Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ”
Факультет інформатики
Кафедра математики

Кваліфікаційна робота
освітній ступінь — бакалавр

на тему: **“ЦЕНТР ТА ПЕРИФЕРІЯ ГРАФА”**

Виконав студент 4-го року навчання
освітньої програми “Прикладна математика”
спеціальності 113 Прикладна математика
Гапоненко Владислав Олександрович

Керівник: Козеренко С.О.,
кандидат фіз.-мат. наук, ст. викладач

Рецензент _____
(прізвище та ініціали)

Кваліфікаційна робота захищена
з оцінкою _____

Секретар ЕК _____
(підпис)
“_____”_____ 2021 р.

Київ – 2021

Зміст

1 Вступ

Центральність графа є фундаментальним поняттям теорії графів, що застосовується в багатьох наукових сферах. Наприклад, якщо розглянути певний словник, ввести метрику на словах таким чином, щоб відстань між словами була тим більшою чим їх непов'язаність за сенсом, то найбільш загальні слова сформують центр даного графа. Проблема розміщення об'єктів таким чином, щоб до них можна було швидко дібратися з любої точки певної локації, також вирішується завдяки центральності. Дуальне поняття до центральності є периферійність. Вона знаходить застосування за необхідності досягти прямо протилежний результат в описаних вище задачах.

Проблема знаходження центру є багатою на дослідження темою. Далеко не останню роль в нашому дослідженні грає результат Каміля Джордана, що центр дерева складається з однієї або двох сусідніх вершин. Відомим також є результат Френка Харарі та Роберта Нормана, що центр зв'язного графа знаходиться в блоці.

Метою дослідження є розробка та реалізація алгоритму знаходження центру та периферії певної будівлі, що задана простим багатокутником. Мета роботи зумовила наступні наукові **завдання**:

1. Дослідити радіус та діаметр зв'язного графа.
2. Дослідити центр та периферію деяких класів графів.
3. Реалізувати алгоритм триангуляції багатокутника.
4. Реалізувати алгоритм знаходження центру та периферії дуального графа триангульованого багатокутника.

Робота складається з чотирьох частин. В розділі про діаметр та радіус зв'язного графа ми досліджуємо згадані раніше поняття в різних класах графів. Також, ми описали центр та периферію певних класів графів.

В розділі про відносні центри ми дослідили зв'язок між центрально критичними графами та згаданими в минулому розділі самоцентральними унікально ексцентричними графами.

В 5 розділі ми описали алгоритм знаходження центру та периферії простого багатокутника. В 6 розділі ми навели пояснення до програмної реалізації алгоритму та показали роботу алгоритму на прикладі.

2 Основні означення та попередні результати

2.1 Означення

Ми розглядаємо прості зв'язні неорієнтовані графи. Відстанню $d(a,b)$ між вершинами графа $a,b \in V(G)$ ми називаємо найкоротший шлях, що їх з'єднує. *Ексцентриситетом* вершини a називатимемо число $\mathrm{ecc}(a) = \max\{d(a,b)|b \in V(G)\}$. Якщо для $u,v \in V(G)$ виконується $d(u,v) = \mathrm{ecc}(u)$, то ми кажемо, що v є *ексцентричною вершиною* для u. *Радіусом* $\mathrm{rad}(G)$ графа називається найменший ексцентриситет його вершин. Протилежним поняттям є *діаметр* $\mathrm{diam}(G)$, що дорівнює найбільшому ексцентриситету серед вершин графа. Відповідно *центром графа*

$$\mathrm{Cen}(G) = \{v \in V(G)|\, \mathrm{ecc}(v) = \mathrm{rad}(G)\}$$

є множина вершин з найменшим ексцентриситетом, а *периферією*

$$\mathrm{Per}(G) = \{v \in V(G)|\, \mathrm{ecc}(v) = \mathrm{diam}(G)\}$$

називатимемо множину вершин з найбільшим ексцентриситетом. Проте, у випадку, коли центр графа співпадатиме з множиною його вершин, тобто $\mathrm{Cen}(G) = V(G)$, граф буде *самоцентральним*.

Граф називатимемо *повним*, коли кожна його пара різних вершин є суміжною. Множина вершин графа називається незалежною, якщо жодні дві не є суміжними в ньому. Граф G є *повним двочастковим*, якщо множина його вершин розділяється на дві незалежні множини $V(G) = V_1 \sqcup V_2$ таким чином, що кожна вершина $v \in V_1$ є суміжною з усіма $u \in V_2$ і навпаки.

Любий максимальний двозв'язний підграф графа G називається *блоком*. Те що граф є двозв'язним значить, що потрібно видалити як мінімум дві вершини, щоб він розпався на різні компоненти зв'язності. Вершину, видаливши яку ми збільшимо кількість компонент зв'язності, називають *скороченною*. Відповідно можемо навести альтернативне означення блоку. Максимальний породжений підграф, що не містить скорочених вершин називається блоком. Ми називатимемо граф G *графом блоків*, якщо кожний блок є повним.

Самодоповняльним графом є граф, що ізоморфний до свого доповнення.

Для підмножини вершин $S \subset V(G)$ та вершини $v \in V(G)$, $\max\limits_{x \in S} d(x,v)$ називається $S-$*ексцентриситетом*. $S-$ексцентриситет вершини v надалі

позначатимемо як $\mathrm{ecc}_S(v)$. Множина вершин з мінімальним $S-$ексцентриситетом називається $S-$*центром*, ми її позначатимемо як $\mathrm{Cen}_S(G)$. Люба множина вершин A, що є $S-$центром хоч для одної множини S, називається *центральною множиною*.

Простим багатокутником є многокутник без самоперетинів та дірок. Многокутник називається *y-монотонним* (*x-монотонним*), коли горизонтальна (вертикальна) пряма перетинає багатокутник щонайбільше раз. *Триангуляція многокутника* – це розкладання простого многокутника на трикутники. Оскільки кожне ребро може одночастно належати двом граням, зручніше використовувати поняття *напівребра*. Кожне ребро ми розглядатимемо як два різнонаправлених відрізки (напівребра), що інцидентні суміжним граням. Напівребра є *близнюками*, якщо їм відповідає однакове ребро.

3 Діаметр та радіус зв'язного графа

Твердження 3.1. *Для кожного зв'язного графа G має місце нерівність* $\mathrm{rad}(G) \leq \mathrm{diam}(G) \leq 2\,\mathrm{rad}(G)$.

Доведення. Оцінка, що $\mathrm{rad}(G) \leq \mathrm{diam}(G)$ очевидна за означенням. Завжди існуватимуть такі $x, y \in V(G)$, що $d(x, y) = \mathrm{diam}(G)$. Зафіксуємо таку вершину $v \in V(G)$, що $v \in \mathrm{Cen}(G)$. З означення ексцентриситету вершини та радіусу графа слідує, що $\mathrm{ecc}(v) = \mathrm{rad}(G)$. Далі з нерівності трикутника $d(x, y) \leq d(x, v) + d(v, y) \leq \mathrm{ecc}(v) + \mathrm{ecc}(v) = 2\,\mathrm{rad}(G)$ слідує доведення твердження. □

Очевидно, що ці тривіальні оцінки досягаються, наприклад, на циклі C_n та на непарному ланцюгу P_{2k+1}. Насправді, окрім нерівностей із Твердження 3.1, іншого зв'язку між радіусом та діаметром зв'язного графа загального вигляду не існує.

Теорема 3.2. *Для кожної пари чисел $r, d \in \mathbb{N}$ із $r \leq d \leq 2r$ існує граф G такий, що* $\mathrm{rad}(G) = r$, $\mathrm{diam}(G) = d$.

Доведення. Для випадку, коли $\mathrm{diam}(G) = \mathrm{rad}(G)$ підійде граф C_{2d}, що є циклом із кількістю вершин, що співпадає з подвоєним діаметром графа. Тоді для кожної вершини її ексцентриситет дорівнюватиме відстані до протилежної вершини в циклі.

Для випадку, коли $\mathrm{diam}(G) = 2\,\mathrm{rad}(G)$ підійде граф P_{2r+1}. Такий граф є ланцюгом, де в центрі знаходиться одна вершина, з чого випливає рівність $\mathrm{diam}(G) = 2\,\mathrm{rad}(G)$.

Розглянемо випадок, коли $\mathrm{diam}(G) = \mathrm{rad}(G)+k$. Для $k = 1$ (див. Рис. 1) розглянемо граф $G = C_{2r}$ і зафіксуємо вершини $v_0, v \in C_{2r}$, що є протилежними. Додамо до $V(G)$ вершину u_1, шо буде суміжною тільки з v_0. Тоді справедливими будуть рівності $d(v, u_1) = \mathrm{diam}(G) = \mathrm{rad}(G) + 1 = d(v, v_0)$. Для будь-яких інших k додаватимемо вершину u_k, що буде суміжною тільки з вершиною u_{k-1} і таким чином будуть справедливими рівності $d(v, u_k) = \mathrm{diam}(G) = \mathrm{rad}(G) + k = d(v, v_0)$. □

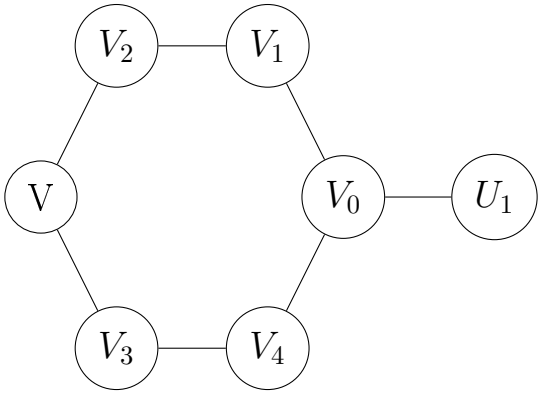


Рис. 1: Випадок $\mathrm{diam}(G) = \mathrm{rad}(G) + 1$

Очевидно, що доповнення до незв'язного графа завжди є зв'язним графом. Зокрема, кожен самодоповняльний граф є зв'язним. Наступні результати встановлюють зв'язок між діаметрами зв'язного графа і його доповнення.

