Міністерство освіти і науки України

Національний університет “Києво-Могилянська академія”

Катедра математики факультету інформатики

Кваліфікаційна робота на тему:
Реберні, бльокові, тотальні графи та споріднені конструкції

Науковий керівник:
Кандидат наук, *Козеренко С.О.*

_____
(підпис)
“____”_____ 2023 р.

Роботу виконав студент
4-го року навчання
бакаляврської програми
113 “Прикладна математика”
Дехтяр Богдан-Ярема

Київ – 2023

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ”

Катедра математики факультету інформатики

ЗАТВЕРДЖУЮ
Зав. катедри математики,
проф., к.ф-м.н.
_____ Р.К. Чорней
(підпис)
“____”_____ 2023 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на кваліфікаційну роботу
студентові 4-го курсу факультету інформатики
Дехтяреві Богдану-Яремі Валерійовичеві

Тема: Реберні, блькові, тотальні графи та споріднені конструкції.
Вихідні дані: Досліджено вказані в темі графові оператори.

Зміст ТЧ до кваліфікаційної роботи:

1 Вступ
2 Реберні графи
3 Графи блькоів
4 Графи клік
5 Висновки
Література

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

Тема: Реберні, бльокові, тотальні графи та споріднені конструкції.
Календарний плян:

Нумер	Назва етапу кваліфікаційної роботи	Реченець виконання етапу	Примітка
1.	Обговорення теми роботи з науковим керівником.	15.10.2022	
2.	Пошук літератури та розробка пляну роботи.	01.11.2022	
3.	Дослідження основних властивостей розглянутих операторів.	01.12.2022	
4.	Алгоритми побудови розглянутих графів.	01.01.2023	
5.	Дослідження композицій бльокового та точко-бльокового операторів.	01.03.2023	
6.	Робота над текстом.	01.05.2023	
7.	Аналіза роботи та виправлення помилок.	10.05.2023	

Зміст

1 Вступ

Ця робота присвячена унарним операторам на графах. У ній розглянуто відображення $f : \mathcal{G} \to \mathcal{G}$ (тут $\mathcal{G}$ є множина всіх простих неорієнтованих графів), де вершинами $f(G)$ є підграфи G заданих типів, а ребра існують між підграфами конкретних типів, що перетинаються. Буде розглянуто 4 типи підграфів: вершини, ребра, бльоки та кліки.

Найвідоміший і найкраще вивчений серед розглянутих операторів є оператор узяття реберного графа. Основні відомості про цю структуру зібрано в [1]. У першому розділі цієї роботи наведено базові властивості реберного графа, а також запропоновано алгоритм його будови. Разом із тим, розглянуто 3 споріднені реберні конструкції: графи підрозбиттів, середущі й тотальні графи. Усі вони побудовані на об'єднанні множин вершин і ребер початкового графа. Для кожної з цих конструкцій також розглянуто основні властивості й наведено алгоритм побудови. Окрему уваги виділено на зв'язки між цими структурами.

Другий розділ присвячено графам, побудованих на вершинах і бльоках початкового графа. Цей розділ багато в чім нагадує перший, бо в нім також розглянуто 4 конструкції, кожна з яких деяким чином аналогічна до відповідної реберної. Ідеться про графи бльоків, бльоко-точкові дерева, точко-бльокові й бльоко-тотальні графи (остання конструкція введена автором із метою довершення аналогії з реберними структурами). Для кожного з цих графів наведено його базові властивості, алгоритм його будови, а також описано зв'язок із відповідною реберною конструкцією. Нові результати одержано при дослідження зв'язків між самими бльоковими конструкціями, зокрема при почерговому застосуванні бльокового та точко-бльокового операторів.

Третій і останній розділ присвячено графам клік. Це також добре досліджена конструкція. Її опис можна знайти, наприклад, у [8]. Крім того, в [9] досліджено метричні властивості графів клік. Ми ж обрали подібний підхід до попередніх розділів: навели базові властивості й алгоритм побудови, і дослідили зв'язки з раніше розглянутими графами. Зокрема, охарактеризували графи, граф бльоків яких ізоморфний до їхнього графа клік.

2 Реберні графи

2.1 Означення

Почнемо з розгляду позначень, які ми використовуватимемо впродовж роботи. *Графом* G ми називатимемо впорядковану пару $(V(G), E(G))$, де $V(G)$ є довільна множина, яку ще зватимемо *множиною вершин* графа G, а $E(G) = \{uv : u, v \in V(G)\}$ є *множина ребер* графа G. Якщо для двох вершин u та v існує ребро $e : u, v \in e$, то казатимемо, що u та v - *суміжні* вершини, кожна з яких *інцидентна* до ребра e. Множину ребер, інцидентних до вершини v у графові G, означатимемо $E_G(v)$. Якщо 2 ребра містять спільну вершину, називатимемо їх *суміжними. Списком суміжностей вершини* називатимемо множину вершин, суміжних до неї, а *списком суміжностей графа* - сукупність списків суміжностей усіх його вершин. Потужність списку суміжностей вершини v називається її *ступенем* та означається $d(v)$. Вершина першого ступеня зветься *кінцевою*. Для множини вершин $A \subset V(G)$ *підграфом* $G[A]$ називатимемо граф $(A, \{uv : u, v \in A; uv \in E(G)\})$.

Маршрутом у графі G називається послідовність вершин $v_1 - v_2 - ... - v_n$, $n \in \mathbb{N}$, де $\forall i \in [1, n-1] : v_i v_{i+1} \in E(G)$. *Довжиня* маршруту буде $n-1$, тобто кількість ребер між послідовними вершинами. *Шляхом* називається маршрут, у якому жадні дві вершини не збігаються. Відстанню між вершинами u й v зватимемо довжиню найкоротшого шляху між ними й означатимемо $d(u, v)$. Граф, у якому між будь-якими двома вершинами існує шлях, є *зв'язний* граф. *Циклом* називатимемо маршрут, у якому перша й остання вершини збігаються. *Простий цикл* є цикл, у якому збігаються лише перша й остання вершини. Зв'язний граф без циклів називається *деревом.*

Повним називається граф, у якому кожні дві вершини суміжні.

Важлива кляса графів, яка використовуватиметься для означення конструкцій, що ми розглядатимемо, є графи перетинів. Визначимо їх наступним чином: нехай X є множина, а $F \subset 2^X$ - деякий набір її підмножин. У такім разі *графом перетинів на множині* F називатимемо граф $\Omega(X, F) = (F, \{AB : A, B \in F, A \cap B \neq \emptyset\})$.

Два графи G й H називаються *ізоморфними* $(G \simeq H)$, коли що між ними існує *ізоморфізм*, себто бієкція $\phi : V(G) \to V(H)$, для якої виконується умова $uv \in E(G) \iff \phi(u)\phi(v) \in E(H)$.

2.2 Власне реберні графи

Нехай нам дано граф G. За умовчанням вважатимемо (якщо явно не вказано протилежного), що G - простий граф, тобто такий, який не містить кратних ребер чи петель. Даний розділ присвячено графовим конструкціям, побудованим на вершинах і ребрах G, себто таким графам $F(G)$, що $V(F(G)) \subset V(G) \cup E(G)$.

Увага 2.1. Як зрозуміло з описаного вище, "конструкцією" ми називатимемо результат застосування певної унарної операції F до графа G, результатом якої буде граф $F(G)$.

Безперечно, найвідоміша й найбільше вивчена з таких конструкцій - це реберні графи. Тож почнемо з розгляду їх.

Означення 2.2. *Реберний граф* $L(G)$ є граф перетинів на множині ребер графа G, або ж $L(G) \stackrel{def}{=} \Omega(V(G), E(G))$.

Приклад 2.3. На **Рис. 1** зображено приклад графа G й відповідного реберного графа.

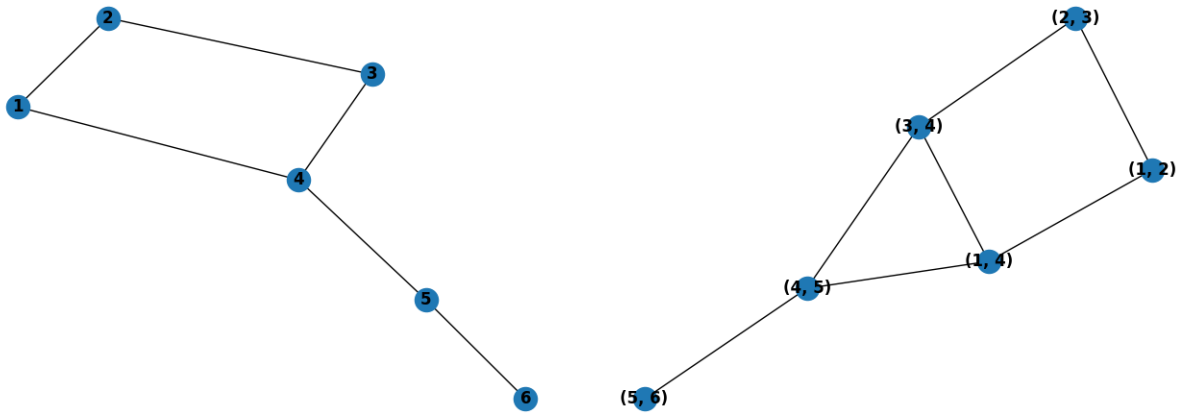

Рис. 1: Граф G (ліворуч) та його реберний граф $L(G)$ (праворуч).

