Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ”

Кафедра математики факультету інформатики

Кваліфікаційна робота на тему:
Графи сум і різницеві графи
(Sum graphs and difference graphs)

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

_____
(підпис)
“____”_____ 2024 р.

Виконав студент
4-го року навчання спеціальності
113 “Прикладна математика”
Севергін Олександр Вадимович
(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)
“____”_____ 2023 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на кваліфікаційну роботу
студенту 4-го курсу факультету інформатики
Севергіну Олександру Вадимовичу

Тема:Графи сум і різницеві графи (Sum graphs and difference graphs).
Вихідні дані: Досліджено графи сум і різницеві графи.
Зміст ТЧ до кваліфікаційної роботи:

Індивідуальне завдання
Анотація
Вступ
1 Основіні означення
Висновки
Література

Дата видачі “____”_____ 2023 р. Керівник _____
(підпис)
Завдання отримав _____
(підпис)

Тема: Графи сум і різницеві графи (Sum graphs and difference graphs).
Календарний план виконання роботи:

Номер	Назва етапу курсової	Термін виконання етапу	Примітка
1.	Отримання теми кваліфікаційної роботи.	Вересень 2023 р.	
2.	Ознайомлення з темою кваліфікаційної.	Вересень 2023 р.	
3.	Розробка плану та структури роботи.	Жовтень 2023 р.	
4.	Робота з науковою літературою, опис основних означень з теорії графів.	Листопад 2023 р.	
5.	Дослідження основних властивостей королей в турнірах.	Листопад - Грудень 2023 р.	
6.	Дослідження королей в турнірах, їх типів та реалізація алгоритмів мовою програмування Python. Отримання основних результатів.	Січень 2023 р. - Квітень 2024 р.	
7.	Робота над текстовим оформленням результатів.	Квітень 2024 р.	
8.	Попередній аналіз кваліфікаційної. Виправлення помилок.	Кінець квітня - середина травня 2024 р.	
9.	Захист кваліфікаційної роботи. роботи.	Початок червня 2024 р.	

Зміст

Анотація

Нехай $G(V, E)$ - неорієнтований граф, у якому дві вершини $u, w \in V$ є суміжними тоді й тільки тоді, якщо їх сума $u + w$ теж є вершиною графа, такий граф назвемо графом суми. Введемо поняття *сумарне число*, яке позначатиме мінімальну кількість вершин, яку треба додати до заданого графа, щоб утворити з нього граф суми.
Нехай граф $G(V, E)$ - Різницеві графи

Мета роботи полягає в дослідженні графів сум і різницевих графів, їх властивостей і розробці алгоритму оптимального нумерування графів різних типів. У кваліфікаційній роботі доведені теореми, що стосуються різних властивостей графів сум, різницевих графів, а також реалізовано алгоритми мовою програмування *Python*.

Ключові слова: графи сум, сумарне число, різницеві графи, маркування графів.

Вступ

Одним із фундаментальних понять у математиці є графи. У цій кваліфікаціній роботі розглядаються два типи графів: графи суми й різницеві графи. Графи сум (англ. sum graph)- це графи, у яких кожне ребро з'єднує вершини, сума яких дає іншу вершину присутню в графі. Різницеві графи (англ. difference graph) - це графи, у яких кожне ребро з'єднує дві вершини, різниця яких по модулю дає іншу вершину присутню в графі.

Ці поняття було вперше введено наприкінці 20-го сторіччя і досліджувались до теперешніх днів. Ф. Харрарі і Д. Бергстанд у 1989 році у роботі "The sum number of a complete graph"(1989) [4] досліджували число суми для повнних графів, а роком пізніше Хараріу своїй роботі "Sum graphs and difference graphs"(1990) [1] вперше ввів поняття графів сум і різницевих графів. У роботі "Sum graphs from trees"Майкл Елінгем [5] досліджував графи сум на основі дерев. У 2020 році Ш. Ширейх у своїй дисертації "Sum graphs" [3] узагальнив попередні дослідження, а також розробив алгоритм маркування графа суми, який реалізовано в роботі псевдокодом. У 2022 році А. Сеуд. Мохамед М. Фарід і Мохамед Анвар дослідили різницеві графи у своїй статті "Some difference graphs" [6].

Графи сум і різницеві графи є актуальною і цікавою темою для дослідження так, як вони використовуються у багатьох галузях: комунікаційні мережі, біоінформатика, криптографія, обробка сигналів, логістика і навіть у соціальних мережах.

У цій кваліфікаційній роботі досліджено й доведено основні властивості й теореми, що стосуються графів сум і різницевих графів, досліджено основні властивості графів сум і різницевих графів для наступних родин графів: зірки, бізірки, графи-гусені, оливкові дерева й дерева-павуки. Також було реалізовано алгоритми для побудови графв сум і різницевих графів для заданих дерев.

1 Основні означення

Означення 1.1. [2] *Граф* G — це трійка, що складається з множини вершин $V(G)$, множини ребер $E(G)$ і відношення, яке пов'язує з кожним ребром дві вершини (не обов'язково різні), які називаються його кінцевими точками.

Означення 1.2. Пара вершин графа i, j графа G називається *суміжною*, якщо $ij \in E(G)$.

Означення 1.3. *Неорієнтований граф* G - це такий граф G, ребрам якого не присвоєно напрямок.

Означення 1.4. *Степінь вершини орієнтованого графа* - це число ребер, які суміжні із вершиною v і позначають $deg(v)$.

Означення 1.5. *Найменший ступінь* графа $G(V, E)$ - це найменший степінь вершини серед всіх $v \in V(G)$. Позначаємо $\delta(G)$.

Означення 1.6. *Ізольована вершина* - це така вершина $v \in V(G)$, яка має $deg(v) = 0$, тобто не має жодної вершини суміжної з v.

Означення 1.7. *Матриця суміжності* - це матриця розмірності $n \times n$, яка задає граф, де v_{ij} дорівнює 0, якщо $arc(i, j) \notin E$ або дорівнює 1, якщо $arc(i, j) \in E$.

Означення 1.8. *Ланцюг* - це послідовність $x_0u_1x_1u_2 \dots u_nx_n$, де x_0 - початкова вершина, u_1 - це ребро, що з'єднує початкову вершину з наступною x_1 і так до останньої вершини x_n.

Означення 1.9. *Шлях* - це ланцюг, у якому всі ребра орієнтовані в нпрямку від початку й до кінця ланцюга.

Означення 1.10. *Зв'язний граф* G - це граф $G(V, E)$, у якого між будь-якими двома вершинами $v_i, v_j \in V(G)$ існує шлях.

Означення 1.11. *Цикл* - це ланцюг у якому збігаються початкова й кінцеві вершини.

Означення 1.12. *Дерево* T - це зв'язний граф без циклів.

Означення 1.13. *Повний граф* K_n - це такий граф K_n, у якому для $\forall v_1, v_2 \in V(K)$ існує ребро $arc(v_1, v_2)$, а число n є натуральним і вказує на кількість вершин у графі.

Означення 1.14. *Двочастковий граф* G - це такий граф $G(V, E)$, у якого множина вершин V складається з двох підмножин V_1 і V_2, які не перетинаються одна з одною, а кожне ребро з множини E з'єднує одну вершину з V_1 і одну вершину з V_2.

Означення 1.15. *Повний двочастковий граф* G - це такий двочастковий граф G,у якого кожна вершина однієї частки з'єднана з усіма вершинами другої підмножини. Позначається $K_{m,n}$, де m - це кількість вершин першої підмножини, а n - кількість вершин другої.

Означення 1.16. *Зірка (англ. star)* S_k - це повний двочастковий граф $K_{1,k}$, у якого вершина 1 є центром.

Означення 1.17. *Бізірка (англ. bistar)* $B_{n,k}$ - це граф, який складається з двох зірок S_k і S_j, центри яких з'єднані між собою.

Означення 1.18. *Граф-гусінь (англ. Caterpillar)* T - це дерево, у якого всі вершини знаходяться на відстані 1 від центрального ланцюгу.

Означення 1.19. *Маркуванням на графі* $G(V, E)$ називається функція вигляду $\sigma : V \to \mathbb{N} \cup \{0\}$.

Означення 1.20. *Граф суми* $G(V, E)$ - це граф G, у якому кожному ребру $\{u, v\} \in E(G)$ відповідає вершина w така, що $\sigma(u) + \sigma(v) = \sigma(w)$.

Означення 1.21. *Різницевий граф* $G(V, E)$ - це граф G, у якому кожному ребру $\{u, v\} \in E(G)$ відповідає вершина w така, що $|\sigma(u) - \sigma(v)| = \sigma(w)$.

Означення 1.22. [6] *Оливкове дерево* T_k - це таке дерево T, яке містить k гілок, де кожна i -та гілка має i листків.

Означення 1.23. *Дерево-павук* T_k - це таке дерево T, яке містить k гілок, де кожна i -та гілка має будь-яку кількість листків.

Означення 1.24. *Множина* - це сукупність об'єктів, які мають спільні характерні риси.

Означення 1.25. *Потужність множини* - це кількість об'єктів з яких складається задана множина.

Означення 1.26. *Послідовність Прюфера* - це множина чисел, потужністю $n - 2$, де n - це кількість вершин у деякому дереві T. Послідовність Прюфера задається наступним алгоритмом: спочатку обирається найменша за номером кінцева вершина v, у послідовність Прюфера вноситься вершина з якою вона з'єднана і вершина v видаляється; цей алгоритм виконується до того моменту, як не залишаться лише дві вершини у графі.

Усі ці означення можна знайти в книгах [1], [2], [6].

2 Графи сум

2.1 Загальні відомості

Означення 2.1. [1] Нехай $S \subset \mathbb{N}$ - це скінчений набір цілих чисел. Кажемо, що граф $G^+(S) = (V, E)$ є графом суми, сумісним із множиною S, якщо $V = S$ і $E = \{\{x, y\} | x + y \in S\}$. У розширенні загальний граф G називається графом суми, якщо існує такий $S \subset \mathbb{N}$, що $G^+(S) \simeq G$.

Приклад 2.2. Припустимо, що $G^+(S)$ є графом суми і $S = \{1, 2, 3, 4\}$. Тоді, $V = S = \{1, 2, 3, 4\}$. Так як кожна вершина в V має бути сумою двох інших, то очевидно, що $v = 3, v \in V$ є сумою двох вершин $x = 1, y = 2$ і аналогічно для $v = 4, v \in V$ - сума двох вершин $x = 1, y = 3$. Отже, множина ребер має наступний вигляд: $E = \{\{1, 2\}, \{1, 3\}\}$.

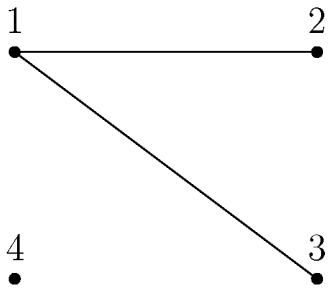


Рис. 2.1.1. Граф суми, що складається з 4 вершин.

Приклад 2.3. Припустимо, що $G^+(S)$ є графом суми і $S = \{1, 2, 3, 4, 5, 6, 7\}$. Тоді, $V = S = \{1, 2, 3, 5, 8\}$. Так як кожна вершина в V має бути сумою двох інших, то очевидно, що $v = 3, v \in V$ є сумою двох вершин $x = 1, y = 2$, для $v = 5, v \in V$ - сума двох вершин $x = 2, y = 3$, для $v = 8, v \in V$, маємо суму вершин $x = 3, y = 5$. Отже, множина ребер має наступний вигляд: $E = \{\{1, 2\}, \{2, 3\}, \{3, 5\}\}$.

