Ізоморфізм графів. Ейлерів та Гамільтонів цикли у графі. Зважені графи й алгоритми пошуку найкоротших шляхів. Обхід графів. Планарні графи.

Ізоморфізм графів

В теорії графів, ізоморфізмом графів Gі H– це бієкція між множинами вершин Gі H. Як і у застосуваннях теорії графів так у цій теорії істотним є те, що існують об'єкти (вершини графа) і зв'язки графа між об'єктами (ребрами).

Нехай $G_1 = (V_1, E_1)$та $G_2 = (V_2, E_2)$- прості графи, а $\varphi: V_1 \to V_2$- бієкція. Якщо для будь-яких вершин uта vграфа G_1їх образи $\varphi(u)$та $\varphi(v)$суміжні в G_2тоді і лише тоді коли uта vсуміжні в G_1, цю бієкцію називають ізоморфізмом графа G_1на граф G_2. Ізоморфізми записуються як $G_1 \simeq G_2$.

Графи одержують один з одного зі зміною позначень вершин, тому їх доцільно не розрізняти. Проте у деяких ситуаціях розрізняють ізоморфні графи, і тоді можна скористатись поняттям “позначеного графа”. Граф з n вершинами називають *позначеним*, якщо кожній з його вершин присвоєно мітку, наприклад, числа 1,2,...,n. Рівність графів G_1та G_2 позначаємо $G_1 = G_2$ тоді і лише тоді, коли $E_1 = E_2$.

Щоб наголосити, що графи розрізняють лише з точністю до ізоморфізму, говорять про *абстрактний граф*. Абстрактний граф приводить до різних матриць суміжності залежно від нумерації вершин. З’ясуємо, як пов’язані між собою ці матриці. НехайG_1та G_2 - позначені графи з nвершинами й $G_1 \simeq G_2$(тобтоG_1 таG_2 ізоморфні). Це означає, що G_1та G_2розрізняються лише нумерацією вершин, тобто існує підстановка на множині вершин, яка зберігає суміжність: вершини uта vтоді й лише тоді суміжні в G_1коли їхні образи $s(u)$та $s(v)$суміжні в G_2. Нехай А=[a_{ij}] та Б=[b_{ij}] — матриці суміжності відповідно графівG_1 та G_2Тоді = a_{ij} (i,j =1,2,...,n).

Задача виявлення ізоморфізму дуже складна. Теоретично алгоритм перевірки пари простих графів на ізоморфізм існує, проте він не знаходить практичного застосування, оскільки може бути потрібно до $n!$ перестановок і перевірок.

Часто неважко довести, що два графи не ізоморфні. Це буде, якщо порушуються інваріанти. *Інваріант* - це властивість, яку мають обидва ізоморфні графи.

Такими інваріантами, наприклад, є:

- кількість вершин;
- кількість ребер;
- кількість вершин конкретного степеня (вершині $v, v \in V_1, deg(v) = d$,має відповідати вершина $u = \varphi(v), u \in V_2, deg(u) = d$).

Є й інші інваріанти, але порушення інваріанта - це лише достатня умова неізоморфності графів. Не існує набору інваріантів для виявлення ізоморфізму.

Ейлерів цикл у графі

Початок теорії графів як розділу математики пов’язують із задачею про кенігсберзькі мости. Сім мостів міста Кенігсберга (нині - Калінінград у Росії) було розміщено на річці

П регель так, як зображено на рис. 2 Чи можна, починаючи з якоїсь точки міста,пройти через усі мости точно по одному разу й повернутись у початкову точку? Швейцарський математик Л. Ейлер розв'язав цю задачу. Його розв'язання, опубліковане 1736 р., було першим явним застосуванням теорії графів. Ейлер поставив у відповідність плану міста мультиграф *G*, вершини якого відповідають чотирьом частинам *A, B, C, D* міста, а ребра - мостам. Цей мультиграф зображено на рис. 3.