Розглянемо основні властивості реберних графів. Їх усі можна знайти в [1].

Твердження 2.4. *Нехай G є довільний граф. Тоді* $|V(L(G))| = |E(G)|$, *а* $|E(L(G))| = \frac{1}{2}\sum_{v \in V(G)} d^2(u) - |E(G)|$.

Теорема 2.5. *(Характеризація реберних графів). Для графа G існує граф H такий, що $L(H) \simeq G$ тоді й тільки тоді, коли що існує набір множин $\mathcal{F} \subset 2^{V(G)}$, який задовольняє 3 умови:*

1) $\forall A \in \mathcal{F} : G[A]$ *повний;*

2) $\forall v \in V(G)\ |\mathcal{F}_v \stackrel{def}{=} \{A : A \in \mathcal{F}, v \in A\}| = 2$*;*

3) $\forall e \in E(G)\ \exists! A \in \mathcal{F} : e \subset A$*.*

Остання теорема стверджує наступне: граф G реберний тоді й тільки тоді, коли існує деяке покриття $\mathcal{F}$ множини його вершин, кожен елемент якого породжує повний підграф G, при чім кожне ребро G належить точно одному елементові покриття, а кожна вершина - точно двом елементам.

Увага 2.6. Умову 2 **Теореми 2.5** можна замінити на $\forall v \in V(G)\ |\mathcal{F}_v| \leq 2$. Із умови 3 випливає, що $|\mathcal{F}_v| \geq 1\ \forall v \in V$. Якщо ж $|\mathcal{F}_v| = 1\ \forall V_1 \subset V$, то $\mathcal{F} := \mathcal{F} \cup \{\{v\} : v \in V_1\}$ - покриття, що задовольняє умову 2 в її нинішньому формулюванні.

Увага 2.7. Для графа $L(G)$ з **Рис. 1** покриття $\mathcal{F}$ можна обрати наступним чином: $\mathcal{F} = \{12, 23, 14, 345, 56, 6\}$.

Теорема 2.8. *(Теорема Вітні про ізоморфізм). Нехай для зв'язних графів G_1 і G_2 ($|V(G_1)| \geq |V(G_2)|$) виконується умова $L(G_1) \simeq L(G_2)$. Тоді або $G_1 \simeq G_2$, або ж $G_1 = K_{1,3}$ й $G_2 = K_3$.*

Теорема 2.9. *Якщо граф G зв'язний, то $G \simeq L(G) \iff G$ є простий цикл.*

Завершимо опис реберних графів алгоритмом їхньої побудови (**Algorithm 1**).

Як і всі алгоритми в цій роботі, напишемо його псевдокодом, а також мовою Python із використанням бібліотеки NetworkX.

```
import networkx as nx

def L(G):
    H = nx.Graph()
    H.add_nodes_from(G.edges)
    for node in G.nodes:
```

Algorithm 1 Find line graph $L(G)$ of graph G

Require: Graph G
Ensure: Line graph $L(G)$
1: Create an empty edge list E'
2: **for** each node v in G **do**
3: View adjacency list adj of v
4: **for** each pair of nodes $first$ and $second$ in adj **do**
5: Create edge $e_1 = \{v, first\}$
6: Create edge $e_2 = \{v, second\}$
7: Add edge $\{e_1, e_2\}$ to edge list E'
8: **end for**
9: **end for**
10: Construct line graph $L(G)$ using edge list E'
11: **return** $L(G)$

```
    for first, _ in G.adj[node].items():
        for second, _ in G.adj[node].items():
            if first < second:
                H.add_edge((min(node, first), max(node, first)),
                    (min(node, second), max(node, second)))
return H
```

Увага 2.10. NetworkX дозволяє працювати як із списком ребер, так і зо списком суміжностей графа.

Схема алгоритму вельми проста: для кожної вершини ми розглядаємо список вершин, суміжних до неї, і стверджуємо, що кожна пара вершин у нім дає ребро в реберному графові. Оскільки кожну пару ми перевіряємо щонайбільше 2 рази, то час роботи алгоритму зростає лінійно від кількости таких пар, а отже й від кількости ребер в $L(G)$. Це означає, що оптимізовувати алгоритм можна хібащо з точністю до сталої величині. Оскільки ж порядок кількости ребер $L(G)$ в найгіршому випадку кубічний від $|V(G)|$, то час роботи можна записати як $O(|V(G)|^3)$ (це також видно зо структури алгоритму).

Увага 2.11. На **Рис. 1** реберний граф зґенеровано саме за допомогою алгоритму та виведено з поміччю методи $nx.draw(G, with\_labels = True, font\_weight =' bold')$. Якщо не вказано іншого, то рисунки графових конструкцій і надалі буде одержано таким самим чином.

2.3 Графи підрозбиттів

Наступна кляса графів, що нас цікавить - це графи підрозбиттів.

Означення 2.12. Дано граф G. Його *граф підрозбиттів* $S(G) \stackrel{def}{=} (V(G) \cup E(G), \{ve : v \in V(G), e \in E(G), v \in e\})$.

Приклад 2.13. Граф G та його граф підрозбиттів $S(G)$ зображено на **Рис. 2**.

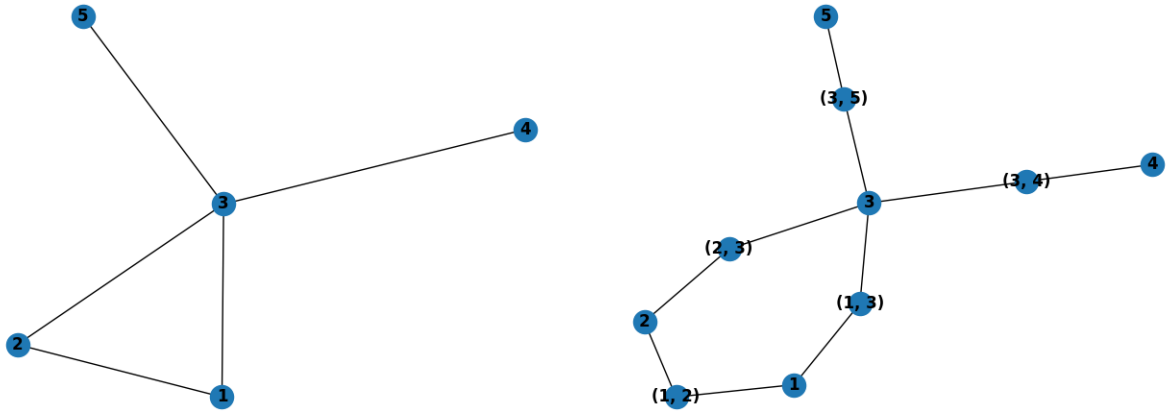

Рис. 2: Граф G (ліворуч) та його граф підрозбиттів $S(G)$ (праворуч).

Увага 2.14. Насправді, маючи граф G, побудувати відповідний граф підрозбиттів $S(G)$ напрочуд просто. Досить розбити кожне ребро G на 2 нові, "вставивши" посередині вершину (звідси й така назва). Утім, подане вище формальне означення цікаве для нас не тільки з точки зору математичної строгости, а й за те, як воно проливає світло на зв'язок між графами підрозбиттів і реберними графами: одні описують зв'язки між вершинами й ребрами графа G, другі - його міжреберні зв'язки.

Подаємо алгоритм побудови графа підрозбиттів (**Algorithm 2**).

```
import networkx as nx

def S(G):
    H = nx.Graph()
    H.add_nodes_from(G.nodes)
    H.add_nodes_from(G.edges)
```

Algorithm 2 Construct subdivision graph $S(G)$ of graph G

Require: Graph G
Ensure: Subdivision graph $S(G)$
1: Create an empty edge list E'
2: **for** each node v in G **do**
3: View adjacency list adj of v
4: **for** each node u adjacent to v in adj **do**
5: Create edge $e = \{v, \{u, v\}\}$
6: Add edge e to edge list E'
7: **end for**
8: **end for**
9: Construct subdivision graph $S(G)$ using edge list E'
10: **return** $S(G)$

```
for node in G.nodes:
   for adj, _ in G.adj[node].items():
      H.add_edge(node, (min(node, adj), max(node, adj)))
return H
```

Оскільки суміжність вершини v до вершини u тотожна до інцидентности вершини v з ребром uv, то список суміжностей графа можна також розглядати як список інцидентностей, де для кожної вершини інцидентні ребра ми задаємо вершинами-сусідами. Враховуючи, що кожна інцидентність, у свою чергу, дає ребро графа підрозбиттів, ми таким чином одержуємо алгоритм для його побудови: проходимося по спискові суміжностей, і для кожної суміжности (інцидентности) додаємо відповідне ребро до графа, що будуємо. Оскільки на додавання кожного ребра йде стала кількість операцій, то наведений алгоритм оптимальний із точністю до сталої. У найгіршому випадкові, час його роботи можна оцінити як $O(|V(G)^2|)$.

