Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ”

Кафедра математики факультету інформатики

Курсова робота на тему:
Вкладання та пакування графів

Керівник курсової роботи:
к. ф.-м. н. *Козеренко С.О.*
(прізвище та ініціали)

_____
(підпис)
“____”_____ 2024 р.

Виконала студентка
3-го року навчання спеціальності
122 “Комп’ютерні науки’
Бондар Катерина Володимирівна
(ПІБ)

Київ – 2024

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ"

Кафедра математики факультету інформатики

ЗАТВЕРДЖУЮ
Зав. кафедри математики, доц., к.ф-м.н.
_____ Р.К. Чорней
(підпис)
"____"_____ 2024 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на курсову роботу
студентці 3-го курсу факультету інформатики
Бондар Катерині Володимирівні

Тема: Вкладання та пакування графів.
Вихідні дані: Алгоритм для $n-2$, та необхідні елементи для $n-1$ алгоритму.
Зміст ТЧ до курсової роботи:

Анотація
Вступ
1 Основні означення
2 Знаходження перестановки для $(\leq n-2)$-реберного графа
2.1 Теорема
2.2 Алгоритм, реалізований на Haskell
3. Приклад антиалгоритмічного доведення існування перестановки
4. Знаходження перестановки для $n-1$-реберного графа
4.1 Теореми
4.2 Як можна реалізувати
Висновки
Література

Дата видачі "____"_____ 2024 р. Керівник _____
(підпис)
Завдання отримав _____
(підпис)

Тема: Вкладання та пакування графів.

Календарний план виконання роботи:

Номер	Назва етапу курсової	Термін виконання етапу	Примітка
1.	Отримання теми курсової роботи.	27.09.2023	
2.	Ознайомлення з темою курсової.	10.11.2023	
3.	Розробка плану та структури роботи.	10.11.2023	
4.	Робота з науковою літературою, опис основних означень теорії графів.	5.02.2024	
5.	Дослідження перестановок для $n-2$ і $n-1$-реберних графів.	26.04.2024	
6.	Робота над текстовим оформленням результатів.	9.05.2024	
7.	Попередній аналіз курсової. Виправлення помилок.	10.05.2024	
8.	Захист курсової. роботи.	20-22.05.2024	

Зміст

Анотація

Вкладання графа в своє доповнення це ізоморфізм(надалі часто використовуватиметься термін перестановка) $\sigma : V(G) \to V(G)$ такий, що $\forall u, v \in V(G)$ і $\{u, v\} \in E(G), \{\sigma(u), \sigma(v)\} \in \overline{G}$ де G – неорієнтований граф, $V(G)$ – множина вершин графа G, $E(G)$ – множина ребер графа G, тобто невпорядкованих пар вершин.

Мета роботи полягає в дослідженні таких перестановок і їх закономірностей, що можуть бути алгоритмічно виражені, та реалізації алгоритму отримання такого ізоморфізму.

Ключові слова: граф, вкладання, доповнення, перестановка, алгоритм.

Вступ

Тема вкладення чи пакування графів в своє доповнення чи в інші графи активно досліджується математиками.

Існує досить багато матеріалів, пов'язаних з вкладанням чи пакуванням графів - для прикладу, M. Wozniak має більше 7-ми статей на цю тематику.

У цій роботі поставлено за мету відшукати взаємозв'язки чи доведення існування вкладеннь, які можна алгоритмізувати, та спробувати це зробити.

Тут розглядаються та досліджуються ізоморфізми, які вкладають графи в свої доповнення на $n-2$ та $n-1$ ребрах, викладаються основні поняття, реалізовується алгоритм на один з вище згаданих типів такого ізоморфізму на мові програмування Haskell, видозмінений для спрощення і скорочення зайвого, та розповідається про можливість реалізувати другий з вище згаданих типів на базі 2-ох теорем, які в свою чергу базуються на перестановках $(n, n-2)$-графів та дерев, спробувавши самостійно знайти алгоритм перестановки для нетривіальних дерев, які не є зірками.

1 Основні означення

Означення 1.1. *Неорієнтований граф* – це пара $G = (V, E)$, де

$$V(G) \text{ – множина } \textit{вершин},$$
$$E(G) \subset V(G)^{(2)} = \{\{u, v\} |\ u, v \in V(G)\} \text{ – множина } \textit{ребер}.$$

Надалі замість позначення $\{u, v\}$, використовуватимемо uv.

Означення 1.2. *Степінь вершини* $d(u)$ – це ціле число від 0 до $|V(G)|-1$, яке позначає кількість ребер, які містять цю вершину або кількість вершин, з якою вона зв'язана.

Означення 1.3. *Простий граф* – це граф без кратних ребер та петель.

Означення 1.4. *Зв'язний граф* – це граф, між будь-якою парою вершин якого існує шлях, який їх сполучає.

Означення 1.5. *Компонента зв'язності* графа – це його максимальний (за включенням) зв'язний підграф.

Означення 1.6. *Підграф* – це зв'язний граф (позначимо як G^*) який має кількість вершин $|V(G^*)| = |V(G)| - n, 0 \geq n \geq |V(G)|$та містить всі ребра, які наявні в G між вершинами, наявними в G^*. Позначається як $G^* \subset G$

Означення 1.7. *Шлях* – між парою вершин $x, y \in V(G)$ це послідовність вершин $v_0, v_1, ..., v_n$, де $x = v_0$ це початок шляху, а $y = v_n$ – кінець шляху, $\forall i = 0, n-1 : v_i v_{i+1} \in E(G)$.

Означення 1.8. *Цикл* – це шлях, у якого початок та кінець збігаються.

Означення 1.9. *Повний граф* K_n – граф, у якому кожні дві вершини з'єднані ребром, тобто $E = V^{(2)}$.

Означення 1.10. *Дерево* – це зв'язний граф, який має кількість ребер $|E(G)| = |V(G)| - 1$.

Також дерево не містить циклів, як своїх підграфів

Означення 1.11. *Нетривіальне дерево* – це дерево, яке не є K_1 чи K_2 .

Означення 1.12. *Двочастковий граф* – це граф, множину вершин якого можна розбити на дві підмножини (частки) так, щоб жодні дві вершини з однієї підмножини не були суміжними, тобто $V = V_1 \sqcup V_2, E \subset \{\{a, b\} | a \in V_1, b \in V_2\}$.

Означення 1.13. *Зірка* $K_{1,n}$ – це дерево, яке є повним двочастковий графом з єдиним внутрішнім вузлом і n листками.

Означення 1.14. *Ланцюг* P_n – це дерево, яке складається тільки з 2-х чи менше вершин, степеня 1, та $(n-2)$-х вершин степеня 2.

Означення 1.15. *Перестановка* – це ізоморфізм $\sigma : V(G) \to V(G)$ з G в $\overline{G}$(або в H, якщо розглядаються різні графи) така, що $\forall u, v \in V(G), uv \in E(G)$ – $\sigma(u)\sigma(v) \in E(\overline{G})$.

2 Знаходження перестановки для $(\leq n-2)$-реберного графа

2.1 Доведення існування перестановки вершин для графа на $(n-2)$-х і менше ребрах, яка вкладає граф в власне доповнення

Теорема 2.1. *Якщо* $|E(G)|, |E(H)| < n-2$, *коли* $n = |V(H)| = |V(G)|$, *тоді* G *і* H *взаємно розміщувані.*

Доведення. Використовується індукція по n. Теорема пряма для малих n, скажімо $n < 5$.

