Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Рязанский государственный радиотехнический университет имени В. Ф. Уткина»

Кафедра «САПР ВС»

Отчет по лабораторной работе №2

по дисциплине «Дискретная математика»

на тему

**«ПОИСК КРАТЧАЙШИХ ПУТЕЙ НА ГРАФАХ»**

Выполнил: студенты гр.1413  
Бригада №4  
Аверкин М.Е.  
Стаховский А.В.

Проверили:   
доц. Орешков В.И.  
доц. Бакулева М.А.

**Цель работы**: целью работы является изучение алгоритмов поиска кратчайших путей на графах на примере метода динамического программирования.

**Граф:**

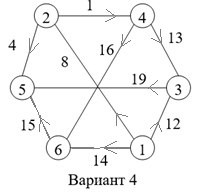
****

Рисунок 1 ‑ вариант задания

**Порядок выполнения работы:**

1. Получить задание у преподавателя в виде исходного ориентированного графа.

2. Составить блок-схему программы, определяющей кратчайший путь на графе от заданной начальной вершины *s* до заданной конечной вершины *t* с помощью метода динамического программирования.

3. Составить блок-схему программы, реализующей алгоритм топологической сортировки с произвольной нумерацией вершин графа.

4. Создать программу, реализующую метод динамического программирования и алгоритм топологической сортировки вершин. Исходный граф задается в виде матрицы смежности, вводимой построчно с помощью консоли. Указание: для определения вершин, входящих в множество используйте *j-*йстолбец матрицы смежности.

**Блок схема:**

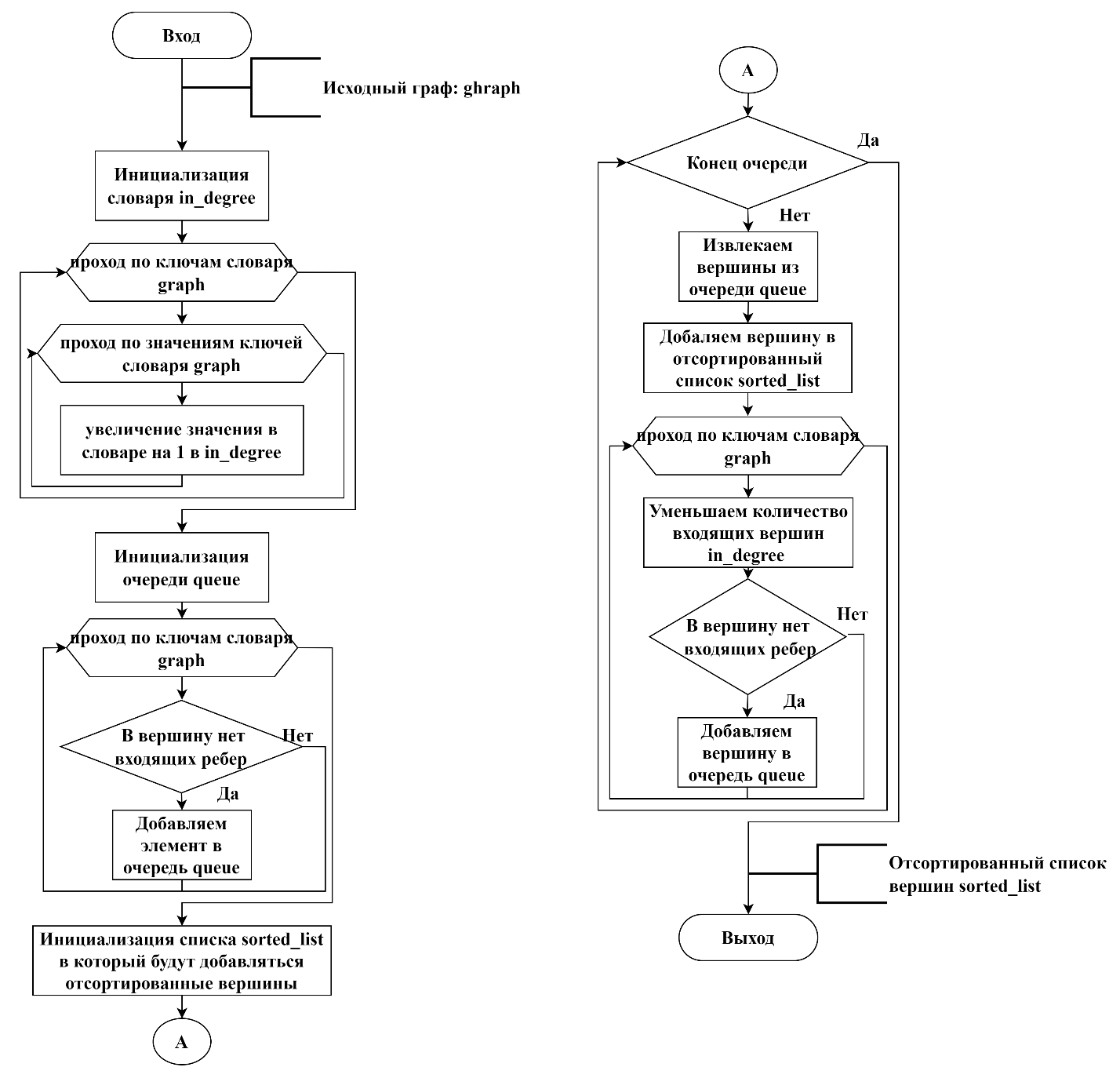


Рисунок 2 - Алгоритм топологической сортировки графа с произвольной нумерацией вершин графа topological\_sort



