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 №7

по курсу «Анализ Алгоритмов»

на тему: «Алгоритмы поиска»

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

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

2024 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ **3**

1 Аналитический раздел **4**

1.1 Двоичное дерево поиска 4

1.2 АВЛ-дерево 5

1.3 Алгоритм поиска в двоичном дереве поиска 5

2 Конструкторский раздел **6**

2.1 Алгоритм поиска 6

3 Технологический раздел **8**

3.1 Требования к программному обеспечению 8

3.2 Средства реализации 8

3.3 Сведения о модулях программы 8

3.4 Реализации алгоритмов 9

4 Исследовательский раздел **11**

4.1 Технические характеристики 11

4.2 Проведение исследования 11

ЗАКЛЮЧЕНИЕ **16**

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ **17**

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 используемый алгоритм поиска;
2) выбрать средства программной реализации;
3) реализовать данный алгоритм поиска;
4) проанализировать алгоритм 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* [4], предоставляющий необходимый функционал для замеров процессорного времени.

3.3 Сведения о модулях программы

Программа состоит из четырех модулей:

1) `main.cpp` — файл, содержащий точку входа в программу;

2) `interface.cpp` — модуль, содержащий функции для взаимодействия с пользователем;

3) `measurements.py` — 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

В листинге 3.1 представлены структуры «*tree_t*» и «*vertex_t*», содержащие описание бинарного дерева и одной вершины бинарного дерева соответственно.

Листинг 3.1 – Реализация структур данных, описывающих бинарное дерево

```
typedef struct vertex_t vertex_t;
typedef struct tree_t tree_t;

struct vertex_t
{
    int data;
    int height;
    vertex_t *left;
    vertex_t *right;
};

struct tree_t
{
    vertex_t *root;
};
```

В листингах 3.2 и 3.3 представлена реализация функции поиска в ДДП и АВЛ-дереве.

Листинг 3.2 – Реализация функции поиска в ДДП и АВЛ-дереве (начало)

```
vertex_t *search(vertex_t *root, int data, int *count_compare)
{
    (*count_compare)++;

    if (!root)
    {
        return NULL;
    }
    if (data < root->data)
    {
        return search(root->left, data, count_compare);
    }
```

Листинг 3.3 – Реализация функции поиска в ДДП и АВЛ-дереве (конец)

```
    if (data > root->data)
    {
        return search(root->right, data, count_compare);
    }

    return root;
}
```

Вывод

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

4 Исследовательский раздел

В данном разделе будут приведены постановка исследования и сравнительный анализ алгоритмов на основе полученных данных.

4.1 Технические характеристики

Технические характеристики устройства, на котором проводились исследования:

- операционная система: Ubuntu 22.04.3 LTS x86_64 [5];
- оперативная память: 16 Гб;
- процессор: 11th Gen Intel® Core™ i7-1185G7 @ 3.00 ГГц × 8, 4 физических ядра, 8 логических ядер.

4.2 Проведение исследования

Определим лучший и худший случаи в алгоритме поиска в ДДП и в АВЛ-дереве:

- `лучший случай`, когда искомый элемент находится в корне дерева;
- `худший случай`, когда искомый элемент находится в листе на максимальной высоте дерева, либо случай отсутствия искомого элемента в дереве.

Цель исследования

Целью исследования является проведение сравнительного анализа количества сравнений, необходимых для решения задачи поиска в ДДП и АВЛ-дереве в лучшем и худшем случае и обоснование выбора худшего случая в программной реализации.

Наборы варьируемых и фиксированных параметров

Замеры времени проводились для числа узлов в дереве, равном 128, 256, 512, 1024, 2048.

В качестве фиксированного параметра для значения, хранящегося в узле бинарного дерева, был выбран разброс от 0 до 2000.

Результаты исследования

На рисунке 4.1 изображены результаты исследования для лучшего случая.

