Московский Авиационный Институт
(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №2 по курсу
«Дискретный анализ»

Сбалансированные деревья

Студент: Эсмедляев Федор Романович
Группа: М8О–212Б–22
Вариант: 2
Преподаватель: Н.Д.Глушин
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2024.

Вариант: 2

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 $2^{64} - 1$. Разным словам может быть поставлен в соответствие один и тот же номер.
Программа должна обрабатывать строки входного файла до его окончания. Каждая строка может иметь следующий формат:

+ word 34 — добавить слово «word» с номером 34 в словарь. Программа должна вывести строку «OK», если операция прошла успешно, «Exist», если слово уже находится в словаре.

- word — удалить слово «word» из словаря. Программа должна вывести «OK», если слово существовало и было удалено, «NoSuchWord», если слово в словаре не было найдено.

word — найти в словаре слово «word». Программа должна вывести «OK: 34», если слово было найдено; число, которое следует за «OK:» — номер, присвоенный слову при добавлении. В случае, если слово в словаре не было обнаружено, нужно вывести строку «NoSuchWord».

! Save /path/to/file — сохранить словарь в бинарном компактном представлении на диск в файл, указанный парамером команды. В случае успеха, программа должна вывести «OK», в случае неудачи выполнения операции, программа должна вывести описание ошибки (см. ниже).

! Load /path/to/file — загрузить словарь из файла. Предполагается, что файл был ранее подготовлен при помощи команды Save. В случае успеха, программа должна вывести строку «OK», а загруженный словарь должен заменить текущий (с которым происходит работа); в случае неуспеха, должна быть выведена диагностика, а рабочий словарь должен остаться без изменений. Кроме системных ошибок, программа должна корректно обрабатывать случаи несовпадения формата указанного файла и представления данных словаря во внешнем файле.

Для всех операций, в случае возникновения системной ошибки (нехватка памяти, отсутсвие прав записи и т.п.), программа должна вывести строку, начинающуюся с «ERROR:» и описывающую на английском языке возникшую ошибку.

Различия вариантов заключаются только в используемых структурах данных:

Красно-чёрное дерево.

Пример

Ввод	Вывод
+ a 1	OK
+ A 2	Exist
+ aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa	OK
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa	OK: 18446744073709551615
A	OK: 1
- A	OK
a	NoSuchWord

Метод решения

Реализация структуры похожую на std::map тоже на основе красно-черного дерева. Есть функционал вставки, удаления, поиска, загрузки и выгрузки из файла с заданным путем до него.

Описание программы

Реализован класс RB_tree и для этого класса реализован класс Node, который имеет 5 параметров:
1)Ключ + Значение

2)Цвет

3)Правый сын

4)Левый сын

5)Отец

Класс RB_tree имеет такие методы:

1)Insert – Добавление

2)Erase – Удаление

3)Seek – поиск

4)Load – загрузка дерева из файла

5) Save – загрузка дерева в файл

И другие вспомогательные методы по типу rotateRight, rotateLeft, fix, fix_delete – это

разворот дерева вправо, влево, балансировка после вставки, балансировка после удаления соответственно.

Ключ + значение представляют собой – std::pair<std::string, unsigned long long>

Дневник отладки

1) Программа прошла все тесты без учета методов Save и Load, но время было чуть больше 20 секунд, что не годится решения, ведь там ограничение 15 секунд, но я просто убрал удаление из массива, который был нужен для восстановления номера, который присваивался строке и реализовал дерево без массива с удалениями, а просто через пары.

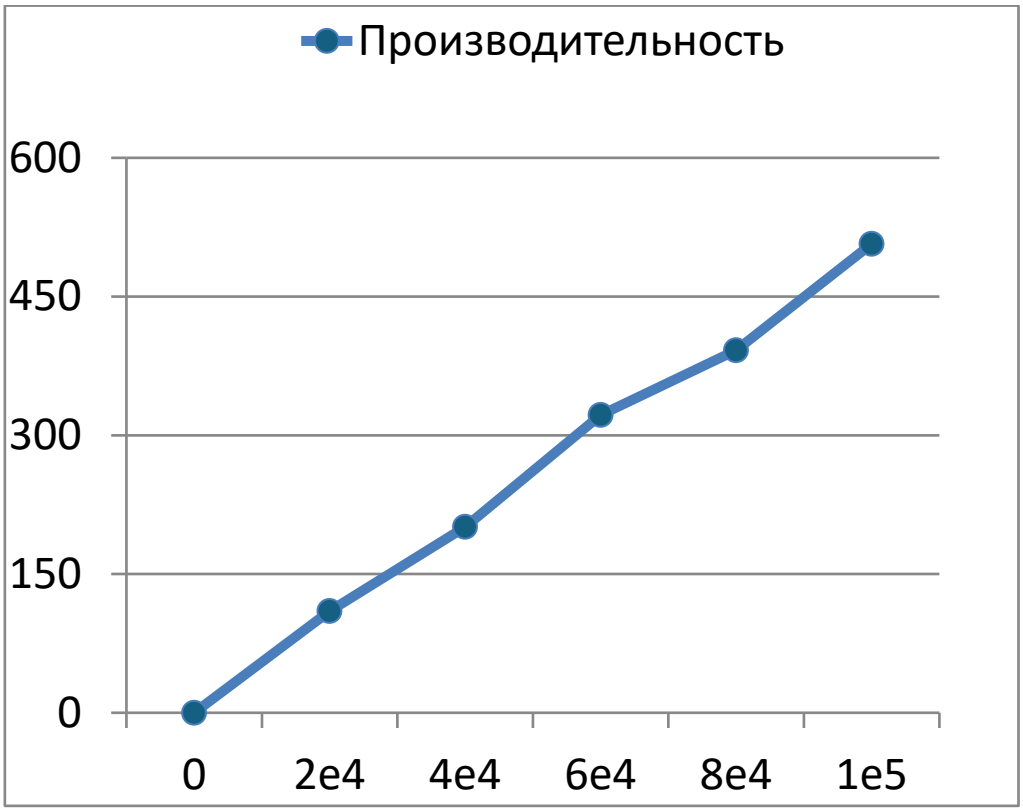
2) Реализация Save и Load и сразу же WA 7

3) Проблема заключалась в записи и чтении переменных(не все переменные записывались и не все прочтенные присваивались)

4)Ошибочный if в main, из-за которого не вызывались методы Load и Save, исправление if и получение результата – OK

Тест производительности

Сложность вставки, удаления, поиска = O(logN), где n – количество элементов дерева.



Вывод:

Я реализовал структуру данных – красно-черное дерево, на которой написаны такие основные типы данных, как std::set, std::map. Весь успех в этом типе данных – мы знаем точное количество черных узлов, которое мы не превысим при поиске любой вершины или иных операций. Именно поэтому эта структура крайне удобна, и скорее всего является самой быстрой из всех.