**Міністерство освіти і науки України**

**Національний університет «Запорізька Політехніка»**

Кафедра програмних засобів

**ЗВІТ**

з лабораторної роботи №3

з дисципліни «Комп’ютерна дискретна математика»

На тему «ОСНОВНІ ОПЕРАЦІЇ НАД ГРАФАМИ»

Варіант №20

**Виконав:**

Студент групи КНТ-122 О. А. Онищенко

**Прийняли:**

Ст. Викладач О. А. Щербина

2023

[Мета роботи 3](#_Toc132872423)

[Текст завдання №1 3](#_Toc132872424)

[Розв’язок завдання №1 3](#_Toc132872425)

[Текст завдання №2 3](#_Toc132872426)

[Розв’язок завдання №2 3](#_Toc132872427)

[Блок-схема завдання №2 3](#_Toc132872428)

[Висновки 4](#_Toc132872429)

Текст завдання №1

**1. Виконати наступні операції над графами:**

- знайти доповнення до першого графу,

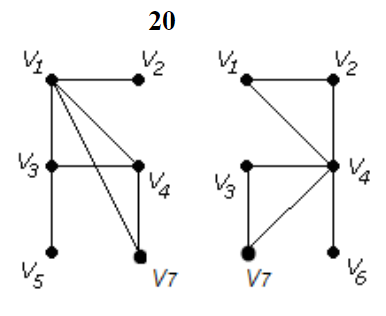
- об’єднання графів,

- кільцеву суму G1 та G2 (G1 ⊕ G2),

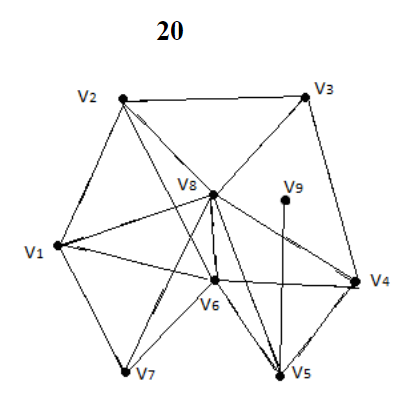
- розщепити вершину у другому графі,

- виділити підграф А, що складається з 3-х вершин в G1 і знайти стягнення А в G1 (G1\ A),

- добуток графів



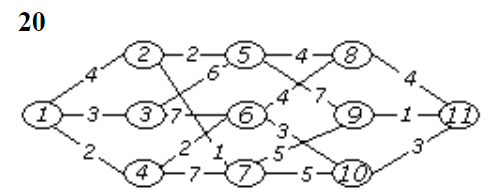
**2. Знайти таблицю суміжності та діаметр графа.**



