**Санкт-Петербургский государственный УНИВЕРСИТЕТ**

**Факультет прикладной математики и процессов управления**

**отчет**

**по лабораторной работе №4**

**по дисциплине «Алгоритмы и структуры данных»**

**на тему «Решение задачи обхода дерева»**

|  |  |  |
| --- | --- | --- |
| Студент гр. 22Б16 |  | Шувалов Ф.В. |
| Преподаватель |  | Дик А.Г. |

**Санкт-Петербург**

**2023 г.**

**Оглавление**

[1. Цель работы 3](#_Toc163902451)

[3. Введение 3](#_Toc163902452)

[4. Схема выполнения алгоритма 3](#_Toc163902453)

[6. Представление программы на языке программирования 5](#_Toc163902454)

[7. Раскрытие смысла работы алгоритма 5](#_Toc163902455)

[8. Контрольный пример 5](#_Toc163902456)

[9. Вывод 7](#_Toc163902457)

[10. Литература 7](#_Toc163902458)

# **Цель работы**

Цель работы – написание алгоритма для обхода структуры данных типа “дерево”.

1. **Задачи**

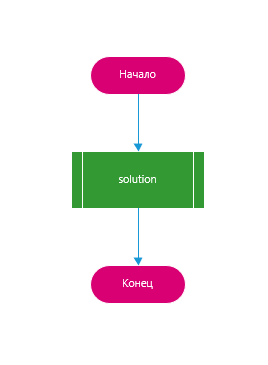
Разработать алгоритм обхода заданных вершин дерева и реализовать его визуализацию.

# **Введение**

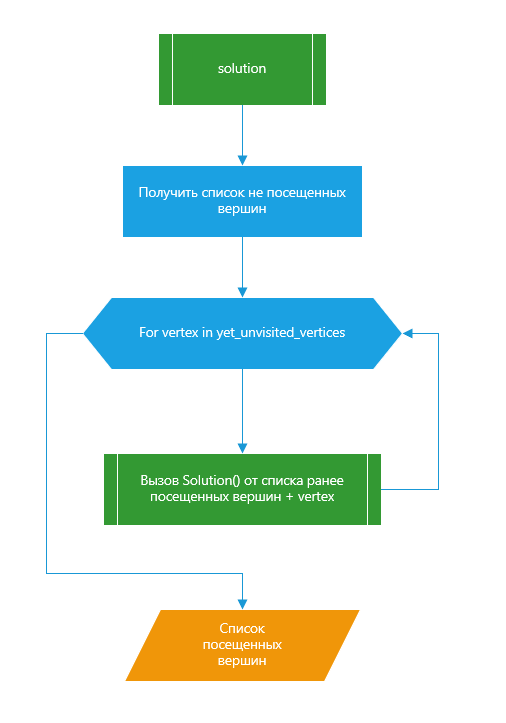
“Ты - умелый архитектор в мире гномов, по указу короля нужно расширить свой Казак Дур, добавив в него новые подземные помещения с уникальными рунными номерами. План расширения подобен лабиринту мыслей, где каждая новая комната должна быть связана с уже существующими. Только после завершения строительства каждая комната должна быть доступна из центральной палаты. Задача заключается в том, чтобы найти скрытый порядок, который будет вести к гармоничному увеличению Казак Дур.”

# **Схема выполнения алгоритма**

На рисунках 4.1-4.2 представлена блок-схема алгоритма.



*Рисунок 4.1 Блок-схема основного алгоритма*



*Рисунок 4.2 Блок-схема функции solution*

1. **Формализация задачи**

Требуется найти все возможные пути обхода дерева из вершины 0.

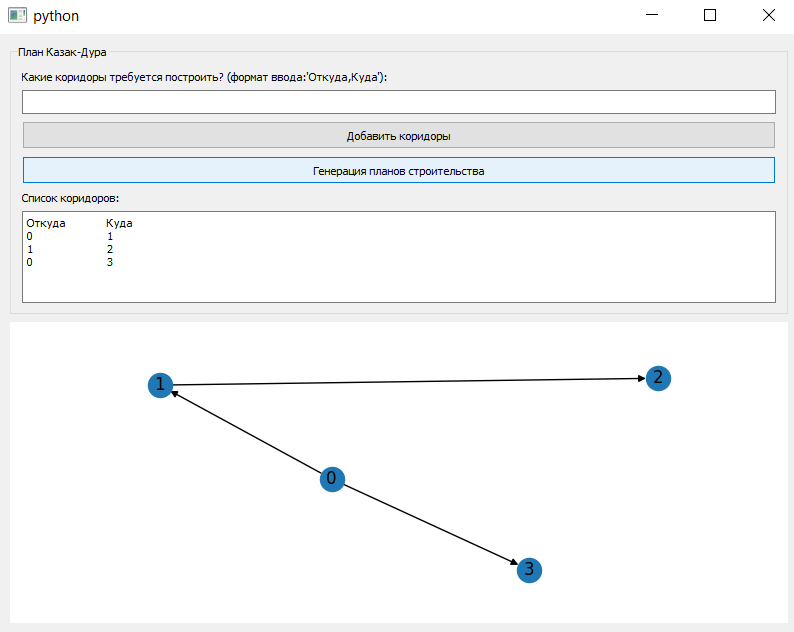
# **Представление программы на языке программирования**

Алгоритм обхода дерева реализован на языке Python3 в виде рекурсивной функции solution.

# **Раскрытие смысла работы алгоритма**

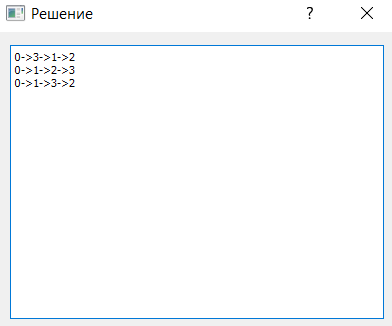
Алгоритм представляет собой рекурсивную функцию, которая при каждом новом вызове увеличивает список пройденных вершин, добавляя последнюю посещенную вершину в конец. Условие выхода из рекурсии – в списке пройденных вершин есть все вершины дерева. Полученный список пройденных вершин будет одним из вариантов обхода дерева.

# **Контрольный пример**



*Рисунок 10.1 Визуализация программы*

На рисунке 10.1 изображен пользовательский интерфейс программы. В левой части окна пользователь может вводить параметры, такие как: *дерево***.**

****

*Рисунок 10.2 результат работы алгоритма*

На рисунке 10.2 изображен результат работы алгоритма.

# **Вывод**

При выполнении данной работы были получены все необходимые навыки и знания по реализации алгоритма обхода дерева. В ходе реализации задачи была создана программа для визуализации его работы.

# **Литература**

* + 1. Документация библиотеки PyQT [Электронный ресурс] - URL: <https://doc.qt.io/qtforpython-6/> (дата обращения: 25.09.2023)

1. **Листинг**

import networkx as nx  
  
  
def solution(graph: nx.DiGraph, visited\_vertices: list) -> list:  
 out = []  
 vertices\_to\_visit = []  
 for vertex in visited\_vertices:  
 vertices\_to\_visit.extend(graph.successors(vertex))  
 vertices\_to\_visit = [v for v in set(vertices\_to\_visit) if v not in visited\_vertices]  
  
 if len(vertices\_to\_visit) == 0:  
 return ["->".join(list(map(str, visited\_vertices)))]  
  
 for vertex in vertices\_to\_visit:  
 out += solution(graph, visited\_vertices + [vertex])  
 return out