МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**ИНДИВИДУАЛЬНАЯ РАБОТА №1**

**Дисциплина: Анализ сложности алгоритмов**

**Тема: Деревья**

Работу выполнил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Фамилия И. О.

Направление подготовки: 02.03.02 Фундаментальная информатика и

информационные технологии

Направленность (профиль): Математическое и программное

обеспечение компьютерных технологий

Преподаватель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Фамилия И. О.

Краснодар

2024

**Задание:**

* Написать рекурсивный алгоритм;
* Найти функцию (рекурсивную и не рекурсивную) сложности алгоритма (программы);
* Произвести экспериментальные расчёты, подтверждающие правильность аналитических результатов.

|  |  |
| --- | --- |
| **Вариант** | **Задание** |
| 10 | Написать процедуру, вычисляющую разность между количеством вершин, имеющих только левых потомков и вершин, имеющих только правых потомков; |

*Таблица 1*

**Ход работы**

Составим рекурсивный алгоритм для решения данного задания.

Суть алгоритма будет заключаться в следующем:

1. Обход начинается с корня дерева;
2. Если у текущей вершины (на первом шаге – корня дерева) есть левый и правый потомки, то рекурсивно вызываем функцию от левого потомка и от правого и возвращаем их сумму;
3. Если у текущей вершины есть левый потомок и нет правого потомка, то рекурсивно вызываем функцию от левого потомка и возвращаем её значение + 1, т. к. была найдена вершина только с левым потомком;
4. Если у текущей вершины нет левого потомка и есть правый потомок, то рекурсивно вызываем функцию от правого потомка и возвращаем её значение -1, т. к. в этом случае была найдена вершина только с правым потомком;
5. Если у текущей вершины нет потомков, то возвращаем 0, это обеспечит выход из рекурсии для всех остальных вызовов.

Данный алгоритм в виде блок-схемы:

Изображение выглядит как текст, диаграмма, линия, оригами

Автоматически созданное описание

*Рис. 1.*

Реализация процедуры на языке Python:

def find\_diff(node):

    if node.left and node.right:

        return find\_diff(node.left) + find\_diff(node.right)

    if node.left and node.right is None:

        return 1 + find\_diff(node.left)

    if node.left is None and node.right:

        return find\_diff(node.right) - 1

    if node.left is None and node.right is None:

        return 0

Найдем функцию сложности данной программы (для полного дерева). Запишем рекуррентное уравнение:

Запишем некоторые значения :

Для данного уравнения трудно найти аналитическое решение, поэтому проведем линейную аппроксимацию методом наименьших квадратов. Получим функцию:

Для вырожденного дерева функция сложности будет иметь вид:

Для экспериментальной проверки определим полные деревья с 1, 3, 7 и 15-ю вершинами и вырожденные деревья с 1, 2, 3 и 4-я вершинами:

# деревья для экспериментальных расчетов

# полные деревья

full\_trees = [

    [1],

    [2, 1, 3],

    [4, 2, 1, 3, 6, 5, 7],

    [8, 4, 2, 1, 3, 6, 5, 7, 12, 10, 9, 11, 14, 13, 15]

]

# вырожденные деревья

degen\_trees = [

    [1],

    [1, 2, 3],

    [1, 2, 3, 4],

    [1, 2, 3, 4, 5]

]

Для оценки время работы программы будем использовать функцию:

# измерение времени работы

def get\_time(f, \*args):

    repeat = 10000000

    total = []

    for \_ in range(repeat):

        start\_time = time.time()

        f(args[0])

        end\_time = time.time()

        execution\_time = end\_time - start\_time

        total.append(execution\_time)

    average\_time = sum(total)/repeat

    return average\_time

Для полных и вырожденных деревьев получены следующие значения:

Полные деревья:

n = 1  T = 2.0287413597106934e-07

n = 3  T = 4.4240436553955077e-07

n = 7  T = 9.488101243972778e-07

n = 15  T = 1.949552059173584e-06

Вырожденные деревья:

n = 1  T = 2.059666395187378e-07

n = 2   T = 2.9630892276763916e-07

n = 3   T = 4.171457290649414e-07

n = 4   T = 5.313817024230957e-07

Проведем линейную аппроксимацию для найденных значений для полных деревьев и получим функцию:

или

В худшем случае функция близка к аналитическому результату.

Для вырожденных деревьев функция будет иметь вид:

В лучшем случае функция также близка к аналитическому результату.