Зафіксуємо G, H на n вершинах і припустимо, що теорема виконується для будь-якої меншої кількості вершин. Достатньо просто показати, що G і H взаємно розміщувані за додатковою гіпотезою $|E(G)| = |E(H)| = n-2$. Тому, якщо задано будь-які G_1, H_1, що задовольняють гіпотези теореми, ми довільно поширюємо їх і на G, H з $n-2$ ребрами та бієкція σ для G, H також працює для G_1, H_1 . Під деревом у G чи H будемо розуміти ізольовану компоненту з t вершинами та $t-1$ ребрами, з деяким натуральним t. Це включає такі випадки, як ізольована точка$(t=1)$, і ізольоване ребро$(t=2)$. Кожен t-вершинний компонент, який не є деревом, повинен мати принаймні t ребер. Тоді елементарно порахувавши, G і H мають містити принаймні 2 дерева. Доведення ділиться на декілька випадків, де випадок a містить основну ідею.

Випадок a: Ні G, ні H не містять ізольованих точок чи ребер. В дереві з ≥ 3 вершин ми завжди можемо знайти висячу точку a степеня $d(a) = 1$, приєднану до вершини β степеня $d(\beta) > 1$.

Використаємо позначення $a \to \beta$ для позначення того, що вершина a з'єднана лише з вершиною β. У графі G в двох деревах знаходимо вершини 1, 2, 3, 4 такі, що $1 \to 2$, $3 \to 4$, 2 і 4 мають степені $d(2), d(4) > 1$ і $\{2,3\} \notin E(G)$ (тому, що вони в різних компонентах зв'язності). В H знаходимо вершини a, b, c, d такі, що $a \to b$, $c \to d$ з аналогічними умовами.

Нехай $G^* = G(V(G) - \{1,2,3,4\})$, $H^* = H(V(H) - \{a,b,c,d\})$. Тоді $|E(G^*)| \leq |E(G)| - 4 = |V(G^*)| - 2$ і $|E(H^*)| \leq |E(H)| - 4 = |V(H^*)| - 2$, тому, що в графі G вершини $\{1,2,3,4\}$ мають прийнаймні 4 ребра. Те саме і для H. Тоді знаходимо перестановку σ^*:

$$\sigma^* : V(G^*) \to V(H^*)$$

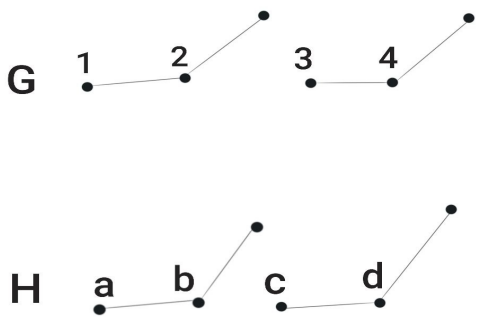

Рис. 1: Випадок а

Розширюємо σ^* до σ:

$$\sigma : V(G) \to V(H),$$

яка визначена як

$$\begin{gathered}\sigma(x) = \sigma^*(x),\ x \neq 1,2,3,4,\\ \sigma(1) = b, \sigma(3) = d,\\ \sigma(2) = c, \sigma(4) = a.\end{gathered}$$

Стверджуємо, що σ є розміщенням. Припустимо, $\{i,j\} \in E(G)$. Якщо $i,j \notin \{l,2,3,4\}$, то $\{\sigma_i, \sigma_j\} \notin E(H)$, оскільки σ^* є розміщенням. Припустимо, $i = 1$. Тоді $\{i,j\} \in E(G)$, тільки якщо $j = 2$, але $\{\sigma_1, \sigma_2\} = \{b,c\} \notin E(H)$. Припустимо, $i = 2$. Тоді, якщо $E(H)$ містить $\{\sigma_i, \sigma_j\} = \{\sigma_c, \sigma_j\}$, ми повинні мати $\sigma_j = d$, тому $j = 3$, але $\{2,3\} \notin E(G)$. Аналогічними є випадки для $i = 3,4$.

Випадок б: У G або H є компонент розміру 2. Ми можемо припустити, що $\{1,2\}$ є двоточковою компонентою в G. Решта $n-2$ вершин мають $n-3$ ребер, тому ми знаходимо точку $3 \in V(G)$ ступеня $d(3) \geq 2$.

Випадок б1: H не має ізольованої точки.

Візьмемо компонент, який є деревом та знайдемо $a,b \in V(H), a \to b$. Виберемо $c \in V(H), \{b,c\} \notin E(H)$, максимального степеня. (Якісь $c, b, c \notin E(H)$ існують, оскільки є лише $n-2$ ребер.) Якщо $d(b) = 1$, то $d(c) \geq 2$. В іншому випадку ми знайдемо $d(c) > 1$, якщо всі ребра H не містять b. У будь-якому випадку $\{a,b,c\}$ мають принаймні 3 ребра.

Тепер знаходимо розміщення

$$\sigma^* : V(G) - \{1,2,3\} \to V(H) - \{a,b,c\}$$

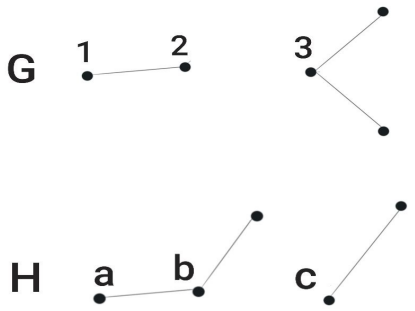

Рис. 2: Випадок б1

і розширюємо до σ, за допомогою

$$\sigma(1) = b, \sigma(2) = c, \sigma(3) = a.$$

Випадок б2: H має ізольовану точку, скажімо, $a \in V(H)$.

Виберемо $b \in V(H)$ максимального степеня та $c \in V(H)$ максимального степеня так, щоб $c \neq a, \{b, c\} \notin E(H)$.

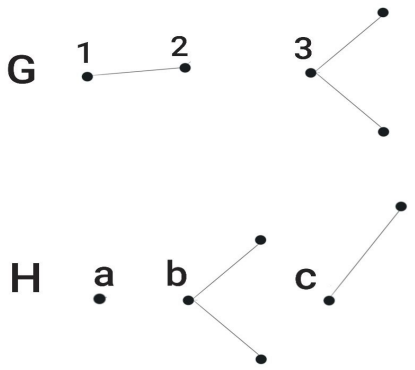

Рис. 3: Випадок б2

Підвипадок б21: якщо такого c не існує, H має бути об'єднанням ізольованої точки a і зірки з центром b степеня $d(b) = n - 2$.

У цьому випадку визначимо $\sigma(1) = b, \sigma(2) = a$, а решту довільно.

Продовження б2: Просто порахувавши, отримуємо $d(b) > 2$ і, виключаючи випадок б21, $d(c) > 1$. Знаходимо перестановку:

$$\sigma^* : V(G) - \{1, 2, 3\} \to V(H) - \{a, b, c\}$$

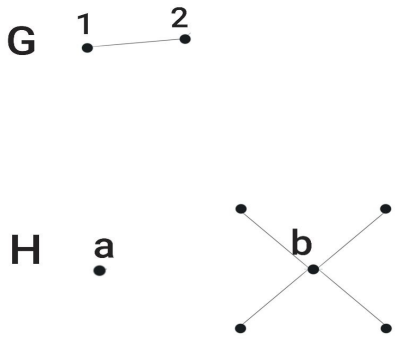

Рис. 4: Випадок б21

і розширюємо її до σ, додавши:

$$\sigma(1) = b, \sigma(2) = c, \sigma(3) = a.$$

Випадок в: немає ізольованих ребер, але є ізольовані точки.

Припустимо, $1 \in V(G)$ є ізольованою точкою. Виберемо $2 \in V(G)$ з $d(2) > 2$.

Підвипадок в1: H має ізольовану точку a.

