# 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 |  | Соломин Д.А. |
| Преподаватель |  | Берленко Н.В. |

Санкт-Петербург 2022

# Цель работы.

Вычислить высоту дерева, протестировать программу

# 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. Первой строкой добавьте #python или #c++, чтобы проверяющая система знала, каким языком вы пользуетесь.

# Протокол.

Программа состоит из функции count, которая подсчитывает высоту графа исходя из полученных данных.

Обработка данных. Для начала возьмем из ввода число узлов n, которое нам не понадобится. Теперь само дерево *data*. Все элементы которого разделим командой *split* и преобразуем к числу.

Выполнение алгоритма

Для выполнения программы преобразуем *data* к объекту формата

{node: […children] }

Где *node* текущий узел, а *children* массив из потомков этого узла

Теперь для того чтобы получить глубину итеративно (без рекурсий)

Возьмем *root* дерева и пробежимся по его потомкам. Посмотрим есть ли у его потомков дети и если да, то запишем их потомков то есть спустимся на 1 уровень вниз по дереву. Теперь корень указывает на всех потомков 2 уровня. Таким образом мы доходим до самого конца, когда потомков у root не остается

# Тестирование

Для теста впишем данные

5

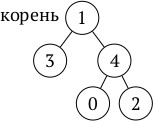
**4 -1 4 1 1**

Рисунок 1. Дерево 1.

Ожидаем ответ 3

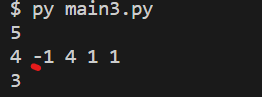


Рисунок 2. Ответы 1.

Проверим

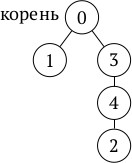
-1 0 4 0 3

Рисунок 3. Дерево 2.

Ожидаем 4

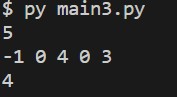


Рисунок 4. Ответы 2.

Разберем крайние случаи. К примеру не пришло ни 1 узла. В ответе получили пустую строку.

К примеру. Пришел просто список из 5 узлов

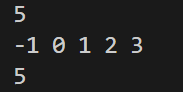


Рисунок 5. Ввод и вывод.

Как видим высота дерва посчиталась верно.

# Вывод:

В этом лаб работе мы научились работать с деревом, поняли как оно устроено. Научились находить его глубину. Написали алгоритм со сложностью n. И протестировали его.

**Приложение**

Исходный код программы

#python

def count(data):

roots = {node: [] for node in data}

i = 0

for parent in data:

roots[parent].append(i)

i+=1

curHead = roots[-1]

height = 1

while curHead:

newRoot = []

for childerNodes in curHead:

for next in roots[childerNodes]:

if next in roots.keys(): newRoot.append(next)

curHead = newRoot

height += 1

return height

if \_\_name\_\_ == "\_\_main\_\_":

n = int(input())

data = list(map(int, input().split()))

print(count(data))