МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики
(повне найменування назва факультету)

Кафедра програмування
(повна назва кафедри)

Дипломна робота

Візуалізація та аналіз графів згенерованих за допомогою моделі Ердеша-Реньї

Виконав: студент групи ПМІ-41
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)

Караваєв К. О
(підпис) (прізвище та ініціали)

Керівник
проф. Заболоцький Т. М.
(підпис) (прізвище та ініціали)

Рецензент
(підпис) (прізвище та ініціали)

Львів – 2023

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики_____

Кафедра програмування_____

Спеціальність «122 Комп'ютерні науки»_____

«ЗАТВЕРДЖУЮ»

Завідувач кафедри

_____

"____"_____20___року

З А В Д А Н Н Я

НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

_____Караваєва Кирила Олеговича_____
(прізвище, ім'я, по батькові)

1 .Тема роботи
Візуалізація та аналіз графів згенерованих за допомогою моделі_____
Ердеша-Реньї_____

керівник роботи Заболоцький Тарас Миколайович, доктор економічних_____
наук, професор,_____
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені Вченою радою факультету від " 13 " вересня 2022 року № 15

2. Строк подання студентом роботи 13.06.2023 року _____

3. Вихідні дані до роботи
Література та інтернет-ресурси за тематикою роботи_____
_____
_____
_____
_____

4. Зміст магістерської роботи (перелік питань, які потрібно розробити)
а) дослідити поняття теорії графів_____
б) встановити зв'язок між укладками та візуалізацією графу_____
в) дослідити модель Ердеша-Реньї_____
г) реалізувати програмний застосунок_____

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
а) малюнки укладки графу_____
б) скріншоти програмної реалізації застосунку_____
в) приклад результату роботи програми_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 13 вересня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	*Аналіз предметної області*	*вересень*	*Виконано*
2	*Постановка задачі*	*жовтень*	*Виконано*
3	*Дослідження теорії графів*	*листопад*	*Виконано*
4	*Дослідження засобів програмування, необхідних для реалізації теорії графів*	*грудень*	*Виконано*
5	*Дослідження моделі Ердеша-Реньї*	*лютий*	*Виконано*
6	*Дослідження зв'язку між укладкою та візуалізацією графу*	*березень*	*Виконано*
7	*Реалізація програмного застосунку*	*квітень*	*Виконано*
8	*Оформлення дипломної роботи*	*травень*	*Виконано*

Студент _____
(підпис) (прізвище та ініціали)

Керівник роботи_____
(підпис) (прізвище та ініціали)

АВТОРЕФЕРАТ

Дипломної роботи Караваєва Кирила Олеговича «Візуалізація та аналіз графів згенерованих за допомогою моделі Ердеша-Реньї.»

Роботу виконано на кафедрі програмування факультету прикладної математики та інформатики Львівського національного університету імені Івана Франка.Робота має на меті дослідити теорію графів та модель Ердеша-Реньї, проаналізувати основи укладки графу та, на основі встановленого взаємозв’язку між цими поняттями, реалізувати програмний застосунок. Робота складається зі вступу, трьох розділів та висновків. У вступі наведено мету роботи, обґрунтовано її актуальність. Перший розділ присвячено опису основ теорії графів та моделі Ердеша-Реньї. Розглянуто базові поняття теорії графів та принцип роботи моделі Ердеша-Реньї. У другому розділі розкрито основи візуалізації графів. Також описано зв’язок між укладкою графа та його візуалізацією. У третьому розділі описано розробку програмного застосунку..Також наведено реалізацію основних методів, методи їхнього тестування. У висновках підсумовано результати роботи.

Всього наведено 10 формул, 12 ілюстрацій з реалізацією алгоритмів та прикладами укладки графу.

ВСТУП

Модель Ердеша-Реньї є однією з найпоширеніших моделей для генерації випадкових графів. Ця модель дозволяє створювати графи з особливостями, що може бути корисним у різних областях, включаючи соціальні мережі, транспортні системи, біологію та ін.

Візуалізація графів є важливим інструментом для розуміння їх структури та властивостей. Аналіз та інтерпретація великих графів може бути складною задачею без візуального представлення. Візуалізація допомагає виявляти взаємозв'язки, класифікувати та визначати ключові елементи графів.

Графи, згенеровані за допомогою моделі Ердеша-Реньї, можуть відображати структуру соціальних мереж, де вузли представляють особи, а ребра - зв'язки між ними. Аналіз та візуалізація таких графів можуть допомогти вивчати властивості соціальних мереж, ідентифікувати ключових впливових осіб, виявляти спільноти та тенденції.

Модель Ердеша-Реньї також може бути використана для моделювання транспортних мереж, де вузли представляють різні локації, а ребра - зв'язки між ними. Аналіз графів транспортних мереж може допомогти в плануванні оптимальних маршрутів, визначенні слабких місць та оптимізації ресурсів.

Використання моделі Ердеша-Реньї дозволяє створювати випадкові графи, які можуть відображати взаємозв'язки в біологічних системах, таких як біологічні молекули, генетичні мережі та інші біологічні процеси. Аналіз графів біологічних мереж може допомогти в розумінні взаємодії між елементами системи та виявленні біологічних функцій та властивостей.

Отже, візуалізація та аналіз графів, згенерованих за допомогою моделі Ердеша-Реньї, є актуальними у багатьох сферах, де графові

структури використовуються для репрезентації та дослідження складних взаємозв'язків та систем.

