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 _____ Беззубов Д.В.

Преподаватель _____ Иванов Д. В.

Санкт-Петербург

2022

Цель работы.

Научиться итеративно обходить дерево, задаваемое списком родительских вершин. Основываясь на данном алгоритме, написать программу, вычисляющую высоту дерево, а так же написать тесты для данной программы.

Задание.

Вычисление высоты дерева.

На вход программе подается корневое дерево с вершинами $\{0, \ldots, n-1\}$, заданное как последовательность $parent_0, \ldots, parent_{n-1}$, где $parent_i$ – 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 $parent_0, \ldots, parent_{n-1}$. Для каждого $0 \le i \le n - 1$, $parent_i$ — 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.

Выполнение работы.

Для вычисления высоты дерева, заданного списком его родительских вершин были реализованы две функции: *check_tree(tree, n)* и *height(tree, n).*

Функция *check_tree()* проверяет входящие данные на корректность и выбрасывает ошибку в трех случаях:

1. если в переданном списке нет значения «-1», которое заявлено корнем любого передаваемого дерева
2. если значение *n* (кол-во узлов в дереве) отрицательно
3. если в списке вершин встречается элемент со значением большим или равным *n.*

Функция *height(tree, n)* вычисляет высоту дерева следующим образом:

Инициализируется пустой словарь, в котором будут храниться длины путей от вершин до корня дерева. Циклом перебираются все значения из списка, поданного на вход. Если в словаре существует запись с ключом, равным

текущему элементу при перечислении, значит, длина пути была вычислена ранее для данного элемента, в таком случае к текущей длине *cur_len* добавляется значение из словаря по ключу, иначе меняется текущий элемент *cur_elem* на следующий узел, увеличивается счетчик *cur_len.* При выходе из внутреннего цикла в словаре создается новая запись для очередного узла. В результате работы алгоритма, функция возвращает максимальное из значений, хранящихся в словаре.

Разработанный программный код см. в Приложении А.

Тестирование.

Для тестирования были написаны тесты, проверяющие функции на различных деревьях: линейном списке, пустом дереве, невырожденном дереве. Так же проверяются случаи с ошибочными данными.

Код файла с тестами содержится в приложении Б.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	0 []	0	test_empty_tree PASSED
2.	5 [-1, 0, 1, 2, 3]	5	test_list_tree PASSED
3.	5 [4, -1, 4, 1, 1]	3	test_common_tree PASSED
4.	-1 [0, 1, 2, 3]	ValueError	test_error_negative_len PASSED
5.	5 [-1, 0, 1, 2, 5]	ValueError	test_error_incorrect_data PASSED
6.	3 [0, 1, 2]	ValueError	test_error_no_root PASSED

Выводы.

В ходе лабораторной работы была написана программа вычисляющая высоту дерева, заданного списком родительских вершин. Так же все написанные

функции были покрыты тестами, проверяющими корректность работы алгоритмов при стандартных, пограничных и нестандартных случаях.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
from modules.tree import *


if __name__ == "__main__":
    n = int(input())
    data = list(map(int, input().split()))
    print(height(data, n))
```

Название файла: tree.py

```
def check_tree(tree, n):
    if n < 0:
        raise ValueError('invalid value for length of the tree')
    for elem in tree:
        if elem >= n:
            raise ValueError('invalid value for child')
    if -1 not in tree:
        raise ValueError('there is no root!')


def height(tree, n):
    if n == 0:
        return 0

    check_tree(tree, n)

    list_of_lens = {}
    for i, root in enumerate(tree):
        cur_len = 1
        cur_elem = root
        while cur_elem != -1:
            if cur_elem in list_of_lens:
                cur_len+=list_of_lens[cur_elem]
                break
            else:
                cur_elem = tree[cur_elem]
                cur_len+=1
        list_of_lens[i] = cur_len
    return max(list_of_lens.values())
```

ПРИЛОЖЕНИЕ Б

ФАЙЛ ТЕСТИРОВАНИЯ ПРОГРАММЫ

Название файла: test.py

```
from modules.tree import *
import pytest

def test_empty_tree():
    len = 0
    data = []
    assert height(data, len) == 0

def test_list_tree():
    len = 5
    data = [-1, 0, 1, 2, 3]
    assert height(data, len) == 5

def test_common_tree():
    len = 5
    data = [4, -1, 4, 1, 1]
    assert height(data, len) == 3

def test_error_negative_len():
    len = -1
    data = [0, 1, 2, 3]
    with pytest.raises(ValueError):
        height(data, len)

def test_error_incorrect_data():
    len = 5
    data = [-1, 0, 1, 2, 5]
    with pytest.raises(ValueError):
        height(data, len)

def test_error_no_root():
    len = 3
    data = [0, 1, 2]
    with pytest.raises(ValueError):
        height(data, len)
```