DOI 10.31891/2307-5732-2024-335-3-37
УДК 004.4,6

ФЕДОРКО МАКСИМ
ЗВО Університет Короля Данила
https://orcid.org/0009-0008-3472-058X
e-mail: dziubamaryna2015@ukr.net
ДЗЮБА МАРИНА
ЗВО Університет Короля Данила
https://orcid.org/0000-0003-2579-9157
e-mail: dziubamaryna2015@ukr.net

РОЗРОБКА ТА РЕАЛІЗАЦІЯ АЛГОРИТМУ УПІЗНАННЯ ІЗОМОРФНОСТІ ГРАФІВ

Стаття присвячена проблемі створення алгоритму та програми встановлення ізоморфності двох даних графів методом перебору; дослідженню обчислювальної ефективності створеного алгоритму упізнання ізоморфності графів. Досліджено проблему ізоморфності графів та запропоновано її практичну реалізацію на мові програмування C#. Розглянуто теоретичні аспекти задачі, запропоновано алгоритм для її вирішення і продемонстровано його втілення у програмному забезпеченні, в тому числі, поняття графу, як структури, що складається з об'єктів (вершин) та зв'язків між ними (ребер). Ізоморфність двох графів означає, що між ними існує особливий зв'язок (бієкція), який зберігає структуру зв'язків. Іншими словами, графи можна "перетворити" один в один, переставляючи та перейменовуючи вершини, не руйнуючи загальної картини. Визначення ізоморфності графів має важливе значення в багатьох сферах, таких як розробка комп'ютерних чипів, аналіз еволюції, контроль натовпу та хімії.

Робота пропонує алгоритм для визначення ізоморфності графів, що ґрунтується на порівнянні матриць суміжності. Матриця суміжності – таблиця, де кожен елемент показує, чи існує зв'язок між двома певними вершинами. Для реалізації алгоритму розроблено програмне забезпечення на C#. Воно дозволяє додавати та видаляти вершини й ребра, отримувати матриці суміжності, визначати ізоморфність графів та їх класифікувати. Програмне забезпечення успішно протестовано на різних типах графів. Відзначено використання об'єктно-орієнтованого програмування (ООП) для чіткої та структурованої організації коду. Також реалізовано зручний інтерфейс командного рядка для роботи з графами. Загалом, програма може бути корисною для вирішення різних задач, пов'язаних з аналізом графів. Важливим напрямком досліджень є час роботи алгоритму та його порівняння з іншими методами визначення ізоморфності графів. Корисним є дослідити приклади використання програмного забезпечення для вирішення реальних задач, зробити код програми відкритим для публічного використання.

Ключові слова: графи, ізоморфність графів, матричний метод, матриця суміжності, об'єктно-орієнтоване програмування, C#, аналіз графів, оптимізація, алгоритми.

FEDORKO MAKSYM, DZIUBA MARYNA
HEI King Danylo University

ON THE DEVELOPMENT AND IMPLEMENTATION OF THE GRAPH ISOMORPHITY RECOGNITION ALGORITHM

The article is devoted to the problem of creating an algorithm and a program for establishing the isomorphism of two given graphs by the iterative method; study of computational efficiency of the created algorithm for graph isomorphism recognition. This paper delves into the problem of graph isomorphism and presents its practical implementation using the C# programming language. It explores the theoretical aspects of the problem, proposes an algorithm for its resolution, and demonstrates its embodiment in software. A graph is a structure composed of objects (vertices) and connections between them (edges). The isomorphism of two graphs implies the existence of a special connection (bijection) between them that preserves the structure of the connections. In other words, graphs can be "transformed" into each other by rearranging and renaming vertices without disrupting the overall picture. Determining graph isomorphism is crucial in various fields, including computer chip design, evolutionary analysis, crowd control, and chemistry. The paper proposes an algorithm for determining graph isomorphism based on comparing adjacency matrices. An adjacency matrix is a table where each element indicates whether a connection exists between two specific vertices. Was developed software on C# to implement the algorithm. It allows adding and removing vertices and edges, obtaining adjacency matrices, determining graph isomorphism, and classifying graphs. The software was successfully tested on various graph types. It is worth noting the use of object-oriented programming (OOP) for clear and structured code organization. A user-friendly command-line interface was also implemented for working with graphs. Overall, the program can be beneficial for solving various tasks related to graph analysis. However, there is room for improvement. An essential research direction is investigating the algorithm's runtime and comparing it to other graph isomorphism determination methods. It would also be beneficial to include examples of using the software to solve real-world problems. Ideally, the program's code should be made open-source.

