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 (ИУ)

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии (ИУ7)

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №6
«РАБОТА С ДЕРЕВЬЯМИ»

Студент, группа **Артемьев И.О., ИУ7-33Б**

2020 г.

Описание 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 метод цепочек для устранения коллизий. Осуществить поиск введенного целого числа в ДДП, в сбалансированном дереве, в хеш-таблице и в файле. Сравнить время поиска, объем памяти и количество сравнений при использовании различных (4-х) структур данных. Если количество сравнений в хеш-таблице больше указанного (вводить), то произвести реструктуризацию таблицы, выбрав другую функцию.

Техническое задание

Входные данные:

1. **Имя файла с деревом:** строка, содержащая имя файла.
2. **Максимальное значение допустимых коллизий:** целое число.
3. **Число для поиска:** число, которое планируется найти в структурах данных

Выходные данные:

1. Псевдографическое изображение бинарного дерева.
2. Псевдографическое изображение сбалансированного бинарного дерева.
3. Хеш-таблица вершин дерева.
4. Время поиска, объем памяти и количество сравнений при использовании 4-х структур данных.

Функции программы:

Программа выполняет ряд функций в порядке:

1. Заполнить данные из файла.
2. Вывести двоичное дерево поиска.
3. Вывести АВЛ дерево.
4. Вывести хэш-таблицу.
5. Сравнить время поиска, объем памяти и количество сравнений при использовании 4-х структур данных.

1. Выйти.

Обращение к программе: запускается из терминала (./app.exe) с аргументом в виде файла (./app.exe data/filename.txt), содержащего значения вершин дерева.

Аварийные ситуации:

1. Некорректный ввод имени файла.
   На входе: имя файла, несуществующего в системе.
   На выходе: сообщение «Ошибка: неверно введен файл.»
2. Некорректный ввод имени файла.
   На входе: число
   На выходе: сообщение «Ошибка: неверно введен файл.»
3. Некорректный ввод максимального числа коллизий.
   На входе: буква или, любой другой нечисловой символ или отрицательное число.
   На выходе: сообщение «Ошибка: неверно введено количество сравнений для хэш-таблицы.»

Структуры данных

Реализация линейного односвязного списка:

```
typedef struct lnode_t
{
   int field;

   struct lnode_t *next;
} lnode_t;
```

Поля структуры:

- **int field –** значение узла;
- **struct lnode_t *next –** указатель на следующий элемент списка.

Реализация листа дерева:

```
typedef struct tnode_t
{
   int field;
   unsigned char height;

   struct tnode_t *left;
   struct tnode_t *right;
} tnode_t;
```

Поля структуры:

- **int field –** значение листа;
- **unsigned char height** – высота листа;
- **struct tnode_t *left –** указатель на левого потомка;

- **struct tnode_t *right –** указатель на правого потомка.

Реализация динамического массива (для хеш-таблицы):

***lnode_t* **hash_table;**
***int* size;**

Поля:
- ***lnode_t* **hash_table –** указатель на массив указателей на списки;
- ***int* size –** размер массива;

Алгоритм

Пользователь вводит номер команды из меню. Пока пользователь не введет 0 (выход из программы), ему будет предложено выполнять действия на основе файла, деревьев и хэш-таблиц.

Тесты

	Тест	Пользовательский ввод	Результат
1	Некорректный ввод имени файла	data/Iu7 (файл не существует)	Ошибка: неверно введен файл.
2	Некорректный ввод имени файла	data/123 (файл не существует)	Ошибка: неверно введен файл
3	Некорректный ввод максимального количества коллизий	A	Ошибка: неверно введено количество сравнений для хэш-таблицы
4	Некорректный ввод максимального количества коллизий	0	Ошибка: неверно введено количество сравнений для хэш-таблицы
5	Некорректный ввод максимального количества коллизий	-1	Ошибка: неверно введено количество сравнений для хэш-таблицы
6	Ввод несуществующего числа	Несуществующее число	Введенное число не было найдено
9	Корректный ввод всех характеристик	Корректный файл, корректный ввод числа для поиска, корректный ввод числа коллизий	Количественная характеристика моделирования

Оценка эффективности

Поиск числа (тактах процессора):

Количество элементов	Бинарное дерево	Сбалансированное бинарное дерево	Хеш-таблица	Файл
10	1122	623	19	12114
25	1738	785	25	13938
50	2064	1211	30	15134
100	2179	1134	36	18556
500	2438	1463	37	28345

***в таблице указаны средние значения**

***указанные результаты справедливы для хеш-таблиц без коллизий**

Количество сравнений:

Количество элементов	Бинарное дерево	Сбалансированное бинарное дерево	Хэш-таблица	Файл
10	5	3	1	10
25	8	4	1	25
50	10	5	1	50
100	12	6	1	100
500	14	9	1	500

***указанные результаты справедливы для хеш-таблиц без коллизий**

***указанные результаты справедливы для худшего случая – элемент в конце файла**

Объём занимаемой памяти (в байтах):

Количество элементов	Бинарное дерево	Сбалансированное бинарное дерево	Хеш-таблица	Файл
10	240	240	160	21
25	600	600	400	68
50	1200	1200	800	137
100	2400	2400	1600	289
500	12000	12000	8000	1882

***для хэш-таблицы указаны наилучшие объемы памяти, которые она может занимать при заданном количестве элементов без коллизий**
В иных случаях хэш-таблица может разрастаться в памяти до нереальных размеров, что не выгодно.

Пример:

```
Введите число для поиска: 3


Введите количество сравнений для хэш-таблицы (> 0): 3

[13] --> [93] --> [693] --> ...

Превышено количество сравнений

1 - Продолжить поиск элемента в других структурах данных
2 - Реструктуризировать таблицу, выбрав другую хэш-функцию

Ответ: 2

[3]--> NULL

Хэш-таблица:
Время поиска элемента: 27
Объем занимаемой памяти: 1504
Количество сравнений для поиска элемента: 1

Файл:
Время поиска элемента: 17126
Объем занимаемой памяти: 38
Количество сравнений для поиска элемента: 2

Двоичное дерево поиска:
Время поиска элемента: 396
Объем занимаемой памяти: 312
Количество сравнений для поиска элемента: 2

АВЛ дерево:
Время поиска элемента: 804
Объем занимаемой памяти: 312
Количество сравнений для поиска элемента: 4
```

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 дереве двоичного поиска.

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).

Вывод

Использование хеш-таблицы всегда эффективно по времени (при 500 элементах 37 тактов), но не всегда эффективно по памяти (из примера – при 10 элементах занимаемая память равна 1504 байтам). В случае деревьев, АВЛ дерево не всегда выигрывает по времени поиска у несбалансированного дерева, так как порядок вершин при балансировке меняется, но всегда выигрывает по среднему времени поиска по дереву (при 500 элементах АВЛ - 1463 тактов, а ДДП – 2438 тактов).