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 Фам Минь Хиеу

Группа ИУ7-52Б

Оценка (баллы)

Преподаватель Волкова Л. Л.

Москва — 2024 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Бинарное дерево или двоичное дерево — это дерево, в котором у каждого из его узлов не более двух дочерних узлов. При этом каждый дочерний узел тоже представляет собой бинарное дерево [1].

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

Для достижения поставленной цели необходимо выполнить следующие задачи:

— изучить алгоритм поиска в двоичном дереве поиска несбалансированном и сбалансированном;

— привести схемы рассматриваемых алгоритмов;

— создать программное обеспечение, реализующее перечисленные алгоритмы;

— провести замеры количества сравнений в лучшем случае и в худшем случае;

— описать и обосновать полученные результаты в отчете о выполненной лабораторной работе.

1 Аналитическая часть

В этом разделе будет представлено описание алгоритмов поиска целого числа в двоичном дереве поиска несбалансированном и сбалансированном.

1.1 Описание двоичного дерева поиска

Бинарные деревья поиска отличаются от обычных бинарных деревьев тем, что хранят данные в отсортированном виде. Хранение значений внутри бинарного дерева поиска организовано в следующем виде:

— все значения в узлах левого дочернего поддерева меньше значения родительского узла;

— все значения в узлах правого дочернего поддерева больше значения родительского узла;

— каждый дочерний узел тоже является бинарным деревом поиска.

Благодаря такой структуре хранения данных поиск узла в бинарном дереве поиска занимает O(log(N)). Это значительно меньше, если хранить значения в списках — O(N) [1].

1.2 Описание сбалансированного двоичного дерева

Дерево поиска называется сбалансированным, т.е. таким, в котором высота левого и правого поддеревьев отличаются не более чем на единицу [2].

Эта структура данных разработана советскими учеными Адельсон—Вельским Георгием Максимовичем и Ландисом Евгением Михайловичем в 1962 году. Аббревиатура АВЛ соответствует первым буквам фамилий этих ученых. Первоначально АВЛ-деревья были придуманы для организации перебора в шахматных программах. Советская шахматная программа «Каисса» стала первым официальным чемпионом мира в 1974 году [1].

Вывод

В данном разделе приведено описание двоичного дерева поиска и сбалансированного двоичного дерева.

2 Конструкторская часть

В данном разделе будут приведены схемы алгоритма поиск узла в двоичном дереве поиска и требования к программному обеспечению.

2.1 Разработка алгоритмов

На рис. 2.1 приведены схемы алгоритма поиск узла в двоичном дереве поиска.

