Министерство образования Российской Федерации
Белгородский Государственный Технологический Университет им. В.Г. Шухова

Курсовая работа по предмету Структуры Данных.

Поиск в недвоичных деревьях – В деревьях

Выполнил: Зубков М.В
Группа ПВ-22
Принял: Синюк В.Г.

Белгород 2007

Оглавление

Введение.

Каждый программист может не мало рассказать об алгоритмах сортировки и поиска. Эти задачи являются пожалуй самыми востребованными в алгоритмах. Не удивительно, что существует огромное множество методов и структур для их решения. Кроме того, подразделяют, в какой памяти находятся данные. Если в оперативной – минимизируют число обменов, сравнений. Если во внешней – число чтений и записей. Для поиска во внешней памяти существует не так много алгоритмов. Естественно существует линейный поиск, хеширование и механизм, называемый B-деревьями. Все остальное представляет собой модификации данных структур, подходов.

Методы поиска во внешней памяти на основе деревьев:

Базовым "древовидным" аппаратом для поиска данных во внешней памяти являются B-деревья. В основе этого механизма лежат следующие идеи. Во-первых, поскольку речь идет о структурах данных во внешней памяти, общее время доступа к которой определяется в основном не объемом последовательно расположенных данных, а временем подвода магнитных головок (см. введение в Часть 3), то выгодно получать за одно обращение к внешней памяти как можно больше информации, учитывая при этом необходимость экономного использования основной памяти. При сложившемся подходе к организации основной памяти в виде набора страниц равного размера естественно считать именно страницу единицей обмена с внешней памятью. Во-вторых, желательно обеспечить такую поисковую структуру во внешней памяти, при использовании которой поиск информации по любому ключу требует заранее известного числа обменов с внешней памятью.

Классические B-деревья:

Механизм классических B-деревьев был предложен в 1970 г. Бэйером и Маккрейтом. B-дерево порядка n представляет собой совокупность иерархически связанных страниц внешней памяти (каждая вершина дерева - страница), обладающая следующими свойствами:

1. Каждая страница содержит не более 2*n элементов (записей с ключом).
2. Каждая страница, кроме корневой, содержит не менее n элементов.
3. Если внутренняя (не листовая) вершина B-дерева содержит m ключей, то у нее имеется m+1 страниц-потомков.
4. Все листовые страницы находятся на одном уровне.

Пример B-дерева степени 2 глубины 3 приведен на рисунке 1.

