**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ**

**Кафедра дискретной математики и алгоритмики**

**Применение B-деревьев для эффективных операций в базах данных**

Курсовой проект

Каменко Яны Александровны

студентки 3 курса,

специальность «информатика»

Научный руководитель:

ассистент кафедры БМИ

Хадарович А.Юав

Мелеха Алексея Александровича

студента 3 курса,

специальность «информатика»

Научный руководитель:

Ассистент кафедры ДМА ФПМИ

Лукьянов Иван Денисович

# ГЛАВА 1. B-ДЕРЕВО

## Определение

B-дерево — сильноветвящееся сбалансированное дерево поиска, позволяющее проводить поиск, добавление и удаление элементов за . B-дерево с *n* узлами имеет высоту . Количество детей узлов может быть от нескольких до тысяч (обычно степень ветвления B-дерева определяется характеристиками устройства (дисков), на котором производится работа с деревом). B-деревья также могут использоваться для реализации многих операций над динамическими множествами за время .

*Сбалансированность дерева* — свойство дерева, при котором длина любых двух путей от корня до листьев различается не более, чем на единицу.

*Сильная ветвистость дерева* — свойство дерева, при котором любой узел может ссылаться на большое число узлов-потомков.

B-дерево было впервые предложено Р. Бэйером и Е. МакКрейтом в 1970 году.

## Структура

B-дерево является идеально сбалансированным, то есть глубина всех его листьев одинакова. B-дерево имеет параметр, называемый минимальной степенью B-дерева, который должен быть не меньше 2. В дальнейшем будем обозначать параметр минимальной степени как ***Tmin***.

Свойства:

* Каждый узел, кроме корня, содержит не менее ***Tmin – 1*** ключей
* Каждый узел, не являющийся листом, имеет по меньшей мере ***Tmin*** узлов-потомков
* Если дерево непустое, то корень содержит как минимум один ключ
* Каждый узел, кроме корня, содержит не более ***2Tmin - 1*** ключей
* Каждый узел, кроме корня и листьев, содержит не более ***2Tmin*** узлов-потомков
* Корень содержит от ***1*** до ***2Tmin - 1*** ключей при высоте большей 0
* Корень содержит от ***2*** до ***2Tmin - 1*** узлов-потомков при высоте большей 0
* Ключи в узлах упорядочены по неубыванию
* Каждый узел, кроме листьев, содержащий ключей, содержит ***n + 1*** узлов-потомков. При этом i-ый потомок содержит ключи из отрезка
* Все листья находятся на одинаковой глубине

### Структура узла:

**template <typename T>**

**struct** BNode

**bool** isLeaf **//** **является ли узел листом**

**size\_t** size **// количество ключей узла**

**T[]** keys **// массив узлов**

**unique\_ptr[]** childrens **// массив указателей на узлов-потомков**

### Структура класса B-дерева:

**template <typename T>**

**struct** BNode

**size\_t** tMin **//** **параметр минимальной степени**

**unique\_ptr<BNode<T>>** root **//** **указатель на корень**

## Применение

B-деревья используются для работы с дисками (в файловых системах) или иных энергонезависимых носителях информации с прямым доступом, а также в базах данных. Данное дерево оптимизирует количество операций чтения-записи с диском, при этом сохраняя асимптотику операций, в сравнении с другими деревьями.

## Структуры данных во внешней памяти

Кроме оперативной памяти, в компьютере используется внешний носитель, как правило, представляющий собой магнитные диски (или твердотельный накопитель). Хотя диски существенно дешевле оперативной памяти и имеют высокую емкость, они гораздо медленнее оперативной памяти из-за механического построения считывания.