Рисунок 3 ‑ Алгоритм нахождения кратчайшего пути shortest\_path

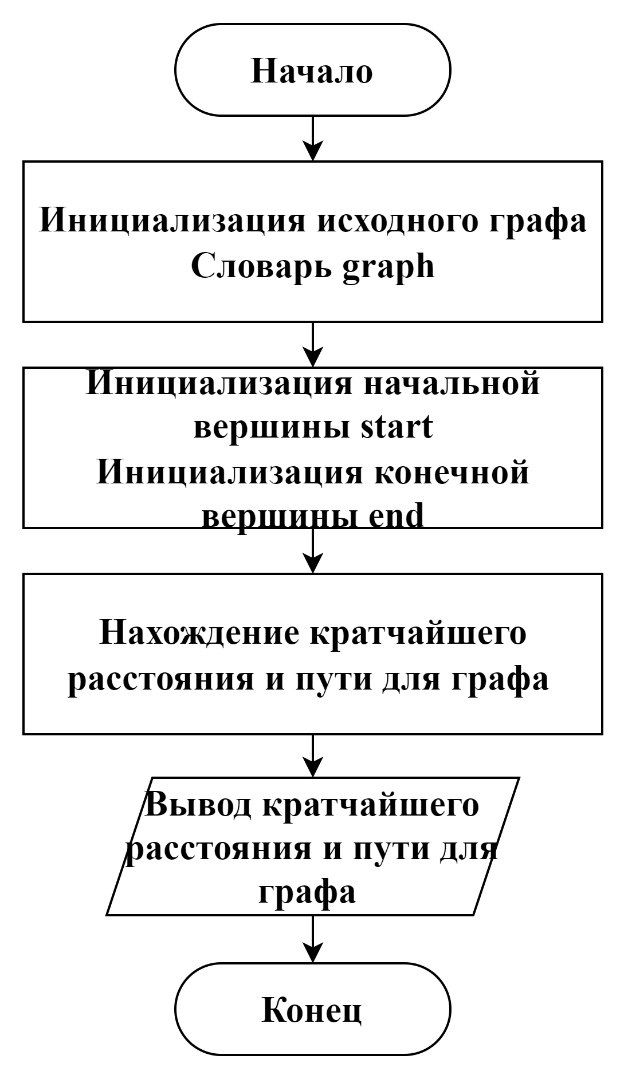
****

Рисунок 4 ‑ Реализация метода динамического программирования и алгоритма топологической сортировки вершин

**Листинг программы:**

import sys

from collections import deque

def topological\_sort(graph):

in\_degree = {u: 0 for u in graph}

for u in graph:

for v in graph[u]:

in\_degree[v] += 1

queue = deque()

for u in graph:

if in\_degree[u] == 0:

queue.append(u)

sorted\_list = []

while queue:

u = queue.popleft()

sorted\_list.append(u)

for v in graph[u]:

in\_degree[v] -= 1

if in\_degree[v] == 0:

queue.append(v)

return sorted\_list

def shortest\_path(graph, start, end):

top\_sorted = topological\_sort(graph)

distances = {v: sys.maxsize for v in graph}

paths = {v: [] for v in graph}

distances[start] = 0

for u in top\_sorted:

for v in graph[u]:

new\_dist = distances[u] + graph[u][v]

if new\_dist < distances[v]:

distances[v] = new\_dist

paths[v] = paths[u] + [u]

return distances[end], paths[end] + [end]

graph = {

'x\_1': {'x\_2': 8, 'x\_3': 12, 'x\_6': 14},

'x\_2': {'x\_4': 1, 'x\_5': 4},

'x\_3': {'x\_5': 19},

'x\_4': {'x\_6': 16, 'x\_3': 13},

'x\_5': {},

'x\_6': {'x\_5': 15}

}

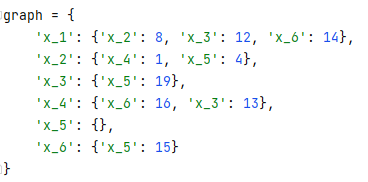
start = 'x\_1'

end = 'x\_5'

distance, path = shortest\_path(graph, start, end)

print(f"Кратчайший путь от вершины {start} до вершины {end}: {path} (расстояние: {distance})")

**Результаты работы программы:**



**ЗАКЛЮЧЕНИЕ**

Таким образом, в этой лабораторной работе мы составили программу, которая использует методы динамического программирования и алгоритм топологической сортировки вершин, что позволяет ей справилась с поставленной задачей вычисления кратчайшего пути.