РОЗДІЛ 1. ТЕОРІЯ ГРАФІВ

1.1 Основні поняття теорії графів

Теорія графів є галуззю математики, що вивчає взаємозв'язки між об'єктами, представленими у вигляді точок (вузлів) і з'єднань (ребер) між ними. Графи використовуються для моделювання і аналізу різних систем, де важливим є взаємозв'язок між їх складовими частинами. Основні поняття теорії графів включають:

1. Вершини або вузли представляють об'єкти, які ми хочемо дослідити або зобразити. Кожна вершина може мати певні характеристики або властивості, пов'язані з об'єктом, який вона представляє.
2. Ребра або з'єднання є лініями, що з'єднують вершини між собою. Вони вказують на наявність взаємозв'язку або відношення між двома вершинами. Ребра можуть мати напрямок або бути ненапрямленими.
3. Графи можуть бути напрямленими або ненапрямленими. У напрямленому графі ребро вказує на одностороннє відношення між вершинами, тоді як у ненапрямленому графі ребра не мають визначеного напрямку.
4. Дві вершини є суміжними, якщо вони з'єднані ребром. Вони можуть бути суміжними безпосередньо або через інші вершини.
5. Ступінь вершини - це кількість ребер, що виходять або входять в цю вершину. У напрямленому графі відрізняють ступінь виходу та ступінь входу.
6. Петля - це ребро, що з'єднує вершину з самою собою. Кратні ребра - це кілька ребер, що з'єднують одну пару вершин.

7. Підграф - це граф, який утворений підмножиною вершин і ребер початкового графа. Він містить лише деякі вершини і ребра, які пов'язані з цими вершинами.
8. Граф називається зв'язним, якщо між будь-якою парою вершин існує шлях, що їх з'єднує. Якщо в графі є дві і більше компоненти зв'язності, то такий граф називається незв'язним.
9. Дерево - це ациклічний зв'язний граф без петель. Всі вершини дерева пов'язані між собою ребрами, і кожна вершина, крім кореня, має одного батька.
10. Теорія графів також включає в себе розробку різних алгоритмів, які застосовуються для розв'язання задач, пов'язаних з графами. Деякі з них включають пошук в ширину, пошук в глибину, алгоритм Дейкстри та алгоритм Пріма для мінімального остовного дерева.

Ці основні поняття теорії графів є фундаментальними для вивчення та розуміння графових структур і допомагають вирішувати різні задачі в багатьох галузях науки, технології та інформатики.

1.2 Степені вершин графу. Теорема про степені вершин.

Степінь вершини в графі вказує на кількість ребер, які з'єднують цю вершину з іншими вершинами графа. Математично, степінь вершини v позначається як deg(v) і підраховується як кількість ребер, що з'єднуються з вершиною v.

Теорема про степені вершин стверджує, що сума степенів усіх вершин в простому (неорієнтованому) графі дорівнює подвоєному числу ребер у графі. Формально, якщо G є простим графом з n вершинами та m ребрами, то теорема про степені вершин може бути сформульована так:

deg(v1) + deg(v2) + ... + deg(vn) = 2m,

де deg(vi) - степінь i-ої вершини, m - кількість ребер в графі[5].

Ця теорема є важливою у теорії графів і може бути використана для розв'язання різних задач, таких як визначення кількості ребер у графі за відомими степенями вершин або виведення властивостей графа на основі його степенів вершин.

Для перевірки коректності аналізу графу введемо дві наступні теореми:

1. Довільний (простий та неорієнтований) граф містить принаймні дві вершини однакового степеня.

Доведемо теорему. Припустимо , що в графі G всі вершини мають різні степені . Тоді , оскільки степінь вершини є цілим числом у межах від 0 до $n_v - 1$ (всього n_v, можливих значень) , граф G має містити вершини всіх степенів від 0 до $n_v - 1$. Отже , граф G має містити ізольовану вершину v_0 ($d_v = 0$) та вершину v_{n_v-1} степеня $n_v - 1$, що неможливо : вершина v_{n_v-1} має бути суміжною з усіма вершинами графу G , зокрема з ізольованою вершиною v_0.

2. Сумма степенів усіх вершин графу дорівнює подвійній кількості ребер:

$$\sum_{v \in V} d_v = 2 * n_e, \text{ де } n_e - \text{ кількість ребер у графі}$$

Застосуємо метод математичної індукції за n_e.

1. База індукції $n_e = 0$. Для порожнього графу твердження теореми справджується.
2. Припущення індукції . Нехай для графів з $n_e \leq n$ твердження теореми справедливе.

3. Крок індукції . Нехай граф G має $n_e = n + 1$ ребро . Для доведення теореми видалимо у графі G довільне ребро. Отримуемо граф G_1 з кількістю ребер $n_e - 1 = n$, для якого твердження теореми, за припущенням індукції, справедливе. Отже маємо:

$$\sum_{v \in V} d_v = 2(n_e - 1), \text{ де } d_v - \text{степінь вершини у графі } G_1$$

Отже, оскільки видалено ребро збільшувало суму степенів степенів вершин на 2(по 1 за кожну вершину інцидентну ребру) маємо наступне:

$$\sum_{v \in V} d_v = 2(n_e - 1) + 2 = 2 * n_e$$

Ось деякі властивості графа, які можуть бути виведені на основі його степенів вершин:

1. Графи зі степенями 0: Якщо вершина має степінь 0, це означає, що вона не має жодних зв'язків з іншими вершинами. Такі вершини називаються ізольованими вершинами.
2. Графи зі степенями 1: Якщо вершина має степінь 1, це означає, що вона має лише один зв'язок з іншою вершиною. Такі вершини називаються листками або кінцевими вершинами.
3. Максимальна та мінімальна степені: Максимальна та мінімальна степені вершини вказують на найбільш та найменш "зв'язаних" вершин у графі відповідно. Це може дати уявлення про те, наскільки густо зв'язаний граф.
4. Регулярні графи: Якщо всі вершини в графі мають однакову степінь, то граф називається регулярним графом. Такі графи мають ряд властивостей, таких як рівномірний розподіл зв'язків між вершинами.