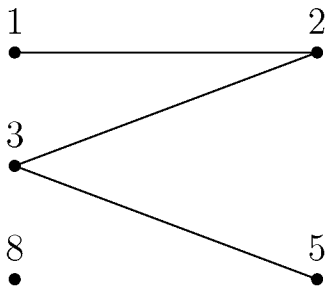


Рис. 2.1.2. Граф суми, що складається з 5 вершин.

Означення 2.4. [3] *Розмітка графа сум.* Нехай маємо граф $G(V, E)$ і ін'єктивну функцію $\sigma : V \to N^+$. Ми кажемо, що σ є рзміткою графа сум G, якщо для будь-яких двох різних вершин $u, v \in V(G)$, $\{u, v\}$ є ребром G тоді й тільки тоді, коли $\sigma(u) + \sigma(v) = \sigma w$ для деякої іншої вершини $w \in V(G)$.

Означення 2.5. [3] *Граф суми.* Граф $G(V, E)$ називається графом суми тоді й тільки тоді, коли існує розмітка графа суми σ для G.

Лема 2.6. *[3] Кожний граф суми $G = (V, E)$ містить ізольовану вершину.*

Доведення. Доведемо від супротивного. Припустимо, що граф суми не має ізольованих вершин. Тоді матимемо деяку неізольовану вершину $x \in V(G)$, яка є найбільшою. Нехай деяка вершина $y \in V(G)$ є суміжною із вершиною x, тоді сума вершин $\{x, y\} \notin V(G)$, а отже такий граф не є графом сум. □

Означення 2.7. [3] *Число суми* $\sigma(G)$ для графа G - це найменше ціле число, для якого $G^{'} = (V(G) \cup \{v_0, \ldots, v_{\sigma(G)-1}\}, E(G))$, де $\forall i \in \{0, \sigma(G) - 1\} : v_i \notin V(G)$, є графом суми.

Означення 2.8. [3] *Одиничний граф* G - це такий гарф G для якого $\sigma(G) = 1$.

Лема 2.9. *[3]* Тривіальна нижня межа. *Для кожного графа G має місце $\sigma(G) > \delta(G)$.*

Доведення. Нехай, маємо граф $G(V, E)$. Оберемо вершину з найбільною міткою $v_i \in V(G)$. Усі вершини, що не суміжні з нею, створюватимуть внршину яка вже є в графі $G(V, E)$, а всі вершини, що будуть суміжними з v_i, створюватимуть вершини з ще більшими номерами, які треба буде додати в якості ізольованих, щоб створити на основі G відповідний йому граф сум G^+. □

Лема 2.10. *[3]* Тривіальна верхня межа. *Будь-який зв'язний граф G може бути перетвеорений у граф сум G^+ шляхом додавання $E|G|$ ізольованих вершин.*

Доведення. [3] Нехай $G = (V, E)$ - це граф з вершинами $v_1, \ldots, v_n$. Пронумеруємо кожну вершину $v_i \in V(G)$ номером $\sigma(v_i) = 10^i$ і для кожного реба $\{v_i, v_j\} \in E(G)$ введемо у відповідний граф суми G_+ ізольовану вершину $v_{i,j}$ з номером $\sigma(v_{i,j}) = 10^i + 10^j$.
З означення зрозуміло, що такий метод нумерування для кожного ребра $\{v_i, v_j\} \in E(G)$ створює вершину z таку, що $\sigma(z) = \sigma(v_i) + \sigma(v_j)$. Вершиною z є додана ізольована вершина $v_{i,j}$. Залишається перевірити, що для кожної $\{u, v\} \notin E(G^+)$ немає вершини z такої, що $\sigma(z) = \sigma(u) + \sigma(v)$.
Ми можемо уявити кожний номер вершини як n-цифрове число з основою 10. Для кожної $v \in V(G)$, номер містить лише одну 1, і на всіх інших позиціях знаходяться 0. З іншого боку, вершини $z \in V(G^+) \backslash V(G)$, тобто додані ізольовані вершини, мають 1 на обох позиціях.
Відповідно, легко побачити, що для кожної пари $v_i, v_j \in V(G)$ такої, що $\{v_i, v_j\} \notin E(G)$ не існуватиме вершини $w \in W(G^+)$ такої, що $\sigma(w) = \sigma(v_i) + \sigma(v_j)$ так, як w має містити дві 1 у своєму номері, але такої вершини в $V(G)$ і ми не додавали

ізольовану вершину з таким номером.

Також, зрозумілий випадок, коли маємо $v \in V(G)$ і $z \in V(G^+)\backslash V(G)$. Вершина, що відповідатиме ребру $\{v, z\}$ має містити або три 1 у своїй мітці, або одну 1 і одну 2, але такої вершини не існує в $V(G^+)$. Аналогічно буде у випадку, коли v, z є доданими зольованими вершинами. Тоді потенційна вершина, що відповідатиме сумі цих вершин, матиме або чотири 1, або хоча б одну 2 у її номері. $\square$

2.2 Зірки

[3] Розглянемо сімейство графів - зірки. Зірку позначаємо S^n, де $n \in \mathbb{N}$, множина вершин дорівнює $\{c, v_1, \ldots, v_{n-1}\}$ і множина ребер дорівнює $\{\{c, v_i\} | i \in [n-1]\}$. Вершина c є центром зірки і всі інші називаються листками. Центральна вершина c має степінь $deg(c) = n - 1$ і $\forall x \; in \; V(G)/c : deg(x) = 1$. Зірки використовуються при проектуванні комп'ютерних мереж і розподілених обчисленнях.
Зірка S^1 - це граф з однією ізольованою вершиною. Такий граф є ізоморфним з P^0, і це є граф суми. Зірка S^2 - це граф ізоморфним з P^1, S^3 ізоморфний з P^2. Усі ці графі мають число суми 1.
Розглянемо більш складний випадок - S^4. Присвоїмо центру c значення 1. Кожній вершині $v_i \in v_1, v_2, v_3$ присвоїмо номер $2 + i$. Для кожного ребра $\{c, v_i\}, i = 1, 2$, окрім $\{c, v_3\}$, сума цих вершин є вершиною v_{i+1}. Для ребра $\{c, v_3\}$ мусимо додати ізольовану вершину z з номером 6. Таким чином додавання ізольованої вершини z не створило нові ребра у зірці, так як центр c з'єднаний з усіма іншими вершинами в $V(S^4)$ і не може бути з'днаною з ізольованою вершиною, адже z має найбільший номер. Якщо взяти суму двох найменших вершин, що є листками, $3 + 4 = 7$, що є більшим за номером доданої ізольованої вершини, а отже сума будь-яких двох інших вершин також не буде дорівнювати номеру ізольованої вершини. Тому, $\sigma(S^4) = 1$. Відповідно до Леми 2.8 це число є оптимальним.

Algorithm 1 Розмітка зірки

---

[3] Нехай S^n, $n \geqslant 2$ буде зіркою і $a \in \mathbb{N}$ - задана константа. Ми присвоїмо центральній вершині c мітку a і додамо одну ізольовану вершину v_n. Тоді кожна вершина $v_i, i \in [n]$, включаючи ізольовану v_n, отримаємо розмітку графа сум $\sigma(v_i) = n\alpha + i\alpha$.

Лема 2.11. *[3] Нехай S^n, де $n \geqslant 4$, - зірка, $\alpha, \beta \in \mathbb{N}$, $\beta > 2\alpha n$ - задані константи. Тоді для заданого графу S^n Алгоритм 1 створює граф суми G^+ і відповідну розмітку графа суми σ таке, що $G^+ \simeq S^n \cap K^1$ і*

$$\forall u, v \in V(G^+) : \{u, v\} \in E(G^+) \iff \exists w \in V(G^+) : \sigma(u) + \sigma(v) = \sigma(w).$$

Доведення. [3] **Зліва направо.** Спочатку покажемо, що для кожного ребра $\{c, v_i\}, i \in [n-1]$, існує вершина $v_k \in V(G^+)$ така, що $\sigma(c) + \sigma(v_i) = \sigma(v_k)$. Якщо ми перепишемо рівняння, то отримаємо

$$\sigma(c) + \sigma(v_i) = \sigma(v_k)$$
$$\sigma(v_k) = \alpha + n\alpha + i\alpha$$
$$\sigma(v_k) = n\alpha + (i+1)\alpha \; (3.4)$$

Рівняння (3.4) завжди має рішення $h = i+1$. Легко побачити, що завдяки доданій ізольованій вершині, усі ребра мають відповідну вершину, що є сумую двох вершин, які з'єднує ребро.

Справа наліво. Тепер доведемо, що для двох вершин $u, v \in V(S^n)$ таких, що $\{u, v\} \notin E(S^n)$, немає індукованого ребра з відповідним маркуванням. Додана ізольована вершина v_n має найбільше маркування, і відповідно до **Леми 2.6**, вона суміжна з усіма вершинами $v \in V(S^n)$. Крім того, центр з'єжна □

Теорема 2.12. *[3] Нехай S^n, де $n \in \mathbb{N}$. Тоді матимемо:*

$$\sigma(S^n) = \begin{cases} 0, \ if \ n = 1, \\ 1, \ otherwise. \end{cases}$$

Доведення. Маємо довільну зірку S^n, де $n \in \mathbb{N}$. Якщо $n = 1$, то зірка складається з однієї ізольованої вершини, відповідна вона вже є графом сум. Якщо ж $n \geqslant 1$, у такому випадку, доцільно буде використати заданий Алгоритм 1 для розмітки зірки, Алгоритм 1 є оптимальним і, використовуючи його, отримаємо зірку до якої треба додали лише одну вершину. □

2.3 Бізірки

Розглянемо сімейство графів бізірок. Бізірки мають дві центральні вершини, позначимо їх як a і b. Центральним вершина a і b приссвоїмо деякі значення α і β відповідно. Бізірку позначатимемо $B_{n,k}$, де $n, k \in \mathbb{N}$ - це цілі числа, які вказують кількість вершин з'єднаних з ценралними вершинами a і b відповідно.

Теорема 2.13. *Використовуючи Алгоритм 1, можна пронумерувати задану бізірку $B_{n,k}$ шляхом додавання 3-ох вершин.*

Доведення. Нехай маємо бізірку $B_{n,k}$, де $n, k \in \mathbb{N}$ і $\alpha = 1$.

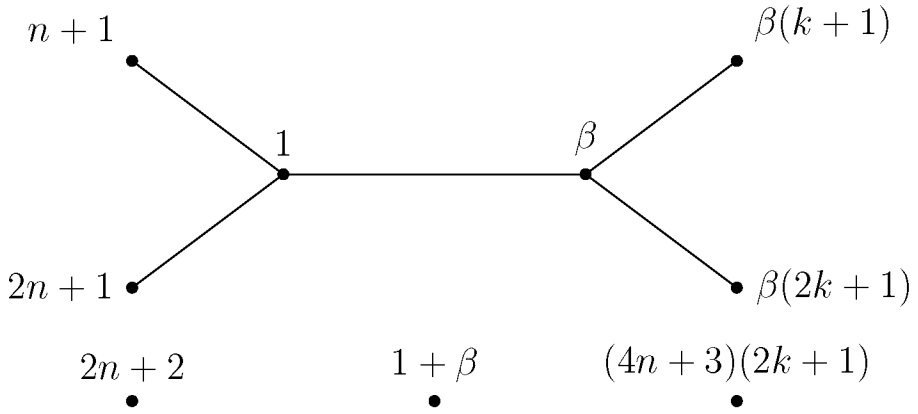


Рис. 2.3.1. Бізірка $B_{n,k}$ для якої $\sigma(B_{n,k}) = 3$.