2.4 Середущі графи

Означення 2.15. Дано граф G. Його *середущий граф* $M(G) \stackrel{def}{=} (V(G) \cup E(G), \{eg : e, g \in E(G), e \cap g \neq \emptyset\} \cup \{ve : v \in V(G), e \in E(G), v \in e\})$.

Увага 2.16. Множини вершин і ребер середущого графа є об'єднання відповідних множин графа підрозбиттів і реберного графа.

Приклад 2.17. На **Рис. 3** зображено приклад графа G та його середущого графа $M(G)$.

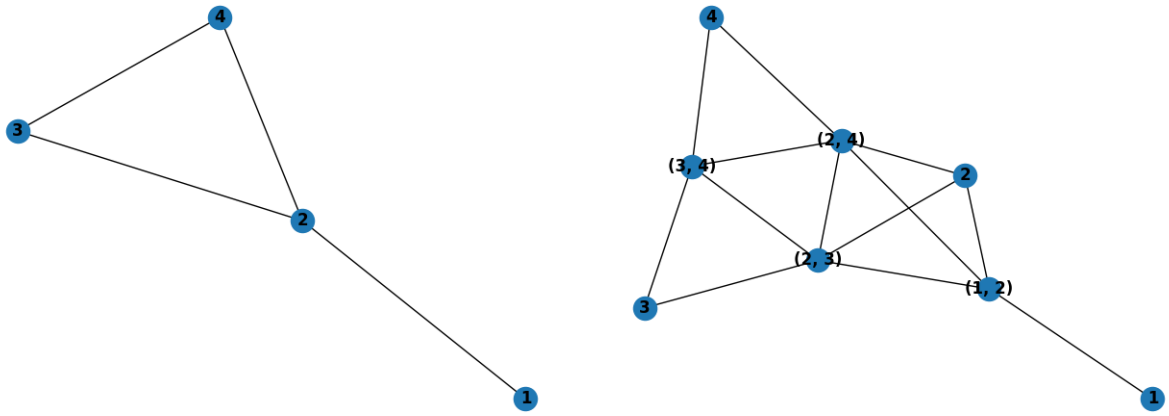

Рис. 3: Граф G (ліворуч) та його середущий граф $M(G)$ (праворуч).

Твердження 2.18. *Нехай $H = M(G)$. Тоді $\forall v \in V(G) : H[\{v\} \cup E_H(v)]$ - повний підграф графа H.*

Довід. З урахуванням того, що $E_H(v)$ є множина ребер G, інцидентних до v, цей факт є очевидний наслідок **Означення 2.15**. □

Увага 2.19. Результатом попереднього твердження є простий спосіб одержати середущий граф із графа підрозбиттів [4]. Для цього слід зробити наступне: для кожної вершини $v \in V(G)$ сполучити ребрами кожні дві вершини $e, g \in E_{S(G)}(v)$. Обґрунтування в тім, що $E_{S(G)}(v)$ і є множина ребер G, інцидентних до v.

Означення 2.20. *Кінцевореберним графом* графа G називатимемо граф, отриманий із G "навішуванням" додаткового ребра на кожну вершину G. Позначатимемо G^+.

Наступний результат, що пов'язує реберні та середущі графи, можна знайти в [4].

Теорема 2.21. *Для графа G його середущий граф є реберний граф від кінцевореберного графа G^+, себто $M(G) \simeq L(G^+)$.*

Довід. Позначимо вершини G як u_i, ребра G як e_j, а "навішені" ребра G^+ як e_{u_i}. Побудуємо бієкцію між вершинами $M(G)$ та $L(G^+)$ наступним чином: $\phi : V(M(G)) \to V(L(G^+))$, $\phi(e_j) = e_j$, $\phi(u_i) = e_{u_i}$. Покажемо, що вона зберігає ребра графів:

- e_j й u_i суміжні в $M(G) \iff u_i \in e_j \iff e_j$ й e_{u_i} суміжні в $G^+ \iff$ вони суміжні в $L(G^+)$;
- e_i й e_j суміжні в $M(G) \iff e_i$ й e_j суміжні в $G \iff$ вони суміжні в G^+;
- Очевидно, що e_{u_i} та e_{u_j} суміжні бути не можуть.

Тим то ϕ є ізоморфізм між графами $M(G)$ та $L(G^+)$, що завершує доведення. □

Наведемо алгоритм побудови середущого графа (**Algorithm 3**).

Algorithm 3 Construct middle graph $M(G)$ of graph G

Require: Graph G
Ensure: Middle graph $M(G)$
1: Construct line graph $L(G)$ of G
2: Construct subdivision graph $S(G)$ of G
3: Set $V(M(G)) := V(S(G))$
4: Set $E(M(G)) := E(L(G)) \cup E(S(G))$
5: **return** $M(G)$

```
import networkx as nx

def M(G):
    H = nx.Graph()
    line_graph = L(G)
    subdivision_graph = S(G)
    H.add_nodes_from(subdivision_graph.nodes)
    H.add_edges_from(line_graph.edges)
    H.add_edges_from(subdivision_graph.edges)
    return H
```

Зверніть увагу, що $E(M(G)) = E(L(G)) \sqcup E(S(G))$ (це випливає з означень відповідних графів). Даний факт робить алгоритм побудови $M(G)$, знаючи $L(G)$ і $S(G)$, тривіяльним. Працюватиме він за $O(|V(G)^3|)$. З оптимальности попередніх двох алгоритмів випливає також оптимальність даного (серед операцій, виконаних при побудові $L(G)$ та $S(G)$, "зайві" тільки ті, що витрачені на пошук $V(L(G))$, але відповідний час - $O(|V(G)^2|)$ - асимптотично незначущий).

2.5 Тотальні графи

Означення 2.22. Дано граф G. Його *тотальний граф* $T(G)$ - це граф, множина вершин якого є об'єднання множин вершин і ребер графа G, при чім дві вершини $T(G)$ з'єднані тоді й тільки тоді, коли що відповідні елементи в G суміжні чи інцидентні.

Приклад 2.23. На **Рис. 4** зображено граф G разом із його тотальним графом $T(G)$.

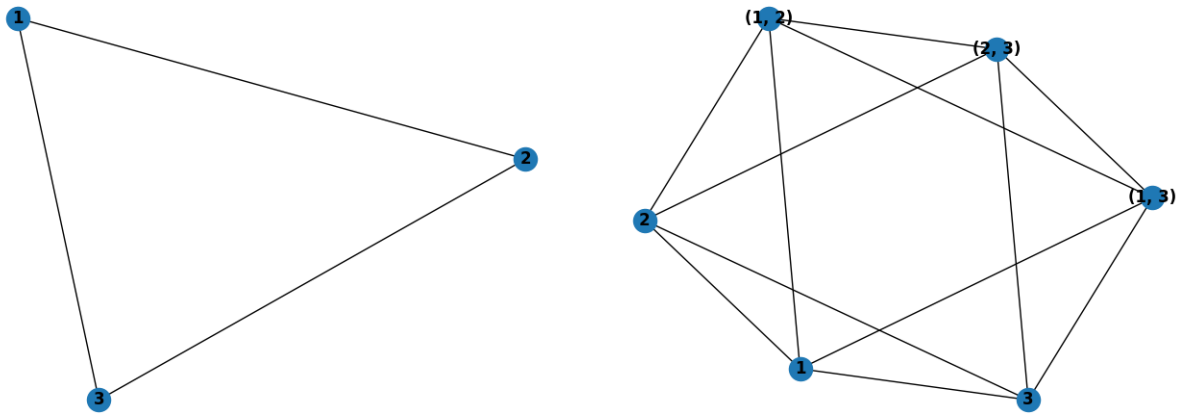

Рис. 4: Граф G (ліворуч) та його тотальний граф $T(G)$ (праворуч).

Наступне означення дозволить компактно записати відношення (**Теорему 2.26**), що ним пов'язані тотальний граф і граф підрозбиттів [5].

Означення 2.24. *Квадратом* графа G називатимемо граф G^2, множина вершин якого збігається з множиною вершин G, при чім дві вершини G^2 суміжні тоді й тільки тоді, коли в G вони сполучені шляхом довжині не більшої за 2.

Увага 2.25. Квадрат графа $L(G)$ позначатимемо $(L(G))^2$, а не $L^2(G)$, бо останній запис позначає ітерування операції взяття реберного графа. Те саме стосується й до всіх інших конструкцій, що були чи будуть розглянуті впродовж роботи.

Теорема 2.26. $T(G) \simeq (S(G))^2$.

Довід. Очевидно, що $V(T(G)) = V((S(G))^2)$. Лишається показати, що множини ребер також збігаються. У графі $S(G)$ дві вершини суміжні тоді й тільки тоді, коли відповідні елементи G інцидентні. Нехай тоді a і b такі вершини $S(G)$, що $d(a, b) = 2$. Тоді $\exists c \in V(S(G)) : ac, bc \in E(S(G))$. У такім разі маємо 2 випадки:

- $c \in V(G) \implies a$ і b - 2 ребра, інцидентні до спільної вершини $\implies a$ і b суміжні;
- $c \in E(G) \implies a$ і b - 2 вершини, інцидентні до спільного ребра $\implies a$ і b суміжні.