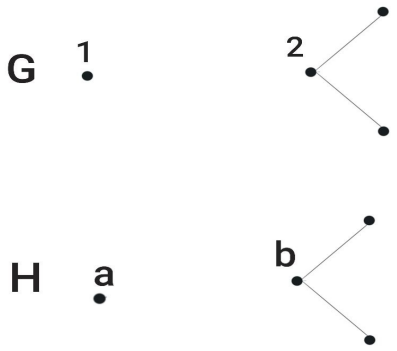

Рис. 5: Випадок в1

Виберемо $b \in V(H), d(b) > 2$. Знаходимо розміщення

$$\sigma^* : V(G) - \{I, 2\} \to V(H) - \{a, b\}$$

і розширимо його до σ, додавши

$$\sigma(1) = b, \sigma(2) = b.$$

Випадок в2: H не має ізольованих точок.

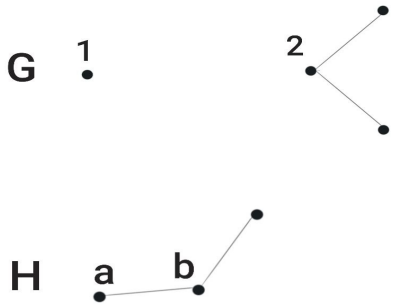

Рис. 6: Випадок в2

Якщо H має двовершинні компоненти, ми використовуємо випадок б. Інакше ми знайдемо компоненту із точками $a, b \in V(H), a \to b, d(b) > 2$.

Знаходимо розміщення:

$$\sigma^* : V(G) - \{I, 2\} + V(H) - \{a, b\}$$

і розширимо його, додавши

$$\sigma(1) = b, \sigma(2) = a.$$

□

Далі розберемо сам алгоритм:

2.2 Алгоритм для знаходження перестановки вершин для графа на $(n-2)$-х і менше ребрах, яка вкладає граф в власне доповнення, на базі теореми 2.1

```
type Graph  = [[Int]]
type Placement = [(Int, Int)]

nodes :: Graph -> [Int]
nodes g = [0..(length g - 1)]

edges :: Graph -> [(Int,Int)]
edges g = [(x,y) | x<-nodes g, y <- g!!x]
```

```
9
10 edgeIn :: Graph -> (Int, Int) -> Bool
11 edgeIn g (x,y) = elem y (g!!x)
```

Для початку потрібно зазначити, як задавався граф: списком зі списків сусідів вершин, номери вершин відповідали їх порядку в списку

В даному фрагменті показані функції, якими можна знайти список вершин графа, список ребер і перевірити чи належить ребро графу.

Також тут визначено яким чином задавалась перестановка: списком пар номерів вершин, де перший елемент пари відповідає оригінальній вершині графа, а другий - її проекції в доповненні.

```
1 a :: (Graph,Placement) -> [[Int]] -> (Graph,Placement)
2 a (gr, p) el = let points = take 2 (head $ filterPs (getPoints gr (el!!2))
     ) in (gr, (act ((snd $ head points):(fst $ head points):(snd $ last
     points):(fst $ last points):[]))++p)
3
4 getPoints :: Graph -> [Int] -> Placement
5 getPoints gr el = [(head gr!!x, x)|x<-nodes gr, length gr!!x ==1, notElem
     (head gr!!x) el]
6
7 filterPs :: Placement -> [Placement]
8 filterPs p = filter (lengthBigger1) (map (filterP []) (subsequences p))
9
10 lengthBigger1 :: [a]->Bool
11 lengthBigger1 l = length l > 1
12
13 filterP :: [Int] -> Placement -> Placement
14 filterP is (p:ps) = if elem (fst p) is then filterP is ps else p:(filterP
     (fst p : is) ps)
15 filterP _ [] = []
```

В цьому фрагменті наведена функція для випадку a і її допоміжні функції

Почати варто з getPoints - тут повертаємо пари висячих вершин з вершинами, до яких вони при'єднані, якщо ці вершини ще не отримали перестановку. Самі висячі вершини не перевіряються, оскільки кожну ітерацію алгоритма граф очищається і вершини з перестановками будуть представлені порожніми списками, тож умова length==1 їх відсіє.

Далі розглянемо функцію filterPs - в функції *a* вона служить для того, аби перелік пар був не пов'язаний(всі вершини різні).

Яким чином:

- на вхід функції передаємо отриманий в getPoints список пар вершина-висяча вершина,
- утворюємо всі можливі їх комбінації(аби раптом не прогавити список, який при фільтрації не стане меншим за 2 - для прикладу випадок,

коли маємо 2 зірки дуже ілюстративний - тоді ми максимум зможемо отримати список довжини 2 при фільтрації, оскільки списки по типу [(0,1),(0,2),...,(0,n)] стануть одноелементними і лише список прийнаймні з однією парою з іншої зірки зможе бути 2-елементним),

- видаляємо за допомогою функції filterP пари, в яких невисячі вершини повторюються(висячі апріорі повторюватись не можуть, оскільки мають єдине ребро, відповідно відображене єдиною парою) (в цій функції ми просто рекурсивно перебираємо пари вершин, кожну ітерацію додаючи вершину до списку використаних або не додаємо пару до результату, якщо вершина вже в списку),

- та видаляємо списки пар довжиною менше 2-х, фільтруючи предикатом lengthBigger1, оскільки нам потрібні саме 2 пари, непов'язані між собою.

Але, оскільки нам потрібні саме 2 пари, допоміжні функції можна скоротити до однієї, з використанням edgeIn:

```
getPoints gr el = head [[(x,y),(u,v)]|x<-nodes gr, y<- gr!!x,u<-nodes gr,
    v<-gr!!u, length gr!!x ==1, length gr!!u ==1, notElem y el, notElem v
    el, not x==u, not (edgeIn gr (y,v))]
```

В цьому фрагменті просто перебираємо вершини так, аби x і u були висячими, yv не є ребром G, бо перестановка стане недійсною(ребро не міститиметься в доповненні на такій перестановці, хоча має вміститись), вершини не були використані, x і u не одна й та ж вершина, ребра $xv, uy \notin E(G)$, бо потрібні саме непов'язані вершини для перестановки.

І скоротити визначення points до

```
let points = getPoints gr (el!!2)
```

Але чому було прийнято рішення залишити цей код - він згодиться для алгоритму на деревах.

Нарешті функція a: вибираємо 2 пари вершин і переставляємо їх таким чином:

$$\sigma(\text{висяча1}) = \text{вершина1},\ \sigma(\text{вершина1}) = \text{висяча2},$$

$$\sigma(\text{висяча2}) = \text{вершина2},\ \sigma(\text{вершина2}) = \text{висяча1}.$$

Чому саме так - в теоремі було визначено вкладення G в , а ми маємо частковий випадок $H = G$ і легко бачити, що $\{a, b, c, d\}$ - можна вважати

тим самим, що й $\{1,2,3,4\}$ відповідно, що = {висяча1, вершина1, висяча2, вершина2}.

Функція act просто повертає циклічно, по-порядку переставлені вершини(для прикладу для списку [0,1,2] поверне перстановки [(0,1),(1,2),(2,0)]) - її код наявний в наступному фрагменті:

```
b1 :: (Graph,Placement) -> [[Int]] -> (Graph,Placement)
b1 (gr, p) el = let edge = findE gr (el!!2) in
  let m = findMaxP gr ((fst $ edge):(snd $ edge):[]) in  (gr,(act ((fst
    $edge):(snd $ edge):m:[]))++p)

act :: [Int] -> Placement
act [] = []
act l = [(l!!i,l!!(i+1))|i<-[0..(length l -2)]]++[(l!!(length l -1),l!!0)]

findE :: Graph -> [Int] -> (Int,Int)
findE gr el = head [(n,y)|n<-nodes gr, y<-(gr!!n), length (gr!!y) == 1,
    length (gr!!n) == 1, notElem n el]

```

Функція findE шукає висячі ребра(K_2): вони мають мати лише одне ребро і лише один з одним, а також кожна з них не бути вже використаною

b1 циклічно переставляє вершини таким чином:

$$\sigma(\text{ребро1}) = \text{ребро2},\ \sigma(\text{ребро2}) = \text{максимальна},$$

$$\sigma(\text{максимальна}) = \text{ребро1}.$$

Чому саме так, якщо в теоремі брали дерево, а не ребро з H:

Вершина 3 має максимальний степінь і, аби не порушити рівновагу $(n, n-2)$ вона має бути прийнаймні степеня 2 - тоді така перестановка, очевидно, вкладається в $\overline{G}$ і доступна для того, аби проводити подальші ітерації її алгоритму. Але що якщо вершина 3 має степінь 1? Тоді це об'єднання висячих ребер, які легко вкладаються в доповнення, оскільки в такому випадку кількість ребер графа $|E(G)| = n/2, n \geq 3$ і тоді граф при кожній ітерації швидше стане порожнім, ніж порушить баланс ребер навіть якщо ітерувати виключно по б1(але після першої ж ітерації такий граф або закине в б2, або в обробку для порожнього графа(без ребер)).

