



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 Маслюков П.В.

Группа ИУ7-52Б

Оценка (баллы) _____

Преподаватель Волкова Л. Л.

Содержание

Введение	3
1 Аналитическая часть	4
1.1 Двоичное дерево	4
1.2 Двоичное дерево поиска	5
1.3 Сбалансированное двоичное дерево поиска	5
2 Конструкторская часть	8
2.1 Описание используемых структур	8
2.2 Разработка алгоритмов	8
2.3 Оценка количества сравнений	9
3 Технологическая часть	10
3.1 Средства реализации	10
3.2 Сведения о файлах программы	10
3.3 Реализация алгоритмов	10
3.4 Функциональные тесты	11
4 Исследовательская часть	12
4.1 Демонстрация работы программы	12
4.2 Замеры времени и количества сравнений	12
Заключение	16
Список использованных источников	17

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 AVL-дереве).

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

Для поставленной цели необходимо выполнить следующие задачи.

- изучить алгоритмы сбалансированного и несбалансированного дерева поиска;
- разработать изученные алгоритмы;
- выполнить замеры количества сравнений, необходимого для решения задачи поиска элементов в лучшем случае и в худшем случае для выполненных реализаций;
- подготовить отчет о выполненной лабораторной работе.

1 Аналитическая часть

В данном 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.

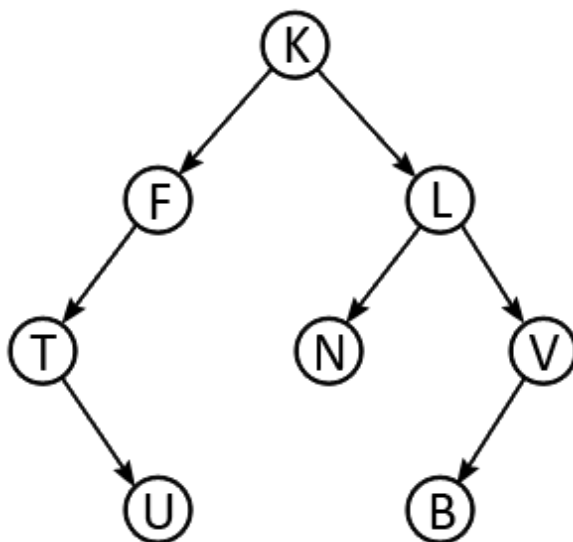


Рисунок 1.1 – Пример бинарного дерева

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 поддерева не меньше узла [2]. Пример двоичного дерева поиска представлен на рисунке 1.2.

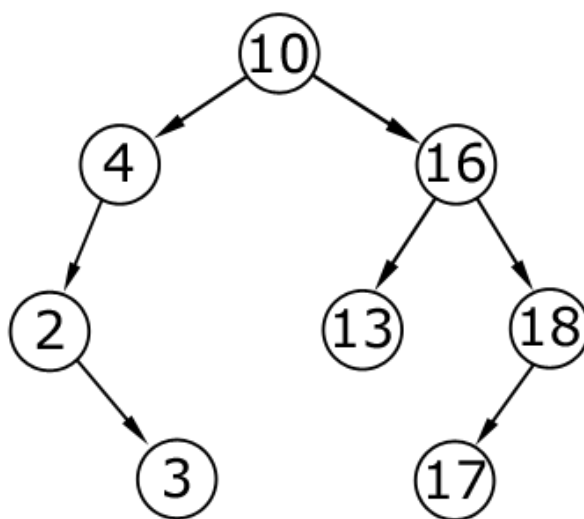


Рисунок 1.2 – Пример бинарного дерева поиска

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.