Отже, 2 вершини $(S(G))^2$ суміжні $\implies$ відповідні елементи G суміжні або інцидентні, тим то $E((S(G))^2) \subseteq E(T(G))$. Подібними міркуваннями можна переконатися, що вкладення виконується і в інший бік, що завершує доведення. □

Покажемо алгоритм для тотальних графів (**Algorithm 4**).

Algorithm 4 Construct total graph $T(G)$ of graph G

Require: Graph G
Ensure: Total graph $T(G)$
1: Construct middle graph $M(G)$ of G
2: Set $V(T(G)) := V(M(G))$
3: Set $E(T(G)) := E(M(G)) \cup E(G)$
4: **return** $T(G)$

```
def T(G):
    H = M(G)
    H.add_edges_from(G.edges)
    return H
```

Поруч середущого графа, тотальний граф відрізняється лише тим, що в нім з'єднані вершини початкового графа. Із цього ми легко одержуємо алгоритм. Працює він за $O(|V(G)^3|)$ операцій, і є оптимальним із точністю до сталої.

3 Графи бльоків

У попередньому розділі ми розглянули конструкції на ребрах графа G. Даний же розділ присвячено конструкціям, що будуються за схожою логікою, але вже на бльоках графа G. Цю "схожість" буде детальніше пояснено в одному з дальших підрозділів.

Для розуміння цієї секції слід знати декілька важливих понять.

Означення 3.1. *Компонента зв'язности* є максимальний за включенням зв'язний підграф.

Означення 3.2. *Точкою з'єднання* називається вершина, вилучення якої приводить до збільшення кількости компонент зв'язности у графові.

Означення 3.3. Зв'язний граф, що не містить точок з'єднання, називається *двозв'язним*.

Означення 3.4. *Бльок* є максимальний за включенням двозв'язний підграф.

Множину бльоків графа G позначатимемо $\beta(G)$.

Означення та властивості *бльока*, а також супровідних понять можна знайти в [1]. Найважливіші з них подамо тут.

Теорема 3.5. *Вершина v графа G є точка з'єднання* $\iff \exists A, B \neq \emptyset, A \sqcup B = V(G), \forall a \in A\ \forall b \in B$: *v лежить на кожному шляхові між a та b.*

Теорема 3.6. *Вершини u й v графа G лежать на спільному циклі* $\iff \exists B \in \beta(G), |V(B)| \geq 3 : u, v \in V(B)$.

Твердження 3.7. *Два бльоки можуть перетинатися по щонайбільше одній вершині, яка є точка з'єднання.*

3.1 Власне графи бльоків

Означення 3.8. Дано граф G. Його *граф бльоків* $B(G)$ є граф перетинів на множині бльоків $G : B(G) = \Omega(V(G), \beta(G))$.

Приклад 3.9. Граф G та його граф бльоків $B(G)$ на **Рис. 5**.

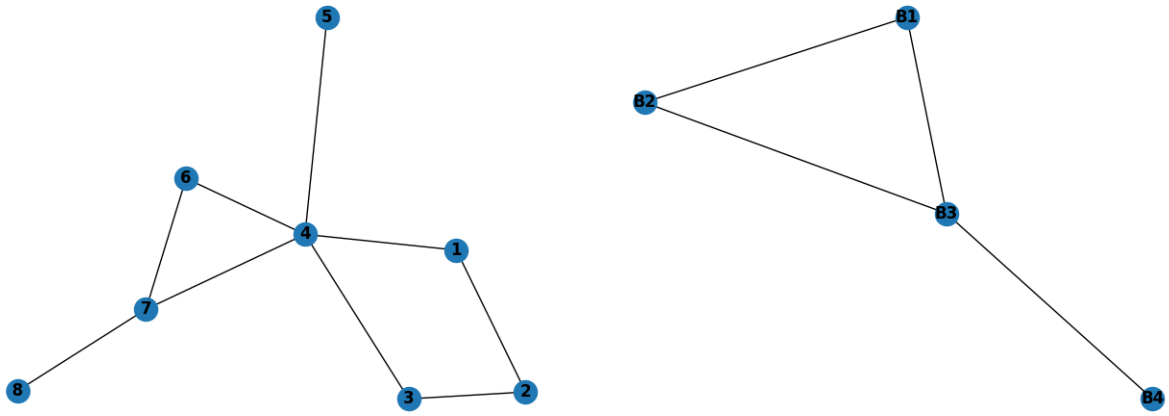

Рис. 5: Граф G (ліворуч) та його граф бльоків $B(G)$ (праворуч). Бльоки містять такі вершини: $B1 : 1, 2, 3, 4$; $B2 : 4, 5$; $B3 : 4, 6, 7$; $B4 : 7, 8$.

Основні властивості графа бльоків можна знайти в [1]. Наступна теорема опублікована в [7].

Теорема 3.10. *(Характеризація графів бльоків). Дано граф G. Тоді існує граф H такий, що $G \simeq B(H) \iff$ кожен бльок G повний.*

Уважний читач міг помітити схожість між означенням графа бльоків та означенням реберного графа. Дійсно, обидва є графи перетинів, тільки що перший на множині бльоків, а другий - на множині ребер графа G. Беручи на увагу, що для такої кляси графів як дерево бльоки й ребра еквівалентні, одержимо наступний факт.

Теорема 3.11. *Нехай G є зв'язний граф. Тоді $L(G) \simeq B(G) \iff G$ є дерево.*

Наведемо алгоритм знаходження графа бльоків. Алгоритм для знаходження бльоків графа можна знайти, наприклад, у [2]. Наразі вважатимемо, що масив бльоків графа в нас уже є. Саме його і бере на вхід алгоритм (**Algorithm 5**).

```
import networkx as nx

def share_node(block1, block2):
   for node in block1:
      if node in block2:
```

Algorithm 5 Construct block graph $B(G)$ of graph G

Require: Graph G
Ensure: Block graph $B(G)$
1: Set $V(B(G)) := \beta(G)$
2: **for all** pairs of blocks (B_1, B_2) of G **do**
3: **if** $B_1 \neq B_2$ and B_1 and B_2 share a node **then**
4: Add edge $\{B_1, B_2\}$ to $B(G)$
5: **end if**
6: **end for**
7: **return** $B(G)$

```
            return True
    return False

def B(blocks_of_G):
    H = nx.Graph()
    H.add_nodes_from(
        ['B{}'.format(i) for i in range(1, len(blocks_of_G) + 1)]
    )
    for first in range(1, len(blocks_of_G) + 1):
        for second in range(first + 1, len(blocks_of_G) + 1):
            if share_node(blocks_of_G[first - 1], blocks_of_G[second - 1]):
                H.add_edge('B{}'.format(first), 'B{}'.format(second))
    return H
```

Метода *share_node* здійснює перевірку на те, чи пара бльоків має спільну вершину, себто чи з'єднані вони у відповідному графі бльоків. Сам алгоритм бере кожні 2 бльоки й додає ребро в разі присутности спільної вершини. Час роботи такого алгоритму можна теоретично оцінити як $O(|V(G)|^4)$, бо кількість вершин у бльоці, як і кількість бльоків у графі, зростають лінійно від $|V(G)|$. З іншого боку, варто розуміти, що на практиці алгоритм працюватиме значно швидше. Інтуїтивно це пояснюється наступним чином: при сталій кількості вершин зростання кількости бльоків супроводжуватиметься спаданням середньої кількости вершин у бльоці.

3.2 Бльоко-точкові дерева

Означення 3.12. Маючи зв'язний граф G, його *бльоко-точкове дерево (БТД)* означимо наступним чином: $bP(G) = (V(G) \cup \beta(G), \{aB : B \in \beta(G), a \in V(\beta)\})$, себто його множина вершин складається з вершин і бльоків G, а дві вершини $bP(G)$ з'єднані, якщо одна з них відповідає вершині G, а інша бльокові, що її містить.

Приклад 3.13. На **Рис. 6** зображено граф G та його БТД $bP(G)$.

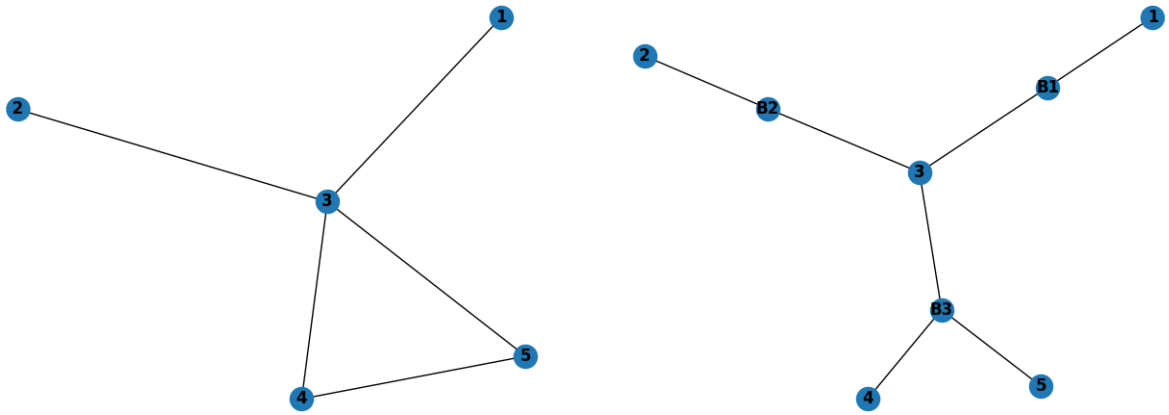

Рис. 6: Граф G (ліворуч) та його БТД $pB(G)$ (праворуч). Бльоки містять такі вершини: $B1 : 1, 3$; $B2 : 2, 3$; $B3 : 3, 4, 5$.

