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»

на тему: «Алгоритмы поиска»

Студент группы ИУ7-54Б _____ (Подпись, дата) Спирин М. П. (Фамилия И.О.)

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

Преподаватель _____ (Подпись, дата) Строганов Ю. В.. (Фамилия И.О.)

Москва — 2023 г.

Содержание

Введение

Каждый день количество информации в цифровом виде в мире увеличивается, что приводит к необходимости разрабатывать новые способы её хранения. Любой новый способ хранения информации должен иметь алгоритм для её извлечения — алгоритм поиска.

Алгоритм поиска — это алгоритм, предназначенный для решения задачи поиска. Задача поиска — это задача извлечения информации из некоторой структуры данных.

Разные структуры данных используют разные алгоритмы поиска, в данной работе будут рассматриваться алгоритмы поиска целого числа в двоичном дереве поиска несбалансированном и сбалансированном (в АВЛ-дереве).

Целью данной лабораторной работы является сравнение алгоритмов поиска в сбалансированном и несбалансированном двоичном дереве поиска.

Для поставленной цели необходимо выполнить следующие задачи.

1) Описать сбалансированное и несбалансированное двоичные деревья поиска.

2) Описать алгоритмы поиска в сбалансированном и несбалансированном дереве поиска.

3) Привести псевдокоды для алгоритма поиска в сбалансированном и несбалансированном деревьях поиска.

4) Определить средства программной реализации.

5) Реализовать разработанные алгоритмы.

6) Выполнить замеры количества сравнений, необходимого для решения задачи поиска элементов в лучшем случае и в худшем случае для выполненных реализаций.

7) Описать и обосновать полученные результаты в отчете о выполненной лабораторной работе.

1 Аналитическая часть

В этом разделе будет представлена информация о сбалансированном и несбалансированном двоичном дереве поиска.

1.1 Двоичное дерево

Сбалансированное и несбалансированное двоичные деревья поиска являются двоичными (бинарными) деревьями.

Двоичное дерево — это вид связного ациклического (не имеющего циклов) графа, характерный тем, что у каждого из его узлов имеется не более двух потомков (связанных узлов, находящихся иерархически ниже) [1].

В двоичном дереве есть только один узел, у которого нет предка, он называется корнем. Конечные узлы называются листьями, у них нет потомков. Все вершины помимо корня и листьев называются узлами ветвления. Длина пути от корня до узла определяет уровень (высоту) этого самого узла. Уровень корня дерева всегда равен нулю, а уровень всех его потомков определяется удаленностью от него.

Пример бинарного дерева представлен на рисунке 1.1.

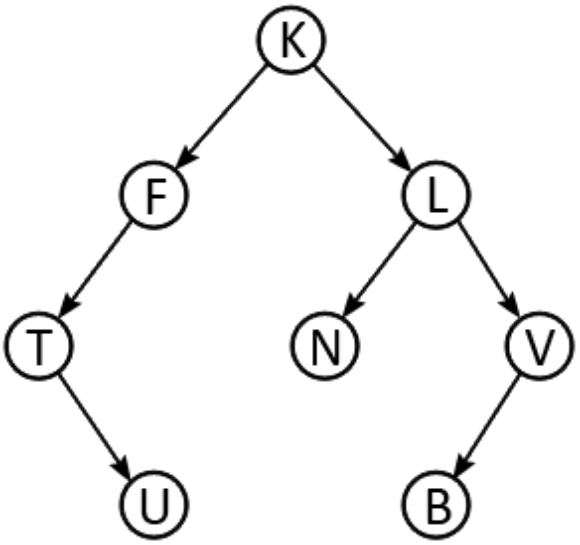

Рисунок 1.1 – Пример бинарного дерева

1.2 Двоичное дерево поиска

Двоичное (или бинарное) дерево поиска (БДП) — это бинарное дерево, обладающее следующим свойством: значение любого из узлов левого поддерева, выходящего из некоторого узла, всегда меньше значения самого узла, тогда как любой узел в правом поддереве не меньше узла [2]. Пример двоичного дерева поиска представлен на рисунке 1.2.