Тому доцільно, аби не робити програму заплутанішою, перерозподілити вершини $\{1,2,3\}$, як $\{a,b,c\}$, відповідно

```
b2 :: (Graph,Placement)  -> [[Int]] -> (Graph,Placement)
b2 (gr, p) el = let min = findZeroP gr (el!!1) in
 let e =  findE gr (el!!2) in
 let m = findMaxP gr ((fst e):(snd e):min:(el!!0)) in if m==(-1) then (gr
    ,(drop 1 (act (min:(fst e):(snd e):[])))++p) else (gr,(drop 1 (act (min
    :(fst e):(snd e):m:[])))++p)
```

В випадку б2 знаходимо висячі вершину і ребро, та вершину максимального степеня - позначимо за $\{0, 1, 2, 3\}$, відповідно(a з теореми тепер 0).

$$\sigma(1) = 2, \sigma(2) = 3, \sigma(3) = 0.$$

Може здатись, що тут буде забрано 4 вершини, але насправді буде забрано 3 - мінімальна і перша по половині, тобто з 0 можна перейти, але надалі вже не можна буде перейти в 0, натомість не можна перейти з 1, але можна перейти в 1.

Отож аби не порушити баланс потрібно прийнаймні 3 ребра - $d(3) \geq 2$. Випадок, коли $d(3) = 1$ вже обговорювався вище. Якщо ж $d(3) = 0$, то після цієї перестановки граф складатиметься з ізольованих вершин і буде представлений списком порожніх списків - тобто очевидно вкладеться і перестановки будуть призначені відповідною функцією.

Також варто зазначити, чому немає варіанта б21: по суті він виконається в б2 - $a = 0, b = 3$, але інакшою перестановкою, ніж надана в теоремі.

```
v :: (Graph,Placement) -> [[Int]] -> (Graph,Placement)
v (gr, p) el = (gr, (findZeroP gr (el!!0), findMaxP gr (el!!1)):(findMaxP
    gr (el!!0), findZeroP gr (el!!1)):p)

findZeroP :: Graph -> [Int] -> Int
findZeroP gr el = head [n|n<-nodes gr, null (gr!!n), notElem n el]

findMaxP :: Graph -> [Int] -> Int
findMaxP gr el = maxL (map (change el) [(length (gr!!n), n)|n<-nodes gr])

maxL :: Placement -> Int
maxL (p1:p2:ps) = if fst p1 > fst p2 then maxL (p1:ps) else maxL (p2:ps)
maxL [p] = snd p
maxL [] = -1

change :: [Int] -> (Int,Int) -> (Int,Int)
change el p = if elem (snd p) el then (-1,-1) else p
```

В цьому фрагменті представлені функції, які використовувались в попередньому - розберемо їх:

findZeroP - повертає першу ж ізольовану вершину графа, яка не використана(і вона точно існує, оскільки ми якось потрапили в випадок в).

maxL з пар степінь-номер обирає останню вершину з найвищим степенем, а у випадку відсутності елементів в списку видає помилку -1.

change по суті робить вершини, які вже були використані, недоступними - якщо недоступна жодна вершина, в результаті в findMaxP викинеться -1

findMaxP утворює з усіх вершин пари з довжинами(степенями) та номерами, замінює використані вершини на (-1, -1) та знаходить номер вершини з найбільшим степенем

Випадок в відповідно до випадку в1 теореми переставляє ізольовану вершину з максимальною і навпаки.

$$\sigma(1) = 2, \sigma(2) = 1.$$

Випадок в2 не представлений, оскільки в випадку $H = G$ ситуація G містить ізольовану вершину, а H - ні не є можливою.

```
fullEmpty :: (Graph,Placement) -> [[Int]] -> (Graph,Placement)
fullEmpty (gr,p) el = let points = [x|x<-nodes gr, notElem x (el!!2)] in (
    gr, (act [x|x<- points, notElem x (el!!0), notElem x (el!!1)])++[(x,y)|
    x<-points,y<-points, elem x (el!!1), elem y (el!!0)]++p)
```

fullEmpty надає перестановки вершинам, що залишились, таким чином - "цілі"вершини циклічно, по-черзі переставляються, а наполовину використані з'єднуються між собою. Так можливо зробити, оскільки єдиний варіант, який утворює їх - утворює однаково по одному напіввикористанню різного типу, а щодо їх використання, скажімо в випадку в, - баланс кількості не буде порушено.

```
haveIsolatedP :: Graph -> [Int] -> Bool
haveIsolatedP gr el = if null [n|n<-nodes gr, null (gr!!n), (notElem n el)
    ] then False else True

haveIsolatedE :: Graph -> [Int] -> Bool
haveIsolatedE gr el = let ps = [gr!!n|n<-nodes gr, length (gr!!n) == 1] in
     if null [x|x<-ps, y<-x, length (gr!!y) == 1, (notElem y el)] then
    False else True
```

haveIsolatedE та haveIsolatedP - це предикати, які поветають істину у випадку, якщо граф(без використаних вершин) має ізольоване ребро чи ізольовану вершину, відповідно.

```
decide :: Bool -> Bool -> (Graph,Placement) -> [[Int]] -> (Graph,Placement
    )
decide False False gp el = a gp el
decide True True gp el = b2 gp el
decide True False gp el = b1 gp el
decide False True gp el = v gp el
```

decide, на основі значень попередніх двох предикатів, запускає відповідний випадок:

- Не має ні ізольованих ребер, ні ізольованих вершин - випадок а,
- Має і ізольовані ребра і вершини - б2,
- Має ізольовані ребра, але без ізольованих вершин - б1,

- Має ізольовані вершини, але не ребра - в.

```
cEL :: [Int] -> [Int] -> [Int]
cEL f s = [e|n<-[0..(length (f++s))-1], let e =(f++s)!!n, elem e (drop (n
    +1) (f++s))]

cELf :: Placement -> [Int]
cELf p = [fst e|e<-p]

cELs :: Placement -> [Int]
cELs p = [snd e|e<-p]
```

Списки з використаних вершин, які утворюються з повністю використаних, з використаних прообразів в G і з виористаних образів в $\overline{G}$, відповідно

```
clean :: (Graph,Placement) -> Graph
clean (gr,p) = let e = cELf p in [[y| y<-gr!!x, notElem y e, notElem x e]|
    x<-nodes gr]
```

Функція, що очищає ребра використаних вершин(аби не завадити виконанню fullEmpty випадку)

```
check :: (Graph,Placement) -> Placement -> (Graph,Placement)
check (gr,pn) pl = let n = [x|x<-pn, notElem x pl] in (gr,(fn [(fst x,snd
    x)|x<-n, y<- nodes gr, notElem (fst x, y) pl, notElem (y, snd x) pl] (
    length gr))++pl)

fn :: Placement -> Int -> Placement
fn (p:ps) l = if length [x|x<-(p:ps), x==p] == l then p:(fn ps l) else fn
    ps l
fn [] _ = []
```