Теорема 3.3. *Нехай G зв'язний граф із $\mathrm{diam}(G) \geq 3$. Тоді його доповнення $\overline{G}$ теж є зв'язним графом, причому $\mathrm{diam}(\overline{G}) \leq 3$.*

Доведення. Припустимо, що $\overline{G}$ є незв'язним. Тоді граф можна буде розбити на різні компоненти зв'язності, нехай такими компонентами будуть $\overline{G_1}$ та $\overline{G_2}$. Тоді для всіх таких вершин $u, v \in V(\overline{G_1})$ $d_G(u, v) \leq 2$, а для всіх вершин $a \in V(\overline{G_1})$ та $b \in V(\overline{G_2})$ виконуватиметься $d_G(a, b) = 1$. Тобто маємо суперечність з фактом, що $\mathrm{diam}(G) \geq 3$ для G.

Зафіксуємо вершини $v, u \in V(G)$, що $d_G(v, u) = 3$. Дослідимо відстань між двома іншими вершинами $a, b \in V(G)$ в графі $\overline{G}$. Підкреслимо, що $d_{\overline{G}}(v, u) = 1$ та ні a, ні b не може бути суміжною з u або v одночасно, бо це суперечить $d_G(v, u) = 3$. Якщо a та b є несуміжними в G, то тривіально $d_{\overline{G}}(a, b) = 1$. В подальшому вважатимемо $d_G(a, b) = 1$. Не втрачаючи загальності, припустимо, що a та b одночасно не суміжні з u G. Тоді $d_{\overline{G}}(a, b) = 2$, тобто a та b будуть з'єднані шляхом через v. Знову ж, не втрачаючи загальності, припустимо, що u суміжна з b, а вершина v суміжна з a. Тоді $d_{\overline{G}}(a, b) = 3$, тобто шлях ab в $\overline{G}$ проходитиме через пару суміжних вершин u та v. З цього випливає твердження, що для любих $a, b \in V(G)$ виконується $d_{\overline{G}}(a, b) \leq 3$. □

Наслідок 3.4. *Кожен нетривіальний самодоповняльний граф має діаметр 2 або 3.*

Доведення. Якщо діаметр буде рівним одиниці, тоді граф G буде повним, а його доповнення — це ізольовані точки. Якщо діаметр буде більшим за 3, тоді за теоремою 3.3 доповнення матиме менший діаметр, тож не вийде побудувати бієкцію зі збереженням суміжності між G та $\overline{G}$. □

Приклад 3.5.

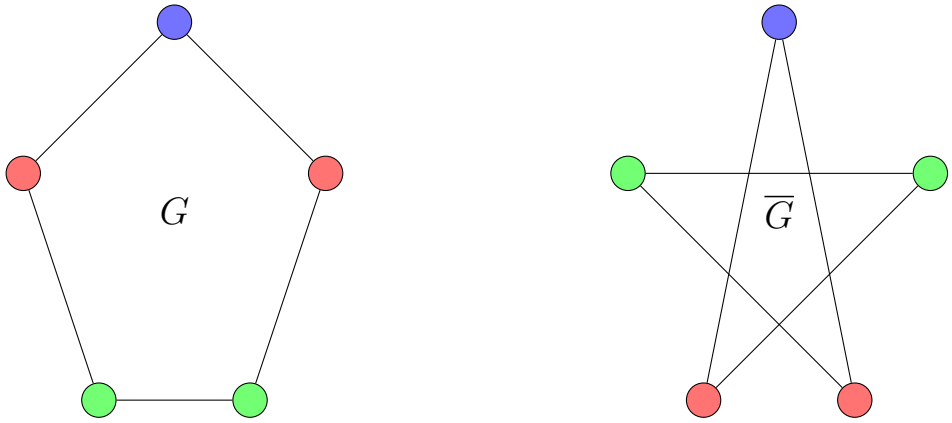


Рис. 2: Самодоповняльний граф G з $\mathrm{diam}(G) = 2$, та $\overline{G}$

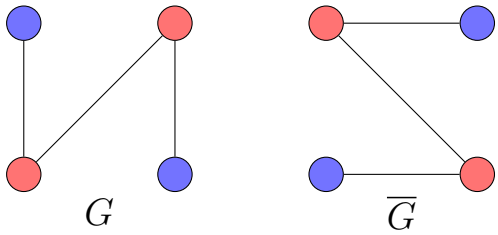


Рис. 3: Самодоповняльний граф G з $\mathrm{diam}(G) = 3$, та $\overline{G}$

Твердження 3.6. *Нехай G зв'язний граф із* $\mathrm{diam}(G) \geq 4$. *Тоді його доповнення $\overline{G}$ теж є зв'язним графом, причому* $\mathrm{diam}(\overline{G}) \leq 2$.

Доведення. Зафіксуємо вершини $v, u \in V(G)$, що $d_G(v, u) = 4$. Дослідимо відстань між двома іншими вершинами $a, b \in V(G)$ в графі $\overline{G}$. Підкреслимо факт, що $d_{\overline{G}}(v, u) = 1$. Якщо a та b є несуміжними в G, то тривіально $d_{\overline{G}}(a, b) = 1$. В подальшому вважатимемо $d_G(a, b) = 1$. Не втрачаючи загальності, припустимо, що a та b одночасно не суміжні з u G. Тоді $d_{\overline{G}}(a, b) = 2$, тобто a та b будуть з'єднані шляхом через v. Знову ж, не втрачаючи загальності, припустимо, що u суміжна з b, а вершина v суміжна з a. Тоді отримаємо суперечність з $d_G(v, u) = 4$. Тож $d_{\overline{G}}(a, b) \leq 2$. □

Теорема 3.7. *Нехай G самоцентральний граф із* $\mathrm{diam}(G) \geq 3$. *Тоді* $\overline{G}$ *теж самоцентральний, причому* $\mathrm{diam}(\overline{G}) = 2$.

Доведення. Розглянемо $u, v \in V(G) : d_G(u, v) = 1$. Зафіксуємо вершину $w \in V(G) : d(u, w) = \mathrm{ecc}(u)$. Зауважимо, що w не є суміжною з v, бо інакше це суперечило б $\mathrm{diam}(G) \geq 3$. З цього слідує $d_{\overline{G}}(u, v) = 2$. Тоді з того, що несусідні вершини в G будуть суміжними в $\overline{G}$, та кожна вершина в G має хоч одну сусідню, слідує доведення. □

Наступний результат був отриманий Френком Харарі та Робертом Норманом.

Теорема 3.8. *Центр зв'язного графа G знаходиться у блоці.*

Доведення. Припустимо, що існує граф G центр якого не знаходиться в одному блоці. Тоді існуватиме скороченна вершина $v \in V(G)$ видалення якої залишить як мінімум дві різні зв'язні компоненти G_1 та G_2, що містять вершини з центру. Зафіксуємо вершину $u \in V(G)$, що $d(v, u) = \mathrm{ecc}(v)$. Найкоротший шлях між v та u не може одночасно містити вершини з G_1 та G_2, бо тоді ці компоненти були б зв'язані одна з одною цим шляхом. Не втрачаючи загальності припустимо, що цей шлях не містить жодної вершини з G_2. Зафіксуємо вершину $w \in V(G_2), w \in \mathrm{Cen}(G)$. Звернемо увагу на факт, що u та w належить різним компонентам зв'язності з чого слідує, що найкоротший шлях з w до u проходить через v. Тоді справедливою буде нерівність $\mathrm{ecc}(w) > \mathrm{ecc}(v)$, що суперечить $w \in \mathrm{Cen}(G)$. □

Наслідок 3.9. *Центр зв'язного графа блоків породжує повний підграф.*

Доведення. Центр зв'язного графа блоків породжує повний підграф, бо з теореми 3.8 центр зв'язного графа знаходиться в блоці, а з означення графа блоків кожний його блок є повним. □

Наслідок 3.10. *Центр дерева складається або з одної або з двох суміжних вершин. При цьому, дерево T містить єдину центральну вершину тоді й тільки тоді, коли* $\mathrm{diam}(T) = 2\,\mathrm{rad}(T)$.

Доведення. Нехай T є деревом з $\mathrm{diam}(T) = 2\,\mathrm{rad}(T)$. Позначимо діаметральну пару вершин як u та u'. Тоді на шляху uu' буде знаходитися єдина вершина $v : d(v, u) = d(v, u') = \mathrm{rad}(T)$.

Нехай T є деревом з єдиною центральною вершиною v. Оскільки v не може бути листком, вона матиме дві сусідні вершини u та u'. Причому мають

існувати шляхи до різних ексцентричних вершин v, що проходять через u та u'. Бо інакше отримаємо суперечність із припущенням про єдиність центральної вершини, або взагалі, що v є центральною. Тоді відстань між ексцентричними вершинами v, що лежать по різні сторони v, буде дорівнювати подвійному радіусу, а з теореми 3.1 слідує, що $\mathrm{diam}(T) = 2\,\mathrm{rad}(T)$.

