|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **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)** |

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №6**

**«РАБОТА С ДЕРЕВЬЯМИ И ХЭШ-ТАБЛИЦАМИ»**

Студент: Светличная Алина Алексеевна

Группа: ИУ7 – 33Б

Проверил: Апальков Федор Станиславович

*2021 г.*

# **Описание технического задания**

Условие 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.

Входные данные:

* Номер пункта меню (от 0 до 4)
* Значение удаляемого элемента (для 4 пункта)
* Файл с целыми числами (имя файла задается через командную строку при запуске)

Выходные данные:

Выходные данные зависят от пункта меню

Пункты меню:

1. Считать числа из файла в дерево и таблицу

2. Сбалансировать дерево

3. Удалить элемент по значению

4. Просмотреть занимаемую память

0. Выход

Обращение к программе:

Запускается через терминал командой ./app.exe [filename] в директории с программой.

Возможные аварийные ситуации:

1. Некорректный пункт меню
2. Некорректный файл
3. Имя файла не указано
4. Файл невозможно открыть (несуществующий файл)
5. Пустой файл
6. Некорректные символы в файле
7. Некорректное значение удаляемого элемента
8. Некорректное значение максимального количества сравнений

# **Описание структур данных**

Поля структуры, описанной ниже, для хранения узла дерева:

value – значение элемента

height – высота дерева их этого узла

\*left – указатель на левое поддерево

\*right – указатель на правое поддерево

**struct tree\_node\_t**

**{**

**int value;**

**unsigned char height;**

**struct tree\_node\_t \*left;**

**struct tree\_node\_t \*right;**

**};**

Поля структуры, описанной ниже, для хранения элемента хеш-таблицы:

key – ключ-значение

status – статус ячейки

**typedef struct**

**{**

**int key;**

**int status;**

**} node\_hash\_t;**

Поля структуры, описанной ниже, для хранения хеш-таблицы:

\*array – указатель на массив элементов

size – размер массива

**struct hash\_t**

**{**

**node\_hash\_t \*array;**

**int size;**

**};**

# **Описание алгоритма**

1. Первый пункт меню:

Все значения из файла считываются с ДДП и хеш-таблицу  
В качестве начальной хеш-функции используется остаток от деления ключа на размер таблицы (определенный как ближайшее простое число большее количества элементов считанного дерева).

Полученные ДДП и хеш-таблица выводятся на экран (здесь и далее деревья выводятся с помощью утилиты graphiz)

Принцип работы закрытого хеширования (открытой адресации): если ячейки, возвращенной хеш-функцией свободна, то элемент записывается туда; если ячейка уже занята, то элемент записывается в ближайшую свободную ячейку.

1. Второй пункт меню:

Происходит балансировка ДДП (путем правых и левых поворотов узлов дерева там, где высота узла больше единицы) и сбалансированное дерево выводится на экран.

1. Третий пункт меню:

Происходит удаление указанного с клавиатуры ключа-значения из ДДП АВЛ-дерева, хеш-таблицы и из исходного файла (с замером времени удаления)

На экран выводятся либо измененная структура данных.

При удалении из хеш-таблицы у пользователя запрашивается максимальное количество сравнений.

При удалении считается количество сравнений и, если оно превысило максимальное – производится реструктуризация хеш-таблицы с новой хеш-функцией.

1. Четвертый пункт меню:

Выводится память, занимаемая всеми структурами данных.

0) Выход из программы

Программа завершает свою работу

**Тесты**

|  |  |  |
| --- | --- | --- |
| Аварийная ситуация | Код ошибки | Уведомление |
| Некорректный пункт меню | | |
| Пункт меню - k | ERROR\_MENU | Ошибка: введен некорректный номер пункта меню |
| Пункт меню - -2 | ERROR\_ MENU | Ошибка: введен некорректный номер пункта меню |
| Пункт меню - 10 (нет такого номера) | ERROR\_ MENU | Ошибка: введен некорректный номер пункта меню |
| Некорректный файл | | |
| Запуск -./app.exe | ERROR\_FILE | Ошибка: имя файла не указано |
| Несуществующий файл | ERROR\_FILE | Ошибка: невозможно открыть файл |
| Пустой файл | ERROR\_FILE | Ошибка: в файле встречены некорректные символы |
| В файле содержатся символьные переменные | ERROR\_FILE | Ошибка: в файле встречены некорректные символы |
| Некорректное значение удаляемого элемента | | |
| Удаляемый элемент - № | ERROR\_SYMBOLS | Ошибка: встречены некорректные символы |
| Некорректное значение максимального количества сравнений | | |
| Количество сравнений - ?\*( | ERROR\_SYMBOLS | Ошибка: встречены некорректные символы |

**Обработка результатов**

**Сравнение скорости (в мс)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Количество элементов в структурах | Дерево двоичного поиска | АВЛ дерево | ХЕШ таблица | Файл |
| 100 | 16.584 | 3.150 | 0.340 | 2336.792 |
| 1000 | 54.834 | 4.180 | 0.096 | 3928.390 |
| 10000 | 492.764 | 6.722 | 0.120 | 26503.090 |

**Сравнение памяти (в байтах)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Количество элементов в структурах | Дерево двоичного поиска | АВЛ дерево | ХЕШ таблица | Файл |
| 100 | 2400 | 2400 | 816 | 294 |
| 1000 | 24000 | 24000 | 8016 | 3895 |
| 10000 | 240000 | 240000 | 80016 | 48896 |

**Контрольные вопросы**

**1. Что такое дерево?**

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

**2. Как выделяется память под представление деревьев?**

Память выделяется в виде связного списка (динамически под каждый узел).

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

Добавление элемента, удаление элемента, поиск по дереву, обход дерева.

**4. Что такое дерево двоичного поиска?**

Двоичное дерево поиска - двоичное дерево, для которого существует обязательное условие распределения потомков: правый больше или равен родителю, левый меньше родителя (может быть 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 (меньшее количество сравнений).

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

Хеш-таблицей называется массив, заполненный элементами в определенном порядке. Порядок определяется хеш-функцией – каждому элементу таблицы ставит в соответствие некоторый индекс.

Функция должна быть:

* простой для вычисления
* распределять ключи в таблице равномерно
* 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.

Методы устранения коллизий:

При методе цепочек к ячейке по данному ключу прибавляется связанны список.

При закрытом хэшировании новый элемент кладется в ближайшую свободную ячейку после данной.

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

Если в таблице возникает большое число коллизий – сложность поиска возрастает по сравнению с О(1) – поиск в хэш-таблице становится неэффективным. В таком случае требуется реструктуризация таблицы.

**10. Эффективность поиска в АВЛ деревьях, в дереве двоичного поиска и в хеш-таблицах.**

В хеш-таблице минимальное время поиск: О(1).

В АВЛ: О(log2n).

В дереве двоичного поиска: О(h), где h - высота дерева (от log2n до n).

**Вывод**

Использование хеш-таблицы всегда эффективно по времени (от 10 до 10000), но не всегда эффективно по памяти, так как требует выделенной памяти под каждый хэш (если присутствуют коллизии эффективность по времени может серьезно снизиться или вообще перестать быть наилучшей, поэтому необходимо наиболее точно подбирать хэш-функции).

В случае деревьев АВЛ дерево всегда выигрывает по времени у несбалансированного дерева (в 5 раз), так как необходимо произвести меньшее число сравнений.