**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 по курсу «Анализ алгоритмов»

**Тема** Алгоритмы поиска

**Студент** Жаворонкова А. А.

**Группа** ИУ7-56Б

**Оценка (баллы)**

**Преподаватель** Волкова Л. Л.

Москва — 2023 г.

# Содержание

**ВВЕДЕНИЕ** **3**

**1 Аналитическая часть** **4**

1.1 Двоичное дерево поиска . . . . . 4

1.2 АВЛ-дерево . . . . . 4

1.3 Вывод . . . . . 5

**2 Конструкторская часть** **7**

2.1 Разработка алгоритмов . . . . . 7

2.2 Классы эквивалентности тестирования . . . . . 7

2.3 Вывод . . . . . 8

**3 Технологическая часть** **9**

3.1 Средства реализации . . . . . 9

3.2 Описание используемых типов данных . . . . . 9

3.3 Сведения о модулях программы . . . . . 10

3.4 Реализация алгоритмов . . . . . 10

3.5 Функциональные тесты . . . . . 11

3.6 Вывод . . . . . 12

**4 Исследовательская часть** **13**

4.1 Технические характеристики . . . . . 13

4.2 Демонстрация работы программы . . . . . 13

4.3 Количество сравнений при поиске элемента . . . . . 15

4.4 Вывод . . . . . 16

**ЗАКЛЮЧЕНИЕ** **17**

**СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ** **18**

# ВВЕДЕНИЕ

**Целью** данной работы является изучение алгоритмов поиска. Для достижения поставленной цели необходимо выполнить следующие задачи:

— описать алгоритмы поиска в двоичном и АВЛ-дереве;

— реализовать указанные алгоритмы;

— провести тестирование по методу черного ящика для реализаций указанных алгоритмов;

— провести сравнительный анализ зависимости количества элементов в дереве от количества сравнений;

— описать и обосновать полученные результаты в отчете о выполненной лабораторной работе, выполненного как расчетно-пояснительная записка к работе.

# 1 Аналитическая часть

В данном разделе будут рассмотрены алгоритмы поиска в двоичном и АВЛ-дереве.

## 1.1 Двоичное дерево поиска

Дерево двоичного поиска — это структура данных, которая позволяет быстро работать с отсортированном списком чисел [3]. Характерные особенности двоичного дерева поиска:

— Все узлы левого поддерева меньше корневого узла;

— Все узлы правого поддерева больше корневого узла;

— Оба поддерева каждого узла тоже являются деревьями двоичного поиска, то есть также обладают первыми двумя свойствами.

**Поиск элемента.**

Если значение меньше корня, то оно находится не в правом поддереве. Поиск выполняется в левом поддереве. А если значение больше корня, то значения нет в левом поддереве. Тогда поиск выполняется в правом поддереве.

*Лучший случай:* искомый элемент расположен в корне дерева. Сложность: $O(1)$.

*Худший случай:* искомый элемент расположен в листе дерева или его нет в дереве. Сложность: $O(n)$, где $n$ — высота дерева.

## 1.2 АВЛ-дерево

АВЛ-дерево отличается от обычного бинарного дерева поиска несколькими особенностями [2]:

— оно сбалансировано по высоте. Поддеревья, которые образованы левым и правым потомками каждого из узлов, должны различаться длиной не более чем на один уровень;

— из первой особенности вытекает еще одна — общая длина дерева и, соответственно, скорость операций с ним зависят от числа узлов логарифмически и гарантированно;

— вероятность получить сильно несбалансированное АВЛ-дерево крайне мала, а риск, что оно выродится, практически отсутствует.

**Поиск элемента** аналогичен поиску в двоичном дереве.

*Лучший случай:* искомый элемент расположен в корне дерева. Сложность: $O(1)$.

*Худший случай:* искомый элемент расположен в листе дерева или его нет в дереве. Сложность: $O(n)$, где $n$ — высота дерева.

Отличие в том, что высота АВЛ-дерева будет меньше или равна высоте двоичного дерева, а значит поиск будет выполняться быстрее.