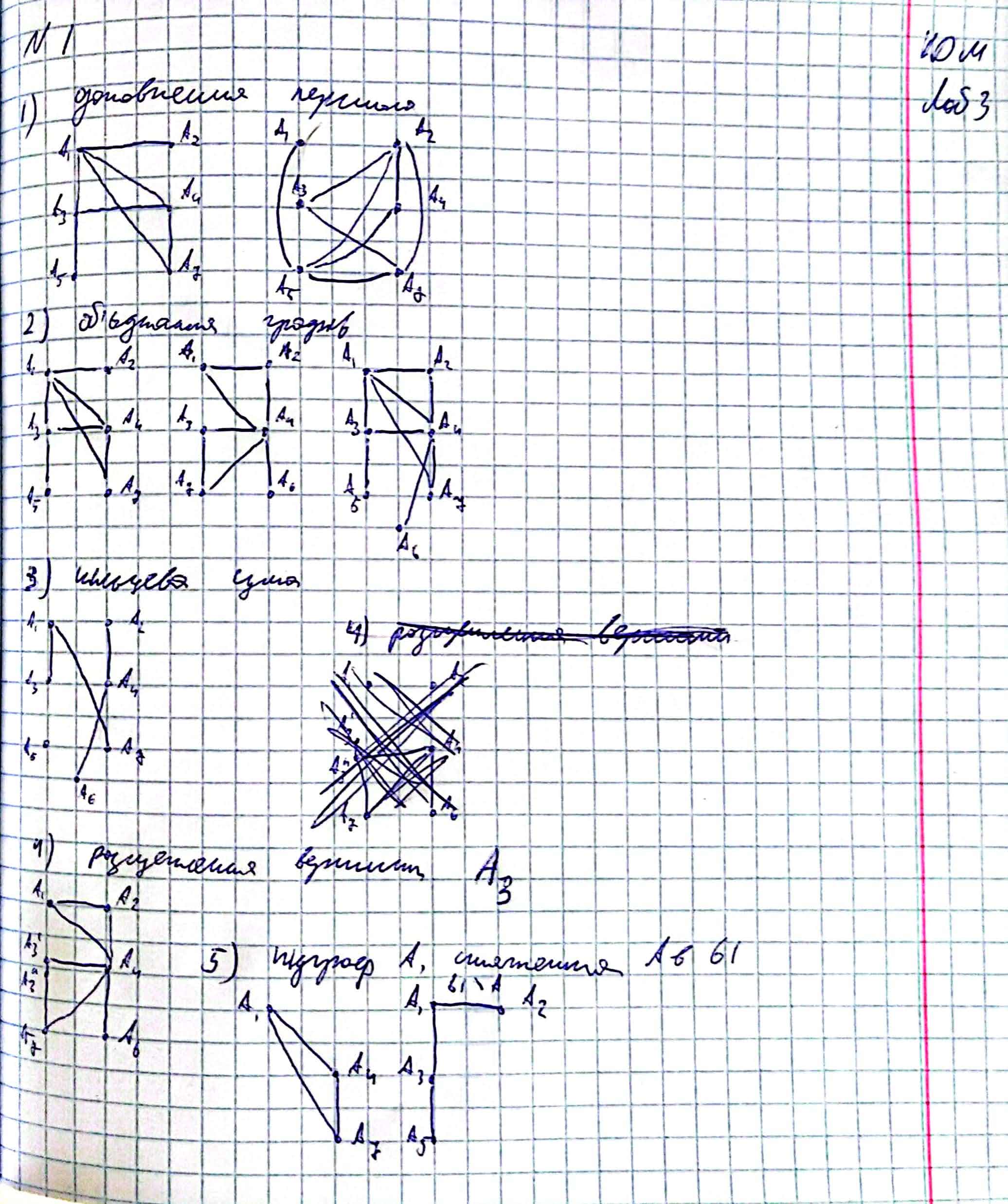
**3. Для наступного графа реалізувати 3 алгоритми:**