Функція, яка прибирає непорозуміння з перестановками - повторювані перестановки, як для графа [[]], скажімо, чи вже напіввикористані вершини в тому вигляді, в якоиу вони не мали б бути - фінальний вигляд функції б2 приносить досить багато незручностей.

```
n2 :: Graph -> Placement
n2 gr = replacement (gr,[]) [[],[],[]]

replacement :: (Graph,Placement) -> [[Int]] -> (Graph, Placement)
replacement (gr,p) el =
 let d = decide (haveIsolatedE gr (el!!0)) (haveIsolatedP gr (el!!0)) (gr,
    p) el in
 let cd = check (clean d, snd d) p in
 let el1 = (((cELf $ snd cd)++(el!!0)):((cELs $ snd cd)++(el!!1)):((cEL (
    cELf $ snd cd) (cELs $ snd cd))++(el!!2)):[]) in if (length (snd cd) ==
     (length gr)) then cd else if null [x|x<-(fst cd), (length x) > 0] then
     fullEmpty cd el1 else replacement cd el1
```

І, на останок, функція яка скріплює функції decide, clean, ckeck, fullEmpty до купи в цикл - replacement, та функція, яка позбавляє від зайвого вводу - n2

3 Приклад антиалгоритмічного доведення існування перестановки для графа

Теорема 3.1. *Якщо* $|E(G)||E(H)| < \binom{n}{2}$ *тоді* G, H *взаємно розміщуються.*

Доведення. Зауважте, що для n навіть приклад із попереднього розділу показує, що теорема 3.1 найкраща з можливих.

Зафіксуємо G, H, що задовольняють умови теореми 3.1. Використаємо *імовірнісний метод*.

Нехай σ — випадкова бієкція з $V(G)$ в $V(H)$. Точніше, розглянемо ймовірнісний простір, $n!$ точок якого, є можливими бієкціями σ, кожна з яких має ймовірність $n!^{-1}$. Для будь-яких $e = \{i, j\} \in E(G)$, $f = \{a, b\} \in E(H)$, нехай A_{ef} позначає подію $\sigma(e) = f$ Тоді

$$Prob[A_{ef}] = 2(n-2)!/n! = \binom{n}{2}^{-1}.$$

Нехай $A = VA_{ef}$, диз'юнкція над усіма $e \in E(G)$, $f \in E(H)$ Тоді

$$Prob[A] = Prob[VA_{ef}] \leq Prob[A_{ef}] = |E(G)||E(H)|\binom{n}{2}^{-1} < 1.$$

Отже, для деякого фіксованого σ подія A не виконується. Тобто σ — це розміщення. □

4 Знаходження перестановки для $n-1$-реберного графа

4.1 Доведення існування перестановки для $n-1$-реберного графа, на базі теореми 4.4

Теорема 4.1. *Кожен граф $(p, p-2)$ вкладається у своє доповнення.*

Використаємо наступну посилену версію цієї теореми, яка доведена в [4].

Теорема 4.2. *Якщо G — будь-який позначений граф $(p, p-2)$, то існує ізоморфічне вкладення σ G у $\overline{G}$ таке, що σ не має фіксованих вершин.*

Очевидно, що вкладення в теоремах 4.1 і 4.2 виконуються для будь-якого $(p, p-n)$ графа, якщо $n \geq 2$.

Що стосується $(p, p-1)$ графів, то, здається, Г. Джозеф Стрейт був першим, хто помітив, що майже всі дерева містяться у своїх доповненнях. Він довів наступне:

Теорема 4.3. *Кожне нетривіальне дерево, яке не є зіркою, вкладається у своє доповнення.*

Тут пропонується завершити дослідження вкладення $(p, p-1)$ графів у їх доповнення, використовуючи теорему 4.3 як окремий випадок. Спочатку ми оголошуємо клас γ заборонених графів: $K_1 \cup C_3, K_1 \cup C_4, K_1 \cup 2C_3, K_{1,1} \cup C_3, K_{1,p-1}$ та $K_{1,n} \cup C_3$ де $n \geq 4$. (Див. рис. 7)

Тепер ми починаємо з утилізації найбільш надокучливого класу $(p, p-1)$ графів.

Теорема 4.4. *Нехай T будь-яке дерево, тривіальне чи нетривіальне. Якщо G є об'єднанням T і m непересічних циклів $(m \geq 1)$ і $G \notin \gamma$, то $G \subset \overline{G}$.*

Доведення. Припустимо, що серед m циклів в G є цикл C_r із $r \geq 5$. Нехай $V(C_r) = \{v_1, v_2, ..., v_{r-1}\}$ю Розглянемо $(k, k-2)$ граф $G_1 = G - \{v_1, v_2, \ldots, v_r\}$. За теоремою 4.2 ми знаємо, що існує ізоморфне вкладення

$$\sigma^* : G_1 \to \overline{G} \text{ таке,}$$

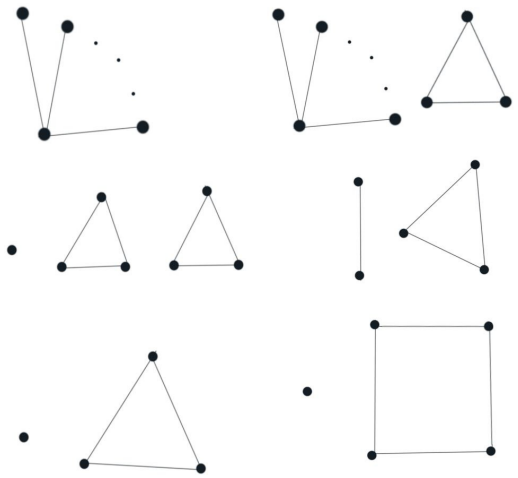

Рис. 7: γ-клас

що σ^* не має фіксованих вершин.

Крім того, за теоремою 4.3 ми знаємо, що існує ізоморфне вкладення

$$\sigma_t : C_r - v_r \to \overline{C_r - v_r}.$$

Об'єднання відображень σ^* і σ_t утворює ізоморфне вкладення σ:

$$\sigma : G \to \overline{G},$$

наступним чином:

$$\sigma(v) = \sigma^*(v) \text{ для } v \in V(G_1),$$

$$(v_i) = \sigma_t(v_i) \text{ для } 1 \leq i \leq r-1.$$

Оскільки $\sigma^*(v_r) \neq v_r$, ребра $\sigma(v_{r-1})\sigma(v_r)$ і $\sigma(v_1)\sigma(v_r)$ знаходяться в $\overline{G}$. Очевидно, всі інші ребра $\sigma(G)$ знаходяться в $\overline{G}$, тому σ є бажаним ізоморфним вкладенням.

Довівши теорему для графів із циклом C_r, де $r \geq 5$, надалі припускаємо, що m циклів G є 3-вершинними або 4-вершинними. Залишок доведення виконується індукцією по m.

$$\text{Якщо } m = 1, \text{ то } G = T \cup C_3 \text{ або } G = T \cup C_4.$$

Розглянемо $G = T \cup C_3$. Оскільки $G \notin \gamma$, T має принаймні 3 вершини. Якщо T є зіркою, то $G = K_{1,2} \cup C_3$ або $G = K_{1,3} \cup C_3$; у кожному випадку легко перевірити, що $G \subset \overline{G}$. Якщо T не є зіркою, то T має порядок

$t \geq 4$. Для $t = 4, 5$ вкладення G в $\overline{G}$ показано на рис. 8, де суцільні лінії позначають ребра G, а пунктирні лінії позначають ребра $\overline{G}$. Якщо $t \geq 6$, то існують кінцеві вершини $x, y \in V(T)$ такі, що $T - \{x, y\}$ не є зіркою; отже, існує ізоморфне відображення σ^*:

$$\sigma^* : T - \{x, y\} \to \overline{T - \{x, y\}}.$$

Викликаючи $V(C_3) = \{u_1, u_2, u_3\}$, ми визначаємо відображення σ наступним чином:

$$\sigma(u_1) = x, \sigma(u_2) = y, \sigma(u_3) = u_3, \sigma(x) = u_1,$$

$$\sigma(y) = u_2 \text{ і } \sigma(v) = \sigma^*(v) \text{ для } v \in V(T - \{x, y\}).$$

Тоді σ є вкладенням $T \cup C_3$ в $\overline{T \cup C_3}$.