## 1.3 Вывод

В данном разделе были теоретически разобраны алгоритмы поиска в двоичном и АВЛ-дереве

К разрабатываемой программе предъявляются следующие требования:

1. Программа должна предоставлять функциональность поиска элемента в дереве;

2. Реализуемое ПО будет работать в двух режимах — пользовательском, в котором можно выбрать алгоритм и вывести для него результат, а также экспериментальном режиме, в котором можно произвести сравнение реализаций алгоритмов по количеству сравнений на различных входных данных;

3. В первом режиме в качестве входных данных в программу будет подаваться двоичное дерево, также реализовано меню для вызова алгоритмов и замеров времени. Программа должна корректно обрабатывать случай поиска несуществующего элемента дерева;

4. Во втором режиме будет происходить измерение количества сравнений, будут построены зависимости количества элементов дерева от количества сравнений при поиске. Деревья будут сгенерированы автоматически для заданного количества элементов.

# 2 Конструкторская часть

В этом разделе будут представлены псевдокоды алгоритмов поиска в двоичном и АВЛ-дереве.

## 2.1 Разработка алгоритмов

Поиск в двоичном и АВЛ-дереве выполняется по одному алгоритму. В листинге 2.1 представлен псевдокод алгоритмов поиска в двоичном и АВЛ-дереве.

Листинг 2.1 – Псевдокод алгоритма поиска в двоичном (АВЛ) дереве

```
1 Функция поиска (дерево, значение):
2     если дерево — пусто:
3         возврат "Элемент␣не␣найден"
4     если значение равно значению корневого узла дерева:
5         возврат "Элемент␣найден"
6     если значение меньше значения корневого узла:
7         вызвать функцию поиска (левое поддерево, значение)
8     иначе:
9         вызвать функцию поиска (правое поддерево, значение)
```

## 2.2 Классы эквивалентности тестирования

Для тестирования выделены следующие классы эквивалентности:

1. искомый элемент расположен в корне дерева;
2. искомый элемент расположен в листе дерева;
3. двоичное дерево — вырожденное;
4. искомого элемента нет в дереве;
5. дерево состоит из одного узла;
6. дерево пустое.

## 2.3 Вывод

В данном разделе были представлены псевдокоды алгоритмов, рассматриваемых в лабораторной работе.

# 3 Технологическая часть

В данном разделе будут рассмотрены средства реализации, а также представлены листинги реализаций алгоритмов поиска в двоичном и АВЛ-дереве.

## 3.1 Средства реализации

В данной работе для реализации был выбран язык программирования *C++* [1].

## 3.2 Описание используемых типов данных

При реализации будут использованы следующие типы и структуры данных:

- *Node* — структура для узла двоичного дерева, содержащая 3 поля:
  - *int* — поле для данных;
  - *Node** — указатель на левое поддерево;
  - *Node** — указатель на правое поддерево;
- *NodeAvl* — структура для узла АВЛ-дерева, содержащая 4 поля:
  - *int* — поле для данных;
  - *NodeAvl** — указатель на левое поддерево;
  - *NodeAvl** — указатель на правое поддерево;
  - *int* — поле, содержащее высоту дерева;

## 3.3 Сведения о модулях программы

Программа состоит из трех модулей:

— *main.cpp* — файл, содержащий код для выбора алгоритма, а также замеров количества сравнений;

— *avl_tree.cpp* — файл, обеспечивающий взаимодействие с АВЛ-деревом;

— *tree.cpp* — файл, обеспечивающий взаимодействие с двоичным деревом;

## 3.4 Реализация алгоритмов

В листингах 3.1 – 3.3 представлены реализации алгоритмов поиска в двоичном и АВЛ-дереве.

