Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №6

По дисциплине: «Языковые процессы интеллектуальных систем»

# Тема: «Метод поиска в глубину на графе, рекурсивная и не рекурсивная реализации алгоритма поиска в глубину»

Выполнил:

Студент 4 курса

Группы ИИ-21

Литвинюк Т. В.

Проверил:

Монтик Н.С.

Брест 2024

**Цель работы**: сформировать представление о методе и алгоритме поиска в глубину на графе и его применении при решении задач.

6. Решите задачу 4 для представления графа списком смежных вершин. Для представления графа используйте динамические структуры данных, массив указателей на связанный список.

class Node:

    def \_\_init\_\_(self, vertex):

        self.vertex = vertex

        self.next = None

class Graph:

    def \_\_init\_\_(self, num\_vertices):

        self.num\_vertices = num\_vertices

        self.adj\_list = [None] \* num\_vertices

    def add\_edge(self, src, dest):

        node = Node(dest)

        node.next = self.adj\_list[src]

        self.adj\_list[src] = node

        node = Node(src)

        node.next = self.adj\_list[dest]

        self.adj\_list[dest] = node

    def print\_graph(self):

        for i in range(self.num\_vertices):

            print(f"Вершина {i}:", end="")

            temp = self.adj\_list[i]

            while temp:

                print(f" -> {temp.vertex}", end="")

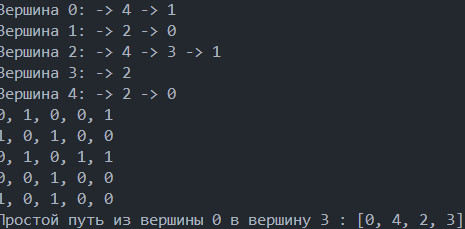
                temp = temp.next

            print("")

    def show(self):

        matrix = [[0] \* self.num\_vertices for \_ in range(self.num\_vertices)]

        for i in range(self.num\_vertices):

            temp = self.adj\_list[i]

            while temp:

                matrix[i][temp.vertex] = 1

                temp = temp.next

        for row in matrix:

            print(', '.join(map(str, row)))

def dfs(graph, v, w, visited, path):

    path.append(v)

    visited[v] = True

    if v == w:

        return True

    temp = graph.adj\_list[v]

    while temp:

        if not visited[temp.vertex]:

            if dfs(graph, temp.vertex, w, visited, path):

                return True

        temp = temp.next

    path.pop()

    return False

def find\_path(graph, v, w):

    visited = [False] \* graph.num\_vertices

    path = []

    if dfs(graph, v, w, visited, path):

        print("Простой путь из вершины", v, "в вершину", w, ":", path)

    else:

        print("Путь из вершины", v, "в вершину", w, "не существует")

g = Graph(5)

g.add\_edge(0, 1)

g.add\_edge(0, 4)

g.add\_edge(1, 2)

g.add\_edge(2, 3)

g.add\_edge(2, 4)

g.print\_graph()

g.show()

v = 0  # Начальная вершина

w = 3  # Целевая вершина

find\_path(g, v, w)

**Вывод:** в ходе выполнения лабораторной работы изучил метод и алгоритм поиска в глубину на графе.