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

ОТЧЕТ

по лабораторной работе №1
по дисциплине «Алгоритмы и структуры данных»
Тема: Вычисление высоты дерева

Студентка гр.1381 _____ Рымарь М.И.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург
2022

Цель работы.

Написать программу, вычисляющую высоту дерева.

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 $0 \le i \le n - 1$, 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.

Примеры

Вход:

5

4 -1 4 1 1

Выход:

3

Вход:

5

-1 0 4 0 3

Выход:

4

Выполнение работы.

Считывание данных происходит вне функции. В переменную num сохраняется целое число – число вершин в дереве. В переменную *nodeList* сохраняем массив вершин-родителей текущих вершин.

Чтобы вывести результат работы алгоритма, в функции print вызывается функция *treeHeight(nodeList, num).*

Для реализации алгоритма поиска высоты дерева была написана функция *treeHeight(nodeList, num)*. Для оптимизации алгоритма использовалась работа со словарём. В функции *treeHeight()* создаётся словарь *dictionary*, ключами которого являются вершины-родителями, значениями этих ключей является максимальная высота дерева для текущей вершины, которое находится при помощи цикла *while*. Результатом работы алгоритма является максимальное значение среди *dictionary.values()*.

Тестирование.

Тесты рассматривают несколько деревьев с разной высотой и количеством узлов. Также протестирован граничный случай (пустой массив и количество вершин, равное нулю). Результаты тестирования представлены на рисунке 1. Файл с тестированием *test.py* представлен в приложении А.

```
============================= test session starts ==============================
collecting ... collected 5 items

test.py::TestTreeHeight::test1 PASSED                                    [ 20%]
test.py::TestTreeHeight::test2 PASSED                                    [ 40%]
test.py::TestTreeHeight::test3 PASSED                                    [ 60%]
test.py::TestTreeHeight::test4 PASSED                                    [ 80%]
test.py::TestTreeHeight::test5 PASSED                                    [100%]

============================== 5 passed in 0.02s ===============================

Process finished with exit code 0
```

Рисунок 1 – Результаты тестирования

Выводы.

В ходе выполнения лабораторной работы была изучена такая структура данных, как дерево. Была написана программа, вычисляющая высоту дерева, которое задаётся вершинами-родителями текущих вершин.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММ

Название файла: *main.py*

```
def treeHeight(nodeList, n):
    dictionary = dict()
    if nodeList == [] or n <= 0:
        raise ValueError("Wrong input")
    for i in range(n):
        height = 1
        node = nodeList[i]
        while node != -1:
            if node not in dictionary:
                node = nodeList[node]
                height += 1
            else:
                height += dictionary[node]
                break
        dictionary[i] = height
    return max(dictionary.values())

if __name__ == '__main__':
    num = int(input())
    nodeList = list(map(int, input().split()))
    print(treeHeight(nodeList, num))
```

Название файла: *test.py*

```
from main import treeHeight
import unittest

class TestTreeHeight(unittest.TestCase):

    def test1(self):
        self.assertEqual(treeHeight([4, -1, 4, 1, 1], 5), 3)

    def test2(self):
        self.assertEqual(treeHeight([-1, 0, 4, 0, 3], 5), 4)

    def test3(self):
        self.assertEqual(treeHeight([2, 2, -1, 0, 2], 4), 3)

    def test4(self):
        self.assertEqual(treeHeight([1, 2, 3, -1, 3, 6, 3], 6), 4)

    def test5(self):
        self.assertRaises(ValueError, treeHeight, [], 0)
```