Основні властивості БТД описані в [6].

Теорема 3.14. *БТД дійсно є дерево, себто зв'язний граф без циклів.*

Теорема 3.15. *Нехай $n = |V(G)|$, b_i є кількість бльоків, до яких належить вершина v_i зв'язного графа G. Тоді $|V(pB(G))| = \sum_{i=1}^{n} b_i + 1$, $|E(pB(G))| = \sum_{i=1}^{n} b_i$.*

Теорема 3.16. *Дерево є БТД тоді й тільки тоді, коли що відстань між будь-якими двома кінцевими вершинами парна.*

Бльоко-точкові дерева за своєю побудовою подібні до графів підрозбиттів. Цю подібність підкреслює наступний результат.

Теорема 3.17. *Нехай G є зв'язний граф. Тоді $S(G) \simeq bP(G) \iff G$ є дерево.*

Algorithm 6 Construct block-point tree $bP(G)$ of graph G

Require: Graph G
Ensure: Block-point tree $bP(G)$
1: Set $V(bP(G)) := V(G) \cup \beta(G)$
2: **for all** blocks B of G **do**
3: **for all** nodes v of B **do**
4: Add edge $\{v, B\}$ to $bP(G)$
5: **end for**
6: **end for**
7: **return** $bP(G)$

Наведемо алгоритм знаходження бльоко-точкового дерева (**Algorithm 6**).

```
import networkx as nx

def bP(G, blocks_of_G):
    H = nx.Graph()
    H.add_nodes_from(G.nodes)
    H.add_nodes_from(
        ['B{}'.format(i) for i in range(1, len(blocks_of_G) + 1)]
    )
    for i in range(1, len(blocks_of_G) + 1):
        for node in blocks_of_G[i - 1]:
            H.add_edge(node, 'B{}'.format(i))
    return H
```

Алгоритм ітерується через кожну вершину v кожного бльока B й щоразу додає ребро vB до будованого графа. Зверніть увагу, що алгоритм оптимальний, бо на додавання вершин і ребер витрачається час, обмежений сталою. Оцінити цей час можна як $O(|V(G)|^2)$, хоча, як і у випадку з алгоритмом побудови графа бльоків, на практиці він працюватиме швидше.

3.3 Точко-бльокові графи

Означення 3.18. Дано граф G. Його *точко-бльоковий граф (ТБГ)* $Pb(G) = (V(G) \cup \beta(G), E(B(G)) \cup E(bP(G)))$, тобто граф, вершини якого є вершини

G, і дві вершини u, $v \in V(Pb(G))$ суміжні, якщо виконується одна з двох умов:

1) u, v є бльоки G зо спільною точкою з'єднання;

2) u є бльок G, а v - вершина, що до нього належить.

Приклад 3.19. Граф G та його ТБГ $Pb(G)$ можна побачити на **Рис. 7**.

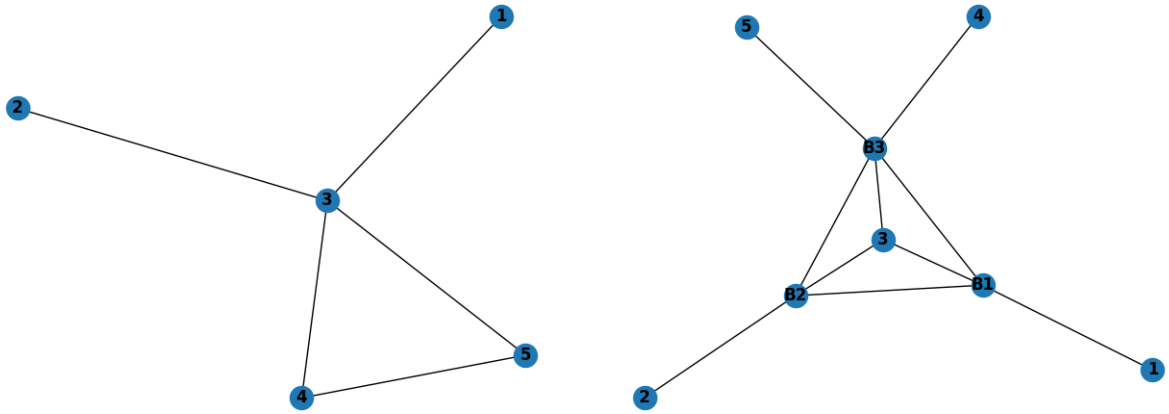

Рис. 7: Граф G (ліворуч) та його ТБГ $Pb(G)$ (праворуч). Бльоки містять такі вершини: $B1 : 1, 3$; $B2 : 2, 3$; $B3 : 3, 4, 5$.

Властивості ТБГ подано в [3].

Твердження 3.20. *Нехай G містить n вершин та m бльоків. Тоді $Pb(G)$ містить n бльоків та m точок з'єднання.*

Твердження 3.21. *Кожен бльок $Pb(G)$ містить єдину вершину G, і кожна вершина G лежить у єдиному бльоці $Pb(G)$.*

Теорема 3.22. *Нехай $n = |V(G)|$, b_i є кількість бльоків, до яких належить вершина v_i зв'язного графа G. Тоді $|V(Pb(G))| = \sum_{i=1}^{n} b_i + 1$, $|E(Pb(G))| = \sum_{i=1}^{n} \frac{b_i(b_i+1)}{2}$.*

Теорема 3.23. *(Характеризація точко-бльокових графів). Для графа G існує граф H такий, що $Pb(H) \simeq G \iff$ для кожного бльока $B \in \beta(G)$ справджуються дві умови:*

- *B повний;*
- *серед вершин B точно одна не є точка з'єднання.*

Наступна теорема пов'язує ТБГ з середущим графом.

Теорема 3.24. *Нехай G є зв'язний граф. Тоді $M(G) \simeq Pb(G) \iff G$ є дерево.*

Поєднання даної операції з узяттям графа бльоків дає на диво красивий результат [3].

Теорема 3.25. *Нехай G є зв'язний граф. Тоді $B(Pb(G)) \simeq G \iff G$ є граф бльоків.*

Довід. $\Longrightarrow$ Очевидно, бо G за твердженням теореми є граф бльоків від $Pb(G)$.

$\Longleftarrow$ За означенням графа бльоків $|V(B(Pb(G)))| = |\beta(Pb(G))|$, себто множина вершин $B(Pb(G))$ та множина бльоків $Pb(G)$ збігаються. Нехай $\phi : V(G) \to V(B(Pb(G)))$ є таке відображення, що переводить вершину a_i графа G у бльок α_i графа $Pb(G)$ (що також є вершина графа $B(Pb(G)))$, який містить a_i, де $i \in [1, |V(G)|]$. Доведемо, що ϕ є ізоморфізм. Із **Твердження 3.21** випливає, що ϕ є бієкція. Лишається показати, що це відображення зберігає ребра. Нехай $a_i a_j \in E(G) \implies \exists A \in \beta(G) : a_{1,2} \in A$, бо якебудь ребро лежить у бльоці. У $Pb(G)$ A є вершина, суміжна з a_i, тому $A \in V(\alpha_i)$. Аналогічно $A \in V(\alpha_j) \implies A \in V(\alpha_i \cap \alpha_j)$, тобто бльоки α_i й α_j мають спільну точку з'єднання $\implies \alpha_i\alpha_j \in E(B(Pb(G)))$. Отже, $a_i a_j \in E(G) \implies \alpha_i\alpha_j \in E(B(Pb(G)))$. Покажемо тепер у другий бік. Нехай $\alpha_i\alpha_j \in E(B(Pb(G))) \implies$ у графі $Pb(G)$ бльоки α_i й α_j містять спільну вершину. Назвемо її A. За характеризацією ТБГ α_i й α_j повні, тим то $a_i A \in E(Pb(G))$ й $a_j A \in E(Pb(G))$. За означенням ТБГ вершини G суміжні в $Pb(G)$ лише з бльоками, що їх містять, тож у G A є бльок такий, що $a_{1,2} \in V(A)$. Оскільки за характеризацією графа бльоків кожен бльок G повний, то з цього випливає, що $a_1 a_2 \in E(G)$. Це ж завершує доведення.

□

Увага 3.26. Насправді у [3] результат звучить дещо по-іншому, а саме, умову "G є граф бльоків" замінено на "кожен бльок G повний". Рівносильність цих двох формулювань випливає з характеризації графа бльоків. Ми

надаємо перевагу наведеному вище формулюванню, бо воно підкреслює схожість результату з отриманим далі.