5. Графи з високими степенями: Вершини з високими степенями мають багато зв'язків з іншими вершинами. У таких графах можуть виникати особливі властивості, такі як шляхи низької довжини, густі підграфи або велика кількість циклів.

1.3 Зв’язність графів

Зв'язність графів є важливим поняттям в теорії графів, яке визначає, наскільки сильно зв'язаний або незв'язаний граф. Зв'язність вказує на наявність шляхів між вершинами графа та відображає можливість досягти будь-якої вершини з будь-якої іншої вершини графа.

Існує декілька рівнів зв'язності, які можуть бути застосовані до графів:

1. Слабка зв'язність: Граф називається слабко зв'язним, якщо є принаймні один шлях між будь-якою парою вершин, які можуть бути досягнуті за допомогою напрямлених або ненапрямлених ребер. Це означає, що є можливість переходу від будь-якої вершини до будь-якої іншої вершини графа, можливо, через проміжні вершини.
2. Сильна зв'язність: Граф називається сильно зв'язним, якщо є принаймні один напрямлений шлях між будь-якою парою вершин. Це означає, що з кожної вершини можна досягти будь-яку іншу вершину графа.
3. Компоненти зв'язності: Якщо граф не є повністю зв'язним, то він може містити кілька компонентів зв'язності. Компонента зв'язності - це підграф, в якому будь-яка вершина досяжна з будь-якої іншої вершини у цій компоненті, але не досяжна з вершини, що належить до іншої компоненти зв'язності.
4. Сильно зв'язні компоненти: Це поняття використовується тільки для напрямлених графів. Сильно зв'язна компонента - це максимальна підмножина вершин у графі, в якій будь-яка вершина досяжна з будь-якої іншої вершини у цій компоненті[5].

Знання про зв'язність графів має важливе значення в багатьох областях, таких як маршрутизація мереж, соціальна мережна аналітика, транспортні мережі, комп'ютерна графіка та інші.

1.4 Базові поняття про випадкові графи

Випадкові графи - це спосіб моделювання графів з використанням ймовірнісних розподілів. Вони дозволяють описати графи, створені випадковим процесом або за допомогою ймовірнісного розподілу. Теорія випадкових графів поєднує в собі концепції з теорії графів та теорії ймовірностей. З математичної точки зору, вони допомагають відповісти на питання про властивості типових графів. Випадкові графи мають практичне застосування в різних галузях, де необхідно моделювати складні мережі. Існує багато моделей випадкових графів, які відображають різноманітні типи складних мереж у різних галузях. У математичному контексті термін "випадковий граф" майже завжди посилається на модель випадкового графа Ердеша-Реньї[2].

1.5 Модель випадкових графів Ердеша-Реньї

Зафіксуємо довільне натуральне число n і розглянемо множину $V = \{1, ..., n\}$. Це буде безліч вершин довільного графа. Власне випадковими будуть лише ребра. Нехай $N = C_n^2$і $e1$, ..., e_n є всі можливі ребра, які можна провести на парах елементів з V, якщо будуємо ми саме граф. Іншими словами $e1$, ..., e_n – ребра повного графа K_n. Задамо деяке $p \in [0, 1]$ та станемо вибирати ребра з множини $\{e1$, ..., $e_n\}$ згідно схеми Бернуллі з ймовірністю успіху p, тобто у разі успіху кладемо чергове ребро в список ребер, що будується E, а у разі невдачі - не кладемо. Виникає випадковий граф $G = (V, E)$. Фактично, як і в схемі Бернуллі, граф - це

послідовність $\omega = (x_1, \ldots, x_n)$ з нулів і одиниць. Просто тепер ми послідовність у певному сенсі представляємо графом: $x_i = 1$ означає, $e_i \in E$; $x_i! = 1$ – значить, $e_i ! \in E$. У результаті маємо імовірнісний простір $G(n, p) = (\Omega_n, F_n, P_n, p)$, у якому, виконуються наступні твердження:

$|\Omega_n| = 2^N = 2^{C_n^2}$, $P_{n,p}(G) = p^{|E|} q^{C_n^2 - |E|}$. Інакше можна ще сказати так: у моделі Ердеша – Реньї кожне ребро незалежно від решти всіх ребер входить у випадковий граф з ймовірністю p[3]. Хоча через $G(n, p)$ ми позначили імовірнісний простір, ми часто говоритимемо про «моделі $G(n, p)$». Кожна подія з F є множиною графів. Спробуємо природньо інтерпретувати події як властивості графів. Справді: що означає вважати ймовірність того, що граф має властивість А? Це означає описати множину А тих графів, які цією властивістю володіють, а потім знайти ймовірність А. Наприклад, якщо А - множина всіх зв'язних графів, тоді ймовірність події «випадковий граф є зв'язаний» і є $P_{n,p}(A) = \sum_{G \in A} P_{n,p}(G)$. Наука про випадкові графи вивчає ймовірність тих чи інших властивостей графів. Найбільший інтерес становить «динаміка» цих ймовірностей, тобто їх зміна зі зростанням n. Кількості вершин і справді нічого не заважає зростати. При цьому ймовірність виникнення ребра графа також цілком може еволюціонувати. Іншими словами, ми інтерпретуємо $G(n, p)$ навіть не як схему Бернуллі, а як схему серій.

Нехай $\{A_n\}$- деяка послідовність властивостей-подій в просторах $G(n, p)$. Наприклад, A_n полягає у зв'язності графа, і тоді залежності від n, по суті, немає, або A_n зводиться до наявності в графі не менше n трикутників (трійок вершин, попарно з'єднаних ребрами), і тоді індекс n важливий, і

т.д. Ми говоримо, що $\{A_n\}$ виконано тільки, якщо $P_{n,p}(A_n) \to 1$ при $n \to \infty$. Інакше кажучи, імовірнісна частка тих графів, у яких властивість виконується, зі зростанням числа вершин сильно ближче до одиниці. Ми побачимо надалі масу ситуацій, коли виконання тієї чи іншої властивості майже напевно спричиняє справді дивовижні наслідки. Модель $G(n, p)$ була запропонована П. Ердешем та А. Реньї наприкінці 50-х років XX століття. З того часу наука зробила крок далеко вперед, і про багато аспектів проблематики півстолітньої давності відомо «майже все». Проте навіть тут залишається безліч невирішених завдань.