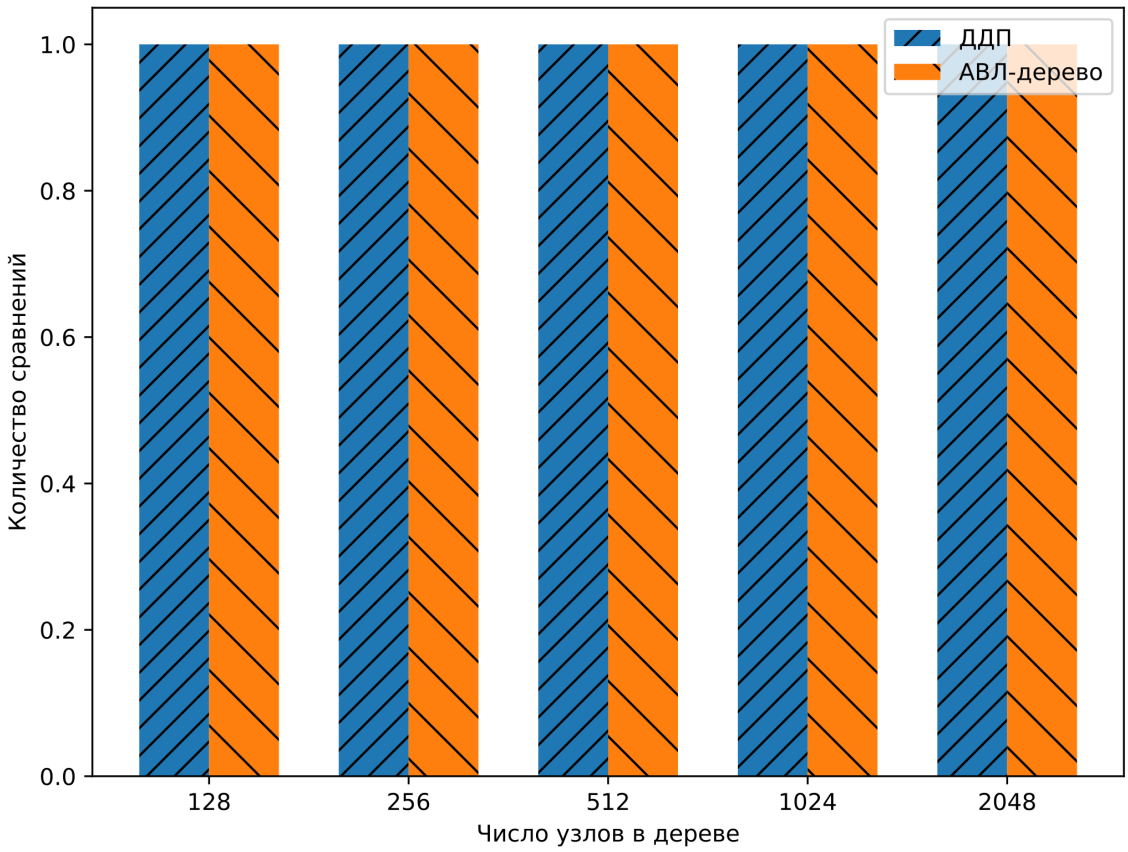

Рисунок 4.1 – Результаты сравнения алгоритма поиска в ДДП и АВЛ-дереве (лучший случай)

На рисунке 4.2 изображены результаты исследования для худшего случая, когда искомый элемент находится в листе на максимальной высоте дерева.