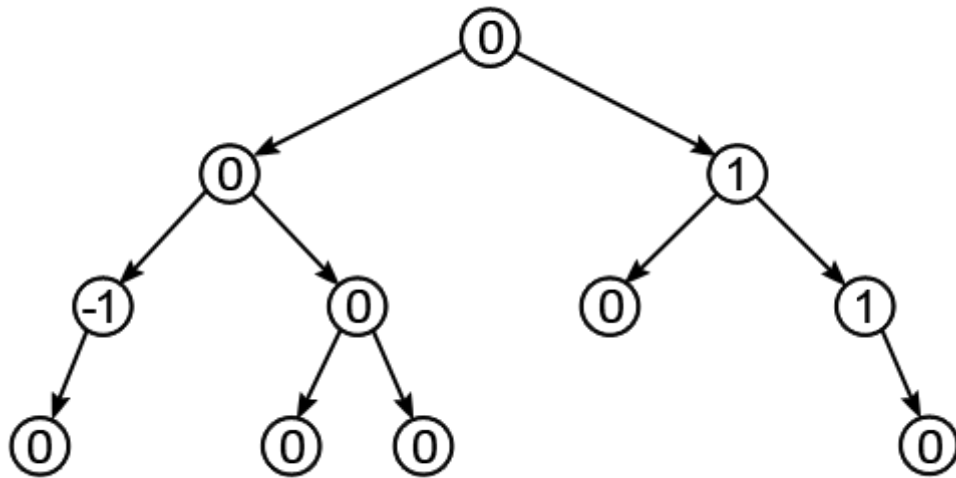


Рисунок 1.3 – Пример коэффициентов сбалансированности АВЛ-дерева

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

2 Конструкторская часть

В данном разделе будут описаны используемые структуры, представлен листинг псевдокода и описан метод оценивания количества сравнений.

2.1 Описание используемых структур

При реализации алгоритмов будут использованы следующие структуры данных:

- `node` — содержит значение и указатель на левого и правого потомка;
- `tree` — содержит корневой узел.

2.2 Разработка алгоритмов

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

Входные данные: `root` - корень дерева, `key` - искомое значение.

Выходные данные: `node` - узел, содержащий искомое значение.

Листинг 2.1 – Псевдокод для поиска в бинарном дереве поиска

```
1  node = root
2  while(True):
3      if node is None or node.key == key then
4          return node
5      end if
6
7      if node.key > key then
8          node = node.left
9      else
10         node = node.right
11     end if
12 end while
```


2.3 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 описаны используемые структуры, представлен листинг псевдокода и описан метод оценивания количества сравнений.

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]. В текущем лабораторном задании необходимо определить количество сравнений, необходимое для поиска элемента в дереве. Все необходимые инструменты для этого имеются в выбранном языке 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.