Keywords: graphs, graph isomorphism, matrix method, adjacency matrix, object-oriented programming, C#, graph analysis, optimization, algorithms.

Постановка проблеми у загальному вигляді та її зв’язок із важливими науковими чи практичними завданнями

Дослідження ізоморфності графів є однією з тих тем, над якими продовжуються дослідження для визначення найефективнішого способу дослідження графів. Однією з галузей, де швидкість і кращі з’єднання мають вирішальне значення, є розробка комп'ютерних чипів. Інтегральні схеми складаються з мільйонів транзисторів. Для поліпшення продуктивності чипа необхідно оптимізувати значну кількість зв’язків. Теорія графів також грає важливу роль в аналізі та візуалізації еволюції тварин і мов, контролю

натовпу і поширення захворювань. В хімії графи використовують для прогнозування хімічних перетворень та візуального зображення залежностей, зв'язків, класифікацій та ієрархій. В кінці XIX ст. А. Келі вивів формулу кількості неорієнтованих дерев з n поміченими вершинами, яка розв'язувала задачу про можливі структури насичених (або граничних) вуглеводнів. Молекула кожного граничного вуглеводню може бути представлена у вигляді дерева (ациклічного графу). При видаленні атомів водню решта атомів також буде утворювати дерева з валентністю вершин не вище 4. Число можливих структур можливих вуглеводів є числом скінченим і дорівнює числу дерев з вершинами степеня не більше 4. До хімічних графів відносяться молекулярні, дводольні та сигнальні графи кінетична рівнянь [1].

Аналіз останніх досліджень і публікацій

Дослідженню методів розв'язанню графів присвячені роботи Н. Робертсона, С. Сеймура, Л. Бабкок, М. Холл, А. Кнудсен та ін. Особливий внесок вніс У. С. Кейлі, який ввів поняття ізоморфності графів, та довів, що два графи є ізоморфними, якщо й лише якщо вони мають однакову структуру зв'язків. Також Л. Поса запропонував ідею в якій знаходження ізоморфності ґрунтується на розкладанні графів на менші підграфи. Було доведено, що визначення ізоморфності графів є NP-повним завданням, тобто не існує відомого алгоритму, який би міг розв'язати цю задачу для будь-якого графу за поліноміальний час. Але за певних умов, от як алгоритм Бабкока-Сеймура, який може визначити ізоморфність двох графів за поліноміальний час, за умови, що це планарні графи [1]. Незважаючи на значний прогрес, все ще залишаються невирішені питання. Розробка універсального методу, який би підходив для розв'язання всіх задач з ізоморфності графів, є складним завданням. Ефективність алгоритмів ізоморфності графів може значно варіюватися залежно від характеристик графів. Розробка нових методів для розв'язання задач з ізоморфності графів великих розмірів є актуальною темою досліджень.

Формулювання цілей статті

Метою роботи є дослідження проблем розв'язування задач, пов'язаних з ізоморфністю графів та аналіз створеного програмного забезпечення для розв'язання такого типу задач.

У статті описано дослідження цієї теми і її реалізації на мові програмування C# і використання такої парадигми програмування, як об'єктно-орієнтоване програмування (ООП).

Виклад основного матеріалу

Граф – це сукупність об'єктів та зв'язків між ними [1]. Об'єкти розглядаються як вузли або вершини, а зв'язки як ребра. Граф можна зобразити як точки та відрізки між ними (рис. 1).

