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. 1304 Шаврин А.П.

Преподаватель Глазунов С.А

Санкт-Петербург

2022

Цель работы.

Изучить основные методы вычисления высоты дерева с помощью языка программирования *Python*. Освоить проверку корректности кода с помощью системы *pytest*.

Задание.

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

Изначально была реализована функция *create_children_list,* принимающая на вход следующие аргументы:

- *patents_list* – список родителей
- *n* – количество вершин графа (узлов дерева)

Внутри функции создается список списков *children_list*, хранящий в себе всех детей, каждого родителя (индекс – родитель, значение – список детей). Список *children_list* заполняется с помощью цикла *for.* В значение с индексом *parents_list[child]* (родитель) будут занесены значения *i* (дети). В конце функция вернет список *children_list.*

Затем была реализована функция *breadth_first_search,* возвращающая высоту дерева, при помощи обхода в ширину с использованием очереди. Функция принимает два аргумента:

- *children_list* – список созданный функцией *create_children_list*
- *cur_level* – список узлов на текущем уровне (изначально корень)

В функции создается список queue, имитирующий очердь, и переменная *height,* хранящая в себе высоту дерева (изначально 0). Цикл *while* будет работать пока в переменной *cur_level* не окажется пустой список, что означает, что программа дошла до самого дальнего листа дерева. В цикле идет присваивание списку *queue* значения *cur_level*, а *cur_level* будет ссылаться на пустой список. Цикл *for* проходит по всем элементам очереди *queue*, и за каждую итерацию цикла в *cur_level* будет добавляться из дерева значение детей данного элемента и увеличиваться значение высоты *height* на единицу.

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.	5 4 -1 4 1 1	3	Результат верный
2.	5 -1 0 4 0 3	4	Результат верный
3.	10 2 2 -1 0 0 3 5 6 6 8	7	Результат верный
4.	11 4 10 3 4 -1 3 2 0 5 7 7	5	Результат верный
5.	1 -1	1	Результат верный
6.	0	0	Результат верный

Выводы.

Были изучены основные методы взаимодействия с древом на языке программирования Python. Изучено несколько способов нахождения высоты заданного древа. Реализована программа по нахождению высоты дерева с помощью структуры данных - очередь. Проведена проверка кода с помощью *pytest*.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла*: tree_on_queue.py*

```
def create_children_list(parents_list, n):
    children_list = [[] for i in range(n+1)]
    for child in range(n):
        children_list[parents_list[child]].append(child)
    return children_list

def breadth_first_search(children_list, cur_level):
    queue = []
    height = 0
    while cur_level:
        queue = cur_level
        cur_level = []
        for child in queue:
            cur_level.extend(children_list[child])
        height += 1
    return height

if __name__ == "__main__":
    n = int(input())
    parents_list = list(map(int, input().split()))
    children_list = create_children_list(parents_list, n)
    print(breadth_first_search(children_list, children_list[-1]))
```

Название файла*: pytests_tree_on_queue.py*

```
from tree_on_queue import create_children_list, breadth_first_search

def test1():
    children_list = create_children_list([4, -1, 4, 1, 1], 5)
    assert breadth_first_search(children_list, children_list[-1]) == 3

def test2():
    children_list = create_children_list([-1, 0, 4, 0, 3], 5)
    assert breadth_first_search(children_list, children_list[-1]) == 4

def test3():
    children_list = create_children_list([2, 2, -1, 0, 0, 3, 5, 6, 6, 8], 10)
    assert breadth_first_search(children_list, children_list[-1]) == 7

def test4():
    children_list = create_children_list([4, 10, 3, 4, -1, 3, 2, 0, 5, 7, 7], 11)
    assert breadth_first_search(children_list, children_list[-1]) == 5

def test5():
    children_list = create_children_list([-1], 1)
    assert breadth_first_search(children_list, children_list[-1]) == 1

def test6():
    children_list = create_children_list([], 0)
assert breadth_first_search(children_list, children_list[-1]) == 0
```