**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.**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 1304 |  | Чернякова В.А. |
| Преподаватель |  | Глазунов С.А. |

Санкт-Петербург

2022

**Цель работы.**

Освоить алгоритм поиска высоты дерева на языке программирования Python. Реализовать проверку корректности создаваемого кода с помощью pytest.

**Задание.**

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.*

**Выполнение работы.**

На вход программе подается целое число *n*, равное количеству вершин дерева, и список родителей, введенный через пробел.

Функции.

Функция *def tree\_creator(data)* принимает на вход в качестве аргумента список, в котором хранятся *n* целых чисел, являющихся родителями вершин дерева. В теле функции создается список *tree* с помощью встроенной функции *list()*. Так как в обрабатываемом списке значения – строки, то с помощью *int()* онипреобразуются в целые числа для дальнейшей работы. Функция с помощью *return* возвращает созданный список.

Функция *def height(tree, n)* принимает на вход в качестве аргументов список с родителями вершин дерева и количество вершин. С помощью условного оператора *if* проверяется количество вершин. Если она одна, то функция с помощью *return* возвращается значение *n*, равное количеству вершин. Иначе работает блок *else*. Создается список *all\_height* длинной *n* и начальными значениями *0*, в нем будут хранится все возможные высоты дерева. С помощью цикла *for* перебираются все вершины от *0* до *n* не включительно. Переменной *current* присваивается значение *i*, вершины, на данном шаге цикла. Далее с помощью цикла *while* перебираются все вершины-родители, связанные с изначальной, пока не будет достигнута *-1*, то есть корень. Для вычисления высоты необходимо перейти к родителю вершины, поэтому переменной *parent* присваивается значение из списка *tree*, которое находится под индексом текущего значения *current*. В *current* теперь находится вершина-родитель. Значение в списке *all\_height* по индексу *i* увеличивается на *1* – значение высоты дерева относительно вершины, обрабатываемой на данной итерации цикла. Далее с помощью условного оператор *if* проверяется, есть ли уже значение высоты в списке *all\_height* по индексу *parent* – родителя, если да, то значение в списке *all\_height* по индексу *i* увеличивается на значение, проверяемое условием – полная высота. Функция с помощью *return* возвращает максимальное значение в списке высот - *max(all\_height).*

Разработанный программный код см. в приложении А.

**Тестирование.**

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | 1  -1 | 1 | Проверка крайнего случая – на вход подан только корень дерева. |
|  | 3  -1 0 0 | 2 | Проверка работы алгоритма для дерева из двух веток. |
|  | 5  3 4 3 4 -1 | 3 | Работа программы корректна. |
|  | 9  4 0 8 -1 3 7 2 1 5 | 9 | Проверка корректного работы алгоритма, когда у дерева только одна ветка. |
|  | 16  15 2 4 2 8 6 4 6 -1 10 12 10 8 14 12 14 | 5 | Корректная работа программы для произвольного большого дерева. |

**Выводы.**

Был изучен алгоритм нахождения высоты дерева. На основе данного алгоритма была создана программа. Освоена работа с pytest и написано тестирование для программного кода, проверяющее его корректность.

# Приложение А Исходный код программы

Название файла: main.py

def tree\_creator(data):

tree = list(int(parents) for parents in data)

return tree

def height(tree, n):

if n == 1:

return n

else:

all\_height = n\*[0]

for i in range(n):

current = i

while current != -1:

parent = tree[current]

current = parent

all\_height[i] += 1

if all\_height[parent]:

all\_height[i] += all\_height[parent]

break

return max(all\_height)

if \_\_name\_\_ == '\_\_main\_\_':

n = int(input())

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

print(height(tree\_creator(parents), n))

Название файла: test.py

from main import height

import pytest

@pytest.mark.parametrize("tree, n, expected\_result",

[([-1], 1, 1),

([-1, 0, 0], 3, 2),

([3, 4, 3, 4, -1], 5, 3),

([4, 0, 8, -1, 3, 7, 2, 1, 5], 9, 9),

([15, 2, 4, 2, 8, 6, 4, 6, -1, 10, 12, 10, 8, 14, 12, 14], 16, 5)])

def test(tree, n, expected\_result):

assert height(tree, n) == expected\_result