Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский государственный технический университет имени Н.Э. Баумана

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №6

«Деревья, хеш-таблицы»

Студент Фролов Евгений

Группа ИУ7 – 35Б

2020 г.

Цель работы

Построить ДДП, сбалансированное двоичное дерево (АВЛ) и хеш-таблицу по указанным данным. Сравнить эффективность поиска в ДДП в АВЛ-дереве и в хеш-таблице (используя открытую или 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.

Условие задачи (4 вариант)

Построить ДДП, в вершинах которого находятся слова из текстового файла. Вывести его на экран в виде дерева. Сбалансировать полученное дерево и вывести его на экран. Удалить указанное слово в исходном и сбалансированном дереве. Сравнить время удаления и объём памяти. Построить хеш-таблицу из слов текстового файла, задав размерность таблицы с экрана, используя метод цепочек для устранения коллизий. Вывести построенную таблицу слов на экран. Осуществить удаление введённого слова, вывести таблицу. Сравнить время удаления, объём памяти и количество сравнений при использовании ДДП, сбалансированных деревьев, хеш-таблиц и файла.

Входные данные

Пункт в меню;

Файл с данными, элемент, который требуется удалить;

Выходные данные

Вывод в консоль обычного дерева/сбалансированного дерева, созданную хеш-таблицу. Эффективность работы: время работы и кол-во сравнений. Вывод деревьев и хеш-таблицы.

Возможные аварийные ситуации

Некорректный ввод.

Функция программы:
Загрузка обычного/сбалансированного дерева, хеш-таблицы из файла. Удаление элементов их этих структур.

Обращение к программе: через консоль командой ./main.exe.

Структуры данных

Структура хеш-таблицы

```
typedef struct hash_t {
    node_h **data;
    int base;
} hash_t;
```

Структура узла АВЛ-дерева и двоичного. Используется одна структура.

```
typedef struct node_t {
    unsigned int height; //высота узла
    struct node_t *left; //левое поддеревье
    struct node_t *right; //правое поддеревье
    char *value; //указатель на строку с данными
} node_t;
```

В качестве хеш-функции ключом является остаток от деления суммы кодов символов строки на размерность массива.

Тесты

При неправильном вводе – выведется ошибка.
При удалении несуществующего слова – выведется ошибка

Вывод дерева

```
Input: 2

         |          1: qwe
         |          |          2: qqw
         0: qqq
         |          1: qew
         |          |          2: eee

MENU
0. Exit

1. Delete word
2. Show binary tree
3. Show AUS-tree
4. Show hash-table
5. Delete comparison
6. Rebase

Input:
```

Удаление элемента «eee»

```
Word to delete: eee
SUCCESS

MENU
0. Exit

1. Delete word
2. Show binary tree
3. Show AUS-tree
4. Show hash-table
5. Delete comparison
6. Rebase

Input:
```

Вывод хеш-таблицы

```
Input: 4

|   HASH   |          VALUE          |
|      2   |                     qqw |
|      3   |                     qqq |
|      4   |                     qwe |
|      4   |                     qew |
```

Хеш-таблица.

Для каждого элемента выделяется узел. В узле хранятся данные и указатель на следующий элемент + указатель на данные. Для хранения массива указателей уходит M / N * 8 байт памяти,

M — количество элементов в таблице.

N – кол-во коллизии = 4

M * 16 + M / N * 8 (байт)

Дерево.

Память отводится на хранение узлов. Указатель на данные и 2 указателя на ветви = 24 байт. 24 * M (байт)

В АВЛ дереве хранятся данные о высоте узла. 28 * M (байт)

Память СД

Количество элементов	Хеш-таблица	Дерево	АВЛ-дерево
100	1800	2400	2800
1000	18000	24000	28000
10000	180000	240000	280000

Контрольные вопросы

1. Что такое дерево?

Дерево – это нелинейная структура данных, используемая для представления иерархических связей, имеющих отношение «один ко многим».

2. Как выделяется память под представление деревьев?

Выделение памяти под деревья определяется типом их представления. Это может быть таблица связей с предками или связный список сыновей. Оба представления можно реализовать в виде матрицы или списка. При реализации списком память выделяется динамически, при реализации матрицей статически.

3. Какие стандартные операции возможны над деревьями?

Поиск, включение в дерево, исключение из дерева, обход.

4. Что такое дерево двоичного поиска?

Дерево двоичного поиска – это такое дерево, в котором все левые потомки моложе предка, а все правые – старше.

5. Чем отличается идеально сбалансированное дерево от АВЛ-дерева?

В АВЛ-дереве критерий сбалансированности «высота 2 поддеревьев отличается не более, чем на 1», а в идеальном сбалансированном дереве - «количество элементов в левом и правом поддеревьях отличается не больше, чем на 1».

6. 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).

7. Что такое хеш-таблица, каков принцип её построения?

Массив, заполненный в порядке, определённым хеш-функцией, называется хеш-таблицей. Хэш-функция — функция, ставящая в соответствие какому-то объекту уникальный хэш (индекс). Хэш-функция находит некоторый «адрес» в хэш-таблице по значению элементов и помещает их по тому адресу. Таким образом, чтобы получить адрес элемента, нужно просто рассчитать значение функции.

8. 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).

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

При большом количестве коллизий или при крайне малом объёме данных.

10. 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.

Вывод: По результатам работы, можно сделать вывод, что если мы ограничены по памяти, то лучше использовать файл.
Если в памяти мы не сильно ограничены, то максимальный выигрыш по времени удаления и времени доступа дает хеш-таблица.
При плохой хеш-функции она может проиграть по времени доступа АВЛ-дереву. АВЛ-дерево требует постоянной балансировки, очень долго обрабатывает удаление элементов.