РОЗДІЛ 2. ВІЗУАЛІЗАЦІЯ ГРАФУ

2.1 Базові поняття

Задача візуалізації графу є досить складною, має багато підходів для її вирішення та напряму залежить від укладки графу.

Укладка графу - таке відображення графа для деякого простору в точки і криві такі, що різним вершинам відповідають різні точки, а криві, що відповідають різним ребрам, перетинаються тільки в інцидентних цим ребрах вершинах. Якщо казати більш прости словами то це спосіб зіставити кожній вершині графа координати. Зазвичай йдеться про координати на площині, хоча взагалі кажучи це не обов'язково має бути площина.

Найпростіші способи укладки графу:

1. Випадково. Насправді ми можемо взагалі не думати та просто зробити укладку випадково. Очевидно, що в результаті такої укладки візуалізація буде виглядати максимально не зрозуміло та не нести за собою ніякого сенсу.

2. Укладка графу по колу. Зробимо уявне коло та будемо розставляти вершини на ньому з однаковим інтервалом. Мінус цього методу в тому, що ми ніяк не відрізняємо вершини одну від одної(наприклад за степенем).

2.2 Силові алгоритми укладки графу

Силові алгоритми(force-directed algorithms) укладки графів на площині відносяться до алгоритмів зображення графа, заснованих на фізичних аналогіях. Як основний інструмент використовується фізична модель, в якій виникає система певних сил. В основі алгоритму може лежати один тип сили, або комбінація кількох. Наприклад: сила тяжіння, відштовхування, тертя, магнітного поля тощо[1].

2.3 Вимоги до графу

Якщо вхідний граф є простим, зв'язним та неорієнтованим то зображення такого графа може розглядатися як система тіл із силами, взаємодіючими між тілами. Будемо, вважати вершини графа тілами, а ребра пружинами. У цьому випадку алгоритм повинен знаходити конфігурацію тіл з локально мінімальною енергією тобто конфігурацію рівноваги сил, у якій кожне тіло займає таку позицію, що сума всіх сил, що додаються до тіла, дорівнює нулю. Графи, намальовані з допомогою силових алгоритмів виходять естетично привабливими, є більш симетричними та у випадку планарного графу ребра не будуть перетинатися[4].

2.4 Як працюють алгоритми?

Варіативність силових алгоритмі є дуже велика. Ми можемо використовувати різні фізичні сили та отримувати зовсім іншу укладку для того й самого графу. Але підхід до побудови силового алгоритму є

однаковим. На першому етапі вершини розміщуються на площині випадковим чином. Другий етап - це послідовність ітерацій до стану рівноваги, на кожній з яких обчислюються для всіх вершин сили і для тих, для якихось відбувається переміщення вершини в напрямку цієї сили на відстань, пропорційне модулю сили. Пропорція зсуву одна для всіх вершин є ще одним параметром алгоритму. Алгоритм досить простий, але не завжди дає найшвидший спосіб досягнення рівноваги. Разом з тим, він дозволяє здійснити інтуїтивно зрозумілий плавний перехід від випадкових розміщень до конфігурацій рівноваги сил. У своїй роботі я буду використовувати два алгоритми.

1. Алгоритм Камада-Кавай
2. Алгоритм Фрюхтермана-Рейнгольда

Алгоритм Камада-Кавай

1. Розрахуємо відстані між усіма вершинами графу
2. Вершини графа розмістимо у випадкові координати
3. Виберемо вершину P на яку діє максимальна сила
4. Інші вершини фіксуємо, енергія системи мінімізуємо
5. Повторюємо кроки 3-4

Опис моделі систем:

$$d_{ij} - \text{ довжина найкоротшого шляху між вершинами } i,\ j$$

$$l_{ij} = L * d_{ij} - \text{ довжина пружини між вершинами } i,\ j$$

$$L - \text{ розмір пікселя на дисплеї}$$

$$k_{ij} = \frac{K}{d_{ij}^2} - \text{ сила пружини між вершинами } i,\ j$$

Для підрахунку енергії систему обчислимо наступну формулу[1].

$$E = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{2} k_{ij} (||p_i - p_j|| - l_{ij})$$

Недоліки алгоритму:

1. Для підрахунку найкоротших шляхів між всіма вершинами нам треба багато часу
2. Для зберігання даних потребує багато пам’яті.

Алгоритм Фрюхтермана-Рейнгольда

Різниця між цим алгоритмом та минулим в тому, що тепер ми будемо рахувати дві сили. Для всіх вершин сили відштовхування, а для суміжних сили тяжіння. Таким чином ми будем отримувати більш згрупований та рівновіддалений малюнок.

РОЗДІЛ 3. ПРОЕКТ

3.1 Аналіз схожих застосунків

На сьогоднішній день існує багато програм для візуалізації графів. Деякі з них спроектовані для роботи з великими графами, інші зосереджені на створенні ефектних візуалізацій для невеликих графів. Ось кілька популярних програм для візуалізації графів:

1. Gephi є потужним інструментом для аналізу та візуалізації графів. Він надає розширені можливості для вивчення структури графа, включаючи вбудовані алгоритми для виявлення спільнот та використання розкладів для подальшого аналізу.
2. Cytoscape є популярним інструментом для візуалізації молекулярних і біологічних мереж. Він має широкий набір функцій для редагування та візуалізації графів, а також підтримку розширень для аналізу даних.
3. NetworkX є бібліотекою Python для роботи з графами. Вона надає можливості для побудови, маніпулювання та візуалізації графів. NetworkX добре підходить для використання в програмах машинного навчання та аналізу соціальних мереж.