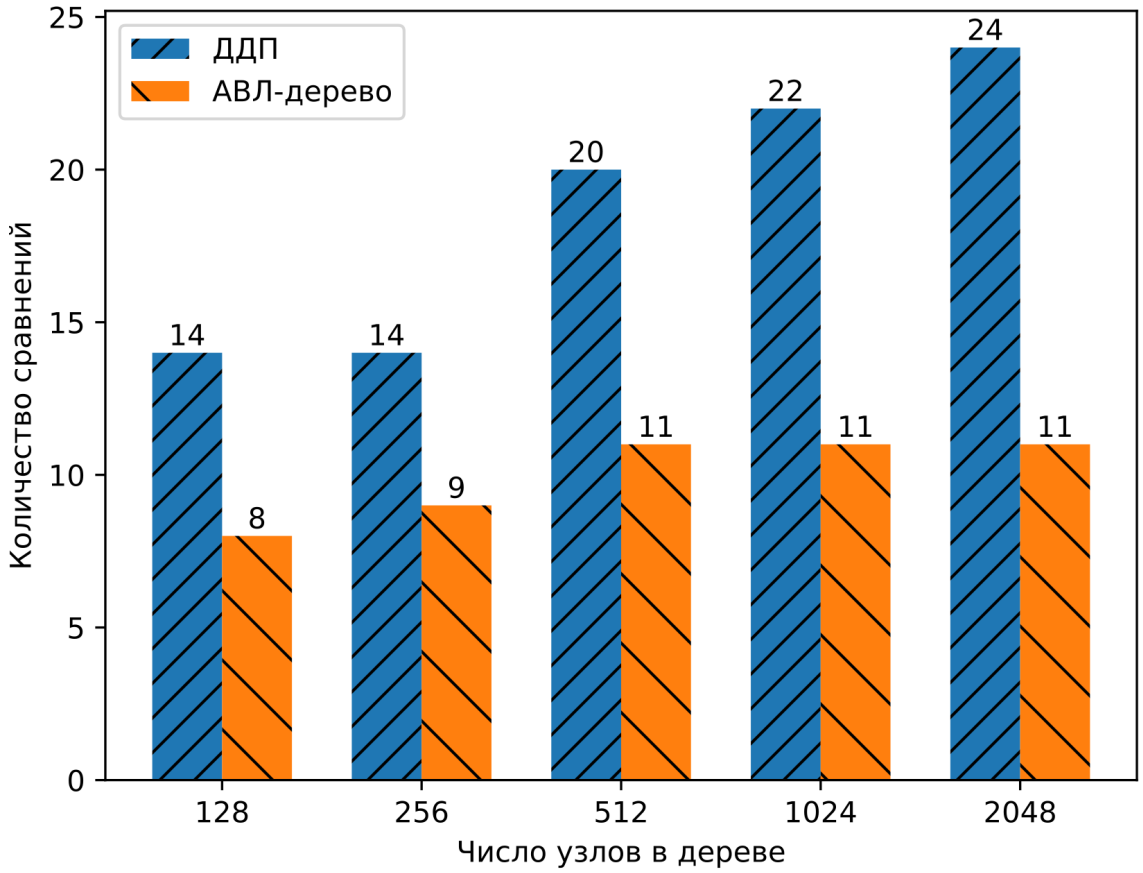

Рисунок 4.2 – Результаты сравнения алгоритма поиска в ДДП и АВЛ-дереве (худший случай)

На рисунке 4.3 изображены результаты исследования для худшего случая, когда искомый элемент отстутствует в дереве.

