****

|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ « Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

**ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №7 ПО ТиСД**

**Сбалансированные деревья, хеш–таблицы**

Группа ИУ7-34Б

Название предприятия **НУК ИУ МГТУ имени Н. Э. Баумана**

|  |  |  |
| --- | --- | --- |
| Студент |  | Гареев Георгий Антонович |
| Преподаватель |  | Барышникова Марина Юрьевна |

## Описание условия задачи

Техническое задание

Построить хеш-таблицу для слов текстового файла (задача №6). Осуществить поиск указанного слова в двоичном дереве поиска (ДДП) и в хеш-таблице, если его нет, то добавить его (по желанию пользователя) в дерево и, соответственно, в таблицу. При необходимости использовать реструктуризацию таблицы. Сбалансировать дерево. Сравнить время поиска, объем памяти и количество сравнений при использовании ДДП, сбалансированных деревьев и хеш-таблиц.

Исходные данные

Номер команды меню.

Описание задачи, реализуемой программой

Строит ДДП, АВЛ деревья, закрытую и открытую хеш-таблицы. Осуществляет поиск слова и удаление элемента. Вывод дерева в графическом виде и хеш-таблиц в символьном.

Способ обращения в программе

Пользователю выдается меню, в котором он должен выбрать номер функции 0-3:

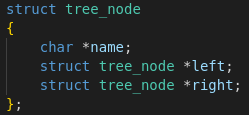
* 0 – выход из программы
* 1 – чтение слов из файла.
* 2 – добавление слова в дерево.
* 3 – удаление слова из дерева.
* 4 – поиск слова в дереве.
* 5 – вывод слов из дерева в инфиксном порядке.
* 6 – поиск слов, начинающихся с введенного символа.
* 7 – запись дерева в файл для последующей визуализации.
* 11 – строит хеш-таблицы из слов, читаемых из файла.
* 12 – выводит таблицы на выбор.
* 13 – Ищет слово в таблицах и выводит кол-во сравнений.
* 14 – удаляет слово из таблицы
* 21 – замерный эксперимент

Описание возможных аварийных ситуаций и ошибок пользователя

1. Динамическая память не сможет выделится.
2. Введенный путь к файлу неправильный.
3. Попытка удалить элемент их пустого дерева/таблицы.

Описание внутренних СД

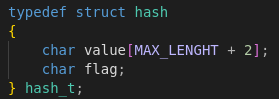
Под вершину деревьев выделяется структура следующего содержания:



Char \*name – указатель на слово (содержание вершины);

Struct tree\_node \*left(right) – указатель на левое (правое) поддерево.

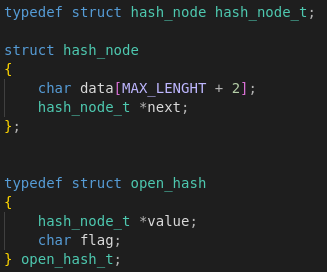
Для хранения закрытой хеш-таблицы выделен статический массив из структур:



Flag – индикатор заполненности;

Value – строка, хранимая в таблице.

Для хранения открытой хеш-таблицы выделен статический массив из структур open\_hash\_t:



В value хранится указатель на связный список, содержащий строку и указатель на следующий элемент.

## Описание алгоритма

При выборе 0 пункта меню (выход из программы) программа завершается с освобождением всей использованной динамической памятью.

При выборе 1 пункта меню пользователь должен ввести путь к читаемому файлу, после чего слова из файла добавляются пустое дерево, если файл нечитабелен или не существует – выдается сообщение об ошибке.

При выборе 2 пункта меню пользователь вводит слово, и оно добавляется в дерево, если такое уже существует, то ничего не происходит.

При выборе 3 пункта меню пользователь вводит слово, и оно удаляется из дерева, если такого слова нет, то ничего не происходит.

При выборе 4 пункта меню пользователь вводит слово, и в дереве происходит его поиск, в качестве результат выводится сообщение о его присутствии в дереве.

При выборе 5 пункта меню слова из дерева выводятся на экран в инфиксном порядке.

При выборе 6 пункта меню пользователь вводит символ, после чего происходит поиск слов, начинающихся с данного символа.

При выборе 7 пункта меню происходит запись дерева в файл ‘.gv’ для последующей визуализации с помощью плагина graphviz.

При выборе 11 пункта меню происходит построение хеш-таблиц с функцией, выбранной пользователем.

При выборе 12 пункта меню на выбор пользователя выводится одна из таблиц.

При выборе 13 пункта меню происходит поиск слова в таблицах и выводится сообщение с кол-вом проведенных сравнений.

При выборе 14 пункта меню введенное слово удаляется из таблиц.

При выборе 21 пункта меню происходи сравнение времени поиска слов в деревьях и таблицах в зависимости от кол-ва слов.

