Санкт-Петербургский политехнический университет Петра великого

Институт машиностроения, материалов и транспорта

Высшая школа автоматизации и робототехники

Курсовой проект

по дисциплине «Объектно-ориентированное программирование»

«В+-дерево»

Выполнил

студент гр. 3331506/90401 Яковлев Д.Э.

Работу принял Ананьевский М.С.

Санкт-Петербург

2022

Оглавление

[**Введение** 3](#_Toc104049927)

[**Описание алгоритма** 3](#_Toc104049928)

[**Описание реализация алгоритма** 4](#_Toc104049929)

[**Поиск** 4](#_Toc104049930)

[**Добавление** 4](#_Toc104049931)

[**Удаление** 5](#_Toc104049932)

[**Исследование алгоритма** 6](#_Toc104049933)

[**Область применения** 8](#_Toc104049934)

[**Заключение** 10](#_Toc104049935)

[**Список литературы** 11](#_Toc104049936)

[**Приложение** 12](#_Toc104049937)

**Введение**

B+ дерево — это оптимизация B-дерева, сбалансированное n-арное дерево поиска с переменным, но зачастую большим количеством потомков в узле, где значения хранятся только в листьях, а в узлах копии этих значений. В этой оптимизации больше ключей умещается в узел, что увеличивает степень ветвления. Пример B+-дерева представлен на рисунке 1.

Изначально структура предназначалась для хранения данных в целях эффективного поиска в блочно-ориентированной среде хранения — в частности, для файловых систем; применение связано с тем, что в отличие от бинарных деревьев поиска, B⁺-деревья имеют очень высокий коэффициент ветвления (число указателей из родительского узла на дочерние — обычно порядка 100 или более), что снижает количество операций ввода-вывода, требующих поиска элемента в дереве.

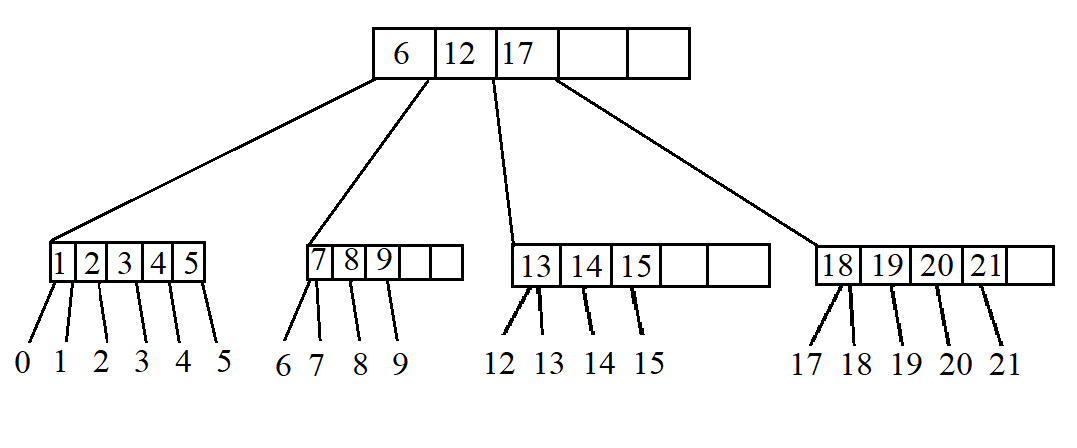


Рисунок 1 – Пример B+-дерева

**Описание алгоритма**

B+-дерево состоит из корня, внутренних узлов и листьев.

Имеются следующие правила:

⦁ Ключи содержат только листья. Узлы содержат копии ключей.

⦁ В каждом узле содержатся минимум (t-1) копий и t указателей. Максимум копий в узле – (2t-1), указателей – 2t. Корень же содержит от 1 до (2t-1) копий и от 2 до 2t указателей. (Указатели – ссылки на узлов-потомков/листья). Все копии и указатели упорядочены по возрастанию.

Здесь t — параметр дерева определяющий количество копий и указателей, иногда именуемый b-фактором, коэффициентом ветвления, не меньший 2 (и обычно принимающий значения от 50 до 2000. Поэтому B+-дерево сильноветвистое).

⦁ У листьев нет потомков (указателей).

⦁ Глубина всех листьев одинакова.