Увага 3.27. На відміну від формулювання теореми, наведений вище довід авторський; насправді, запропонований у [3] довід спирається на хибне твердження. Точніше, при доведенні достатности авторі показують, що G та $B(Pb(G))$ містять однакову кількість вершин і бльоків, при чім в обох кожен бльок повний. Із цього відразу робиться висновок про ізоморфізм даних графів, хоча в загальному випадку такої інформації для цього недосить. Контрприклад зображено на **Рис. 8** (цей рисунок одержаний без використання Python).

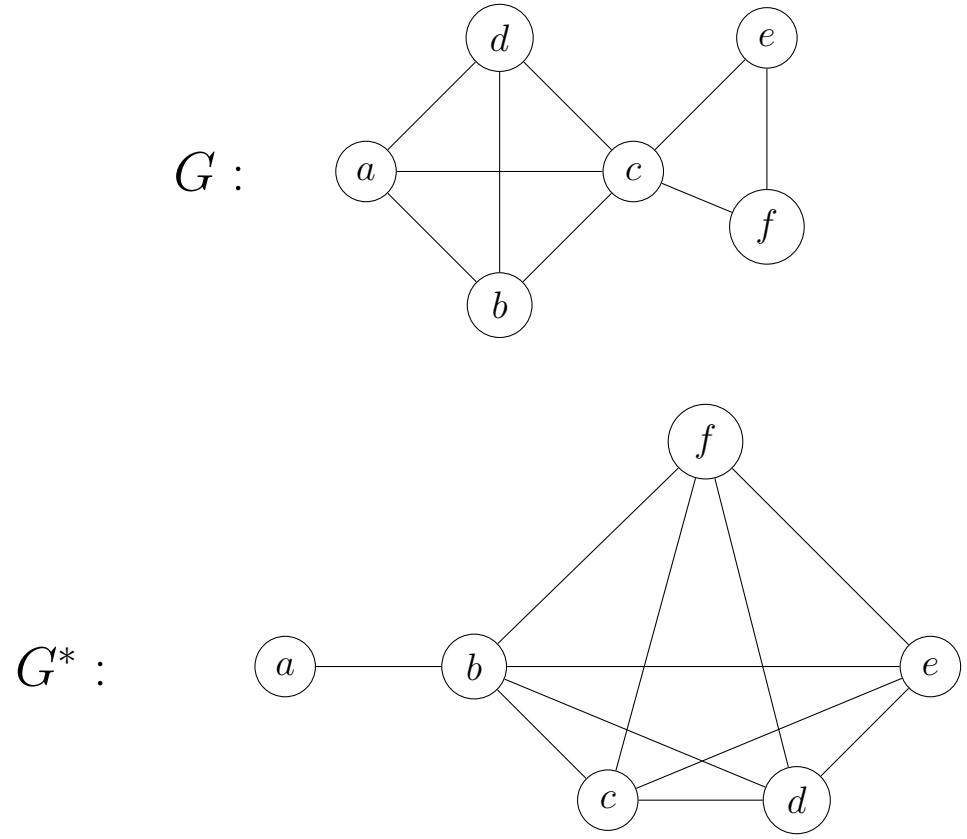

Рис. 8: Графи G та G^* містять по 6 вершин, по 2 бльоки, і обидва бльоки кожного графа повні. Втім, $G \not\simeq G^*$.

Наслідок 3.28. $B(Pb(B(G))) \simeq B(G)\ \forall G$.

Довід. Нехай $H \simeq B(G)$. За **Теоремою 3.25** $B(Pb(H)) \simeq H \iff H$ є граф бльоків. Але H є граф бльоків від графа G, тож твердження наслідку виконується завсіди. □

Теорема 3.29. *$Pb(B(G)) \simeq G \iff G$ є ТБГ.*

Зверніть увагу, що даний результат можна розглядати як дуальний до попереднього: власне, ми тільки поміняли порядок узяття операторів у лівій частині, і "граф бльоків" на "ТБГ" у правій.

Довід. $\Longrightarrow$ Очевидно, бо G є ТБГ від $B(G)$.

$\Longleftarrow$ Нехай $C(G)$ є множина точок з'єднання графа G, c_i є її елементи, а v_i є елементи множини $V(G) \setminus C(G)$. За характеризацією ТБГ, кожен бльок G містить точно одну вершину, що не є точка з'єднання $\Longrightarrow$ можна природно встановити взаємно однозначну відповідність між v_i та бльоками B_{v_i}, що їх містять. Оскільки в $B(G)$ бльоки B_{v_i} є вершини, а c_i відповідають бльокам, то існує бієкція $\phi : V(G) \to V(B(G)) \cup \beta(B(G))$ така, що $\phi(v_i) = B_{v_i}$, а $\phi(c_i)$ - це бльок $B(G)$, вершини якого є бльоки G, що містять точку з'єднання c_i. З означення $Pb(G) : V(Pb(B(G))) = V(B(G)) \cup \beta(B(G))$, тож альтернативно можна записати $\phi : V(G) \to V(Pb(B(G)))$ є бієкція між множинами вершин графів G та $Pb(B(G))$. Покажемо, що ϕ є ізоморфізм. Для цього слід довести, що ϕ зберігає ребра. Розглянемо 3 випадки:

1) Ребро між двома точками з'єднання: $c_i c_j \in E(G) \iff \exists B \in \beta(G) : c_{i,j} \in V(B) \iff$ у графі $B(G)$: $V(\phi(c_i)) \cap V(\phi(c_j)) \neq \emptyset \iff \phi(c_i)\phi(c_j) \in E(Pb(B(G)))$. Перша з рівносильностей виконується, бо за характеризацією ТБГ кожен бльок G повний.
2) Одна з вершин ребра є точка з'єднання, а інша - ні: $c_i v_j \in E(G) \iff c_i \in B_{v_j} \iff \phi(v_j) \in V(\phi(c_i)) \iff \phi(c_i)\phi(v_j) \in E(Pb(B(G)))$. Знов таки, перша рівносильність є наслідок повноти кожного бльока G.
3) $v_i v_j \notin E(G)$, бо за характеризацією ТБГ кожен бльок G містить єдину вершину, що не є точка з'єднання.

Отже ϕ є ізоморфізм графів, що завершує доведення.

□

Наслідок 3.30. *$Pb(B(Pb(G))) \simeq Pb(G)$ $\forall G$.*

Довід аналогічний до наведеного в **Наслідкові 3.28**, тільки з опертям на **Теорему 3.29**.

Увага 3.31. Наслідок 3.28 до **Теореми 3.25** можна пов'язаний із **Теоремою 3.29**: власне, він твердить, що якщо в обох частинах ізоморфізму **Теореми 3.29** "накинути" бльоковий оператор, то одержимо вираз, правдивий уже для будь-якого графа G. Насправді, граф $Pb(B(G))$ можна одержати з G, якщо в кожному бльокові залишити лиш одну вершину-не точку з'єднання (або додати, якщо такої немає), і "заповнити" бльок (себто сполучити кожні 2 його вершини). Зверніть увагу, що ці маніпуляції не впливають на $B(G)$.

Подібним чином **Наслідок 3.30** до **Теореми 3.29** пов'язаний із **Теоремою 3.25**: сього разу до обох частин іще раз застосовуємо точко-бльоковий оператор. Інтуїція наступна: $B(Pb(G))$ отримується з G "заповненням" кожного бльока, що не впливає на результат узяття оператора Pb.

Насамкінець наведемо алгоритм побудови $Pb(G)$, знаючи G і явно маючи список його бльоків (**Algorithm 7**).

Algorithm 7 Construct point-block graph $Pb(G)$ of graph G

Require: Graph G
Ensure: Point-block graph $Pb(G)$
1: Construct block graph $B(G)$ and block-point tree $bP(G)$
2: Set $V(Pb(G)) := V(bP(G))$
3: Set $E(Pb(G)) := E(B(G)) \cup E(bP(G))$
4: **return** $Pb(G)$

```
import networkx as nx

def Pb(G, blocks_of_G):
    block_graph = B(blocks_of_G)
    block_point_tree = bP(G, blocks_of_G)
    H = nx.Graph()
    H.add_nodes_from(block_point_tree.nodes)
    H.add_edges_from(block_graph.edges)
    H.add_edges_from(block_point_tree.edges)
    return H
```

Алгоритм використовує те, що $V(Pb(G)) = V(bP(G))$, а $E(Pb(G)) = E(B(G)) \cup E(bP(G))$. Час його роботи можна оцінити як $O(|V(G)|^4)$.

3.4 Бльоко-тотальні графи

Означення 3.32. Маємо граф G. Його *бльоко-тотальний граф (БТГ)* $bT(G)$ є граф такий, що $V(bT(G)) = V(G) \cup \beta(G)$, а вершини $u, v \in V(bT(G))$ суміжні в кожному з наступних випадків:

- u та v є суміжні вершини графа G;
- u є вершина G, а v - бльок, що її містить;
- u й v є бльоки G зо спільною точкою з'єднання.

Приклад 3.33. Приклад графа G та його БТГ $bT(G)$ на **Рис. 9**.

