

Міністерство освіти і науки України  
Національний університет «Києво-Могилянська академія»

Кафедра математики факультету інформатики

Курсова робота на тему:

**Відновлення автоморфізму графа за визначальною множиною**

Керівник курсової роботи:

професор, доктор фіз.-мат. наук

Олійник Б. В.

*(прізвище та ініціали)*

---

(підпис)

« \_\_\_\_ » \_\_\_\_\_ 2023 р.

Виконав студент

3-го року навчання спеціальності

113 «Прикладна математика»

Болотов Єгор Сергійович

(ПІБ)

**Тема:** Відновлення автоморфізму графа за визначальною множиною

**Календарний план виконання роботи:**

Номер	Назва етапу курсової	Термін	Примітка
1.	Отримання теми курсової роботи.	08.09.2022	
2.	Ознайомлення з темою курсової.	12.11.2022	
3.	Розробка плану та структури роботи.	08.12.2022	
4.	Робота з науковою літературою.	15.02.2023	
5.	Опис основних означень, дослідження методів визначення визначальною множини.	22.03.2023	
6.	Робота над прикладами, оформлення текстових результатів.	25.04.2023	

## Зміст

Вступ.....	4
Основні означення.....	5
Знаходження визначальних множин.....	10
Приклади відновлення автоморфізму графа за допомогою визначальної множини.....	12
Використання при роботі з даними.....	14
Використання при роботі з алгоритмами знаходження автоморфізмів .....	17
Висновок.....	21
Список використаних джерел .....	22

## Вступ

Відновлення автоморфізму графа є важливою задачею у теорії графів й знаходить широке поле використання у різних сферах.

**Основною метою** є дослідження й аналіз алгоритмів й методів визначення автоморфізму графа, а саме за його визначальною множиною. Задля цього буде розглянуто основні поняття, приклади та використання у деяких галузях.

Це може бути непростю задачею, оскільки вимагає пошуку невеликого набору вершин, котрі однозначно визначають усі симетрії графа. Тому вважатимемо, що для цього може знадобитись додаткова інформація або припущення.

Відновлення автоморфізму графа за визначальною є цікавою темою для дослідження. Воно дає можливість зрозуміти, як можна відновити автоморфізм графа з використанням лише обмеженої кількості інформації.

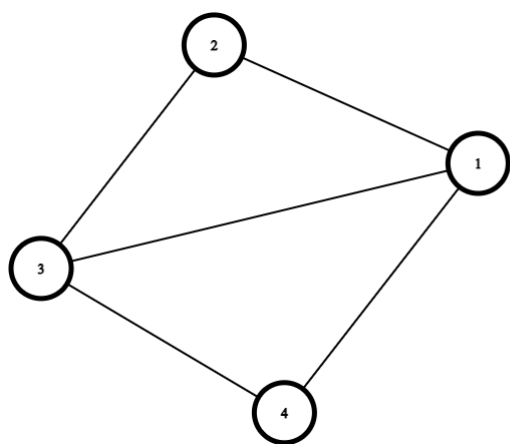
## Основні означення

Дані визначення знадобляться нам для розбору алгоритмів відновлення автоморфізму графа за визначальною множиною та в деяких доведеннях щодо цього.

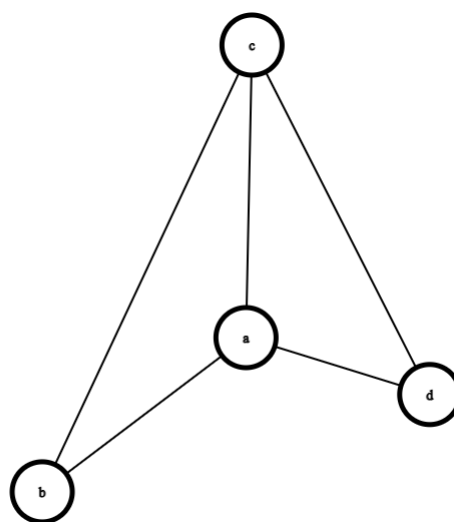
**Означення 1.** Два графи  $G_1$  та  $G_2$  є **ізоморфними**, коли у них існує бієктивне відображення, зберігаючи відношення суміжності вершин графа. Позначають як  $G_1 \cong G_2$ . Для доведення ізоморфізму існують необхідні умови, котрі розглянемо у прикладі.

Якщо коротко, то ізоморфний граф – граф, котрий отримали шляхом змінення зображення на площині.

**Приклад 1.** Ілюстрація двох графів, вони є ізоморфними, незважаючи на їх зовнішню різницю.



Граф  $G$



Граф  $H$

Графи є ізоморфними тоді і тільки тоді, коли їх матриці суміжностей можна отримати одна з другої однаковими перестановками рядків і стовпчиків.