⦁ Листья имеют ссылку на соседа, позволяющую быстро обходить дерево в порядке возрастания ключей, и ссылки на данные.

**Описание реализация алгоритма**

В+-дерево – структура данных, предназначенная для эффективного доступа к информации. При работе алгоритма реализуются добавление, удаление и поиск ключа.

**Поиск**

Отправная точка – Корень B+-дерева. Начинаем с него:

1. Проходимся по копиям пока не найдем копию больше искомого значения, тогда переходим к потомку перед этой копией.

1.2. Если значение нашего листа больше всех имеющихся копий – переходим к последнему сыну в данном узле.

2. Повторять эти действия пока не дойдем до листа

**Добавление**

Для добавления нового листа в первую очередь необходимо найти предлистовой узел, в который его необходимо добавить. В этом случае алгоритм таков:

⦁ Если предлистовой узел полностью не заполнен (количество указателей после вставки не более чем **2t**, то создать лист и добавить ссылку на него, а также копию значения листа.

⦁ В противном случае необходимо расщепить узел:

⦁ создать новый узел, затем переместить половину элементов из текущего в новый;

⦁ создать копию наименьшего ключа из нового узла и адрес на него (узел) в родительский;

⦁ если родительский узел заполнен, аналогично разделить его;

⦁ повторять до тех пор, когда родительский узел не будет нуждаться в расщеплении.

⦁ Если расщепляется корень — создать новый корень, имеющий одну копию и два указателя.

Пример операции добавления представлен на рисунке 2.

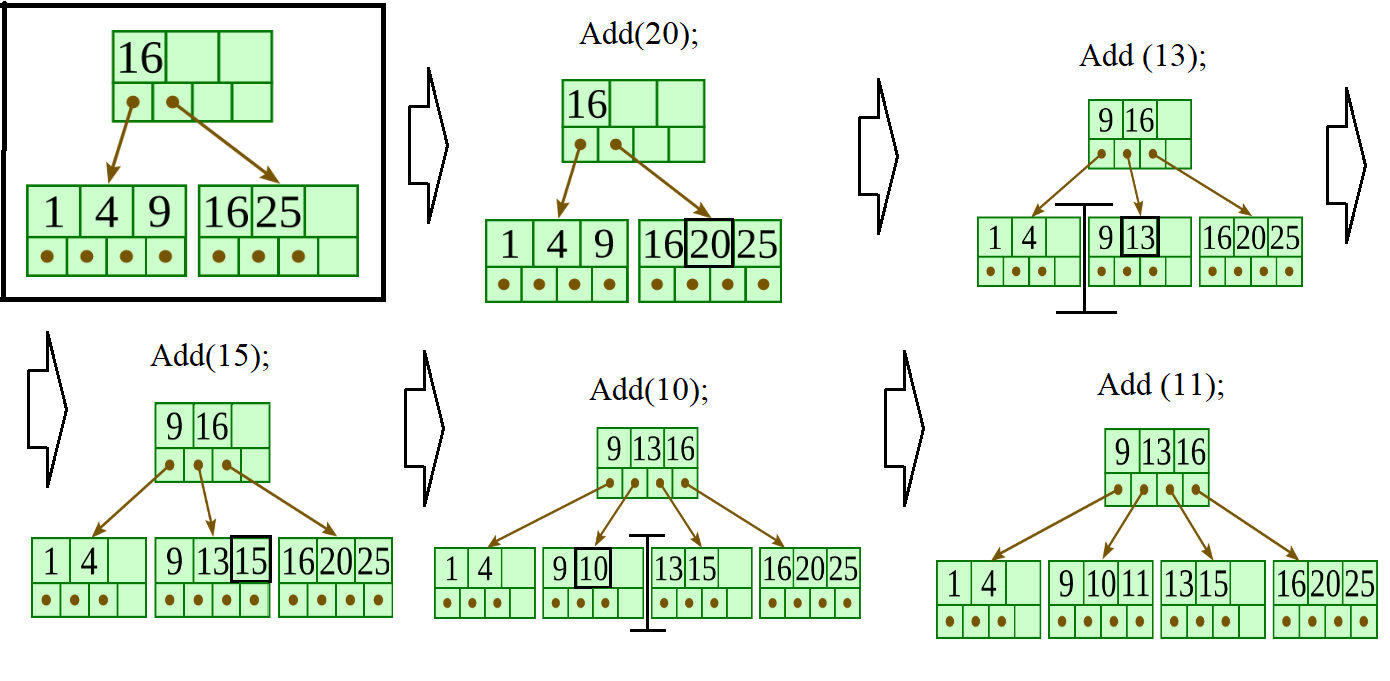


Рисунок 2 – Операции добавления узла

**Удаление**

Для удаления листа в первую очередь необходимо найти предлистовой узел, в котором есть ссылка на искомый лист. Алгоритм удаления от предлистового узла:

