B-дерево

С. А. Шершаков

29 мая 2017 г.

Ред. 1.0 от 29.05.2017 г.

В документе представлено задание для самостоятельного выполнения студентами курса «Алгоритмы и структуры данных» и методические рекомендации по их выполнению, посвященное деревьям поиска на долгосрочной памяти — B-деревьям. Реализация задачи включает взаимодействие с инструментом автодокументирования кода Doxygen и unit-тестированием на основе библиотеки gtest.

1 Требования, цели и ожидаемые результаты

1.1 Требования

Студенты должны владеть следующими знаниями, умениями и навыками для выполнения задания.

- Разработка модульных приложений на языке C++.
- Концепция *шаблонов*, *обобщенного программирования*, параметризованные функции и классы C++.
- Основы стандартной библиотеки C/C++.
- Деревья поиска и их основные операции.

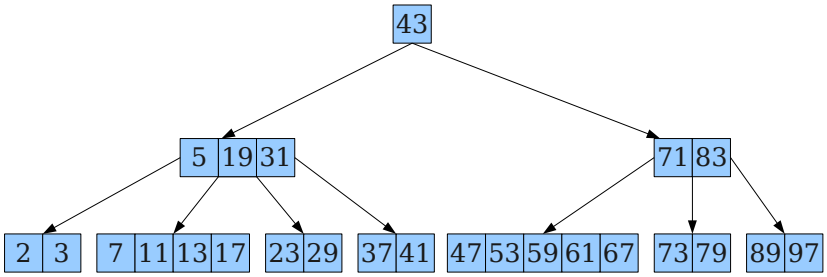

Рис. 1: B-дерево порядка 3.

Основная цель работы — практическое изучение реализации B-дерева с хранением в долговременной памяти и основных операций над ним.

Связанные задачи: автодокументирование кода с использованием инструмента Doxygen[1], unit-тестирование с использованием библиотеки gtest[2].

[1] `http://www.doxygen.org`

[2] `https://github.com/google/googletest`

Задачи для выполнения в рамках самостоятельной работы:

- Изучить способы реализации B-дерева поиска и основных операций над ним (рис. 1).
- Изучить интерфейсы и частичную реализацию класса `BaseBTree`, представляющего B-дерево с бинарным ключом фиксированной

длины (заголовочный файл `btree.h`) по предложенному коду и Doxygen-документации.
- Изучить интерфейсы и реализацию класса `FileBaseBTree`, конкретизирующего поток сериализации B-дерева для работы с файлами.
- Изучить интерфейсы и частичную реализацию класса `BaseBTree::PageWrapper`, представляющего объектно-ориентированную обертку (wrapper) над бинарной страницей, являющейся внутренним компактным представлением узла B-дерева.
- Изучить принципы поиска, вставки и удаления элемента в B-дереве по базовой книге [1].
- Закончить реализацию некоторых методов *поиска* и *вставки* элементов в B-дерево, а также реализовать связанный с ними функционал, ориентируясь на спецификацию метода (см. Doxygen-документацию) и настоящее руководство.
- [*Опционально*] протестировать полученную реализацию на множестве предложенных unit-тестов.
- [* *Опционально*] реализовать и протестировать методы удаления элемента(ов) из B-дерева[3].
- [** *Опционально*] реализовать и протестировать методы удаления элемента(ов) из B-дерева[4].

[3] Задача «со звездочкой» на 10 баллов.

[4] Задача «с двумя звездочками» на «специальных условиях».

2 Предпосылки

B-дерево является самобалансирующимся сильно ветвистым деревом поиска, основные операции над которым выполняется за логарифмическое время. В отличие от бинарных деревьев поиска, B-дерево хранит в узле большое число ключей, что существенно сокращает высоту дерева. Это позволяет сократить количество операций ввода/вывода на медленных носителях, что обуславливает широкое применения этой структуры данных для представления индексов в базах данных, расположенных на медленных дисковых накопителях.

Дополнительная информация о B-деревьях может быть почерпнута из литературы: [1].

3 Описание задания

В рамках самостоятельной работы студентам предлагается закончить реализацию указанных ниже классов и методов.

1. Метод `BaseBTree::PageWrapper::splitChild()` разделения дочернего узла на два доверних.
2. Метод `BaseBTree::PageWrapper::insertNonFull()` вставки ключа в незаполненный узел дерева.
3. Метод `BaseBTree::insert()` вставки ключа в дерево.

4. Методы поиска одного `BaseBTree::search()` и всех `BaseBTree::searchAll()` ключей в дереве по заданному значению.
5. * Методы удаления одного `BaseBTree::remove()` и всех `BaseBTree::removeAll()` ключей в дереве по заданному значению.

Полная информация по предложенному программному коду может быть получена из автосоздаваемой документации Doxygen[5] или из комментариев к программным файлам.

[5] Предоставляется в виде архива с каталогом `/html`, из которого необходимо открыть файл `/html/index.html` в любом браузере.

*3.1 Описание расширенного опционального задания ***

Получается индивидуально у преподавателя.

4 Структура проекта

Структура проекта является стандартной для аналогичных заданий курса.

5 Указания по выполнению задания

Выполнение задачи сводится к следующей рекомендуемой последовательности действий.

- Ознакомиться с настоящим руководством-заданием.
- Получить исходные файлы для работы[6].
- Ознакомиться с предлагаемыми для работы исходными файлами и автодоксидокументацией к ним.
- Определиться с объемом выполняемой работы. Задекларировать макрос `BTREE_WITH_DELETION`, если планируется выполнять задачу «со звездочкой».
- Реализовать недостающие методы, модифицировав файлы `btree.h/.cpp`, а также при необходимости `btree_adapters.h`.
  Важный момент: необходимо внимательно просмотреть код модуля на предмет комментариев, указывающих, какие методы в каком объеме надо реализовывать.
- Протестировать (по желанию) реализацию на предложенном наборе тестов (файлы `btree1_tests.cpp` и `adapters1_tests.cpp`).
- При необходимости расширения множества тестов, необходимо поместить их в новый файл `btree_student_test.cpp`.
- Загрузить результаты работы в виде единого архива в ассоциированный проект системы LMS до крайнего срока, указанного в сводке проекта.

[6] С использованием системы LMS: `http://lms.hse.ru`

5.1 Сдаваемые файлы

- `btree.h/.cpp` — реализация основной задачи;

- `btree_student_test.cpp` — дополнительные тесты, если разрабатывались;
- *при необходимости*, другие файлы с учетом стандартных замечаний[7].

[7] Если такая необходимость возникает, при сдаче такого файла необходимо сопроводить его комментариями в виде файла `student_notes.md`, который следует поместить в каталог со сдаваемым файлом. В `student_notes.md` комментарии, почему соответствующий файл сдается.

Список литературы

[1] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. *Introduction to Algorithms*, 3 ed. The MIT Press, 2009.