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. 1384 _____ Усачева Д.В.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург2022

Цель работы.

Реализация алгоритма обхода графа для нахождения высоты дерева.

Задание.

На вход программе подается корневое дерево с вершинами {0, . . . , n−1}, заданное как последовательность $parent_0$, . . . , $parent_{n-1}$, 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 parent0 , ... , parentn−1. Для каждого $0 \le i \le n-1$, 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.

Была разработана функция height_t(n, tree)),принимающая ссылку на дерево и количество элементов.

В ней сначала был создан массив arr_hg размером n. Далее в цикле поданная вершина определяется как родитель, и от нее шел подсчет высоты до корня дерева. Результат вычислений добавляется в массив arr_hg. Так обрабатывается каждая вершина, и тогда максимальное значение из полученного массива возвращается в качестве результата функции (количество вершин в самом длинном пути от корня к листу).

Тестирование.

№	Входные данные	Выходные данные	Комментарии
1.	4 -1 4 1 1	3	Верно
2.	7 -1 0 0 0 3 4 2	4	Верно
3.	12 -1 0 1 2 3 4 4 4 4 6 6 6	7	Верно

4.	3 -1 0 2	2	Верно
5.	1 -1	1	Высота дерева из одной вершины

Выводы

В ходе работы была написана программа, использующая алгоритм обхода графа для вычисления высоты переданного дерева. Программа прошла все предложенные тесты.

ПРИЛОЖЕНИЕ 1
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
#python

def sum_ii(matrix):
    work = 0
    for i in range(len(matrix)):
        work += matrix[i][i]
    return work



def merge(arr):
    if len(arr) == 1:
        return
    middle = len(arr) // 2
    left, right = arr[:middle], arr[middle:]
    merge(left)
    merge(right)
    index_left = index_right = index = 0
    result = [0] * (len(left) + len(right))
    while index_left < len(left) and index_right < len(right):
        if left[index_left][0] <= right[index_right][0]:
            result[index] = left[index_left]
            index_left += 1
        else:
            result[index] = right[index_right]
            index_right += 1
        index += 1
    while index_left < len(left):
        result[index] = left[index_left]
        index_left += 1
        index += 1
    while index_right < len(right):
        result[index] = right[index_right]
        index_right += 1
        index += 1
```

```
    res = [str(k[1]) for k in result]
    print(" ".join(res))
    for i in range(len(arr)):
        arr[i] = result[i]
    return arr


matrix_res = []
count = int(input())
for i in range(count):
    matrix = []
    mi = int(input())
    for j in range(mi):
        line = list(map(int, input().split()))
        matrix.append(line)
    matrix_res.append((sum_ii(matrix), i))

result = merge(matrix_res)
res = [str(k[1]) for k in result]
print(" ".join(res))
```

Название файла: tests.py

```
import unittest
from main import *


class TestMethods(unittest.TestCase):
    def test_1(self):
        arr1 = [(32, 0), (11, 1), (3, 2)]
        self.assertEqual(merge(arr1),[(32, 0), (11, 1), (3, 2)])

    def test_2(self):
        arr2 = []
        self.assertEqual(merge(arr2), None)

    def test_3(self):
```

```
        arr3 = [(10, 0), (10, 1), (10, 2)]
        self.assertEqual(merge(arr3), [(10, 0), (10, 1), (10, 2)])

    def test_4(self):
        arr4 = [(10, 0), (10, 1), (10, 2), (11, 3)]
        self.assertEqual(merge(arr4), [(10, 0), (10, 1), (10, 2), (11,
3)])

    def test_5(self):
        arr5 = [(10, 0), (19, 1), (8, 2)]
        self.assertEqual(merge(arr5), [(10, 0), (9, 1), (8, 2)])

    def test_6(self):
        arr6 = [(10, 0), (11, 1), (14, 2), (12, 3), (13, 4)]
        self.assertEqual(merge(arr6), [(10, 0), (11, 1), (14, 2), (12,
3), (13, 4)])


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