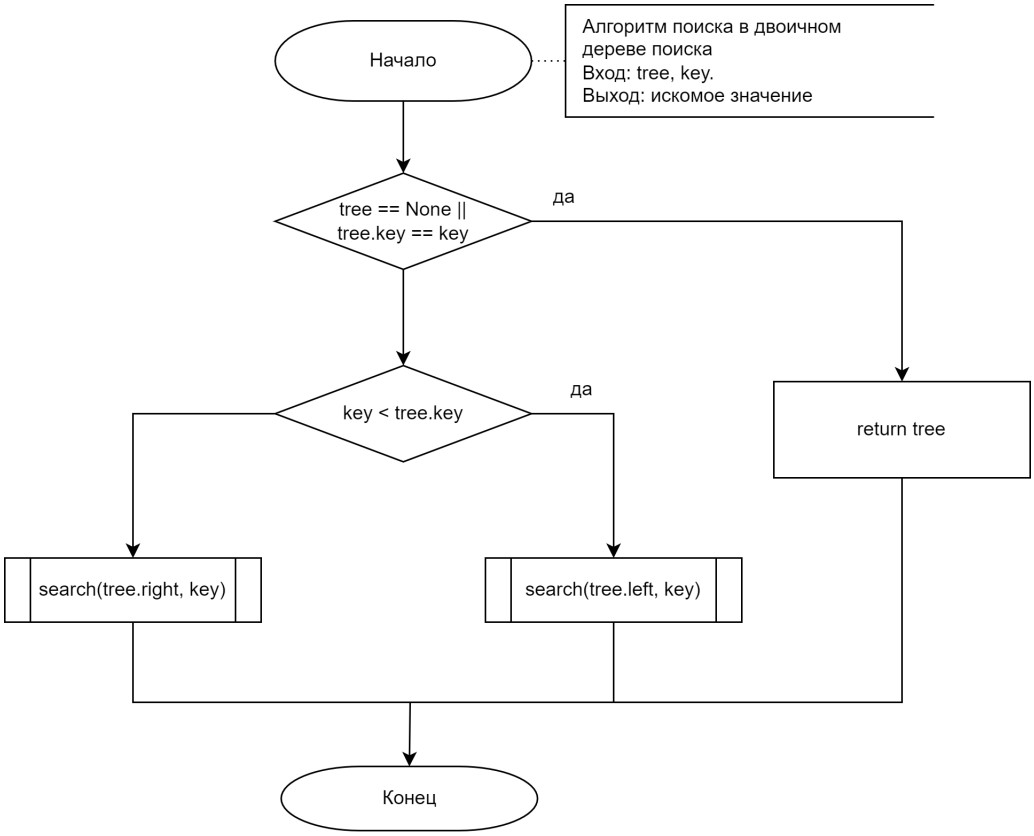

Рисунок 2.1 – Схема алгоритма поиск узла в двоичном дереве поиска

2.2 Требования к программному обеспечению

К программе предъявляются следующие требования:

— Программа должна предоставлять 2 режима работы: режим поиска целого числа в двоичном дереве поиска несбалансированном и сбалансированном;

— В начале работы программы пользователю нужно ввести целое число — это выбор пункта меню.

Вывод

В данном разделе приведены схемы алгоритма поиск узла в двоичном дереве поиска и требования к программному обеспечению.

3 Технологическая часть

В данном разделе будут приведены средства реализации, реализация алгоритма, а также функциональные тесты.

3.1 Средства реализации

В данной работе для реализации был выбран язык программирования *Python* [3], так как он предоставляет весь необходимый функционал для выполнения работы.

Визуализация графиков с помощью библиотеки *Matplolib* [4].

3.2 Реализация алгоритмов

В листингах 3.1 представлены функции для алгоритма поиска.

Листинг 3.1 – Функция алгоритма поиска

```
1 def search(self, root, key):
2     if root is None or root.key == key:
3         return root
4     if key < root.key:
5         return self.search(root.left, key)
6     else:
7         return self.search(root.right, key)
```

3.3 Функциональные тесты

В таблице 3.1 приведены функциональные тесты для двоичного дерева поиска и сбалансированного дерева. Все тесты пройдены успешно.

Таблица 3.1 – Функциональные тесты

N	Элементы	число	Ожидаемый результат
-1			Сообщение об ошибке
7	1 4 2 3 -1 0 -2	0	Узел в дерево
5	5 2 3 4 1	-2	None

Вывод

В данном разделе будут приведены средства реализации, реализация алгоритма, а также функциональные тесты.

4 Исследовательская часть

4.1 Технические характеристики устройства

Тестирование проводилось на устройстве со следующими техническими характеристиками:

- операционная система Window 10 Home Single Language;
- память 8 Гб;
- процессор 11th Gen Intel(R) Core(TM) i7-1165G7 2.80 ГГц, 4 ядра.

4.2 Демонстрация работы программы

На рисунках 4.1 – 4.2 приведены примеры работы программы поиска в двоичном дереве поиска несбалансированном и сбалансированном.

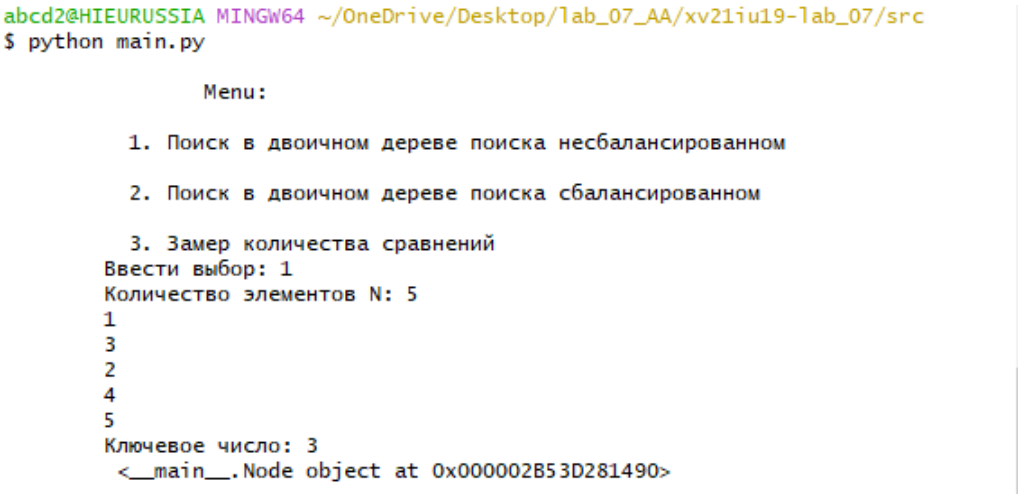

Рисунок 4.1 – Пример работы программы поиска в двоичном дереве поиска несбалансированном

```
abcd2@HIEURUSSIA MINGW64 ~/OneDrive/Desktop/lab_07_AA/xv21iu19-lab_07/src
$ python main.py

        Menu:

  1. Поиск в двоичном дереве поиска несбалансированном

  2. Поиск в двоичном дереве поиска сбалансированном

  3. Замер количества сравнений
Ввести выбор: 2
Количество элементов N: 5
1
3
2
4
5
Ключевое число: 6
 None
```

Рисунок 4.2 – Пример работы программы поиска в двоичном дереве поиска сбалансированном.