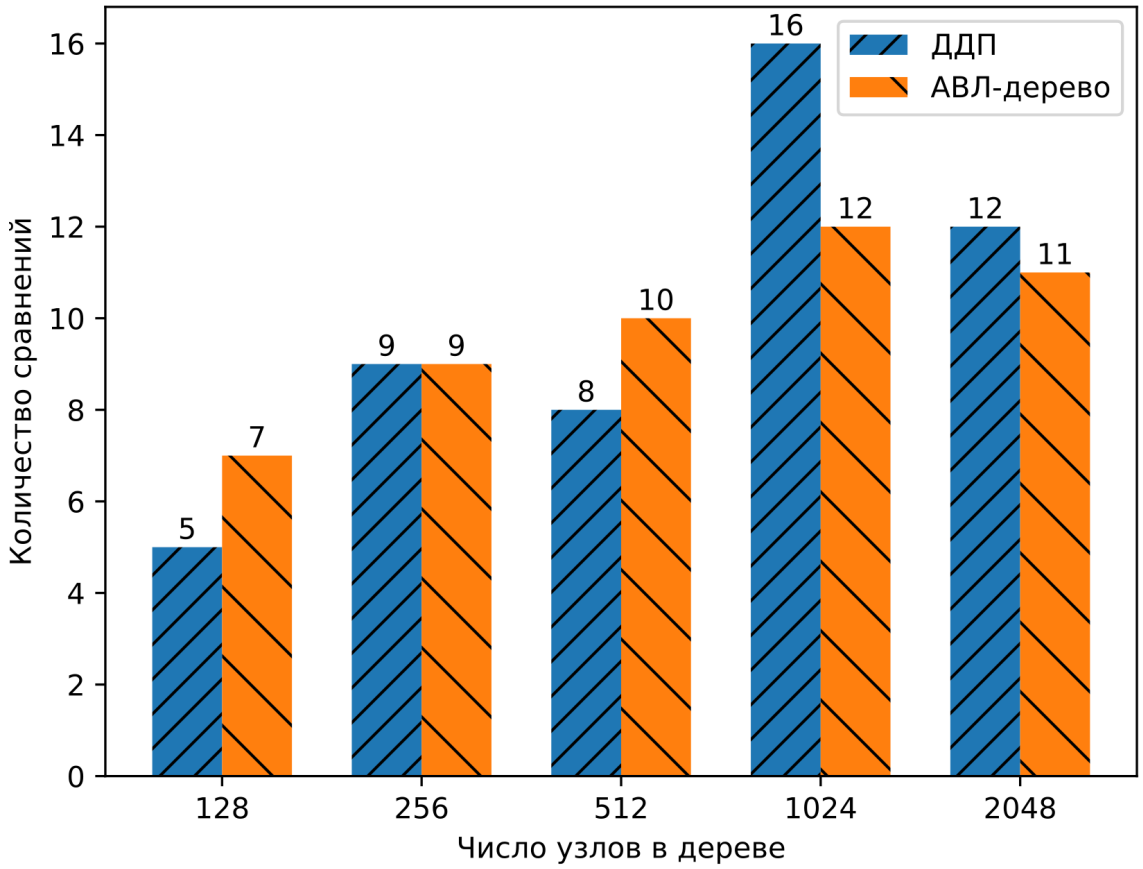

Рисунок 4.3 – Результаты сравнения алгоритма поиска в ДДП и АВЛ-дереве (худший случай 2)

Вывод

Число сравнений в лучшем случае всегда равно единице и не зависит от числа узлов в дереве, так как алгоритм поиска начинает свою работу с корня дерева, производит первое сравнение и находит нужный результат.

С ростом числа узлов растет максимальная высота дерева, и, следовательно растет число сравнений для худшего случая, когда искомый элемент находится в листе на максимальной высоте дерева.

Для случая, когда искомый элемент отсутствует в дереве, число сравнений зависит структуры дерева. Для АВЛ-деревьев среднее число сравнений остается относительно невысоким, так как высота дерева ограничена.

Таким образом, в данной программной реализации алгоритма поиска целого числа в ДДП и АВЛ-дереве худшим случаем считается такой, когда искомый элемент находится в листе на максимальной высоте дерева.

ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы были решены следующие задачи:

1) описан используемый алгоритм поиска;
2) выбраны средства программной реализации;
3) реализован данный алгоритм поиска;
4) проанализирован алгоритм по количеству сравнений.

Цель данной лабораторной работы, а именно исследование лучших и худших случаев алгоритма поиска целого числа в несбалансированном двоичном дереве поиска (ДДП) и сбалансированном (АВЛ-дереве), также была достигнута.

Число сравнений в лучшем случае всегда составляет единицу и не зависит от количества узлов в дереве. Это объясняется тем, что алгоритм поиска начинает свою работу с корня дерева, выполняет первое сравнение и находит требуемый результат.

С увеличением числа узлов в дереве растет максимальная высота дерева, следовательно, возрастает и число сравнений в худшем случае, когда искомый элемент расположен в листе на максимальной высоте дерева.

В случае отсутствия искомого элемента в дереве, количество сравнений зависит от структуры дерева. В сбалансированных деревьях, таких как АВЛ-деревья, среднее число сравнений остается относительно низким, поскольку высота дерева ограничена.

Таким образом, в данной программной реализации алгоритма поиска целого числа в ДДП и АВЛ-дереве, худшим случаем считается ситуация, когда искомый элемент находится в листе на максимальной высоте дерева.

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. clock_getres [Электронный ресурс]. — Режим доступа: `https://pubs.opengroup.org/onlinepubs/9699919799/functions/clock_getres.html` (дата обращения: 28.09.2022).
5. Ubuntu 22.04.3 LTS (Jammy Jellyfish) [Электронный ресурс]. — Режим доступа: `https://releases.ubuntu.com/22.04/` (дата обращения: 28.09.2022).