МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ”ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра «Систем Автоматизованого Проектування»

Звіт
Лабораторна робота №5
з курсу *“* Дискретні моделі*”*
на тему: « ІЗОМОРФІЗМ ГРАФІВ»

Виконав:
Ст.гр. КН-409
Єлечко Олег
Прийняв:
Кривий Р.З.

Львів 2023

Мета роботи: Метою лабораторної роботи є вивчення і дослідження основних підходів до встановлення ізоморфізму графів.

Теоретичні відомості:

Теорія графів дає простий, доступний і потужній інструмент побудови моделей і рішення задач впорядкування взаємозвязаних обєктів. Нині є багато проблем де необхідно дослідити деякі складні системи з допомогою впорядкування їх елементів. До таких проблем відносяться і задачі ідентифікації в електричних схемах, в авіації, в органічній хімії і т.д. Вирішення таких проблем досягається з допомогою встановлення ізоморфізму графів.
Два графа G=(X,U,P) і G'=(X',U',P') називаються ізоморфними, якщо між їх вершинами, а також між їхніми ребрами можна встановити взаємно однозначне співвідношення X <-> X', U <-> U', що зберігає інцидентність, тобто таке, що для всякої пари (x,y)єX ребра u є U, що з'єднує їх, обов'язково існує пара (x',y') є X' і ребро u' є U', що з'єднує їх, і навпаки. Тут P - предикат, інцидентор графа G. Зауважимо, що відношення ізоморфізму графів рефлексивне, симетричне і транзитивне, тобто представляє собою еквівалентність.
На даний час існує досить детальна класифікація розроблених методів рішення такого типу задач. Розглядаючи комбінаторно-логічну природу вказаної задачі можна всі роботи в цьому напрямку розділити на дві групи:
рішення теоретичної задачі встановлення ізоморфізму простих графів;
розробка наближених методів, які найбільш повно враховують обмеження і специфіку задачі з
застосуванням характерних ознак об’єкту дослідження.
До першої групи відносяться алгоритми: повного перебору і почергового “підвішування” графів за вершини.

а) Одним з найпростіших з точки зору програмної реалізації, є алгоритм перевірки ізоморфізму графів повним перебором(можливої перенумерації вершин), але складність цього алгоритму є факторіальною.

б) Почергове “підвішування” графів за вершини (всі ребра зрівноважені). Суть цього алгоритму полягає в знаходженні однакових “підвішаних” графів (за довільні вершини), ізоморфність яких визначаємо. При чому в одному з графів почергово змінюється вершина за яку він “підвішується”.
Ізоморфізм графів визначається по їх матрицях суміжності, які формуються по однотипних правилах: індекс в матриці вершини за яку закріплений (“підвішаний”) граф рівний одиниці; кортеж вершин в матриці визначається рівнями сусідів; кортеж вершин в межах кожного рівня сусідів визначається степінню вершини, а також кількістю ребер над нею і нижче її.

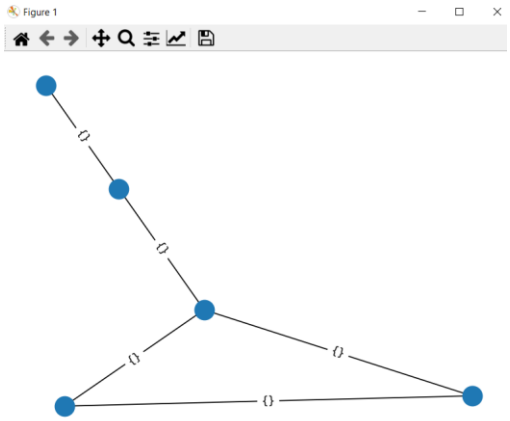
ЛАБОРАТОРНЕ ЗАВДАННЯ

1. Отримати у викладача індивідуальне завдання.
2. Підготувати програму для вирішення виданого завдання.
3. Запустити на виконання програму відповідного методу.