Для того чтобы снизить время ожидания, связанное с механическим перемещением, при обращении к диску выполняется обращение одновременно сразу к нескольким элементам, хранящимся на диске. Информация разделяется на несколько страниц одинакового размера, которые хранятся последовательно друг за другом в пределах одного цилиндра (набора дорожек на дисках на одном расстоянии от центра), и каждая операция чтения или записи работает сразу с несколькими страницами. Для типичного диска размер страницы варьируется от ***2*** до ***16*** КБайт. После того, как головка установлена на нужную дорожку, а диск поворачивается так, что головка становится на начало интересующей нас страницы, чтение и запись становятся полностью электронными процессами, не зависящими от поворота диска, и диск может быстро читать или писать крупные объёмы данных.

В типичном приложении с B-деревом, объём хранимой информации так велик, что вся она просто не может храниться в основной памяти единовременно. Алгоритмы B-дерева копируют выбранные страницы с диска в основную память по мере надобности и записывает обратно на диск страницы, которые были изменены. Алгоритмы B-дерева хранят лишь определённое количество страниц в основной памяти в любой момент времени; таким образом, объём основной памяти не ограничивает размер B-деревьев, которые можно создавать.

Система в состоянии поддерживать в процессе работы в оперативной памяти только ограниченное количество страниц. Мы будем считать, что страницы, которые более не используются, удаляются из оперативной памяти системой; наши алгоритмы работы с В-деревьями не будут заниматься этим самостоятельно. Поскольку в большинстве систем время выполнения алгоритма, работающего с В-деревьями, зависит в первую очередь от количества выполняемых операций чтения/записи с диском, желательно минимизировать их количество и за один раз считывать и записывать как можно больше информации. Таким образом, размер узла В-дерева обычно соответствует дисковой странице. Количество потомков узла В-дерева, таким образом, ограничивается размером дисковой страницы. Для больших В-деревьев, хранящихся на диске, степень ветвления обычно находится между ***50*** и ***2000***, в зависимости от размера ключа относительно размера страницы. Большая степень ветвления резко снижает как высоту дерева, так и количество обращений к диску для поиска ключа. Например, если есть миллиард ключей, и ***Tmin=1001***, то поиск ключа займёт две дисковые операции.

## Высота дерева

Количество обращений к диску напрямую зависит от высоты дерева. Оценим высоту дерева в наихудшем и наилучшем случае:

**Теорема**

Пусть,

— высота дерева  
— количество узлов в дереве   
— параметр минимальной степени (каждый узел может иметь не более ключей)

Тогда

▹

Корень B-дерева содержит как минимум 1 ключ, остальные узлы — Tmin – 1 ключей. На каждом уровне дерева содержится . Таким образом количество ключей удовлетворяет неравенству:

Из этого следует

Аналогично для .

## Основные операции

### Добавление ключа

Совершая проход от корня к листьям, пытаемся найти лист, в который добавим новый ключ. Если нашли какой-то незаполненный лист — добавляем ключ в него. Иначе разбиваем узел на два новых (первые Tmin – 1 ключей отойдут к первому узлу, последние Tmin – 1 отойдут ко второму узлу, средний элемент отойдет к родительскому узлу, который является разделительным узлом двух новых). Подробнее операцию разбиения рассмотрим дальше. Если родительский узел заполнен — повторим алгоритм, пока не дойдем до корня. Корень разбивается на два узла, и высота дерева увеличивается на 1.

### Поиск ключа

Если ключ находится в текущем узле, возвращаем его, иначе определяем интервал поиска (бинарным поиском) и переходим к соответствующему потомку. Интервал поиска определяется промежутком двух последовательных ключей, между которыми находится желаемый ключ. Алгоритм продолжается пока не найдется ключ или не дошли до листа.