Запишемо матриці суміжностей для графа  $G$  та  $H$ .

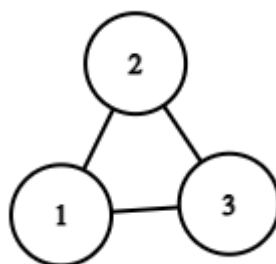
$$G = \begin{vmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{vmatrix}$$

$$H = \begin{vmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{vmatrix}$$

Можна побачити, що дані графи мають однакову матриці суміжностей й можна сказати, що вони – ізоморфні.

**Означення 2. Автоморфізм графу** – перестановка  $\sigma$  множини вершин  $V$  для графа  $G = (V, E)$ , що для будь-якого ребра  $w = (u, v)$ ,  $\sigma(w) = (\sigma(u), \sigma(v))$  також ребро. Тобто, це ізоморфне відображення графа самого на себе. Позначають як  $Aut(G)$ .

**Приклад 2.** Для прикладу використаємо повний граф з 3-ма вершинами



Граф  $H$

1 ↦ 1		1 ↦ 2		1 ↦ 3	
2 ↦ 2	2 ↦ 3	2 ↦ 1	2 ↦ 3	3 ↦ 1	2 ↦ 1
3 ↦ 3	3 ↦ 2	3 ↦ 3	3 ↦ 1	2 ↦ 2	3 ↦ 2
$\varepsilon$	$\alpha_1(2,3)$	$\alpha_2(1,2)$	$\alpha_3(1,2,3)$	$\alpha_4(1,3)$	$\alpha_5(1,3,2)$

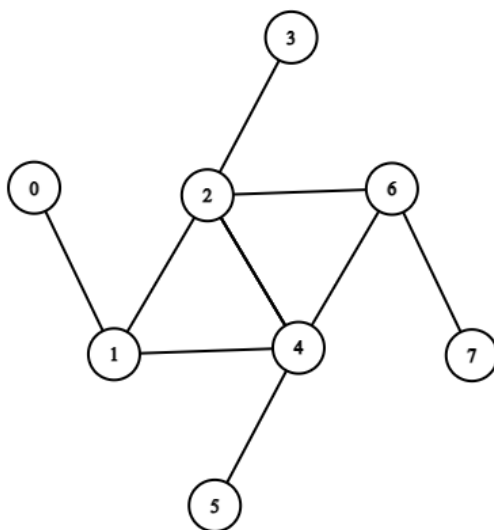
Можна побачити, що існує 6 варіантів перестановки множини вершин даного графа. Тобто  $|Aut(K_3)| = 6$ .

**Означення 3.** Сукупність усіх вершин графа, які можна поміняти місцями за допомогою певного розташування, називається **орбітою** графа. Тобто орбітами графа є класи еквівалентності вершин графа під дією автоморфізмів.

**Приклад 3.** Розглянемо граф  $R$ . Визначимо всі його автоморфізми.

$|Aut(G)| = 4$ , групою автоморфізмів є  $[ \varepsilon, (2\ 4)(3\ 5), (0\ 7)(1\ 6), (0\ 7)(1\ 6)(2\ 4)(3\ 5) ]$ .

Орбітами даного графа є множина:  $\{ (0\ 7), (2\ 4), (3\ 5), (1\ 6) \}$

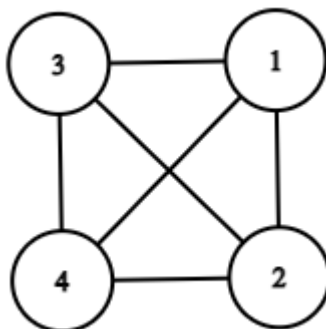


Граф R

**Означення 4.** Підмножина  $S$  вершин графа  $G$  є **визначальною множиною**, якщо для всіх  $g$  і  $h$  є автоморфізмами з властивістю, що  $g(s) = h(s)$  для всіх  $s \in S$ , то  $g = h$ .

Кожен граф має визначальну множину, з цього можна сказати, що будь-яка множина, котра містить всі вершини, окрім однієї, є визначальною множиною.

**Приклад 4.** Розглянемо повний граф з 4-ма вершинами.

Граф  $K_4$ , позначимо як  $G$ 

Використовуючи алгоритм, котрий ми розглянемо пізніше, ми знаходимо, що визначальною множиною графа  $G$  буде  $S = \{1, 2, 3\}$ .

**Означення 5.** **Визначальним числом** графа  $G$  є найменше число  $r$ , для якого  $G$  має визначальну множину  $S$  розміру  $r$ . Позначають як  $Det(G)$ .