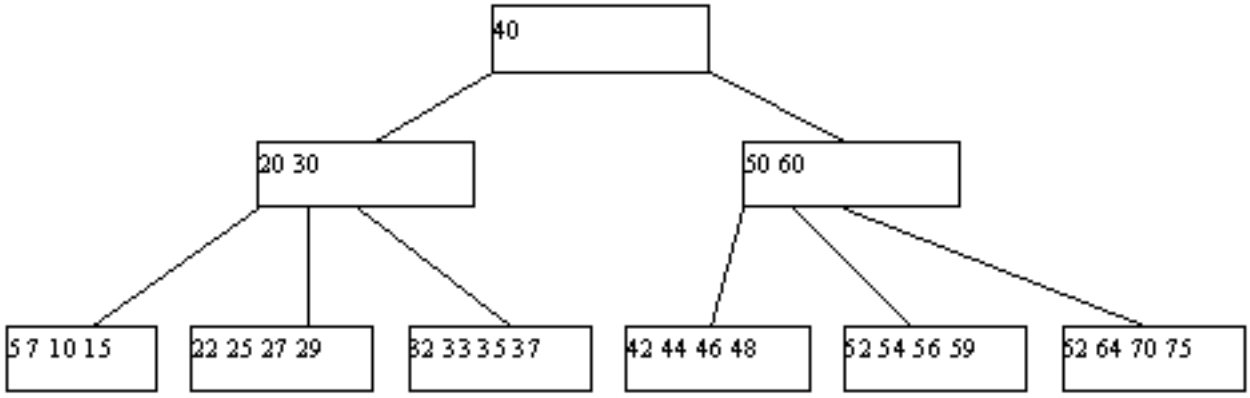


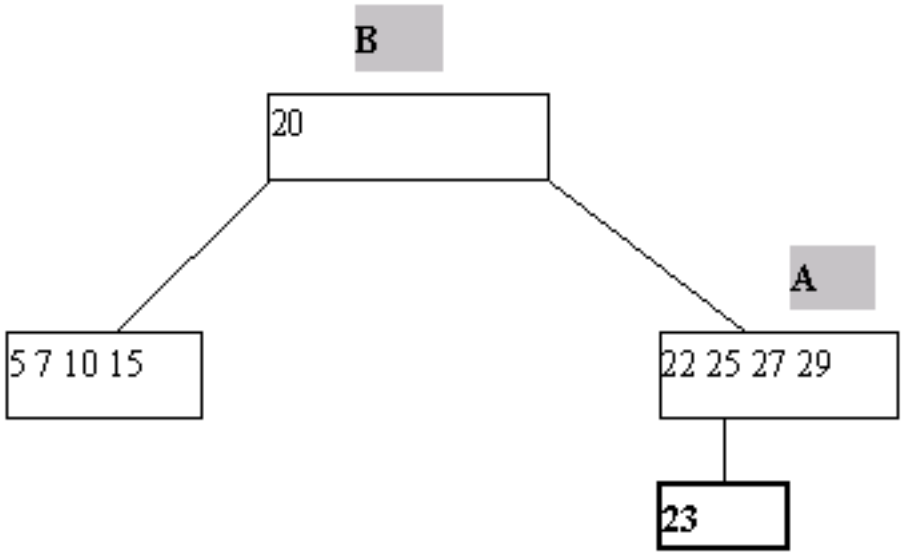
Рис. 1. Классическое B-дерево порядка 2

Поиск в B-дереве производится очевидным образом. Предположим, что происходит поиск ключа K. В основную память считывается корневая страница B-дерева. Предположим, что она содержит ключи k1, k2, ..., km и ссылки на страницы p0, p1, ..., pm. В ней последовательно (или с помощью какого-либо другого метода поиска в основной памяти) ищется ключ K. Если он обнаруживается, поиск завершен. Иначе возможны три ситуации:

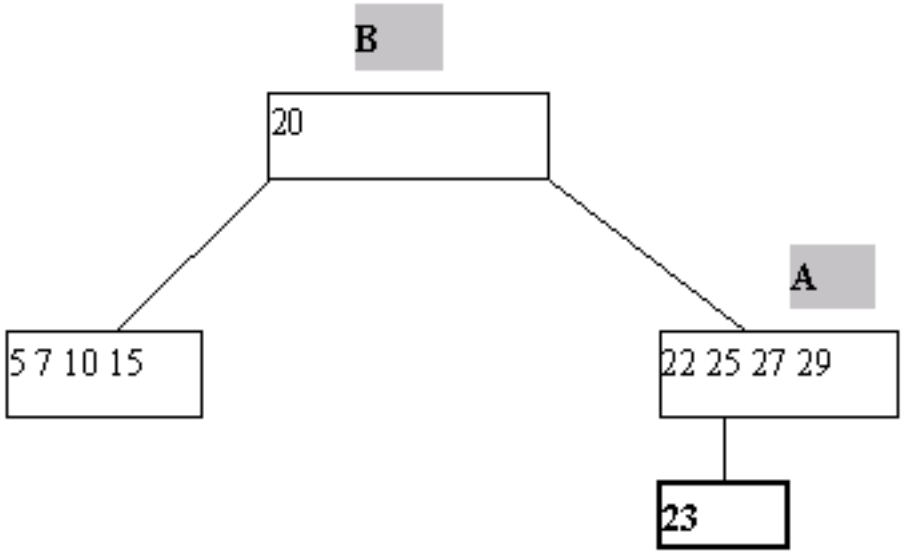
1. Если в считанной странице обнаруживается пара ключей ki и k(i+1) такая, что ki < K < k(i+1), то поиск продолжается на странице pi.
2. Если обнаруживается, что K > km, то поиск продолжается на странице pm.
3. Если обнаруживается, что K < k1, то поиск продолжается на странице p0.

Для внутренних страниц поиск продолжается аналогичным образом, пока либо не будет найден ключ K, либо мы не дойдем до листовой страницы. Если ключ не находится и в листовой странице, значит ключ K в B-дереве отсутствует.

Включение нового ключа K в B-дерево выполняется следующим образом. По описанным раньше правилам производится поиск ключа K. Поскольку этот ключ в дереве отсутствует, найти его не удастся, и поиск закончится в некоторой листовой странице A. Далее возможны два случая. Если A содержит менее 2*n ключей, то ключ K просто помещается на свое место, определяемое порядком сортировки ключей в странице A. Если же страница A уже заполнена, то работает процедура расщепления. Заводится новая страница C. Ключи из страницы A (берутся 2*n-1 ключей) + ключ K поровну распределяются между A и C, а средний ключ вместе со ссылкой на страницу C переносится в непосредственно родительскую страницу B. Конечно, страница B может оказаться переполненной, рекурсивно сработает процедура расщепления и т.д., вообще говоря, до корня дерева. Если расщепляется корень, то образуется новая корневая вершина, и высота дерева увеличивается на единицу. Одношаговое включение ключа с расщеплением страницы показано на рисунке 2.