Листинг 3.1 – Реализация алгоритма поиска в АВЛ-дереве

```
1  bool searchAvl(NodeAvl* root, int key, int& count_compares) {
2      if (root == NULL) {
3          return false;
4      } else if (root->key == key) {
5          count_compares++;
6          return true;
7      } else if (key < root->key) {
8          count_compares++;
9          return searchAvl(root->left, key, count_compares);
10     } else {
11         count_compares++;
12         return searchAvl(root->right, key, count_compares);
13     }
14 }
```

Листинг 3.2 – Реализация алгоритма поиска в двоичном дереве (начало)

```
1  bool search(Node* root, int value, int& count_compares) {
2      if (root == NULL) {
3          return false;
4      } else if (root->data == value) {
5          count_compares++;
6          return true;
```

Листинг 3.3 – Реализация алгоритма поиска в двоичном дереве (окончание)

```
        return true;
    } else if (value <= root->data) {
        count_compares++;
        return search(root->left, value, count_compares);
    } else {
        count_compares++;
        return search(root->right, value, count_compares);
    }
}
```

## 3.5 Функциональные тесты

В таблице 3.1 приведены тесты для функций, реализующих алгоритмы поиска в двоичном и АВЛ-дереве. В качестве результата ожидается сообщение (найден ли элемент). Тесты *для всех алгоритмов* пройдены успешно.

Таблица 3.1 – Функциональные тесты

| Значения, содержащиеся в узлах дерева | Искомый элемент | Ожидаемый результат | Результат поиска в АВЛ-дереве | Результат поиска в двоичном дереве |
|---|---|---|---|---|
| {0, 1, 2, 3} | 0 | «Элемент найден» | «Элемент найден» | «Элемент найден» |
| {0, 1, 2, 3} | 3 | «Элемент найден» | «Элемент найден» | «Элемент найден» |
| {0} | 0 | «Элемент найден» | «Элемент найден» | «Элемент найден» |
| {0} | 1 | «Элемент не найден» | «Элемент не найден» | «Элемент не найден» |
| {} | 1 | «Элемент не найден» | «Элемент не найден» | «Элемент не найден» |

| | | | | |
|---|---|---|---|---|
| {0, 1, 2, 3} | 5 | «Элемент не найден» | «Элемент не найден» | «Элемент не найден» |

## 3.6 Вывод

Были представлены реализации алгоритмов поиска в двоичном и АВЛ-дереве, которые были описаны в предыдущем разделе. Также в данном разделе была приведена информация о выбранных средствах для разработки алгоритмов.

# 4 Исследовательская часть

В данном разделе будут приведены примеры работы программы, а также проведен сравнительный анализ алгоритмов при различных ситуациях на основе полученных данных.

## 4.1 Технические характеристики

Технические характеристики устройства, на котором выполнялось тестирование представлены далее:

— операционная система: Windows 11, x64;

— оперативная память: 8 Гб;

— процессор: AMD Ryzen 5 5500U с видеокартой Radeon Graphics 2.10 ГГц.

Во время замеров времени ноутбук был нагружен только встроенными приложениями окружения.

## 4.2 Демонстрация работы программы

На рисунке 4.1 представлен результат работы программы.

```
1. Поиск в несбалансированном дереве
2. Поиск в сбалансированном дереве
3. Замерить время

0. Выход

Выберите пункт меню: 1
Введите размер дерева: 10
   4
   0 ---   8
     ---   2 ---   6 ---   9
     ---     ---   1 ---   3 ---   5 ---   7 ---     ---
Введите искомый элемент: 5
Элемент найден
Количество сравнений: 4
1. Поиск в несбалансированном дереве
2. Поиск в сбалансированном дереве
3. Замерить время

0. Выход

Выберите пункт меню: 2
Введите размер дерева: 10
   5
   2 ---   8
   0 ---   3 ---   6 ---   9
     ---   1 ---     ---   4 ---     ---   7 ---     ---
Введите искомый элемент: 20
Элемент НЕ найден
Количество сравнений: 3
```

Рисунок 4.1 – Пример работы программы

## 4.3 Количество сравнений при поиске элемента

Результаты замеров приведены в таблице 4.1.

Таблица 4.1 – Результаты замеров количества сравнений

| Количество элементов в дереве | Количество сравнений в АВЛ-дереве | Количество сравнений в двоичном дереве |
|---|---|---|
| 128 | 8 | 14 |
| 256 | 9 | 17 |
| 512 | 10 | 18 |
| 1024 | 11 | 21 |
| 2048 | 12 | 27 |

На рисунке 4.2 приведена визуализация результатов замеров.

