Министерство цифрового развития, связи и массовых коммуникаций
Российской Федерации Сибирский Государственный Университет
Телекоммуникаций и Информатики СибГУТИ

Кафедра вычислительных систем

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3
по дисциплине «Моделирование»

Выполнил:
студент гр. ИВ-921
Ярошев Р. А.

Проверил:
Старший преподаватель
Петухова Я.В.

Новосибирск 2023

Оглавление

Формулировка задания

Реализовать программу, генерирующую случайное дерево по заданным ширине и глубине.

Теоретические сведения

Дерево - структура данны, эмулирующая древовидную структуру в виде набора связанных узлов.
Содержит:

- Корневой узел — самый верхний узел дерева (узел 8 на примере).
- Корень — одна из вершин, по желанию наблюдателя.
- Лист, листовой или терминальный узел — узел, не имеющий дочерних элементов.
- Внутренний узел — любой узел дерева, имеющий потомков, и таким образом, не являющийся листовым узлом.

Глубина дерева — количество уровней, на которых расположены его вершины.

Ширина дерева — максимальное количество узлов из всех на уровне.

Ход работы

В результате выполнения программы формируется pdf — файл с изображением дерева.

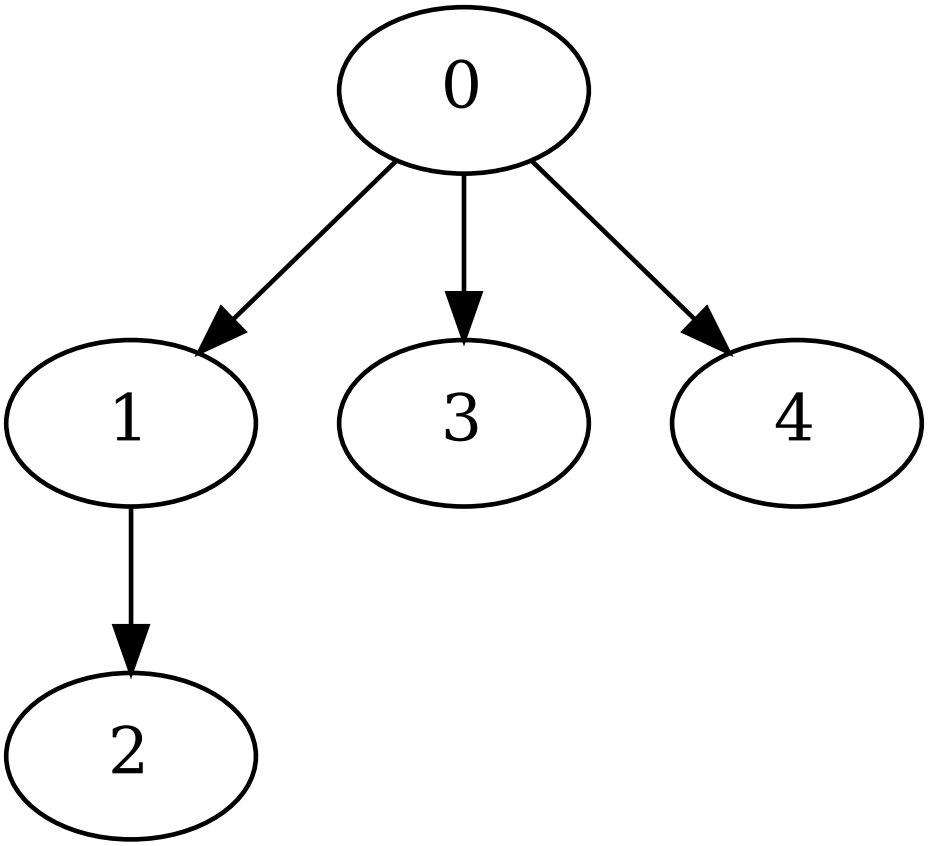

Рисунок 1. Ширина — 3. Глубина — 2.