(a) Попытка вставить ключ 23 в уже заполненную страницу

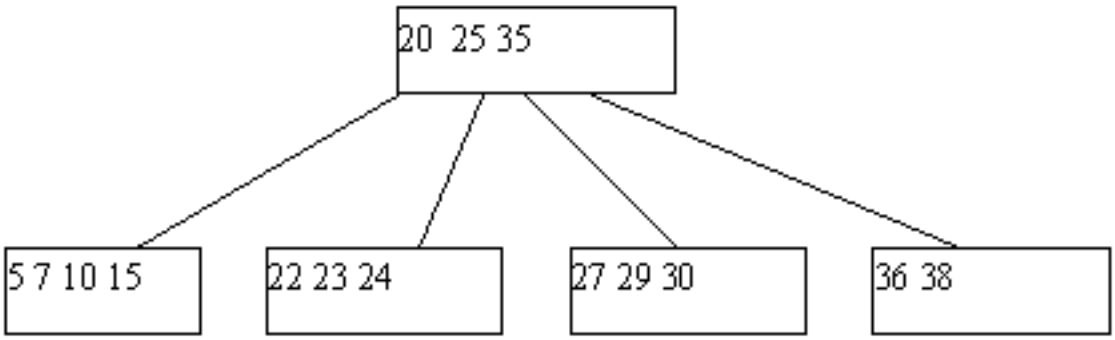


(b) Выполнение включения ключа 22 путем расщепления страницы A

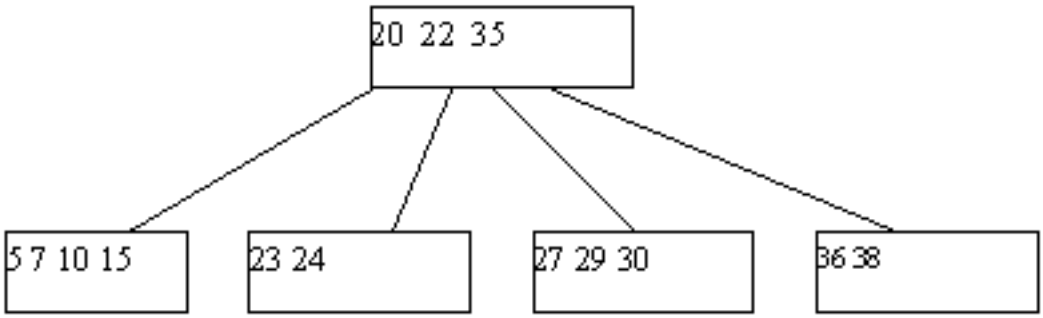
Рис. 2. Пример включения ключа в B-дерево

Процедура исключения ключа из классического B-дерева более сложна. Приходится различать два случая - удаление ключа из листовой страницы и удаления ключа из внутренней страницы B-дерева. В первом случае удаление производится

просто: ключ просто исключается из списка ключей. При удалении ключа во втором случае для сохранения корректной структуры B-дерева его необходимо заменить на минимальный ключ листовой страницы, к которой ведет последовательность ссылок, начиная от правой ссылки от ключа K (это минимальный содержащийся в дереве ключ, значение которого больше значения K). Тем самым, этот ключ будет изъят из листовой страницы (рисунок 3).



(a) Начальный вид B-дерева



(b) B-дерево после удаления ключа 25

Рис. 3. Пример исключения ключа из B-дерева

Поскольку в любом случае в одной из листовых страниц число ключей уменьшается на единицу, может нарушиться то требование, что любая, кроме корневой, страница B-дерева должна содержать не меньше n ключей. Если это действительно случается, начинает работать процедура переливания ключей. Берется одна из соседних листовых страниц (с общей страницей-предком); ключи, содержащиеся в этих страницах, а также средний ключ страницы-предка поровну распределяются между листовыми страницами, и новый средний ключ заменяет тот, который был заимствован у страницы-предка (рисунок 4).