З теореми 3.8 слідує, що між двома вершинами з $\mathrm{Cen}(T)$ не може існувати вершини не з центру. Центр дерева, очевидно, не може складатися з трьох суміжних вершин, бо ексцентриситет "внутрішньої" вершини буде менше за ексцентриситет "зовнішніх". З цих же причин неможливе існування центру дерева з більшою кількістю сумжних вершин. Тож теорема доведена. □

Твердження 3.11. *Для кожного графа H існує граф G такий, що породжений підграф $G[\mathrm{Cen}(G)]$ ізоморфний H.*

Доведення. Покладемо $V(G) = V(H) \cup \{v_1, v_2, v_1', v_2'\}$, де вершини v_1 та v_2 суміжні з кожною вершиною з $V(H)$, але не одна з одною(див. Рис. 4).Вершина v_1' є суміжною тільки з v_1, а v_2' є суміжною тільки з v_2. Причому, якщо $u_1, u_2 \in V(H)$ є суміжними в H, то вони є суміжними в G. Таким чином у графі G для кожної $v \in V(H)$ справедливим буде твердження, що $\mathrm{ecc}(v) = \mathrm{rad}(G) = 2$. Також, для інших вершин виконується наступне: $\mathrm{ecc}(v_1) = \mathrm{ecc}(v_2) = 3$, $\mathrm{ecc}(v_1') = \mathrm{ecc}(v_2') = 4$. Таким чином для всіх вершин $v \in V(H)$ виконується $v \in \mathrm{Cen}(G)$, а породжений підграф $G[\mathrm{Cen}(G)]$ є ізоморфним графу H. □

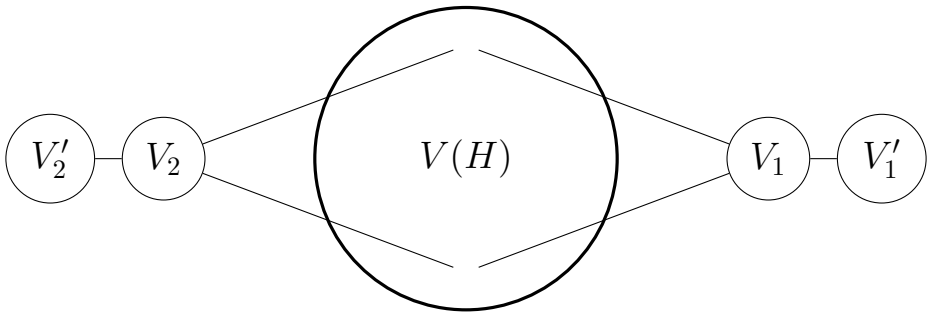


Рис. 4: Граф G, де породжений підграф $G[\mathrm{Cen}(G)]$ ізоморфний графу H

Зауваження 3.12. Зауважимо що, якщо $\mathrm{rad}(H) = 1$, то неможливо запропонувати такий граф G, щоб підграф утворений його центром був ізоморфний H . Якщо ж $\mathrm{rad}(H) \geq 2$ то граф H завжди можна реалізувати

з деякого графа G зі збереження радіусу H ($\mathrm{rad}(H) = \mathrm{rad}(G)$). Дійсно, візьмемо конструкцію наведену у теоремі 3.11, але замість додавання по одній вершині до v_1 і v_2, додамо по одному ланцюгу $P_{\mathrm{rad}(H)-1}$ до v_1 і v_2 з'єднуючи їх із одним з кінців відповідного ланцюга.

Теорема 3.13. *Для графа H існує граф G такий, що $G[\mathrm{Per}(G)]$ ізоморфний H тоді і тільки тоді, коли H є повним, або не має універсальної вершини.*

Доведення. Нехай для H існує граф G, що $G[\mathrm{Per}(G)]$ ізоморфний H. Припустимо, що H має унікальну вершину і не є повним. Виходить, що в такому разі існуватиме $v \in \mathrm{Per}(G)$, що $\forall u \in \mathrm{Per}(G)$, $u \neq v$, $d(u, v) = 1$. З цього випливає $\mathrm{diam}(G) = 1$, а оскільки за припущенням периферія G ізоморфна до H, отримуємо суперечність, бо це можливо тільки у випадку, коли H є повним графом.

Розглянемо спочатку тривіальний випадок коли граф H є повним. У такому випадку $\mathrm{Per}(H) = H$, адже всі вершини повного графа мають однаковий ексцентриситет. В іншому випадку означимо граф G, де $V(G) = V(H) \cup \{v\}$, а $E(G) = E(H) \cup \{vx : x \in V(H)\}$. У ньому $ecc(v) = 1$ а $ecc(v') = 2, \forall v' \in V(H)$. Таким чином всі $v' \in V(H)$ будуть належати периферії графа G і, оскільки ми не додавали жодних ребер між цими вершинами, справедливим буде твердження, що $G[\mathrm{Per}(G)]$ ізоморфний H. □

Зауваження 3.14. Зауважимо що зв'язний граф H можна реалізувати як периферію деякого зв'язного графа G зі збереженням діаметру H тоді і тільки тоді коли H самоцентральний. Дійсно, якщо H є самоцентральним, то тривіально $\mathrm{Per}(H) = H$. З іншого боку, якщо $H = G(\mathrm{Per}(G))$, тоді для всіх $u \in V(H)$ справедливо, що $\mathrm{ecc}_H(u) = \mathrm{diam}(G) = \mathrm{diam}(H)$ з чого слідує, що $u \in \mathrm{Per}(H)$, тобто $H = \mathrm{Per}(H)$, тож $V(H) = \mathrm{Per}(H)$ з чого слідує самоцентральність H.

3.1 Унікально ексцентрично точкові графи

Означення 3.15. *Унікально ексцентрично точковий граф* — це граф для кожної вершини якого існує єдина ексцентрична вершина. Далі в тексті буде зустрічатися як *у.е.т.* граф.

Лема 3.16. *Якщо x та y є сусідніми вершинами у.е.т. графа і мають різні ексцентриситети, то $\bar{x} = \bar{y}$.*

Доведення. Нехай граф G є *у.е.т* графом. Зафіксуємо вершини $x, y \in V(G)$, що є сусідніми. Без втрати загальності припустимо, що $\text{ecc}(x) < \text{ecc}(y)$. Розглянемо вершину $v \in V(G)$, що $v \neq x$ та $v \neq \bar{x}$. Тоді $d(x, v) \leq \text{ecc}(x) - 1$. З цього слідує, що наступна оцінка відстані буде справедливою $d(y, v) \leq d(x, y) + d(x, v) \leq 1 + \text{ecc}(x) - 1 = \text{ecc}(x)$. Тоді за нашим припущенням $d(y, \bar{x}) = \text{ecc}(y)$. □

Теорема 3.17. *Для у.е.т графа G наступні умови еквівалентні:*

1. *G — самоцентральний граф.*

2. *Кожна вершина в G є ексцентричною.*

3. *$Ecc(G)$ є неперетинним об'єднанням циклів довжини 2.*

4. *$\forall u \in V(G) : u = \bar{\bar{u}}$.*

Доведення. (2⇒1):Припустимо, що в *у.е.т* графі G, де кожна вершина є ексцентричною, існує така вершина $v \in V(G)$, що $v \notin \text{Cen}(G)$. Без втрати загальності в силу зв'язності для v існує сусідня вершина $u \in \text{Cen}(G)$, причому $\bar{u} \in \text{Cen}(G)$. Тоді з властивостей *у.е.т* графів слідує $\bar{\bar{u}} = u$. За лемою 3.16 отримуємо суперечність, бо з неї слідує $\bar{v} = \bar{u}$ і з припущення, що v не в центрі $\bar{\bar{u}} = v$.

(1⇒4):Нехай G є самоцентральним *у.е.т.* графом. Розглянемо вершину $v \in V(G)$. З самоцентральності випливає факт, що $ecc(v) = ecc(\bar{v})$, а з властивостей *у.е.т.* графів слідує $\bar{\bar{v}} = v$.

(4⇒3):За означенням орієнтованого графа ексцентриситетів та тим фактом, що G є *у.е.т.* графом справедливим буде твердженн, що існування циклів довших за 2 як і існування висячих вершин або ланцюгів суперечать умові, що $\forall u \in V(G) : u = \bar{\bar{u}}$. Оскільки кожна вершина хоч десь досягає свого ексцентриситета не існуватиме ізольованих вершин, тож імплікація доведена.

(3⇒2):Розглянемо $v \in V(G)$. Оскільки G є зв'язним *у.е.т.* графом, то існуватиме $\bar{v} \in V(G)$, а з третього твердження випливає $\bar{\bar{v}} = v$ з чого слідує імплікація.

□

Твердження 3.18. *Орграф ексцентриситетів нетривіального у.е.т. графа G має цикли тільки довжини 2.*

Доведення. Від супротивного. Припустимо, є цикл довжини 3. Тоді в G існує така трійка вершин $a, b, c : ecc(a) = b, ecc(b) = c, ecc(c) = a$. Оскільки G є у.е.т. графом, српаведливим має бути ланцюг нерівностей $d(a, b) > d(b, c) > d(c, a) > d(a, b)$. Отримали суперечність. □

Розглядаючи орграфи ексцентриситетів у.е.т. графів нам вдалося охарактеризувати їх будову в такому класі графів як граф блоків [5]. Для цього введемо означення $D_{m,k}$ орграфу.

Означення 3.19. Для пари натуральних чисел $m, k \in \mathbb{Z}_+$ означимо орграф $D_{m,k}$ наступним чином: $V(D_{m,k}) = \{u, v, x_1, ..., x_m, y_1, ..., y_k\}$, $A(D_{m,k}) = \{(u, v), (v, u)\} \cup \{(x_i, u), (y_j, v) : 1 \leq i \leq m, 1 \leq y \leq k\}$.

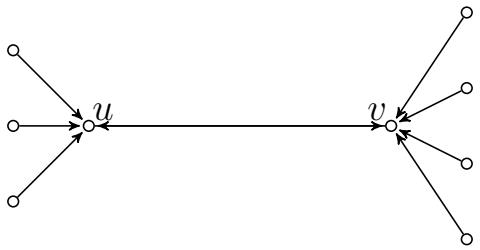


Рис. 5: $D_{3,4}$

Для опису у.е.т графів блоків та їх орграфів ексцентриситетів ми використаємо наступний метричний критерій графів блоків:

Теорема 3.20 (6). *Зв'язний граф G є графом блоків тоді і тільки тоді, коли для кожних чотирьох вершин $x, y, z, c \in V(G)$ найбільші дві з трьох сум відстаней*

$$d(x, y) + d(c, z), d(x, c) + d(y, z), d(x, z) + d(c, y)$$

завжди рівні.