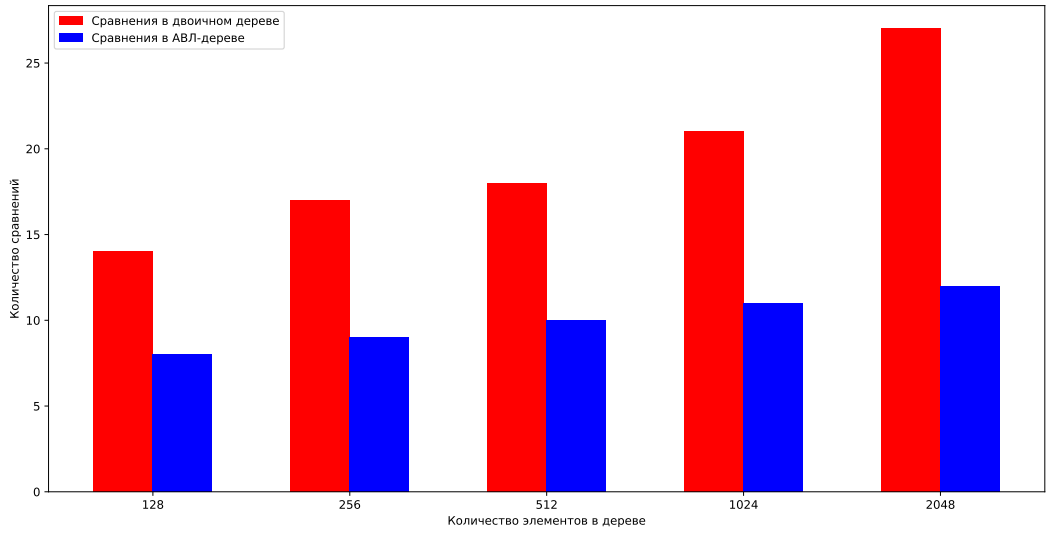


Рисунок 4.2 – Визуализация результатов замеров

## 4.4 Вывод

Как видно из графика 4.2, при поиске в АВЛ-дереве выполняется меньше сравнений. Например, при количестве элементов в дереве 1024 и более количество сравнений в АВЛ-дереве в 2 раза меньше количества сравнений в двоичном дереве. Связано это с тем, что высота АВЛ-дерева меньше высоты двоичного дерева с тем же количеством элементов, а значит требуется меньшее количество сравнений.

# ЗАКЛЮЧЕНИЕ

В результате было получено, что поиск в АВЛ-дереве требует меньше сравнений, чем в двоичном дереве в связи с тем, что высота АВЛ-дерева меньше высоты двоичного дерева. При количестве элементов в дереве 1024 и более для поиска элемента в двоичном дереве требуется в 2 раза больше сравнений, чем для поиска в АВЛ-дереве.

Цель, которая была поставлена в начале лабораторной работы была достигнута: изучены алгоритмов поиска. В ходе выполнения были решены все задачи:

— описаны алгоритмы поиска в двоичном и АВЛ-дереве;

— реализованы указанные алгоритмы;

— проведено тестирование по методу черного ящика для реализаций указанных алгоритмов;

— проведен сравнительный анализ зависимости количества элементов в дереве от количества сравнений;

— описаны и обоснованы полученные результаты в отчете о выполненной лабораторной работе, выполненного как расчетно-пояснительная записка к работе.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ


1. C++ documnetation. — URL: `https://cplusplus.com/` (дата обр. 01.12.2023).
2. АВЛ-дерево (AVL-Tree) - что это: построение, балансировка, удаление [Электронный ресурс]. — URL: `https://blog.skillfactory.ru/glossary/avl-derevo/` (дата обр. 20.12.2023).
3. Дерево двоичного поиска [Электронный ресурс]. — URL: `https://codechick.io/tutorials/dsa/dsa-binary-search-tree` (дата обр. 20.12.2023).