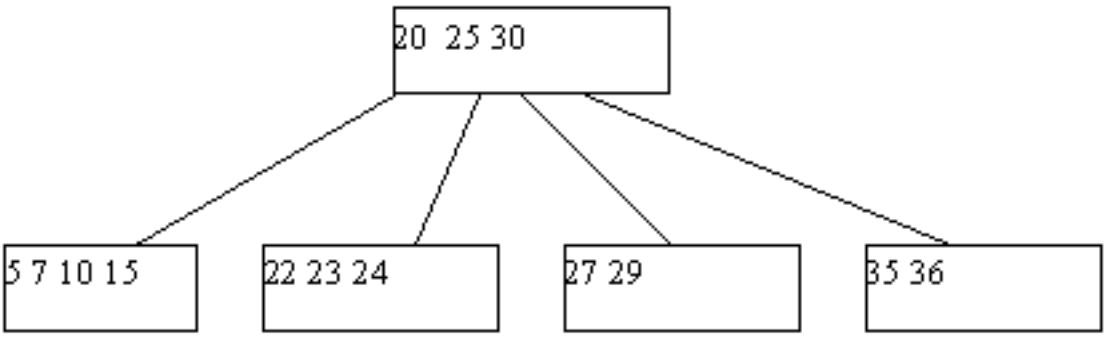
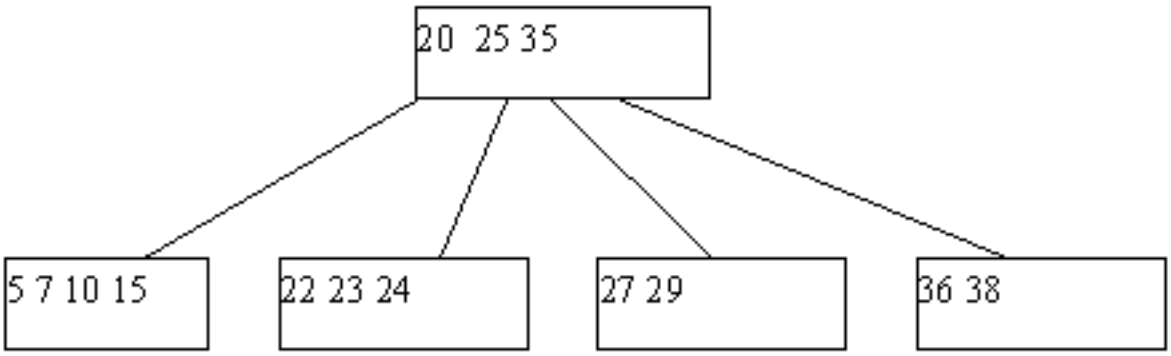
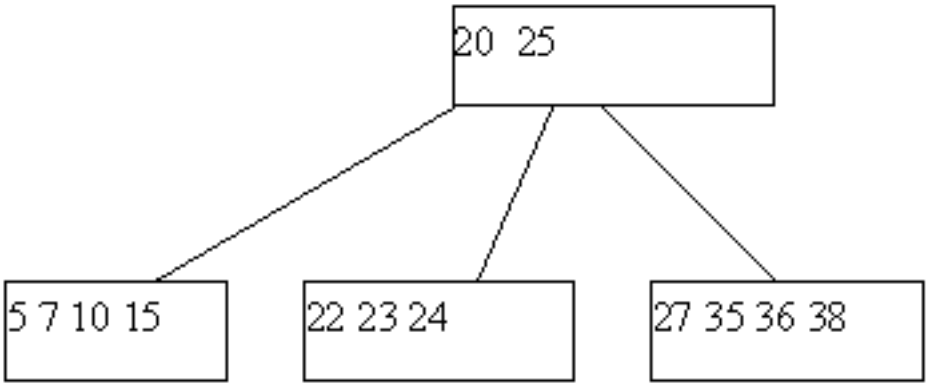


Рис. 4. Результат удаления ключа 38 из B-дерева с рисунка 5.3

Может оказаться, что ни одна из соседних страниц непригодна для переливания, поскольку содержат по n ключей. Тогда выполняется процедура слияния соседних листовых страниц. К 2*n-1 ключам соседних листовых страниц добавляется средний ключ из страницы-предка (из страницы-предка он изымается), и все эти ключи формируют новое содержимое исходной листовой страницы. Поскольку в странице-предке число ключей уменьшилось на единицу, может оказаться, что число элементов в ней стало меньше n, и тогда на этом уровне выполняется процедура переливания, а возможно, и слияния. Так может продолжаться до внутренних страниц, находящихся непосредственно под корнем B-дерева. Если таких страниц всего две, и они сливаются, то единственная общая страница образует новый корень. Высота дерева уменьшается на единицу, но по-прежнему длина пути до любого листа одна и та же. Пример удаления ключа со слиянием листовых страниц показан на рисунке 5.



(a) Начальный вид B-дерева



(b) B-дерево после удаления ключа 29

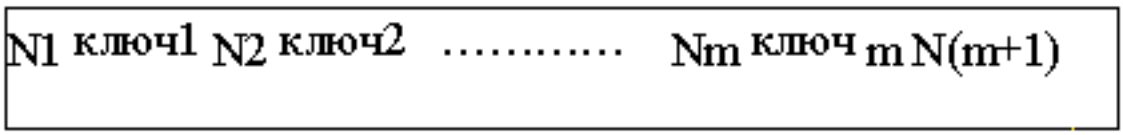
Рис. 5. Пример удаления ключа из B-дерева со слиянием листовых страниц

B+-деревья:

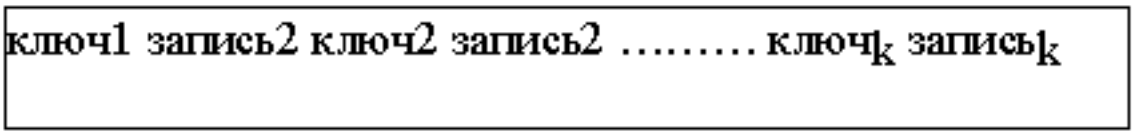
Схема организации классических B-деревьев проста и элегантна, но не очень хороша для практического использования. Прежде всего это связано с тем, что в большинстве практических применений необходимо хранить во внешней памяти не только ключи, но и записи. Поскольку в B-дереве элементы располагаются и во внутренних, и в листовых страницах, а размер записи может быть достаточно большим, внутренние страницы не могут содержать слишком много элементов, по причине дерево может быть довольно глубоким. Поэтому для доступа к ключам и записям, находящимся на нижних уровнях дерева, может потребоваться много обменов с внешней памятью. Во-вторых, на практике часто встречается потребность хранения и поиска ключей и записей переменного размера. Поэтому тот критерий, что в каждой странице B-дерева содержится не меньше n и не больше 2*n ключей, становится неприменимым.