4. Graphviz є пакетом програмного забезпечення для візуалізації графів, який надає ряд інструментів та бібліотек для генерації зображень графів з описової мови DOT. Він має простий синтаксис і може бути використаний як самостійний інструмент або вбудований в інші програми.
5. Tableau є інструментом для візуалізації даних, який також має можливості для роботи з графами. Він дозволяє створювати інтерактивні візуалізації графів, зокрема мережеві діаграми, які можуть бути використані для відображення зв'язків та залежностей у даних.

Зважаючи на велику кількість програм для візуалізації графів, наведу опис плюсів і мінусів кожної з вищезазначених програм:

Gephi:

Плюси:

- Має потужні алгоритми для виявлення спільнот, центральності, кластеризації та інших аналізів графів.
- Надає широкий набір інструментів для маніпулювання візуалізаціями, зокрема для керування розміщенням вузлів та ребер.
- Підтримує імпорт даних з різних форматів графів.

Мінуси:

- Інтерфейс може бути складним для новачків і вимагати певного часу для оволодіння.
- Може бути важким у використанні для великих графів, оскільки обробка та візуалізація можуть бути повільними.

Cytoscape:

Плюси:

- Має потужні функції для візуалізації біологічних та молекулярних мереж.
- Підтримує розширення для розширеного аналізу даних та візуалізації.
- Надає можливості для інтерактивної маніпуляції візуалізаціями та взаємодії з даними.

Мінуси:

- Може бути спрямований переважно на біологічні дослідження, що може обмежити його застосування в інших галузях.
- Деякі функції можуть бути складними для освоєння і вимагати додаткових знань.

NetworkX:

Плюси:

- Легко використовувати в програмах Python.
- Має багато вбудованих алгоритмів для аналізу та модифікації графів.
- Може бути використаний для автоматичної генерації візуалізацій графів з даних.

Мінуси:

- Візуалізація графів в NetworkX може бути обмеженою, іноді потрібно додатково використовувати інші бібліотеки для більш привабливого візуального представлення.
- Може бути менш потужним для великих графів порівняно з спеціалізованими програмами.

Graphviz:

Плюси:

- Простий синтаксис і можливість швидко створювати зображення графів.

- Має вбудовані алгоритми для автоматичного розміщення вузлів та ребер.

Мінуси:

- Інтерактивність та маніпулювання візуалізаціями обмежені у порівнянні з іншими програмами.
- Вимагає знань мови DOT для повноцінного використання всіх можливостей.

Tableau:

Плюси:

- Має потужні функції для візуалізації даних, включаючи можливість створювати мережеві діаграми.
- Інтерактивність та можливості фільтрації дозволяють проводити детальний аналіз графів.

Мінуси:

- Зазвичай використовується для загальної візуалізації даних, тому його функціональні можливості для візуалізації графів можуть бути обмеженими порівняно зі спеціалізованими програмами.
- Платформа Tableau потребує підписки для повного використання всіх його функцій.

Вибір певної програми для візуалізації графів залежить від ваших конкретних потреб, навичок, розміру графа та контексту аналізу даних.

3.2 Основна ідея

Граф - це є фундаментальна структура для збереження більшості існуючих даних. У вигляді графу ми можемо зберегти як транзакції біткоїну так й рекомендацію музики та фільмів. Але візуалізація графів потребує вибору

правильного методу укладки. Тому я вирішив провести дослідження видів цих укладок та потім ще й проаналізувати отриманні графи.

3.3 Які технології використовував?

Середовищем розробки виступає Jupyter Notebook. Обрав саме його через те, що він дає змогу зручно працювати з графіками та віджетами.

Для роботи з графами використовував igraph. Обрав саме цю технологію через гнучкість та велику функціональність.

Також додатково для візуалізації використовував графічну бібліотеку cairo.

3.4 Програмна реалізація

Спочатку я описав метод який генерую випадковий граф за моделлю Ердеша-Реньї.

```
def random_graph(n, p):
    graph = Graph()

    N_range = range(n)
    graph.add_vertices(N_range)

    edges = []
    for pair in itertools.permutations(N_range, 2):
        if rnd.random() < p:
            #print(pair)
            edges.append(pair)

    graph.add_edges(edges)
    return graph
```

Рис. 1.1 Опис функції для моделі Ердеша-Реньї

Ця функція повертає граф. Спочатку я створю порожній граф, потім додаю до нього n вершин. Далі за допомогою бібліотеки itertools генерую всі можливі варіанти пар вершин, які можуть бути поєднанні ребрами. Далі

генерую випадкове число з інтервалу [0, 1], якщо вони вийде більше ніж ймовірність додавання ребра до графу, то відповідно це ребро додаю.

Далі я реалізував візуалізацію отриманого графу.

```
def visualization(g, lt):
    out_fig_name = "graph.eps"

    visual_style = {}

    colours = ['#fecc5c', '#a31a1c']


    visual_style["bbox"] = (3000,3000)
    visual_style["margin"] = 17

    visual_style["vertex_color"] = 'grey'

    visual_style["vertex_size"] = 20

    visual_style["vertex_label_size"] = 8

    visual_style["edge_curved"] = False

    my_layout = g.layout(lt)
    visual_style["layout"] = my_layout

    return plot(g, "social_network.png",**visual_style)
```

Рис 1.2 Код функції з параметрами візуалізації

Ця функція повертає графік графу. Спочатку я зберігаю результат візуалізації до файлу graph.eps. Цей файл потрібен для того, щоб відтворити отриманий графік в інших програмах для візуалізації. Наприклад:GraphViz або Gephi. Далі я задаю візуальний стиль графіку, тобто колір, розмір вікна графіку, товщину, колір та розмір вершин,розмір тексту підписів вершин та забороняє ребра бути під кутом. Далі я задаю спосіб укладки графу. Після цього крім повернення графіку ще й зберігаю його в файл.

