

ФАКУЛЬТЕТ: Информатика и системы управления

КАФЕДРА: Программное обеспечение ЭВМ и информационные технологии

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №6**

**«Деревья, хеш –таблицы»**

Студент: Зайцева Алена Андреевна

Группа ИУ7 – 32Б

*2020 г.*

Оглавление

Оглавление

[1. Описание условия задачи 2](#_Toc57758360)

[2. Описание ТЗ 3](#_Toc57758361)

[1) Описание исходных данных: 3](#_Toc57758362)

[2) Описание результатов: 3](#_Toc57758363)

[3) Описание задачи, реализуемой программой: 3](#_Toc57758364)

[4) Способ обращения к программе: 4](#_Toc57758365)

[5) Описание возможных аварийных ситуаций и ошибок пользователя. 4](#_Toc57758366)

[3. Описание внутренних структур данных 4](#_Toc57758367)

[4. Описание алгоритма 5](#_Toc57758368)

[5. Набор тестов с указанием, что проверяется 5](#_Toc57758369)

[1) Негативные тесты на командую строку и файл 5](#_Toc57758370)

[2) Негативные тесты на ввод пользователя в самой программе 5](#_Toc57758371)

[3) Проверка построения ДДП и его балансировки 6](#_Toc57758372)

[4) Проверка построения хеш-таблицы на основании деления на ближайшее простое и в случае необходимости – методом середины квадрата (так же с делением) 6](#_Toc57758373)

[5) Проверка поиска 7](#_Toc57758374)

[6. Оценка эффективности 8](#_Toc57758375)

[7. Выводы по проделанной работе 9](#_Toc57758376)

[8. Ответы на вопросы 10](#_Toc57758377)

# Описание условия 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 метод цепочек для устранения коллизий.
* Осуществить поиск введенного целого числа 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.

# Описание ТЗ

Внешняя спецификация:

## Описание исходных данных:

* Текстовый файл с целыми числами (имя файла передается через параметры командной строки)
* Целое число, которое необходимо найти
* Максимальное количество сравнений в хеш-таблице

## Описание результатов:

* Графическое изображение ДДП
* Графическое изображение сбалансированного ДДП
* Хеш-таблица чисел из файла
* Реструктурированная хеш-таблица в случае необходимости
* Для каждой из 4-х структур данных (ДДП, сбалансированного ДДП, хеш-таблицы, файла):
  + Результат поиска (найдено/не найдено)
  + Время поиска данного числа
  + Среднее время поиска (считается как среднее для всех чисел из файла)
  + Объем занимаемой памяти
  + Количество сравнений при поиске данного числа
  + Среднее количество сравнений при поиске (считается как среднее для всех чисел из файла)

## Описание задачи, реализуемой программой:

* Строит ДДП из чисел файла и выводит его на экран в виде дерева.
* Строит сбалансированное ДДП из чисел файла и выводит его на экран в виде дерева.
* Строит хеш-таблицу из чисел файла (реализованы 2 возможные хеш-функции) и выводит ее на экран.
* Осуществляет поиск в каждой из 4-х структур данных (ДДП, сбалансированного ДДП, хеш-таблицы, файла) и выводит количественную информацию о поиске (см. список выше)

## Способ обращения к программе:

Для сборки исполняемого файла в папке есть файл makefile. Чтобы собрать исполняемый файл app.exe нужно написать команду “make app.exe”. Программа запускается из папки, содержащей исполняемый файл app.exe и файл с числами командой “./app.exe file\_name.txt”.

## Описание возможных аварийных ситуаций и ошибок пользователя.

Возможные ошибки пользователя:

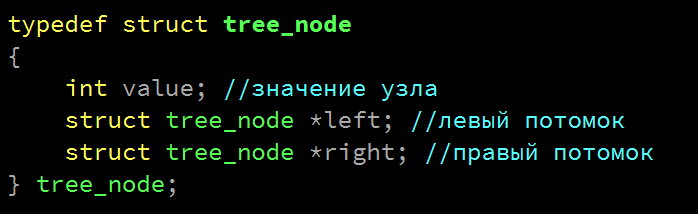
1. Не указано имя файла
2. Ввод названия несуществующего файла
3. Переданный файл пустой
4. В переданном файле нецелочисленное значение
5. Невозможное максимальное количество сравнений (<1)
6. Нецелочисленное значение для поиска

Возможные аварийные ситуации:

1. Программа не сможет выделить достаточное количество памяти для реализации задачи.