Широкое практическое применение получила модификация механизма B-деревьев, которую принято называть B+-деревьями. Эти деревья похожи на обычные B-деревья. Они тоже сильно ветвистые, и длина пути от корня к любой листовой странице одна и та же. Но структура внутренних и листовых страниц различна. Внутренние страницы устроены так же, как у B-дерева, но в них хранятся только ключи (без записей) и ссылки на страницы-потомки. В листовых страницах хранятся все ключи, содержащиеся в дереве, вместе с записями, причем этот список упорядочен по возрастанию значения ключа (рисунок 6).

Поиск ключа всегда доходит до листовой страницы. Аналогично операции включения и исключения тоже начинаются с листовой страницы. Для применения переливания, расщепления и слияния используются критерии, основанные на уровне заполненности соответствующей страницы. Для более экономного и сбалансированного использования внешней памяти при реализации B+-деревьев иногда используют технику слияния трех соседних страниц в две и расщепления двух соседних страниц в три. Хотя B+-деревья хранят избыточную информацию (один ключ может храниться в двух страницах), они, очевидно, обладают меньшей глубиной, чем классические B-деревья, а для поиска любого ключа требуется одно и то же число обменов с внешней памятью.



(a) Структура внутренней страницы B+-дерева



(b) Структура листовой страницы B+-дерева
Рис. 6. Структуры страниц B+-дерева

Дополнительной полезной оптимизацией B+-деревьев является связывание листовых страниц в одно- или двунаправленный список. Это позволяет просматривать списки записей для заданного диапазона значений ключей с лишь одним прохождением дерева от корня к листу.

Разновидности B+-деревьев для организации индексов в базах данных:

B+-деревья наиболее интенсивно используются для организации индексов в базах данных. В основном это определяется двумя свойствами этих деревьев: предсказуемостью числа обменов с внешней памятью для поиска любого ключа и тем, что это число обменов по причине сильной ветвистости деревьев не слишком велико при индексировании даже очень больших таблиц.

При использовании B+-деревьев для организации индексов каждая запись содержит упорядоченный список идентификаторов строк таблицы, включающих соответствующее значение ключа. Дополнительную сложность вызывает возможность организации индексов по нескольким столбцам таблицы (так называемых "составных" индексов). В этом случае в B+-дереве может появиться очень много избыточной информации по причине наличия в разных составных ключах общих подключей. Имеется ряд технических приемов сжатия индексов с составными ключами, улучшающих использование внешней памяти, но, естественно, замедляющих выполнение операций включения и исключения.

R-деревья и их использование для организации индексов в пространственных базах данных:

Коротко рассмотрим еще одно расширение механизма B-деревьев, используемое главным образом для организации индексов в пространственных базах данных, - R-деревья. Подобно B+-деревьям, R-дерево представляет собой ветвистую сбалансированную древовидную структуру с разной организацией внутренних и листовых страниц.

Но информация, хранящаяся в R-дереве несколько отличается от той, которая содержится в B-деревьях. В дополнение к находящимся в листовых страницах идентификаторам пространственных объектов, в R-деревьях хранится информация о границах индексируемого объекта. В случае двумерного пространства сохраняются горизонтальные и вертикальные координаты нижнего левого и верхнего правого углов наименьшего прямоугольника, содержащего индексируемый объект. Пример простого R-дерева, содержащего информацию о шести пространственных объектах, приведен на рисунке 7.