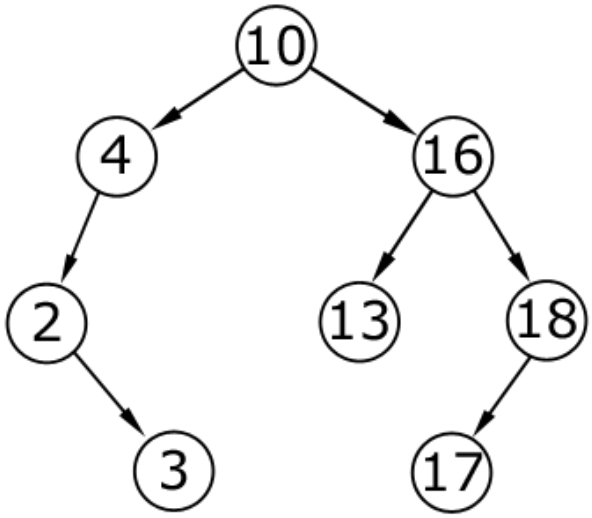

Рисунок 1.2 – Пример бинарного дерева поиска

Основным преимуществом бинарного дерева поиска над обычным бинарным деревом является максимальное количество операций сравнения, необходимых для нахождения значения в дереве. Максимальное количество сравнений для бинарного дерева определяется количеством узлов N, в то время как для бинарного дерева поиска оно определяется максимальной высотой дерева h, причем $h \leq N$.

1.3 Сбалансированное двоичное дерево поиска

Сбалансированное двоичное дерево поиска — это двоичное дерево поиска, в котором высота каждого из поддеревьев, имеющих общий корень,

отличается не более чем на некоторую константу k, и при этом выполняются условия, характерные для двоичного дерева поиска. Если $k = 1$, то такое дерево называется АВЛ-деревом.

Для узлов АВЛ-дерева определяется коэффициент сбалансированности — разность высот правого и левого поддеревьев, принимающая одно значение из множества $\{-1, 0, 1\}$. Ниже изображен пример АВЛ-дерева, каждому узлу которого поставлен в соответствие его реальный коэффициент сбалансированности.