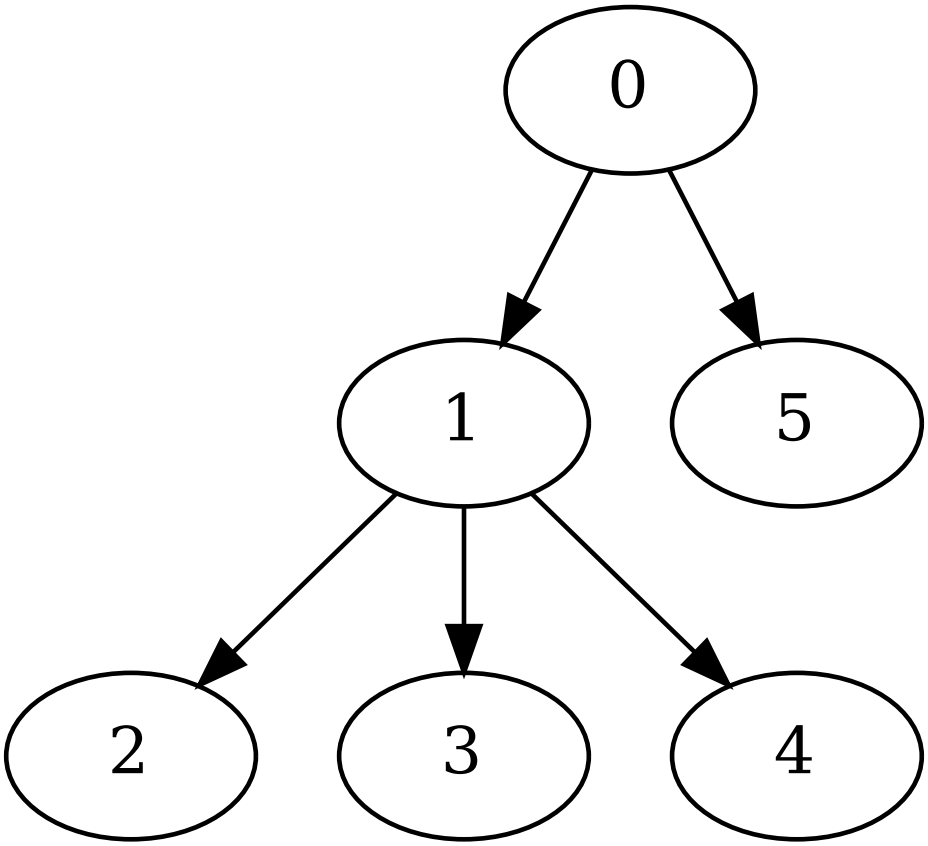

Рисунок 2. Ширина — 3. Глубина — 2.

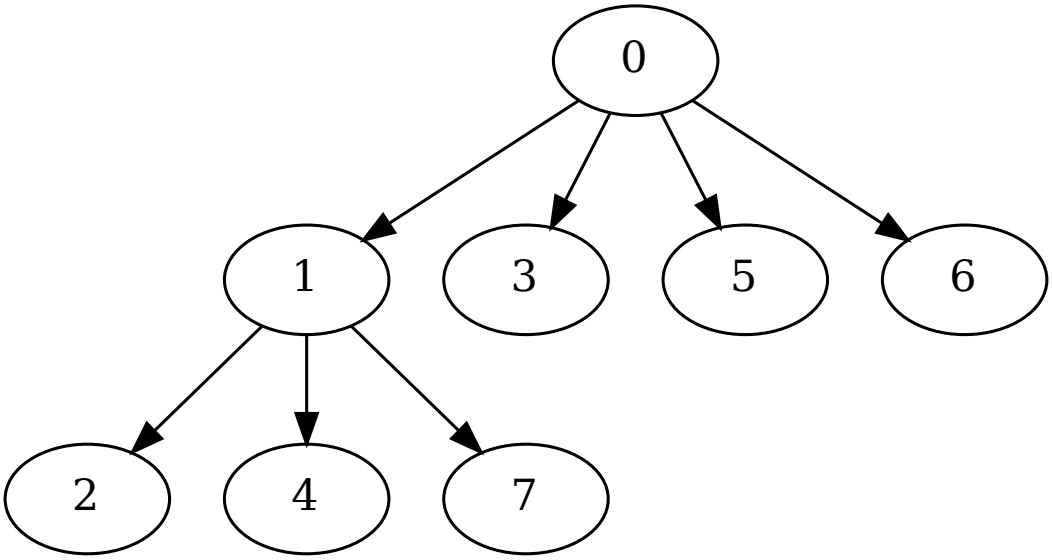

Рисунок 3. Ширина — 3. Глубина — 2.

Вывод

Реализована генерация случайных деревьев по заданным ширине и высоте.

Проведено большое количество опытов, из которых ясно, что вершины распределяются независимо от предыдущих результатов и не зависит от выбора конкретных параметров. Таким образом, вероятность получения любой конкретной формы равномерно распределена.

Листинг

```
import graphviz
from collections import deque
from math import ceil
import random

class Node:
    def __init__(self, value):
        self.value = value
        self.children = []

    def add_child(self, child):
        self.children.append(child)

    def to_graphviz(self):
        graph = graphviz.Digraph()
        self._add_nodes(graph)
        return graph

    def _add_nodes(self, graph):
        graph.node(str(id(self)), str(self.value))
        for child in self.children:
            graph.edge(str(id(self)), str(id(child)))
            child._add_nodes(graph)

def build_tree(depth, width):
    width -= 1
    if depth <= 0 or width <= 0:
        raise ValueError("Depth and width must be positive integers")
    root = Node(0)
    current_depth = 0
    current_width = [[] for _ in range(depth)]
    counter = 1
    parent = root
    while current_depth < depth:
        node = Node(counter)
        parent.add_child(node)
        counter += 1
        parent = node
        current_depth += 1
    flag = False
    while not flag:
        for i in current_width:
            if len(i) == width:
                flag = True
                break
            r = random.random()
            idx_a = (1 / (depth))
            idx = 0
            while idx_a < r:
                idx += 1
                r -= idx_a
            current_width[idx].append(counter)
            parent = root
            for _ in range(idx):
```

```
                parent = parent.children[0]
            parent.add_child(Node(counter))
            print(idx)
            counter += 1
    print(current_width)
    return root
depth = 3
width = 5
root = build_tree(depth, width)
graph = root.to_graphviz()
graph.render('tree')
```