Спершу, пронумеруємо ліву частину бізірки, тобто всі вершини суміжні з a. Тоді, за Алгоритмом 1, усі вершини зліва отримають номери за правилом $\sigma(v_i) = \alpha(n+i)$, де v_i - вершина суміжна з a,$\alpha = 1$, i - це номер вершини, тобто $\sigma(v_i) = n+i$, відповідно це будуть від $n+1$, для першої вершини суміжкої з a й до $2n+1$ для останньої n-ої вершини суміжної з a.
Тоді додамо ізольовану вершину з номером $2n+2$, яка відповідає ребру $\{1, 2n+1\} \in E(G)$.
Тепер треба пронумерувати вершину цетнральну вершину b. Нехай $\beta = 2*(2n+2)$, тобто $\beta = 4n+4$. Тоді всі вершини, що суміжні з нею будуть пронумервоані за правилом: $\sigma(v_i) = \beta(k+j)$, де v_i - вершина суміжна з b, $\beta = 4n+4$, i - це номер вершини, тобто $\sigma(v_i) = (4n+4)(n+i)$.
Як другу ізольовану вершину додамо вершину з номером $1+\beta$, тобто $1+4n+4$, відповідно матимемо $4n+5$. І залишається додати останню ізольовану вершину, яка відповадатиме сумі найбільшої вершини суміжної з b і b, і матиме номер $(4n+4)(2k+1)$.
Перевіримо, що будь-які дві несуміжні вершини в сумі не дадуть третю вершину, яка вже є в графі. Центральна вершина a й суміжні їй вершини пронумеровані за оптимальним праввилом, яке виключає таку ситуацію. Треба перевірити чи будь-які дві вершини суміжні з центральною вершиною a не дадуть в сумі іншу

вершину, що є в графі. Оберемо дві вершину з найменшими номерами: $x, y \in V(G)$, де $x = n+1, y = n+2$. Їх сума $x+y = 2n+3$, що вже більше за найбільшу вершину суміжну з центральною вершиною a і ізольовану вершину з номером $2n+2$. Аналогічно для центральної вершини b і вершин суміжних з ним

Отже, матимемо наступну бізірку:

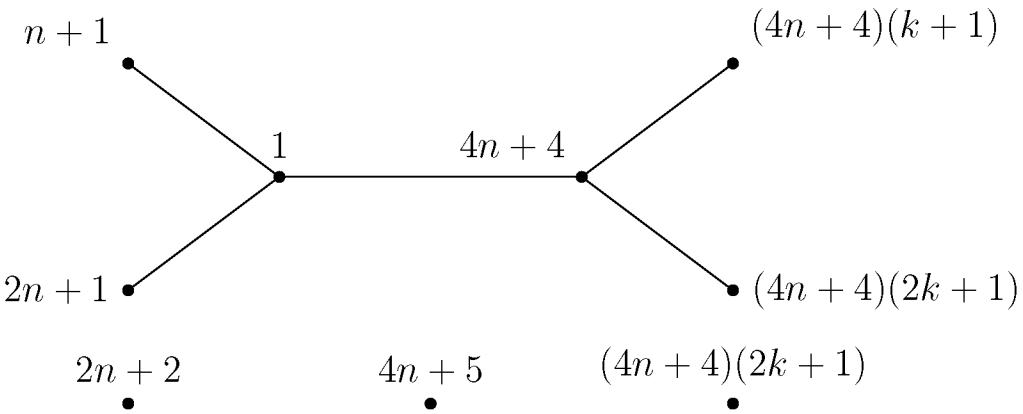


Рис. 2.3.2. Бізірка $B_{n,k}$ з маркуванням для всіх вершин $\sigma(B_{n,k}) = 3$.

□

Як бачимо із заданої бізірки можна отримати граф сум шляхом додавання 3-ох вершин. Є оптимальніший алгоритм для цього.

Теорема 2.14. *Нехай маймо задану бізірки $B_{n,k}$. Її можна перетворити на граф суми шляхом додавання 2-ох вершин. Таким чином, існуватиме алгоритм до якого $\sigma(B_{n,k}) = 2$.*

Доведення. Нехай маємо бізірку $B_{n,k}$, де $n, k \in \mathbb{N}$ і $\alpha = 1$.

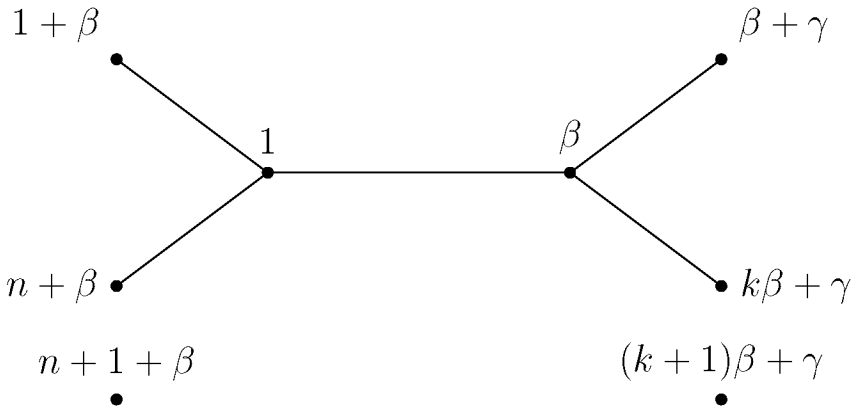


Рис. 2.3.3. Бізірка $B_{n,k}$ для якої $\sigma(B_{n,k}) = 2$.

Спершу, пронумеруємо ліву частину: промаркуємо ліву вершину сумою центральних вершин, тоді для лівої частини виконається наступне правило: $\sigma(v_i) = \sigma(v_1) * i + \beta$. Тоді суму вершин v_n, яка промаркована найбільшим номером, і лівим центром бізірки a винесемо як висячу вершину з маркуванням $\sigma(v_n) + \sigma(a) =$

$n+1+\beta$. Вершину b промаркуємо наступним чином: $\beta > n+1$. Для марквання правої частини додамо деяке $\gamma > 2 * (\beta + n + 1)$. Тоді загальне маркування правої частини матиме наступний вигляд: $\sigma(v_j) = j * \beta + \gamma$. Тоді суму вершин v_j, яка промаркована найбільшим номером, і правим центром бізірки b винесемо як висячу вершину з маркуванням $\sigma(v_j) + \sigma(b) = (k+1) * \beta + \gamma$.
Тепер доведемо, що сума маркувань жодних двох вершин $\sigma(u)+\sigma(v) \mid u, v \in V(G)$ не дасть маркування деякої вершини, яка є в $V(G)$. Візьмемо дві вершини з лівої частини: $v_i, v_k \in V(G)$. $\sigma(v_i)+\sigma(v_k) = c+2*\beta$, де c - деяка константа, $c \in [2; 2n-1]$. Так як сума $\sigma(v_i) + \sigma(v_k) = c + 2 * \beta$, очевидно, що через одночлен 2β сума $\sigma(v_i)+\sigma(v_k)$ не дорівнює будь-якій іншій вершині з лівої частини, а також висячій вершині в лівій частині. Тепер доведемо, що ця сума не дорівнює будь-якій вершині v_j з правої частини. $\sigma(v_j) = c*\beta+\gamma = c*\beta+2*(\beta+n+1) = c*\beta+2*\beta+2n+2 = (c+2)*\beta+2n+2$, де c - деяка константа, яка позначає номер елемента c. Очевидно, що $\sigma(v_j) \neq \sigma(v_i) + \sigma(v_k)$, а також $\sigma(v_j)$ не дорівнює другій іхольованій вершині. Перевіримо, чи сума будь-яких двох вершин v_i, v_k з правої частини не дасть іншу вершину, яка вже є в бізірці. $\sigma(v_i) + \sigma(v_k) = 2 * ((c+2) * \beta + 2n + 2)$, де c - деяка константа. Очевидно, що сума будт-яких двох вершин з правої частини не дорівнбє іншій вершині у правій чи лівій частинах, а також доданим ізольованим вершинам.

На рис. 2.3.4. приклад заданого алгоритму розмітки $B_{5,7}$.

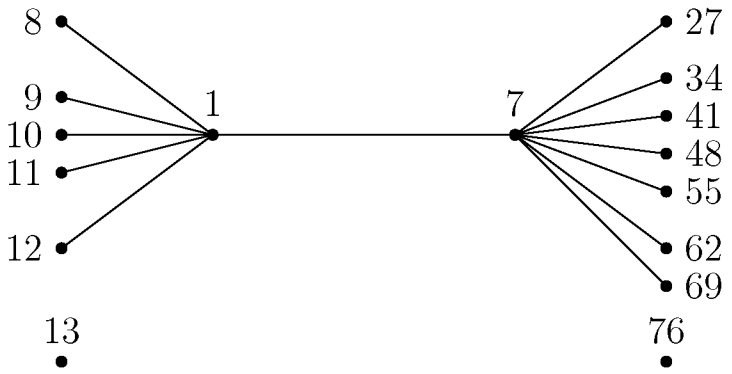


Рис. 2.3.4. Бізірка $B_{n,k}$ з маркуванням для всіх вершин для якої $\sigma(B_{n,k}) = 2$.

□

2.4 Повні графи

[3] *Повний граф* - це граф, у якому кожна вершина v є інсцедентною з усіма іншими вершинами $w \in V(G)\backslash\{v\}$. Більше формально, граф $G = (V, E)$ є повним графом на n вершинах тоді й тільки тоді, якщо $E(G) = [V(G)]^2$. Повний граф позначаємо як K_n.
Маркування сум повних графів вперше було досліджено Бергстрандом та ін. [4]. Вони представили поліноміальний часовий алгоритм, який використовує $2n-3$ ізольованих вершини. Крім того, вони довели, що така кількість ізольованих вершин є оптимальною для повних графів, тобто $\sigma(K_n) = 2n-3$ для $n \geq 4$. Тепер сформулюємо трохи узагальнений алгоритм, натхненний їхнім підходом.

Algorithm 2 Розмітка повних графів

[3] Нехай $K_n, n \geq 4$, є повним графом з вершинамии $\{v_1, \ldots, v|n\}$. Присвоїмо кожній вершині v_i номер $1 + 4 * (i-1)$ і ми додамо $2n-3$ ізольованих вершин $z_j, 1 \leq 2n-3$, з відповідним маркуванням еквівалентним $2 + 4j$.

Лема 2.15. *[3] Нехай K_n є повним графом. Тоді* ***Алгоритм 2*** *продукує граф суми G^+ і маркування $\sigma : V(G^+) \to \mathbb{N}$ таке, що $G^+ \simeq K_n \cup \overline{K_{2n-3}}$ і*
$\forall u, v \in V(G^+) : \{u, v\} \in E(G^+) \Longleftrightarrow \exists w \in V(G^+) : \sigma(u) + \sigma(v) = \sigma(w)$.

