Лабораторная работа № 2 по курсу дискретного анализа: Словарь

Выполнил студент группы 08-207 МАИ *Хренов Геннадий.*

Условие

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.

2. Вариант: Красно-чёрное дерево.

Метод решения

Для решения задачи был создан класс RB-дерево, содержащий все основные функции работы с деревом. Дерево устроенно так, что все листья(и корень) указывают на особый узел Tnil, который из всех данных имеет только цвет - черный. Структура узла дерева состоит из ключа, значения, цвета, а также указателей на родителя и двух сыновей. Вставка производится как в обычном бинаром дереве, однако новый элемент получает красный цвет и выполняется операция FixInsert, сохраняющая свойства красно-черного дерева. В зависимости от цвета дяди нового узла при восстановлении нужно рассмотреть 3 случая(и 3 зеркальных им). Аналогично происходит и для удаления. При восстановлении в зависимости от цвета брата и сыновей брата удаляемого узла возникают 4 случая, которые нужно рассмотреть(и 4 зеркальных им).Для сохранение в файл используется обход дерева КЛП, при этом указывается, есть ли у узла левый и правый сын.
Брал информацию из:
Кормен - Алгоритмы: построение и анализ.

Описание программы

Программа состоит из двух файлов: rbtree.h и lab2.cpp. RBTree.h - програмная библиотека, реализующая красно-черное дерево. Основные функции:
RBInsert - добавление элемента в дерево
RBDelete - удаление элемента из дерева
RBFind - поиск элемента в дереве
TreeSave - сохранение дерева в файл
TreeLoad - загрузка элемента из файла

Clean - удаление всех элементов дерева

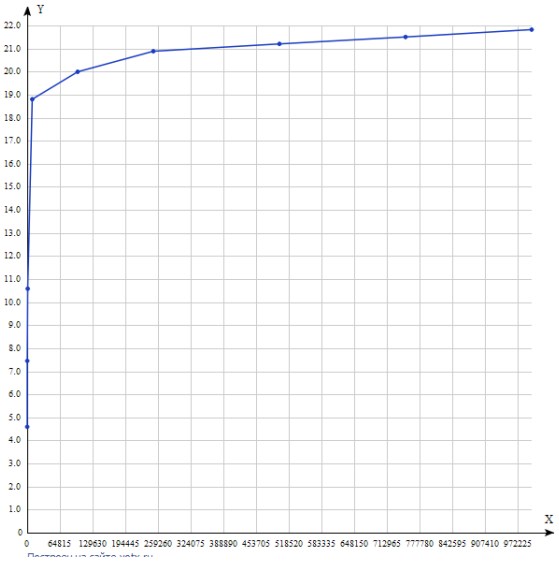
lab2.cpp - прописан интерфейс для работы с деревом
функция Low - убирает разницу в регистрах букв

Дневник отладки

1-3 - ошибка компиляции - Исправил ошибку в функции DeleteFix, исправил неправильное освобождение памяти.
4 - неправильный ответ - забыл выводить "ок"после Save и Load.
5 и далее - превышено реальное время работы - пробовал заменять на scanf, отключение синхронизации, избавился от лишних копирований при передаче, но главное - заменил свой неоправданно долго работающий алгоритм сравнения строк на обычный.

Тест производительности

количество элементов в дереве; время на млн. операций(мс)
(10;4.59)
(100;7.45)
(1000;10.58)
(10000;18.80)
(100000;19.99)
(250000;20.88)
(500000;21.20)
(750000;21.50)
(1000000;21.88)



На графике примерно просматривается логарифмическая сложность.

Недочёты

Для хранения цвета узла использовал int, хотя достаточно одного бита. Не использовал шаблоны, поэтому дерево будет работать только с типами ключей и значений из условия.

Выводы

Красно-черное дерево - неплолая модификация бинарного дерева. Засчет дополнительной информации в узле(цвета) и красно-черных свойств, дерево балансируется при вставке и удалении. Из плюсов можно отметить всего один бит дополнительной памяти и довольно эффективную баллансировку. Из минусов - сложность реализации. Красно-чёрные деревья являются наиболее активно используемыми на практике. Многие контейнеры STL типа map, set и т.д. основаны на красно-чёрных деревьях.