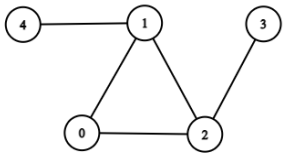

Рис. 1. Зображення графу

Графи розрізняють за направленістю, повнотою. Приклад вигляду напрямленого графу (рис. 2) і неповного графу (рис. 3) [2].

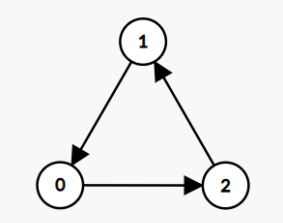

Рис. 2. **Зображення напрямленого графу**

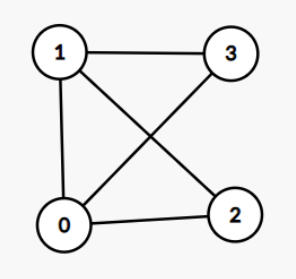

Рис. 3. Зображення неповного графу

Отже, графом є певні об'єкти і зв'язки між ними. Наприклад, об'єктом може бути людина, а зв'язком чи знає певна людина іншу людину [3].

Ізоморфність графа – два графа називаються ізоморфними якщо існує бієкція між їхніми вершинами, що має властивість: якщо в першому графі є ребро, то у другому графі має бути відповідне ребро, яке проходить через відповідні вершини [4].

Питання про складність задачі ізоморфності графів [5] є досі відкритим і на цей час не доведена ні її NP – повнота ні її належність до класу P. Якщо це все узагальнити, то треба взяти два графа і перевірити чи існують між ними відповідні вершини та ребра (рис. 4).

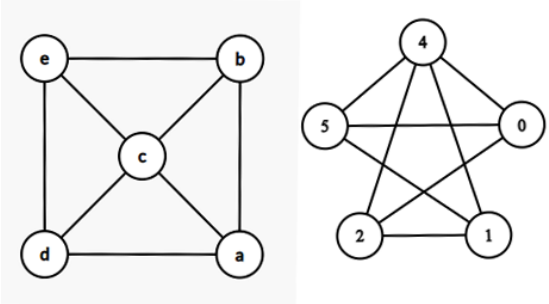

Рис. 4. Зображення двох ізоморфних графів

Одним із способів знаходження ізоморфності двох графів є метод перебору матриці суміжностей. Перевірка ізоморфності графів за матрицею суміжностей має на увазі перестановку рядків і стовпців у матриці суміжності одного графа, щоб у результаті вийшла матриця суміжності іншого графа, якщо така є – то графи ізоморфні. До прикладу (рис. 5) наведені дві матриці суміжностей для двох графів (рис. 5).

Adjacency Matrix					
	0	1	2	3	4
0	0	1	1	1	0
1	1	0	0	1	1
2	1	0	0	1	1
3	1	1	1	0	1
4	0	1	1	1	0

Adjacency Matrix					
	0	1	2	3	4
0	0	1	1	1	0
1	1	0	0	1	1
2	1	0	0	1	1
3	1	1	1	0	1
4	0	1	1	1	0

Рис. 5. Матриці суміжності до графів на рис. 4

Щодо реалізації в цьому алгоритмі, кількість операцій N буде дорівнювати факторіалу від довжини помножену на факторіал висоти матриці суміжності, оскільки матриця завжди квадратна, то це можна скоротити до довжини матриці у квадраті. Тобто за нотацією Ландау виходить O(n!2), де *n* - це довжина матриці, або кількість вершин в графі, що є тим самим. Можна побачити як швидко зростає функція (рис. 6). Тобто кількість операцій, а надалі й час визначення ізоморфності дуже чутливі до кількості вершин в графі.