Доведення. [3] **Злівна направо**. Нехай $v_i, v_j, 1 \leq i \leq j \leq n$, дві вершини. Припускаємо, що існує вершина $z_k, 1 \leq k \leq 2n-3$ така, що $\sigma(z_k) = \sigma(v_i) + \sigma(v_j)$. Виконаємо заміну в заданому рівнянні й отримаємо:

$$\begin{aligned} \sigma(z_k) &= 1 + 4(i-1) + 1 + 4(j-1) \\ 2 + 4k &= 2 + 4(i+j-2) \\ k &= i + j - 2 \qquad (2.4.1) \end{aligned}$$

Мінімальне значення правої частини рівняння (2.4.1) є $i + j - 2 = 1 + 2 - 2 = 1$ і максимальне значення $(n-1) + n - 2 = 2n - 3$. Усі інші можливі комбінації i і j, з увахуванням зазначених меж, потрапляють в цей інтервал і всі є цілими числами. Отже, твердження завжди істинне.
Справа наліво. Ми повинні перевірити чи немає ребра між доданими ізольованими вершинами. Припустимо, що існує така вершина $v \in V(G+)$, що $\sigma(v) = \sigma(v_i) + \sigma(z_j)$, де $v_i \in V(K_n)$ і $z_j \in V(G^+)\backslash V(K_n)$. Якщо переписати рівняння, то отримаємо $\sigma(v) = 1+4(i-1)+2+4j = 3+4(i+j-1)$. Але в $V(G^+)$ немає вершини з міткою, конгруентною до 3 (mod4). Залишилося довести, що між двома ізольованими вершинами немає ребра. Припустимо, що існує така вершина $v \in V(G^+)$, що для двох різних вершин $z_i, z_j \in V(G^+)\backslash V(K_n)$ виконується $\sigma(v) = \sigma(z_i)+\sigma(z_j)$.

Легко побачити, що $2 + 4i + 2 + 4j = 4(1 + i + j)$, але ми не присвоюємо мітку, конгруентну $0 \pmod 4$ жодній вершині в v. Таким чином, вершини v не існує. □

Теорема 2.16. *[3] Нехай $n \in \mathbb{N}, n \geq 1$, і K_n є повним графом. Тоді виконується наступне:*

$$\sigma(K_n) = \begin{cases} 0 & if n = 1, \\ 1 & if n = 2, \\ 2 & if n = 3, \\ 2n - 3 & otherwise \end{cases}$$

Лема 2.17. *[3] Використовуючи маркування для $G^+ = K_n \cup \overline{K_{2n-3}}$ зазначене вище, не існує таких $i, j, k \leq n$, що $\sigma(v_i) + \sigma(v_j) = \sigma(v_k)$.*

Доведення. Розглянемо дві множини $A = \{\sigma(v_1) + \sigma(v_i) \mid i \in \{2, \ldots, n\}\}$ і $B = \{\sigma(v_n) + \sigma(v_n) \mid i \in \{2, \ldots, n-2\}\}$. Розмір множини A складає $n - 1$ і розмір множини B складає $n - 2$. Легко побачити, що $A \cap B = \emptyset$. Отже, за **Лемою 2.17** отримуємо $\sigma(K_n) \geq |A| + |B| = n - 1 + n - 2 = 2n - 3$, доведення завершене. □

2.5 Графи гусені

Означення 2.18. *Граф-гусінь (англ. Caterpillar)* T - це дерево, у якого всі вершини знаходяться на відстані 1 від центрального ланцюгу.

[5] Щоб довести, що $s(T) = 1$ для будь-якого дерева $T \neq K_1$, наведемо строгий алгоритм, який знаходить розмітку суми $T \cup \{z\}$, де z - це ізольована вершина. Алгоритм має два етапи. Етап 1 досить простий і в деяких випадках дає повну маркування суми для $T \cup \{z\}$. Однак в інших випадках етап 1 дає лише часткове маркування, і на етапі 2 необхідно вжити подальших дій для завершення маркування.

Algorithm 3 Наївний алгоритм для розмітки графів-гусеней.

[5] Довільно позначте деяку вершину $v_0 \in V(T)$ додатним цілим числом α. Виберемо v_1 сусіда з v_0 та позначимо v_1 β, де $\beta > \alpha$ і $\beta \neq 2\alpha$ (так що $\sigma(v_0)+\sigma(v_0) \neq \sigma(v_1)$). Тепер, оскільки v_0v_1 є ребром, $\sigma(v_0) + \sigma(v_1)$ має з'явитися як мітка деякої вершини. Виберемо суміжну непозначену вершину v_2 до вже позначеної вершини u_2 ($u_2 = v_0$ або v_1), і позначимо v_2 за допомогою $\sigma(v_0) + \sigma(v_1)$. Тепер $\sigma(v_2) + \sigma(u_2)$ має з'явитися на деякій вершині, тому оберемо непозначену вершину v_3, що суміжна з вже позначеною вершиною u_3, і позначає її $\sigma(v_2) + \sigma(u_2)$. Продовжуємо таким чином: на кожному кроці вершина v_{i+1} отримує мітку $\sigma(v_i) + \sigma(u_i)$, де u_i є єдиним позначеним сусідом v. Коли позначається остання вершина $v_{n-1} \in T$, сума $\sigma(v_{n-1}) + \sigma(u_{n-1})$ розподіляється ізольованій вершині z.

Наївний алгоритм описує мислення кожного, хто намагався розробити власний алгоритм маркування суми для певного графа. Проблема цього алгоритму полягає в тому, що, використовуючи його, не завжди можна створити розмітку графа суми для заданого дерева. Приклад графа гусені, яке не є графом суми:

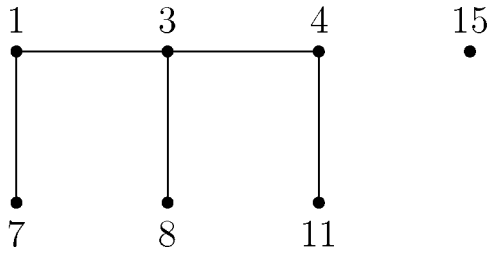


Рис. 2.5.1. Граф гусінь T ромаркований за **Алгоритмом 3**.

Тепер опишемо алгоритм маркування суми для графів-гусеней. Починаючи з хвоста графа-гумені, позначаємо кожну вершину хребта, а потім будь-які стопи, що прилягають до неї, перш ніж переходити до наступної вершини хребта.

Algorithm 4 Алгоритм графа гусені.

[5] Припустимо, що кожна вершина хребта s_i графа-гусені C має стопи $t_{ij}, 1 \leq j \leq fi$, що прилягають до неї. Маємо $f_i \geq 0$ для $1 \leq i \leq l-1$, а $f_0 = f_i = 0$ - хвіст і голова гусениці є листям. Використовуємо **Наївний Алгоритм**, щоб позначити вершини в порядку: $s_0, s_1, t_{11}, t_{12}, \ldots, t_{1f_1}, s_2, t_{21}, t_{22}, t_{2f_2}, s_3, \ldots, s_l$ (за ним z).

Приклад маркування, створеного за допомогою алгоритму графа гусені, наведено на малюнку 2.5.2 (де $\alpha = 2$ і $\beta = 5$). Треба зауважити, що $\sigma(v_{i+1}) = \sigma(v_i) + \sigma(u_i)$, де u_i є унікальною позначеною вершиною хребта, суміжною з v_i в момент, коли v_i отримує свою мітку.

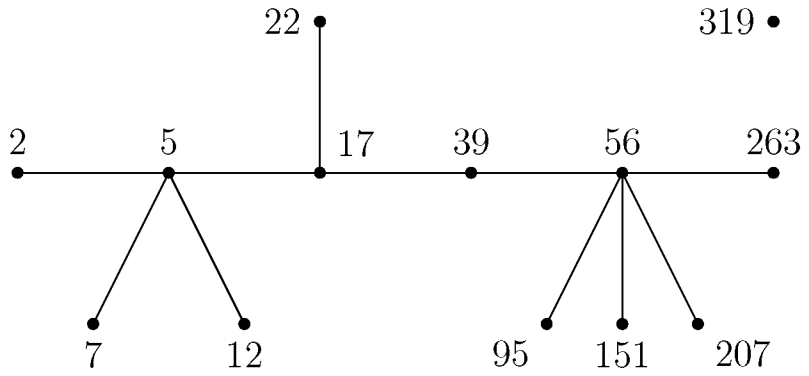


Рис. 2.5.2. Граф гусінь T ромаркований за **Алгоритмом 4.**

3 Різницеві графи

3.1 Загальні відомості

Означення 3.1. *Різницевий граф* $G(V, E)$ - це граф G, у якому кожному ребру $\{u, v\} \in E(G)$ відповідає вершина w така, що $|\sigma(u) - \sigma(v)| = \sigma(w)$.

Наведемо декілька прикладів різницевих графів:

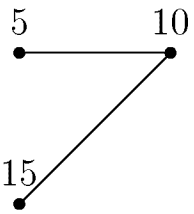


Рис. 3.1.1. Різницевий граф G з 3-ма вершинами й 2-ма ребрами.

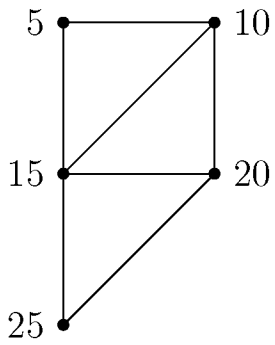


Рис. 3.1.2. Різницевий граф G з 5-ма вершинами й 8-ма ребрами.

Лема 3.2. *[1]*

1 Вершини з маркуванням s і $2s$ вважаються суміжними (перший тип).

2 Вершини з маркуванням r і t вважаються вершинами суміжними з $r + t$ (другий тип).

3 Інших суміжностей у різницевих графах немає.

Доведення. [6] (1) і (2) є очевидними. Щоб довести (3), треба зазначити що якщо вершини з маркуванням r і $r + t$ є суміжними, тоді $|(r + t) - r| = t$ належать S. Отже, або $t = r$ і маємо ребро першого роду, або $t \neq r$ і $|(r + t) - r| = t \in S$, тобто маємо ребро другого роду. □

Лема 3.3. *[6] Нехай S є набором вершин різницевого графа G,*

1 Для всіх $s \in S, s \in \{2a, a/2, a + b, |a - b|\}$ для деяких $a, b \in S$.

2 Мінімальне маркування має бути $a/2$ *або* $|a-b|$ *для деяких* a *і* $b \in S$.

3 Максимальне маркування має бути $2a$ *або* $a+b$ *для деяких* $a, b \in S$.

4 Степінь вершини з найбільшим маркуванням s *є непарним, якщо вершина* a *промаркована* $a/2$.

Доведення. [6]

1 Це очевидно з **Леми 3.2**.

2 Нехай c є найменшим маркуванням в G і нехай вершина промаркована c є суміжною з вершиною промаркованою a, а отже, $a-c \in S$, тому, або $a-c=c$, тобто $c=1/2$, або $a-c=b$ для деякого $b \in S$, з чого випливає, що $c=a-b$.