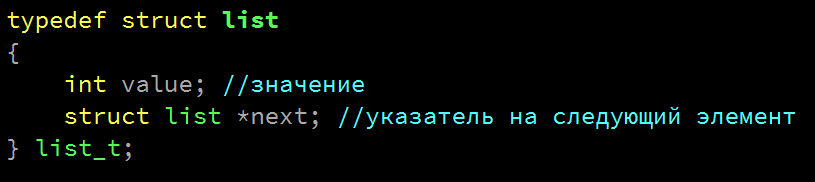
# Описание внутренних структур данных

1. Представление узла дерева

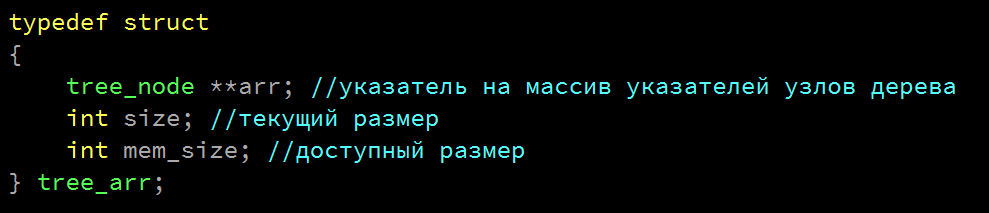


Для работы с хеш-таблицей:

1. Узел линейного односвязного списка



1. Динамический массив



## Описание алгоритма

Программа последовательно реализует следующие действия:

1. Строит ДДП из чисел файла. Выводит его на экран в виде дерева.
2. Строит сбалансированное ДДП. Выводит его на экран в виде дерева.
3. Строит хеш-таблицу из чисел файла (в качестве начальной хеш-функции используется вычисление остатка от деления целочисленного ключа на первое простое число p, большее количества чисел в файле: h(ki) = (ki % p)). Выводит ее на экран.
4. Запрашивает максимальное количество сравнений в хеш-таблице и в случае необходимости реструктурирует ее (в качестве второй хеш-функции используется метод “середины квадрата”, объединенный с вычислением остатка, как описано выше. Так как в методе середины квадрата осуществляется побитовый сдвиг числа, данные, которые были схожи (близки по значению) в исходной таблице, становятся более далекими друг от друга, что позволяет составить хеш-таблицу с меньшим количеством коллизий, при этом не слишком увеличивая количество незаполненных областей памяти)
5. Запрашивает число, которое необходимо найти.
6. Осуществляет поиск в каждой из 4-х структур данных (ДДП, сбалансированного ДДП, хеш-таблицы, файла) и выводит количественную информацию о поиске (см. список выше):

# Набор тестов с указанием, что проверяется

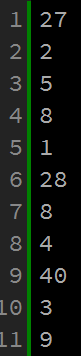
## Негативные тесты на командую строку и файл

|  |  |  |
| --- | --- | --- |
| Что проверяет | Командная строка | Результат |
| Не указано имя файла | ./app.exe | ERROR! Expected: app.exe file\_name.txt. |
| Несуществующий файл | ./app.exe unknown.txt | ERROR! Can't open file. |
| Пустой файл | ./app.exe empty.txt | ERROR! File is empty. |
| Файл с некорректным значением | ./app.exe incorrect.txt | ERROR! Incorrect value in file. |

## Негативные тесты на ввод пользователя в самой программе

|  |  |  |
| --- | --- | --- |
| Что проверяет | Ввод | Результат |
| максимальное количество сравнений | 0 | Сообщение об ошибке и предложение повторить ввод |
| -1 |
| a |
| значение для поиска | a |
| .1 |

Исходные данные:

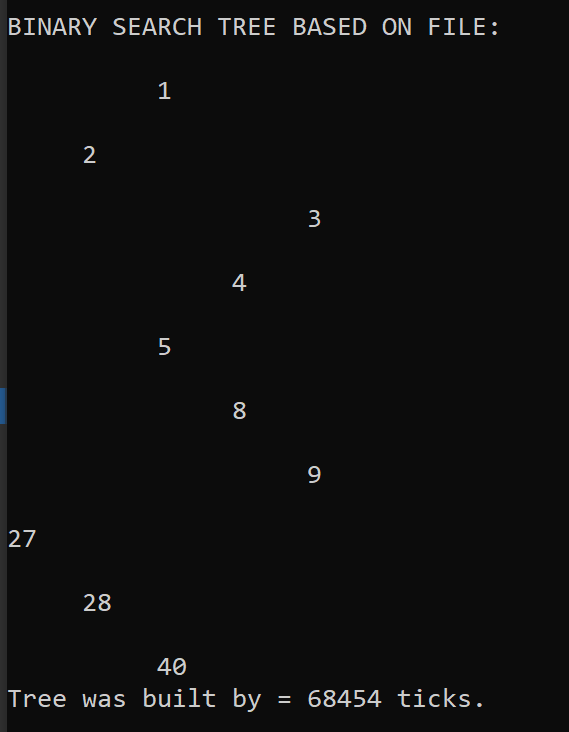
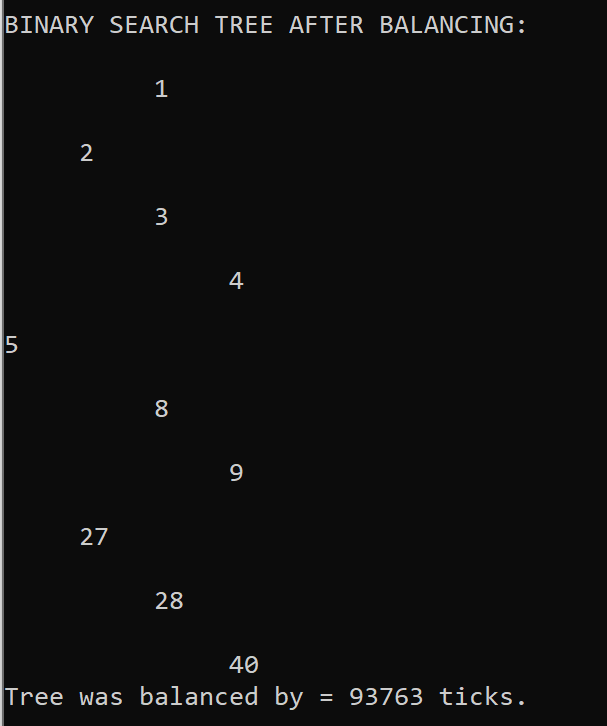


## Проверка построения ДДП и его балансировки

(смотреть слева-направо)

