Лабораторная работа №2 по курсу дискретного анализа: Сбалансированные деревья

Выполнил студент группы М8О-212Б-22 МАИ *Корнев Максим.*

Условие

Вариант: 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 264 - 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.

Различия вариантов заключаются только в используемых структурах данных: **AVL дерево.**

Метод решения

Был реализован класс узла Node и класс AVL дерева. Эта структура данных предоставляет функционал для вставки, удаления, поиска данных, загрузки дерева из файла и сохранения дерева в файл.

Описание программы

Класс `AVL_Tree` реализует сбалансированное дерево поиска. Поддерживает операции вставки, удаления, поиска, загрузки дерева из файла и сохранения дерева в файл.

Класс `Node` содержит в себе структуру узла.

Дневник отладки

1. Программа получила WA на тесте 1 в "0. Декартово дерево". Проблема была в фукции балансировки, которую я в скором времени пофиксил.

2. Программа получила RE на тесте 1 в "0. Декартово дерево". Проблема была в том, что функция, возвращающая unsigned long long не возвращала число, путем небольшого фикса в виде добавления return -1 все заработало.

3. Программа получила TL на тесте 5 в "0. Декартово дерево". Проблема была в функциях балансировки, пришлось подебажить, чтобы исправить.

4. После исправления всех ошибок, программа прошла тесты "0. Декартово дерево".

5. Программа с первого раза прошла все тесты "1. AVL-дерево".

Тест производительности

Сложность вставки и удаления элементов в дереве $O(logn)$, где n - количество узлов в дереве. Сложность поиска элемента в дереве также $O(logn)$.

Для построения графика (Рис. 1) использовались тесты с 10, 100, 250, 750, 1000, 2500, 5000, 7500, 10000, 25000, 50000, 75000, 100000 строками данных.

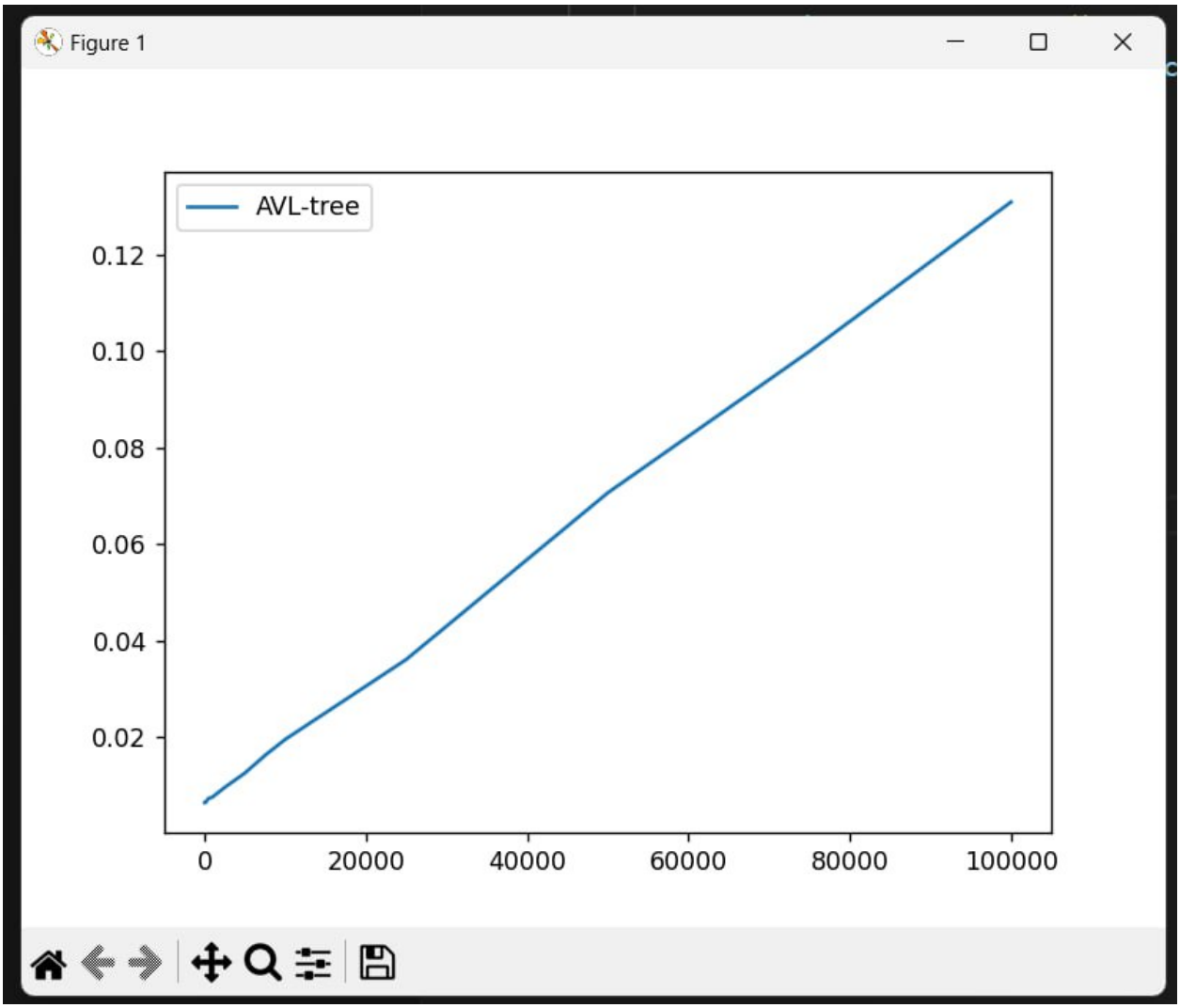

Рис. 1: График зависимости времени работы программы от количества введенных данных

Выводы

Я познакомился с сбалонсированными деревьями, в частности научился реализовывать AVL-дерево. Сравнил время работы программы при различных размерах входных данных. Сбалансированные деревья играют ключевую роль в хранении и поиске данных, обеспечивая быстрое выполнение операций вставки, поиска и удаления за логарифмическое время. Это делает их особенно ценными для приложений, где требуется оперативный доступ к данным, таких как эффективные базы данных и ассоциативные массивы. Однако создание и поддержание баланса в таких деревьях для операций вставки и удаления элементов является далеко не тривиальной задачей.