Лабораторная работа № 2 по курсу дискртного анализа: сбалансированные деревья

Выполнил студент группы М8О-209Б-23 *Кривошапкин Егор*.

Условие

Кратко описывается задача:

1. Необходимо создать программную библиотеку, реализующую указанную структуру данных, на основе которой разработать программу-словарь. В словаре каждому ключу, представляющему из себя регистронезависимую последовательность букв английского алфавита длиной не более 256 символов, поставлен в соответствие некоторый номер, от 0 до $2^{64} - 1$. При этом разным словам может быть поставлен в соответствие один и тот же номер.

2. Вариант задания: 4. Реализация с использованием B-дерева.

Метод решения

B-дерево — это сбалансированное дерево поиска, где каждый узел содержит от $t-1$ до $2t-1$ ключей (кроме корня), а количество потомков на единицу больше числа ключей. Здесь t — это минимальная степень дерева, определяющая нижнюю границу количества потомков узла (узел может иметь минимум t потомков, если он не лист). Сбалансированность достигается за счёт того, что все листья находятся на одном уровне, а операции поддерживают строгие ограничения на заполнение узлов.

Балансировка происходит через:

1. Разделение узла — если при вставке узел переполняется (превышает $2t-1$ ключей), его делят на два, а средний ключ поднимается в родителя.
2. Слияние или перераспределени**е** — если при удалении узел опустошается ниже минимума ($t-1$ ключей), он объединяется с соседом или получает ключи от «брата».

Вставка аналогична бинарному дереву, но ключ добавляется в лист. Если узел переполнен, разделение повторяется рекурсивно до корня.

Удаление выполняется как в бинарном дереве, но после проверяют заполнение узла. Если нарушены условия, проводят слияние или перераспределение ключей.

Поиск работает через сравнение ключей в узле и переход в нужное поддерево, сохраняя сложность $O(t*\log_t n)$ (линейный поиск элемента в каждом узле).

Далее реализованное B-дерево используется в качестве словаря, над которым производятся операции добавления, удаления, поиска, записи в файл или чтения из файла в зависимости от команд пользователя.

Словарь хранит пары ключ-значение. Ключом выступает строка (не более 256 значащих символов), значением беззнаковое 8-ми байтовое целое.

Описание программы

Программа состоит из файла main.cpp, в котором реализован класс B-дерева и присущие ему методы вставки, удаления, поиска, вспомогательные скрытые методы для балансировки, поиска наименьшего элемента и т.д., а также описана основная логика работы программы: общение с пользователем, чтение/запись словаря в файл.

Дневник отладки

1. 27.04: TL на 15 тесте.

   Решение: использование бинарного поиска для операций поиска и определения индексов элементов при вставке и удалении из узла. Решение не увенчалось успехом, теперь появляется runtimeError.

2. 28.04: RE на 5 тесте.

   Решение: оптимизировал функции сравнения строк, убрал функцию приведения строки к нижнему регистру, теперь регистр понижается в самих функциях. Решение увенчалось успехом.

Тест производительности

Реализованное B-дерево было протестировано в сравнении со стандартным контейнером std::map, внутренняя реализация которого основана на красно-чёрном дереве — ещё одном типе сбалансированного дерева поиска. Для начального этапа анализа производительности минимальная степень B-дерева была установлена на значение t=3. Такой выбор параметра позволил оценить баланс между частотой операций разделения/слияния узлов и эффективностью использования памяти, характерный для B-деревьев с малыми значениями t.

