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 №7
«СБАЛАНСИРОВАННЫЕ ДЕРЕВЬЯ, ХЕШ-ТАБЛИЦЫ»

Студент Дьяченко Артём Александрович

Группа ИУ7 – 33Б

Преподаватель Барышникова М. Ю.

2022 год.

Оглавление

Описание условия задачи

Цель работы: построить и обработать хеш-таблицы, сравнить эффективность поиска в сбалансированных деревьях, в двоичных деревьях поиска и в хеш-таблицах. Сравнить эффективность устранения коллизий при внешнем и внутреннем хешировании.

Используя бинарное дерево следующего выражения: 9+(8*(7+(6*(5+4)-(3-2))+1)), и процедуру постфиксного обхода дерева, вычислить значение каждого узла и результат записать в его вершину. Получить массив, используя процедуру инфиксного обхода полученного дерева. Построить для этих данных дерево двоичного поиска (ДДП), сбалансировать его. Построить хеш-таблицу для значений этого массива. Осуществить поиск указанного значения. Сравнить время поиска, объем памяти и количество сравнений при использовании ДДП, сбалансированных деревьев и хеш-таблиц.

Описание технического задания

Входные данные:

Целочисленное значение ключа.

Выходные данные:

Полученное бинарное дерево, ДДП, АВЛ-дерево, выражения, полученные через различные обходы дерева, результат выражения, время его вычисления.

Обращение к программе:

Запускается через терминал командой: ./app.exe.

Аварийные ситуации:

1. Некорректные данные переменной.
2. Ошибка выделения памяти.
3. Возникновение коллизий.

Набор тестов

№	Название теста	Пользовательский ввод	Вывод
1	Корректный ввод переменных	1 2 3 4 5 6 7 8 9	
2	Некорректный ввод переменной	e10	Ошибка! Требуется целое число.
3	Некорректный ввод переменной	a	Ошибка! Требуется целое число.
4	Невозможно выделить память под вершину дерева		Ошибка выделения памяти для узла дерева!
5	Невозможно выделить память для стека		Ошибка выделения памяти под стек!

Описание структуры данных

Структура узла дерева.

```
struct node {
    int depth;       // глубина вершины
    node_t *left;    // левый "ребёнок"
    node_t *right;   // правый "ребёнок"
    node_t *parent; // узел-родитель
    int value;       // значение в узле
    char option;     // операция узла
};
```

Структура узла дерева.

```
struct set
{
  int key;       // ключ
  int data;      // значение
  set_t *next;  // указатель на след. элемент
};
```

Оценка эффективности (такты)

	ДДП	АВЛ-дерево	Хеш-таблица
Время поиска	0.014513	0.007014	0.005643
Кол-во сравнений	8.94	6.53	1

Для оценки эффективности было проведено 1.000.000 расчётов и взято среднее время.

АВЛ-дерево в следствие своей балансировки работает в два раза быстрее, чем ДДП. По этой же причине у АВЛ меньшее кол-во сравнений. Хеш-таблица работает быстрее всех, т.к. нам достаточно только вычислить индекс по ключу и обратиться к нужной ячейке.

Память (байт)

ДДП	АВЛ-дерево	Хеш-таблица
680	520	136

ДДП занимает на 23% больше места, чем АВЛ-дерево и в 5 раз, чем хеш-таблица. АВЛ-дерево в свою очередь занимает почти в 4 раза больше памяти, чем та же хеш-таблица.

Описание алгоритма

1. Создаётся бинарное дерево на основе выражения:
   9 + (8 * (7 + (6 * (5 + 4) - (3 - 2)) + 1)).
2. Программа проходит по созданному дереву, инфиксным и постфиксным обходом.
3. При постфиксном обходе в каждой вершине высчитывается её значение. Так, значение в корне дерева – результат выражения. При инфиксном – создаётся массив со значениями всех вершин.

4. Создаётся ДДП со значениями из массива и выводится на экран.
5. Создаётся АВЛ-дерево со значениями из массива, балансируется и выводится на экран.
6. Выводится значение выражения.
7. Создаётся и выводится на экран хеш-таблица на основе значений дерева. Для предотвращения коллизий используется метод цепочек.
8. Пользователь вводит значение ключа, по которому нужно получить значение. Если он верный – выводится пара: ключ | значение.

Ответы на контрольные вопросы

1. Чем отличается идеально сбалансированное дерево от АВЛ дерева?

В идеально сбалансированном дереве кол-во элементов в правом и левом поддереве отличается не более чем на единицу. В АВЛ дереве высоты правого и левого поддерева отличается не более чем на единицу

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

Алгоритм одинаков.

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

Структура данных позволяющая получать по ключу элемент массива называется хеш-таблицей.

Для доступа по ключу используется хеш-функция. Она по ключу получает нужный индекс массива. Хеш-функция должна возвращать одинаковые значения для одного ключа и использовать все индексы с одинаковой вероятностью.

4. Что такое коллизии? Каковы методы их устранения.

Ситуация, когда из разных ключей хеш-функция выдаёт одни и тот же индекс, называется коллизией.

Метод цепочек – при коллизии элемент добавляется в список элементов этого индекса.

Линейная адресация – при коллизии ищется следующая незаполненная ячейка.

Произвольная адресация - используется заранее сгенерированный список случайных чисел для получения последовательности.
Двойное хеширование – использовать разность 2 разных хеш-функций.

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

При большом количестве коллизий.

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

Скорость поиска в хеш-таблице зависит от числа коллизий. При небольшом числе коллизий для поиска элемента совершается мало сравнений и поиск быстрее чем в деревьях.
АВЛ дерево быстрее при поиске за счёт более равномерного распределения элементов чем в ДДП.

Вывод

В ходе лабораторной работы я написал программу, строящую бинарное дерево, АВЛ-дерево и хеш-таблицу и измерил их эффективность.
Быстрее всего работает хеш-таблица, но она обладает большим недостатком: в ней могут появиться коллизии, которые замедляют работу (т. к. в методе цепочек нужно каждый раз проходить по всему списку до последнего элемента). Для уменьшения кол-ва коллизий нужно иметь хеш-функцию с хорошим распределением.
Деревья работают медленнее, но они лишены этого недостатка.
АВЛ-деревья будет быстрее бинарного в задачах с частым поиском, и наоборот, когда чаще поиска происходит вставка.
По памяти самое эффективное решение – хеш-таблица.