Прізвище: Пастернак
Ім'я: Вероніка
Група: КН-406
Варіант: 24
GitHub: https://github.com/veronikalpnu/8semest-kriviy
Кафедра: САПР
Дисципліна: Дискретні моделі САПР
Перевірив: Кривий Р.З.

ЗВІТ

до лабораторної роботи №5

на тему: "Ізоморфізм графів"

Мета роботи:

вивчити і дослідити основні підходи до встановлення ізоморфізму графів.

Теоретичні відомості:

Теорія графів дає простий, доступний і потужній інструмент побудови моделей і рішення задач впорядкування взаємозв'язаних об'єктів. 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 $(x, y) \in X$ ребра $u \in U$, що з'єднує їх, обов'язково існує пара $(x', y') \in X'$ і ребро $u' \in U'$, 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 /1/. 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.

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

Роботи першої групи рідко використовуються для розробки безпосередньо алгоритмів рішення задачі ідентифікації, але вони дозволяють дати оцінку обчислювальної складності алгоритмів її рішення, які відносяться до задач зі складним рішенням (*NP* - повні задачі), складність алгоритмів яких є експоненціальною і в деяких випадках сягає *O(N!)*, (*N* - кількість вершин), тому доводиться відмовлятись від спроб досягнути точними методами їх рішення.

Наближені алгоритми використовують або допустимі рішення точних методів, або побудованні на використанні евристичних прийомів. На даний час створена певна кількість алгоритмів встановлення ізоморфізму графів, які за рахунок різних евристичних прийомів знижують складність задачі від факториальної до степеневої функції. Серед них можна виділити наступні:

c) Метод побудови оптимального коду графа.

Цей метод базується на алгоритмі формування еквівалентних матриць зв'язності щляхом ідентифікації вершин з однаковими топологічними характеристиками. Згідно цього алгоритму перетворюються матриці зв'язності графів однотипними методами. У випадку, коли модифіковані матриці однакові графи ізоморфні. Ізоморфізм між двома чи більше графами може бути визначений при допомозі використання їхніх оптимальних кодів. Оптимальний код графу отримується з верхнього трикутника матриці суміжності після перенумерації вершин графа.

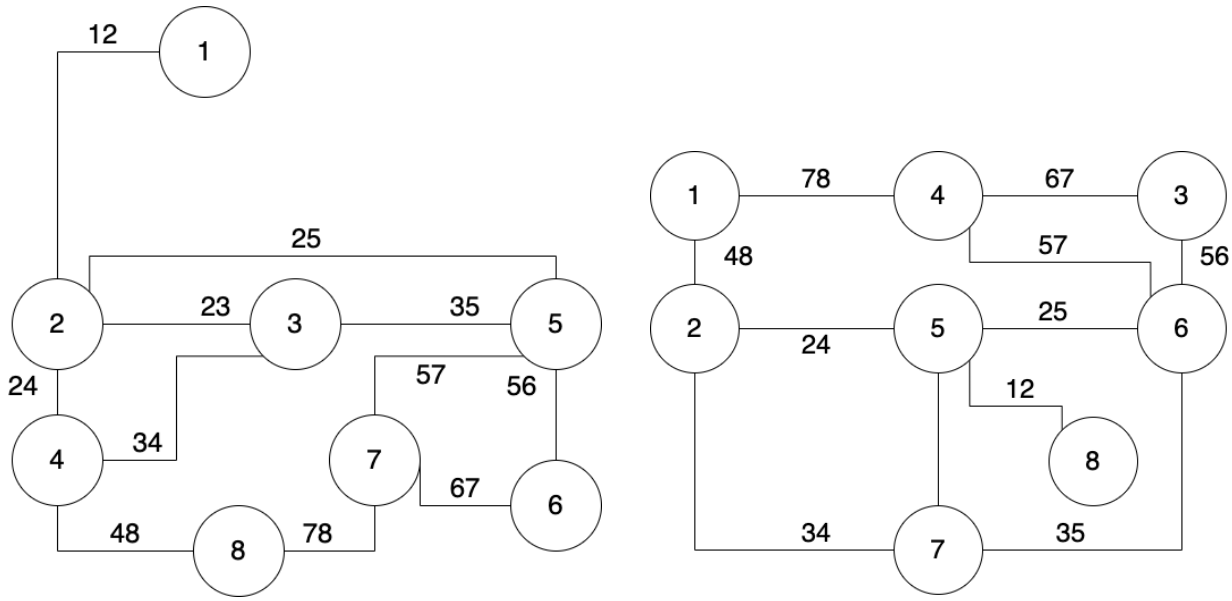
Згідно з методом побудови оптимального коду графа здійснюється перенумерація вершин графів, які досліджуються на ізоморфізм. Перенумерація вершин графа відбувається по певних вибраних критеріях оцінки вершин графу. Оцінка здійснюється по їх числових значеннях, які використовуються при формуванні оптимального коду графа

