МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА № 42

ОТЧЁТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| старший преподаватель |  |  |  | С.Ю. Гуков |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЁТ О ЛАБОРАТОРНОЙ РАБОТЕ №2 |
| Высота дерева |
| по курсу: |
| АЛГОРИТМЫ И СТРУКТУРЫ ДАННЫХ |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ гр. № | 4329 |  |  |  | Д.С. Шаповалова |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2024

ОГЛАВЛЕНИЕ

[Цель работы 3](#_30j0zll)

[Постановка задачи 3](#_1fob9te)

[Схема алгоритма решения 4](#_3znysh7)

[Полное описание реализованной функции 4](#_gqc1nhfmq022)

[Листинг программы 5](#_2et92p0)

[Результат выполнения программы. 5](#_tyjcwt)

[ВЫВОДЫ 6](#_3dy6vkm)

# Цель работы

Изучить работу деревьев. Вычислить высоту данного дерева. Научиться хранить и эффективно обрабатывать деревья, даже если в них сотни тысяч вершин.

# Постановка задачи

Задание: Написать программу для построения дерева. В соответствии с параметрами варианта задания нужно реализовать сортировку слов в тексте и провести его простейший анализ. Вариант задания приведён в таблице 1.

Таблица 1. Индивидуальное задание

| № | Текст задания | Вход | Выход |
| --- | --- | --- | --- |
| 0 | **Реализовать** программу на любом языке, реализующую построение дерева, вывести его высоту.  **Вход** - Корневое дерево с вершинами {0, ..., n−1}, заданное как последовательность parent[0], ..., parent[n−1], где parent[i] – родитель i-й вершины.  **Гарантируется**, что корень ровно один. **Гарантируется**, что входная последовательность задаёт дерево.  **Ограничения**. 1 ≤ n ≤ 10^5 | Первая строка содержит натуральное число n. Вторая строка содержит n целых чисел parent[0], ..., parent[n−1]. Для каждого 0 ≤ i ≤ n−1, parent[i] – родитель вершины i; если parent[i] = −1, то i является корнем. | Высота дерева. |

# Схема алгоритма решения

1. Считываем данные.
2. Строим дерево.
3. Заполняем его детьми.
4. Вычисляем высоту дерева - ищем максимальную.

# Полное описание реализованной функции

Для начала теория:

В вводимой строке - вводятся *родители*. Их индексы - это их *дети*, то есть значения, которые должны быть в кружочке, а родители - это значения в кружочке, которые выше ребёнка. Если родитель “-1”, значит это кружочек на самом верху - то есть *корень*; его индекс это значение в этом кружочке.

Если значение ребёнка меньше родителя, то суём его в левую ветвь, если больше значения родителя, оно(его кружочек) записывается вправо, если есть значение больше, оно записывается ещё правее.

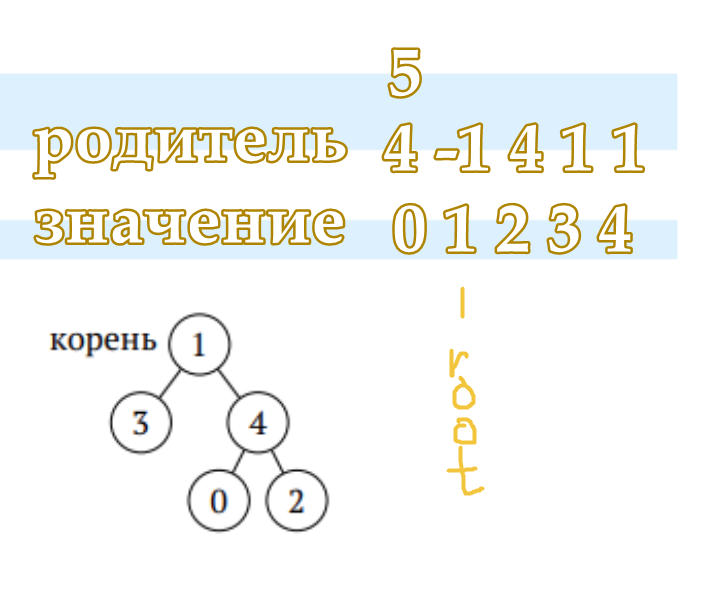


Рисунок 1.1. - Иллюстрация построения дерева.

1. Считываем входные данные.
2. Читаем количество узлов, строку с родителями.
3. Строку с родителями разделяем(split), размечаем(map) в массив(list), как целые числа.
4. Вызываем функцию, строящую дерево.
5. Вызываем функцию, вычисляющую высоту полученного дерева.
6. Выводим информацию в консоль.
7. Функция строительства дерева:
8. Инициализируем (tree) список для хранения детей каждой вершины.
9. Список из пустых списков, в дальнейшем тут будут храниться дети соответствующего узла; переменную для хранения индекса корня (root).
10. Заполняем дерево детьми для каждой вершины.
11. Перебираем пары: индекс - значение. Если находим элемент “-1”, то это корень, записываем его индекс. Иначе добавляем его в дерево - список подсписков узлов “родителей” - в подсписок с индексом родителя.
12. Возвращаем построенное дерево и корень.
13. Функция(рекурсивная) вычисления высоты дерева:

Принимает список детей(tree) и (node) узел, для которого мы хотим найти высоту.

1. База рекурсии

Если у узла нет детей - возвращаем 1. Так как высота дерева - это количество его уровней, то мы должны дойти до конца, до кроны дерева, где родители уже чайлдфри, и то считать за 1-вый уровень.

1. Заполняем массивы значениями высот.

Рекурсивно идём по разным ветвям, пока не дойдём до базы рекурсии, записываем значение высоты. Запускаем генератор списков, который вызывает эту же функцию для каждого ребенка узла node. Так имеем список высот всех детей.

1. Возвращаем максимальную высоту + 1.

# Листинг программы

def build\_tree(n, parents):

# Создаем список для хранения детей каждой вершины

tree = [[] for \_ in range(n)]

root = None

# Заполняем дерево детьми для каждой вершины

for child, parent in enumerate(parents):

if parent == -1:

root = child # Находим корень

else:

tree[parent].append(child)

return tree, root

def calculate\_height(tree, node):

if not tree[node]: # Если у узла нет детей

return 1

heights = [calculate\_height(tree, child) for child in tree[node]]

return 1 + max(heights) # Возвращаем высоту

n = int(input()) # Чтение количества узлов

parents = list(map(int, input().split())) # Чтение родителей

tree, root = build\_tree(n, parents) # Строим дерево

height = calculate\_height(tree, root) # Вычисляем высоту

print(height) # Печатаем высоту дерева

# 

# Результат выполнения программы.

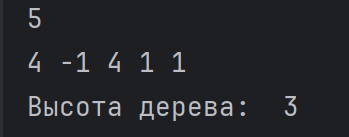


Рисунок 2.1. - 1-ый результат работы программы

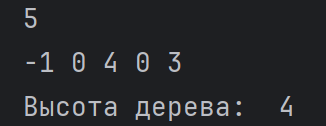


Рисунок 2.2. - 2-ой результат работы программы

# 

# Выводы

В ходе выполнения лабораторной работы мной были освоены и изучены: понятие дерево, высота дерева; алгоритм построения деревьев; алгоритм вычисления высоты дерева. Написанная программа была протестирована, полученный результат соответствует значению в примере.