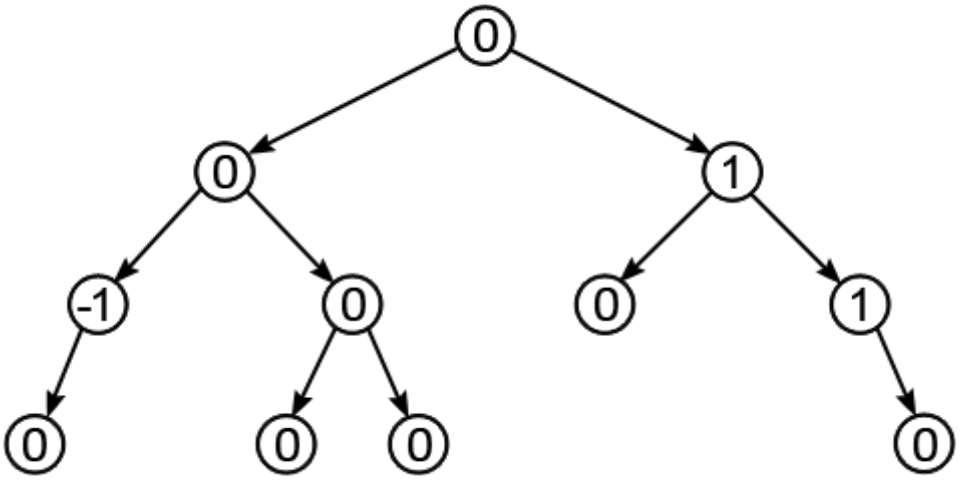

Рисунок 1.3 – Пример коэффициентов сбалансированности АВЛ-дерева

Основным преимуществом АВЛ-дерева над бинарным деревом поиска является меньшее максимальное количество операций сравнения, необходимых для нахождения значения в дереве [3]. Так как на любом уровне l АВЛ-дерева, кроме последнего, количество вершин всегда равно 2^l, то для N вершин высота дерева h будет равна $\log_2 N$. В АВЛ-дереве, как и в бинарном дереве поиска, максимальное количество сравнений ограничено высотой дерева h. Из чего следует, что в АВЛ-дереве, в отличие от бинарного дерева поиска, есть ограничение на максимальное количество сравнений, необходимых для поиска значения в дереве, что может быть важно, например, при использовании рекурсии.

Вывод

В этом разделе была представлена информация о сбалансированном и несбалансированном двоичном дереве поиска.

2 Конструкторская часть

В данном разделе будут представлены листинги псевдокода для алгоритма поиска в сбалансированном и несбалансированном двоичном дереве поиска.

2.1 Требования к ПО

К программе предъявлен ряд требований:

— программа должна предоставить пользователю возможность выбрать сбалансированное или несбалансированное бинарное дерево поиска;

— программа должна дать пользователю возможность заполнить выбранное дерево значениями;

— программа должна дать пользователю возможность провести поиск значения в выбранном бинарном дереве;

— программа должна дать пользователю возможность замерить количество сравнений, необходимое для поиска значения в дереве поиска.

2.2 Описание используемых типов данных

При реализации алгоритмов будут использованы следующие структуры данных:

— узел дерева — содержит значение и указатель на левого и правого потомка;

— бинарное дерево — содержит корневой узел.

2.3 Разработка алгоритмов

Псевдокод для поиска в сбалансированном и несбалансированном бинарном дереве поиска представлен в листинге 2.1.

Листинг 2.1 – Псевдокод для поиска в бинарном дереве поиска

```
1     Входные данные: узел root — корень дерева,
2                     key — искомое значение
3     Выходные данные: node — узел, содержащий искомое значение
4
5     node = root
6     while(True):
7         if node is None or node.key == key then
8             return node
9         end if
10
11        if node.key > key then
12            node = node.left
13        else
14            node = node.right
15        end if
16    end while
```

2.4 Оценка количества сравнений

Для дальнейших замеров количества сравнений необходимо определить, что является лучшим и худшим случаем для разработанного алгоритма.

Лучшим случаем является нахождение числа в корне дерева. В таком случае потребуется 1 сравнение для нахождения искомого числа.

Худшим случаем в данной реализации будет являться отсутствие узла в дереве, так как в таком случае возможно будет выполнить $h+1$ сравнений, где h — высота дерева, в то время как при нахождении искомого элемента на максимальной высоте дерева количество сравнений равно h.

Вывод

В данном разделе были представлены листинги псевдокода для алгоритма поиска в сбалансированном и несбалансированном двоичном дереве поиска.

3 Технологическая часть

В данном разделе рассмотрены средства реализации, а также представлены листинги реализаций рассматриваемых алгоритмов.

3.1 Средства реализации

В данной работе для реализации был выбран язык программирования *Python* [4]. Выбор обусловлен наличием опыта работы с данным языком программирования.

3.2 Сведения о файлах программы

Данная программа разбита на следующие файлы:

— *main.py* — файл, содержащий точку входа;

— *utils.py* — файл, содержащий служебные алгоритмы;

— *avl.py* — файл, содержащий реализацию АВЛ-дерева;

— *binary.py* — файл, содержащий реализацию бинарного дерева поиска.

3.3 Реализация алгоритмов

В листинге 3.1 представлена реализация алгоритма поиска в бинарном дереве поиска.

