Лабораторная работа № 2 по курсу дискретного анализа: сбалансированные деревья

Выполнил студент группы М08-312Б МАИ *Лобанов Олег.*

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.

+ 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.

Вариант дерева: Красно-черное дерево

Метод решения

Красно-черное дерево представляет собой бинарное дерево поиска с одним дополнительным битом цвета в каждом узле. Цвет узла может быть красным или черным. В соответсвии с накладываемыми на узлы дерева ограничениями ни один пустой путь от корня в красно-черном дереве не отличается от другого по длине не более чем в два раза, так как красно-черные деревья являются приближенно сбалансированными.

Дерево называется *красно-черным*, если удовлетворяет следующим условиям:

- Каждый узел красный или черный
- Корень древа черный
- Каждый лист дерева является черным узлом
- Если узел красный, то его дочерние узлы черные
- Для каждого узла все простые пути от него до листьев-потомков данного узла, содержат одинаковое количество черных узлов.

Описание программы

- **Поиск.** Идентичен поиску в бинарном дереве
- **Вставка.** При вставке нового узла, он изначально добавляется как красный. Затем выполняются проверки и возможные повороты для сохранения балансировки дерева. Если родительский узел красный, необходимо проверить соседний узел и выполнить соответствующие действия.
- **Удаление.** Удаление узла может быть сложным, особенно если удаляемый узел имеет детей. В этом случае узел заменяется его наименьшим узлом в правом поддереве и затем удаляется. После удаления выполняются проверки и повороты для поддержания баланса.
- **Сохранения в файл.** Происходит сохранение дерева. Сначала записывается длина ключа. Затем сам ключ. Затем значение ключа. Этот процесс повторяется рекурсивно для всех узлов дерева.
- **Загрузка из файла.** Происходит загрузка из бинарного файла, считывая ключи и значения узлов в порядке, в котором они были записаны.

Класс **TNode** представляет собой узел дерева, который содержит поля для ключа, значения и цвета, а ткаже указатели на левое и правое поддерево и методы - повороты, вставка, удаление и окраска.

Класс **TRTree** представляет само красно-черное дерево, которое содержит указатель на корень дерева и методы - вставка, поиск, удаление, соранение, загрузка и очистка.

Дневник отладки

Изначально выполнял все, что нужно было по условию. Однако после было сказано не обрабатывать ошибки, и мною было принято решение написать программу на C++. Так как запись и чтение из файла производится легче чем на C. Также решил прописать вывод в функциях, а не в мейне.

Тест производительности

Буду использовать разное количество добавляемых элементов в дерево.

Для 100 элементов:
- Find - 326ms
- Insert - 453ms
- Delete - 381ms

Для 1к элемнтов:
- Find - 8248ms
- Insert - 3303ms
- Delete - 2352ms

Выводы

Выполнив вторую лабораторную работу, я познакомился с такой тяжелой для меня реализации в структурных данных, как красно-черное дерево. Данное задание заставило меня посмотреть по-новому на то как хранить и использовать информацию.