|  |  |
| --- | --- |
|  | **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)** |

Отчет по лабораторной работе № 6

«Обработка деревьев и хеш-таблиц»

Работу выполнил:

студент группы ИУ7-32Б

Лемешев А.П.  
Работу проверил:  
Силантьева А.В.

Москва, 2021 г.

# Условие задачи

# 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) и в файле. 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.

# Техническое задание

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. Построить хеш-таблицу из слов текстового файла. Использовать метод цепочек для устранения коллизий. Осуществить поиск введенного слова 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

**Входные данные**

- Имя файла, в котором хранятся слова.

- Целое число: значение максимальных коллизий

- Слово для поиска

**Выходные данные**

- Графическое изображение бинарного дерева

- Графическое изображение сбалансированного бинарного дерева

- Хеш-таблица

- Сравнительная характеристика для 4-х структур данных

**Действие программы**

- Построение и вывод бинарного дерева

- Построение и вывод сбалансированного бинарного дерева

- Построение и вывод хеш-таблицы

- Устранение коллизий, если это необходимо, и вывод новой хеш-таблицы

- Моделирование сравнительной характеристики по введённому слову

**Обращение к программе**

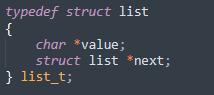
Запуск программы производится через командную оболочку MSYS2 для Windows, терминал sh/bash для Linux. Запуск программы происходит с помощью команды:   
*./app.exe [имя файла]*.

**Возможные аварийные случаи**

1. Некорректный ввод имени файла: файла не существует
2. Некорректный ввод имени файла: файл пустой
3. Некорректный ввод максимального числа коллизии: буква или отрицательное число
4. Некорректный ввод слова для поиска: слово не найдено

# Структуры данных

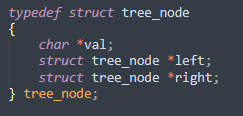
**Структура линейного односвязного списка:**



*value* – строка

*next* – следующий элемент списка

**Структура листа дерева:**

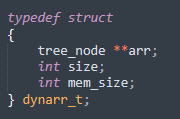


*value* – строка

*left* – левое поддерево

*right* – правое поддерево

**Структура динамического массива:**



*arr* – массив листьев дерева

*size* – текущий размер массива

*mem\_size* – выделенный размер под массив

**Описание алгоритма**

Сначала происходит балансировка дерева. Исходное дерево вытягивается в отсортированный массив, затем происходит рекурсивное построение АВЛ-дерева. После постройки дерева строится хеш-таблица с помощью **хеш-функции Дженкинса** по дереву. В случае, если количество коллизий превосходит допустимое, таблица самоперестраивается и выводится количественная характеристика моделирования.

# Оценка эффективности

Поиск слова (в тактах)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Количество слов | Бинарное дерево | Сбалансированное  бинарное дерево | Хэш-таблица | Файл |
| 16 | 342 | 357 | 385 | 12547 |
| 64 | 472 | 374 | 431 | 17207 |
| 256 | 630 | 483 | 478 | 39739 |
| 1024 | 810 | 579 | 428 | 106627 |

Среднее количество сравнений

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Количество слов | Бинарное дерево | Сбалансированное  бинарное дерево | Хэш-таблица  (без коллизий) | Файл |
| 16 | 3.875000 | 3.375000 | 1.000000 | 8.500000 |
| 64 | 6.109375 | 5.125000 | 1.000000 | 32.500000 |
| 256 | 9.266667 | 7.031373 | 1.000000 | 128.500000 |
| 1024 | 11.977574 | 8.967380 | 1.000000 | 512.500000 |

Занимаемая память (в байтах)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Количество слов | Бинарное дерево | Сбалансированное  бинарное дерево | Хеш-таблица | Файл |
| 16 | 384 | 384 | 264 | 130 |
| 64 | 1536 | 1536 | 1032 | 525 |
| 256 | 6024 | 6024 | 4024 | 2113 |
| 1024 | 21600 | 21600 | 14408 | 8295 |

# Вывод

Самой эффективной структурой данный является хеш-таблица. При верно подобранной хеш-функции, таблица будет без коллизий. Такая таблица самая эффективная по времени поиска *(количество сравнений будет 1)* и проигрывает по памяти только файлу.

Если сравнивать реализации деревьев ДДП и АВЛ, то АВЛ дерево является лучше структурой дня поиска, т.к. высота у этого дерева обычно меньше, чем у ДДП дерева, отсюда и меньшее количество сравнений. Память деревья занимают одинаковую, т.к. количество вершин у них одинаковое.

\*Если говорить о добавлении, то АВЛ дерево проигрывает ДДП по причине того, что при добавлении элемента нужна перебалансировка дерева, что занимает дополнительное время.

\**перебалансировка при добавлении* - заключается в том, что сначала новая вершина включается в АВЛ дерево и при возникновении ситуации, когда разница высот левого и правого поддерева == 2, происходит изменение связей предок-потомок в поддереве данной вершины так, чтобы разница стала равна <= 1. Изменение связей происходит вращениями поддерева данной вершины.

# Ответы на контрольные вопросыы

**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?**

Память выделяется динамически под каждый узел дерева, во время выполнения программы.

**3. Какие стандартные операции возможны над деревьями?**

Возможны следующие операции: добавление узла в дерево, удаление узла из дерева, обход 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?**

Скоростью поиска: поиск в АВЛ дереве происходит быстрее, чем в двоичном дереве 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 О(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 О(1). В АВЛ: О(log2n). В дереве двоичного поиска О(h), где h - высота дерева (от log2n до n).