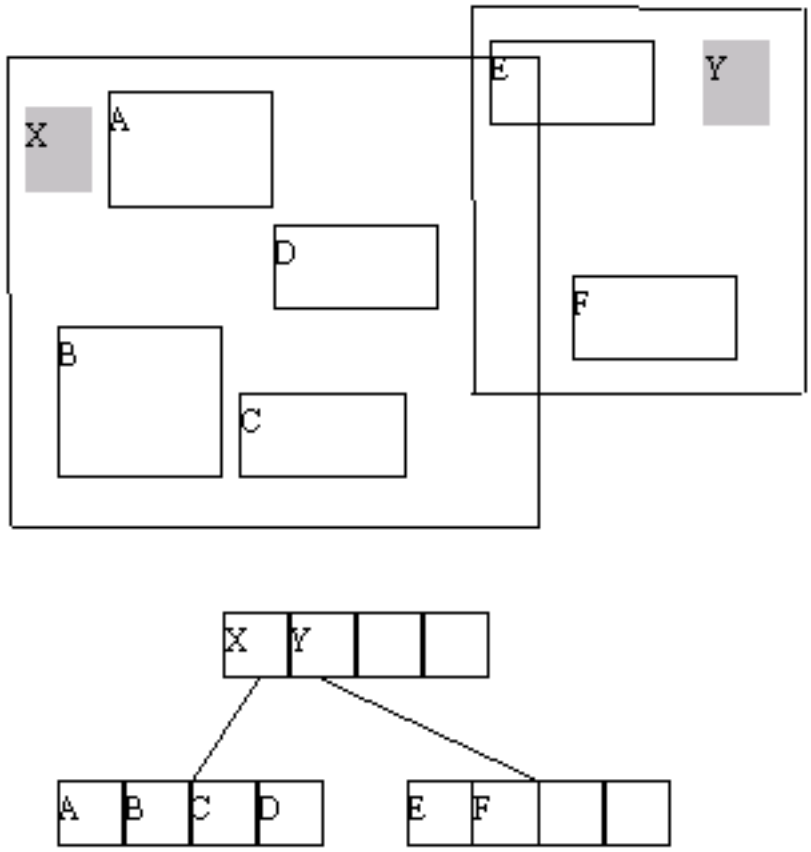


Рис. 7. Простое R-дерево для представления шести пространственных объектов

Временная сложность.

Поиск в B-дереве - это прохождение от корня к листу в соответствии с заданным значением ключа. Поскольку деревья сильно ветвистые и сбалансированные, то для выполнения поиска по любому значению ключа потребуется небольшое число обменов с внешней памятью. Более точно, в сбалансированном дереве, где длины всех путей от корня к листу одни и те же, если во внутренней странице помещается n ключей, то при хранении m записей требуется дерево глубиной $\log_n(m)$, где $\log_n$ вычисляет логарифм по основанию n. Если n достаточно велико (обычный случай), то глубина дерева невелика, и производится быстрый поиск. Наибольшее число обменов с внешней памятью происходит тогда, когда ключ находится в листовой странице. Т.е происходит $\log_n(m)$ операций считывания. Таким образом, наихудшей ф-цией временной сложности для B-дерева является $O(\log_n(m))$. Эта же временная сложность и для B+-дерева, но для это уже не худший случай а постоянная функция временной сложности. Что касается среднего значения, то это уже зависит от распределения ключей. Допуская, что на каждой странице находится k ключей, получаем

$$O=(k*1 + 2*k*(k+1) +3*k*(k+1)*(k+1)+\ldots..+ \log_n(m)*k*(k-1)^{(\log n(m)-1)})/m$$

Данные экспериментальных измерений.

Таблица 1.1 Максимальное количество операций чтения страницы для поиска всех ключей.

Кол-во ключей:	**1000**	**2000**	**4000**	**8000**	**16000**
B-дерево порядка 2	6000	14000	28000	64000	144000
B-дерево порядка 4	4000	10000	20000	48000	96000
B-дерево порядка 8	3000	8000	16000	32000	80000

Таблица 1.2 Максимальное количество операций чтения страницы для поиска ключа.

Кол-во ключей:	**1000**	**2000**	**4000**	**8000**	**16000**
B-дерево порядка 2	6	7	7	8	9
B-дерево порядка 4	4	5	5	6	6
B-дерево порядка 8	3	4	4	4	5

Таблица 2.1 Среднее количество операций чтения страницы для поиска всех ключей.

Кол-во ключей:	**1000**	**2000**	**4000**	**8000**	**16000**
B-дерево порядка 2	2763	13009	26011	60012	136014
B-дерево порядка 4	2812	9505	19006	46007	92007
B-дерево порядка 8	2878	7755	15505	31006	78007

Таблица 2.2 Среднее количество операций чтения страницы для поиска ключа (округлено).

Кол-во ключей:	**1000**	**2000**	**4000**	**8000**	**16000**
B-дерево порядка 2	2,7	6,5	6,5	7,5	8,5
B-дерево порядка 4	2,8	4,8	4,8	5,8	5,8
B-дерево порядка 8	2,9	3,9	3,9	3,9	4,9

Таблица 3. Объем памяти, занимаемый деревом (в байтах).

Кол-во ключей:	**1000**	**2000**	**4000**	**8000**	**16000**
B-дерево порядка 2	21920	43920	87832	175832	351788
B-дерево порядка 4	18932	38008	75932	151932	303932
B-дерево порядка 8	17368	34868	69868	139728	279728