3 Нехай c — найбільше маркування в G, і нехай вершина, позначена c, є суміжною з вершиною, промаркованою a, отже, $c?a \in S$, а отже, або $c?a=a$, тобто $c=2a$, або $c?a=b$ для деякого $b \in S$, звідки випливає, що $c=a+b$.

4 Маємо два типи маркування в різницевому графі. У випадку першого типу вершина з максимальним маркуванням є суміжною з вершиною, позначеною $s/2$, тому вона має 1 у ступені вершини з максимальним маркуванням. Отже, доведено. У випадку існування другого типу, оскільки вершина з максимальною міткою є суміжною з двома вершинами, так що сума їхніх міток дорівнює s, вона розділяє кратні 2 у ступені вершини з максимальним маркуванням.

□

3.2 Повні графи

Теорема 3.4. *[6] Повний граф K_3 є різницевим графом з унікальним набором вершин: $S = \{3a, 2a, a\}$.*

Доведення. Нехай маємо деякий поаний граф K_3. Припустимо, що деяка вершина v_1 має найменший номер a. Тоді вершина v_2, що з'єднана з нею, матиме деякиий номер b. Логічно, що $|b - a| \geq a$, так як у випадку, коли $|b - a| < a$ матимемо, що в графі мусить існувати вершина v_3 з номером меншим за a, а це суперечить умові задачі. Випадок, коли $|b - a| > a$ продукує нову вершину v_3, яка має бути з'єднана з вершинами v_1 і v_2. Очевидно, що різниця по модулю в номерах вершин v_1 і v_3, v_2 і v_3 відрізнятиметься від уже наявних вершин, а отже, треба буде додати ще інші вершини, що суперечить умові задачі. Якшо ж $|b - a| = a$, тоді $b = 2a$. Аналогічно, вершина v_3 матиме номер $3a$. □

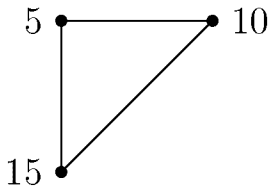


Рис. 3.2.1. Різницевий граф K_3.

3.3 Зірки

Теорема 3.5. *[6] Зірка є різницевим графом. Більш того, S э набором вершин тоді й тільки тоді, коли граф має одну з наступних форм. 1) Якщо n є парним, тоді $S = \{a, b_i, a - b_i\}$ таке, що a є найбільшим маркуванням і $|b_i - (a - b_i)| \notin S$ або $S = \{4a, 2a, a, b_i, 2a - b_i\}$ таке, що різниця між будь-якими двома елементами $\{4a, a, b_i, 2a - b_i\} \notin S$. 2) Якщо n є непарним, тоді $S = \{2a, a, b_i, 2a - b_i\}$ таке, що різниця між будь-якими двома елементами $\{a, b_i, 2a - b_i\} \notin S$ або $S = \{2a, a, b_i, a - b_i\}$ таке, що різниця між будь-якими двома елементами $\{2a, b_i, a - b_i\} \notin S$.*

Доведення. [6] 1) Нехай S_n є зіркою, коли n є парним числом. Припустимо, що u_0 є вершиною зі степенем n і максимальним номером a, тоді з **Леми 3.3** маркування $\{u_1, u_2, \ldots, u_n\}$ має бути $\{b_i, a - b_i\}$ таке, що $|b_i - (a - b_i)| \neq b_j$ або $a - b_j$. З іншої сторони, нехай u_1 - будь-яка вершина зі степенем 1 і найбільшим маркуванням $4a$, тоді за **Лемою 3.3**, і де степінь u_1 є одиницею, маркування u_0 має бути $2a$, так як жодна вершина з $u_2, u_3, \ldots, u_n$ може бути промаркована числом більшим, ніж маркування вершини u_0 (якщо таке стається, то її маркуванням є $4a$, що відкидається). Тоді u_0 буде вершиною із найбільшим маркуванням серед вершин $\{u_0, u_2, u_3, \ldots, u_n\}$, так як кількість елементів набору $\{u_2, u_3, \ldots, u_n\}$ є непарною. Тоді з **Леми 3.3** маркуванням вершин $\{u_2, u_3, \ldots, u_n\}$ будуть $\{a, b_i, 2a - b_i\}$ такі, що пізниця будь-яких двох елементів $\{4a, a, 2a - b_i \notin S\}$. 2) Аналогічно до 1). □

Приклад 3.6. [6] Різницеві графи для зірки S_8 проілюстровано на малюнках 3.3.1 і 3.3.2.

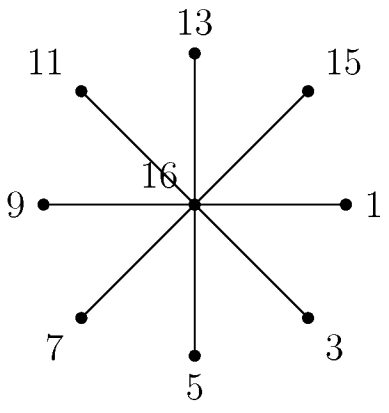


Рис. 3.3.1. Маркування різниці для зірки S_8.

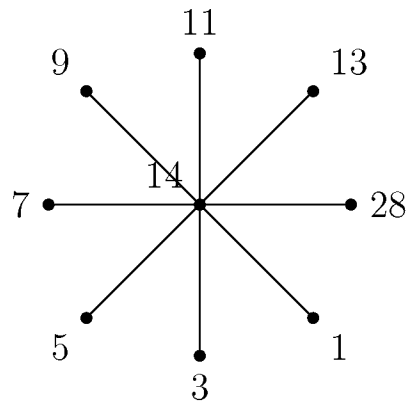


Рис. 3.3.2. Маркування різниці для зірки S_8.

3.4 Бізірки

Теорема 3.7. *[6] Бізірка є різницевим графом.*

Доведення. [6] Зобразимо бізірку як на малюнку 3.4.1.

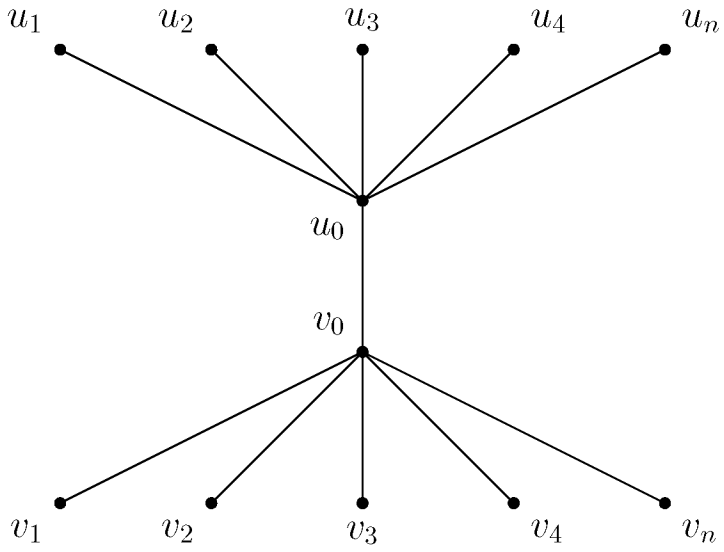


Рис. 3.4.1. Бізірка із загальним маркуванням.

Визначимо функцію маркування $f : V \to S$ наступним чином:

$$f(u_0) = 2n.$$
$$f(u_i) = 2i - 1, i = 1, 2, \ldots, n.$$
$$f(v_j) = 2n + j(2n + 2), j = 1, 2, \ldots, m.$$
$$f(v_0) = f(u_0) + f(v_m) = 4n + 2m(n + 1).$$

□

Приклад 3.8. [6] Приклад різницевої розмітки для бізірки

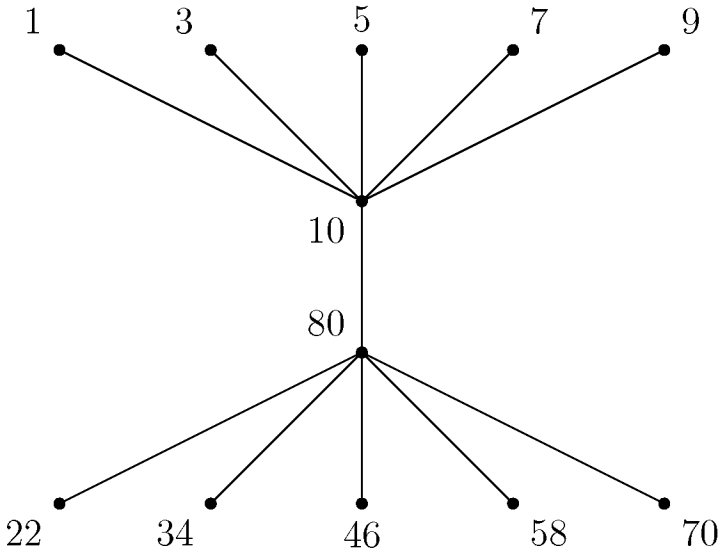


Рис. 3.4.2. Бізірка промаркована за алгоритмом вище.

3.5 Оливкові дерева

Означення 3.9. [6] *Оливкове дерево* T_k - це таке дерево T, яке містить k гілок, де кожна i -та гілка має i листків.

Теорема 3.10. *[6] Оливкове дерево* (T_k) *є різницевим графом.*

Доведення. Промаркумо оливкове дерево як зображено на малюнку 3.5.1. Визначмо функцію $f : V(T_k) \to S$ наступним чином:

$$f(root) = , c = 1, 2, \ldots, 9.$$
$$f(v_i) = 10^i, i = 1, 2, \ldots, k.$$
$$f(v_{i,1}) = 10^i - c, i = 2, \ldots, k.$$
$$f(v_{i,n}) = f(v_{i,n-2}) + f(v_{i,n-2}), i = 3, 4, \ldots, k, n = 2, 3, \ldots, m.$$

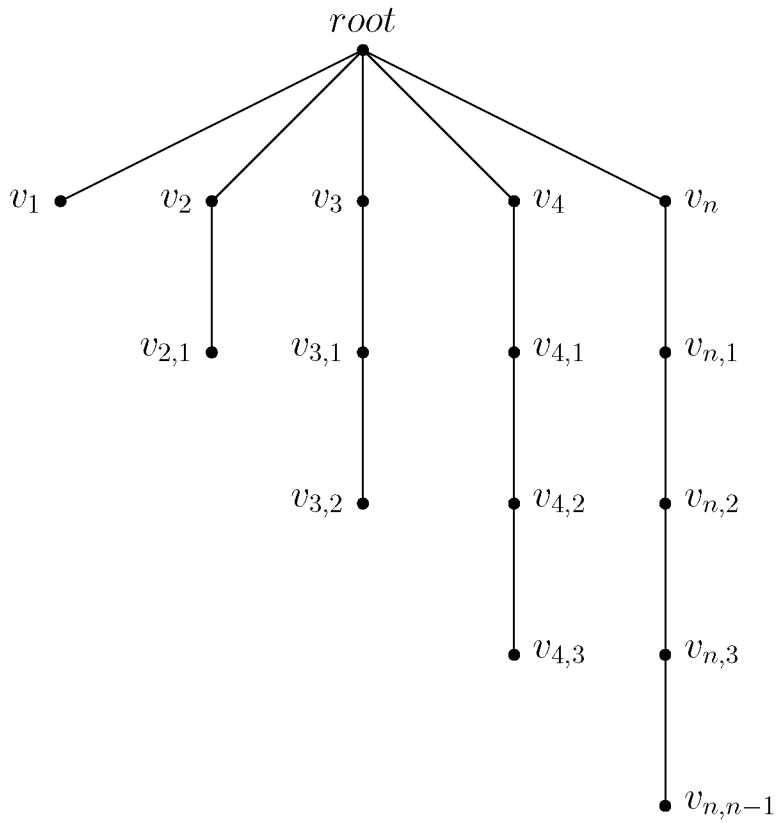


Рис. 3.5.1. Оливкове дерево промарковано загальним чином.