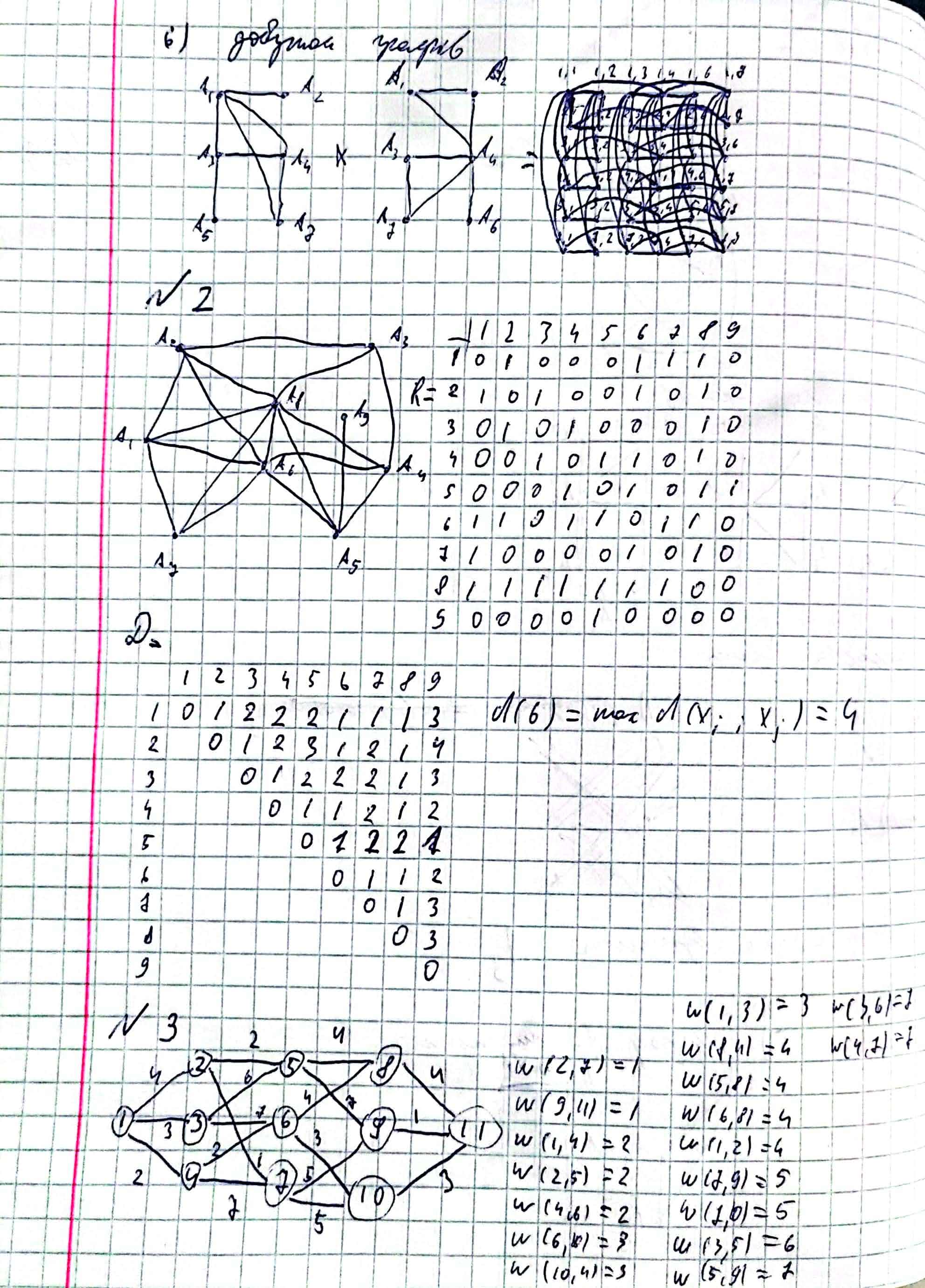
1. Знайти двома методами: Краскала і Прима, мінімальне остовне дерево графа. По крокам розписати вибір ребер за кожним алгоритмом. Знайти вагу остовного дерева.

2. Знайти найкоротший ланцюг між вершинами 1 та 11 методом Дейкстра. Розписати алгоритм по крокам.



Розв’язок завдання №1





Текст завдання №2

Написати програму, яка реалізує один з вивчених алгоритмів завдання 1.3: знаходження остовного дерева мінімальної ваги за алгоритмом Краскала, алгоритм Дейкстра пошуку найкоротшого ланцюга. Етапи розв'язання задачі виводити на екран. Протестувати розроблену програму на задачі 3 із завдання №1.