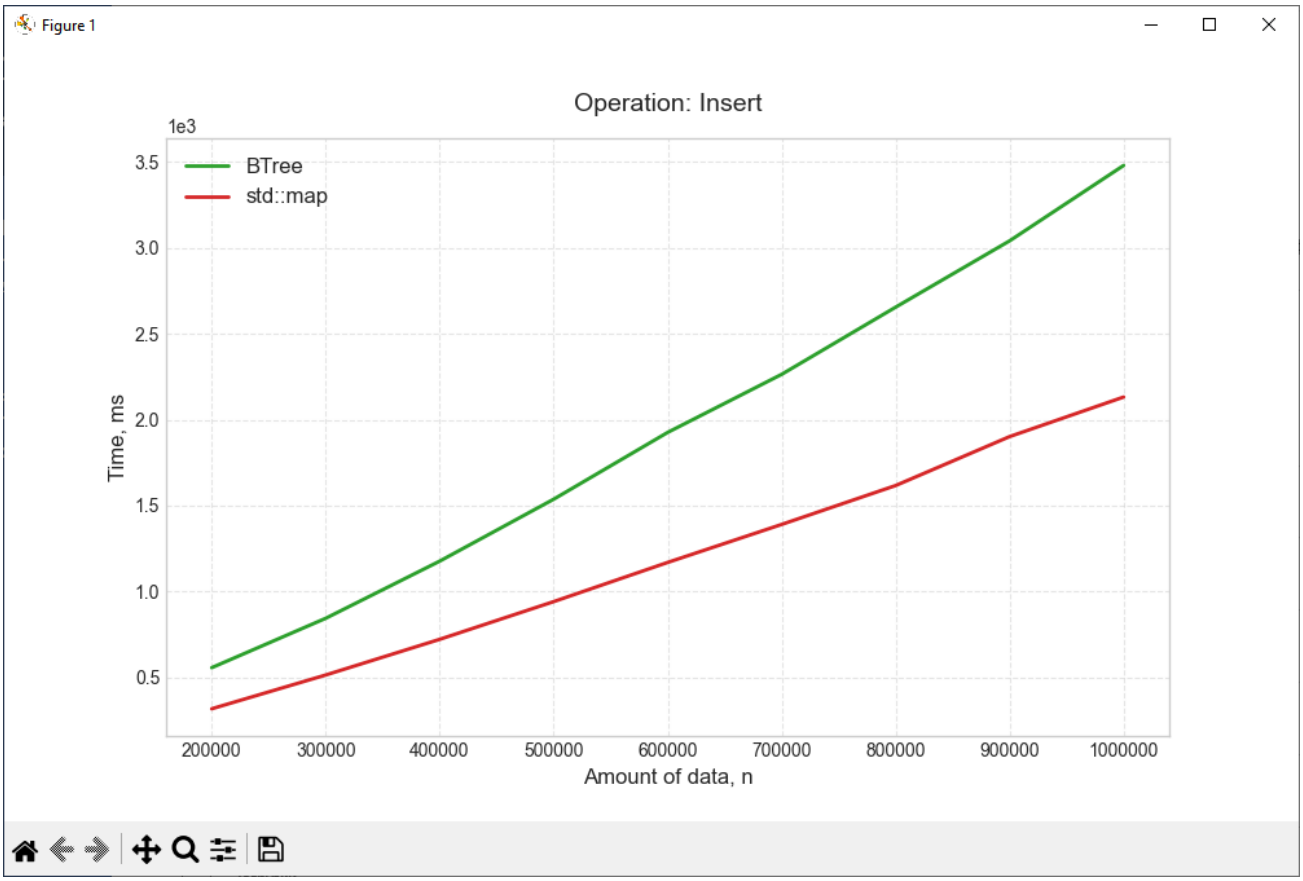


Figure 1
Operation: Insert
1e3
BTree
std::map
Time, ms
3.5
3.0
2.5
2.0
1.5
1.0
0.5
200000
300000
400000
500000
600000
700000
800000
900000
1000000
Amount of data, n