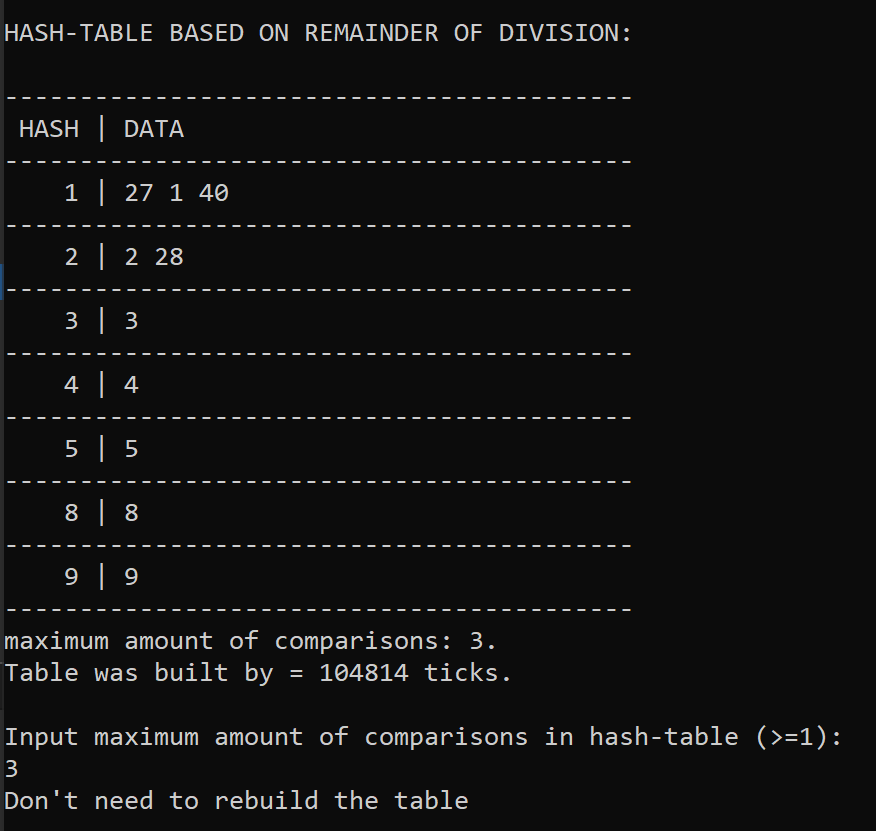
Все левые потомки старше предка, а все правые – моложе.

Сбалансированное дерево: число вершин в левом и правом поддеревьях отличается не более, чем на единицу.

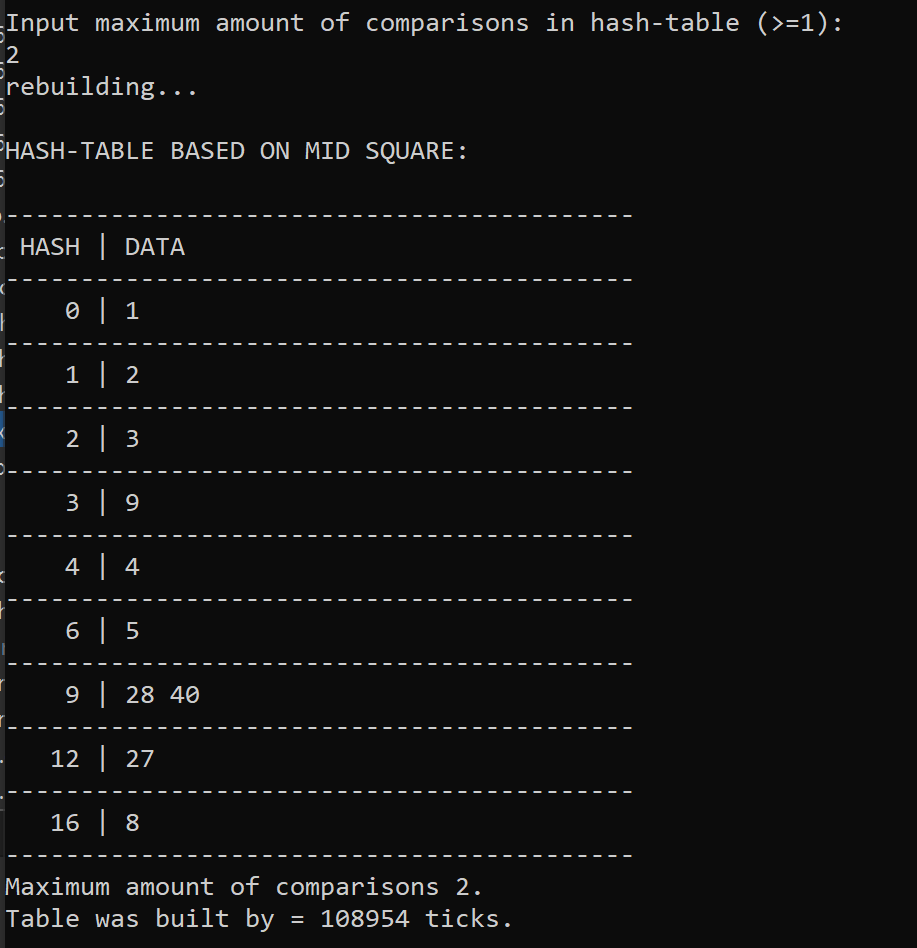
 

## Проверка построения хеш-таблицы на основании деления на ближайшее простое и в случае необходимости – методом середины квадрата (так же с делением)