Розв’язок завдання №2

import networkx as nx

graph = nx.Graph()

num\_vertices = int(input("\nВведіть кількість вершин: "))

print()

for i in range(1, num\_vertices+1):

    for j in range(i+1, num\_vertices+1):

        if i != j:

            weight = int(

                input(f"Введіть вагу між вершиною {i} та {j}: "))

            if weight != 0:

                graph.add\_edge(i, j, weight=weight)

T = nx.minimum\_spanning\_tree(graph)

print(f"\nМінімальне остовне дерево:")

total\_weight = 0

for u, v, data in T.edges(data=True):

    weight = data['weight']

    print(f"{u} -- {v} = {weight}")

    total\_weight += weight

print(f"Вага дерева = {total\_weight}")

def dijkstra(graph, start, end):

    dist = {v: float('inf') for v in graph.nodes()}

    dist[start] = 0

    prev = {}

    unvisited = set(graph.nodes())

    while unvisited:

        current = min(unvisited, key=lambda v: dist[v])

        unvisited.remove(current)

        if dist[current] == float('inf'):

            break

        for neighbor in graph.neighbors(current):

            tentative\_dist = dist[current] + graph[current][neighbor]['weight']

            if tentative\_dist < dist[neighbor]:

                dist[neighbor] = tentative\_dist

                prev[neighbor] = current

    path = []

    current = end

    while current != start:

        path.append(current)

        current = prev[current]

    path.append(start)

    path.reverse()

    path\_str = " -> ".join(str(v) for v in path)

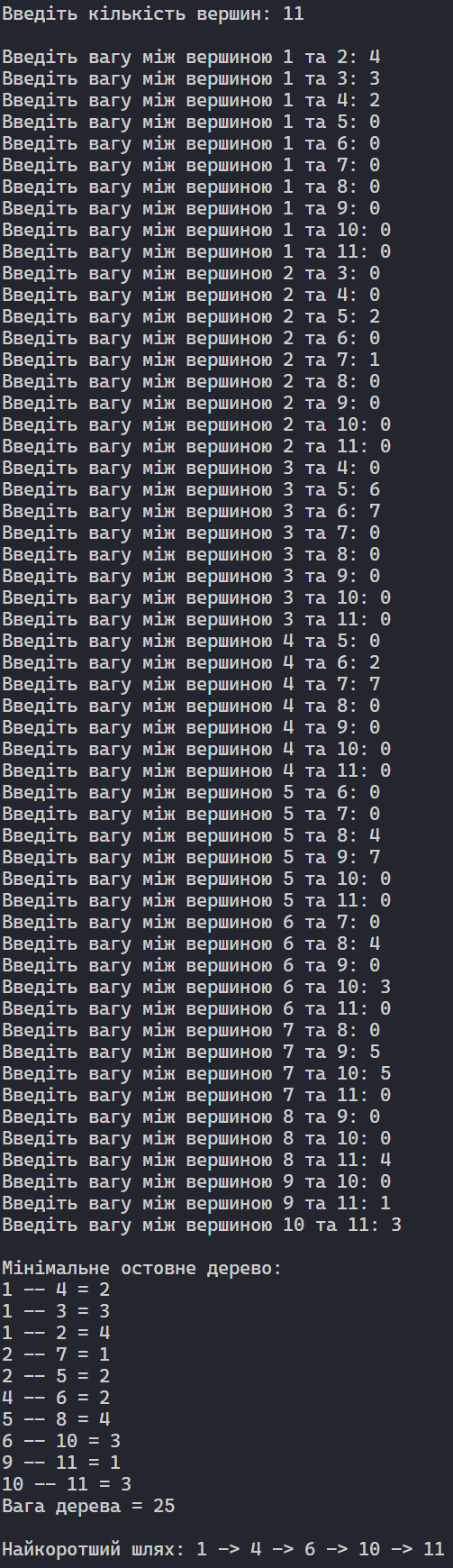
    return path\_str

start = list(graph.nodes())[0]

end = list(graph.nodes())[-1]

shortest\_path = dijkstra(graph, start, end)

print(f"\nНайкоротший шлях: {shortest\_path}\n")

Результати роботи завдання №2

Блок-схема завдання №2

Фото