### Удаление ключа

Процесс удаления ключа является самым трудоемким в данной структуре. Для начала проделаем алгоритм поиска ключа, и, в случае нахождения, рассмотрим несколько вариантов:

* Текущий узел является листом — если количество ключей в узле не меньше чем Tmin — просто удалим ключ из узла. Легко заметить, что структура дерева, которая была обговорена вначале, не нарушится. Если количество ключей не более чем Tmin – 1, то рассмотрим соседние узлы с данным через его предка. Если оба этих узла содержат Tmin – 1 ключей, проведем слияние листьев (эта операция будет описана дальше), иначе произведем перемещение ключа из одного из соседних узлов (эта операция будет описана дальше). Структура дерева не нарушится, так как суммарное количество ключей будет равно 2(Tmin – 1).
* Текущий узел не является корнем и листом. Имеется внутренний узел xx и ключ, который нужно удалить, k. Если дочерний узел, предшествующий ключу k, содержит больше Tmin−1 ключа, то находим k1 – предшественника k в поддереве этого узла. Удаляем его. Заменяем k в исходном узле на k1. Проделываем аналогичную работу, если дочерний узел, следующий за ключом k, имеет больше Tmin−1 ключа. Если оба (следующий и предшествующий дочерние узлы) имеют по Tmin−1 ключу, то [объединяем](https://neerc.ifmo.ru/wiki/index.php?title=B-%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE#.D0.A1.D0.BB.D0.B8.D1.8F.D0.BD.D0.B8.D0.B5) этих детей, [переносим](https://neerc.ifmo.ru/wiki/index.php?title=B-%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE#.D0.9F.D0.B5.D1.80.D0.B5.D0.BC.D0.B5.D1.89.D0.B5.D0.BD.D0.B8.D0.B5_.D0.BA.D0.BB.D1.8E.D1.87.D0.B0) в них k, а далее удаляем k из нового узла.
* Если [сливаются](https://neerc.ifmo.ru/wiki/index.php?title=B-%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE#.D0.A1.D0.BB.D0.B8.D1.8F.D0.BD.D0.B8.D0.B5) 2 последних потомка корня – то они становятся корнем, а предыдущий корень освобождается.

### Разбиение узла

Данная операция выполняется для разбиения заполненных узлов. Процесс заключается в разбиении узла на два новых. Первому узлу отходят первые Tmin−1 ключей, а второму последние Tmin−1, средняя вершина становится разделяющей вершиной. Стоит отметить, что высота дерева может быть увеличена только после операции разбиения в корне.

### Перемещение ключа

Если выбранное для нисходящего прохода поддерево содержит минимальное количество ключей Tmin−1, и предшествующие и следующие узлы-братья имеют по меньшей мере Tmin ключей, то ключ перемещается в выбранный узел. При выборе перемещающегося ключа, у левого брата будем брать больший элемент, у правого меньший элемент.

### Слияние узлов

Для произвольных же слияний потребуется приведение сливаемых деревьев к одной степени и высоте. Для этого откладывается ключ из родительского узла x, который разделяет ключи на два сливаемых узла, в то время средний ключ перемещается в слитый узел. Ссылки на слитые дочерние узлы заменяются ссылкой на новый узел. Так как алгоритм гарантирует, что узел, в который будет совершаться спуск, содержит по меньшей мере Tmin ключей вместо требуемых условиями B-дерева Tmin−1 ключей, родительский узел x содержит достаточное количество ключей, чтобы выделить ключ для слияния. Это условие может быть нарушено, только в том случае, если два ребенка корня сливаются, так как поиск начинается с этого узла. По условиям B-дерева у корня должен быть как минимум один ключ, если дерево не пусто. При слиянии двух последних детей корня последний ключ перемещается во вновь возникшего единственного ребёнка, что приводит к пустому корневому узлу в не пустом дереве. В этом случае пустой узел корня удаляется и заменяется на единственного ребенка.

### Сложность

|  |  |  |
| --- | --- | --- |
| **Алгоритм** | **Обращений к диску** | **Процессорное время** |
| Добавление ключа |  |  |
| Удаление ключа |  |  |
| Поиск ключа |  |  |
| Разбиение узла  Слияние узлов  Перемещение ключа |  |  |