Министерство образования и науки 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 ПО ЛАБОРАТОРНОЙ РАБОТЕ 6

По дисциплине «Типы и структуре данных»

Название **«Деревья»**

Студент Дубов Андрей Игоревич
фамилия, имя, отчество
Группа ИУ7-33Б

Вариант 5

Тип лабораторной работы Учебная

Студент _____ Дубов А. И.
подпись, дата *фамилия, и.о.*

Преподаватель Рыбкин Ю. А.
_____ Силантьева А. В.
подпись, дата *фамилия, и.о.*

2022 г.

Оглавление

Условие задачи

Построить хеш-таблицу по указанным данным. Сравнить эффективность поиска в сбалансированном двоичном дереве, в двоичном дереве поиска и в хеш-таблице (используя открытую и закрытую адресацию). Вывести на экран деревья и хеш-таблицу. Подсчитать среднее количество сравнений для поиска данных в указанных структурах. Произвести реструктуризацию хеш-таблицы, если среднее количество сравнений больше указанного. Оценить эффективность использования этих структур (по времени и памяти) для поставленной задачи. Оценить эффективность поиска в хеш-таблице при различном количестве коллизий и при различных методах их разрешения.

Описание технического задания

Построить хеш-таблицу для слов текстового файла. 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. Сравнить эффективность добавления ключа в таблицу или ее реструктуризацию для различной степени заполненности таблицы.

Входные данные:

Файл с текстом.

Выходные данные:

Дерево, сбалансированное дерево, хэш-таблица, информация о частотности слова.

Аварийные ситуации:

1. Некорректный ввод номера команды.

Описание структуры данных

Структура дерева

```
typedef struct tree
{
    char word[100];
    int key;
    struct tree *left;
    struct tree *right;
    struct tree *parent;
} node_t;
```

word – слово ветки

key – количество повторений
left – указаетель на левое поддерево
right – указаетель на правое поддерево
parent – указаетель на поддерево родителя

Структура открытого хэширования

```
typedef struct item
{
    char word[100];
    int key;
    struct item *next;
    struct item *parent;
} item_t;
```

word – слово ветки
key – количество повторений
next – указатель на следущий элемент списка
parent – указаетель на поддерево родителя

Структура закрытого хэширования

```
typedef struct item
{
    char word[100];
    int key;
} item_t;
```

word – слово ветки
key – количество повторений

Описание алгоритма

1. В меню предлагается поработать деревом бинарного поиска, его сбалансированной версией и работать с хэштаблицами двух типов. Деревья и хэштаблицы инициализируются при старте программы.
2. Пока пользователь не введет 0 (выход из программы), ему будет предложено вводить номера команд и выполнять действия по выбору.

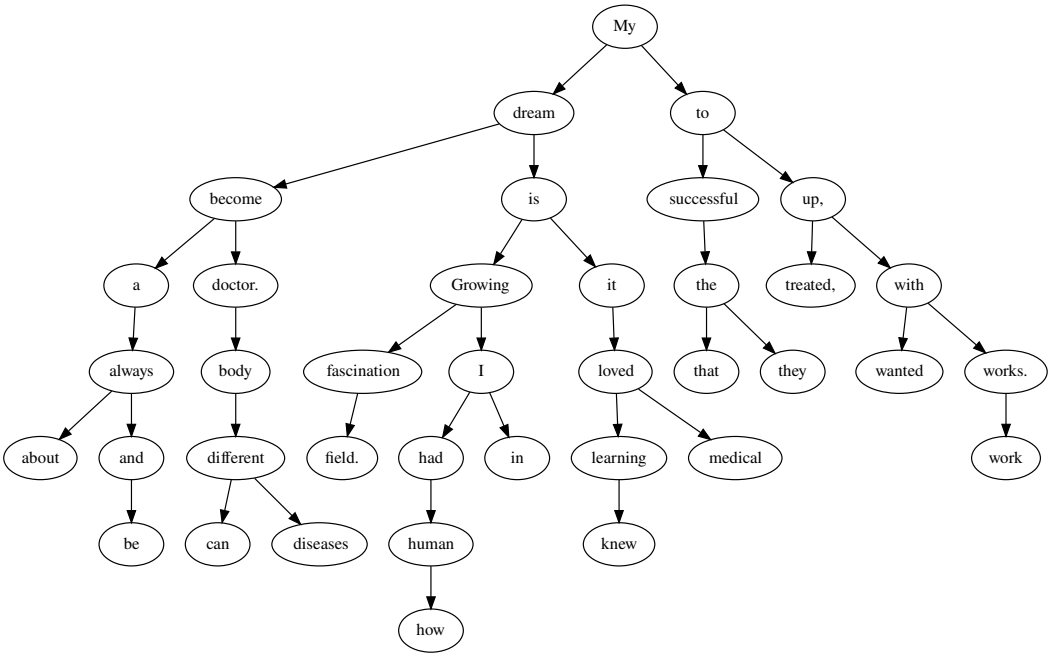
```
static int get_hash(char *word)
{
    long long sum = 7;
    for (int i = 0; word[i] != '\0'; i++)
        sum = (sum % CAPACITY) * mult  + word[i];
    return sum % CAPACITY;
}
```

