МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Алгоритмы и структуры данных»

Тема: «Вычисление высоты дерева»

Студент гр. 1303 _____ Коренев Д.А.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург

2022

Цель работы.

Написать программу, которая вычисляет высоту корневого дерева, и тесты, проверяющие корректную работу программы в том числе граничные случаи.

Задание.

Вычисление высоты дерева.

На вход программе подается корневое дерево с вершинами {0,…, n−1}, заданное как последовательность parent0 , . . . , parentn−1 , где parenti — родитель i-й вершины. Требуется вычислить и вывести высоту этого дерева.

Формат входа.

Первая строка содержит натуральное число n. Вторая строка содержит n целых чисел parent0 , ... , parentn−1. Для каждого 0 ≤ i ≤ n − 1, parenti — родитель вершины i; если parent i = −1, то i является корнем. Гарантируется, что корень ровно один и что данная последовательность задаёт дерево.

Формат выхода.

Высота дерева.

Примечание: высотой дерева будем считать количество вершин в самом длинном пути от корня к листу.

Выполнение работы.

На вход программе вводится число n - количество узлов дерева, следующей строкой подаются данные — n целых чисел. Файл с кодом см. в Приложении А Исходный код программы.

Был создан класс Tree (см. Приложение Б Класс дерева) с приватными полями data — вторая строчка входных данных, и length — количество узлов дерева. Данный класс имеет метод getHeight, который вычисляет высоту дерева. Сначала проверяется корректность введенных данных и в случаи их некорректности возвращается значение или выбрасывается ошибка.

Далее создается словарь, в котором ключ — значение узла, значение — высота этого узла в дереве. Для каждого элемента из поля data применяется алгоритм вычисляющий его высоту. Чтобы оптимизировать программу используется проверка наличия этого элемента в словаре, если он там есть, то высота для него уже просчитана, это позволяет остановить алгоритм и перейти к следующему узлу. После того как алгоритм закончил свою работу, метод возвращает максимальное значение из словаря, которое является высотой дерева.

Тестирование.

Используется фрэймворк pytest для тестирования программы.

Файл tests.py (см. Приложение В Тестировочный файл).

Тесты рассматривают разные структуры деревьев. Проверяются граничные случаи с высотой деревьев 0 и 1, также тестируется случай, когда дерево представляет собой список: на каждой высоте по одному узлу, т.е. высота дерева равна количеству узлов. Проверяется случай собственного дерева.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	0 []	0	Граничное значение обработано корректно
2.	1 [-1]	1	Граничное значение обработано корректно
3.	5 [2 0 4 1 -1]	5	Тест обработан корректно
4	14 [5 10 1 1 7 8 3 5 10 8 -1 6 6 12]	6	Тест обработан корректно

Выводы.

В ходе работы был написан класс с методом вычисляющий высоту дерева. Для тестирования был рассмотрены граничные, нестандартные и стандартные случаи.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: src/main.py

```
from modules.Tree import Tree


def main():
    length = int(input())
    data = list(map(int, input().split(" ")))
    print(Tree(data, length).getHeight())
    return 0


if __name__ == "__main__":
    main()
```

ПРИЛОЖЕНИЕ В

КЛАСС ДЕРЕВА

Название файла: src/modules/Tree.py

```
class Tree:
    __data__ = list()
    __length__ = 0

    def __init__(self, data, length):
        self.__data__ = data
        self.__length__ = length

    def getHeight(self):
        if self.__length__ < 0:
            raise ValueError("Length value must be greater then
0")
        if self.__length__ == 0:
            return 0
        nodes = dict()

        for index in range(self.__length__):
            treeHeight = 1
            currentNode = self.__data__[index]
            while currentNode != -1:
                if currentNode in nodes:
                    treeHeight += nodes[currentNode]
                    break
                else:
                    currentNode = self.__data__[currentNode]
                    treeHeight += 1
            nodes[index] = treeHeight

        return max(nodes.values())
```

ПРИЛОЖЕНИЕ В

ТЕСТИРОВОЧНЫЙ ФАЙЛ

Название файла: src/tests.py

```
import pytest
from modules.Tree import Tree


def test_empty():
    length = 0
    data = []
    assert Tree(data, length).getHeight() == 0


def test_one():
    length = 1
    data = [-1]
    assert Tree(data, length).getHeight() == 1


def test_line():
    length = 5
    data = [2, 0, 4, 1, -1]
    assert Tree(data, length).getHeight() == 5


def test_custom():
    length = 14
    data = [5, 10, 1, 1, 7, 8, 3, 5, 10, 8, -1, 6, 6, 12]
    assert Tree(data, length).getHeight() == 6


def test_value_error():
    with pytest.raises(ValueError):
        length = -1
        data = []
        Tree(data, length).getHeight()
```