Теорема 3.21. *Нехай G є у.е.т. графом блоків з $|V(G)| \geq 2$. Тоді $Ecc(G)$ ізоморфний до $D_{m,k}$ для $m = k = 0$ або $m = k = 1$, або $m, k \geq 2$. Навпаки, для кожного такого $D_{m,k}$ існує у.е.т. граф блоків (навіть дерево) такий, що його $Ecc(G)$ ізоморфний до $D_{m,k}$.яки $m, k \geq 0$. (Розглянути погані випадки)*

Доведення. Нехай G є у.е.т. графом блоків. Спочатку покажемо, що G містить лише дві периферійні вершини. Від супротивного. Дійсно, нехай G містить три периферійні вершини a, b, c. Не втрачаючи загальності, покладемо $d(a,b) = \mathrm{diam}(G)$. Якщо хоча б одна з a, b є ексцентричною точкою для c, то G не є у.е.т. Отже $\overline{c} \notin \{a, b\}$. Застосуємо теорему 3.20 для четвірки $a, b, c, \overline{c}$. Тоді однією з сум відстаней буде $d(a,b) + d(c,\overline{c}) = 2\,\mathrm{diam}(G)$. З цього слідує, що одна із сум $d(a,c) + d(b,\overline{c})$ або $d(a,\overline{c}) + d(b,c)$ також дорівнює подвоєному діаметру, але це можливо тільки у випадку, коли a або b є ексцентричною для c вершиною. Отримали суперечність.

Тепер покажемо, що кожна ексцентрична вершина в G є периферійною. Від супротивного. Нехай $\overline{v}$ для неперифериної вершини $v \in V(G)$ не є периферійною. Зафіксуємо єдину діаметральну пару a, b та розглянемо четвірку вершин $a, b, v, \overline{v}$. Слідуючи теоремі 3.20, розглянемо три суми відстаней цих точок. Тоді сума $d(a,b) + d(v,\overline{v}) = diam(G) + \mathrm{ecc}(v)$ точно має бути найбільшою з трійки, адже вона є сумою відстаней двох ексцентричних пар. З цього слідує, що $\overline{v} = a$ або $\overline{v} = b$, бо інакше інші дві суми не зрівняються з розглянутою вище. Отримуємо суперечність.

Таким чином, $Ecc(G)$ ізоморфний $D_{m,k}$.

Оскільки кожне дерево є графом блоків, то нам достатньо запропонувати спосіб побудови дерева для $D_{m,k}$ згаданих в теоремі. Для випадків, коли $m = k = 0$, або $m = k = 1$ підійдуть P_2 і P_4 відповідно. Коли $m, k \geq 2$ у випадку $m = k$ нам підійде P_{2m+2}. Інакше нам підійде P_6, де центральні вершини будуть додатково інцидентні $k-2$ та $m-2$ вершинам (див. рис. 6). □

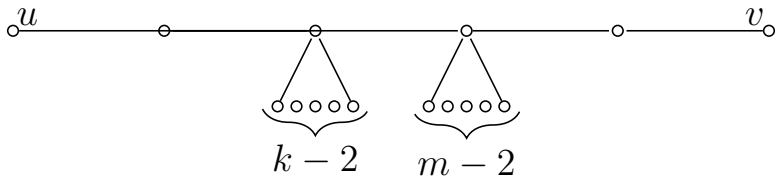


Рис. 6: Дерево G, що його $Ecc(G)$ ізоморфний до $D_{m,k}$

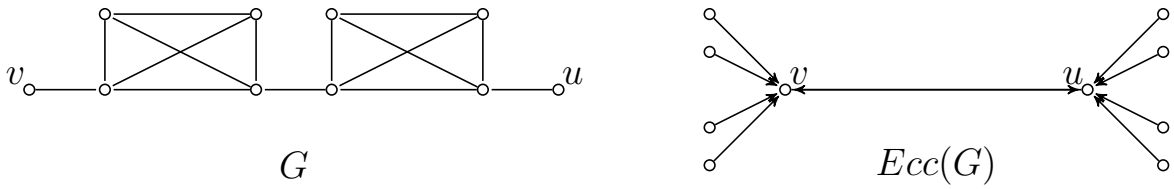


Рис. 7: Граф G та $Ecc(G)$

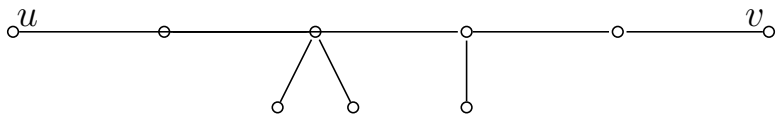


Рис. 8: Дерево G, що ізоморфне до $D_{3,4}$(див. рис. 5)

Наступний результат містить опис всіх у.е.т. графів, які є деревами.

Твердження 3.22. *Дерево є у.е.т. графом тоді й тільки тоді, коли воно містить рівно дві центральні та рівно дві периферійні вершини.*

Доведення. Нехай дерево G є у.е.т. графом. З наслідку 3.10 слідує, що центр дерева складається або з двох суміжних вершин, або з однієї. Причому центром дерева буде єдина вершина у випадку коли $\operatorname{diam}(G) = 2\operatorname{rad}(G)$, що неможливо за твердженням 3.22. Очевидно, що не може бути єдиної периферійної вершини. Якщо існуватиме більше ніж дві периферійні вершини, то щонайменше одна з центральних вершин матиме більше однієї ексцентричної точки, що суперечить тому. що G є у.е.т. графом.

Нехай дерево G має дві суміжні центральні вершини v та u і дві периферійні вершини v' та u'. Покладемо $\operatorname{rad}(G) = r$. Без втрати загальності припустимо, що $d(v, v') = \operatorname{ecc}(v) = r$ та $d(u, u') = \operatorname{ecc}(u) = r$. З суміжності v та u слідує $d(v, u') = d(u, v') = r - 1$. Наступний результат будується на факті, що вершини дерева набувають ексцентриситет на периферійних вершинах. Оскільки v та u суміжні, то кожна вершина $a \in V(G)$ знаходиться ближче або до v, або до u і відповідно найдалі від a знаходиться або тільки u', або тільки v'. Тож всі вершини матимуть тільки одну ексцентричну ним точку, з чого слідує, що G є у.е.т. графом. □

Виявляється, що можливо розширити цей же критерій на зв'язні графи блоків.

Теорема 3.23. *Зв'язний граф блоків є у.е.т. графом тоді й тільки тоді, коли він має точно дві центральні і точно дві периферійні вершини.*

Доведення. Нехай G є у.е.т графом блоків. Доведення того, що такий граф має точно дві периферійні вершини, є в доведенні теореми 3.21. Тож нам залишилося довести, що існує точно дві центральні вершини. Розглянемо шлях між двома діаметральними вершинами графа G. Їх має з'єднувати тільки один шлях з точністю до блоків, бо, оскільки G є графом блоків, інакше вони належатимуть одному блоку. З цього слідує, що центральні вершини знаходяться на цьому шляху. З теореми 3.8 випливає, що всі центральні вершини будуть знаходитися в блоці. Якщо існуватиме тільки одна центральна вершина, то вона набуватиме ексцентриситету одразу на двох периферійних вершинах. Якщо в такому "центральному" блоці будуть інші вершини крім скороченних, тоді вони набуватимуть ексцентриситет одразу на двох периферійних вершинах. З цього слідує, що можуть існувати тільки дві центральні вершини.

Нехай G є графом блоків з точно двома центральними і точно двома периферійними вершинами. Оскільки діаметральна пара вершин p_1 та p_2 не лежить в одному блоці, то між ними існує тільки один шлях з точністю до блоків. В такому випадку блок з центральними вершинами c_1 та c_2 знаходиться на цьому шляху. Тоді кожна інша вершина a знаходиться або ближче до c_1, або ближче до c_2 і відповідно найдалі від a буде або тільки p_2, або тільки p_1. Тож граф G буде у.е.т. графом. □

Наслідок 3.24. *Кожен антиподальний граф є самоцентральним у.е.т. графом.*

Зауваження 3.25. Зауважимо, що властивість "бути *у.е.т.* графом" та властивості 1 та 2 теореми 3.17 попарно непорівнювані. Надалі позначаючи факт того, що G є *у.е.т.* графом як 0, наведемо контрприклади до імплікацій між цими властивостями.

$(1 \not\Rightarrow 0)$: Контрприкладом є C_{2n+1}.

$(0 \not\Rightarrow 1)$: Контрприкладом є P_4.

$(1 \Rightarrow 2)$: Хоч дана імплікація виконується, але обернене твердження є невірне.

$(2 \not\Rightarrow 1)$: Контрприкладом є $K_{3,3-e}$(див. Рис. 10).

$(0 \not\Rightarrow 2)$: Контрприкладом є P_4.

$(0 \not\Rightarrow 1)$: Контрприкладом є повний двочастковий граф $K_{2,3}$(див. Рис. 9).

