Домашнее задание №4
Алгоритмы и структуры данных 2020

B-дерево

Авторы задания: команда курса «Алгоритмы и структуры данных 2020».
Вопросы по заданию направлять: Рахмановский Андрей Дмитриевич a.rahmanovsky@yandex.ru или в канал в MS Teams “**Домашнее задание 4 B-дерево**”

Отправка решения

Результатом решения должен быть один файл на языке C++ (.cpp), а также файлы тестов (входные и выходные данные). Именно файл на языке C++ (.cpp) и тесты будут оцениваться. Решение нужно загрузить в LMS в проект «**Домашнее задание 4 B-дерево**» по дисциплине Алгоритмы и структуры данных 2020 уч. год Б 2 курс (код 131940, ОП: М090304ПИНЖ)
При выявлении плагиата будет выставлена оценка 0 баллов

Дедлайн. 19 ноября 2020 г. 23:59 МСК

Пожалуйста, загружайте работы заранее, чтобы не возникало технических препятствий в последний момент. Время на сервере LMS и вашем устройстве может отличаться. Пожалуйста, настройте доступ в LMS (даже если Вы проходите курс по ИУП): архивы с решениями не принимаются по электронной почте (даже с небольшим опозданием).

Что нужно сделать?

1. Задание разбито на 3 уровня сложности. **Выберите только один уровень**. Выполнение более сложного задания не гарантирует автоматически более высокую оценку.
2. Решите задание (один файл C++).
3. Напишите 5 своих тестов (пары файлов входных и выходных данных) к задаче, пронумеруйте их соответственно. Положите файлы с входными данными в папку input, а с выходными в папку output.
4. Создайте папку с решением по шаблону <Имя>_<группа>_<уровень>, например RahmanovskiyAndrey_161_2_2 (обратите внимание, есть два вида одного уровня сложности – 2_1 и 2_2). Положите в неё папки с тестами и один файл .cpp с решением задачи.
5. Упакуйте всё в Zip-папку и отправьте в LMS (размер файла должен быть не более 1МБ).

Внимание: не публикуйте свое решение в открытом доступе, напр. на GitHub либо других открытых источниках. Это действие будет приравниваться к плагиату и повлечет за собой оценку 0 баллов за домашнее задание (в том числе при обнаружении такового факта уже после выставления оценки).

Мотивация

В современных компьютерах память – иерерхична. Основные виды памяти - оперативная память и HDD/SSD. Оперативная память обладает высокой скоростью на чтение и запись, однако она сильно ограничена в своих размерах, в связи с чем при работе с большими данными отсутствует возможность разместить все необходимые элементы в оперативной памяти.

HDD/SSD, в свою очередь, обладают гораздо большими объемами памяти, однако скорость доступа на чтение и запись является намного более медленной, нежели чем у оперативной памяти.

Для эффективной работы с вторичной памятью была разработана структура данных B-дерево. Она позволяет хранить большие объемы данных во вторичной памяти, при этом сокращает необходимое количество обращений к ней для внесения изменений и считывания данных. В настоящее время разновидности B-дерева используются почти во всех современных СУБД.

Задание

Уровень 1. Максимум 6 баллов из 10.
Необходимо реализовать B-дерево, поддерживающее операции вставки и поиска и работающее исключительно в оперативной памяти.

Формат входных данных.
На вход программе подается файл с несколькими командами двух типов:
insert key value - вставляет значение по ключу.
find key - ищет значение по ключу.
Ключ и значение - целые числа, по модулю не превышающие 10^9
Количество команд не превышает 10^9

Формат выходных данных.
На каждую команду типа insert необходимо вывести true, если операция прошла успешно и false, если данный ключ уже присутствует в структуре.
На каждую операцию типа find необходимо вывести значение, которое хранится по переданному ключу или null, если такого ключа в структуре нет.

Пример

Входные данные	**Выходные данные**
insert 1 2 insert 3 4 insert 1 10 find 2 find 3	true true false null 4

Уровень 2_1. Максимум 8 баллов из 10.
Необходимо реализовать B-дерево, поддерживающее операции вставки, поиска и удаления и работающее исключительно в оперативной памяти.

Формат входных данных.
На вход программе подается файл с несколькими командами двух типов:
insert key value - вставляет значение по ключу.
find key - ищет значение по ключу.
delete key - удаляет ключи и значение из структуры.
Ключ и значение - целые числа, по модулю не превышающие 10^9
Количество команд не превышает 10^9

Формат выходных данных.
На каждую команду типа insert необходимо вывести true, если операция прошла успешно и false, если данный ключ уже присутствует в структуре.
На каждую операцию типа find необходимо вывести значение, которое хранится по переданному ключу или null, если такого ключа в структуре нет.
На каждую операцию типа delete необходимо вывести значение, которое было удалено, если переданный ключ присутствует в структуре и null, если такого ключа нет.

Пример

Входные данные	**Выходные данные**
insert 1 2 insert 3 4 insert 1 10 delete 5 find 2 find 3 delete 1 find 1	true true false null null 4 2 null

Уровень 2_2. Максимум 8 баллов из 10.
Вместо реализации операции удаления и работы в оперативной памяти можно написать реализацию B-дерева, аналогичную заданию на 6 баллов, но работающую со вторичной памятью.

