (или более) упорядоченных последовательностей в одну упорядоченную последовательность при помощи циклического выбора элементов, доступных в данный момент. Слияние — намного более простая операция, чем сортировка; она используется в качестве вспомогательной в более сложном процессе последовательной сортировки.

Метод сортировки простым слиянием состоит в следующем:

1. Последовательность *a* разбивается на две половины *b* и *c.*

2. Последовательности *b* и *c* сливаются при помощи объединения отдельных элементов в упорядоченные пары.

3. Полученной последовательности присваивается имя a, и повторяются шаги 1 и 2; упорядоченные пары сливаются в упорядоченные четверки.

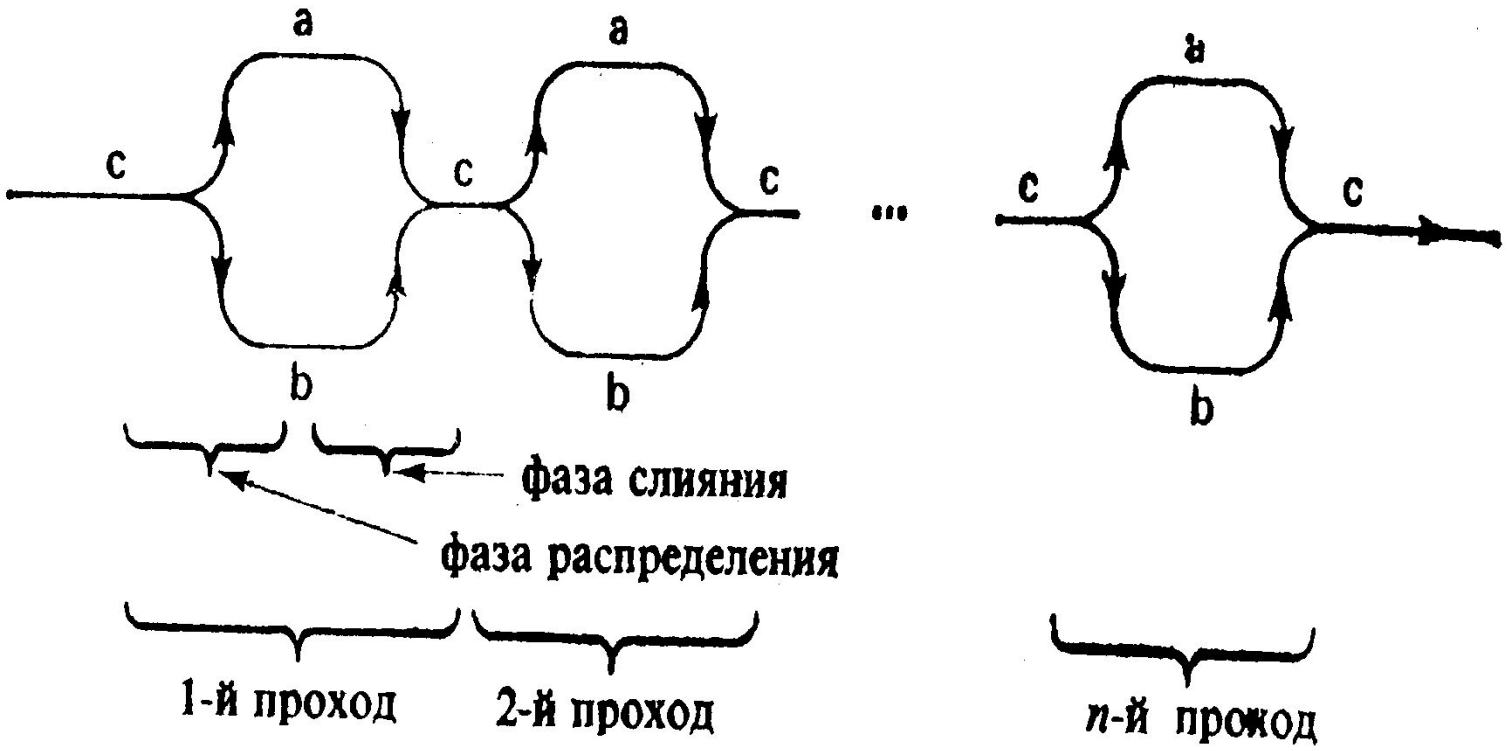
4. Предыдущие шаги повторяются: четверки сливаются в восьмерки, и весь процесс продолжается до тех пор, пока не будет упорядочена вся последовательность, ведь длины сливаемых последовательностей каждый раз удваиваются.