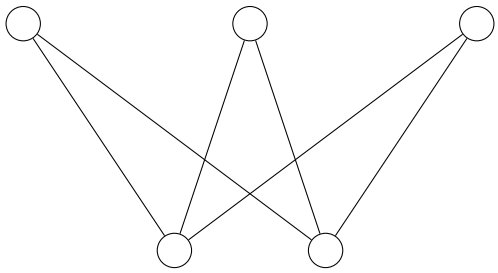


Рис. 9: Повний двочастковий граф $K_{2,3}$

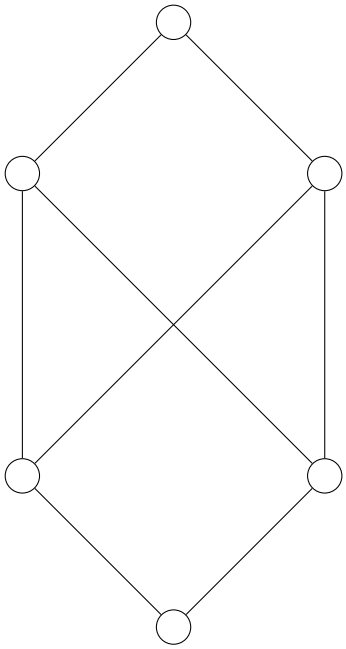


Рис. 10: $K_{3,3-e}$

Твердження 3.26. *Для кожного у.е.т. графа* G *виконується* $\operatorname{diam}(G) < 2\operatorname{rad}(G)$

Доведення. Доведення слідує з факту, що всі периферійні точки знаходяться на відстані $\operatorname{rad}(G)$ від центральної точки, коли $\operatorname{diam}(G) = 2\operatorname{rad}(G)$. □

4 Відносні центри

Означення 4.1. *Центрально критиний граф* - це граф для кожної нетривіальної підмножини $S \subset V(G)$ якого виконується $\mathrm{Cen}_S(G) \neq \mathrm{Cen}(G)$. Інакше можна сказати, що для всіх центральних множин A нетривіальних підмножин $S \subset V(G)$ виконується $A \neq \mathrm{Cen}(G)$.

Теорема 4.2. *Граф G є центрально критичним тоді й тільки тоді, коли він є самоцентральним у.е.т. графом.*

Доведення. Спочатку доведемо, що кожна $v \in V(G)$ має єдину $v' : \bar{v} = v'$. Припустимо протилежне. Нехай існує v, що не є ексцентричною вершиною для жодної іншої вершини. Покладемо $S = V(G) \setminus \{v\}$. Тоді з припущення випливає, що $\forall a \in V(G) : \mathrm{ecc}_S(a) = \mathrm{ecc}(a)$. Оскільки жоден з ексцентриситетів вершин не змінився, то $\mathrm{Cen}_S(G) = \mathrm{Cen}(G)$, що суперечить центрально критичності графа G. Далі припустимо, що $v \in V(G)$ це така вершина, що у випадку коли вона є ексцентричною для іншої вершини u, ця u матиме ще іншу ексцентричну точку $v' : v \neq v'$. Знову покладемо $S = V(G) \setminus \{v\}$. Тоді знову $\forall a \in V(G) : \mathrm{ecc}_S(a) = \mathrm{ecc}(a)$, що суперечить критично центральності G. Отже, ми довели, що для кожної вершини $a \in V(G)$ існує інша $b \in V(G)$, що a є єдиною ексцентричною вершиною для b, а з теореми 3.17 слідує, що G є самоцентральним.

Припустимо, що G є самоцентральним у.е.т. графом і покладемо $\mathrm{rad}(G) = r$. З теореми 3.17 слідує, що кожна вершина є ексцентричною. Тоді для кожної вершини $a \in V(G)$ існує $b \in V(G) : \bar{b} = a$. З цього слідує, що для випадку, коли $a \in V \setminus S$ виконується $\mathrm{ecc}_S(b) < r = \mathrm{ecc}(b)$. В іншому випадку, коли $a \in S$ справджується $\mathrm{ecc}_S(b) = r = \mathrm{ecc}(b)$. Оскільки для кожної нетривіальної множини $S \subset V(G)$ завжди існуватимуть такі вершини $a \in V(G) \setminus S$, то завжди виконуватиметься $\mathrm{Cen}_S(G) \neq \mathrm{Cen}(G)$, тобто G є критично центральним графом. □

Наступний результат є узагальненням теореми 3.8 для $S-$центрів.

Теорема 4.3. *Кожний $S-$центр зв'язного графа G знаходиться у блоці.*

Доведення. Для $S \subseteq V$ припустимо, що $\mathrm{Cen}_S(G)$ належить більше ніж одному блоку. Тоді в G існуватиме така точка v, що $G \setminus v$ містить щонайменше дві компоненти G_1 та G_2, кожна з яких має вершини з $\mathrm{Cen}_S(G)$. Зафіксуємо $u \in S$ та нехай для неї виконується $d(u, v) = \mathrm{ecc}_S(v)$. Справедливим буде твердження, що найкоротший $u - v$ шлях не перетинає G_1 або

G_2, припустимо, що він не перетинає саме G_1. Тоді зафіксуємо вершину з $w \in G_1$, що $w \in \mathrm{Cen}_S(G)$. Тоді v має належати найкоротшому $w-u$ шляху з чого випливає нерівність $\mathrm{ecc}_S(w) \geq d(u,w) = d(w,v)+d(v,u) \geq \mathrm{ecc}_S(v)+1$, що суперечить факту, що $w \in \mathrm{Cen}_S(G)$. Тож всі $S-$центри знаходяться в блоці. □

Наступна теорема описує вигляд центральних множин для графів блоків.

Теорема 4.4. *Нехай граф G з множиною вершин V і блоками $B_1, ..., B_r$. Для $1 \leq i \leq r$, покладемо $V(B_i) = V_i$. Центральними вершинами G є одноточкові множини $\{v\}$, $v \in V(G)$ та V_i.*

Доведення. Розглянемо спочатку тривіальний випадок коли $S = \{v\}$. Тоді очевидно, що $\mathrm{Cen}_S(G) = \{v\}$. Тож всі $\{v\}|v \in V$ є центральними множинами.

Нехай S є нетривіальною підмножиною V_i для $1 \leq i \leq r$, що містить дві або більше вершин. З означення графу блоків випливає, що $\mathrm{ecc}_S(v) = 1$ для всіх $v \in V_i$ і $\mathrm{ecc}_S(x) > 1$ для всіх $x \in V \setminus V_i$. З цього слідує рівність $\mathrm{Cen}_S(G) = V_i$. Тож всі $V_i, 1 \leq i \leq r$ є центральними вершинами.

Розглянемо випадок коли $S \subseteq V(G)$ містить щонайменше дві вершини з різних блоків. Нехай x є скороченною вершиною графу G, що $\mathrm{ecc}_S(x) = k$. Також припускаємо, що $d(x,v) = k$ де $v \in S$. Зафіксуємо найкоротший шлях між x та v $P = x = x_0x_1...x_ix_{i+1}...x_k = v$. Припустимо, що $\mathrm{ecc}_S(x_1) = k-1$. Оскільки ексцентриситет вершин з шляху P ніколи не дорівнюватиме 0, ми можемо дві вершини x_i та x_{i+1}, що $\mathrm{ecc}_S(x_i) = \mathrm{ecc}_S(x_{i+1}) = k-i$. Звертаємо вашу увагу, що можливий випадок $x_i = x_{i+1}$. Тоді для кожної вершини y з блоку, що містить x_i та x_{i+1}, справджуватиметься $\mathrm{ecc}_S(y) = k-i$, а ексцентриситети вершин з інших блоків будуть збільшуватися. Таким чином $S-$центром у даному випадку буде блок, що містить x_i та x_{i+1}.

Розглянемо випадок коли $\mathrm{ecc}_S(x_i) = k-i$, а $\mathrm{ecc}_S(x_{i+1}) = k-i+1$. Всі вершини y з блоку, що містить x_i та x_{i+1}, такі що $y \neq x_i$, матимуть $\mathrm{ecc}_S(y) = k-i+1$. Всі вершини з інших блоків матимуть більший ексцентриситет. З цього випливає, що $S-$центром графу G у цьому випадку буде $\{x_i\}$. Цей результат закінчує доведення твердження, що центральними вершинами графу блоків є або одноточкові множини, або блоки.

□

5 Знаходження центральної та периферійної області простого многокутника

Текст пов'язаний з алгоритмом триангуляції багатокутника є рефератним матеріалом з [1].

5.1 Постановка задачі

Алгоритм отримує на вхід масив кортелів з двох дійсних чисел, що є репрезентацією точок багатокутника, що задані проти годинникової стрілки.

На вихід він має повернути матрицю відстаней дуального графа триангульованого багатокутника.

5.2 Структура даних

Головною структурою даних, що зберігатиме прогрес роботи алгоритмів буде двозв'язний список ребер. За допомогою цієї структури даних можна легко відновити зовнішній вигляд багатокутника та знаходити множини суміжних та інцидентних елементів. Дана сруктура даних складається з трьох множин: множини вершин, ребер та граней. Кожна вершина двозв'язного списку окрім збереження власних координат, також відсилається на одне напівребро, що в ній починається. Кожна грань двозв'язного списку посилається на одне напівребро, що належить кривій, яка обмежує зовнішність грані, та на одне напівребро, що належить кривій, яка обмежує внутрішність грані. Кожне напівребро посилається на свого близнюка, свою початкову точку, інцидентну грань, наступне та попереднє напівребра.

5.3 Опис алгоритму розбиття многокутника на монотонні частини

Задача триангуляції простого багатокутника не є тривіальною, проте триангуляція монотонного багатокутника є більш легкою завдяки наступній характеризації y-монотонного трикутника: проходячи від найвищої до найнижчої вершини y-монотонного багатокутника лівим ланцюгом вершин (правим ланцюгом вершин), напрямок руху завжди буде спадним.

На вхід алгоритм має отримати двозв'язний список ребер простого багатокутника. Результатом роботи алгоритму буде двозв'язний список ребер

з додатковими внутрішніми напівребрами, що розділяють простий многокутник на y-монотонні частини. Робота алгоритму заключається в проходженні від найвищої до найнижчої вершини і певній реакції на кожну вершину. Варто зауважити, що у випадку, коли вершини мають однакову y-координату, вищою ми вважатимемо лівішу вершину.

Реакція алгоритму на точку залежатиме від її типу. Тип точки визначає її взаємне розташування з сусідніми вершинами. Таким чином, точку, що лежить нижче обох сусідніх вершин та має внутрішній кут менше π, ми називатимемо *кінцевою*. У випадку, коли кут є більшим за π, така вершина буде *вершиною злиття*. Таку, що лежить вище обох сусідніх вершин та має внутрішній кут менше π, ми називатимемо *початковою*. У випадку, коли кут є більшим за π, така вершина буде *роздільною*. *Регулярною* вершиною називатимемо точку, що лежить нижче однієї з сусідніх вершин, але вище іншої.

