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 работа №6

по курсу «Анализ Алгоритмов»

на тему: «Алгоритмы поиска»

Студент ИУ7-53Б _____ Лысцев Н. Д.
(Группа) (Подпись, дата) (И. О. Фамилия)

Преподаватель _____ Волкова Л. Л.
(Подпись, дата) (И. О. Фамилия)

*2024 г.*

# СОДЕРЖАНИЕ

# ВВЕДЕНИЕ

С развитием компьютерной техники проблема хранения и обработки больших объемов данных становилась все более актуальной. Возникла необходимость организации хранилища для больших объемов данных, которое предоставляет возможность быстро находить и модифицировать данные. Один из способов организации такого хранилища — двоичные деревья поиска [1].

Целью данной лабораторной работы является исследование лучших и худших случаев алгоритма поиска целого числа в несбалансированном двоичном дереве поиска (ДДП) и сбалансированном (АВЛ-дереве).

Для достижения поставленной цели необходимо решить следующие задачи:

1) описать используемые алгоритмы поиска;
2) выбрать средства программной реализации;
3) реализовать данные алгоритмы поиска;
4) проанализировать алгоритмы по количеству сравнений.

# 1 Аналитический раздел

В данном разделе будет рассмотрено понятие двоичного дерева поиска, АВЛ-дерева, было дано описание алгоритма поиска в двоичном дерева поиска.

## 1.1 Двоичное дерево поиска

Двоичное дерево представляет собой в общем случае неупорядоченный набор узлов, который либо пуст (пустое дерево), либо разбит на три непересекающиеся части:

– узел, называемый корнем;

– двоичное дерево, называемое левым поддеревом;

– двоичное дерево, называемое правым поддеревом.

Таким образом, двоичное дерево — это рекурсивная структура данных.

Каждый узел двоичного дерева можно представить в виде структуры данных, состоящей из следующих полей:

– данные, обладающие ключом, по которому их можно идентифицировать;

– указатель на левое поддерево;

– указатель на правое поддерево;

– указатель на родителя (необязательное поле);

Значение ключа уникально для каждого узла.

Дерево поиска — это двоичное дерево, в котором узлы упорядочены определенным образом по значению ключей: для любого узла $X$ значения ключей всех узлов его левого поддерева меньше значения ключа $X$, а значения ключей всех узлов его правого поддерева больше значения ключа $X$ [1].

## 1.2 АВЛ-дерево

Важной характеристикой двоичного дерева поиска, непосредственно влияющей на скорость поиска данных является коэффициент сбалансированности. Коэффициентом сбалансированности называют некоторую константу $k$, на которую могут отличаться высоты левого и правого поддерева любого произвольного узла $X$.

Таким образом АВЛ-дерево — это двоичное дерево поиска, для которого определен коэффициент сбалансированности k = 1 [2].

## 1.3 Алгоритм поиска в двоичном дереве поиска

Процедура поиска узла по ключу заключается в том, что на каждом шаге значение искомого ключа сравнивается со значением ключа рассматриваемого узла, начиная с корня. Если значение искомого ключа меньше, чем значение ключа рассматриваемого узла, то поиск продолжается в левом поддереве, если больше — то в правом поддереве. И так, пока не будет найден узел с искомым ключом или пока поиск не достигнет того узла, ниже которого этот узел не может находиться. Если при поиске мы обнаруживаем, что узел далее надо искать, например, в правом поддереве, а оно пусто, следовательно, мы можем сделать вывод, что искомого ключа в дереве нет [1].

## Вывод

В данном разделе было рассмотрено понятие двоичного дерева поиска, АВЛ-дерева, было дано описание алгоритма поиска в двоичном дерева поиска.

# 2 Конструкторский раздел

## 2.1 Алгоритм поиска

Определим следующие операторы и функции:

– оператор $\leftarrow$ обозначает присваивание значение переменной;

– функция $root(T)$ возвращает узел — корень дерева $T$;

– функция $key(x)$ возвращает ключ узла $x$;

– функция $left(x)$ возвращает узел — левое поддерево узла $x$;

– функция $right(x)$ возвращает узел — правое поддерево узла $x$;

На рисунке 2.1 представлен псевдокод рекурсивного алгоритма поиска узла по ключу в ДДП и АВЛ-дереве.

**Листинг 2.1** Псевдокод рекурсивного алгоритма поиска узла по ключу в ДДП и АВЛ-дереве

**На входе**: дерево $T$, $k$ — значение ключа

```
1: function TreeRecSearch(T, k)
2:     x ← root(T)
3:     if x = NULL and k = key(x) then
4:         return x
5:     end if
6:     if k < key(x) then
7:         return TreeRecSearch(left(x), k)
8:     else
9:         return TreeRecSearch(right(x), k)
10:    end if
11: end function
```

## Вывод

В данном разделе были описан псевдокод для алгоритма поиска значения по ключу в несбалансированнном двоичном дереве поиска и в АВЛ-дереве.

# 3 Технологический раздел

В данном разделе будут перечислены требования к программному обеспечению, средства реализации и листинги кода.

## 3.1 Требования к программному обеспечению

К программе предъявляется ряд требований:

- наличие меню для взаимодействия с программой;
- предоставление интерфейса для выбора файла, содержащего данные для построения ДДП и АВЛ-дерева, для ввода количества узлов в дереве и генерации нового файла со значениями узлов;
- предоставление интерфейса для осуществления поиска целого числа в ДДП и АВЛ-дереве.

## 3.2 Средства реализации

В качестве языка программирования для этой лабораторной работы был выбран $C++$ [3], поскольку в нем есть встроенный модуль *ctime*, предоставляющий необходимый функционал для замеров процессорного времени:

В качестве функции, которая будет осуществлять замеры процессорного времени, будет использована функция *clock_gettime* из встроенного модуля *ctime* [5].

## 3.3 Сведения о модулях программы

Программа состоит из пяти модулей:

1) `main.cpp` — файл, содержащий точку входа в программу;
2) `interface.cpp` — модуль, содержащий функции для взаимодействия с пользователем;

3) `measurements.cpp` — модуль, содержащий функции для проведения замеров количества сравнений в алгоритме поиска;

4) `tree.cpp` — модуль, содержащий описание бинарного дерева и набора функций для работы с бинарным деревом.

## 3.4 Реализации алгоритмов

## Вывод

В данном разделе были перечислены требования к программному обеспечению, средства реализации и листинги кода.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ


1. *Сенюкова О. В.* Сбалансированные деревья поиска. — М. : Издательский отдел факультета ВМиК МГУ имени М.В. Ломоносова, 2014.
2. АВЛ-деревья, выполнение операций над ними [Электронный ресурс]. — Режим доступа: `https://cyberleninka.ru/article/n/avl-derevya-vypolnenie-operatsiy-nad-nimi/viewer` (дата обращения: 04.02.2024).
3. Справочник по языку C++ [Электронный ресурс]. — Режим доступа: `https://learn.microsoft.com/ru-ru/cpp/cpp/cpp-language-reference?view=msvc-170` (дата обращения: 28.09.2022).
4. std::thread [Электронный ресурс]. — Режим доступа: `https://en.cppreference.com/w/cpp/thread/thread` (дата обращения: 27.01.2024).
5. clock_getres [Электронный ресурс]. — Режим доступа: `https://pubs.opengroup.org/onlinepubs/9699919799/functions/clock_getres.html` (дата обращения: 28.09.2022).