Результат візуалізації для різних укладок:

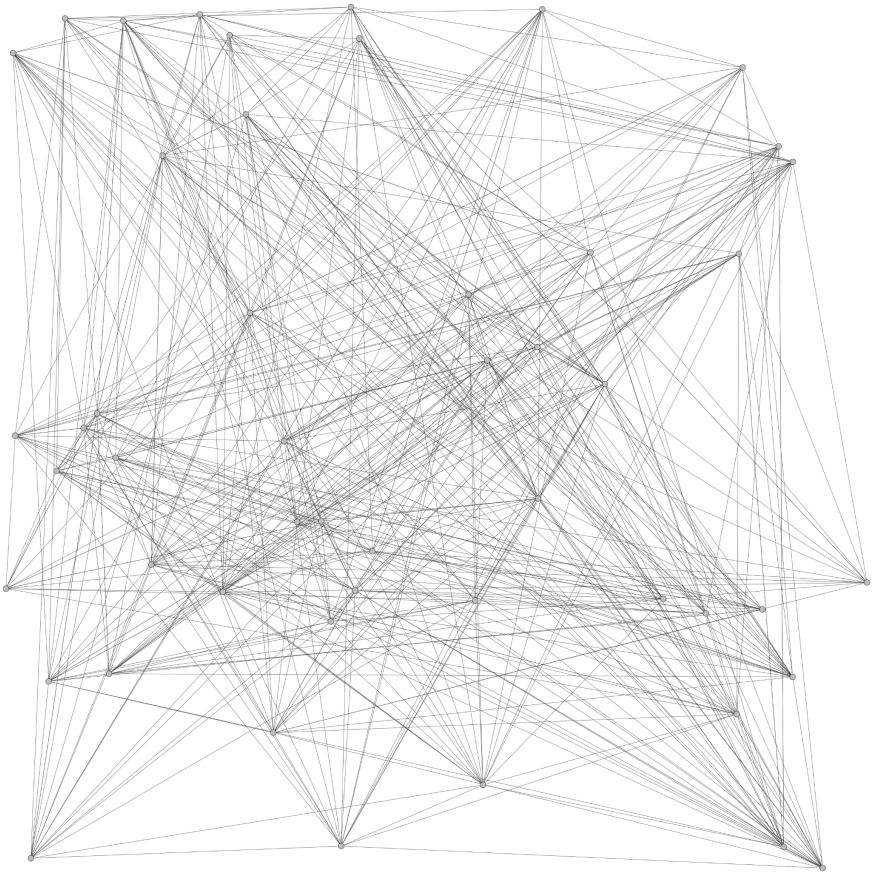


Рис 1.3 Випадкова укладка

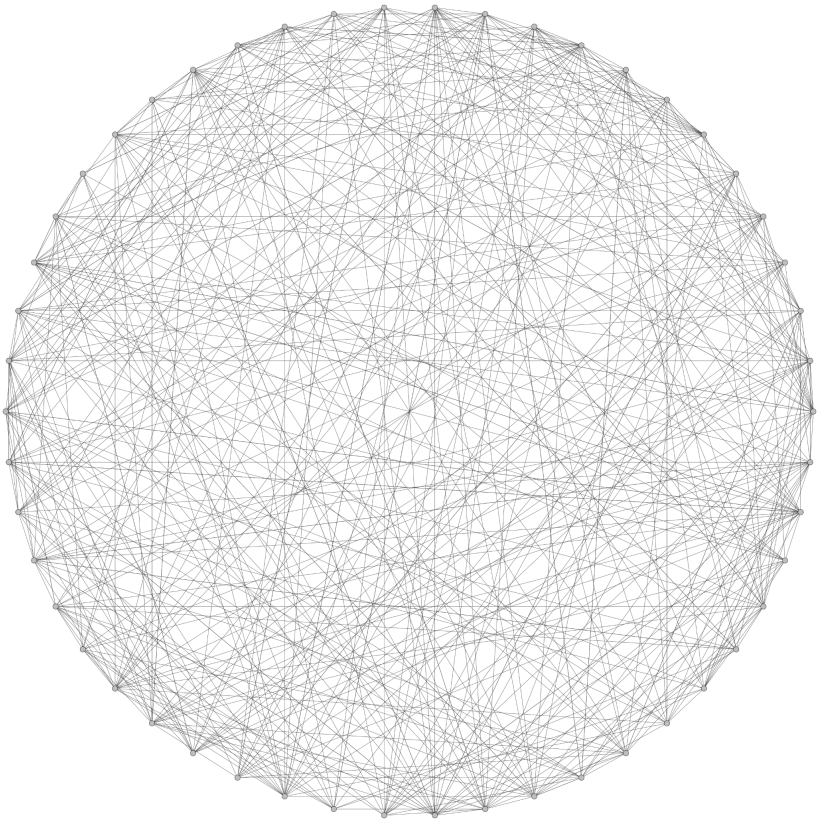


Рис. 1.4 Укладка по колу

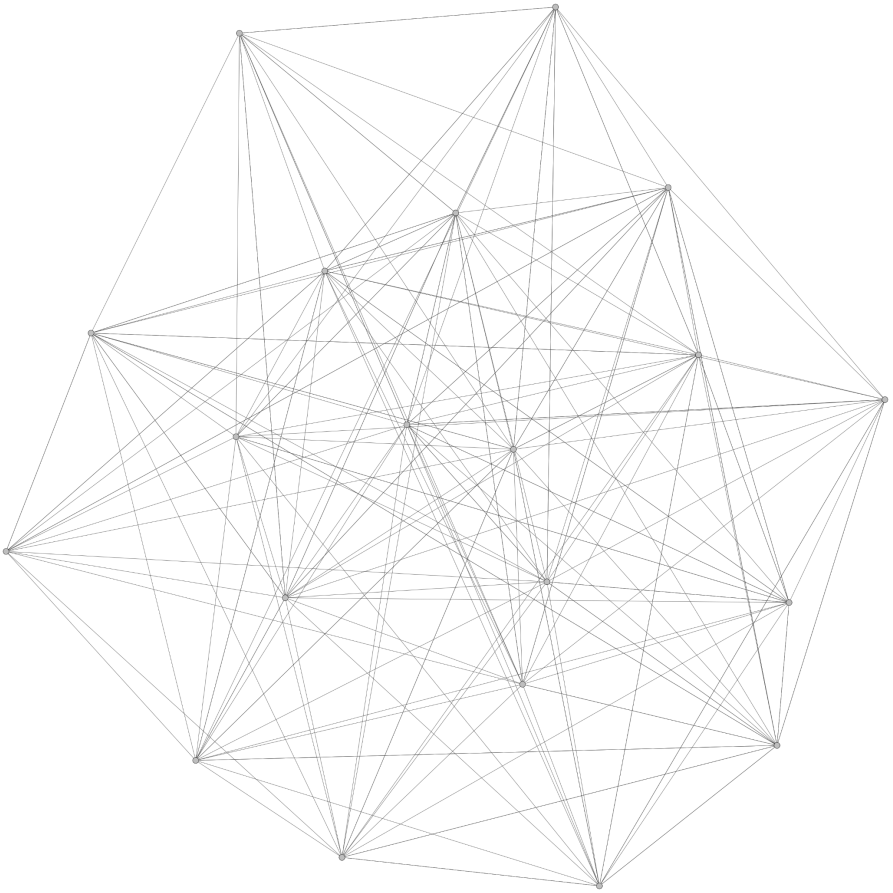


Рис 1.5 Укладка за допомогою алгоритму Камада-Кавай

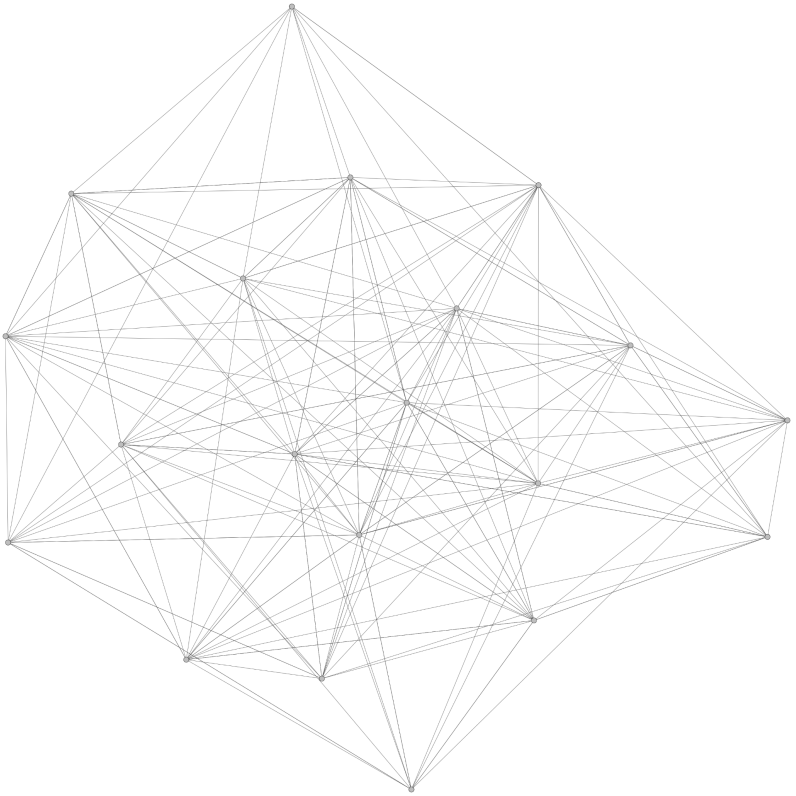


Рис 1.6 Укладка за допомогою алгоритму Фрюхтермана-Рейнгольда

Наступним кроком я почав робити аналіз графу. Спочатку я зробив максимально базовий аналіз.

```
def basic_analyse(g):
    print("Кількість вершин:", g.vcount())
    print("Кількість ребер:", g.ecount())
    print("Щільність графу:", 2 * g.ecount() / (g.vcount() * (g.vcount() - 1)))
```

Рис 1.7 Функція базавого аналізу

Я повертаю кількість вершин та ребер. Також рахую щільність графу.

Щільність графу - це наближення кількості ребер графу, до повного графу.

Тобто 1 - граф повний, 0 - граф пустий.

Далі я проводжу аналіз степенів графу.

```
def degree_analyse(g, n_vertices):
    degrees = []
    total = 0

    for n in range(n_vertices):
        neighbours = g.neighbors(n, mode='ALL')
        total += len(neighbours)
        degrees.append(len(neighbours))

    print("Середій степінь:", total / n_vertices)
    print("Максимальний степінь:", max(degrees))
    print("ID вершини з максимальним степенем:", degrees.index(max(degrees)))
    count_degree_analyse(g, degrees)
```

Рис 1.8 Функція для аналізу степенів вершин графу

Для кожної вершини знаходжу суміжні з нею вершини - це й є степінь вершини. Також я рахую середній та максимальний степінь, а також знаходжу індекс вершини з максимальним ступенем.

