Лабораторная работа № 2 по курсу дискретного анализа: сбалансированные деревья

Выполнил студент группы М8О-207Б-20 *Белоусов Егор Владимирович*

Условие

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 параметром 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.

Вариант 1: AVL дерево

Метод решения

AVL дерево является бинарным самобалансирующимся деревом поиска с высотой $h = O(log(n))$. Для балансировки дерево должно обладать одним важным свойством:
Высота левого сына отличается от высоты правого сына не более чем на 1, т.е. $|h_{left} - h_{right}| \leq 1$. Можно показать, что, если всегда поддерживать данное свойство, то высота дерева будет пропорциональна $O(log_{\phi}(n))$. Для соблюдения этого свойства используются малые и большие повороты дерева. В каждой вершине дерева хранится *key* – ключ, *value* – значение, *balance* – разница между высотой левого и правого сына. Для поиска используется самый обычный алгоритм для бинарного дерева поиска: если ключ совпал с ключом в вершине, то мы нашли элемент, если ключ меньше ключа в вершине, то идем налево, иначе направо. При удалении и добавлении вершины: после операции с заданной вершиной дерево перебалансируется используя *balance* и повороты. Итого все основные операции выполняются за $O(h)$, где h – высота дерева, в нашем случае $h = O(log(n))$.

Описание программы

В лабораторной работе есть два файла **AVL.h** и **main.cpp**. В **AVL.h** реализуется само дерево AVL в виде класса, а в **main.cpp** реализован интерфейс взаимодействия, использующий AVL.

Дневник отладки

При выполнении лабораторной работы основные проблемы возникали с удалением и перебалансировкой. После правильной обработки краевых случаев программа стала корректно работать, но не проходила тесты из-за слишком неоптимизированного чтения дерева из файла.

Тест производительности

Раз уж в лабораторной работе реализуется самобалансирующееся дерево, то логично сравнивать его с *std::map*. Для тестирования был реализован генератор тестов на C++ и решение, использующее *std::map*.

N	AVL	std::map
10^3	221 ms	156 ms
10^4	2311 ms	1419 ms
10^5	24304 ms	14062 ms

Как видно из таблицы, решение, использующее AVL, немного хуже *std::map*, данное кратное ухудшение связано с использованием меньшего основания в логарифме, а также с более частыми поворотами в дереве.

Выводы

В данной лабораторной работе я освежил свои знания и умения в сбалансированных деревьях. После написания кода AVL дерева возникло множество проблем с правильной обработкой указателей и частыми утечками памяти. Все эти проблемы были решены с помощью невероятно полезной утилиты valgrind.