Уровень 3. Максимум 10 баллов из 10.
Необходимо реализовать B-дерево, поддерживающее операции вставки, поиска и удаления и работающее со вторичной памятью.

Структура B-дерева

Должны выполняться следующие правила:

- В каждом узле должны быть следующие атрибуты:
    1. Количество ключей, которые хранятся в узле.
    2. Массив ключей, которые хранятся в данном узле, отсортированный по возрастанию.
    3. Булево значение, равное True, если данный узел является листом и False в противоположном случае.
    4. Список узлов, которые являются дочерними для текущего узла:
        a. При работе с оперативной памятью необходимо использовать указатели C на дочерние узлы.
        b. При работе со вторичной памятью особые требования к организации данного пункта не предъявляются. В качестве **рекомендации**, данный список можно хранить как массив string, в котором будут содержаться пути до необходимых файлов, или всегда называть папки и файлы определенным образом и хранить только названия файлов из которых будет понятно, где данный файл располагается. Но любые другие способы организации не запрещены.
    5. Массив значений, где каждое значение сопоставляется соответствующему ключу (не запрещается хранить один массив пар ключ-значение).
    6. Переменная string, хранящая имя/путь файла, который соответствует данному узлу.
- Ключи, хранимые в узле, должны определять поддиапазоны ключей, хранящихся в поддеревьях. Формально, если k_iявляется произвольным ключом, хранящимся в поддереве с корнем X и ключами X.keys, то k_iдолжно содержаться в поддереве, за которое отвечает диапазон $X.keys_i \leq k_i \leq X.keys_{i+1}$.
- Все листья расположены на одинаковой высоте.
- Каждый узел, кроме корневого, должен содержать не менее t-1 ключей, и следовательно, не менее t дочерних узлов.
- Каждый узел должен содержать не более 2t-1 ключа и, соответственно, не более 2t потомков.
- Параметр t передается в аргументах командной строки.
- Все ключи и значения являются целыми числами, не превышающими по модулю 10^9
- При выборе уровня, в котором необходимо работать со вторичной памятью, все узлы дерева должны храниться в виде двоичных файлов, за исключением корневого узла, он всегда хранится в оперативной памяти.
- В качестве структур для организации хранения данных в узле можно использовать любые контейнеры из STL на Ваше усмотрение.

Таблица примерной организации узла для 3 уровня сложности.

Тип	Тип данных	Название	Пояснение
Поле	int	cntKeys	Хранит количество ключей
Поле	vector<int>	keys	Хранит сами ключи
Поле	bool	isLeaf	Хранит, является ли узел листом
Поле	vector<string>	children	Хранит ссылки на потомков
Поле	vector<int>	values	Хранит значения
Поле	string	fileName	Имя файла, где хранятся данные для этого узла
Метод	void	read	Считывает данные из файла, соответствующего данному узлу
Метод	void	write	Записывает данные в файл, соответсвующий данному узлу

Данная таблица является примерной и отклонения от неё не повлекут снижение оценки.

Как будет оцениваться работа?

Решения будут проверяться на наборах тестов, каждый тест это два файла - один для входных данных, другой для выходных. Таким образом, ввод и вывод данных происходит через файлы.

Будут оценены: корректность кода, составленные тесты (граничные условия, сложные случаи и т.п.), декомпозиция, аккуратность кода. Возможны устные собеседования при сомнении в самостоятельности выполнения работы.

В приложенной к условию задачи папке расположено 5 тестов, на которых можно проверить работоспособность решения, однако стоит учитывать, что прохождение приложенных тестов не гарантирует полное отсутствие ошибок в решении (решения будут также проверяться и на других, приватных тестах). Файлы входных данных именуются по принципу “test<номер теста>.txt”, файлы выходных данных (или ответов) называются “answer<номер теста>.txt”.

Названия файлов для входных и выходных данных должны быть указаны через командную строку. Кроме того, в параметрах командной строки могут указываться не только имена файлов и директорий, но и относительные/абсолютные пути к ним. Помимо этого, в аргументах командной строки будет передаваться параметр t и путь до папки (для уровней 2_2 и 3), в которой необходимо хранить двоичные файлы.

Например, так будет выглядеть тестирование программы RahmanovskiyAndrey_161_2_2.cpp

```
cl RahmanovskiyAndrey_161_2_2.cpp
RahmanovskiyAndrey_161_2_2 10 folder test1.txt answer1.txt
```

либо

```
RahmanovskiyAndrey_161_2_2 1000 path/to/folder path/to/test1.txt path/to/answer1.txt
```

где

1-ый параметр - параметр t
2-ой параметр - папка, в которой необходимо хранить двоичные файлы
3-ый параметр - путь к входному файлу
4-ый параметр - путь к выходному файлу
(пути могут быть абсолютными и относительными)

Список литературы

1. Cormen, T. H. (2009). Introduction to Algorithms (Vol. 3rd ed). Cambridge, Mass: The MIT Press. Retrieved from
2. Лекция «СД для вторичной памяти, B-trees» (на OneDrive),
3. Семинарский репозиторий с примерами кода https://github.com/b1nd/cpp