Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«Московский государственный технический университет имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ _____ИУ_____

КАФЕДРА _____ИУ7_____

Отчет

по лабораторной работе № 6

Дисциплина: Типы и Структуры Данных

Название лабораторной работы: Двоичные деревья

Студент гр. **ИУ7-34Б Малышев Илья Николаевич**

(Подпись, дата) (И.О. Фамилия)

Преподаватель **Силантьева Александра Васильевна**

(Подпись, дата) (И.О. Фамилия)

Москва, 2021

Цель работы

Построить дерево, вывести его на экран в виде дерева, реализовать основные операции работы с деревом: обход дерева, включение, исключение и поиск узлов, сбалансировать дерево, сравнить эффективность алгоритмов сортировки и поиска в зависимости от высоты деревьев и степени их ветвления; построить хеш-таблицу и вывести ее на экран, устранить коллизии, если они достигли указанного предела, выбрав другую хеш-функцию и реструктуризировав таблицу; сравнить эффективность поиска в сбалансированных деревьях, в двоичных деревьях поиска (ДДП), в хеш-таблицах и в файлах. Сравнить эффективность реструктуризации таблицы для устранения коллизий и поиска в ней с эффективностью поиска в исходной таблице.

Условия задачи

В текстовом файле содержатся целые числа. Построить ДДП из чисел файла. Вывести его на экран в виде дерева. Сбалансировать полученное дерево и вывести его на экран. Построить хеш-таблицу из чисел файла. Использовать закрытое хеширование для устранения коллизий. Осуществить удаление введенного целого числа в ДДП, в сбалансированном дерево, в хеш-таблице и в файле. Сравнить время удаления, объем памяти и количество сравнений при использовании различных (4-х) структур данных. Если количество сравнений в хеш-таблице больше указанного, то произвести реструктуризацию таблицы, выбрав другую функцию.

Техническое задание

Входные данные:

- Целое число, которое обозначает действие, которое должна выполнить программа (смотри “Задача программы”).
- Запрашиваемые данные, необходимые для выполнения некоторых действий;

Выходные данные:

- Представления используемых структур;
- Время обработки при для этих структур;
- Сгенерированная статистика по обработке этих структур.

Задача программы:

Программа позволяет выполнить перечисленные ниже операции:

0. Выход;
1. Добавление элемента в структуры;
2. Поиск элемента по структурам;
3. Удаление элемента по ключу из структур;
4. Вывод структур;
5. Реструктуризация хеш-таблицы;
6. Генерация статистики по использованию структур.

Способ обращения к программе:

Программа является консольной. После запуска исполняемого файла из консоли, программа выведет справку и приглашение на ввод.

Аварийные ситуации:

- Некорректный ввод номера команды;
- Отсутствие требуемых для работы программы текстовых файлов;
- Испорченное содержание файлов.

Структуры данных:

node_t - структура, хранящая информацию о «узле» двоичного дерева.

```
typedef struct node node_t;
struct node
{
  int value;
  node_t *left;
  node_t *right;
  int height;
};
```

element_t – структура, хранящая информацию о элементе хеш-таблицы.

```
typedef struct element element_t;
struct element
{
  int value;
  int free;
};
```

table_t- структура, отражающая хеш-таблицу.

```
typedef struct table table_t;
struct table
{
  element_t *data;
  int allocated;
  int elem_amount;
  int factor_number;
  int collision;
};
```

structures_t – вспомогательная структура, хранящая указатели на используемые структуры.

```
typedef struct
{
  node_t *casual_tree;
  node_t *balanced_tree;
  table_t *table;
  text_file_t *text_file;
} structures_t;
```

text_file_t – структура, хранящая информацию о текстовом файле.

```
typedef struct
{
  char *filename;
  int len;
  int *data;
} text_file_t;
```

Тесты

#	Ошибка	Пример
1	**Неправильный ввод команды**	**q**
2	**Неправильный ввод команды**	**20**
3	**Неправильное содержание файла**	**Пустой файл**
4	**Отсутствие файла.**	**nofile.txt**

Оценка эффективности
Измерения эффективности способов сложения будут производиться в единицах измерения – тактах процессора.