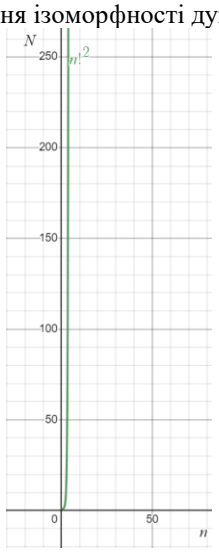

Рис. 6. Графічне зображення O(n!2)

Звідси випливає, що рішення не є найефективнішим способом вирішення даної задачі, але це найпростіший спосіб вирішення поданої задачі і для підтвердження її роботи розроблено наступну архітектуру програмного коду, на основі UML діаграм [8] (рис. 7).

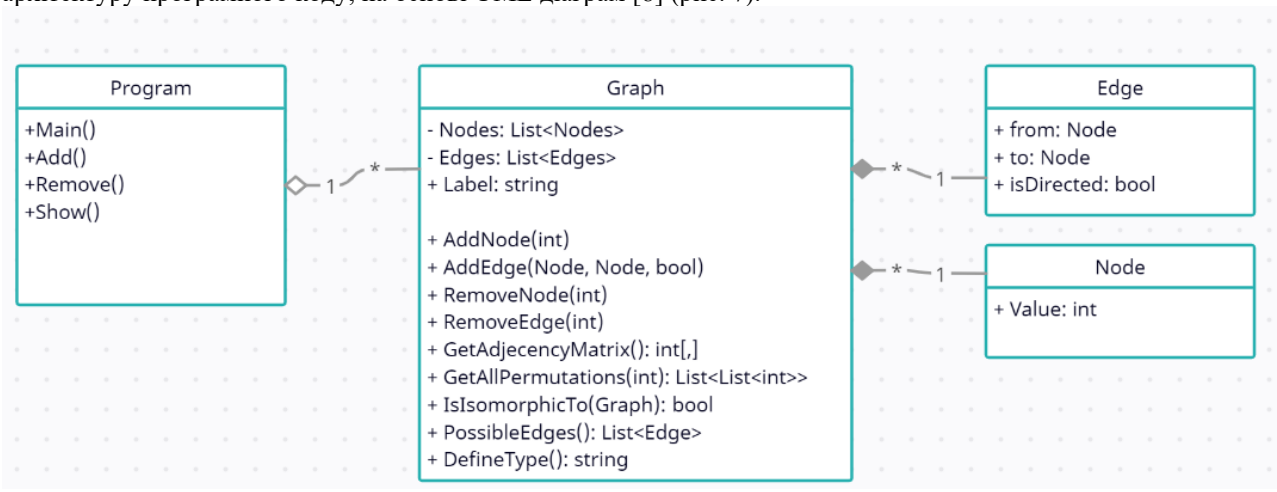

Рис. 7. UML зображення структури програми

Клас Program виступає інтерфейсом вводу/виводу інформації за допомогою консолі. Клас Graph, в якому описані всі потрібні функції і поля, буде основним класом для роботи з графами та визначення їхньої ізоморфності. Також застосовано дві структури Edge та Node, щоб описати ноди (точки, об'єкти) і зв'язки графа, для зручнішої реалізації програмного коду.

Реалізувавши цю програму, опираючись на норми і рекомендації написання коду [7], протестувавши її, можна отримати наступні результати.

Випадок 1

В цьому випадку було створено два не ізоморфних графа і перевірено їхню ізоморфність.

Введені команди:

add graph firstGraph
add graph secondGr aph
add node firstGraph 10
add node firstGraph 20
add node firstGraph 15
add node secondGraph 10
add node secondGraph 20
add node secondGraph 15
add edge firstGraph 1 2 false
add edge firstGraph 2 3 false
add edge secondGraph 2 3 false
show isIsomorphic firstGraph secondGraph

І у відповідь було отримано **false**, і відповідь правильна, матриці суміжності двох графів можна побачити на рисунку 8.

Рис. 8. Матриця суміжності двох неізоморфних графів

Випадок 2

В цьому випадку було створено два ізоморфні графи і продемонстровано результат (рис. 9).