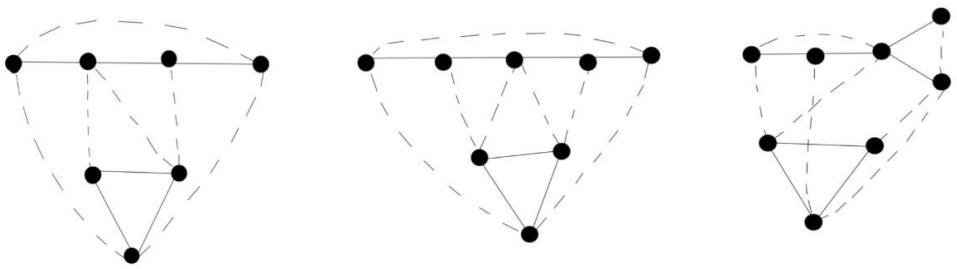

Рис. 8: $t = 4, 5 : T \cup C_3$

Розглянемо $G = T \cup C_4$: оскільки $G \notin \gamma$ то, T — нетривіальне. Якщо T — зірка, то легко побачити, що $G \subset \overline{G}$. Якщо T не є зіркою, то нехай σ^* є вкладенням T у $\overline{T}$. Крім того, нехай $v_1, v_2 \in V(T)$ і $V(C_4) = \{u_1, u_2, u_3, u_4\}$.

Ми визначаємо σ наступним чином:

$$\sigma(u_1) = u_1, \sigma(u_2) = \sigma^*(v_1), \sigma(u_3) = u_3, \sigma(u_4) = \sigma^*(v_2).$$

$$\sigma(v_1) = u_2, \sigma(v_2) = u_4 i \sigma(v) = \sigma^*(v)$$

$$\text{для } v \in V(T) \text{ і } v \notin v_1, v_2.$$

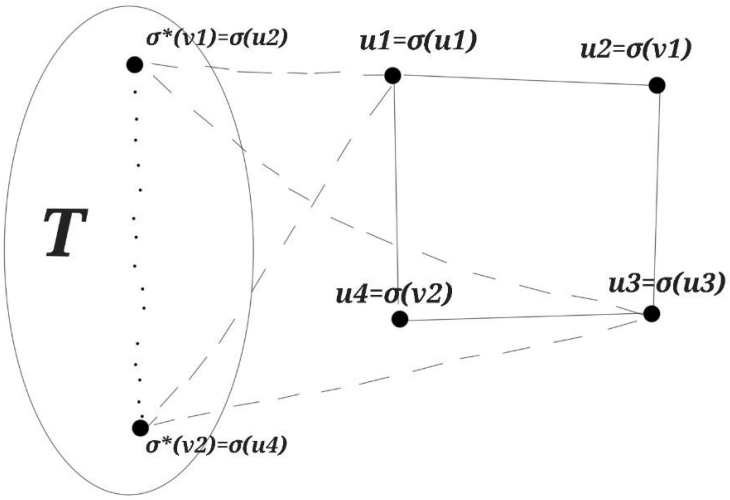

Рис. 9: $T \cup C_4$

Тоді σ — це вкладення G у $\overline{G}$, як показано на рис. 9. Це завершує доведення для $m = 1$.

Припустимо тепер, що $G \subset \overline{G}$ для будь-якого графа G, який задовольняє гіпотезу теореми, де $m < k$ і $k \geq 2$. Нехай H — граф $(p, p-1)$, який задовольняє гіпотезу, де H — об'єднання дерева та k циклів C_r, де $r = 3$ або 4. Ми розглядаємо два випадки залежно від того, чи є C_4 компонентом H.

Випадок а. Припустимо, що один компонент H є C_4. Тоді H має інший цикл як компонент. Розглянемо $H^* = H - \{C_4, C_r\}$, де $r = 3$ або 4.

Перш ніж ми зможемо застосувати індукцію до H^*, ми повинні усунути ті випадки, коли $H^* \in \gamma$ або H^* вироджуються, перетворюючись на K_1.

Якщо $H^* = K_1$, тоді $H = K_1 \cup C_4 \cup C_3$ або $H = K_1 \cup 2C_4$. В обох випадках, вкладення H у $\overline{H}$ легко побудувати. Якщо $H^* = K_1 \cup C_3$, то $H = K_1 \cup C_4 \cup 2C_3$ або $H = K_1 \cup 2C_4 \cup C_3$; знову ж таки, легко перевірити, що $H \subset \overline{H}$.

Якщо $H^* = K_1$, тоді $H = K_1 \cup C_4 \cup C_3$ або $H = K_1 \cup 2C_4$ в обох випадках, вкладення H у $\overline{H}$ легко побудувати. Якщо $H^* = K_1 \cup C_3$, то $H = K_1 \cup C_4 \cup 2C_3$ або $H = K_1 \cup 2C_4 \cup C_3$; знову ж таки, легко перевірити, що $H \subset \overline{H}$.

Якщо H^* є зіркою, то вкладення H в $\overline{H}$ показано на рис. 10, де суцільна вершина вказує на наявність пунктирних ребер між цією вершиною та всіма іншими вершинами на діаграмі які ще не інцидентні з пунктирними краями. Якщо $H^* = K_{1,1} \cup C_3$, то $H = K_{1,1} \cup 2C_4 \cup C_3$ або $H = K_{1,1} \cup C_4 \cup 2C_3$. Якщо $H^* = K_1 \cup C_4$, то $H = K_1 \cup 3C_4$ або $H = K_1 \cup 2C_4 \cup C_3$. А якщо

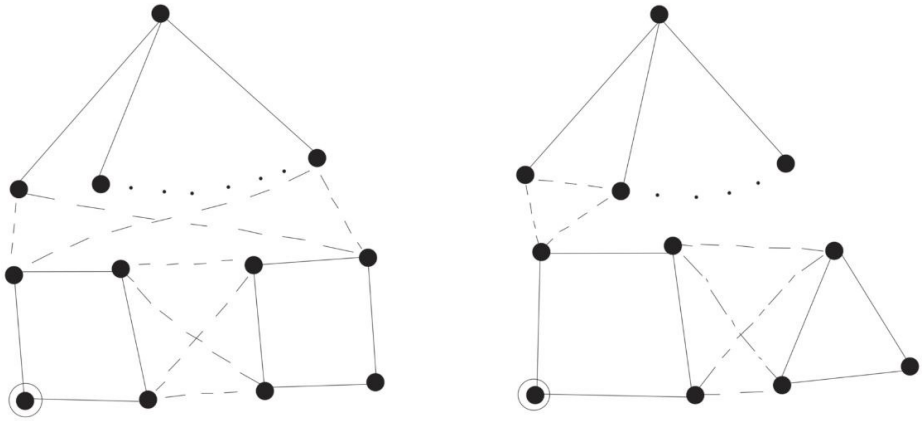

Рис. 10: $K_{1,n} \cup C_4 \cup C_r$

$H^* = K_1 \cup 2C_3$, то $H = K_1 \cup 2C_4 \cup 2C_3$ або $H^* = K_1 \cup C_4 \cup 3C_3$. У всіх цих випадках легко перевірити, що $H \subset \overline{H}$.