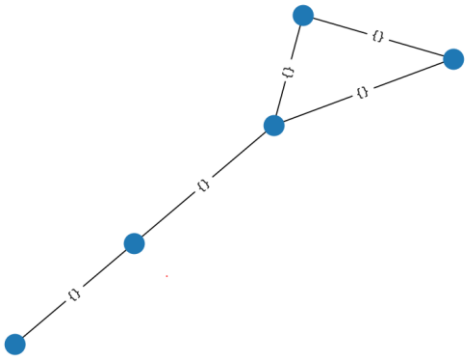
4. Проглянути результат роботи програм. Результат роботи може бути: ізоморфізм встановлено або не встановлено.
5. У випадку, коли ізоморфізм встановлено (не встановлено), необхідно модифікувати граф, коректуючи два або три зв'язки, щоб знайти такий граф, на якому ізоморфізм не встановлюється (встановлюється).
6. Зафіксувати результати роботи у викладача.
7. Оформити і захистити звіт
   Github-link: https://github.com/OlehYelechko/Labs_DM/tree/main/Lab5

Код алгортиму повного перебору вершин.

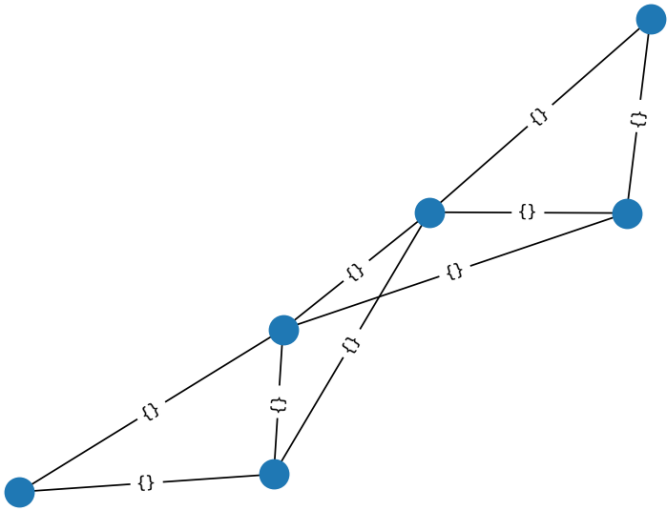
```
import networkx as nx
import matplotlib.pyplot as plt
import numpy as np

# Створюємо графи
G1 = nx.Graph()
G1.add_edges_from([(1, 2), (1, 3), (2, 3), (3, 4), (4, 5)])
layout = nx.spring_layout(G1)
nx.draw(G1, layout)
nx.draw_networkx_edge_labels(G1, pos=layout)
plt.show()

G2 = nx.Graph()
G2.add_edges_from([(10, 20), (10, 30), (20, 30), (30, 40), (40, 50)])
layout = nx.spring_layout(G2)
nx.draw(G2, layout)
nx.draw_networkx_edge_labels(G2, pos=layout)
plt.show()
# Перевірка на ізоморфізм
isomorphic = nx.is_isomorphic(G1, G2)
print("Isomorphic: ", isomorphic)

# Якщо графи ізоморфні, модифікуємо один з графів
if isomorphic:
    # Додаємо вершину, якої немає в іншому графі
    G1.add_node(6)
    G1.add_edge(6, 1)
    G1.add_edge(6, 3)
    G1.add_edge(6, 4)
    G1.add_edge(6, 5)
    # Перевірка на ізоморфізм після модифікації
    isomorphic = nx.is_isomorphic(G1, G2)
    print("Isomorphic after modification: ", isomorphic)

layout = nx.spring_layout(G1)
nx.draw(G1, layout)
nx.draw_networkx_edge_labels(G1, pos=layout)
plt.show()
```



Результат виконання:



```
Isomorphic:  True
Isomorphic after modification:  False
```



Висновок: на цій лабораторній роботі було реалізовано метод перебору вершин для визначення ізоморфмності графів на мові Python. Наведено результат роботи, коли графи є ізомофрні і не ізоморфні.