**Дисциплина: Методы оптимизации**

**Раздел: Потоки в сетях**

**Автор: Филиппова Анна Сергеевна, кафедра ПИ, БГПУ им.М.Акмуллы Обязательно к изучению по направлению 09.03.03 Прикладная информатика (уровень бакалавриат)**

**Студент: Нурализода Исломиддин**

**Задания на лабораторную работу №1**

**Программный модуль для задачи о потоках в сети**

Цель работы: получит навыки разработки программных модулей для решения задач о потоках в сетях.

**ПОТОКИ В СЕТЯХ**

Дана сеть, для которой выполняются условия:

дуги ориентированы;

два специальных узла: источник (исток) V0 и слив (сток) Vn

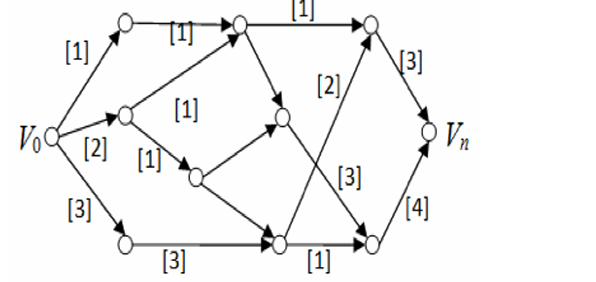
V0 – узел, у которого инвалентность равна нулю,

а Vn – узел, у которого аутвалентность равна нулю;

Для каждой дуги eij сети известны пропускные способности

bij, bij >0 при всех i, j

Пример на рисунке:



Кроме того, для некоторых постановок задач задается стоимость cij передачи единицы потока через каждую дугу eij

**ЗАДАНИЕ**

Варианты заданий в зависимости от последней цифры номера зачетки студента группы, см. табл 1.

**Таблица 1. Номер варианта для студентов группы**

**ПИНФ\_ПРЦ\_31-22**

**№ варианта Последняя цифра зачетки студента группы**

**А 1**

Вариант А.

Дано: Сеть, заданная матрицей ориентированного графа, элементы матрицы представляют собой пропускные способности bij. Требуется: разработать программу, которая: 1) отвечает на вопрос выполняются ли для сети условия (1) – (4); 2) находит три различных направленных пути (с наибольшим возможным значением потока для пути) от источника к стоку, обладающего тем свойством, что для каждой дуги пути, поток, проходящий через дугу, оказывается строго меньшим емкости дуги; 3) указывает из трех выбранных путей (см.п. 2) ), тот который несет больший поток от источника в сток.

Ответ на задание: Программу написано на язык python

import numpy as np

from collections import defaultdict

class FlowNetwork:

    def \_\_init\_\_(self, graph):

        self.graph = np.array(graph)

        self.num\_nodes = len(graph)

        self.source = 0

        self.sink = self.num\_nodes - 1

    def check\_conditions(self):

        # Условия (1) - (4) должны быть определены в задании

        # Здесь предполагается, что условия уже известны

        pass

    def bfs(self, residual\_graph, parent):

        visited = [False] \* self.num\_nodes

        queue = []

        queue.append(self.source)

        visited[self.source] = True

        while queue:

            u = queue.pop(0)

            for ind, val in enumerate(residual\_graph[u]):

                if not visited[ind] and val > 0:

                    queue.append(ind)

                    visited[ind] = True

                    parent[ind] = u

                    if ind == self.sink:

                        return True

        return False

    def find\_paths(self):

        residual\_graph = self.graph.copy()

        parent = [-1] \* self.num\_nodes

        paths = []

        while len(paths) < 3 and self.bfs(residual\_graph, parent):

            path\_flow = float("Inf")

            s = self.sink

            path = []

            while s != self.source:

                path.append(s)

                path\_flow = min(path\_flow, residual\_graph[parent[s]][s])

                s = parent[s]

            path.append(self.source)

            path.reverse()

            paths.append((path, path\_flow))

            for i in range(len(path) - 1):

                u = path[i]

                v = path[i + 1]

                residual\_graph[u][v] -= path\_flow

                residual\_graph[v][u] += path\_flow

        return paths

    def find\_max\_flow\_path(self, paths):

        max\_flow = -1

        max\_path = None

        for path, flow in paths:

            if flow > max\_flow:

                max\_flow = flow

                max\_path = path

        return max\_path, max\_flow

# Пример использования

graph = [

    [0, 16, 13, 0, 0, 0],

    [0, 0, 10, 12, 0, 0],

    [0, 4, 0, 0, 14, 0],

    [0, 0, 9, 0, 0, 20],

    [0, 0, 0, 7, 0, 4],

    [0, 0, 0, 0, 0, 0]

]

network = FlowNetwork(graph)

paths = network.find\_paths()

max\_path, max\_flow = network.find\_max\_flow\_path(paths)

print("Найденные пути:")

for path, flow in paths:

    print(f"Путь: {path}, Поток: {flow}")

print(f"\nПуть с наибольшим потоком: {max\_path}, Поток: {max\_flow}")

Ответ :