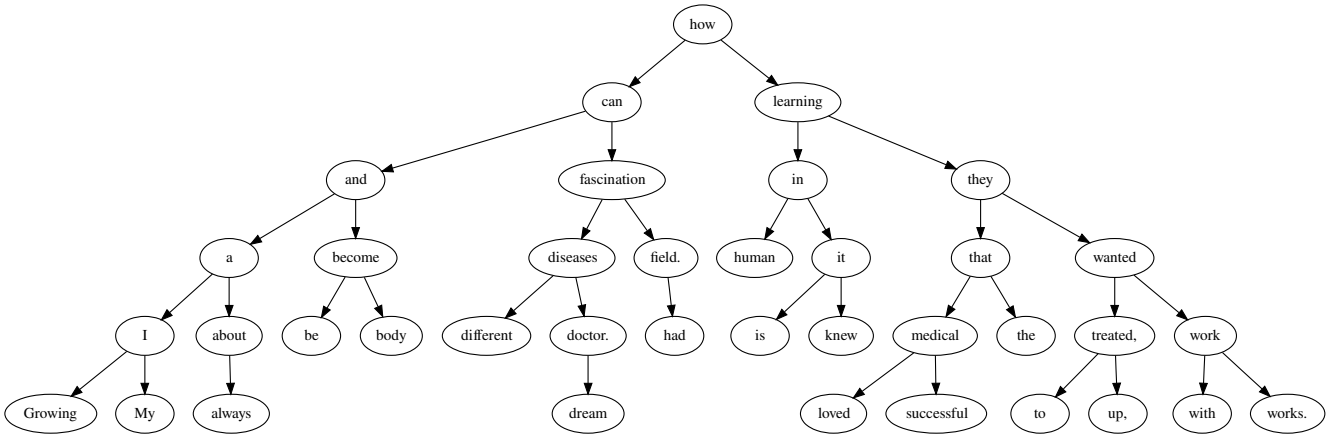
mult — глобальная переменная, которая используется для реструктуризации таблицы
CAPACITY — макрос размера таблицы

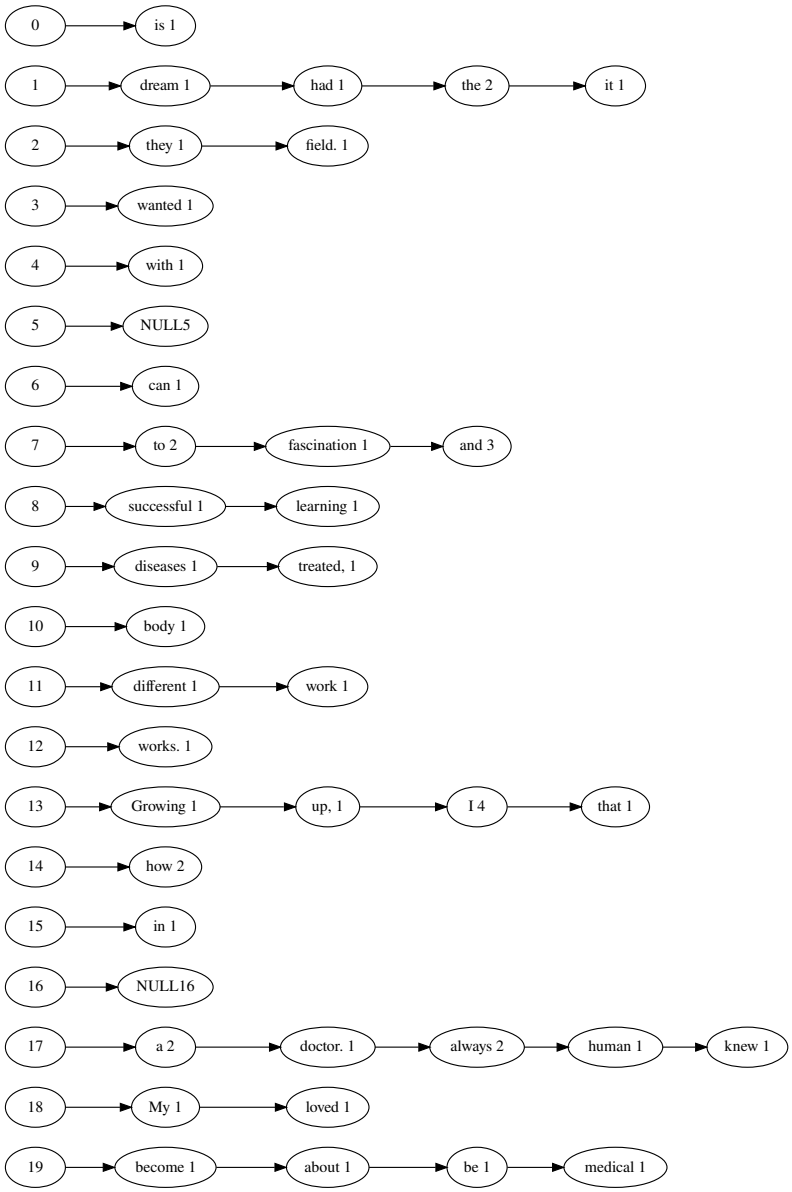
Набор тестов

	Название теста	Пользователь вводит	Вывод
1	Некорректный ввод команды	45	No such option or wrong input
2	Пустой ввод	Пустой ввод.	No such option or wrong input