⦁ Если узел хотя бы наполовину заполнен — завершение алгоритма;

⦁ Если узел имеет меньше элементов, то:

⦁ если «брат» — элемент с общим предком, заполнен больше чем на половину выполнить попытку перераспределения элементов, то есть добавить в узел элемент из «брата» .

⦁ если выполнить перераспределение не удалось, объединить узел с «братом».

⦁ Если произошло объединение, удалить копию ключа и указатель, которые ссылаются на удалённый узел или его «брата» из родительского узла.

Объединение может распространяться на корень, тогда происходит уменьшение высоты дерева.

**Исследование алгоритма**

1. Теоретическая временная сложность представлена на рисунке 3.

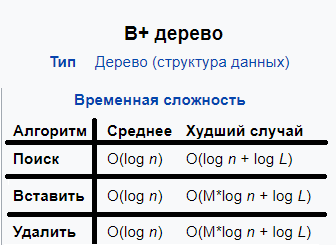


Рисунок 3 – Временная сложность, заявленная в Википедии

Проверим экспериментально время выполнения алгоритма при t=5.

1. Усредненные данные времени выполнения операций поиска, создания n-количества листов, удаления представлены в таблицах 1-3 соответственно.

Таблица 1 – Временная сложность операции поиска

|  |  |  |  |
| --- | --- | --- | --- |
| Искомое значение: | Случайное | Последний лист | Первый лист |
| Количество листов | Затраченное время, нс | | |
| 10 | 400 | 600 | 300 |
| 100 | 800 | 500 | 600 |
| 1 000 | 1 100 | 1 200 | 1 000 |
| 10 000 | 1 600 | 800 | 1 300 |
| 100 000 | 1 900 | 600 | 1 700 |
| 1 000 000 | 2 200 | 800 | 2 300 |

Таблица 2 – Временная сложность операции добавления n – листов

|  |  |  |  |
| --- | --- | --- | --- |
| Значение в ключах: | Случайное | Непрерывно растет | Непрерывно падает |
| Количество создаваемых листов | Затраченное время, мкс | | |
| 1 | 11 | 10 | 9 |
| 10 | 33 | 35 | 27 |
| 100 | 124 | 424 | 180 |
| 1 000 | 343 | 2 551 | 2 434 |
| 10 000 | 3 765 | 23 268 | 16 909 |
| 100 000 | 28 492 | 180 887 | 135 031 |
| 1 000 000 | 183 017 | 1 850 540 | 1 378 438 |

Таблица 3 – Временная сложность операции удаления

---

Все выполненные измерения проводились с помощью библиотеки «chrono».

1. Графики времени выполнения операций поиска, создания n-количества листов, удаления представлены на рисунках 4–6 соответственно.