Нарешті, якщо $H^* = K_{1,n} \cup C_3$, з $n \geq 4$, $H = K_{1,n} \cup 2C_4 \cup C_3$ або $H = K_{1,n} \cup 2C_3 \cup C_4$. В останньому випадку вкладення H в $\overline{H}$ є очевидним. У першому випадку ми розглядаємо графік $H^{**} = H - \{C_4, C_3\} = KI_{1,n} \cup C_4$, у цьому випадку ми можемо застосувати індукційну гіпотезу до H^{**}, а оскільки $H^{**} \subset \overline{H}^{**}$ і $C_4 \cup C_3 \subset C_4 \cup C_3$, ми маємо $H \subset \overline{H}$.

У всіх инших випадках H^{**} задовольняє індукційну гіпотезу, тому $H^* \subset \overline{H}^*$. Оскільки $C_4 \cup C_r \subset C_4 \cup C_r$ для будь-якого $r \geq 3$, ми приходимо до висновку, що $H \subset \overline{H}$.

Випадок б. Припустимо, що кожна циклічна компонента H є 3-циклом, $H = T \cup kC_3$. Припустимо, що $T = K_1$. Тоді $k > 2$, інакше $H \in \gamma$. Оскільки $kC_3 \subset \overline{kC_3}$ для $k > 2$, ми маємо $H \subset \overline{H}$.

Припустимо, що T — зірка $K_{1,n}$. Якщо $n = 1$, то вкладення $H = K_{1,1} \cup kC_3$ у його доповнення є очевидним для $k = 2, 3$ і 4. Для $n > 1$ і $k = 2, 3$ і 4 ми демонструємо вкладення на рис. 11. Якщо $k > 4$, графік $H^{**} = H - 3C_3$ підкоряється індукційна гіпотеза; отже, $H^{**} \subset \overline{H}^{**}$. І, оскільки $3C_3 \subset \overline{3C_3}$, ми маємо $H \subset \overline{H}$. На цьому аналіз для T на зірку завершено.

Тепер перейдемо до загального випадку, коли T не є ні K_1, ні зіркою $K_{1,n}$. Нехай $v_1, v_2 \in V(T)$ і $\sigma^* : T \to \overline{T}$ — ізоморфне вкладення, в якому $\sigma^*(v_i) = v_1$ і $\sigma^*(v_j) = v_2$. Якщо $k = 2$, ми називаємо 3-цикли u_1, u_2, u_3 і w_1, w_2, w_3, а потім визначаємо σ наступним чином (див. рис. 12):

$$\sigma(u_1) = u_1, \sigma(u_2) = w_2, \sigma(u_3) = v_1,$$

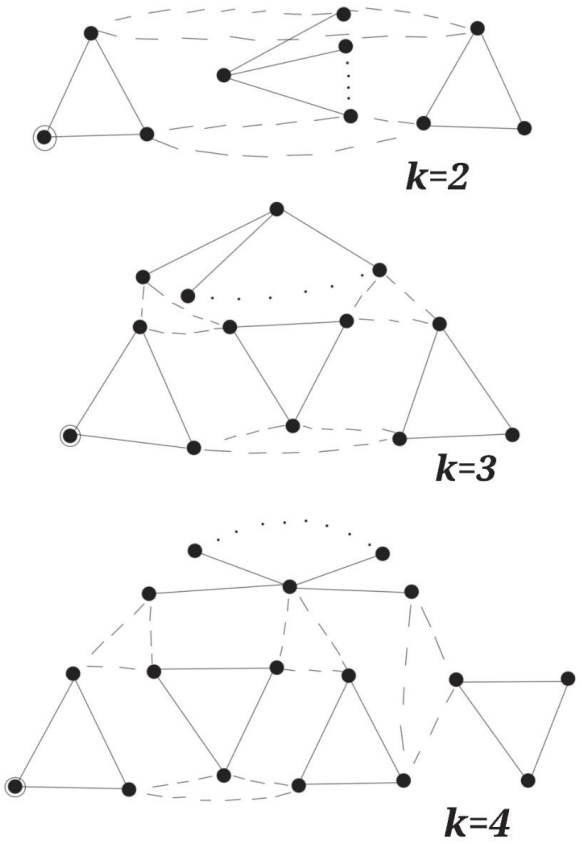

Рис. 11: $k = 2, 3, 4 : K_{1,n} \cup kC_3$

$$\sigma(w_1) = w_1, \sigma(w_2) = u_2, \sigma(w_3) = v_2,$$

$$\sigma(v_i) = u_3, \sigma(v_j) = w_3 i\sigma(v) = \sigma^*(v)$$

для решти $v \in V(T)$. Відображення σ є ізоморфним вкладенням $T \cup 2C_3$ у його доповнення. Якщо $k = 3$, то $3C_3 \subset \overline{3}C_3$ у поєднанні з $T \cup \overline{T}$ передбачає, що $H = T \cup 3C_3$ міститься у своєму доповненні. Для $k > 3$ $H^{***} = H - 3C_3$ задовольняє індукційну гіпотезу, отже, $H^{***} \subset \overline{H}^{***}$ і $3C_3 \subset \overline{3}C_3$ завершує доказ того, що $H \subset \overline{H}$. □

Теорема 4.5. *Нехай G — будь-який граф $(p, p-1)$ із $p \geq 4$. Тоді G вкладається у своє доповнення тоді і тільки тоді, коли $G \notin \gamma$.*

Доведення. Зрозуміло, що якщо $G \in \gamma$, то G не може бути ізоморфно вкладено в $\overline{G}$.

Наша увага буде обмежена випадком, коли G є незв'язним, оскільки якщо G є зв'язним, то це дерево, і в цьому випадку застосовна теорема 4.3.

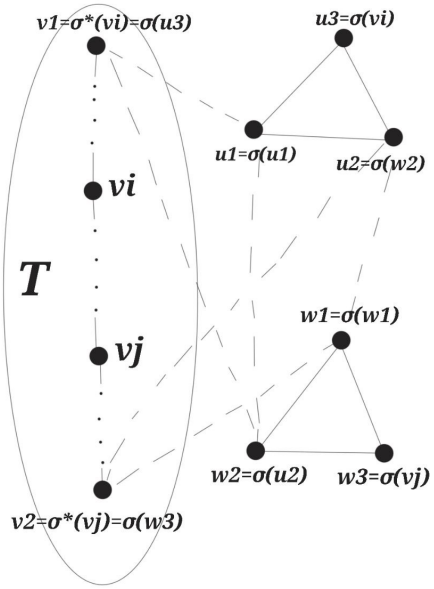

Рис. 12: $T \cup 2C_3$

Якщо v є ізольованою вершиною G, то $G-v$ є $(p-1, p-1)$ графом. Отже, $G - v$ або є об'єднанням циклів, або містить вершину u степеня $d(u) \geq 3$. Перший випадок охоплює теорема 2.5. В другому випадку $G - \{v, u\}$ є $(p-2, p-k)$ графом з $k \geq 4$. Таким чином, із зауваження після теореми 2.3 ми знаємо, що існує ізоморфне вкладення σ^*

$$\sigma^* : G - \{v, u\} \to \overline{G - \{v, u\}}.$$

Визначення $\sigma^*(v) = u$ і $\sigma^*(u) = v$ забезпечує отримання перестановки, що вбудовує G в $\overline{G}$.