Листинг 3.1 – Реализация алгоритма поиска в бинарном дереве поиска

```
1     def recSearch(self, root, key):
2         node = root
3         while(True):
4             self.total += 1
5             if node is None or node.key == key:
6                 return node
7
```

```
8                if node.key > key:
9                    node = node.left
10               else:
11                   node = node.right
```

3.4 Функциональные тесты

В таблице 3.1 приведены тесты для функций программы. Все функциональные тесты пройдены *успешно*.

Таблица 3.1 – Функциональные тесты

Узлы дерева	**Искомое число**	**Результат**
1 2 3 4 5	1	Число найдено
1 2 3 4 5	-1	Число не найдено
1 2 3 4 5	a	Сообщение об ошибке
183 12 134 25 67 31	25	Число найдено
183 12 134 25 67 31	31	Число найдено

Вывод

Были представлены листинги алгоритмов поиска в бинарном дереве поиска. Также в данном разделе была приведена информации о выбранных средствах для разработки алгоритмов и сведения о файлах программы, проведено функциональное тестирование.

4 Исследовательская часть

В данном разделе будет приведен пример работы программы, а также проведен сравнительный анализ алгоритма поиска для сбалансированного и несбалансированного бинарного дерева поиска.

4.1 Демонстрация работы программы

На рисунке 4.1 представлен пример работы программы для поиска в сбалансированном бинарном дереве поиска. Осуществляется выбор типа дерева поиска, в него происходит добавление элементов, после чего в нём проводится поиск числа.

```
Меню
1. Сбалансированное дерево поиска
2. Несбалансированное дерево поиска
3. Вывести текущее дерево
4. Добавить элементы в дерево
5. Найти элемент в дереве
6. Провести замеры количества сравнений
0. Выход
Выбор: 1
Меню
1. Сбалансированное дерево поиска
2. Несбалансированное дерево поиска
3. Вывести текущее дерево
4. Добавить элементы в дерево
5. Найти элемент в дереве
6. Провести замеры количества сравнений
0. Выход
Выбор: 4
Введите числа через пробел:
1 2 3 4 5
Меню
1. Сбалансированное дерево поиска
2. Несбалансированное дерево поиска
3. Вывести текущее дерево
4. Добавить элементы в дерево
5. Найти элемент в дереве
6. Провести замеры количества сравнений
0. Выход
Выбор: 5
Введите искомый элемент: 1
Число было найдено
```

Рисунок 4.1 – Пример работы программы

4.2 Замеры количества сравнений

Были произведены замеры необходимого количества сравнений для нахождения числа в дереве в худшем и лучшем случае.

Результаты замеров представлены в таблице 4.1.

Таблица 4.1 – Результаты замеров количества сравнений

Количество узлов	**Количество сравнений**			
	АВЛ		**Несбалансированное**	
	Худший	**Лучший**	**Худший**	**Лучший**
129	9	1	130	1
257	10	1	258	1
513	11	1	514	1
1025	12	1	1026	1
2049	13	1	2050	1

По таблице 4.1 были построены рисунки 4.2 – 4.3, на них можно увидеть зависимость количества сравнений при поиске в худшем случае для АВЛ-дерева и бинарного дерева поиска.