Далі я проводжу більш детальний аналіз ступенів вершин.

```
def count_degree_analyse(g, degrees):
    plt.rcParams.update({'font.size': 25})

    x = [x for x in range(max(degrees)+1)]
    degree_counts = [0 for x in range(max(degrees)+1)]

    for i in degrees:
        degree_counts[i] += 1

    print("Найчастіша степінь:", max(degree_counts))

    plt.figure(figsize=(40,10))
    plt.plot(x, degree_counts, linewidth=3.0);
    plt.ylabel('Кількість вершин з заданим степенем')
    plt.xlabel('Степінь')
    plt.title('Опись степенів графу')

    plt.xlim(0,max(degrees)+1)
    plt.xticks(np.arange(min(x), max(x)+1, 2.0))
    plt.grid(True)
    plt.savefig('degree_distribution.png', bbox_inches='tight');
    plt.show();
    plt.draw();
```

Рис 1.9 Функція для більш детального аналізу степенів вершин

Я рахую найчастіший степені вершин та також будую графік кількості степенів.

Рис 1.10 Приклад графіку степенів вершин

Також я рахую кількість трикутників у графі.

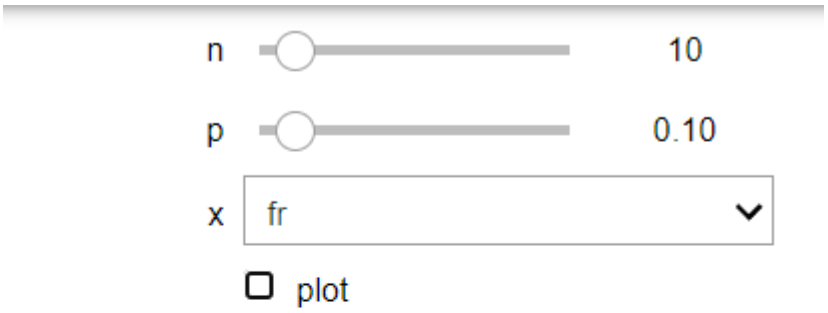
```
def triangle_analyse(g):
    cliques = g.cliques(min=3, max=3)
    triangle_count = [0] * g.vcount()
    for i, j, k in cliques:
        triangle_count[i] += 1
        triangle_count[j] += 1
        triangle_count[k] += 1

    print("Середня кількість трикутників:", sum(triangle_count) / g.vcount())
    print("Максимальна кількість трикутників:", max(triangle_count))
    print("ID вершини з максимальною кількістю трикутників:", triangle_count.index(max(triangle_count)))
```

Рис 1.11 Функція для підрахунку “трикутників” в графі

Я знаходжу середню та максимальну кількість трикутників, а також знаходжу індекс вершин з максимальною кількістю.

Також для зручності я зробив віджети за допомогою яких задаю початкові данні.

Остаточно це виглядає так:




```
Кількість вершин: 10
Кількість ребер: 6
Щільність графу: 0.13333333333333333
Середій степінь: 1.2
Максимальний степінь: 2
ID вершини з максимальним степенем: 0
Найчастіша степінь: 4
```

```
Середня кількість трикутників: 0.0
Максимальна кількість трикутників: 0
ID вершини з максимальною кількістю трикутників: 0
Діаметр графу: 5
Ассортативність графу: 0.2499999999999995
```

Рис 1.12 Фінальний вигляд аналізу

ВИСНОВКИ

Модель Ердеша-Реньї є потужним інструментом для генерації випадкових графів зі стохастичними властивостями. Ця модель знайшла широке застосування в різних галузях, включаючи соціальні мережі, транспортні системи, біологію та інші області.

Візуалізація графів, згенерованих за допомогою моделі Ердеша-Реньї, є важливим кроком у розумінні та аналізі структури цих графів. Вона дозволяє представити складні взаємозв'язки та властивості

графів у вигляді зрозумілих візуальних образів, що сприяє кращому їх сприйняттю та інтерпретації.

Аналіз графів, згенерованих за допомогою моделі Ердеша-Реньї, дозволяє виявляти ключові характеристики цих графів, такі як глобальні та локальні метрики, спільноти, підграфи та інші структурні взаємозв'язки. Це відкриває можливості для дослідження властивостей графів і виявлення важливих особливостей та закономірностей.

Візуалізація та аналіз графів, згенерованих за допомогою моделі Ердеша-Реньї, мають практичне застосування в різних областях. Наприклад, вони можуть допомогти в розумінні соціальних мереж, транспортних систем, біологічних мереж та інших складних систем. Це дозволяє виявляти нові знання, приймати кращі рішення та планувати оптимальні стратегії у відповідних галузях.

Під час дослідження поняття теорії графів було з'ясовано, що граф є абстрактною математичною структурою, яка складається з вершин і зв'язків між ними. Теорія графів вивчає властивості та взаємозв'язки графів і має широкі застосування в різних галузях, включаючи комп'ютерні науки, транспортні мережі, соціальні мережі та інші.

Було встановлено зв'язок між укладками графу та його візуалізацією. Укладки визначають спосіб розташування вершин та зв'язків графу на площині, тоді як візуалізація графу передає цю структуру у вигляді графічного зображення. Правильний вибір укладки може полегшити сприйняття та аналіз графу.

Реалізовано програмний застосунок, який використовує отримані знання про теорію графів, укладки та модель Ердеша-Реньї для створення і аналізу графів. Такий застосунок має різні функції, генерацію випадкових графів за моделлю Ердеша-Реньї, візуалізацію графів тв виконання

алгоритмів на графах, що допомагає вивчати їх властивості та розв'язувати різні завдання.

СПИСОК ЛІТЕРАТУРИ

1. Kamada T., Kawai S. An algorithm for drawing general undirected graphs. Information Processing Letters
2. Handbook of graph drawing and visualization / ed. by R. Tamassia
3. А. М. Райгородський Моделі випадкових графів
4. Fruchterman T. M. J., Reingold E. M., Graph Drawing by Force-Directed Placement. Software -Practice and Experience
5. Спекторський І. Я. Дискретна математика