Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

# Лабораторная работа №2 по курсу «Дискретный анализ»

Студент: С. Я. Симонов
Преподаватель: А. А. Кухтичев
Группа: М8О-206Б
Дата:
Оценка:
Подпись:

Москва, 2020

# Лабораторная работа №2

**Задача:** Не обходимо 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 тотже номер.

**Вариант структуры:** АВЛ-дерево.

# 1 Описание

АВЛ-дерево – это двоичное дерево поиска, особенностью которого является то, что оно является сбалансированным в следующем смысле: для любого узла дерева высота его правого поддерева оличается от высоты левого поддерева не более чем на единицу. Доказанно, что этого свойства достаточно для того, чтобы высота дерева логарифмически зависела от числа его узлов: высота h АВЛ-дерева с n дежит в диапазоне от $log_2(n+1)$ до $1.44 * log_2(n+2) - -0.328$. А так как основные операции над двоинми деревьями поиска (поиск, вставка и удаление узлов) линейно зависят от его высоты, то получаем гарантированную логариифмическую зависимость времени работы этих алгоритмов от числа ключей, хранимых в дереве.

В процессе добавления и удаления узлов в АВЛ-дереве возможно возникновение ситуации, когда balance factor некотороых узлов оказывается равным 2 или -2 т.е. возникает расбалансировка поддерева. Для исправления ситуации применяются хорошо нам изветные повороты вокруггех или иных узлов дерева.

Сохранение и запись словаря осуществляется при помоци индексации вершин, и последовательной записи их в бинарном компактном представлении.

# 2 Исходный код

| main.cpp | |
|---|---|
| char ToLower(char ch) | Возвращает символ строки в нижнем регистре |
| bool IsLetter(char ch) | Возврщает true, если символ, переданной функции, является буквой английского алфавита, иначе false. |
| struct TNnode | |
| uint8_t height | Высота узла |
| TData *data | Данные узла |
| TNode *left | Ссылка на левого сына |
| TNode *right | Ссыдка на правого сына |
| class TTree | |
| TTree() | Конструктор по умолчанию, обеспечивает пустое дерево |
| void Insert(TData data) | Втавка элемента в дерево |
| void Delete(TData data) | Удаление элемента из дерева |
| bool Search(TData data) | Возвращает true если элемент в дереве найден, иначе false |
| bool SearchSub(TNode *node, TData data) | Функция помощник, для поиска элемента в дереве, возвращает true, если элемент был найден в дереве, иначе false |
| bool Save(const char* filename) | Сохранение дерева на диск, возвращает true, если успешно, иначе false |
| bool SaveSub(FILE *f, TNode *node) | Функция помощник для сохранения дерева на диск, возвращает true, если успешно, иначе false |
| bool Load(const char* filename) | Загрузка дерева с диска, возвращает true, если успешно, иначе false |
| bool LoadSub(FILE *f, TNode *node) | Функция помощник для загрузки дерева с диска, возвращает true, если успешно, иначе false |
| void Erase() | Удаление дерева |
| Tnode *root | Ссылка на корень дерева |
| TNode *CreateNode(TData data) | Создание листа с данными |
| void DeleteNode(TNode *node) | Удаление узла из дерева |
| TNode *DleteSub(TNode *node, TData data) | Функция помощник, для удаления узла из дерева |

| | |
|---|---|
| TNode *InsertSub(TNode *node, TData data) | Функция помощник, для вставки элемента |
| TNode *Balance(TNode *node) | Функция балансировки узла, возвращает ссылку на сбалансированный узел |
| TNode *RotateLeft(TNode *q) | Поворот дерева налево |
| TNode *RotateRight(TNode *q) | Поворот дерева направо |
| void SetHeight(TNode *node) | Установить или обновить высоту узла |
| TNode *Min(TNode *node) | Вернуть минимальный узел из переданного узла дерева |
| TNode* DelMin(TNode *node) | Удалить минимальный узел в дереве вернуть изменное дерево |
| int BalFact(TNode *node) | Вернуть баланс фактору узла |
| int GetHeight(TNode *node) | Вернуть высоту узла |

# 3 Тест производительности

Реализованный алгоритм сравнивается с std::map, в котором ключом будет std::string, а значением unsigned long long.

Для добавления в std::map используется метод insert(std::make_pair(string, value)). Он возвращает пару значений: итератор на добавленный элемент и булевское значение – истина, если вставка прошла успешно, и ложь, если такой уже присутствует. Для удаления из std::map используется метод erase(string), который возвращает количество удаленнх элементов. Для поска в std::map ипользуется метод find(string). Он возвращает итератор на найденный элемент, либо итератор указывающий на конец контейнера – end().

1000 команд:

```
sergey@sergey-RedmiBook-14: /labs/DA/lab2$ g++ benchmark.cpp -o ben
sergey@sergey-RedmiBook-14: /labs/DA/lab2$ g++ ll2.cpp -o l2
sergey@sergey-RedmiBook-14: /labs/DA/lab2$ ./ben <test.txt >b_log
sergey@sergey-RedmiBook-14: /labs/DA/lab2$ ./l2 <test.txt >l_log
sergey@sergey-RedmiBook-14: /labs/DA/lab2$ diff l_log b_log
974,975c974,975
<Container: avl-tree
<runtime = 41.504
- - -
>Container: map
>runtime = 66.88
```

10000 команд:

```
sergey@sergey-RedmiBook-14: /labs/DA/lab2$ g++ benchmark.cpp -o ben
sergey@sergey-RedmiBook-14: /labs/DA/lab2$ g++ ll2.cpp -o l2
sergey@sergey-RedmiBook-14: /labs/DA/lab2$ ./ben <test.txt >b_log
sergey@sergey-RedmiBook-14: /labs/DA/lab2$ ./l2 <test.txt >l_log
sergey@sergey-RedmiBook-14: /labs/DA/lab2$ diff l_log b_log
9824,9825c9824,9825
974,975c974,975
<Container: avl-tree
<runtime = 1456.72
- - - >Container: map
>runtime = 3155.48
```

# 4 Выводы

Выполнив вторую лабораторную работу по курсу «Дискретный анализ», я изучил различные структуры вида дерево, осознал их преимущества и недостатки, написал програмную библиотеку, реализующую сложный тип данных, тщательно изучил и отладил ее работу. Я попробовал на практике бинарный ввод и вывод данных в C++, работу с файлами, функции fread, fwrite, fopen.

# Список литературы


[1] *Роберт Седжвик, Чарльз И.Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн* Фундаментальные алгоритмы на C++ – Издательство «DiaSoft», 2001. – 688 с.(ISBN 966-7393-89-5)

[2] *АВЛ-деревья – Habr*
URL: `http://habr.com/post/150732`.

[3] *std::map - cppreference.com*
URL: `http://en.cppreference.com/w/cpp/container/map`.