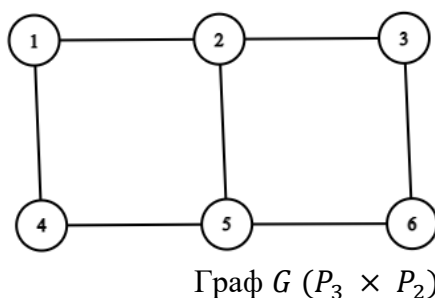
**Твердження 1.** Для повного графа  $K_n$  (де  $n$  – кількість вершин),  $Det(K_n) = n - 1$ .

*Доведення.* Якщо  $n = 2$ , то граф утворює визначальну множину з розміром 1. При видаленні ребра граф буде складатися лише з ізольованих вершин.

Якщо  $n > 2$ , то будь-яке вершина повного графа належить хоча-б в одному з автоморфізмів, але завжди залишається одна вершина, котра не використовується у перестановці, тому з **означення 4** можна сказати, що розмір визначальної множини графа з кількістю вершин  $n$ ,  $Det(K_n) = n - 1$ .

**Твердження 2.** Для довільного графа  $P_k$ ,  $Det(P_k) = 1$ .

*Доведення.* Нехай є два графа-ланцюга  $P_3$  та  $P_2$ . Розглянемо їх декартовий добуток  $P_3 \times P_2$ , отримаємо граф  $G$ . Знайдемо його визначальне число.



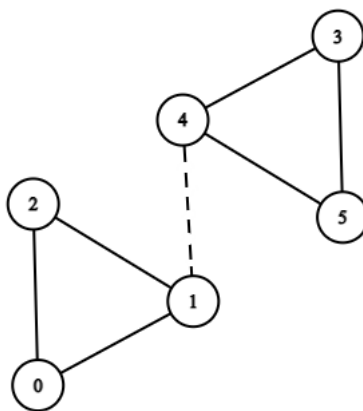
У даного графа 6 вершин та 7 ребер. З **означення 2** знаходимо варіанти перестановки множини вершин даного графа. Отримали  $|Aut(G)| = 4$ , де групою автоморфізмів є  $[\varepsilon, (1\ 3)(4\ 6), (1\ 4)(2\ 5)(3\ 6), (1\ 6)(2\ 5)(3\ 4)]$ .

Використовуючи **означення 5** можемо побачити, що найменшими визначальними множинами може бути одна з вершин підмножини  $S = [\{1\}; \{4\}; \{3\}; \{6\}]$ .

Найменший розмір визначальної множини графа  $G$  буде рівна 1,  $Det(G) = 1$ .

**Означення 6. Зв'язний граф** є графом, що містить хоча б один маршрут між будь-якими двома його вершинами.

**Приклад 5.** Розглянемо граф  $P$ . Він є зв'язним, але якщо прибрати ребро, визначене пунктирною лінією, то він перестане бути зв'язним



Граф Р

**Твердження 3.** Нехай  $G$ - граф з визначальним числом  $r$  та вершинами  $n$ , тому  $r \geq \log_s |Aut(G)| \geq \log_n |Aut(G)|$ .

*Доведення.* Кожна з  $r$  вершин визначальною множини має не більше  $s$  вершин, для яких вона може бути автоморфізмом. Кожен автоморфізм однозначно визначається тим, куди ці вершини відправляються. Таким чином існує не більше  $s^r$  автоморфізмів,  $|Aut(G)| \leq s^r \Rightarrow r \geq \log_s |Aut(G)|$ .

Дане твердження дає краще оцінку при ситуації, коли максимальний розмір орбіти малий (порівняно з загальною кількістю вершин).

**Твердження 4. [1]** Нехай  $G$ - граф з визначальним числом  $r$  та вершинами  $n$ , тому  $\frac{n!}{|Aut(G)|} \geq (n - r)!$

*Доведення.* Оскільки кожне автомофне відображення визначальної множини має бути унікальним, й точно існує  $\frac{n!}{(n-r)!}$  різних впорядкованих  $r$  – підмножин  $V(G)$  є не більше, аніж  $\frac{n!}{(n-r)!}$  різних автоморфізмів графа  $G$ .

Порівняно з **твердженням 1**, дане твердження дає кращу оцінку при транзитивності графа.

**Означення 7.** Граф є **транзитивним**, якщо для будь-яких двох вершин існує автоморфізм.

## Знаходження визначальних множин

Існує простий спосіб отримання визначального набору, який полягає саме у знаходженні набору з тривіальним стабілізатором.

**Означення 8.** Для кожного підграфа  $S \subseteq V(G)$ ,  $Stab(S) = \{g \in Aut(G) \mid g(v) = v, \forall v \in S\}$ . Це **точковий стабілізатор** підграфа  $S$ .

**Твердження 5.** Нехай  $S$  є підмножиною графа  $G$ . Тоді  $S$  є визначальною множиною графа  $G$  й тільки тоді, коли  $Stab(S) = \{id\}$ .