4.3 Сравнение количество сравнений

В ходе эксперимента было подсчитано количество сравнений, которые понадобились, чтобы найти каждый ключ в двоичном дереве поиска несбалансированном и сбалансированном, и на основе полученных данных составлены гистограммы.

Гистограммы для алгоритма поиска в двоичном дереве поиска несбалансированном в случае отсутствия элемента и случае нахождения элемента в листе на максимальной высоте дерева представлены на рисунке 4.3

Гистограммы для алгоритма поиска в двоичном дереве поиска сбалансированном в случае отсутствия элемента и случае нахождения элемента в листе на максимальной высоте дерева представлены на рисунке 4.4

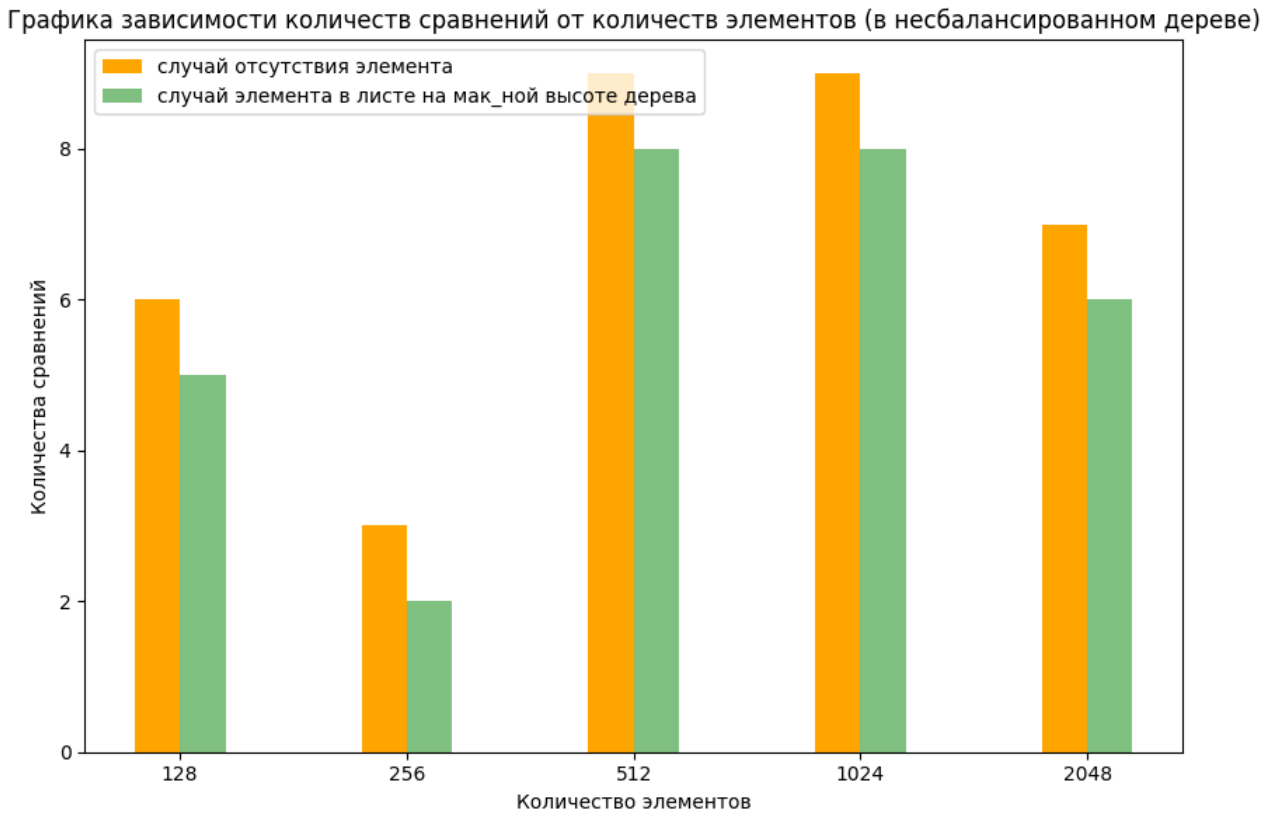

Рисунок 4.3 – Сравнение количеств сравнений алгоритмов поиска в двоичном дереве поиска несбалансированном