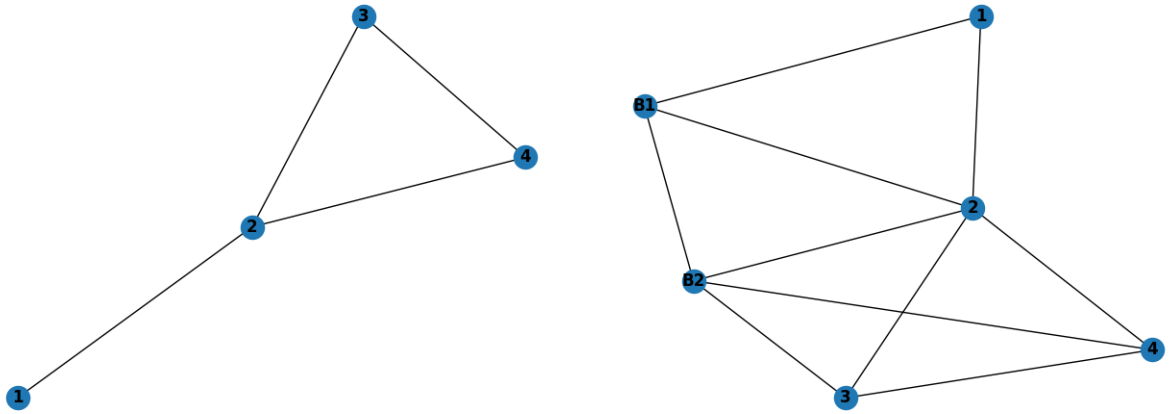

Рис. 9: Граф G (ліворуч) та його БТГ $bT(G)$ (праворуч). Бльоки G містять такі вершини: $B1 : \{1, 2\}$, $B2 : \{2, 3, 4\}$.

Теорема 3.34. *Нехай G є зв'язний граф. Тоді $bT(G) \simeq (bP(G))^2$.*

Довід. Очевидно, що $V((bP(G))^2) = V(bT(G))$. З'ясуємо, чи вершини u й v графа $(bP(G))^2$ суміжні за допомогою означення графа у квадраті:

- $d_{bP(G)}(u, v) = 1 \implies uv \in E(bP(G)) \implies u$ є вершина графа G, а v - бльок, що містить u (або навпаки) $\implies uv \in E(bT(G))$.
- $d_{bP(G)}(u, v) = 2$. Тоді виконується один із двох варіянтів: або u та v є вершини G, що лежать у спільному бльокові, або ж це бльоки графа G, що містять спільну точку з'єднання. В обох випадках $uv \in E(bT(G))$.

Якщо навпаки $uv \in E(bT(G))$, то схожими міркуваннями можна отримати, що або $d_{bP(G)}(u,v) = 1$, або $d_{bP(G)}(u,v) = 2$. Тож дійсно $bT(G) \simeq (bP(G))^2$. □

Цей граф пов'язаний із тотальним графом $T(G)$ наступною теоремою.

Теорема 3.35. *Маємо граф G. Тоді $bT(G) \simeq T(G) \iff G$ є дерево.*

Довід. $\Longrightarrow$ Із $|V(T(G))| = |V(bT(G))|$, враховуючи означення графа підрозбиттів і БТД, робимо висновок, що $|E(G)| = |\beta(G)|$. Оскільки кожен бльок містить хоча б одне ребро, ми можемо стверджувати, що в G бльоки й ребра еквівалентні, тим то G є дерево.

$\Longleftarrow$ У дереві $E(G) = \beta(G)$, тому твердження теореми прямо випливає з означень $T(G)$ й $bT(G)$.

□

Як і з усіма попередніми конструкціями, покажемо алгоритм побудови бльоко-тотального графа (**Algorithm 8**).

Algorithm 8 Construct block-total graph $bT(G)$ of graph G

Require: Graph G
Ensure: Block-total graph $bT(G)$
1: Construct point-block graph $Pb(G)$
2: Set $V(bT(G)) := V(Pb(G))$
3: Set $E(bT(G)) := E(Pb(G)) \cup E(G)$
4: **return** $bT(G)$

```
import networkx as nx

def bT(G, blocks_of_G):
    point_block_graph = Pb(G, blocks_of_G)
    H = nx.Graph()
    H.add_nodes_from(point_block_graph.nodes)
    H.add_edges_from(G.edges)
    H.add_edges_from(point_block_graph.edges)
    return H
```

Цей алгоритм спирається на наступні факти: $V(bT(G)) = V(Pb(G))$ та $E(bT(G)) = E(G) \cup E(Pb(G))$. Складність його роботи - $O(|V(G)|^4)$.

На завершення розділу продемонструємо зв'язок між реберними та бльоковими конструкціями за допомогою наступної таблиці.

Пари суміжних елементів	Реберна конструкція	Пари суміжних елементів	Бльокова конструкція
ребра-ребра	реберний граф $L(G)$	бльоки-бльоки	граф бльоків $B(G)$
вершини-ребра	граф підрозбиттів $S(G)$	вершини-бльоки	БТД $bP(G)$
ребра-ребра, вершини-ребра	середущий граф $M(G)$	бльоки-бльоки, вершини-бльоки	ТБГ $Pb(G)$
ребра-ребра, вершини-ребра, вершини-вершини	тотальний граф $T(G)$	бльоки-бльоки, вершини-бльоки, вершини-вершини	БТГ $bT(G)$

4 Графи клік

Остання графова конструкція, яку ми розглянемо в межах цієї роботи, є граф клік.

Означення 4.1. *Кліко*ю називається максимальний за включенням повний підграф.

Множину клік графа G означатимемо як $\mathcal{K}(G)$.

Означення 4.2. Дано граф G. Його *граф клік* є граф перетинів $K(G) = \Omega(V(G), \mathcal{K}(G))$.

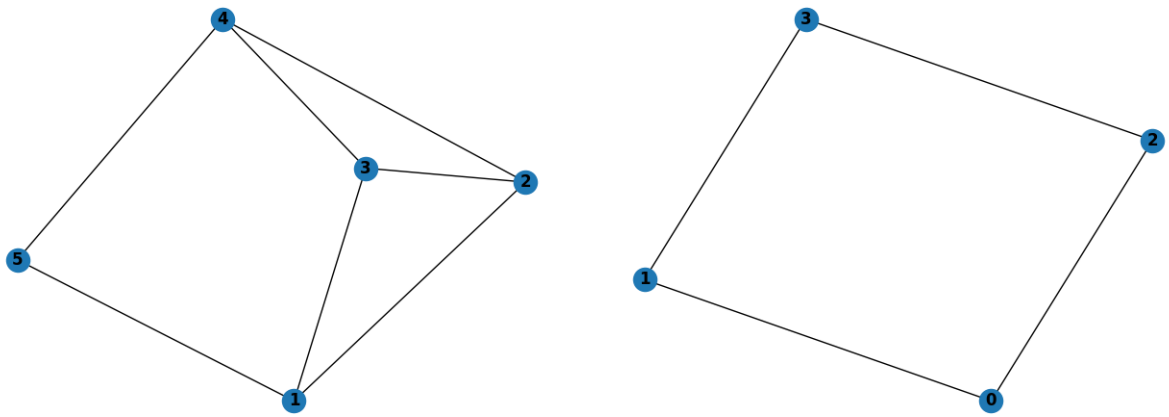

Рис. 10: Граф G (ліворуч) та його граф клік $K(G)$ (праворуч). Кліки G містять такі вершини: $0 : \{1, 2, 3\}$, $1 : \{2, 3, 4\}$, $2 : \{4, 5\}$, $3 : \{1, 5\}$.

Граф клік тісно пов'язаний із графами, описаними в попередніх секціях. Зокрема, наступний факт описує його зв'язок із графом бльоків.

Теорема 4.3. *Дано нетривіяльний граф G. Тоді $B(G) \simeq K(G) \iff G$ є граф бльоків.*

Довід. $\implies$ Оскільки кліка є двозв'язний підграф, то $\forall C \in \mathcal{K}(G)\ \exists! B \in \beta(G) : V(C) \subset V(B)$. Із того, що $|\beta(G)| = |\mathcal{K}(G)|$, робимо висновок, що $\forall B \in \beta(G)\ \exists! C \in \mathcal{K}(G) : V(C) \subset V(B)$. Отже існує взаємно однозначна відповідність поміж кліками графа G та бльоками, що їх містять. Враховуючи, що кліки покривають усі ребра G, можна стверджувати, що кожна кліка дорівнює відповідному бльокові (інакше ребро, яке не належить до цієї кліки, не може належати до жадної

кліки в G). Отже кожний бльок є повний підграф, тим то G є граф бльоків.

$\Longleftarrow$ У цім разі кожен бльок повний. Додавши до бльока B вершину $v \in V(G)\backslash V(B)$ одержимо підграф, що не є двозв'язним, а отже, не є повним. Тому B є максимальний за включенням повний підграф, тим то він є кліка. Інших клік бути не може, бо кожна кліка лежить у якімсь бльокові, тому з її максимальности слідує, що вона має збігатися з бльоком. Отже $B(G) \simeq K(G)$.