*Доведення.* Якщо  $S$  є визначальною множиною тоді, коли  $g \in Aut(G)$  фіксує кожен  $s \in S$ ,  $g = id$ . Таким чином  $Stab(S) = \{id\}$ . Крім того, якщо  $Stab(S) = \{id\}$  та  $g, h \in Aut(G)$ , так, що  $g(s) = h(s)$  для всіх  $s \in S$ .

Однак, якщо ми хочемо знайти визначальний набір для визначення групи автоморфізмів без використання стабілізатора потрібен інший підхід. Один з таких методів полягає у знаходженні набору вершин  $S$ , для якого кожна вершина в графі може однозначно ідентифікований за її відстанями від вершини  $S$ . Дану множину називають *еталонним набором* або *роздільним набором*.

**Означення 9.** Нехай  $G$  — зв'язний граф. Нехай  $S = \{s_1, \dots, s_r\}$  є упорядкованою підмножиною вершин  $G$ . Вектор  $S$ -відстаней вершини  $v$  рівна  $(d(v, s_1), \dots, d(v, s_r)) \in Z^r$ , де кожна відстань є довжиною найкоротшого шляху між двома вершинами. Даний набір  $S$  називають множиною визначальним відстаней для графа  $G$ , якщо вектори  $S$ -відстаней  $G$  є різними.

Припустимо, що ми маємо множину  $S$ , яка визначає відстань, автоморфізм  $\sigma$  та довільну вершину  $v$ . Оскільки  $v$  визначається своїми відстанями до вершин  $S$  і  $\sigma$  зберігає відстань, то  $\sigma(v)$  також визначається своїми відстанями до  $\sigma(S)$ . Таким чином,  $S$  є визначальною множиною для  $G$ . З цього виходить наступне.

**Твердження 6.** Нехай  $S$  – підмножина вершин зв'язного графа  $G$ . Якщо  $S$  є множиною, котра визначає відстань до  $G$ , то вона є визначальною множиною для  $G$ .

Якщо граф незв'язний, то ми можемо знайти набір, що визначає відстань для кожної компоненти. Їх об'єднання є визначальним набором для графа, тільки якщо граф не містить кілька ізольованих вершин. У даному випадку ми додаємо усі ізольовані вершини, окрім однієї до набору щоб зробити його визначальним набором.

Додатково ми можемо дізнатись про визначальні множини за допомогою орбіт вершин та їх індукованих підграфів.

**Означення 10. Індукований підграф** графа  $G$  – це граф  $H$ , який утворюється з підмножини вершин графа і всіх ребер  $G$ , що з'єднують пари вершин у цій підмножині.

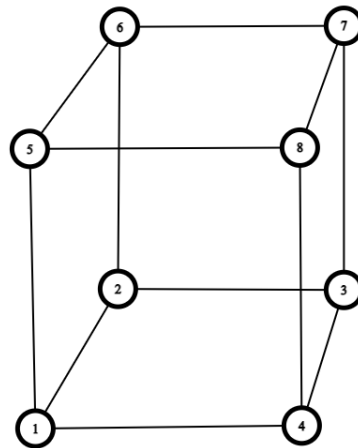
**Твердження 7. [1]** Нехай  $H_1, \dots, H_k$  – асоційовані індуковані графи,  $O_1, \dots, O_k$  – орбіти вершин графа  $G$ ,  $S_1, \dots, S_k$  – визначальні множини для  $H_1, \dots, H_k$ . Тоді  $S = S_1 \cup \dots \cup S_k$  є визначальною множиною для  $G$ .

*Доведення.* Оскільки кожен  $O_i$  є інваріантним під дією  $Aut(G)$ , ми можемо розглядати фактор-групу  $Aut(G)/Stab(O_i)$  як підгрупу  $Aut(H_i)$ . Оскільки  $S_i$  визначає  $H_i$  під дією цієї фактор-групи. Це означає, що  $Stab(S_i) = Stab(O_i)$  під дією  $Aut(G)$ .

Оскільки дія  $Aut(G)$  є точною, то стабілізатор  $G$  є тривіальним. Тоді  $Stab(S) = Stab(S_1 \cup \dots \cup S_k) = Stab(S_1) \cap \dots \cap Stab(S_k) = Stab(O_1) \cap \dots \cap Stab(O_k) = Stab(V(G)) = \{id\}$ . Таким чином множина  $S$  є визначальним набором для  $G$ .