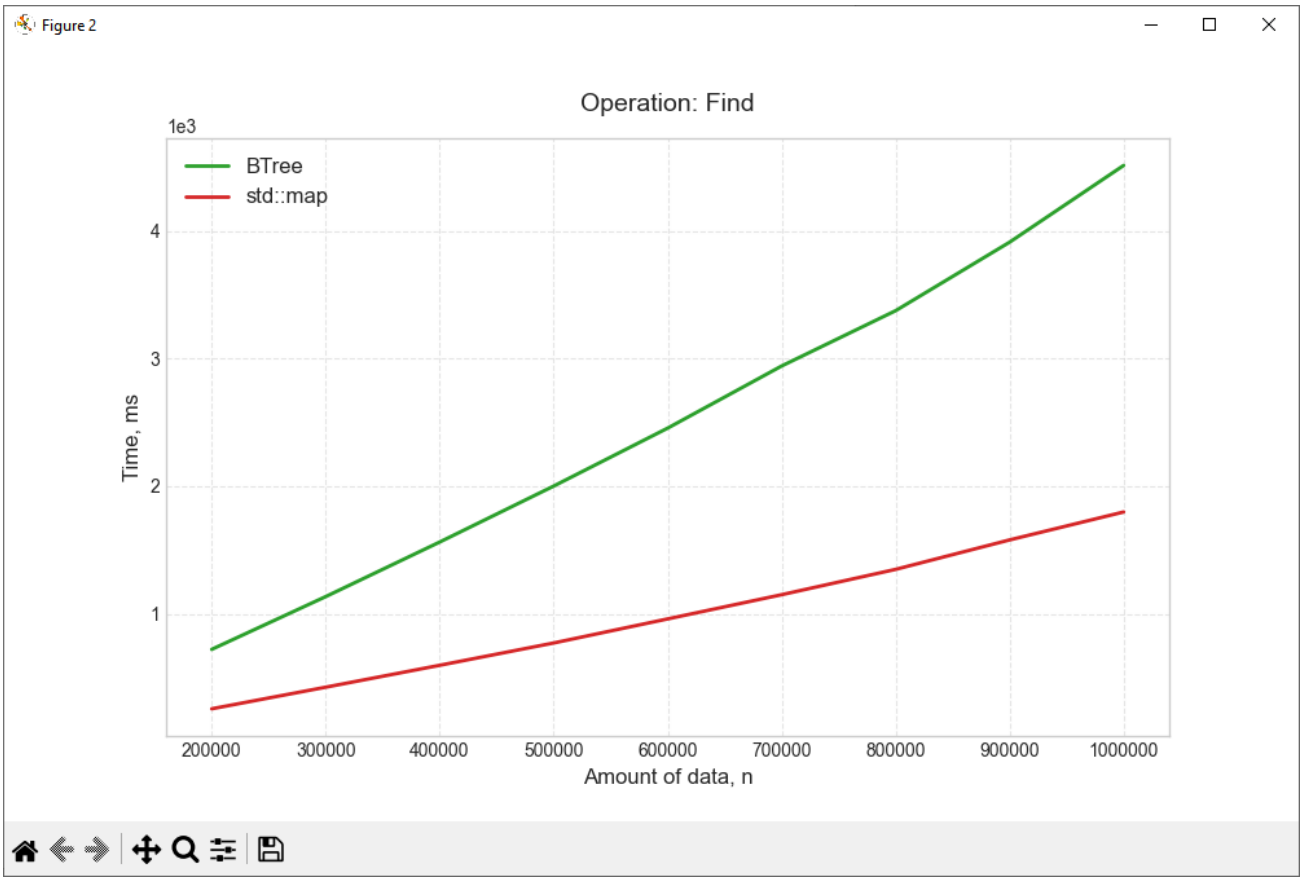
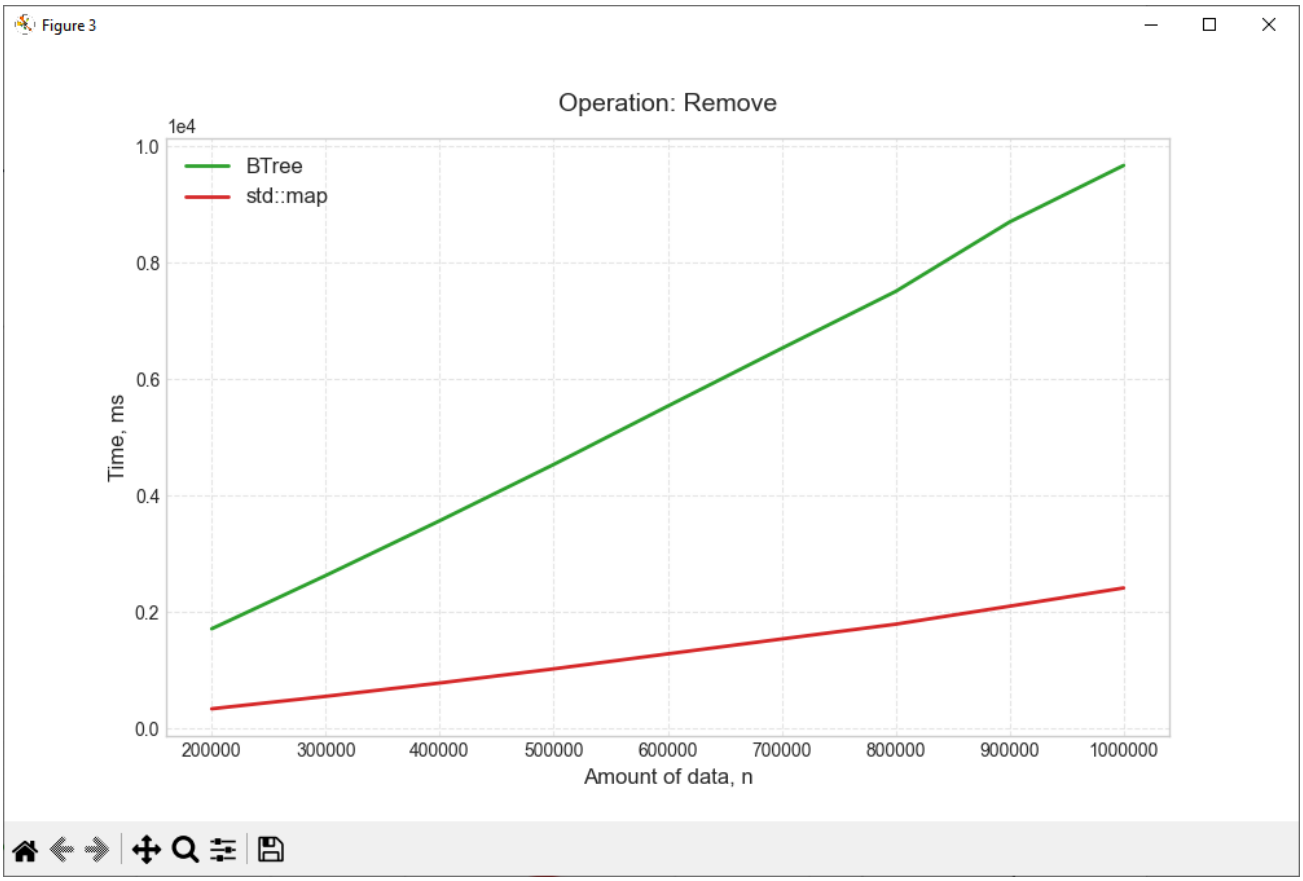