Індивідуальне 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

Ручні обчислення:

1) Я створила два графа:



2) Спочатку перевіряємо степені вершин графа та кількість ребер:

Для графа 1:

- 11 ребер
- Впорядковані степені для вершин [1, 2, 2, 3, 3, 3, 4, 4]

Для графа 2:

- 11 ребер
- Впорядковані степені для вершин [1, 2, 2, 3, 3, 3, 4, 4]

3) Як бачимо кількість ребер і степені вершин одникові для обох графів.

4) Вершина першого графа 1 – це вершина друго графа 8.

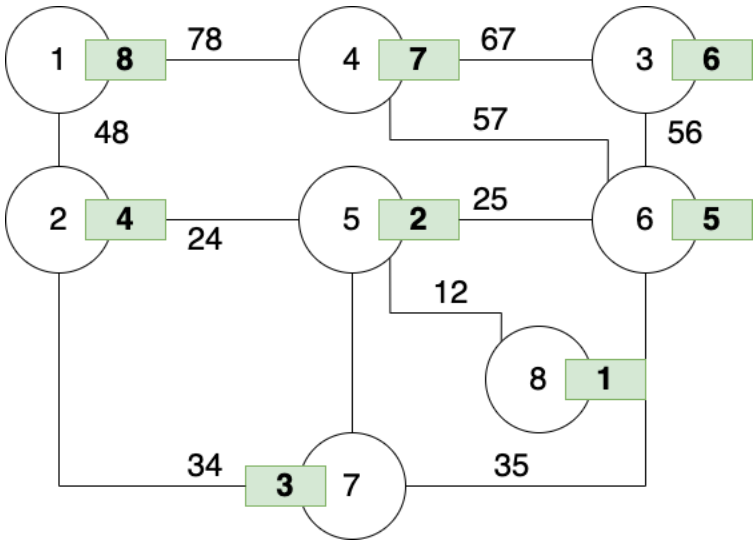
Далі в першому графі маємо вершину 2, так як вона єдина з’єднана з вершиною 1, аналогічно в другому графі маємо вершину 5.

Вага ребер між вершина співпадає – 12.

5) На кожному кроці порівнюємо степені вершин і їхніх сусідів у другого графа і розставляємо у відповідність їм вершини першого графа.

Якщо графи ізоморфні, усі вершини другого графа мають співпасти так, щоб їхні степені та степені їхніх сусід співпали зі аналогічним з першого графа.

6) Отже, зображено другий граф з підписаними відповідними вершинами з першого графа, які сходяться з другим:



Програмна реалізація (python):

```
import networkx as nx

G1 = nx.Graph()
G1.add_nodes_from(['KA', 'TN', 'TL', 'AP', 'KL', 'GO'])
G1.add_edges_from([('KA', 'TN'), ('KA', 'TL'), ( 'KA', 'TL'), ( 'KA', 'AP'), ('KA',
'KL'), ('KA', 'GO')])
# nx.draw(G1,with_labels=1)

G2 = nx.star_graph(5)

start = nx.is_isomorphic(G1, G2)
```

Результати виконання програми:

```
>>>
======= RESTART: /Users/verikpaster/Documents/NULP/4.2/CAI
>>> start
True
>>>
```

Висновок

В ході даної лабораторної роботи я ознайомилась з поняттями ізоморфності графів та методами, за якими можна перевірити це. Було проведені ручні обчислення для графів, які я створила, а також реалізовано програму, яка буде виконувати перевірку вказаних графів в коді за допомогою модифікатора - networkx.