□

Також граф клік пов'язаний із реберним графом. Зокрема, за його допомоги можна описати реберні графи двочасткових графів.

Означення 4.4. Граф G називається *двочастковим*, якщо $V(G) = A \sqcup B$, де $\forall e \in E(G) : e = uv$ при $u \in A$ та $v \in B$.

Для нас зараз важлива характеризація двочасткових графів [1].

Теорема 4.5. *Граф G двочастковий $\Longleftrightarrow$ G не містить циклів непарної довжині.*

Теорема 4.6. *(Характеризація реберних графів двочасткових графів). Для графа G існує двочастковий граф $H : L(H) \simeq G$ тоді й тільки тоді, коли:*

1) *$K(G)$ двочастковий;*

2) $\forall C_{1,2} \in \mathcal{K}(G) : |V(C_1 \cap C_2)| \leq 1$.

Довід. $\Longrightarrow$ Впочатку покажемо, що виконується 2). Зробимо це від супротивного. Нехай $\exists C_{1,2} \in \mathcal{K}(G), C_1 \neq C_2 : \exists u, v \in V(C_1 \cap C_2), u \neq v$. Оскільки C_1 і C_2 породжують повні підграфи, то $uv \in E(G)$. Із $C_1 \neq C_2$ маємо, що $\exists w_1 \in V(C_1)\backslash V(C_2)$ та $\exists w_2 \in V(C_2)\backslash V(C_1)$. Очевидно, що $uw_1, uw_2, vw_1, vw_2 \in E(G), w_1w_2 \notin E(G)$. Отже в нас 2 цикли довжині 3 зо спільним ребром. Граф G реберний, тож для нього існує вершинове покриття F, для якого виконуються умови **Теореми 2.5**. У такому разі один із циклів, а також 2 інші ребра з другого циклу лежатимуть у трьох різних елементах F. Без обмеження загальности, скажемо, що $u, v, w_1 \in A \in F$, $u, w_2 \in B \in F$ та $v, w_2 \in C \in F$. Тоді A, B й C є попарно суміжні вершини графа H. Тому H містить цикл

непарної довжині 3, тим то він не двочастковий. Суперечність.
Покажемо, що виконується 1). Оскільки G не містить 3-циклів, то $\forall C \in \mathcal{K}(G)\ \exists u \in V(H) : V(C) = E_H(u)$. Знову підемо від супротивного. Нехай $K(G)$ містить цикл $C_1 - C_2 - ... - C_n - C_1$, $2 \nmid n$, при чім $V(C_i) = E_H(u_i)$. Тоді H містить цикл $u_1 - u_2 - ...u_n - u_1$ непарної довжині, тож H не двочастковий. Суперечність.

$\Longleftarrow$ Спочатку доведемо, що G є реберний граф. Для цього покажемо, що множина $\mathcal{F} \stackrel{def}{=} \mathcal{K}(G) \cup \{\{x\} : \exists! C \in \mathcal{K}(G)\ x \in V(C)\}$ задоволяє умови **Теореми 2.5**:

- $\forall A \in \mathcal{F} : G[A]$ повний за побудовою $\mathcal{F}$;
- Якщо $\exists v \in V(G)\ \exists A, B, C \in \mathcal{K}(G) : v \in V(A \cap B \cap C)$, то A, B й C утворюють цикл довжині 3 в $K(G)$, тому $K(G)$ не двочастковий. Отже вершина G лежить щонайбільше у двох кліках, із чого, у свою чергу, маємо $\forall v \in V(G) : |\mathcal{F}_v| = 2$.
- Нехай $\exists uv \in E(G)\ \exists C_{1,2} \in \mathcal{K}(G) : uv \subset V(C_1 \cap C_2)$. Тоді $|V(C_1 \cap C_2)| \geq 2$, що суперечить умові 2). Тож $\forall e \in E(G)\ \exists! A \in \mathcal{F} : e \subset A$.

Отже $\exists H : L(H) \simeq G$. Доведемо, що H двочастковий. Для цього зафіксуємо найменший непарний цикл $C = v_1 - v_2 - ... - v_n - v_1$ та розглянемо два випадки:

- Нехай C лежить у деякій кліці в H. Тоді з мінімальности C відразу одержуємо, що $n = 3$. Якщо C є компонента зв'язности в H, то $\exists H'$, який одержується з H заміною C на $K_{1,3}$ та для якого $L(H') \simeq L(H) \simeq G$. Нехай тоді C не є компонента зв'язности. Тоді $\exists u \in V(H) \backslash V(C)\ \exists i \in [1, 3] : v_i u \in E(H)$. Без обмеження загальности вважатимемо, що $v_1 u \in E(H)$. У такому разі $G \simeq L(H)$ містить два 3-цикли $v_1 u - v_1 v_2 - v_1 v_3 - v_1 u$ та $v_1 v_2 - v_2 v_3 - v_1 v_3 - v_1 v_2$ зо спільним ребром $\{v_1 v_2, v_1 v_3\}$. Очевидно, що вони належать до двох різних клік $C_{1,2} \in \mathcal{K}(G)$, тим то $|V(C_1 \cap C_2)| \geq 2$, що суперечить умові 2).
- Отже жадна кліка не містить C повністю. Тоді в G розглянемо цикл $C' \stackrel{def}{=} L(C) = v_1 v_2 - v_2 v_3 - ... - v_{n-1} v_n - v_n v_1$. Його довжиня також дорівнює n. За умовою 2) кожне ребро лежить точно в одній кліці. Покажемо від супротивного, що всі ребра C' лежать у

різних кліках. Дійсно, якби два суміжні ребра $\{v_iv_{i+1}, v_{i+1}v_{i+2}\}$ та $\{v_{i+1}v_{i+2}, v_{i+2}v_{i+3}\}$ лежали в одній кліці, то з повноти кліки ми б мали $\{v_iv_{i+1}, v_{i+2}v_{i+3}\} \in E(G)$. Отже серед $v_i, v_{i+1}, v_{i+2}, v_{i+3}$ були б дві однакові вершини, тож довжиня циклу C' (а отже й C) була б три й C лежав би в одній кліці. Якби ж два несуміжні ребра лежали в одній кліці, то вони б породжували повний підграф на 4 вершини. Отже відповідні 4 ребра були б попарно суміжні в H, тож вони не могли б лежати на простому циклові. Але оскільки C мінімальний, то він має бути простим. У кожному разі одержали суперечність. Тому всі ребра в C' дійсно належать до різних клік, а отже, відповідні кліки утворюють цикл довжині n у $K(G)$, що суперечить його двочастковості, бож $2 \nmid n$. Тому H двочастковий.

□

Увага 4.7. Бібліотека NetworkX мови програмування Python має методу $make\_max\_clique\_graph$, що якраз повертає граф клік для даного графа.

5 Висновки

У цій роботі розглянуто 3 важливі типи графових конструкцій, які є результати взяття певних графових операторів, що переводять деякі підграфи початкового графа у вершини нового. У залежності від типів підграфів, із якими працюємо, виділяємо реберні, бльокові та клікові конструкції.

Розглянуто 4 різні реберні конструкції. Навдено основні властивості кожної з них, окрему увагу виділено не взаємозв'язки. Для кожної з конструкцій запропоновано алгоритм її побудови, подано його псевдокод, а також реалізовано мовою Python.

Аналогічно розглянуто 4 бльокові конструкції. Для кожної з них наведено основні властивості й алгоритм побудови. Досліджено композиції взяття бльокового й точко-бльокового операторів. З аналогії з реберними конструкціями введено поняття бльоко-тотального графа, досліджено його основні властивості. Пояснено зв'язки між бльоковими та реберними конструкціями.

У кінці описано граф клік, його основні властивості та зв'язки з попередньо розглянутими конструкціями. Зокрема, наведено характеризацію графів, граф клік яких ізоморфний їхньому графові бльоків.

Література

[1] F. Harary, *Graph Theory (1969).*

[2] K. Paton, *An Algorithm for the Blocks and Cutnodes of a Graph, Communications of the ACM (1971), Vol. 14,* ***7****.*

[3] V. R. Kulli, M. S. Biradar, *The Point Block Graph of A Graph, Journal of Computer and Mathematical Sciences Vol. 5, Issue 5 (2014), 476-481.*

[4] T. Hamada, I. Yoshimura, *Traversability and Connectivity of the Middle Graph of a Graph, Discrete Mathematics 14 (1976) 247-255.*

[5] M. Behzad, *A Criterion for the Planarity of the Total Graph, Proc. Camb. Phil. Soc.,* ***63****, 679-681.*

[6] V. R. Kulli, *The Block-Point Tree of a Graph, Indian Journal of Pure and Applied Mathematics (1976), Vol. 7,* ***6****.*

[7] F. Harary, *A Characterization of Block Graphs, Canad. Math. Bull.* **6** *(1963), 1-6.*

[8] Terry A. McKee, F. R. McMorris (1999), *Topics in intersection graph theory*, Society for Industrial and Applied Mathematics.

[9] E. Howorka, On metric properties of certain clique graphs, *J. Combin. Theory Ser. B* **27** (1979), 67–74.