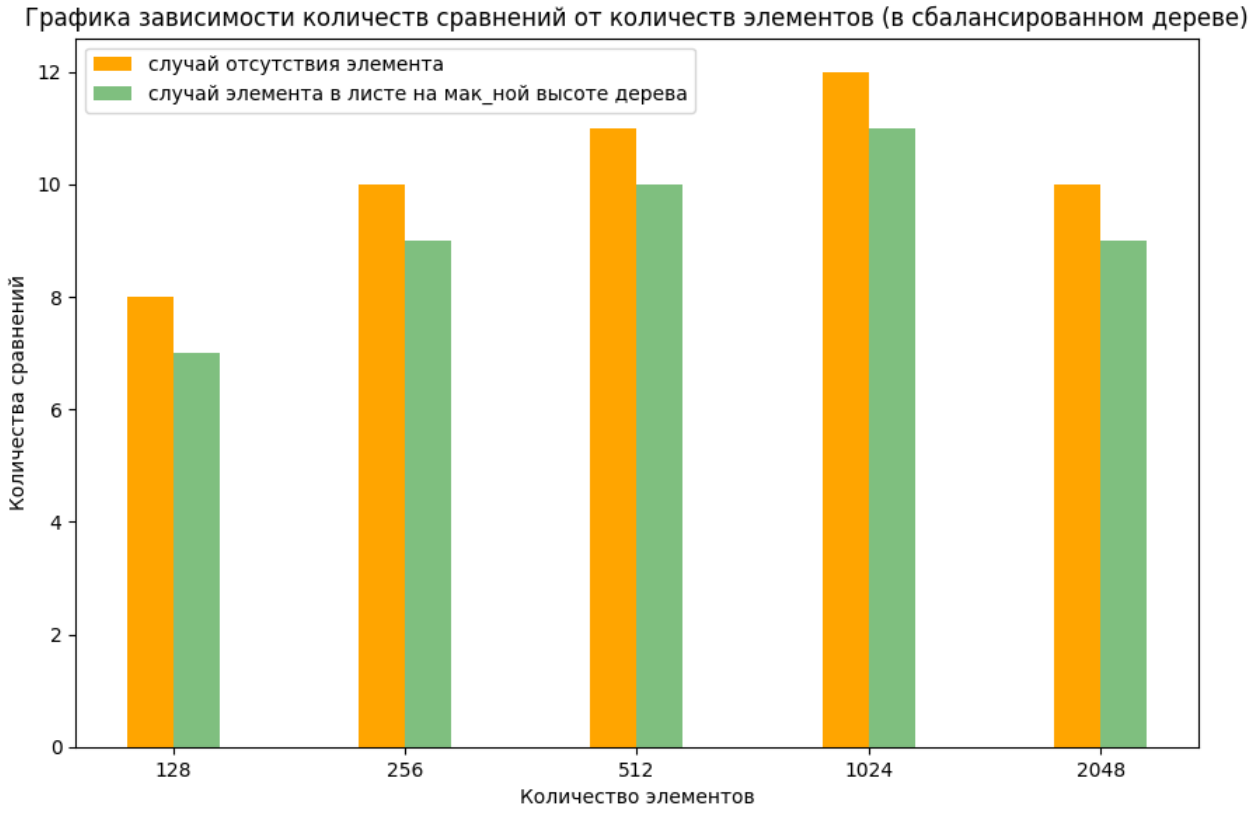

Рисунок 4.4 – Сравнение количеств сравнений алгоритмов поиска в двоичном дереве поиска сбалансированном

4.4 Вывод

В результате замеров количеств сравнений алгоритмов поиска в случае отсутствия элемента или случае нахождения элемента в листе на максимальной высоте дерева было установлено, что случай отсутствия является худшим.

ЗАКЛЮЧЕНИЕ

В результате исследований можно сделать вывод о том, что в случае отсуствия является худшим при поиске целого числа в двоичном дереве поиска несбалансированном и сбалансированном.

Поставленная цель была достигнута, в ходе выполнения данной лабораторной работы были решены следующие задачи:

— изучены алгоритм поиска в двоичном дереве поиска несбалансированном и сбалансированном;

— приведены схемы рассматриваемых алгоритмов;

— создано программное обеспечение, реализующее перечисленные алгоритмы;

— проведены замеры количества сравнений в лучшем случае и в худшем случае;

— описаны и обоснованы полученные результаты в отчете о выполненной лабораторной работе.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Бинарные деревья. Алгоритмы на деревьях [Электронный ресурс]. Режим доступа: https://ru.hexlet.io/courses/algorithms-trees/lessons/binary/theory_unit.

2. Сбалансированное бинарное дерево [Электронный ресурс]. Режим доступа: https://medium.com/@vitkarpov/cracking-the-coding-interview-4-2-9567d6986853

3. Welcome to Python [Электронный ресурс]. Режим доступа: https://www.python.org/

4. Matplotlib documentation [Электронный ресурс]. Режим доступа: https://matplotlib.org/stable/index.html