Для пояснення подальшої роботи алгоритму знадобиться поняття *допоміжної вершни ребра*. Для ребра e в певний момент роботи алгоритму допоміжною вершиною буде найнижча "опрацьована" алгоритмом точка p, що горизонтальний відрізок, що з'єднує e та p, повністю лежить в багатокутнику.

Багатокутник буде монотонном у випадку, коли він не матиме роздільних вершин або вершин злиття. Щоб позбутися роздільної вершини необхідно з'єднати її з допоміжною вершиною найближчого ребра ліворуч від неї. З означення допоміжної вершини також зрозуміло, що кожна вершина в певний момент виконання алгоритму є допоміжною для якогось ребра. Щоб позбутися вершини злиття p необхідно її з'єднати з наступною допоміжною вершиною ребра, для якого p є допоміжною вершиною.

Наведемо детальний опис дій алгоритму з прикладами реалізації мовою програмування Python:

- **point** – точка, що розглядається
- **dceList** – двозв'язний список ребер
- **tree** – бінарне дерево, де зберігаються ребра зліва направо.
- **supports** – словник, де для ребер зберігаються відповідні їм допоміжні вершини

Опрацьовуючи роздільну v вершину, алгоритм знаходить найближче ребро e ліворуч від вершини, з'єднує ребром v та допоміжну вершину e,

робить v новою допоміжною вершиною e, також знаходить ребро, що починається в v, додає його до tree і робить v його допоміжною вершиною:

```
def handleSplitPoint(point, dceList, tree, supports):
  edge = getLeftEdge(point, tree)
  dceList.addEdge(supports[edge], point)
  supports[edge] = point
  edge_i = dceList.findEdgeByOrigin(point)
  tree.insert(edge_i, edge_i)
  supports[edge_i]=point
```

Опрацьовуючи початкову точку v, алгоритм знаходить ребро e ліворуч від v, додає e до tree та робить його допоміжною вершиною v:

```
def handleStartPoint(point, dceList, tree, supports):
  edge = dceList.findEdgeByOrigin(point)
  tree.insert(edge,edge)
  supports[edge]=point
```

Опрацьовуючи точку злиття v, алгоритм знаходить ребро e, що закінчується у v, у випадку, коли допоміжна вершина e є точкою злиття, з'єднує її ребром з v, з tree видаляється e, алгоритм знаходить ребро r ліворуч від v, у випадку, коли допоміжна вершина r є точкою злиття, з'єднує її ребром з v, робить v новою допоміжною вершиною r:

```
def handleMergePoint(point, dceList, tree, supports):
  edge = dceList.findEdgeByEnd(point)
  supVert = supports[edge]
  if supVert.typeOfPoint() == 'm':
    dceList.addEdge(supVert, point)
  tree.remove(edge)
  edge_l = getLeftEdge(point, tree)
  supVert= supports[edge_l]
  if supVert.typeOfPoint() == 'm':
    dceList.addEdge(supVert, point)
  supports[edge_l]=point
```

Опрацьовуючи регулярну вершину v, алгоритм у випадку, коли праворуч від v знаходиться внутрішність багатокутника, якщо допоміжна вершина ребра r, що закінчується у v, є точкою злиття, додає ребро між нею та v, видаляє r з tree, знаходить ребро, що починається у v, додає його до tree і робить v його допоміжною вершиною.

А у випадку, коли внутрішність багатокутника знаходиться ліворуч від v, алгоритм знаходить ребро e ліворуч від v, якщо допоміжна вершина e є точкою злиття, додає між нею та v ребро, робить v новою допоміжною вершиною e:

```
def handleRegularPoint(point, dceList, tree, supports):
  if innerRight(point, dceList) :
    prevEdge = dceList.findEdgeByEnd(point)
    prevSupp = supports[prevEdge]
    if prevSupp.typeOfPoint() == 'm':
      dceList.addEdge(prevSupp, point)
    tree.remove(prevEdge)
    edge = dceList.findEdgeByOrigin(point)
    tree.insert(edge, edge)
    supports[edge]=point
  else:
    edge = getLeftEdge(point, tree)
    edgeSupp = supports[edge]
    if edgeSupp.typeOfPoint() == 'm':
      dceList.addEdge(edgeSupp, point)
    supports[edge]=point
```

5.4 Опис алгориму триангуляції монотонного трикутника

Наступний крок полягає в триангуляції кожної монотонної грані багатокутника. Алгоритм буде опрацювувати вершини грані від найвищої до найнижчої, а у випадку коли вони матимуть однакову y-координату, вищою вважатиметься лівіша вершина. Як допоміжна структура даних буде використовуватися стек. На початку алгоритму стек буде пустий і в подальшому міститиме точки, що потребуватимуть більше ребер. Опрацьовуючи кожну точку, ми зв'єднуватимемо її з якнайбільшою кількістю вершин зі стеку.

Шлях від найвищої до найнижчої точки багатокутника, що лежить ліворуч від найвищої вершини ми називатимемо лівим ланцюгом, а шлях праворуч — правим ланцюгом. Алгоритм опрацьовує точку v відносно того чи знаходиться вона на тому ж ланцюгу що і минула опрацювана точка. Якщо так, тоді ми дістаємо зі стеку найверхню вершину, дістаємо зі стеку вершини поки їх можна з'єднувати ребрами з v та повертаємо до стеку v з останньою вершиною яку дістали:

```
def handlePointsFromSameChain(currentVert, prevVert,
                              prevPrevVert, dceList, stack, rightC):
  check = checkRightTurn if (currentVert in rightC) else checkLeftTurn
  prev = stack.pop()
  prev2 = prevPrevVert
  while check(prev2.coords(), prev.coords(), currentVert.coords()):
    prev2 = stack.pop()
    dceList.addEdge(prev2, currentVert)
    prev = prev2
    if len(stack)==0 :
      break
    prev2 = stack[-1]
  stack.extend([prev, currentVert])
  return
```

Інакше, якщо точка v, що розглядається, з іншого ланцюга аніж попередня вершина, ми дістаємо, всі вершини зі стеку, проводимо між ними та v ребра, та кладемо в стек v з попередньою точкою:

```
def handlePointsFromDifferentChains(currentVert, prevVert, dceList, stack):
  while not len(stack)==0 :
    vert = stack.pop()
    if len(stack)!=0:
      dceList.addEdge(vert, currentVert)
  stack.extend([prevVert, currentVert])
  return
```

5.5 Алгоритм знаходження центру та периферії дуального графа триангульованого многокутника

Маючи триангульований багатокутник, ми будуємо дуальний граф цього багатокутника наступним чином: трикутні грані будуть вершинами, а ребро між вершинами буде у випадку, коли відповідні грані є суміжними.

Твердження 5.1. *Дуальний граф простого багатокутника є деревом.*

Наслідок 5.2. *Центром триангульованого простого багатокутника будуть одна або дві грані.*

Алгоритм знаходження центру та периферії графа має доволі тривіальне рішення, якщо знайти матрицю відстаней графа D. Найбільше число

в рядку матриці відстаней відповідатиме ексцентриситету відповідної вершини, після цього найменший та найбільший ексцентриситети будуть відповідно центром та периферією графа. Завдяки двозв'язному списку ребер ми можемо знайти матрицю суміжності A для вершин графа. Алгоритм побудови матриці відстаней базується на наступному твердженні:

Твердження 5.3. *Якщо граф G зв'язний, то відстань між v_i та v_j дорівнює найменшому n, що $a_{i,j}^{(n)} > 0$, де $v_i, v_j \in V(G)$ та i, j є додатніми цілими числами, що відповідають за рядок та стовпчик в матриці суміжності, а $a_{i,j}^{(n)}$ відповідає числу в i-тому рядку на j-тому стовпчику матриці суміжності піднесеної до n-тої степені.*

Наводимо реалізацію даного алгоритму знаходження матриці відстаней за матрицею суміжності:

```
def makeDistanceMatrix(A):
  n=len(A)
  D=A.copy()

  for i in range(0, n):
    B=A.copy()
    k=1
    while thereAreZeroes(i,D) and k < n-1:
      k+=1
      B=A*B
      for j in range(0,n):
        if D[i,j]==0 and B[i,j]!=0 and i!=j:
          D[i,j]=k
          D[j,i]=k
  return D
```

Далі наведемо приклад реалізації алгоритмів знаходження центру та периферії графа за матрицею відстаней D:

```
def center(D):
  eccs_list = eccs(D)
  rad = min(eccs_list)
  radius = []
  for i in range(0, len(eccs_list)):
    if eccs_list[i]==rad:
      radius.append(i)
  return radius
```

```python
def periphery(D):
  eccs_list = eccs(D)
  maxecc = max(eccs_list)
  periphery = []
  for i in range(0, len(eccs_list)):
    if eccs_list[i]==maxecc:
      periphery.append(i)
  return periphery
```

6 Програмна реалізація

Для реалізації алгоритму була застосована мова програмування Python та біліотеки: Numpy, functools, matplotlib, bintrees.

6.1 Опис класів

Клас **Point**
Репрезентація точки на площині.

Поля:

- **x** – координата точки на вісі абсцис.
- **y** – координата точки на вісі ординат.
- **edge** – ребро, що інцидентне до даної точки.
- **pointType** – тип точки в контексті багатокутника.

Статичні методи:

calcTurn(i,j,k)
Обраховує напрям повороту кривої, що задана трьома точками. Вважаємо, що перший аргумент відповідає точці з якої починається крива.

Аргументи:

- **i** – перша вершина кривої, що задана кортелем з двох координат.
- **j** – друга вершина кривої, що задана кортелем з двох координат.
- **k** – третя вершина кривої, що задана кортелем з двох координат.

Повертає:
Значення в межах $[-1, 1]$. Якщо значення більше 0, то вважаємо, що крива повертає ліворуч, інакше – праворуч.

determineVert(verts, i)
Визначає тип точки в контексті багатокутника.