Таблица сравнения памяти

Количество записей	Двоичное дерево поиска	Сбалансированное дерево поиска	Хеш-таблица	Файл
0	0	0	656	24
10	320	320	792	64
50	1600	1600	848	224
100	3200	3200	1216	424
200	6400	6400	1888	824

Таблица скорости добавления в тактах

Количество записей	Двоичное дерево поиска	Сбалансированное дерево поиска	Хеш-таблица	Файл
0	3654	1824	2369	1279351
10	3256	2004	2311	1206650
50	4384	2875	1901	1286719
100	5399	3642	2565	1437227
200	6094	3682	3014	1472205

Таблица скорости удаления в тактах

Количество записей	Двоичное дерево поиска	Сбалансированное дерево поиска	Хеш-таблица	Файл
0	1044	1171	702	407342
10	1084	1135	636	463431
50	1684	1669	852	614064
100	2147	2251	1215	962132
200	2594	2623	1051	1297166

Вывод

Из таблицы памяти можно заключить, что бинарные деревья самые неэффективные, с точки зрения занимаемой памяти. Они требуют, в среднем, на один порядок больше памяти, чем структура, хранящая данные из файла. Хеш-таблица занимает промежуточное положение, смещенное в сторону файла.

С точки зрения скорости обработки, хеш-таблица показывает наилучший результат, бинарные деревья остаются в том же порядке, но отстают с некоторым коэффициентом, обработка текстового файла отстает, в среднем, на 2-3 порядка, что говорит о крайней неэффективности ее, в сравнении с другими структурами.

Контрольные вопросы

1. Что такое дерево?
Дерево – это рекурсивная структура данных, используемая для представления иерархических связей, имеющих отношение «один ко многим».
2. Как выделяется память под представление деревьев?
В виде связного списка — динамически под каждый узел.
3. Какие стандартные операции возможны над деревьями?
Обход дерева, поиск по дереву, включение в дерево, исключение из дерева.
4. Что такое дерево двоичного поиска?
Двоичное дерево поиска - двоичное дерево, для каждого узла которого сохраняется условие: левый потомок больше или равен родителю, правый потомок строго меньше родителя (либо наоборот).
5. Чем отличается идеально сбалансированное дерево от АВЛ дерева?
У АВЛ дерева для каждой его вершины высота двух её поддеревьев различается не более чем на 1, а у идеально сбалансированного дерева различается количество вершин в каждом поддереве не более чем на 1.
6. Чем отличается поиск в АВЛ-дереве от поиска в дереве двоичного поиска?
Поиск в АВЛ дереве происходит быстрее, чем в ДДП.
7. Что такое хеш-таблица, каков принцип ее построения?
Хеш-таблицей называется массив, заполненный элементами в порядке, определяемом хеш-функцией. Хеш-функция каждому элементу таблицы ставит в соответствие некоторый индекс. Функция должна быть простой для вычисления, распределять ключи в таблице равномерно и давать минимум коллизий.
8. Что такое коллизии? Каковы методы их устранения?
Коллизия – ситуация, когда разным ключам хеш-функция ставит в соответствие один и тот же индекс. Основные методы устранения коллизий: открытое и закрытое хеширование. При открытом хешировании к ячейке по данному ключу прибавляется связанны список, при закрытом – новый элемент кладется в ближайшую свободную ячейку после данной.
9. В каком случае поиск в хеш-таблицах становится неэффективен?
Поиск в хеш-таблице становится неэффективен при большом числе коллизий –сложность поиска возрастает по сравнению с O(1). В этом случае требуется реструктуризация таблицы – заполнение её с использованием новой хеш-функции.
10. Эффективность поиска в АВЛ деревьях, в дереве двоичного поиска и в хеш-таблицах.
В хеш-таблице минимальное время поиска O(1). В АВЛ: O(log2n). В дереве двоичного поиска O(h), где h - высота дерева (от log2n до n).