## Приклади відновлення автоморфізму графа за допомогою визначальної множини

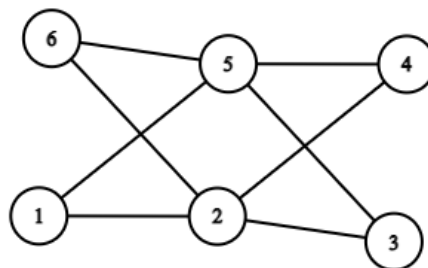
**Приклад 6.** Розглянемо граф-куб  $M$  з 8-ма вершинами. Визначимо визначальний набір для даного графа. Так, як граф є зв'язним то використаємо означення 9 й з нього можна виділити набір  $S = \{1, 2, 3\}$ .



Граф  $M$

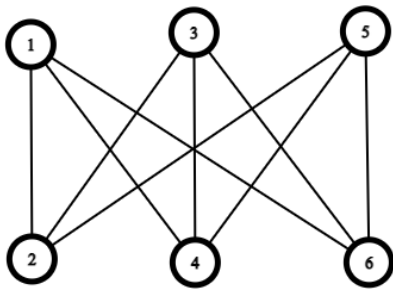
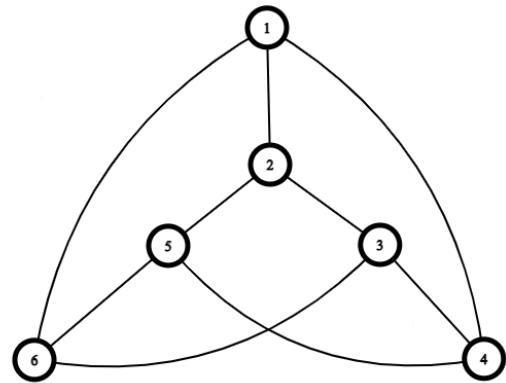
З заданої визначальної множини можна легко знайти один з автоморфізмів графа як  $(1\ 2\ 3\ 4)(5\ 6\ 7\ 8)$ . Взагалом можна визначити, що  $|Aut(M)| = 48$ , вдостовіритись цим ми зможемо за допомогою алгоритму bliss, котрий буде представлений нижче.

**Приклад 7.** Розглянемо граф  $L$  з 6-ма вершинами. Використовуючи означення 10 й твердження 5 визначимо визначальний набір для даного графа. Орбітами даного графа є  $\{(1\ 3\ 4\ 6), (2,5)\}$ , й одразу можна побачити що дана перестановка є одним з автоморфізмів графа, тобто  $(1\ 3\ 4\ 6)(2\ 5)$ . Визначальним набором є  $S = \{1, 2, 4, 5\}$ . Визначимо, що  $|Aut(L)| = 48$ .



Граф  $L$

**Приклад 8.** Розглянемо два графи, а саме  $K_{3,3}$  (граф з 6 вершинами у двох множинах по 3 вершини, з одним ребром між кожною парою вершин з протилежних вершин, такі графи називають повними дводольними) та  $ML_3$ , що є драбиною Мебіуса, отриманий з кругових сходів (circular ladder graph  $CL_n$ ) шляхом видалення з нього двох паралельних вигнутих ребер.

Граф  $K_{3,3}$ Граф  $ML_3$ 

Можна побачити що зовнішньо вони різні, але вони є ізоморфними й можна зрозуміти що вони будуть мати одну й теж групу автоморфізмів. Дані графи містять по 9 вершин, використовуючи **означення 9** можемо визначити групу  $S = \{1, 2, 3, 4\}$ , одразу можна зазначити один з автоморфізмів  $(1, 2, 3, 4)(5, 6)$ . Знайдемо, що  $|Aut(K_{3,3})| = |Aut(ML_3)| = 72$ .

Даний приклад демонструє нам, що зовні різні графи, але котрі є ізоморфними, мають однакову групу автоморфізмів.

Використання визначальних множин є корисним у знаходження автоморфізмів графа. Приклади показують що цей метод є ефективним у розв'язанні задач з теорії графів. Однак знаходити автоморфізми великих графів за даним методом буде складно, тому останнім часом все більше використовуються алгоритми пошуку, котрі будуються на визначальних множинах.

## **Використання при роботі з даними**

Одним з прикладів є стиснення даних. Якщо ми можемо зберегти лише визначальну множину та використовувати її для відновлення повної структури графа, то ми можемо значно скоротити кількість даних, які необхідно передавати або зберігати. Це може бути корисним у розподілених системах, де обсяг передаваних даних є критичним фактором.

Стиснення даних - це процес зменшення обсягу даних, які необхідно передавати або зберігати. Це може бути досягнуто шляхом використання різних алгоритмів стиснення, які дозволяють зберегти інформацію у більш компактному вигляді.

Одним з способів застосування визначальної множини для стиснення даних є збереження лише визначальної множини графа. Якщо ми можемо використовувати визначальну множину для відновлення повної структури графа, то ми можемо значно скоротити кількість даних, які необхідно передавати або зберігати.

