|  |  |
| --- | --- |
|  | **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 К ЛАБОРАТОРНОЙ РАБОТЕ №6 ПО «ТИПАМ И СТРУКТУРАМ ДАННЫХ»**

**Тема: «Обработка деревьев»**

Группа: ИУ7-32Б

|  |  |  |
| --- | --- | --- |
| Студент: |  | Сидоров Максим Михайлович |

Преподаватель: Никульшина Татьяна Александровна

**Цель работы:** научиться работать с бинарными деревьями.

**Условие задачи:** построить бинарное дерево, в вершинах которого находятся слова из текстового файла. Вывести его на экран в виде дерева. Удалить все слова, начинающиеся на указанную букву. Сравнить время удаления слов, начинающихся на указанную букву, в дереве и в файле.

**Входные данные:** номер команды из меню, текстовый файл и терминальный ввод.

Команды:

1. Вывести граф
2. Удалить слова, начинающиеся с буквы и вывести на экран
3. Сравнить удаление из файла и из дерева
4. Сравнить поиск самого правого элемента в бинарном дереве и дереве-списке
5. Выйти

**Выходные данные:**

- исходное дерево в виде картинки

- дерево с удаленными словами в виде картинки

- сравнение удаления в дереве и в файле

- сравнение поиска

**Аварийные ситуации:**

- ввод неправильной команды

- некорректный файл

**Описание СД:**

1. Для хранения узла дерева было выбрано следующее представление:

typedef struct tree tree;

struct tree

{

char \*word;

tree \*left;

tree \*right;

tree \*parent;

};

*Листинг 1. Представление узла дерева.*

**Алгоритм удаления слов из бинарного дерева:** во время префиксного обхода бинарного дерева проверяется, содержит ли узел слово, которое следует удалить. Если нет, то дальше рекурсивно выполняем данную операцию с правым и левым узлами. Если надо удалить, то алгоритмом поиска, начиная с данного узла в поддереве ищем первый узел с словом, которое удалять не требуется и заменяем слова. Если в данном поддереве нет подходящих слов, то удаляем все узла данного поддерева.

**Тестовые данные:**

|  |  |  |
| --- | --- | --- |
| **Описание** | **Вход** | **Выход** |
| Некорректная команда | А, -1, 15 | Некорректная команда |
| Запуск программы с некорректным файлом | Dsfsj.chh | Некорректный файл |
| Удалить слова, начинающиеся с буквы | а | Удаление слов и вывод на дерева экран |
| Сравнение удаления из файла и из дерева | - | Время удаления для файла и для дерева |
| Сравнения поиска самого правого элемента в бинарном дереве и дереве-списке | - | Время поиска для бинарного дерева и дерева-списка |

**Функции:**

void print\_menu() – вывод меню

void compare\_search(char \*file\_name) – сравнение поиска самого правого элемента

char \*find\_right(tree \*a) – поиск самого правого элемента

void new\_search\_word(tree \*\*find, tree \*a, bool \*flag, char \*s) – поиск слова

tree \*push\_one\_side\_tree(tree \*a, FILE \*f) – создание дерева-списка

void compare\_delete(char \*file\_name) – сравнение удаления

void delete\_file(FILE \*f, char s) – удаление из файла

void read\_word(char \*a, FILE \*f) – считывание слова из файла

int tree\_height(FILE \*f) – вычисление высоты дерева

tree \*push\_tree(tree \*a, FILE \*f, int cur\_height, int max\_height, tree \*parent) – заполнение дерева из файла

void to\_dot(tree \*a, FILE \*f) – запись узлов в формате DOT

void apply\_pre(tree \*a, void (\*f)(tree\*, FILE\*), FILE \*arg) – префиксный обход дерева

void export\_to\_dot(tree \*a, char \*file\_name) – создание DOT-файла

void search\_word(tree \*\*find, tree \*a, bool \*flag, char s) – поиск слова в дерева по первой букве

tree \*delete\_words(tree \*a, char s) – удаление слов, начинающихся с данной буквы

void free\_tree(tree \*a) – удаление дерева

**Сравнение эффективности при удалении из дерева и из файла:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Количество элементов** | **Количество элементов, которые надо удалить** | **Дерево, мкс** | **Файл, мкс** | **Эффективность** |
| 1 | 0 | 0 | 40 | 4000% |
| 5 | 1 | 1 | 41 | 4000% |
| 10 | 1 | 1 | 46 | 4600% |
| 5 | 3 | 56 | 1767% |
| 10 | 2 | 50 | 2400% |
| 20 | 1 | 1 | 49 | 4800% |
| 10 | 3 | 58 | 1833% |
| 20 | 2 | 52 | 2500% |
| 40 | 1 | 1 | 51 | 5000% |
| 20 | 3 | 64 | 2033% |
| 40 | 2 | 54 | 2600% |

|  |  |  |  |
| --- | --- | --- | --- |
| **Количество элементов** | **Дерево, байты** | **Файл, байты** | **Эффективность** |
| 1 | 32 | 3 | -967% |
| 5 | 160 | 15 | -967% |
| 10 | 320 | 34 | -841% |
| 20 | 640 | 66 | -870% |
| 40 | 1280 | 125 | -924% |

Удаление слов из дерева намного быстрее, чем удаление слов из файла, время работы не зависит от количества слов, которые нужно удалить. Память, затрачиваемая на дерево примерно в 9 раз больше, чем память, затрачиваемая на хранения файла.

**Вывод:** бинарное дерево — это структура данных, в которой есть главный элемент – корень, и у каждого элемента есть один родитель и не более 2 потомков. Обработка дерева намного быстрее, чем обработка файлов, но затраты по памяти больше, чем у файла.

**Контрольные вопросы:**

1. **Что такое дерево?**

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

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

В памяти деревья можно представить в виде связей с предками (еще их называют родителями); связного списка потомков (сыновей) или структуры данных.

Память под представление деревьев выделяется таким же образом, как и для односвязного списка: выделяется по мере нужды для каждого нового узла дерева.

1. **Какие бывают типы деревьев?**

Типы деревьев:

* N-арное дерево
* сбалансированное дерево
* бинарное дерево
* бинарное дерево поиска
* дерево AVL
* красно-чёрное дерево
* 2-3 дерево

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

Стандартные операции над деревьями: обход дерева, поиск по дереву, включение в дерево, исключение из дерева.

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

Двоичное дерево поиска (ДДП) — двоичное дерево. В нем для каждого узла выполняется условие, что левый потом больше или равен родителю, а правый потомок строго меньше родителя (или наоборот). Это свойство называется характеристическим свойством дерева двоичного поиска и выполняется для любого узла, включая корень.