**Міністерство освіти і науки України**

**Національний університет «Запорізька Політехніка»**

Кафедра програмних засобів

**ЗВІТ**

з лабораторної роботи №4

з дисципліни «Комп’ютерна дискретна математика»

На тему «ПЛОСКІ І ПЛАНАРНІ ГРАФИ. ПОТОКИ В МЕРЕЖАХ»

Варіант №20

**Виконав:**

Студент групи КНТ-122 О. А. Онищенко

**Прийняли:**

Ст. Викладач О. А. Щербина

2023

[Мета роботи 3](#_Toc134948639)

[Текст завдання №1 3](#_Toc134948640)

[Розв’язок завдання №1 3](#_Toc134948641)

[Текст завдання №2 3](#_Toc134948642)

[Розв’язок завдання №2 3](#_Toc134948643)

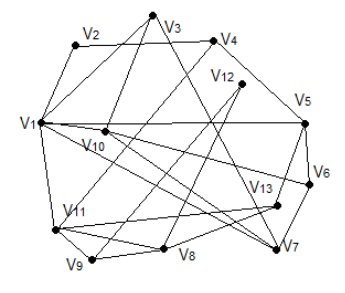
[Результати роботи завдання №2 3](#_Toc134948644)

[Блок-схема завдання №2 3](#_Toc134948645)

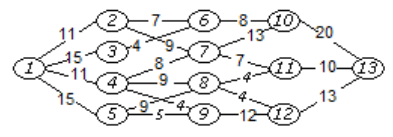
[Висновки 4](#_Toc134948646)

Текст завдання №1

1. За допомогою y-алгоритму зробити укладку графа у площині, або довести що вона неможлива



1. Побудувати повний потік, а потім скорегувати його до найбільшого (дуги спрямовані зліва направо)



Розв’язок завдання №1

Фото

Текст завдання №2

Написати програму, яка знаходить максимальний потік в мережі. Протестувати зробленою програмою мережу із завдання №1, задача 2

Розв’язок завдання №2

import networkx as nx

def manual\_max\_flow(G, s, t):

    flow = {}

    for u in G.nodes:

        flow[u] = {}

        for v in G.nodes:

            flow[u][v] = 0

    while True:

        path = bfs(G, s, t, flow)

        if not path:

            break

        df = float("inf")

        for u, v in path:

            df = min(df, G[u][v]["capacity"] - flow[u][v])

        for u, v in path:

            flow[u][v] += df

            flow[v][u] -= df

    max\_flow\_value = sum(flow[s][v] for v in G.nodes)

    return max\_flow\_value

def bfs(G, s, t, flow):

    queue = [s]

    paths = {s: []}

    while queue:

        u = queue.pop(0)

        for v in G[u]:

            if G[u][v]["capacity"] - flow[u][v] > 0 and v not in paths:

                paths[v] = paths[u] + [(u, v)]

                if v == t:

                    return paths[v]

                queue.append(v)

    return None

def create\_graph():

    G = nx.DiGraph()

    nodes = ['1', '2', '3', '4', '5', '6',

             '7', '8', '9', '10', '11', '12', '13']

    edges = [('1', '2', 11), ('1', '3', 15), ('1', '4', 11), ('1', '5', 15), ('2', '6', 7), ('2', '7', 9), ('3', '6', 4), ('4', '7', 8), ('4', '8', 9), ('4', '9', 4), ('5', '8', 9),

             ('5', '9', 5), ('6', '10', 8), ('7', '10', 13), ('7', '11', 7), ('8', '11', 4), ('8', '12', 4), ('9', '12', 12), ('10', '13', 20), ('11', '13', 10), ('12', '13', 13)]

    G.add\_nodes\_from(nodes)

    for edge in edges:

        G.add\_edge(edge[0], edge[1], capacity=edge[2])

    return G

G = create\_graph()

print()

print("Потік:")

for edge in G.edges():

    edge\_str = edge[0] + " - " + edge[1] + " -- " + \

        str(G.get\_edge\_data(edge[0], edge[1])['capacity'])

    print(edge\_str)

max\_flow\_value = manual\_max\_flow(G, "1", "13")

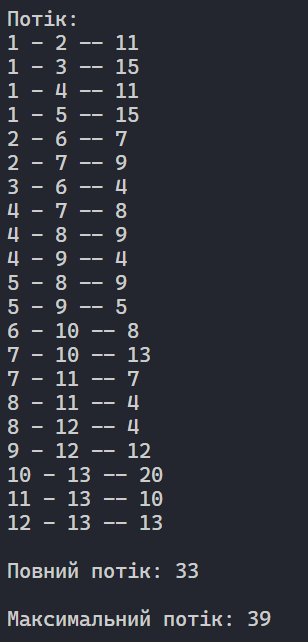
print("\nПовний потік:", max\_flow\_value)

max\_flow\_value, max\_flow\_dict = nx.maximum\_flow(G, '1', '13')

print("\nМаксимальний потік:", max\_flow\_value)

print()

Результати роботи завдання №2



Блок-схема завдання №2

Фото