□

Приклад 3.11. [6] Різницеве маркування оливкового дерева (T_5) зображено на малюнку 3.5.2 і 3.5.3.

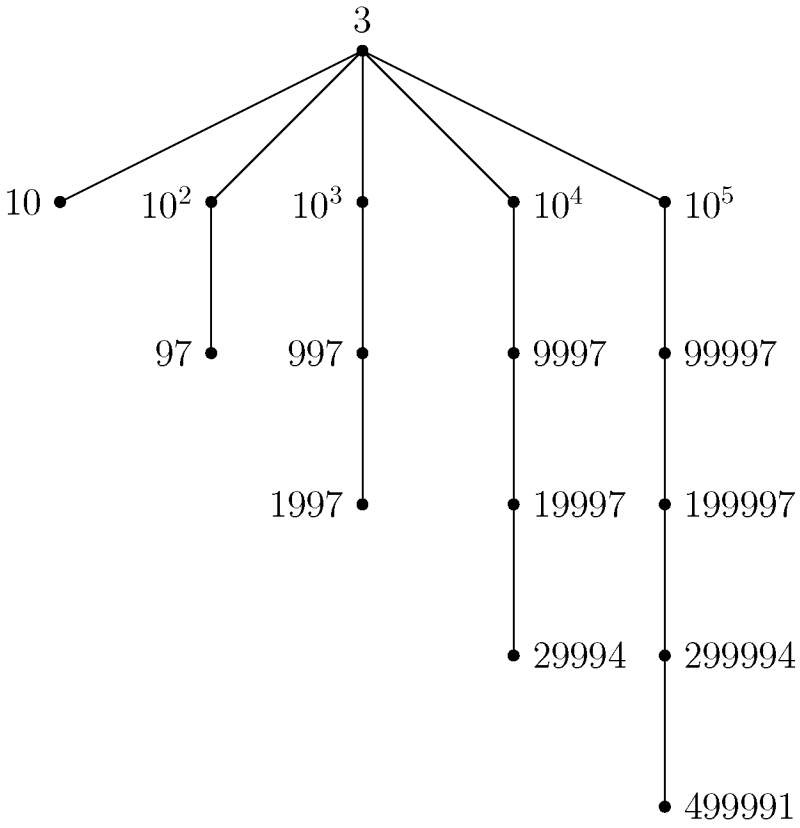


Рис. 3.5.2. Оливкове дерево із коренем 3, промарковано зазначеним алгоритмом.

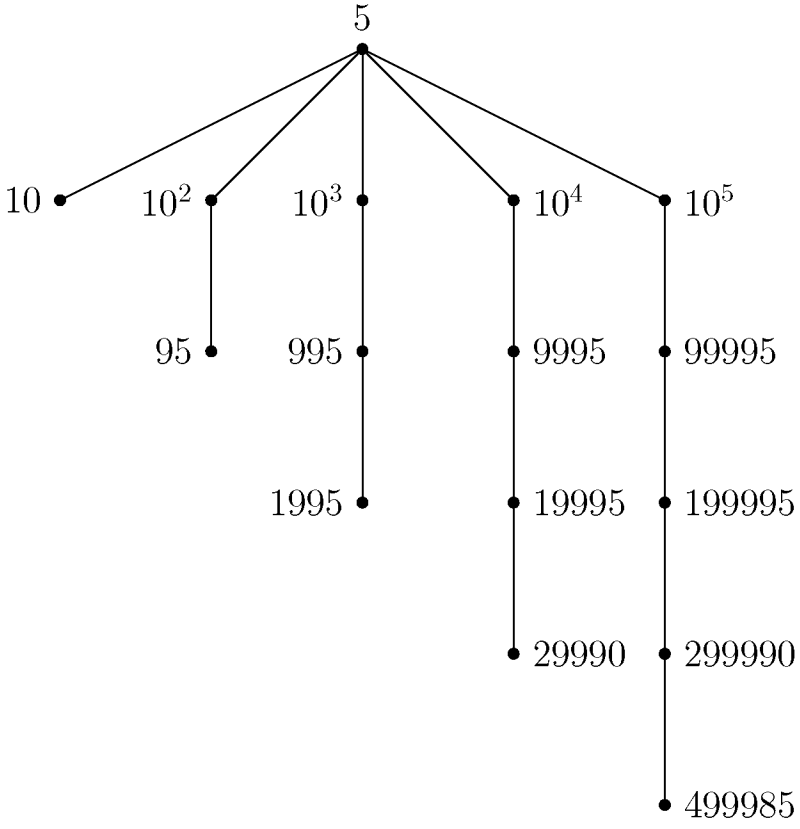


Рис. 3.5.3. Оливкове дерево із коренем 5, промарковано зазначеним алгоритмом.

3.6 Дерева-павуки

Означення 3.12. *Дерево-павук* T_k - це таке дерево T, яке містить k гілок, де кожна i -та гілка має будь-яку кількість листків.

Теорема 3.13. *Дерево павук* (T_k) *є різницевим графом.*

Доведення. Промаркумо дерево-павук як зображено на малюнку 3.6.1. Визначмо функцію $f : V(T_k) \to S$ наступним чином:
Маємо два випадки.
Якщо $c \neq 5$:

$$f(root) = , c \in \{1, 2, 3, 4, 6, 7, 8, 9\}.$$
$$f(v_i) = 10^i, i = 1, 2, \ldots, k.$$
$$f(v_{i,1}) = 10^i - c, i = 1, 2, \ldots, k.$$
$$f(v_{i,n}) = f(v_{i,n-2}) + f(v_{i,n-2}), i = 1, 2, \ldots, k, n = 2, 3, \ldots, m.$$

Якщо $c = 5$:

$$f(root) = , c = 5.$$
$$f(v_i) = 10^i, i = 1, 2, \ldots, k.$$
$$f(v_{1,1}) = 10^i + 5.$$
$$f(v_{i,1}) = 10^i - 5, i = 2, 3, \ldots, k.$$
$$f(v_{i,n}) = f(v_{i,n-2}) + f(v_{i,n-2}), i = 1, 2, \ldots, k, n = 2, 3, \ldots, m.$$

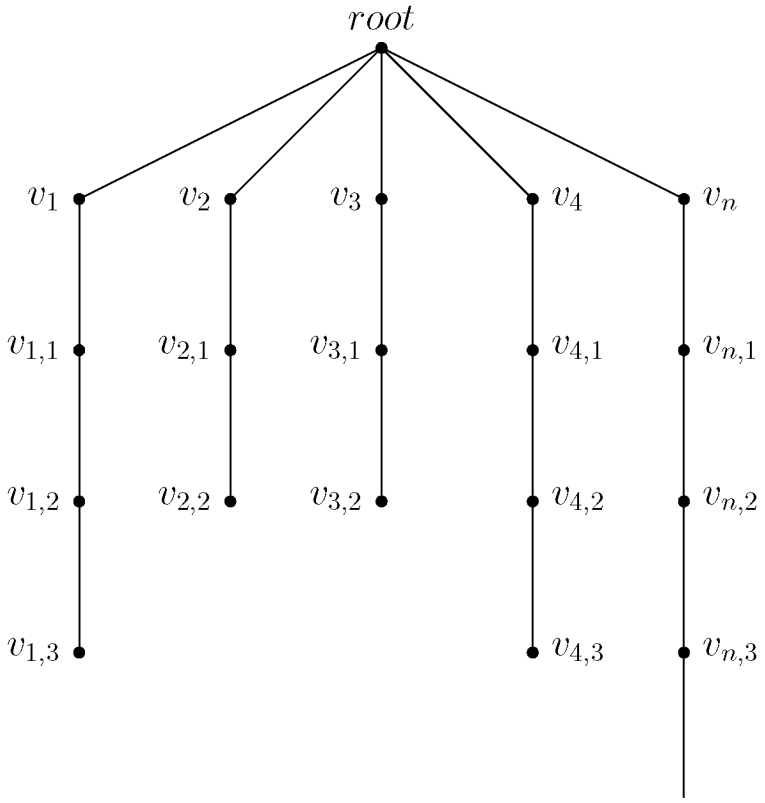


Рис. 3.6.1. Дерево-павук промарковано загальним чином.

□

Приклад 3.14. Різницеве маркування дерева-павука (T_5) зображено на малюнку 3.6.2. (для випадку, коли корінь не дорівнює 5) і на малюнку 3.6.3. (для випадку, коли корінь дорівнює 5).

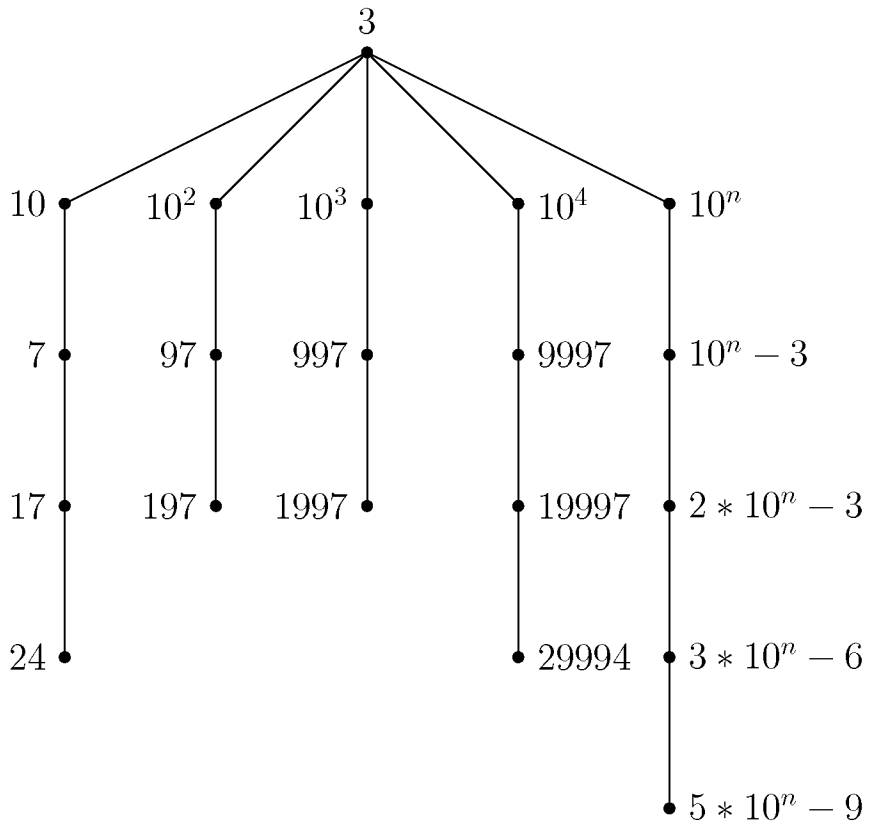


Рис. 3.6.2. Дерево-павук з коренем 3, промарковано за алгоритмом вище.