## Тесты

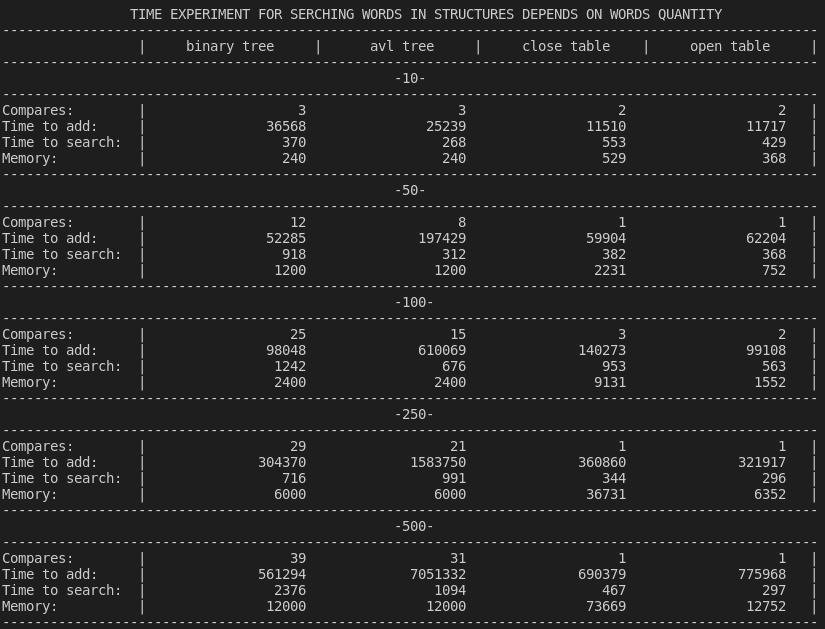
Позитивные

1. Вывод пустого дерева/таблицы
2. Поиск в пустом дереве/таблице
3. Удаление и добавление различных ветвей
4. Сравнение времени

Негативные

1. Неправильный путь к файлу
2. Удаление из пустого дерева/таблицы

## Оценка эффективности



Выше приведена таблица, а которой память измеряется в байтах, а время в тиках процессора.

Как можно заметить, по памяти всегда выигрывают деревья – память, выделяемая под вершины меньше, чем под элементы таблицы.

Однако в количестве сравнений хеш-таблицы однозначно выигрывают: кол-во сравнений в них не может быть больше, чем указано в программе (по умолчанию 4), когда как в дереве слово ищется рекурсивно, пока не найдется. В частности, про деревья можно сказать, что кол-во сравнений в АВЛ дереве в среднем меньше, чем в обычном – оно и понятно, длина левого и правого поддерева отличаются не более, чем на один.

Время добавления у ддп и хеш-таблиц примерно одинаково по сравнению со временем добавления в АВЛ дерево, поскольку требуется дополнительное время на перебалансировку.

Время поиска напрямую связано с кол-вом сравнений.

## Вывод

Для задачи поиска следует выбираться структуру данных, исходя из приоритетного ресурса. Сложно выбрать какую-то конкретную, т.к. всегда можно изменить параметры хеш-таблиц. Если важно время поиска, то следует использовать хеш-таблицы, поскольку макс. кол-во сравнений всегда можно регулировать и при желании, если память не важна, то можно установить параметр на 1-2.

## Ответы на контрольные вопросы

1. **Чем отличается идеально сбалансированное дерево от АВЛ дерева?**

Если при добавлении узлов в дерево мы будем их равномерно располагать слева и справа, то получится дерево, у которого число вершин в левом и правом поддеревьях отличается не более, чем на единицу. Такое дерево называется идеально сбалансированным.

Двоичное дерево называется сбалансированным, если у каждого узла дерева высота двух поддеревьев отличается не более чем на единицу. Такое дерево называется АВЛ-деревом.

1. **Чем отличается поиск в АВЛ-дереве от поиска в дереве двоичного поиска?**

Нужные данные найдутся в худшем случае за такое же кол-во времени, а в лучшем сильно быстрее, поскольку в АВЛ дереве высоты поддеревьев отличаются не более чем на 1, благодаря чему поиск может занять заметно меньше времени.

1. **Что такое хеш-таблица, каков принцип ее построения?**

Это массив, в ячейках которого находятся данные, номер, полученный из хеш-функции, которых совпадает с индексом ячейки.

1. **Что такое коллизии? Каковы методы их устранения.**

Может возникнуть ситуация, когда разным ключам соответствует одно значение хеш-функции, то есть, когда h(K1) = h(K2), в то время как K1 ≠ K2. Такая ситуация называется коллизией.

1. **В каком случае поиск в хеш-таблицах становится неэффективен?**

Когда кол-во коллизий превышает некоторый допустимый порог.

Тогда кол-во вынужденных сравнений увеличивается.

1. **Эффективность поиска в АВЛ деревьях, в дереве двоичного поиска, в хештаблицах и в файле.**

Эффективность поиска слева направо в порядке эффективности:

Хеш-таблица -> АВЛ дерево -> ДДП -> файл