Нет необходимости перестраивать

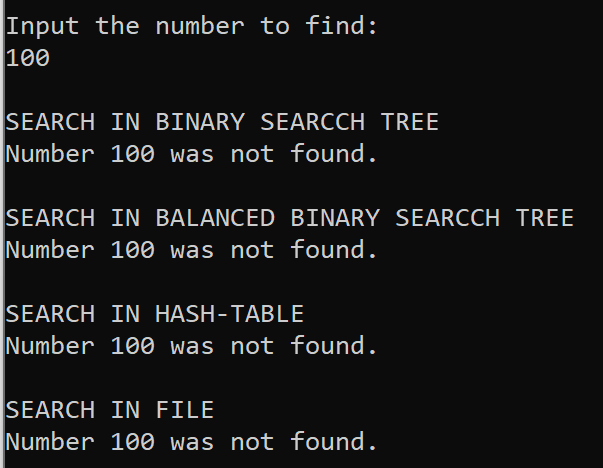


Нужно перестраивать:

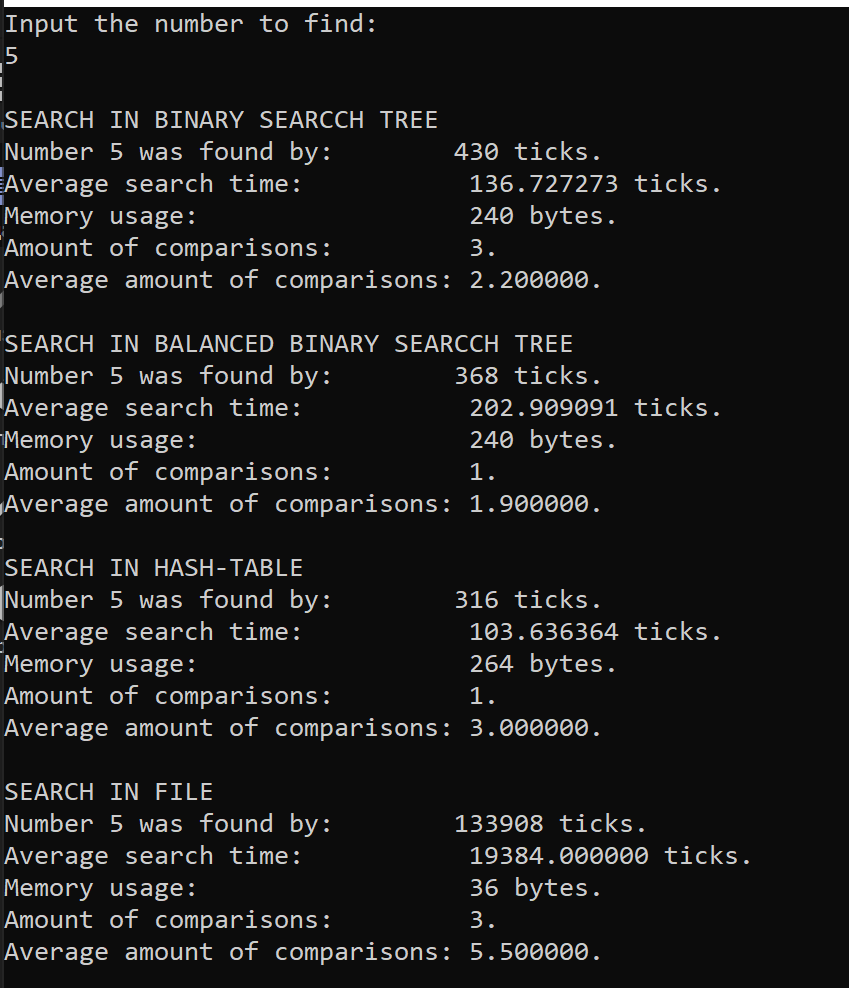


## Проверка поиска

Не найдено:



Найдено:



# Оценка эффективности

Сравним время поиска, объем памяти и количество сравнений при использовании 4-х структур данных (ДДП, сбалансированного ДДП, хеш-таблицы, файла) при различном количестве элементов. (измерения проводятся при условии, что в хеш-таблице нет коллизий)

**Время поиска** (в тактах процессора, среднее время для поиска всех элементов)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Количество элементов | ДДП | Сбалансированное ДДП | Хеш-таблица | Файл |
| 10 | 280 | 183 | 126 | 111083 |
| 100 | 546 | 366 | 324 | 203951 |
| 1000 | 4043 | 3670 | 996 | 789263 |

**Количество сравнений** (штук, среднее для поиска всех элементов)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Количество элементов | ДДП | Сбалансированное ДДП | Хеш-таблица | Файл |
| 10 | 3.7 | 1.9 | 1 | 3 |
| 100 | 40.6 | 4.8 | 1 | 50 |
| 1000 | 201 | 9 | 1 | 201 |

**Объем занимаемой памяти** (байт)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Количество элементов | ДДП | Сбалансированное ДДП | Хеш-таблица | Файл |
| 10 | 240 | 240 | 184 | 31 |
| 100 | 2400 | 2400 | 1624 | 293 |
| 1000 | 24000 | 24000 | 16152 | 3096 |

# Выводы по проделанной работе

Хеш-таблица наиболее эффективна по всем параметрам за исключением объёма памяти (Файл более эффективен). Это связано с необходимостью хранить каждый хеш, если отсутствуют коллизии и у функции хорошая дистрибуция. ДДП по памяти соответствует сбалансированному ДДП из-за использования одной и той же структуры данных,

Для более быстрого поиска в хэш-таблице должно быть как можно меньше коллизий. Количество коллизий зависит от того, сколько было выделено памяти под хеш-таблицу, следовательно, для большей эффективности по времени придется выделить больше памяти. Количество коллизий также зависит от равномерности заполнения хеш-таблицы хеш-функцией.

В случае деревьев, АВЛ дерево не всегда выигрывает по времени поиска у несбалансированного дерева, так как порядок вершин при балансировке меняется, но всегда выигрывает по среднему значению количества сравнений и среднему времени поиска по дереву, так как организовано лучше

# Ответы на 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 зависимости от их представления: таблица связей с предками/связный список сыновей, матрица/список. При реализации списком память выделяется динамически – под каждый узел, при реализации матрицей - статически.

1. Какие стандартные операции возможны над деревьями?

Основные операции с деревьями: обход дерева, поиск по дереву, включение в дерево, исключение из дерева.

1. Что такое дерево двоичного поиска?

Дерево двоичного поиска – это такое дерево, в котором все левые потомки моложе предка, а все правые – старше. (или наоборот)

1. Чем отличается идеально сбалансированное дерево от АВЛ дерева?

Дерево, у которого число вершин в левом и правом поддеревьях отличается не более, чем на единицу называется идеально сбалансированным. Двоичное дерево, у каждого узла которого высота двух поддеревьев отличается не более чем на единицу называется АВЛ-деревом.

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

Поиск в АВЛ дереве происходит быстрее, чем в ДДП. Поиск в АВЛ дереве имеет сложность О(logn), в то время как в обычном ДДП сложность О(n).

1. 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.

1. Что такое коллизии? Каковы методы их устранения.

Коллизия – ситуация, когда разным ключам соответствует одно значение хеш-функции. Существует несколько возможных вариантов разрешения коллизий: внешнее (открытое) хеширование (метод цепочек) и внутреннее (закрытое) хеширование (открытая адресация).

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

Если для поиска элемента необходимо более 3–4 сравнений, то эффективность использования такой хеш-таблицы пропадает и ее следует реструктуризировать (т.е. найти другую хеш-функцию), чтобы минимизировать количество сравнений для поиска элемента

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

В хэш-таблице минимальное время поиска О(1). В АВЛ дереве О(log2n). В дереве двоичного поиска О(h), где h – высота дерева.