Аргументи:

- **verts** – вершини багатокутника, що задані проти годинникової стрілки.
- **i** – індекс точки в verts.

Повертає:
Стрічку “e” якщо точка є кінцевою. Стрічку “m” якщо точка є точкою злиття. Стрічку “s” якщо точка є роздільною. Стрічку “st” якщо точка є початковою. Стрічку “r” якщо точка є регулярною.

Клас **Edge**
Репрезентація напівребра багатокутника.

Поля:

- **origin** – точка з якої починається напівребро.
- **twin** – ребро-двійник.

- **face** – суміжна грань багатокутника.
- **next** – наступне ребро в багатокутнику.
- **prev** – попереднє ребро.

Клас **Face**
Репрезентація грані багатокутника.

Поля:

- **name** – назва грані.
- **outComponent** – напівребро, що є частиною зовнішньої межі грані.
- **innComponent** – напівребро, що є частиною внутрішньої межі грані.

Відкриті методи:

getVertsByFace()
Знаходить всі вершини, що інцидентні даній грані

Повертає:
Масив точок, що інцидентні даній грані.

centroidOfFace()
Знаходить центроїду трикутної грані

Повертає:
None якщо грань не трикутна, або точку, що є центроїдою грані.

Клас **DceList**
Репрезентація двозв'язного списку ребер.

Поля:

- **points** – точки багатокутника.
- **edges** – ребра багатокутника.

- **faces** – грані багатокутника.

Відкриті методи:

findEdgeByOrigin(origin)
Знаходить напівребро, що починається в даній точці.

Аргументи:

- **origin** – точка.

Повертає:
Ребро, що починається в origin, або None якщо такого не існує.

findOnFaceByOriginAndEnd(origin, end)
Знаходить два ребра, що належать одній грані.

Аргументи:

- **origin** – точка.
- **end** – точка.

Повертає:
Кортель з двох ребер, що належать одній грані. Перше має починатися в точці origin, друге ребро має закінчуватися в точці end.

findEdgeByEnd(end)
Знаходить два ребра, що належать одній грані.

Аргументи:

- **end** – точка.

Повертає:
Два ребра, що належать одній грані, але одне з них має починатися в точці origin, а інше закінчуватися в точці end.

getVertByCoord(tup)
Знаходить точку за координатами.

Аргументи:

- **tup** – кортель з двох чисельних значень.

Повертає:
Точку, що належить двозв'язному списку ребер і має координати tup, або None якщо такої немає.

getFaceByName(name)
Знаходить грань за ім'ям.

Аргументи:

- **name** – стрічка, або цифра.

Повертає:
Грань, що належить двозв'язному списку ребер і має ім'я name, або None якщо такої немає.

getIncFaces(face)
Знаходить суміжні грані до даної на вхід.

Аргументи:

- **face** – грань.

Повертає:
Масив суміжних граней з двоз'язного списку реберр до грані face.

getIncMatrix()
Знаходить матрицю суміжності граней.

Повертає:
Повертає numpy.matrix, що є матрицею суміжності граней у двозв'язному списку.

addEdge(origin, end)
Додає в двозв'язний список ребро, що починається в origin та закінчується в end.

Аргументи:

- **origin** – точка.
- **end** – точка.

6.2 Опис методів

getLeftEdge(point, tree)
Знаходить найближче ребро ліворуч від заданої точки.

Аргументи:

- **point** – точка.
- **tree** – бінарне дерево, де у вузлах зберігаються ребра.

Повертає:
Ребро, що є найближчим ребром ліворуч від point.

handlePoint(point, dceList, tree, supports)
Викликає потрібну функцію обробки точки відносно її типу.

Аргументи:

- **point** – точка.
- **dceList** – двозв'язний список ребер.
- **tree** – бінарне дерево, де у вузлах зберігаються ребра.
- **supports** – словник, де для точок зберігаються допоміжні ребра.

handleStartPoint(point, dceList, tree, supports)
Обробляє початкову точку.

Аргументи:

- **point** – точка.
- **dceList** – двозв'язний список ребер.
- **tree** – бінарне дерево, де у вузлах зберігаються ребра.
- **supports** – словник, де для точок зберігаються допоміжні ребра.

handleEndPoint(point, dceList, tree, supports)
Обробляє кінцеву точку.

Аргументи:

- **point** – точка.
- **dceList** – двозв'язний список ребер.
- **tree** – бінарне дерево, де у вузлах зберігаються ребра.
- **supports** – словник, де для точок зберігаються допоміжні ребра.

handleSplitPoint(point, dceList, tree, supports)
Обробляє роздільну точку.

Аргументи:

- **point** – точка.
- **dceList** – двозв'язний список ребер.
- **tree** – бінарне дерево, де у вузлах зберігаються ребра.
- **supports** – словник, де для точок зберігаються допоміжні ребра.

handleMergePoint(point, dceList, tree, supports)
Обробляє точку злиття.

Аргументи:

- **point** – точка.
- **dceList** – двозв'язний список ребер.
- **tree** – бінарне дерево, де у вузлах зберігаються ребра.
- **supports** – словник, де для точок зберігаються допоміжні ребра.

handleRegularPoint(point, dceList, tree, supports)
Обробляє регулярну точку.

Аргументи:

- **point** – точка.
- **dceList** – двозв'язний список ребер.
- **tree** – бінарне дерево, де у вузлах зберігаються ребра.
- **supports** – словник, де для точок зберігаються допоміжні ребра.

innerRight(point, dceList)
Перевіряє чи внутрішність багатокутника знаходиться праворуч від даної регулярної точки.

Аргументи:

- **point** – точка.
- **dceList** – двозв'язний список ребер.

Повертає:
Значення True or False.

highestInd(verts)
Знаходить індекс найвищої точки в масиві verts.

Аргументи:

- **verts** – масив кортелей з двох циферних значень.

Повертає:
Ціле число.

lowestInd(verts)
Знаходить індекс найнижчої точки в масиві verts.

Аргументи:

- **verts** – масив кортелей з двох циферних значень.

Повертає:
Ціле число.

chains(verts, dceList)
Знаходить дві послідовності точок від найвищої до найнижчої точки в масиві verts. Перша послідовність йде праворуч від найвищої точки, інше – ліворуч.

Аргументи:

- **verts** – масив кортелей з двох циферних значень.
- **dceList** – двозв'язний список ребер.

Повертає:
Кортель з двох масивів чисельних значень.

getTopLowPoints(verts, dceList)
Знаходить точки з двозв'язного списку ребер, що відповідають найвищій та найнижчій точці з масиву verts.

Аргументи:

- **verts** – масив кортелей з двох циферних значень.
- **dceList** – двозв'язний список ребер.

Повертає:
Кортель з точок.

handlePointsFromDifferentChains(currentVert, prevVert, dceList, stack)
У двозв'язному списку dceList з'єднує ребрами currentVert та всі вершини з stack крім останньої.

Аргументи:

- **currentVert** – точка.
- **prevVert** – точка.
- **dceList** – двозв'язний список ребер.
- **stack** – масив точок.

handlePointsFromSameChain(currentVert, prevVert, prevPrevVert, dceList, stack, rightC)
У двозв'язному списку ребер dceList з'єднує currentVert зі всіма точками зі stack у випадку, якщо між ними і currentVert не існує ребра і нове ребро повністю належатиме многокутнику.

Аргументи:

- **currentVert** – точка.
- **prevVert** – точка.
- **prevPrevVert** – точка.
- **dceList** – двозв'язний список ребер.
- **stack** – масив точок.
- **rightC** – масив точок.

handleLastPoint(lowestVert, dceList, stack)
У двозв'язному списку ребер dceList з'єднує lowestVert зі всіма точками зі stack.

Аргументи:

- **lowestVert** – точка.
- **dceList** – двозв'язний список ребер.
- **stack** – масив точок.

thereAreZeroes(row, matrix)
Перевіряє чи присутні нулі в рядку row матриці matrix.

Аргументи:

- **row** – індекс рядка в матриці matrix.
- **matrix** – матриця цілих чисел.

Повертає:
Значення True або False.

makeDistanceMatrix(A)
За матрицею суміжності A робить матрицю відстаней.

Аргументи:

- **A** – матриця.

Повертає:
Матрицю.

center(D)
За матрицею цілих чисел D знаходить рядки в яких найбільше число є найменшим серед найбільших чисел в інших рядках.

Аргументи:

- **D** – матриця.

Повертає:
Масив цілих чисел, що є індексами рядків матриці D.

periphery(D)
За матрицею цілих чисел D знаходить рядки в яких найбільше число є найбільшим серед найбільших чисел в інших рядках.

Аргументи:

- **D** – матриця.

Повертає:
Масив цілих чисел, що є індексами рядків матриці D.

eccs(D)
За матрицею відстаней D знаходить найбільшу відстань в кожному рядку.

Аргументи:

- **D** – матриця.

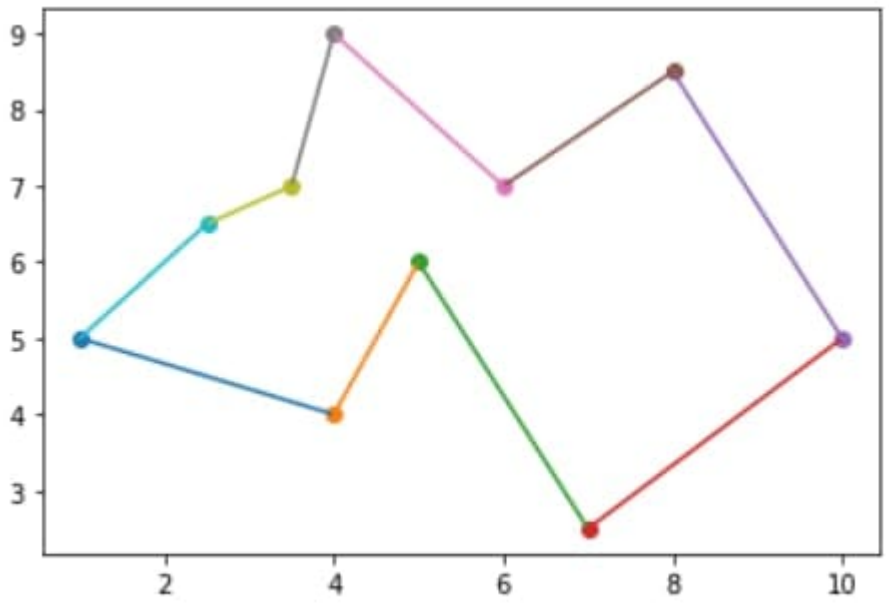
Повертає:
Масив цілих чисел.