Наприклад, уявімо, що ми маємо граф з тисячею вершин та десятками тисяч ребер. Якщо ми збережемо усю цю інформацію, то нам знадобиться багато місця для збереження даних. Однак, якщо ми можемо знайти визначальну множину для цього графа, то ми можемо зберегти лише цю множину та використовувати її для відновлення повної структури графа. Це може значно скоротити кількість даних, які необхідно передавати або зберігати.

Таким чином використання визначальної множини для стиснення даних дає нам можливість оптимізувати передачу та збереження даних у розподілених системах.

### **Використання визначальних множин у GraphZip**

GraphZip – алгоритм стиснення графів, котрий ґрунтується на використанні словників для знаходження повторюваних вузлів й ребер у графі. Дані елементи кодуються за допомогою індексів у словнику, що допомагає зменшити розмір вихідного графа.

Графи стискаються по мірі їх надходження, що дозволяє зберегти тимчасову й просторову локальність графових даних. Даний алгоритм також використовує принцип мінімальної довжини опису для визначення оптимальної ступені стиснення графа.

В основі GraphZip лежить алгоритм надходження визначальних множин графів. Після їх знаходження він будує індекс словника для кожного підграфа. Індекс словника представляє набір пар вузлів/ребер й індекс, де вузол/ребро – елементи підграфа, а індекс – відповідний індекс у словнику. Як було зазначено, алгоритм використовує словники вузлів й ребер, де кожен з них містить інформацію про повторювані елементи.

В процесі кодування GraphZip визначає словник, який найкраще буде використовувати для стиснення графів, що підвищує ступінь стиснення для кожного підграфа. В результаті отримується файл, котрий містить інформацію про визначальні множини, котрі були використані при стисненні, а також індекс вершин, котрі входять у визначальні множини. Дана інформація дає можливість відновити граф з стисненого файлу за допомогою GraphZip.

Даний алгоритм є одним з найбільш ефективним алгоритмів стиснення графів. Він показує велику ступінь стиснення при невеликих обчислювальних затратах. Крім того, він легко масштабується й може оброблювати графи великого розміру завдяки використанню паралельних обчислень. Однак, у нього є свої недоліки. Його ефективність знижується при графах з високою щільністю ребер, тобто мають великі визначальні множини.

Нижче представлений приклад компресування графів за допомогою алгоритму.



а)            б)

Приклад використання GraphZip з даними (а) до використання стиснення й (б) після стиснення.

GraphZip представляє собою потужним інструментом для стиснення графів, котрий може бути використаний в різних областях, пов'язаних з аналізом графів. Його здатність оброблювати графи великого розміру й можливість масштабування робить його особливо корисним для роботи з великими наборами даних, наприклад, у області соціальних мереж й біоінформатики.

## **Використання при роботі з алгоритмами знаходження автоморфізмів**

Багато мов програмування, такі як Java, Python, C++ та інші, містять в собі бібліотеки з алгоритмами пошуку автоморфізму графів, основаних на визначальних множинах. В основі вони використовують алгоритми nauty або bliss.

### **Використання визначальних множин у Bliss**

Одним з найрозповсюдженішим алгоритмом є Bliss, котрий ще з кінця 20-го століття допомагає з вирішенням даної задачі. Він використовує два основних компоненти для знаходження автоморфізмів – алгоритми пошуку визначальних множин й алгоритм перебору з використанням backtrack.

Алгоритм пошуку визначальних множин використовується для знаходження мінімальної кількості визначальних множин графа, котрим буде таким набором вершин, що для будь-яких двох вершин з цієї множини, всі несуміжні вершини мають однакове оточення з ними. Це дає можливість зменшити кількість можливих перестановок вершин графа при пошуку автоморфізмів, так як люба перестановка вершин, зберігаючи визначальну множину, також буде автоморфізмом графу.

Алгоритм перебору з використанням backtrack використовується для знаходження всіх можливих автоморфізмів графа, й ґрунтується на переборі всіх можливих перестановок вершин, зберігаючих визначальну множину. Він може бути оптимізований, якщо використовувати інформацію, отриману в процесі пошуку визначальної множини.

Алгоритм Bliss починає роботу з побудови дерева пошуку всіх можливих перестановок графа. Кожен вузол дерева є частковим автоморфізмом, котрий може бути розширений до повного автоморфізму. На кожному рівні дерева алгоритм обирає нову перестановку вершин й перевіряє, чи зберігає вона структуру графа. Якщо так, то переходить до

наступного рівня дерева, якщо ні, то ця гілка відкидається, бо не може бути автоморфізмом графа.

Одним з ключових кроків алгоритму є знаходження визначної множини. Її побудова допомагає скоротити кількість перестановок, котрі потрібно перевірити на кожному рівні дерева.

Наприклад, бібліотека SageMath у мові програмування Python використовує алгоритм Bliss як один з методів визначення груп автоморфізму графа.