Якщо граф G не має ізольованих вершин, то він повинен мати дерево T з $t \geq 2$ вершин як одну зі своїх компонент (бо кожна циклічна компонента з n вершинами має принаймні n ребер). Тоді $G - T$ є $(p-t, p-t)$ графом. Або $G - T$ є непересічним об'єднанням циклів, або $G - T$ містить вершину w, степінь якої дорівнює принаймні 3 ($d(w) \geq 3$). Перший варіант, в якому G є об'єднанням дерева та циклів, охоплюється теоремою 4.4. У другому випадку $G-T-w$ є графом $(p-t-1, p-t-s)$ з $s \geq 3$; отже, існує ізоморфне вкладення графа $G - T - w$ у його доповнення σ^*. Крім того, якщо y є вершиною максимального степеня в T, то існує ізоморфне вкладенняσ_t

$$\sigma_t : T - y \to \overline{T - y},$$

тому що $T-y$ є або $(t-1,t-n)$ графом з $n \geq 3$, або це K_1. Визначивши

$$\sigma(w) = y, \sigma(y) = w,$$

$$\sigma(x) = \sigma^*(x) \text{ для } x \in V(G-T-w),$$

$$\sigma(v) = \sigma_t(v) \text{ для } v \in V(T-y),$$

ми отримуємо ізоморфне вкладення σ G в $\overline{G}$. □

4.2 Алгоритм знаходження перестановки для $n-1$-реберного графа, на базі теореми 4.5 - необхідні компоненти

Оскільки реалізувати алгоритм поки не вдалось – поговоримо як це можна зробити і які компоненти потрібні:

Перш за все потрібні перестановки для $n-2$ та T, причому перша – без фіксованих вершин, чи ні? Насправді нам потрібно всього лиш аби вершина, взята з цикла не лишилась нерухомою. Реалізований в цій роботі алгроритм, лише через випадок б1 може залишити щось нерухомим, але остерігатись варто лише випадку на 4 вершини і це легко виправити, накинувши вершині з циклу 2 петлі(такі ребра зникнуть лише при використанні цієї вершини, але гарантуватимуть вибір цієї вершини як максимальної, тож на місці вона не залишиться і не завадить застосуванню fullEmpty)

Отож далі T - насправді можна спростити собі завдання таким чином(див. рис.13): Обираємо вершину максимального степеня, вершину, яка знаходиться від неї на відстані в 2 ребра, і сусідні вершини(це можливо, оскільки T– не зірка).В залежності від того, скільки ребер зникне, якщо забрати такий ланцюг - можна використати попередній алгоритм або багато чого сказати про граф.

Якщо ми забрали більше 4 ребер, застосуємо алгоритм для $n-2$ для утвореного графа і перестановку для P4, яка заздалегідь відома, оскільки P4 є яскравим представником самодоповняльних графів і в такій перестановці вершини не залишаються на місці.

Якщо забрали 4 ребра, то всі вершини T мають степінь 2 чи 1, бо ми вибрали вершину з найвищим степенем, або маємо 1 висячу вершини, з'єднану з 1. В першому випадку, оскільки граф T зв'язний, то маємо, що T – ланцюг $P_n, n \geq 6$. В другому після перестановки на P_4 залишиться одна ізольована вершина, яка вкладається сама в себе.Якщо забрали всього 3(менше не можна), то граф єдиний - P_4 і, як було сказано раніше, має визначену перестановку.

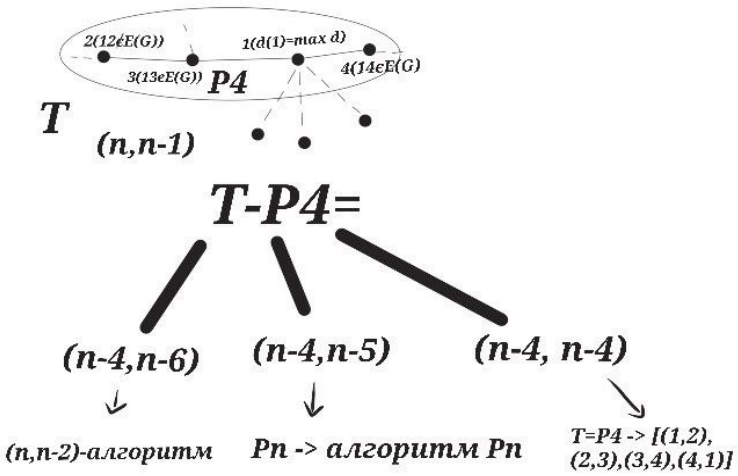

Рис. 13: T

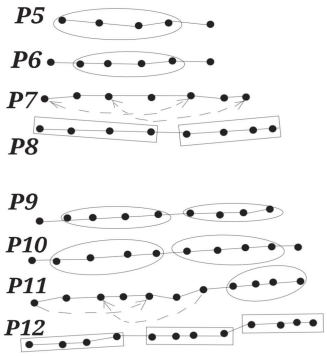

Рис. 14: P_n

Не зрозуміло, що робити тоді з ланцюгом — з ланцюгів можна виділяти ланцюги P_4 таким чином, який залежить від ділення n на 4(див. рис.14). Колами та прямокутниками позначені ланцюги P_4, які переставляються за вже відомим принципом, інші ж залишаються на місці(вкладаються в себе ж), або переставляються між собою за стрілками.

Також дуже важливим буде алгоритм для знаходження вершин циклу на 5 і більше вершин та на 3,4 вершини.

Висновки

У роботі проведено дослідження перестановок, що вклвдають G в $\overline{G}$ і їх закономірностей, що можуть бути алгоритмічно виражені, та реалізації алгоритму отримання такого ізоморфізму.. Мета роботи досягнута частково - тема дуже обширна, тож можна ще багато досліджувати і знаходити алгоритмічні патерни.

У **Розділі 1** розглянуто базові означення, які використовуються надалі в роботі. Зокрема вводиться поняття парно-знакового графа, позитивних та негативних ребер та компонент.

У **Розділі 2** наведено алгоритм вкладання графів з n вершин та $\leq n-2$ ребер, а також реалізація на Haskell.

У **Підрозділі 2.1** наведено теорему вкладання графів з n вершин та $\leq n-2$ ребер одне в одного

У **Підрозділі 2.2** наводиться опис програми на Haskell, видозмінюючи алгоритм з теореми 2.1 під частковий випадок $H = G$ і простоту кодування

У **Розділі 3** представлено теорему, доведення якої не є алгоритмічним – а навіть є антиалгоритмічним, аби показати, що далеко не все в цій темі можливо перетворити в алгоритм.

У **Розділі 4** розглянуто випадок знаходження перестановки для для n вершин та $n-1$ ребра, яке в подальшому можна закодувати.

У **Підрозділі 4.1** наведено теореми для n вершин та $n-1$ ребра, які можуть бути алгоритмічно представлені та перетворені в програмний код використовуючи реалізацію для $\leq n-2$ та нетривіальних дерев, які не є зірками.

У **Підрозділі 4.2** наведено компоненти, з яких можна скласти алгоритм знаходження перестановки для графа з n вершин та $n-1$ ребра.

Література

[1] Mariusz Wozniak, *Packing of graphs and permutations—a survey*, (2004), 379-391

[2] D. Burns, S. Schuster, *Embedding* $(n, n-1)$ *graphs intheir complements*, Israel J. Math. 30, (1978), 313-320.

[3] N. Sauer, J. Spencer, *Edge disjoint placement of graphs*, J. Combin. Theory Ser. B 25,(1978) 295-302.

[4] F. Harary: Graph Theory. Addison-Wesley, Reading, Mass., 1969.

[5] M. Wozniak, *A note on embedding graphs without small cycles*, Colloq. Math. Soc. J. Bolyai 60 (1991) 727-732.

[6] M. Wozniak, *Embedding graphs of small size*, Discrete Appl. Math. 51 (1994) 233-241.

[7] M. Wozniak, *Packing three trees*, Discrete Math. 150 (1996) 393-402.

[8] M. Wozniak, *Packing of Graphs*, Dissertationes Math. 362 (1997) 1-78.

[9] M. Wozniak, *On cyclically embeddable graphs*, Discuss. Math. Graph Theory 19 (2) (1999) 241-248.

[10] M. Wozniak, *On cyclically embeddable (n, n - 1) graphs*, Discrete Math. 251 (2002) 173-179.