Введені команди:

add graph firstGraph
add graph secondGraph
add node firstGraph 10
add node firstGraph 20

add node firstGraph 15
add node secondGraph 10
add node secondGraph 20
add node secondGraph 15
add edge firstGraph 1 2 false
add edge firstGraph 2 3 false
add edge secondGraph 2 3 false
add edge secondGraph 1 2 false
show isIsomorphic firstGraph secondGraph

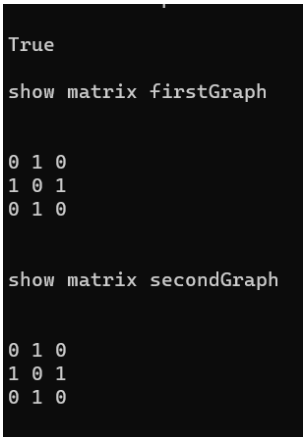

Рис. 9. Матриця суміжності двох ізоморфних графів

Випадок 3

В цьому випадку протестовано можливість програми визначати тип графу. Було створено два графи з 3 нодами, але в firstGraph є два зв'язки між 1 і 2, та між 2 і 3, а ось в secondGraph ці зв'язки відстуні. Відповідно firstGraph є простим графом, а secondGraph є нульовим, оскільки в нього немає зв'язків (рис. 10).

Введені команди:

add graph firstGraph
add graph secondGraph
add node firstGraph 10
add node firstGraph 20
add node firstGraph 15
add node secondGraph 10
add node secondGraph 20
add node secondGraph 15
add edge firstGraph 1 2 false
add edge firstGraph 2 3 false

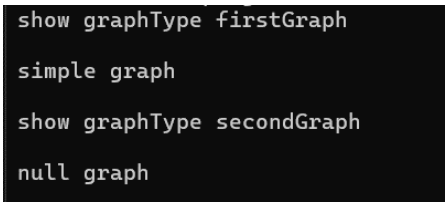

Рис. 10. Результат перевірки програми на спроможність визначати тип графа

Висновки з даного дослідження і перспективи подальших розвідок у даному напрямі

У програмному коді реалізовано клас Graph який дозволяє додавати вершини, додавати ребра, видаляти вершини, видаляти ребра, отримувати матрицю суміжності, знаходити ізоморфність двох графів та класифікувати граф. Це все наочно можна побачити не тільки в коді, але і в UML діаграмах.

Реалізовано роботу програми як консольної програми для зручності користувача оперувати нею і виконувати наведені вище операції над множиною графів. Парадигма об'єктно-орієнтованого програмування дуже вдало вписалась в умови цієї задачі і використання цієї парадигми було доцільним, так само як і вибір мови C#.

References

1. Reinhard Diestel Graph Theory: 5th edition. Springer-Verlag, 2017. 447 p. URL: https://books.google.com.ua/books/about/Graph_Theory.html?id=zIxRDwAAQBAJ&redir_esc=y

2. John Adrian Bondy Graph theory with applications. AEPC, 1976. 264 p. URL: https://www.google.com.ua/books/edition/_/4bwrAAAAYAAJ?hl=uk&gbpv=0
3. Brenda D. Mckay Practical graph isomorphism, II. Australian National University, 2014. 19 p. URL: https://www.sciencedirect.com/science/article/pii/S0747717113001193
4. James Eells, Domingo Toledo Hassler Whitney Collected Papers. Volume I, AMS. 1931. 592 p. URL: https://link.springer.com/book/10.1007/978-1-4612-2972-8
5. Scott Frotin The Graph Isomorphism Problem. University of Alberta. 1996. 25 p. URL: https://era.library.ualberta.ca/items/f8153faa-71bf-4b64-9eb4-f0c6d3b529dd
6. Robert C. Martin. Clean Code. Pearson Education. 2008. 464 p. URL: https://www.google.com.ua/books/edition/Clean_Code/_i6bDeoCQzsC
7. Martin Fowler. UML Distilled: A Brief Guide to the Standard Object Modeling Language. Pearson Education. 2018. 208 p. URL: https://www.google.com.ua/books/edition/UML_Distilled/VTdtDwAAQBAJ