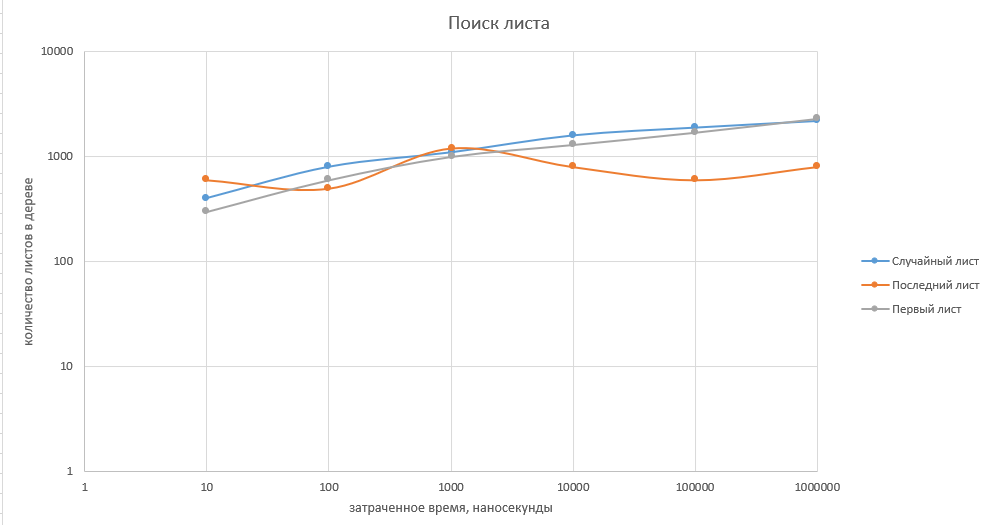


Рисунок 4 – Временная сложность операции поиска

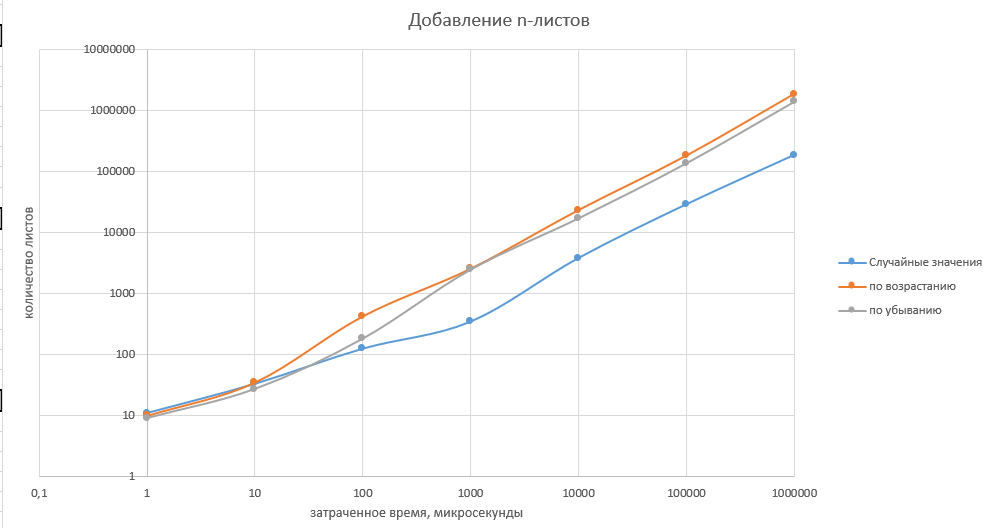


Рисунок 5 – Временная сложность операции добавления n – листов

---

Рисунок 6 – Временная сложность операции удаления

Из графиков мы можем сделать вывод, что в моей работе временная сложность у функция имеет линейную зависимость, нежели заявленную логарифмическую.

**Область применения**

Файловые системы ReiserFS, NSS, XFS, JFS, ReFS и BFS используют этот тип дерева для индексации метаданных; BFS также использует деревья B+ для хранения каталогов. NTFS использует деревья B+ для индексации каталогов и метаданных, связанных с безопасностью. APFS использует деревья B+ для хранения сопоставлений идентификаторов объектов файловой системы с их расположениями на диске и для хранения записей файловой системы (включая каталоги), хотя в листовых узлах этих деревьев отсутствуют родственные указатели. Системы управления реляционными базами данных, такие как IBM Db2, Informix, Microsoft SQL Server, Oracle 8, Sybase ASE, и SQLite поддерживают этот тип дерева для индексов таблиц. Системы управления базами данных типа "ключ-значение", такие как CouchDB и Tokyo Cabinet, поддерживают этот тип дерева для доступа к данным.

**Заключение**

Был изучен и написан алгоритм В+-дерево, а также проведены исследования этого алгоритма. Реализация его оказалась довольна трудна вследствие предусмотра большого количества критичных сценариев, однако несмотря на громоздкий код алгоритм В+-дерево прост и понятен.

**Список литературы**

1. <https://ru.wikipedia.org/wiki/B%E2%81%BA-%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE>
2. <https://habr.com/ru/company/sberbank/blog/413749/>
3. <https://ru.bmstu.wiki/B+_%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE>
4. <https://www.bibliofond.ru/view.aspx?id=702439>

**Приложение**