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. 1381 Возмитель В. Е

Преподаватель Шевская Н. В.

Санкт-Петербург

2022

Цель работы.

Создать алгоритм поиска высоты дерева

Задание.

Вычисление высоты дерева:

На вход программе подается корневое дерево с вершинами {0, . . . , n−1}, заданное как последовательность parent0 , . . . , parentn−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 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.

Для начала программа считывает данные, введенные пользователем, сохраняя их в переменные *N* и *values* (количество узлов и их значения)

Следом вызывается функция *max_height(),* которая считает высоту дерева. В качестве аргумента ей подаётся строка – список смежности. Далее строка преобразуется в список целых чисел через функции *map(), list()* и метода *split()*.

Создается словарь и после реализуется поиск высоты дерева.

Алгоритм работает перебором всех узлов в дереве. Начиная с первого узла программа последовательно просматривает каждый узел в списке *tree*. Временная высота узла увеличивается до тех пор, пока не дойдет до корневого узла, или в словаре с высотой «детей» не будет записано значение.

После, если в ключе не записано значение высоты, то идет запись, если максимальная высота меньше временной высоты, то временная высота становиться максимальной.

В итоге функция возвращает высоту дерева.

Тестирование.

Для проверки работы программы был разработан код тестовой программы.

Всего 3 теста:

- *test_1*. Данный тест был взят с условия лабораторной с сайта. Вход: массив, дерево которого имеет высоту 3.

- test_2. Тест для проверки особого случая. Вход: массив множества значений.

- *test_3*. Тест для проверки особого случая. Вход: массив из одного элемента (дерево, состоящее только из корня)

Код файла с тестами находится в приложении А.

Выводы.

Была изучена структура дерева и реализована функция по поиску его высоты.

ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: algos1.py

```
#python
def max_height(tree):
    tree = tuple(map(int, tree.split()))
    node_heights = dict()
    res = 0

    for node in tree:
        tmp_node = node
        tmp_h = 1
        while True:
            if tmp_node == -1:
                break

            tmp_node = tree[tmp_node]
            cached_node_height = node_heights.get(tmp_node, None)

            if cached_node_height is not None:
                tmp_h += cached_node_height
                break
            tmp_h += 1

        if node_heights.get(node, None) is None:
            node_heights[node] = tmp_h

        if tmp_h > res:
            res = tmp_h

    return res


N = input()
values = input()
print(max_height(values))
```

Название файла: test.py

```
import unittest
from algos1 import *


class TestMethods(unittest.TestCase):
    def test_1(self):
        self.assertEqual(max_height([4, -1, 4, 1, 1]), 3)

    def test_2(self):
         self.assertEqual(max_height([9, 7, 5, 5, 2, 9, 9, 9, 2, -
1]), 4)
```

```
    def test_3(self):
        self.assertEqual(max_height([-1]), 1)


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