Листинг 3.1 – Реализация алгоритма 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: 0
Количество элементов, худший случай AVL, лучший случай AVL, худший случай BST, лучший случай BST
259,10,1,260,1
1027,12,1,1028,1
4099,14,1,4100,1
16387,16,1,16388,1
65539,18,1,65540,1
Меню
```

Рисунок 4.1 – Пример работы программы

4.2 Замеры времени и количества сравнений

Были произведены замеры времени нахождения числа в дереве в худшем и лучшем случае.

Результаты замеров в секундах представлены в таблице 4.1.

Таблица 4.1 – Результаты временных замеров

Элементы	Худший AVL	Лучший AVL	Худший BST	Лучший BST
259	0.0011	0.0001	0.0034	0.0001
1027	0.0026	0.0001	0.0434	0.0001
4899	0.0039	0.0001	0.7493	0.0001
16387	0.0075	0.0001	5.1382	0.0001
65539	0.0124	0.0001	53.3482	0.0001

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

Результаты замеров представлены в таблице 4.2.

Таблица 4.2 – Результаты замеров количества сравнений

Элементы	Худший AVL	Лучший AVL	Худший BST	Лучший BST
259	10	1	260	1
1027	12	1	1028	1
4899	14	1	4100	1
16387	14	1	16388	1
65539	16	1	65540	1

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

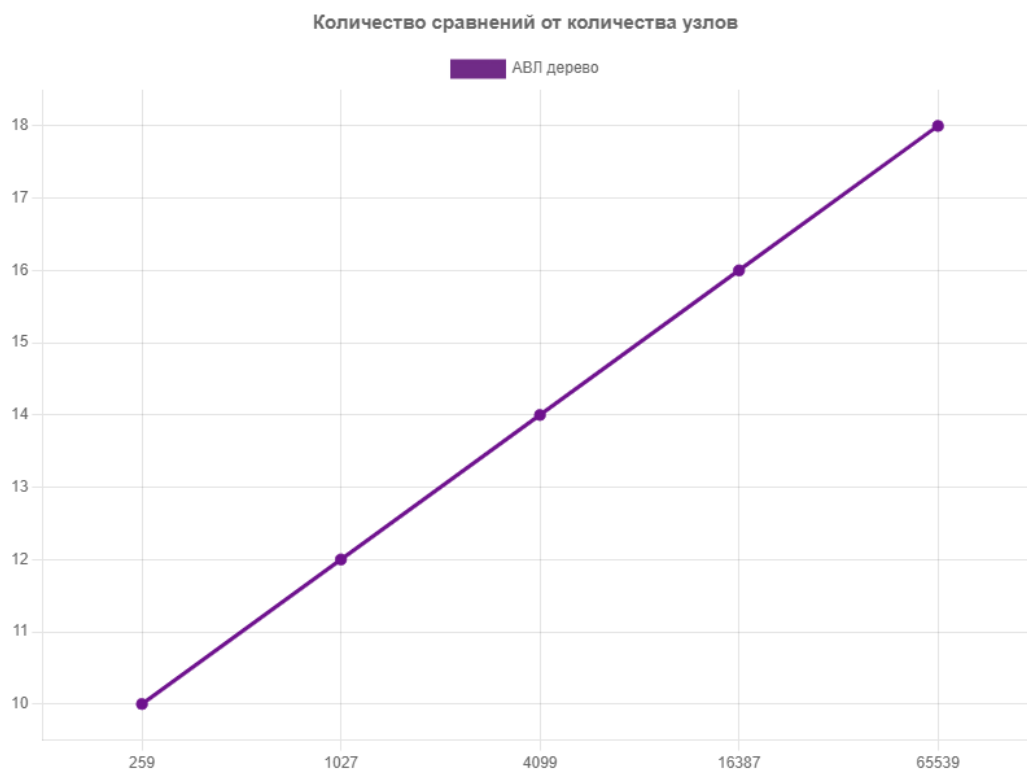


Рисунок 4.2 – Результаты замеров количества сравнений в худшем случае при поиске в AVL-дереве

По полученным данным можно увидеть, что количество сравнений при поиске в AVL-дереве действительно логарифмически зависит от количества узлов, в то время как для БДП зависимость линейная.

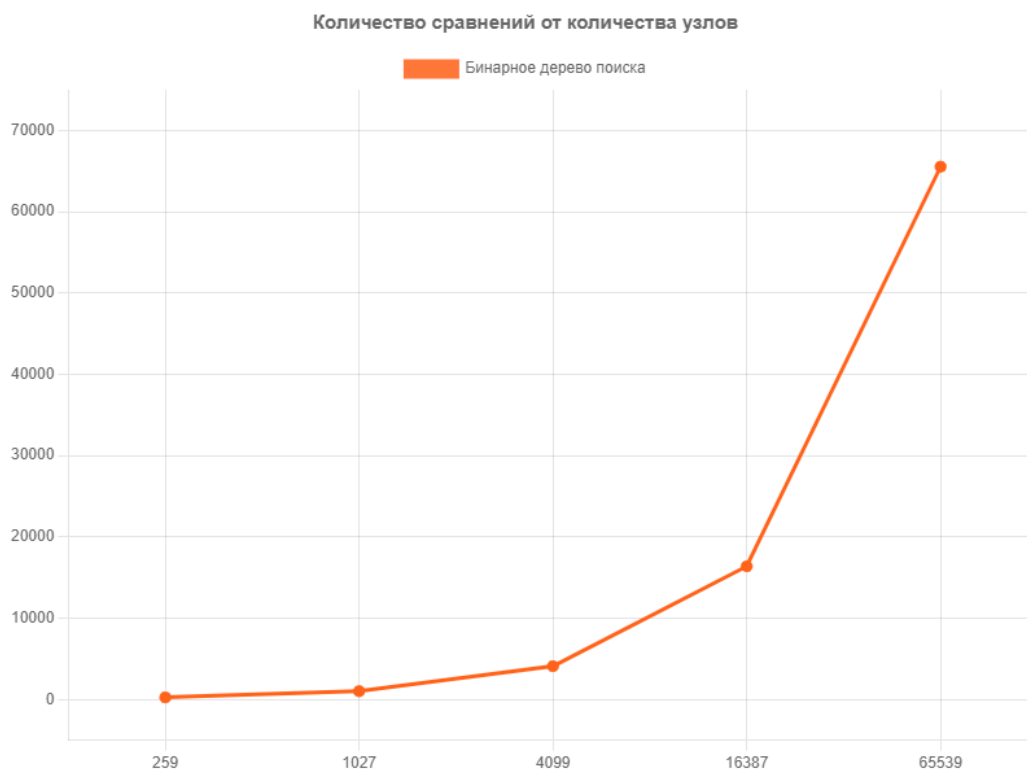


Рисунок 4.3 – Результаты замеров количества сравнений в худшем случае при поиске в несбалансированном бинарном дереве поиска

Вывод

В результате эксперимента было получено, что в лучшем случае количество сравнений для AVL-дерева и несбалансированного бинарного дерева поиска одинаково и равно 1 для любого рассмотренного количества узлов. В худшем случае количество сравнений в AVL-дерева более чем в 1000 раз меньше, чем в несбалансированном бинарном дереве поиска при количестве узлов большем или равном 16388.

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:

- были изучены алгоритмы сбалансированного и несбалансированного дерева поиска;
- были разработаны изученные алгоритмы;
- были выполнены замеры количества сравнений, необходимого для решения задачи поиска элементов в лучшем случае и в худшем случае для выполненных реализаций;
- был подготовлен отчет 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: 05.01.2024).