6.3 Робота алгоритму на прикладі

Наведемо результат роботи алгоритму для масиву точок:

$(1, 5), (4, 4), (5, 6), (7, 2.5), (10, 5), (8, 8.5), (6, 7), (4, 9), (3.5, 7), (2.5, 6.5)$

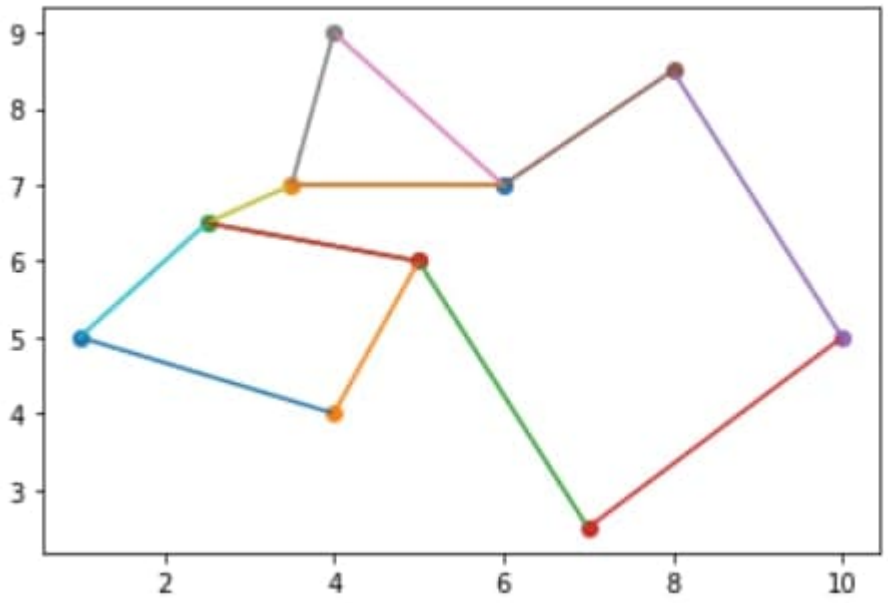
Зобразимо отриманий багатокутник:



Далі створимо список точок eventQ від найвищої до найнижчої. Ініціалізуємо бінарне дерево tree, що зберігатиме ребра зліва направо та словник supports ребер та відповідних їм допоміжних точок. Опрацюємо кожну точку з eventQ за допомогою функції handlePoint:

```
eventQ = sorted(dceList.getPoints().copy(),
                key=lambda point: (point.coords()[1],
                                   point.coords()[0]), reverse=True)
tree = bintrees.BinaryTree()
supports = {}
for point in eventQ:
  handlePoint(point, dceList, tree, supports)
```

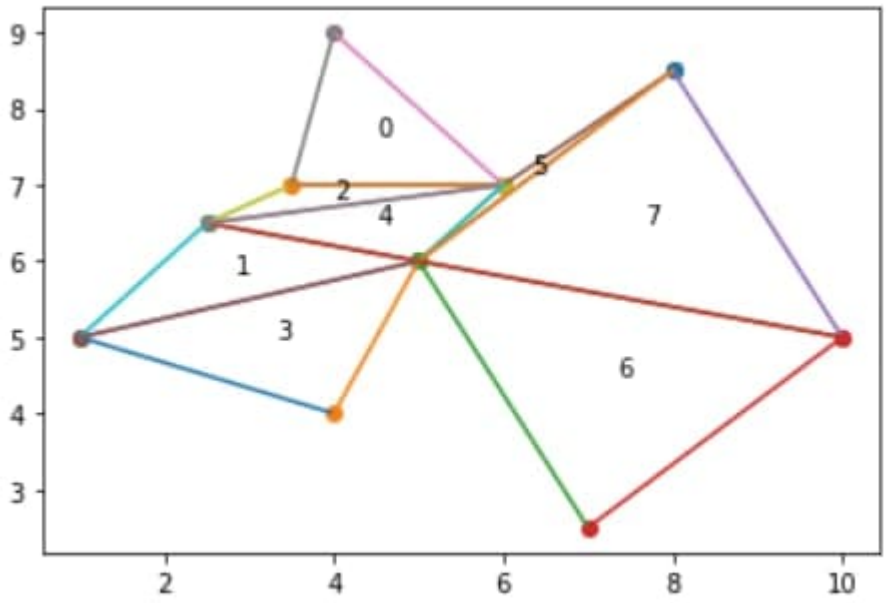
Отримали розбиття багатокутника на y-монотонні грані:



Далі для кожної грані, що не є трикутною проводимо триангуляцію. Заради цього для кожної грані визначається послідовність точок від найвищої до найнижчої eventQ, знаходяться лівий та правий ланцюги, знаходитяться найвища та найнижча точки і ініціалізується стек stack точок необхідний для виконання алгоритму, що описаний в розділі 5.4 :

```
eventQ = sorted(points.copy(),
    key=lambda point: (point.coords()[1], point.coords()[0]), reverse=True)
verts=list(map(lambda point: point.coords(), points))
rightC, leftC = chains(verts, dceList)
top, low = getTopLowPoints(verts, dceList)
stack = []
stack.append(eventQ[0])
stack.append(eventQ[1])
```

Після триангуляції отримуємо триангульований багатокутник:



Можемо вважати кожну грань вершиною графа, що з'єднана ребром з іншою вершиною у випаку коли вони є суміжними гранями. Таким чином далі ми працюватимемо із зв'язним графом. Знаходимо його таблицю суміжності за допомогою наступних функцій:

```
def getIncFaces(self, face):
  ar = [0]*len(self.getFaces())
  edge = face.getOutComponent()
  nextEdge = edge
  while True :
    if nextEdge.getTwin() != None:
      ar[nextEdge.getTwin().getFace().getName()]=1
    nextEdge = nextEdge.getNext()
    if nextEdge == edge :
      break
  return ar
```

```
def getIncMatrix(self):
  faces = sorted(self.getFaces().copy(), key= lambda face: face.getName())
  ar = []
  for face in faces:
    ar.append(self.getIncFaces(face))
  return np.matrix(ar)
```

Та застосувавши алгоритм, що описаний в 5.5 отримаємо таку матрицю відстаней:

```
matrix([[0, 3, 1, 4, 2, 3, 5, 4],
        [3, 0, 2, 1, 1, 2, 4, 3],
        [1, 2, 0, 3, 1, 2, 4, 3],
        [4, 1, 3, 0, 2, 3, 5, 4],
        [2, 1, 1, 2, 0, 1, 3, 2],
        [3, 2, 2, 3, 1, 0, 2, 1],
        [5, 4, 4, 5, 3, 2, 0, 1],
        [4, 3, 3, 4, 2, 1, 1, 0]])
```

Такі індекси граней, що є центральними:

```
[4, 5]
```

Такі індекси граней, що є периферійними:

```
[0, 3, 6]
```

7 Висновки

В ході дослідження були проаналізовані центр та периферія різних класів зв'язних графів. Особлива увага була приділена унікально ексцентрично точковим графам. Також, ми ознайомилися з концепцією відносних центрів.

Новизна робити полягає доведенні твердження про відсутність циклів довжини більше 2 в орграфах ексцентриситетів у.е.т. графів (твердження 3.18), в повному описі орграфів ексцентриситетів для у.е.т. графів блоків (теорема 3.21, отриманні характеризації самих у.е.т. графів блоків (теорема 3.23). Ці результати ми доповіли та опублікували в тезах X Всеукраїнської наукової конференції молодих математиків [4].

Практична частина роботи складається з реалізації алгоритму знаходження центру та периферії простого багатокутника мовою програмування Python. Цей алгоритм є комплексним і включає в себе алгоритм монотонізації простого багатокутника, алгоритм триангуляції монотонного багатокутника та алгоритм знаходження матриці відстаней графа за його ма рицею суміжності.

Література

[1] M. de Berg, O. Cheong, M. van Kreveld, M. Overmars, *Computational Geometry - Algorithms and Applications, 3rd Ed* — 1997. —P. 30–60.

[2] K. Balakrishnan, M. Changat, R. Kumar, P.G. Narasimha-Shenoi, A. Sreekumar, On the Center Sets of Some Graph Classes. In: Algorithms and Discrete Applied Mathematics, Springer, 2016, pp. 240–253.

[3] F.Buckley and F. Harary, Distance in Graphs, Addison-Wesley,Redwood City,CA, 1990, p. 213.

[4] A. Hak, V. Haponenko, S. Kozerenko, *Eccentric digraphs of unique point eccentric graphs //X* All-Ukrainian Conference of Young Scientists in Physics and Mathemathics, April 16-17, Kyiv, Ukraine. — 2021. —P. 84–85.

[5] F. Harary, A characterization of block-graphs. Canad. Math. Bull. 6 (1963), 1 - 6.

[6] E. Howorka, On metrick properties of certain clique graphs, J. Comb. Theory (B) 27 (1979), 67-74

[7] F. Gobel, H.J. Veldman, 1986, 'Even graphs', Journal of graph theory, vol. 10, no. 2, pp. 225-239.

[8] Parthasarathy K.R., Nandakumar R. *Unique eccentric point graphs //* Discrete Math. — 1983. — V. 46. — P. 69–74.