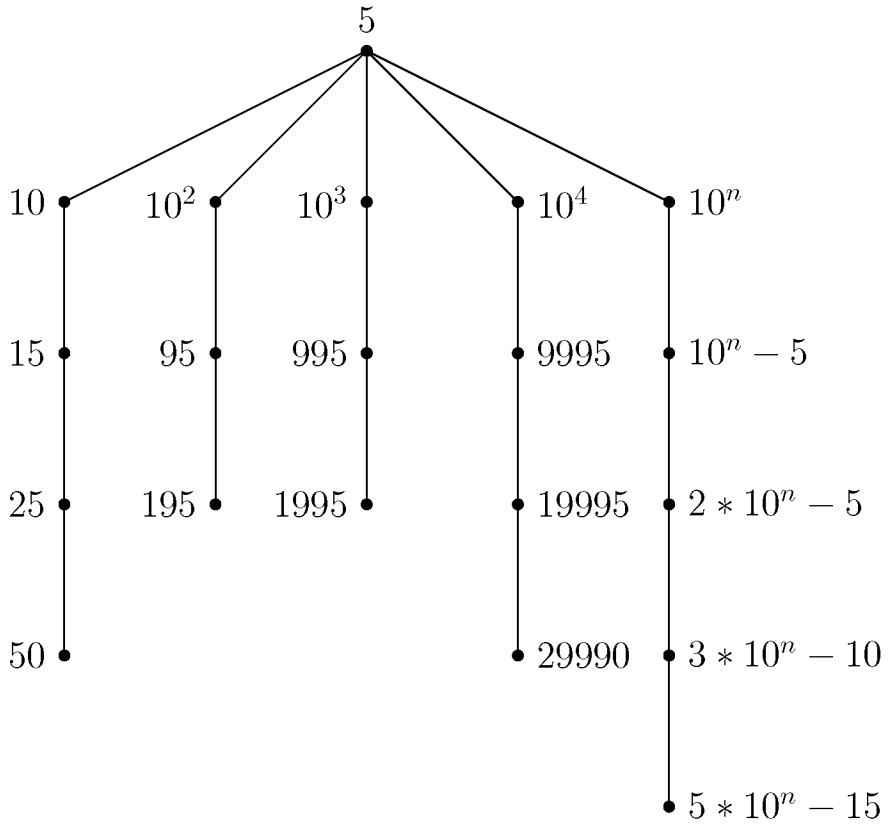


Рис. 3.6.2. Дерево-павук з коренем 5, промарковано за алгоритмом вище.

4 Алгоритми

4.1 Основні функції

У цьому підрозділі наведено реалізацію основних функцій, які будуть потрібними для реалізації алгоритмів створення графів сум і різницевих графів.

Необхідні імпорти.

```
import networkx as nx
import matplotlib.pyplot as plt
from collections import defaultdict, deque

```

Перевірка правильності введеного користувачем кода Прюфера.

```
def is_prufer_sequence(seq):
 """Tests whether the given sequence is a valid Pr?fer sequence.

 A Pr?fer sequence for a tree with n vertices is a sequence of n-2 numbers,
 which determines the structure of the tree.

 Parameters:
 seq (list): A list of integers representing a Pr?fer sequence.

 Returns:
 bool: True if seq is a valid Pr?fer sequence, False otherwise.
 """
 if not isinstance(seq, list):
  raise ValueError("Вхідні дані мають бути списком цілих чисел.")

 if not all(isinstance(x, int) for x in seq):
  raise ValueError("Усі елементи списку мають бути цілими числами.")

 if len(seq) == 0:
  raise ValueError("Послідовність не може бути порожньою.")

 n = len(seq) + 2

 for node in seq:
  if node < 1 or node >= n:
   raise ValueError('Усі вершини в послідовності Прюфера мають бути більші за 0
  і менші за n.')
 degree = [1] * n

 for node in seq:
  degree[node - 1] += 1

 for node in seq:
  found = False
  for i in range(1, n + 1):
   if degree[i - 1] == 1:
    degree[i - 1] -= 1
    degree[node - 1] -= 1
    found = True
    break
  if not found:
   return False

 return True

```

Тестування функції перевірка кода Прюфера.

```
def test_is_prufer_sequence():
 """
 Tests is_prufer_sequence function.
 """
 test_cases = [
  {
   "input": [4, 1, 3, 4],
   "expected": True
  },
  {
   "input": [4, 1, 3, 5],
   "expected": False
  },
  {
   "input": [1, 1, 1],
   "expected": True
  },
  {
   "input": [1, 2, 3, 4, 5, 6, 7, 8],
   "expected": True
  },
  {
   "input": [0, 1, 2],
   "expected": False
  },
  {
   "input": [-1, 1, 2],
   "expected": False
  },
  {
   "input": [1, 2, 2, 1, 3, 3],
   "expected": True
  },
  {
   "input": [1, 2],
   "expected": True
  }
 ]

 for i, case in enumerate(test_cases):
  input_seq = case["input"]
  expected_output = case["expected"]

  try:
   result = is_prufer_sequence(input_seq)
   assert result == expected_output, f"Тест {i+1} не пройдено для вводу {input_seq}. Очікуваний результат: {expected_output}, Отриманий результат: {result}"
   print(f"Тест {i+1} пройдено для вводу {input_seq}.")
  except Exception as e:
   assert not expected_output, f"Тест {i+1} не пройдено для вводу {input_seq}. Очікувана помилка, але отримано {result}"
   print(f"Тест {i+1} пройдено для вводу {input_seq} (очікувана помилка: {e}).")

# Виконання тестів
test_is_prufer_sequence()
```

Функція, що будує ребра на основі заданої послідовності Прюфера.

```
def prufer_sequence_to_edges(seq):
 """Converts a Prufer sequence into edges.

 Parameters:
 seq (list): A list of integers representing the Pr?fer sequence.

 Returns:
 edges (list with tuples): A list with tuples that contains edges of the graph.
 """
 if not isinstance(seq, list):
  raise ValueError("Вхідні дані мають бути списком цілих чисел.")

 if not all(isinstance(x, int) for x in seq):
  raise ValueError("Усі елементи списку мають бути цілими числами.")

 if len(seq) == 0:
  raise ValueError("Послідовність не може бути порожньою.")

 if len(seq) >= 2:
  n = len(seq) + 2
 else:
  raise ValueError("Недостатня кількість елементів у послідовності для
 створення дерева.")

 for node in seq:
  if node < 1 or node > n:
   raise ValueError('Усі вершини в послідовності Прюфера мають бути більші за 0
  і менші за n.')

 degree = [1] * n
 for node in seq:
  degree[node - 1] += 1
 edges = []
 for node in seq:
  for i in range(n):
   if degree[i] == 1:
    edges.append((i + 1, node))
    degree[i] -= 1
    degree[node - 1] -= 1
    break

 u, v = [i + 1 for i in range(n) if degree[i] == 1]
 edges.append((u, v))

 return edges

```

Тестування функції створення ребер.

```
def test_prufer_to_edges():
 """
 Tests prufer_to_edges function.
 """
 test_cases = [
  {
   "input": [4, 4, 6, 6],
   "expected": [(1, 4), (2, 4), (3, 6), (4, 6), (5, 6)]
  },
  {
   "input": [1, 1, 1, 1],
   "expected": [(2, 1), (3, 1), (4, 1), (5, 1), (1, 6)]
  },
  {
   "input": [4, 4, 4],
   "expected": [(1, 4), (2, 4), (3, 4), (4, 5)]
  },
  {
   "input": [],
   "expected": ValueError
  },
  {
   "input": [1, 1, 0],
   "expected": ValueError
  },
  {
   "input": [1, 1, 7],
   "expected": ValueError
  }
 ]

 for case in test_cases:
  input_seq = case["input"]
  expected_output = case["expected"]
  try:
   result = prufer_sequence_to_edges(input_seq)
   assert result == expected_output, f"Тест не пройдено для вводу {input_seq}. Очікуваний результат: {expected_output}, Отриманий результат: {result}"
   print(f"Тест пройдено для вводу {input_seq}.")
  except ValueError as e:
   assert expected_output == ValueError, f"Тест не пройдено для вводу {input_seq}. Очікуваний результат: ValueError, Отримана помилка: {e}"
   print(f"Тест пройдено для вводу {input_seq} (очікувана помилка: {e}).")

# Виконання тестів
test_prufer_to_edges()
```

Конвертування введеної через коми послідовності у список.

```
def input_to_number_list(text):
 user_input = input(text)
 input_list = user_input.split(',')
 try:
  number_list = [int(num) for num in input_list]  # Конвертуємо рядки в числа
  return number_list
 except ValueError:
  return "Помилка: Введені дані не є числами."

```

4.2 Створення графів суми

У цьому підрозділі наведено реалізацію функції, яка на основі введеної користувачем послідовності Прюфера створює дерево й застосовує на ньому маркування суми, відображає графічно отриманий результат.

Перевірка чи отримане дерево є графом-гусінню.

```
def is_caterpillar(edges):
 """Tests whether the given list of edges represents a caterpillar graph.

 A caterpillar is a tree in which the removal of all leaves leaves a path.

 Parameters:
 edges (list of tuples): List of edges in the form of tuples (u, v), where u and
   v are vertices.

 Returns:
 bool: True if the graph is a worm graph, False otherwise.
 """
 if not edges:
  return False

 adj_list = defaultdict(list)
 degrees = defaultdict(int)

 for u, v in edges:
  adj_list[u].append(v)
  adj_list[v].append(u)
  degrees[u] += 1
  degrees[v] += 1

 leaves = [node for node in degrees if degrees[node] == 1]

 if not leaves:
  return False

 while leaves:
  new_leaves = []
  for leaf in leaves:
   for neighbor in adj_list[leaf]:
    if neighbor in degrees:
     degrees[neighbor] -= 1
    if neighbor in degrees and degrees[neighbor] == 1:
     new_leaves.append(neighbor)
   if degrees[leaf] != 0:
    degrees.pop(leaf)
  leaves = new_leaves

 remaining_nodes = [node for node in degrees]

 if not remaining_nodes:
  return False
 if len(remaining_nodes) == 1:
  return True
 if len(remaining_nodes) == 2:
  return degrees[remaining_nodes[0]] == 1 and degrees[remaining_nodes[1]] == 1

 return all(degrees[node] == 2 for node in remaining_nodes)

```

Тестування функції, що перевіряє чи заданий список ребер відображає граф-гусінь.

```
def test_is_caterpillar():
 """Тести для функції is_caterpillar"""


 edges_caterpillar_1 = [(1, 2), (2, 3), (3, 4), (4, 5), (3, 6), (3, 7)]
 edges_caterpillar_2 = [(1, 2), (2, 3), (3, 4), (2, 5), (5, 6), (6, 7), (7, 8)]
 edges_caterpillar_3 = [(1, 2), (2, 3), (3, 4), (4, 5), (5, 6), (6, 7)]
 edges_caterpillar_4 = [(1, 2), (2, 3), (3, 4), (2, 5), (5, 6), (6, 7), (3, 8)]

 edges_not_caterpillar_2 = [(1, 2), (2, 3), (3, 4), (4, 1)]
 edges_not_caterpillar_3 = [(1, 2), (3, 4), (5, 6)]

 edges_empty = []

 assert is_caterpillar(edges_caterpillar_1) == True, "Test case 1 failed"
 assert is_caterpillar(edges_caterpillar_2) == True, "Test case 2 failed"
 assert is_caterpillar(edges_caterpillar_3) == True, "Test case 3 failed"
 assert is_caterpillar(edges_caterpillar_4) == True, "Test case 4 failed"
 assert is_caterpillar(edges_not_caterpillar_2) == False, "Test case 5 failed"
 assert is_caterpillar(edges_not_caterpillar_3) == False, "Test case 6 failed"
 assert is_caterpillar(edges_empty) == False, "Test case 7 failed"

 print("Усі тести пройдено успішно!")

test_is_caterpillar()

