|  |  |
| --- | --- |
|  | **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 К ЛАБОРАТОРНОЙ РАБОТЕ №6**

***«Деревья, хеш – таблицы»***

Выполнил студент: Казаева Татьяна Алексеевна

*фамилия, имя, отчество*

Группа: ИУ7-36Б

Проверил

*подпись, дата*

*2020 г.*

**Описание условия задачи**

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. Добавить указанное слово, если его нет в дереве (по желанию пользователя) в исходное и сбалансированное дерево. Сравнить время добавления и объем 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 добавление введенного слова, вывести таблицу. Сравнить время добавления, объем памяти и количество сравнений при использовании ДДП, сбалансированных деревьев, хеш-таблиц и файла.

**Техническое задание**

Построить ДДП, сбалансированное двоичное дерево (АВЛ) и хеш-таблицу по указанным данным. Сравнить эффективность заданной операции в ДДП в АВЛ дереве и в хеш-таблице. Вывести на экран деревья и хеш-таблицу. Произвести реструктуризацию хеш-таблицы, если среднее количество сравнений больше указанного. Оценить эффективность использования этих структур (по времени и памяти) для поставленной задачи.

*Исходные данные:* Введенный пользователем с консоли номер операции, выполняемой программой. В зависимости от номера:

1. Операции с ДДП:
   1. Создание ДДП;
   2. Балансировка ДДП;
   3. Вставка слов в сбалансированную и несбалансированную ДПП;
   4. Замеры времени операций c.
2. Операции с хеш – таблицой:
   1. Создание таблицы;
   2. Вставка слов в хеш – таблицу;
   3. Вывод хеш – таблицы на экран.
3. *Описание задачи, реализуемой программой:* Пользователю дается выбор задач (см. «Исходные данные»).

*Возможные аварийные ситуации:*

1. Пустой файл;
2. Несуществующий файл;
3. Попытка работы с не аллоцированными сируктурами;
4. Попытка работы с несбалансированным деревом (см. «Исходные данные» 1.c)
5. Ввод некорректного значения размера таблицы;
6. Ввод некорректного допустимого количества коллизий.

*Способ обращения к программе:* Программа работает в консоли. Для обращения к программе в консоли вводится имя программы (здесь: ./app.exe).

**Описание внутренних структур**

1. *Хеш – таблица.*

Для хеш – таблицы создание отдельной структуры избыточно. Хэш – таблица представляет собой динамический массив, элементами которых являются связные списки со следующими узлами:

typedef struct linked\_hash linked\_hash\_t;

struct linked\_hash

{

char \*key;

linked\_hash\_t \*next;

};

hash\_table[size] – сама хеш - таблица.

Где:

1. key – символьное поле, полученное из источника слово;
2. \*next – указатель на следующий узел.

Хеш – код в описанной реализации – индекс элемента массива.

1. *Двоичное дерево поиска.*

Двоичное дерево поиска описано следующей структурой:

typedef struct tree\_node tree\_node\_t;

struct tree\_node

{

char \*key;

tree\_node\_t \*left;

tree\_node\_t \*right;

};

Где:

1. key – символьное поле, полученное из источника слово;
2. \*left – указатель на левого потомка;
3. \*right – указатель на правого потомка.

**Описание алгоритма**

1. *Хеш – таблица.*

Поскольку хеширование открытое, элементы с одинаковым хеш – кодом попадают в одну и ту же ячейку таблицы. Количество коллизий при каждом добавлении считается как «глубина» списка значений в этой ячейке.

Используемые хеш – функции : хеш – функция Дженкинса и хеш – функция, определяющая сумму кодов букв строки.

1. *Дерево двоичного поиска*

При построении сбалансированного дерева происходят повороты вокруг тех или иных узлов дерева при обнаружении разбалансировки.

Вставка осуществляется рекурсивно, сравниваются ключи узлов дерева и перемещаются по левой или правой ветви соответственно.

**Тестовые данные**

|  |  |  |
| --- | --- | --- |
| Наименование теста | Входные данные | Выходные данные |
| [Н] Некорректное имя файла | 1 -> к | Файл не найден. Проверьте имя файла и попробуйте еще раз. |
| [Н] Пустой файл | 1 -> \пустой файл\ | Файл пуст. Проверьте содержимое файла и попробуйте еще раз. |
| [Н] Неверно введенное число коллизий | 1 -> f | Число введено неверно. |
| [Н] Работа с непостроенной хеш - таблицей | 2 -> 2 | Сначала хеш - таблицу нужно построить. |
| [Н] Работа с несбалансированным деревом | 1 -> 1 -> 4 | Сначала дерево нужно сбалансировать. |
| [Н] Работа с непостроенным деревом | 1 -> 4 | Сначала дерево нужно построить. |
| [П] Построение дерева | 1 -> 1 | См. приложение 1 |
| [П] Балансировка дерева | 1 -> 1 -> 2 | См. приложение 2 |
| [П] Вывод хеш - таблицы | 2 -> | См. приложение 3 |
| [П] Добавление слова в хеш - таблицу |  | Успешно. |

**Сравнительный анализ**

Размер исходного файла : 15 слов. Замеры производятся в тиках процессора.

|  |  |  |  |
| --- | --- | --- | --- |
| Cбалансированное дерево | Несбалансированное дерево | Хеш - таблица | Файл |
| 1665.2 | 2473.06 | 510 | 32904 |

**Выводы по проделанной работе**

Хэш – таблица всегда выигрывает по времени у деревьев, но она не всегда эффективна по памяти (требует выделения памяти на каждый хеш – код, и в случае хорошей хеш – функции, распределение не плотное). В случае деревьев, неотсортированное выигрывает у отсортированного, т. к после вставки дерево небоходимо сбалансировать снова.

**Контрольные вопросы**

1. Что такое дерево?

Дерево – это абстрактный тип данных с принципом связи «один ко многим».

2. Как выделяется память под представление деревьев?

Как и под связный список – выделяется память под узлы дерева.

3. Какие стандартные операции возможны над деревьями?

Добавление узлов, вставка, удаление; поиск узла в дереве.

4. Что такое дерево двоичного поиска?

Дерево, левый потомок каждого узла которого больше или равен родителю, а правый – строго меньше.

5. Чем отличается идеально сбалансированное дерево от АВЛ дерева?

У АВЛ - дерева разница глубины узлов не более 1, а у идеально сбалансированного количество вершин в каждом поддереве различается на 1(не более).

6. Чем отличается поиск в АВЛ-дереве от поиска в дереве двоичного поиска?

Поиск в АВЛ дереве быстрее чем в дереве двоичного поиска.

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

Таблица вида «ключ – хеш код». Такая таблица возвращает один и тот же код для одного значения, но не всегда возвращает разыне коды для разных значений.

8. Что такое коллизии? Каковы методы их устранения.

Коллизия – это совпадение хеш – кодов. Из принципов устранения выделяют открытое и закрытое хеширование.

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

Бри большом числе коллизий.

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

Таблица : O(1), в худшем случае O(n). АВЛ – O(log2n). В ДДП – O(n), где n – высота дерева.