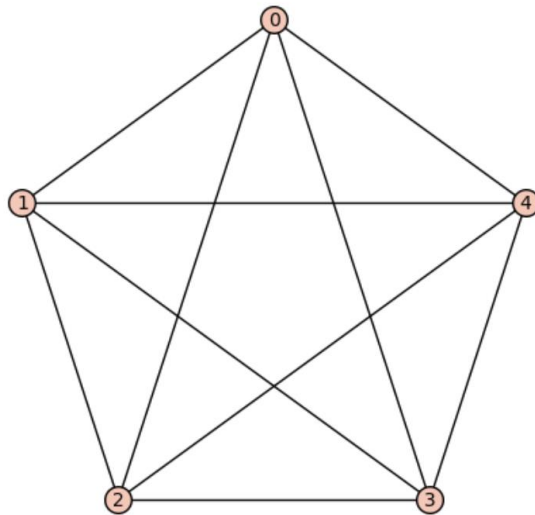
З його допомогою можна знайти всі автоморфізми будь-якого графу.

У даному прикладі буде використаний повний граф з 5-ма вершинами.

```
g = graphs.CompleteGraph(5)
group = g.automorphism_group(edge_labels=True, algorithm='bliss').list()
print("Перелік автоморфізмів: ", group)
print("Довжина групи: ", len(group))
plot(g)
```

```
Перелік автоморфізмів: [(), (0,1), (0,2,1), (0,3,2,1), (0,4,3,2,1), (1,2), (0,
1,2), (0,2), (0,3,2), (0,4,3,2), (1,3,2), (0,1,3,2), (0,2)(1,3), (0,3,1,2), (0,
4,3,1,2), (1,4,3,2), (0,1,4,3,2), (0,2)(1,4,3), (0,3,1,4,2), (0,4,2)(1,3), (2,
3), (0,1)(2,3), (0,2,3,1), (0,3,1), (0,4,3,1), (1,2,3), (0,1,2,3), (0,2,3), (0,
3), (0,4,3), (1,3), (0,1,3), (0,2,1,3), (0,3)(1,2), (0,4,3)(1,2), (1,4,3), (0,
1,4,3), (0,2,1,4,3), (0,3)(1,4,2), (0,4,2,1,3), (2,4,3), (0,1)(2,4,3), (0,2,4,
3,1), (0,3,1)(2,4), (0,4,2,3,1), (1,2,4,3), (0,1,2,4,3), (0,2,4,3), (0,3)(2,4),
(0,4,2,3), (1,3)(2,4), (0,1,3)(2,4), (0,2,4,1,3), (0,3)(1,2,4), (0,4,1,2,3),
(1,4,2,3), (0,1,4,2,3), (0,2,3)(1,4), (0,3)(1,4), (0,4,1,3), (3,4), (0,1)(3,4),
(0,2,1)(3,4), (0,3,4,2,1), (0,4,2,1), (1,2)(3,4), (0,1,2)(3,4), (0,2)(3,4), (0,
3,4,2), (0,4,2), (1,3,4,2), (0,1,3,4,2), (0,2)(1,3,4), (0,3,4,1,2), (0,4,1,2),
(1,4,2), (0,1,4,2), (0,2)(1,4), (0,3,2)(1,4), (0,4,1,3,2), (2,3,4), (0,1)(2,3,
4), (0,2,3,4,1), (0,3,4,1), (0,4,1), (1,2,3,4), (0,1,2,3,4), (0,2,3,4), (0,3,
4), (0,4), (1,3,4), (0,1,3,4), (0,2,1,3,4), (0,3,4)(1,2), (0,4)(1,2), (1,4),
(0,1,4), (0,2,1,4), (0,3,2,1,4), (0,4)(1,3,2), (2,4), (0,1)(2,4), (0,2,4,1),
(0,3,2,4,1), (0,4,1)(2,3), (1,2,4), (0,1,2,4), (0,2,4), (0,3,2,4), (0,4)(2,3),
(1,3,2,4), (0,1,3,2,4), (0,2,4)(1,3), (0,3,1,2,4), (0,4)(1,2,3), (1,4)(2,3),
(0,1,4)(2,3), (0,2,3,1,4), (0,3,1,4), (0,4)(1,3)]
Довжина групи: 120
```