Диаграмма 1.2 Максимальное количество операций чтения

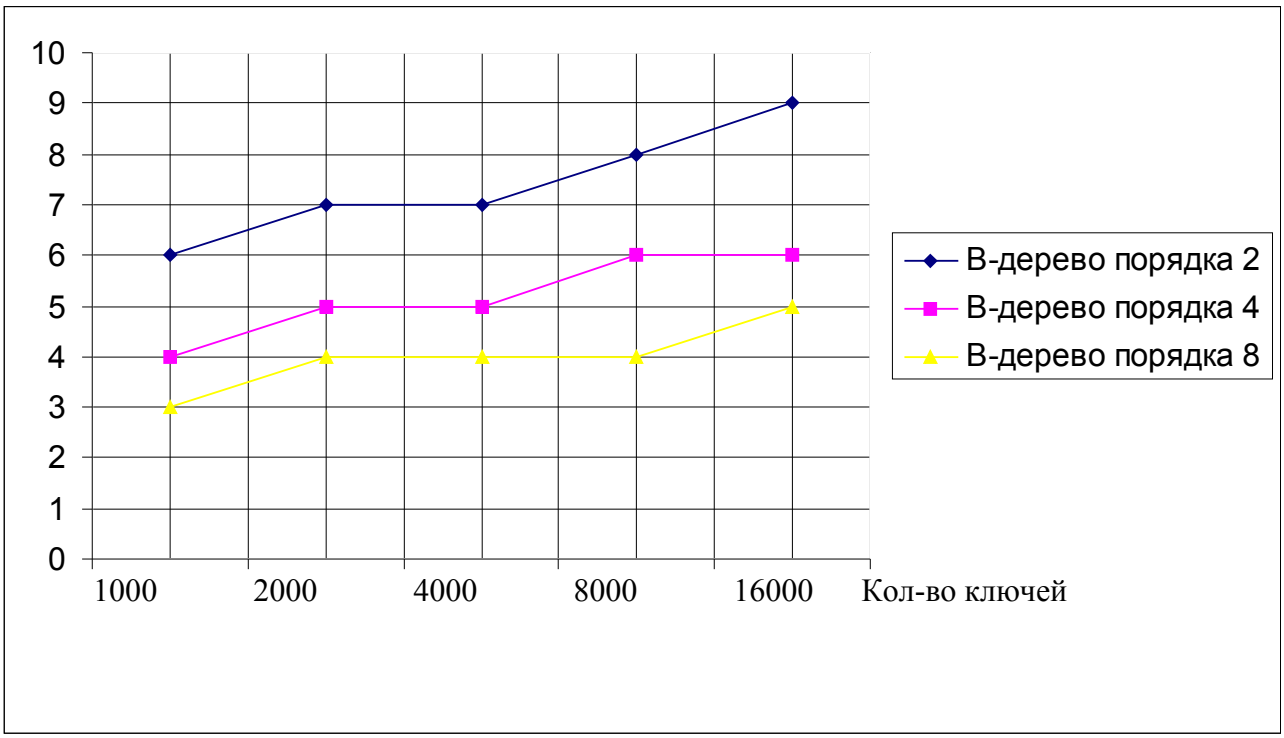
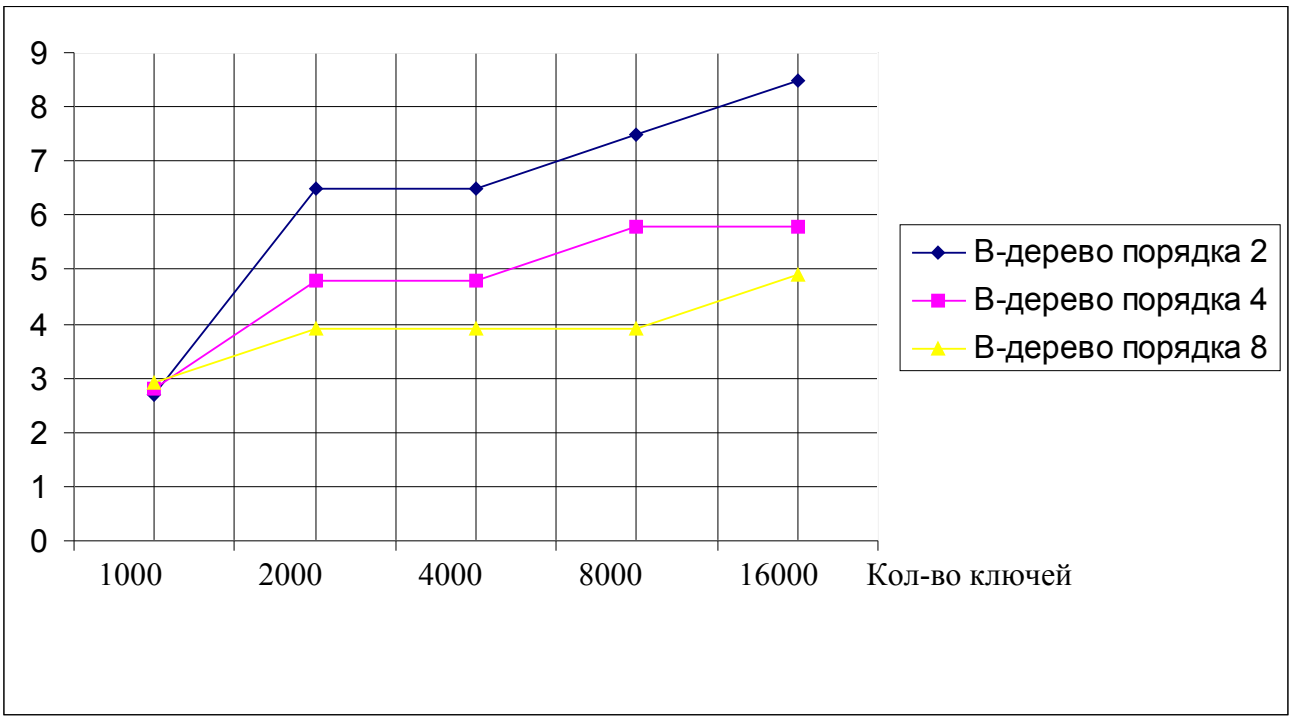
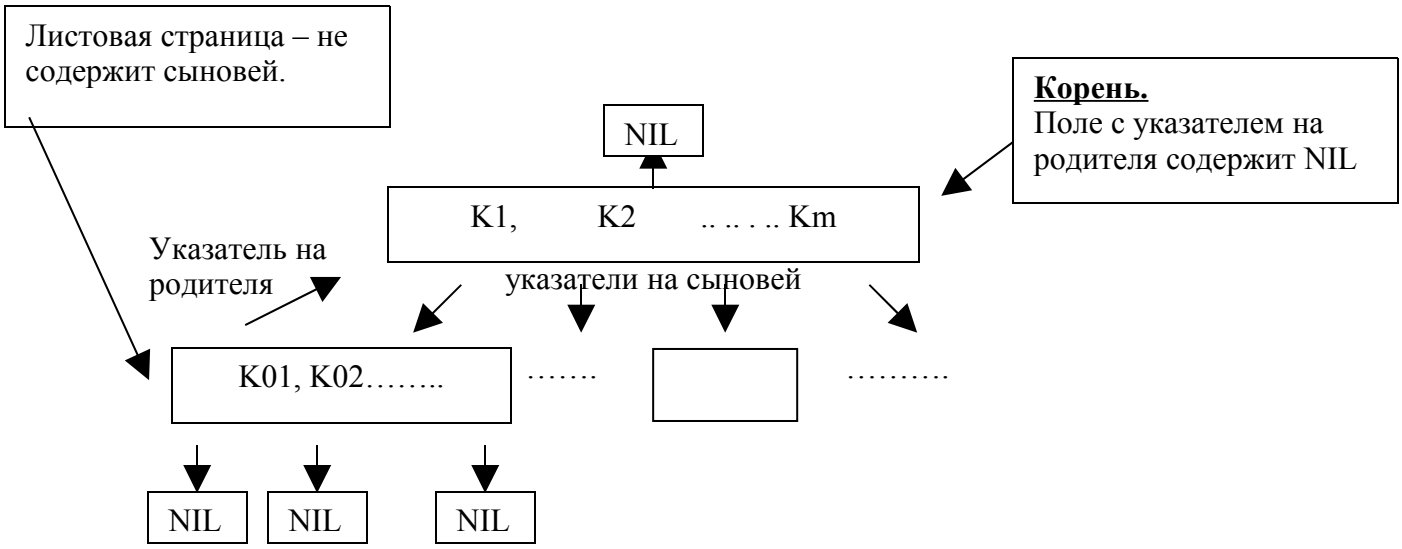