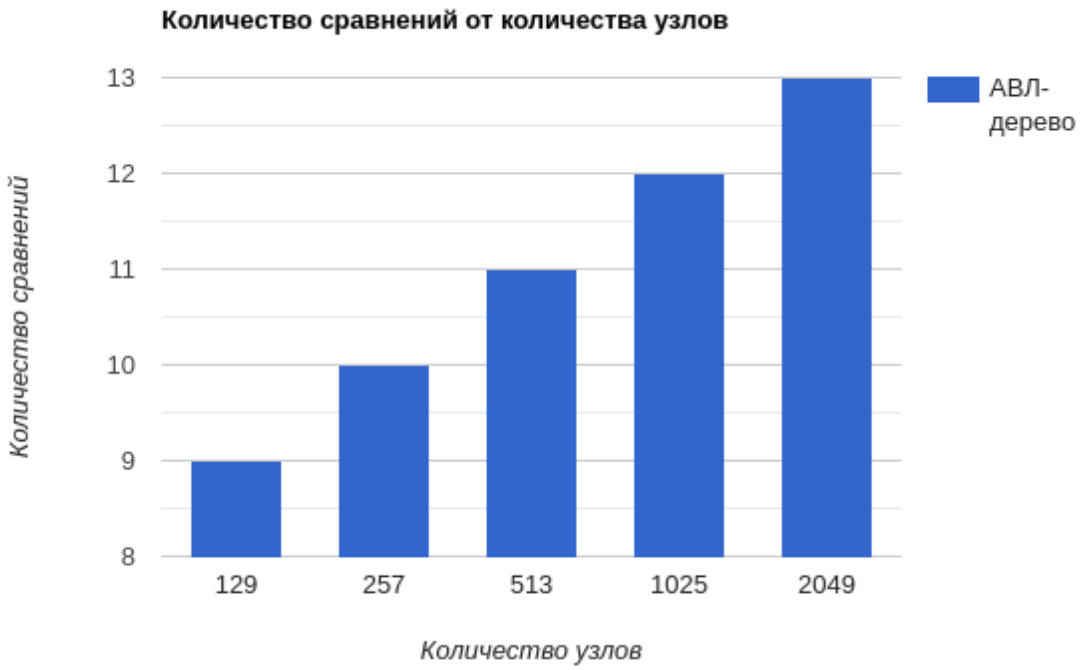

Рисунок 4.2 – Результаты замеров количества сравнений в худшем случае при поиске в АВЛ-дереве

Рисунок 4.3 – Результаты замеров количества сравнений в худшем случае при поиске в несбалансированном бинарном дереве поиска

По полученным данным можно увидеть, что количество сравнений при поиске в АВЛ-дереве действительно логарифмически зависит от количества узлов, в то время как для БДП зависимость линейная, что приводит к разнице в 10 и более раз при количестве узлов большем или равном 129.

Вывод

В результате эксперимента было получено, что в лучшем случае количество сравнений для АВЛ-дерева и несбалансированного бинарного дерева поиска одинаково и равно 1 для любого рассмотренного количества узлов. В худшем случае количество сравнений в АВЛ-дереве более чем в 10 раз меньше, чем в несбалансированном бинарном дереве поиска при количестве узлов большем или равном 129.

Заключение

Поставленная цель достигнута: было проведено сравнение алгоритмов поиска в сбалансированном и несбалансированном бинарном дереве поиска.

В ходе выполнения лабораторной работы были решены все задачи:

1) были описаны сбалансированное и несбалансированное двоичное деревья поиска;

2) были описаны алгоритмы поиска в сбалансированном и несбалансированном дереве поиска;

3) были приведен псевдокод для алгоритма поиска в сбалансированном и несбалансированном деревьях поиска;

4) были определены средства программной реализации;

5) были реализованы разработанные алгоритмы;

6) были выполнены замеры количества сравнений, необходимого для решения задачи поиска элементов в лучшем случае и в худшем случае для выполненных реализаций;

7) были описаны и обоснованы полученные результаты в отчете о выполненной лабораторной работе.

Список использованных источников

1 Ахо, А. Структуры данных и алгоритмы / А. Ахо, Д. Хопкрофт, Д.Ульман. – М.: Вильямс, 2010. – 400 с.

2 Кнут, Д. Искусство программирования. Том 1. Основные алгоритмы/ Д. Кнут. – М.: Вильямс, 2010. – 720 с.

3 Скиена, С. Алгоритмы. Руководство по разработке / С. Скиена. — СПб.: БХВ-Петербург, 2011. – 720 с.

4 Welcome to Python [Электронный ресурс]. Режим доступа: `https://www.python.org` (дата обращения: 23.01.2023).