```

Алгоритм знаходження зреба графа-гусені.

```
def find_caterpillar_backbone(adj_list):
 """
 This function finds the backbone nodes of a caterpillar graph based on its
 adjacency list.

 Parameters:
 adj_list (dict): The adjacency list representation of the graph.

 Returns:
 backbone_nodes (list): List of backbone nodes found in the graph.
 """
 degrees = {vertex: len(neighbors) for vertex, neighbors in adj_list.items()}

 leaves = [node for node, degree in degrees.items() if degree == 1]

 nodes = []
 for leaf in leaves:
  if degrees[leaf] == 1:
   degrees[leaf] = 0
   nodes.append(leaf)
   for neighbor in adj_list[leaf]:
    degrees[neighbor] -= 1

 for vertex in adj_list:
  if degrees[vertex] == 2:
   nodes.append(vertex)

 vertexes = list(adj_list.keys())
 backbone_nodes = [x for x in vertexes if x not in nodes]

 return backbone_nodes

```

Обраховує маркування суми для заданого графа-гусені.

```
def mark_caterpillar_graph(edges):
 """
 Marks the vertices of a caterpillar graph based on the given edges.

 Parameters:
 edges (list of tuples): List of edges in the form of tuples (u, v), where u
 and v are vertices.

 Returns:
 dict: Dictionary with vertices as keys and their corresponding labels as
 values.
 """
 if not edges:
  raise ValueError("The edge list is empty")

 adj_list = defaultdict(list)

 for u, v in edges:
  adj_list[u].append(v)
  adj_list[v].append(u)

 backbone = find_caterpillar_backbone(adj_list)

 if not backbone:
  raise ValueError("Задані ребра не формують граф-гусінь")

 label = {}
 current_label = 2
 for i, node in enumerate(backbone):
  if node not in label:
   label[node] = current_label
   current_label += 1

  sum_of_adj_labels = label[node]
  for neighbor in adj_list[node]:
   if neighbor not in label:
    label[neighbor] = current_label
    current_label += sum_of_adj_labels

 largest_neighbors = sorted(label.items(), reverse=True)[:2]
 label[max(label.keys()) + 1] = largest_neighbors[0][1] + largest_neighbors
 [1][1]

 return label

```

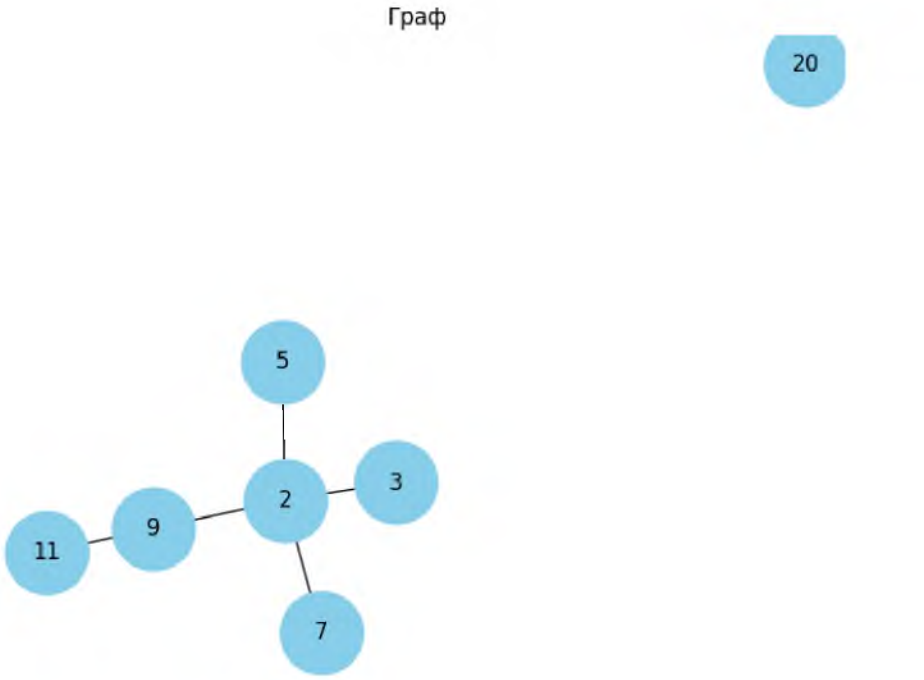
Малює граф, використовуючи наступні параметри: маркування суми й ребра.

```
def draw_graph(nodes, edges):
    """
 This function draws a graph using NetworkX library.

 Parameters:
 nodes (dict): Dictionary where keys represent node names and values represent
  node numbers.
 edges (list of tuples): List of edges in the form of tuples (u, v), where u and
   v are vertices.

 Returns:
 No explicit return value. The function displays the graph.
 """
 G = nx.Graph()

 for node, number in nodes.items():
 G.add_node(node, number=number)

 G.add_edges_from(edges)

 pos = nx.spring_layout(G)
 nx.draw(G, pos, with_labels=False, node_size=2000, node_color='skyblue',
  font_size=16)

 labels = nx.get_node_attributes(G, 'number')
 nx.draw_networkx_labels(G, pos, labels, font_size=12)

 plt.title("Граф")
 plt.axis('off')
 plt.show()

```

Основна функція, яка викликає всі функції у правильному порядку.

```
 def sum_mark_graph():
  cntn = True
  while cntn:
   seq = input_to_number_list('Введіть послідовність Прюфера (через кому): ')
   while not is_prufer_sequence(seq):
    print('Послідовність, яку Ви ввели, не є послідовністю Прюфера')
    seq = input_to_number_list('Введіть послідовність Прюфера (через кому): ')
   try:
    edges = prufer_sequence_to_edges(seq)
    labels = mark_caterpillar_graph(edges)
 draw_graph(labels, edges)
   except Exception as e:
    print(e)
    choice = input('Запустити алгоритм ще раз? (Y/y, якщо так і будь-що інше,
  якщо ні):')
    if choice.lower() != 'y':
     cntn = False

```

Приклад 4.1. Викличемо функцію *sum_mark_graph* і введемо наступну послідовність: $4, 4, 4, 5$. Отримаємо наступний результат:



4.1.1. Граф суми для послідовності Прюфера: 4,4,4,5.

Висновки

У роботі було досліджено графи сум і різницеві графи, створені алгоритми маркування суми й маркування різниці мовою програмування Python. Було доведено основні властивості й теореми та створено додаткові функції задля роботи алгоритмів .

У **Розділі 1** були наведені основні поняття з теорії графів, а також вводяться поняття графів сум, різницевих графів, маркування суми, маркування різниці і додаткові поняття, як-от зірка, бізірка й граф-гусінь.

У **Розділі 2** було наведено й доведено основні властивості й теореми, що стосуються грфів суми. У **Підрозділі 2.1** наводяться приклади графів суми й доведення лем про існування верхньої й нижньої тривіальних меж, існування іщольованих вершин у графах суми. У **Підрозділі 2.2**, наводяться основні поняття про зірки, алгоритм для розмітки зірки, доведення леми, що підтверджує правильність наведеного адгоритму, а також теореми про кількість вершин, які будуть ізольованями, задля створення із заданої зірки графу суми. У **Підрозділі 2.3** було розглянуто поняття бізірок, наведено й доведено теореми про маркування бізірок, а також приклади маркування суми для бізірок. У **Підрозділі 2.4** наведено основні поняття про повні графи, алгоритм для розмітки повних графів, лему про правильність алгоритму і її доведення, а також теорему про кількість ізольоаних вершин, які треба додати до повного графу, щоб зробити із нього граф суми. **Підрозділі 2.5** було розгялнуто поняття графа-гусені, наведено наївний алгоритм для розмітки графів-гусеней, а також алгоритм графа-гусені, який створює розмітку суми із заданого графа-гусені.

У **Розділі 3** було наведено й доведено основні властивості й теореми, що стосуються різницевих графів й наводяться приклади різницевих графів. У **Підрозділі 3.1** наведено освне означення різницевих графів, а також наведено приклади різницевих графів. У **Підрозділі 3.2** досліджується теорема про існування маркування різниці для повних графів, а також зображуються графічно повні графи із запропонованим маркуванням. У **Підрозділі 3.3** було розглянуто й доведено теорему про існування маркування різниці для графів-зірок, наведено графічні зображення. У **Підрозділі 3.4** доведено, що графи-бізірки також можна промаркувати таким чином, щоб вони були різницевими графами, наведено відповідні малюнки. У **Підрозділі 3.5** наведено загальне маркування різниці для оливкових дерев, зображено відповідні дерева із наведеним маркуванням. У **Підрозділі 3.6** запропоноване й доведене загальне маркування різниці для дерев-павуків, зображено відповідні дерева-павуки з запропонованим маркуванням.

У **Розділі 4** були наведені функції та алгоритми створення маркування суми й маркування різниці для заданих дерев. У **Підрозділі 4.1** описано загальні функції, які будуть використані в натсупних алгоритмах, такі як перевірка правиль-

ності введеної послідовності Прюфера, а також побудова дерева на основі заданої послідовності. У **Підрозділі 4.2** наведено алгоритми, які викликають необхідні функції, а також, використовуючи задану послідовність, створюють маркування суми й графічно зображають отриманий граф-суми.

Отже, у цій кваліфікаційній роботі досліджено основні поняття з тем "Графи сум і різницеві графи було доведено основні теореми й властивості, що стосуються наступних родин графів: зірки, бізірки, оливкові дерева, дерева-павуки, кактуси. Також було реалізовано алгоритми маркування суми для дерев, які дозволяють промаркувати відповідним чином заданий граф і відобразити його графічно.

Література

[1] Harary, F. *Sum graphs and difference graphs*. Congressus Numerantium, volume 72, 1990, pp. 101–108

[2] Douglas B. West, *Introduction in Graph Theory*, Pearson Education Inc., 2001, 589 p.

[3] Šimon Schierreich, *Sum graphs*, [Електронний ресурс], https://dspace.cvut.cz/bitstream/handle/10467/90024/F8-DP-2020-Schierreich-Simon-thesis.pdf?sequence=-1&isAllowed=y.

[4] Bergstrand, D.; Harary, F.; et al. *The Sum Number of a Complete Graph*. Bulletin of the Malaysian Mathematical Sciences Society, volume 12, 1989: pp. 25–28.

[5] Ellingham, M. N. *Sum graphs from trees*. Ars Combinatorica, volume 35, 1993: pp. 335–349

[6] M. A. Seoud, M. M. Farid, M. Anwar *Some Difference Graphs*, [Електронний ресурс], https://arxiv.org/pdf/2209.07317v1