Диаграмма 2.2 Среднее количество операций чтения



Структура дерева в реализации.

Фактически, по структуре B-дерево представляет собой обычное дерево. Оно может содержать до 2n+1 указателей на сыновей. Кроме того, для облегчения перемещений по дереву, существуют указатели на родителей. Такая сильно связанная структура весьма эффективна для поиска, но и значительно неудобна для модификаций.



На физическом уровне дерево представляет собой последовательность участков файла, одного размера (размера страницы). Кроме того, в начале файла существует заголовок, содержащий порядок дерева и указатель на корень. В данной работе отсутствуют функции удаления элемента из B-дерева. Их реализация потребует сильного усложнения структуры. В случае слияния 2х страниц в одну, произойдет фрагментация файла. Потребуется модифицировать большое кол-во указателей. Еще один способ – связать свободные блоки в список и выбирать их в первую очередь оттуда. Однако производить дефрагментацию файла перед закрытием придется в любом случае.

Заключение

В данной курсовой работе была поставлена попытка реализовать базовое множество операций (инициализация, поиск, вставка, удаление) предназначенных для работы с B+ деревьями.

Данная работа не претендует на всеобъемлющее рассмотрение принципов реализации B+ деревьев. Естественно, программная реализация далека от совершенства, но выполнение данной работы позволило получить достаточно подробную информацию об основных методика ра боты и свойствах B деревьев.

Литература

1. Зубов В.С. Шевченко И.В. - Структуры и методы обработки данных. Практикум в среде Delphi – М. Информационно-издательский дом «Филинъ» 2004г. 304с.
2. Кнут Дональд – Искусство программирования. Том 3 – Сортировка и поиск : М. Издательский дом «Вильямс» 2000г. 832с.
3. Седжви Роберт – Фундаментальные Алгоритмы на С. Части 1 – 4 : Спб ООО «ДиаСофтЮП» 2003г. 672с.

Приложение