В качестве примера рассмотрим последовательность:
44 55 12 42 94 18 06 67
На первом шаге разбиение дает последовательности:
44 55 12 42
94 18 06 67
Слияние отдельных компонент (которые являются упорядоченными последовательностями длины 1) в упорядоченные пары дает:
44 94 / 18 55 / 06 12 / 42 67
Новое разбиение пополам и слияние упорядоченных пар дают:
06 12 44 94 / 18 42 55 67
Третье разбиение и слияние приводят к нужному результату:
06 12 18 42 44 55 67 94

Естественное слияние

Исходная последовательность элементов задана в виде файла *c*, который в конце работы должен содержать результат сортировки. Каждый проход состоит из фазы распределения, которая распределяет упорядоченные подпоследовательности поровну из *c* в *a* и *b*, и фазы слияния, которая сливает упорядоченные подпоследовательности из *a* и *b* в *c*.

Этот процесс показан на рисунке:



Сбалансированное многопутевое слияние

Имеется равное число входных и выходных файлов, в которые поочередно распределяются следующие одна за другой упорядоченные подпоследовательности, т.е. входные и выходные файлы меняются ролями после каждого отдельного прохода.

Достоинство: уменьшение числа проходов.

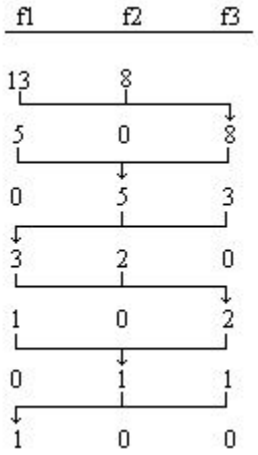
Многофазная сортировка

Этот метод был изобретён Р.Л. Гилстадом.

В каждый момент элементы сливаются из нескольких файлов в один. Как только один из входных файлов окажется исчерпанным, он становится выходным файлом для слияния из того файла, который ещё не исчерпан, и из тех, которые до этого были входными.

Рассмотрим пример, изображенный на рисунке. Вначале два входных файла f1, f2 содержат соответственно 13 и 8 упорядоченных подпоследовательностей. На первом «проходе» 8 подпоследовательностей сливаются с f1 и f2 в f3, на втором «проходе» оставшиеся 5 подпоследовательностей сливаются с f3 и f1 в f2 и т. д. В конце работы в файле f1 содержится отсортированная последовательность.

Достоинство: более эффективен, чем сбалансированная сортировка.



Анализ алгоритмов

Основное требование к методам сортировки массивов – экономное использование памяти. Это означает, что переупорядочивание элементов нужно выполнять на том же месте. Таки образом, выбирая метод сортировки, руководствуются критерием эффективности. Удобная мера эффективности получается при подсчете числа необходимых сравнений и (или) пересылок элементов.

Рассмотрим наглядный пример исследования алгоритма на основе сортировки выбором:

Пусть нам дан одномерный массив с цифрами **1 2 3 4 5**

a1 a2 a3 a4 a5

Рассмотрим несколько вариантов сортировок :

1. **Все элементы упорядочены** (лучший случай)

Сравниваем элементы с первым начиная со второго :

A2>A1 A2:=a2; 12345 шаг 1 действий 4

A3>a2 A3:=a3; 12345

A4>a3 A4:=a4; 12345

A5>a4 A5:=a5; 12345

Итого шаг 1 действий 4.

Сортировка не требуется .