Figure 2
Operation: Find
1e3
BTree
std::map
Time, ms
1
2
3
4
200000
300000
400000
500000
600000
700000
800000
900000
1000000
Amount of data, n



Наблюдаемое отклонение от логарифмической зависимости на графиках, вероятно, вызвано частыми операциями выделения и освобождения памяти в куче — эти процессы требуют значительных временных ресурсов. Кроме того, схожесть форм кривых указывает на идентичную асимптотическую скорость роста времени выполнения операций, что подтверждает их сопоставимую алгоритмическую сложность.

Недочёты

Программа демонстрирует снижение производительности при работе со строковыми ключами, так как регистронезависимое сравнение требует дополнительных вычислительных ресурсов, особенно для длинных ключей. Поиск внутри узла реализован через линейный перебор, что замедляет работу при большом количестве элементов в узлах. Кроме того, рекурсивный подход к удалению узлов в деструкторе может вызвать переполнение стека для глубоких деревьев.

Выводы

B-дерево демонстрирует свою универсальность в различных сценариях. Как показано в

ходе лабораторной работы, на его основе можно создать аналог контейнера std::map, обеспечивающий эффективное хранение пар ключ-значение. Кроме того, структура идеально подходит для реализации простых баз данных, так как минимизирует количество операций чтения с диска за счёт группировки данных в узлах.