3	Команда 0	0	Выход из программы







Оценка эффективности

Время поиска и количество сравнений во всех структурах

35 уникальных слов, прогонялось 10 раз, время в тиках

Среднее время поиска для обычного дерева 1842, для сблансированного дерева 1653, для открытой таблицы 479, для закрытой таблицы 686. Для дерева количества сравнений 5.54, для сбалансированного дерева количество сравнений 4.49, для открытой таблицы количество сравнений 1.14, для закрытой таблицы 11.57.

Время добавления и количество сравнений во всех структурах

50 уникальных слов, прогонялось 5 раз, время в тиках. В изначальной таблице 35 уникальных слов. Размер таблиц 20

Среднее время поиска для обычного дерева 1344, для сблансированного дерева 22062, для открытой таблицы 343, для закрытой таблицы 561. Для дерева количества сравнений 7.22, для сбалансированного дерева количество сравнений 6.04, для открытой таблицы количество сравнений 1.78, для закрытой таблицы 20.

Если увеличить размер таблиц до 50

Для открытой таблицы 331, для закрытой таблицы 435. Для открытой таблицы количество сравнений 0.76, для закрытой таблицы 6.02.

Время поиска в таблицах

0 коллизий 14 слов

Для открытой таблицы 353, для закрытой таблицы 386. Сравнений 0

Коллизий больше половины слов 40 слов

Для открытой таблицы 534, для закрытой таблицы 589. Сравнений для открытой 1.25, для закрытой 3.38

Меньше половины слов коллизий 120 слов

Для открытой таблицы 432, для закрытой таблицы 617. Сравнений для открытой 0.48, для закрытой 6.76

Количество затраченной памяти

Размер таблиц 20

Количество элементов	Дерево, байт	Сбалансированное дерево, байт	Хэш-таблица открытая, байт	Хэш-таблица закрытая, байт
100	13600	13600	12008	2088
1000	136000	136000	120008	2088
10000	1360000	1360000	1200008	2088

Вывод

Время поиска в таблицах на порядок быстрее чем в деревьях. Поиск в сбалансированном дереве в среднем быстрее, поскольку высота дерева меньше. А вот добавление намного медленнее, поскольку требуется огромное количество времени на перестроение дерева. Таблицы спрпалвяются с этими задачами намного лучше и удобнее использовать конечно же их. Но вот

использование таблицы с закрытым хэшированием под большим вопросом. Во первых, количество мест для элементов может просто не хватить, а если даже хватит то время поиска будет больше. Усредненное количество сранений сильно уступает открытому хэшированию, потому что может получится так, что единственное свободное место находится на ячейку выше и придется идти через всю таблицу по кругу, хотя в открытом хэшировании возможно даже не надо больше ничего сравнивать.

Ответы на контрольные вопросы

1) Чем отличается идеально сбалансированное дерево от АВЛ дерева?

   Дерево, у которого число вершин в левом и правом поддеревьях отличается не более, чем на единицу называется идеально сбалансированным. Двоичное дерево, у каждого узла которого высота двух поддеревьев отличается не более чем на единицу называется АВЛ-деревом.

2) Чем отличается поиск в АВЛ-дереве от поиска в дереве двоичного поиска?

   Поиск в АВЛ дереве имеет сложность O(log2n), в то время как в обычном ДДП сложность O(n).

3) Что такое хеш-таблица, каков принцип ее построения?

   Массив, элементы в котором распределяются в зависимости от Хеш-функции. Минимальная трудоемкость поиска в хеш-таблице равна O(1).

4) Что такое коллизии? Каковы методы их устранения.

   Коллизия – ситуация, когда разным ключам соответствует одно значение хеш-функции. Существует несколько возможных вариантов разрешения коллизий: внешнее (открытое) хеширование (метод цепочек) и внутреннее (закрытое) хеширование (открытая адресация).

5) В каком случае поиск в хеш-таблицах становится неэффективен?

   Поиск в хэш-таблице становится неэффективным при большом числе коллизий – сложность поиска возрастает.

6) Эффективность поиска в АВЛ деревьях, в дереве двоичного поиска и в хеш- таблицах

   В хэш-таблице минимальное время поиска O(1).

   В АВЛ дереве O(log2n).

   В дереве двоичного поиска O(h), где h – высота дерева.