2. Некоторые элементы находятся не на своём месте

1 3 4 5 2

A2>a1 A2:=a2; 13452 шаг 1 действий 3
A3>a2 A3:=A3; 13452
A4>a3 A4:=a4 13452

A5<a4 a5:=a4; 13425 шаг 2 действий 4
A4<a3 a4:=a3; 13245
A3<a2 A2:=a3; 12345
A2>a1 a2:=a2; 12345

Итого шагов 2, действий 7.

3. Все элементы находятся не на своём месте (худший случай)

5 4 3 2 1

A2<a1 a2:=a1; 45321 шаг 1 действий 1

A3<a2 a3:=a2; 43521 шаг 2 действий 2
A2<a1 a2:=a1; 34521

A4<a3 a3:=a4; 34251 шаг 3 действий 3
A3<a2 a2:=a3; 32451
A2<a1 a2:=a1; 23451

A5<a4 a4:=a5; 23415 шаг 4 действий 4
A4<a3 a3:=a4; 23145
A3<a2 a2:=a3; 21345
A2<a1 a2:=a1; 12345

Итого шагов 4, действий 10.

Статические структуры данных

Тема 9: Прямой доступ и хеширование

Сортировка справочных таблиц

1. если основная таблица расположена на внешней памяти, то справочная таблица может быть размещена в оперативной памяти, и поиск ключа, таким образом, будет выполняться в оперативной памяти, что гораздо быстрее;

2. для одной основной таблицы могут быть построены несколько справочников, обеспечивающих использование в качестве ключа разных полей записи основной таблицы.

Хешированные таблицы и функции хеширования

Поскольку память является одним из самых дорогостоящих ресурсов вычислительной системы, из соображений ее экономии целесообразно назначать размер пространства записей равным размеру фактического множества записей или превосходящим его незначительно. В этом случае мы должны иметь некоторую функцию, обеспечивающую отображение точки из пространства ключей в точку в пространстве записей, т.е. преобразование ключа в адрес записи: r = H(k), где - r адрес записи, k - ключ.

Такая функция называется функцией хеширования

При попытке отображения точек из некоторого широкого пространства в узкое неизбежны ситуации, когда разные точки в исходном пространстве отобразятся в одну и ту же точку в целевом пространстве. Ситуация, при которой разные ключи отображаются в один и тот же адрес записи, называется коллизией или переполнением, а такие ключи называются синонимами. Коллизии - основная проблема для хешированных таблиц

К функции хеширования в общем случае предъявляются следующие требования:

- она должна обеспечивать равномерное распределение отображений фактических ключей по пространству записей;
- она должна порождать как можно меньше коллизий для данного фактического множества записей;
- она не должна отображать какую-либо связь между значениями ключей в связь между значениями адресов;
- она должна быть простой и быстрой для вычисления.

Простейшей функцией хеширования является **деление по модулю числового значения ключа на размер пространства записи.** Результат интерпретируется как адрес записи.

Функция середины квадрата. Значение ключа преобразуется в число, это число затем возводится в квадрат, из него выбираются несколько средних цифр и интерпретируются как адрес записи.

Функция свертки. Цифровое представление ключа разбивается на части, каждая из которых имеет длину, равную длине требуемого адреса. Над частями производятся какие-то арифметические или поразрядные логические операции, результат которых интерпретируется как адрес.

Функция преобразования системы счисления. Ключ, записанный как число в некоторой системе счисления P, интерпретируется как число в системе счисления Q>P. Обычно выбирают Q = P+1. Это число переводится из системы Q обратно в систему P, приводится к размеру пространства

Проблема коллизий в хешированных таблицах

- Удачно подобранная функция хеширования может минимизировать число коллизий, но не может гарантировать их полного отсутствия.
- Ниже мы рассмотрим методы разрешения проблемы коллизий в хешированных таблицах.

Контрольные вопросы

1. Какие наиболее распространенные операции логического уровня над статическими структурами?
2. Перечислите основные стратегии алгоритмов сортировки статических структур?
3. Какие из алгоритмов сортировки подходят под стратегию слияния?
4. Какие из алгоритмов сортировки реализуют под стратегию выборки?

Спасибо за внимание!