Приклад коду для знаходження автоморфізмів повного графа  $K(5)$

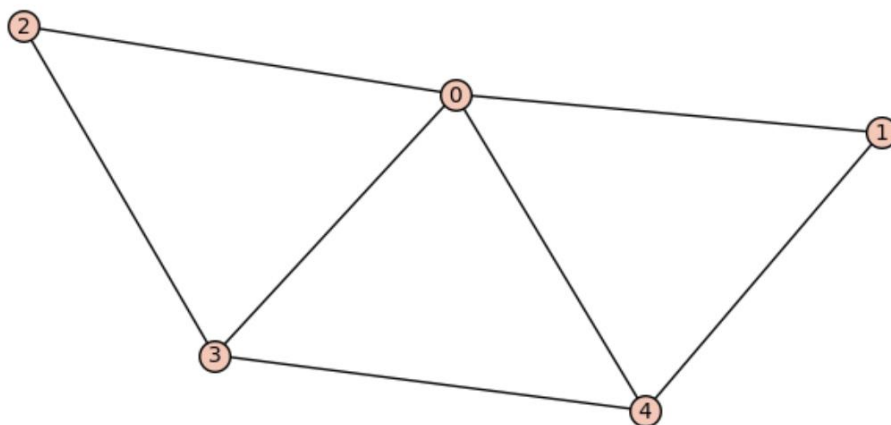
Повний граф  $K(5)$ 

Можна побачити, що  $|Aut(K(5))| = 120$ , й зрозуміло, що визначати перелік графів з більшою кількістю вершин вручну вийде набагато довше, аніж використовувати алгоритм

Нижче наведено приклад використання вже з довільним графом з 5-ма вершинами.

```
edges = [(0,1), (0,2), (0,3), (0,4), (3,0), (3,2), (4,0), (4,3), (4,1)]
h = Graph(edges, format='list_of_edges')
group = h.automorphism_group(edge_labels=True, algorithm='bliss').list()
print("Перелік автоморфізмів: ", group)
print("Довжина групи: ", len(group))
plot(h)
```

```
Перелік автоморфізмів: [(), (1,2)(3,4)]
Довжина групи: 2
```

Довільний граф  $H$  з 5-ма вершинами

Даний граф має  $|Aut(H)| = 2$ .

Алгоритм Bliss є одним з найефективнішим з знаходження автоморфізмів графу. Найбільш ефективним він є для графів великого розміру, но менш ефективний з графами малого розміру.

## Висновок

У даній роботі було визначено твердження для графа  $P_k$  (граф-ланцюг), котре доводить, що їх визначальне число буде рівно 1. Для цього було використано декартів добуток двох графів-ланцюгів та пошук їх визначальних множин.

Показані приклади визначення автоморфізмів графу, а саме для графа-куба, довільного та повного дводольного. Визначено, що визначальні множини є корисними у знаходженні автоморфізмів графа й допомагають швидше їх визначити.

Було розглянуто використання визначальної множини у різних галузях життя, а саме при роботі з даними та різних алгоритмах. За допомогою GraphZip, котрий представляє собою потужний інструмент для стиснення графів, котрий може бути використаний в різних областях, пов'язаних з аналізом графів. При огляді мови програмування Python й бібліотеки SageMath було визначено алгоритм, котрий використовує визначальні множини для відновлення автоморфізму графа, а саме bliss. Він допомагає швидко визначати групи автоморфізму, тим самим допомагаючи швидко працювати з графами.

## Список використаних джерел

- [1] Boutin D. Identifying Graph Automorphisms Using Determining Sets [Електронна стаття] / Debra L. Boutin // Vol. 13. of *The Electronic Journal of Combinatorics* // Hamilton College // R78. – 2006. – Режим доступу до ресурсу: <https://www.combinatorics.org/ojs/index.php/eljc/article/view/v13i1r78/pdf>.
- [2] Graph Editor [Електронний ресурс] – Режим доступу до ресурсу: [https://csacademy.com/app/graph\\_editor/](https://csacademy.com/app/graph_editor/).
- [3] Кузьменко І. М. Теорія графів [Електронна книга] / І. М. Кузьменко // навч. посіб. для здобувачів ступеня бакалавра за освітньою програмою «Комп'ютерний моніторинг та геометричне моделювання процесів і систем» спеціальності 122 «Комп'ютерні науки» // КПІ ім. Ігоря Сікорського // 71с. – 2020. – Режим доступу до ресурсу: [https://ela.kpi.ua/bitstream/123456789/35854/1/Teoriia\\_hrafiiv.pdf](https://ela.kpi.ua/bitstream/123456789/35854/1/Teoriia_hrafiiv.pdf).
- [4] The bliss tool [Електронний ресурс] – Режим доступу до ресурсу: <https://users.aalto.fi/~tjunttil/bliss/index.html>.
- [5] Qian Y. GraphZip: A Fast and Automatic Compression Method for Spatial Data Clustering [Електронна стаття] / Y. Qian, K. Zhang // *SAC '04: Proceedings of the 2004 ACM symposium on Applied computing* // The University of Texas at Dallas // 571-575с. – 2004. – Режим доступу до ресурсу: <https://personal.utdallas.edu/~kzhang/Publications/SAC04.pdf>.