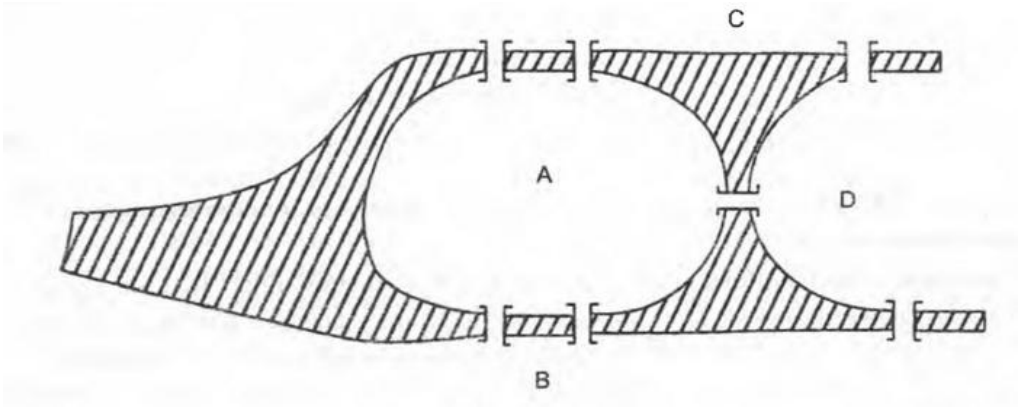

Рис.2

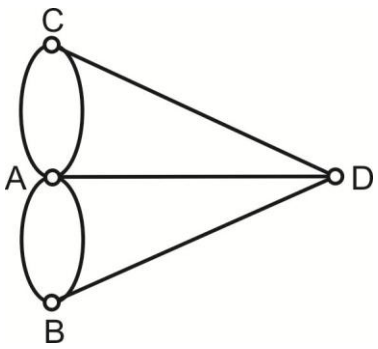

Рис.3

Отже, задачу про кенігсберзькі мости мовою теорії графів можна сформулювати так: чи існує в мультиграфі *G* простий цикл, який містить, усі ребра цього мультиграфа?
Ейлер довів нерозв'язність задачі про кенігсберзькі мости. Нагадаємо, що в простому циклі ребра не повторюються, а вершини можуть повторюватись.

Ейлеровим циклом у зв'язному мультиграфі *G* називають простий цикл, який містить усі ребра графа. *Ейлеровим шляхом* у зв'язному мультиграфі *G* називають простий шлях, який містить усі ребра графа.

Існує простий критерій (необхідна й достатня умова) для виявлення, чи має граф ейлерів цикл.

Алгоритм Флері побудови ейлерового циклу

Робота полягає в нумерації ребер у процесі побудови ейлерового циклу.

Крок 1 (початок). Починаємо з довільної вершини и та присвоюємо довільному ребру $\{u, v\}$ номер 1. Викреслюємо ребро $\{u, v\}$ і переходимо v вершину v.

Крок 2 (ітерація). Нехай *w*- вершина, у яку ми перейшли на попередній ітерації, к-останній присвоєний номер. Вибираємо довільне ребро, інцидентне вершині *w*, причому міст

вибираємо лише тоді, коли немає інших можливостей. Присвоюємо вибраному ребру номер $(k+1)$і викреслюємо його.

Крок 3 (закінчення). Цей процес закінчуємо, коли всі ребра графа викреслено та пронумеровано - ці номери задають послідовність ребер в ейлеровому циклі.

Будь-який ейлерів шлях починається в одній із цих двох вершин непарного степеня, а закінчується в іншій.

Оскільки мультиграф для кенігсберзьких мостів має чотири вершини з непарними степенями, можна дійти висновку про неможливість пройти кожний міст по одному разу, навіть якщо не потрібно повертатись у початкову точку.

Граф, який має ейлерів цикл, часто називають *ейлеровим графом*.

Ейлеровим циклом у слабко зв'язному орієнтованому мультиграфі називають орієнтований простий цикл, який містить усі дуги графа.

Гамільтонів цикл у графі

Шлях$x_0, x_1, \ldots, x_{n-1}, x_n$ у зв'язному графі $G = (V, E)$називають *гамільтоновим шляхом,* якщо $V = \{x_0, x_1, \ldots, x_{n-1}, x_n\}$і$x_i \neq x_j$ для$0 \leqslant i < j \leqslant n$. Цикл $x_0, x_1, \ldots, x_{n-1}, x_n, x_0$ (тут $n > 1$) у графі $G = (V, E)$називають *гамільтоновим циклом*, якщо $x_0, x_1, \ldots, x_{n-1}, x_n$- гамільтонів шлях. Інакше кажучи, гамільтонів цикл і гамільтонів шлях проходять через кожну вершину графа точно один раз (можливо, окрім першої й останньої вершини). Зазначимо, що гамільтонів цикл і шлях, узагалі кажучи, не містять усіх ребер графа.

Термін "гамільтонів" у цих означеннях походить від імені відомого ірландського математика У. Гамільтона (W. Hamilton), який у 1857 р. Запропонував гру "Навколосвітня подорож". Кожній із дванадцяти вершин додекаедра (правильного дванадцятигранника, грані якого — п'ятикутники) приписано назву одного з великих міст світу. Подібно, розпочавши з довільного міста, відвідати решту 19 міст точно один раз і повернутись у початкове місто. Перехід дозволено ребрами додекаедра.

Ту саму задачу можна зобразити й на площині (рис. 4). Вона зводиться до відшукання в графі гамільтонового циклу. Один із можливих розв'язків показано синіми потовщеними лініями.

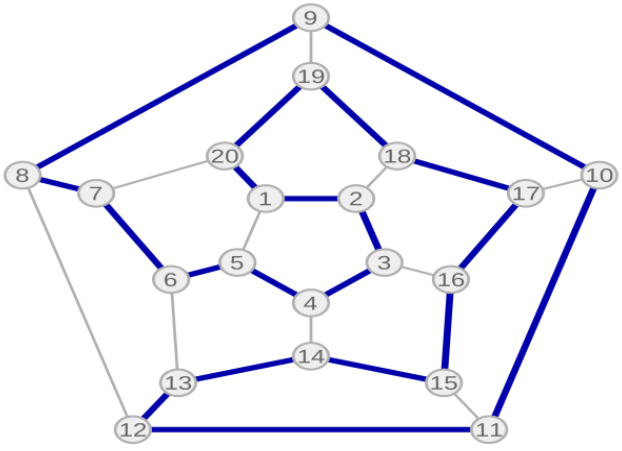

Рис. 4

Не всі зв'язані графи мають гамільтонів цикл хоча б тому, що такий граф має бути

двозв'язним (тобто граф, який має точки з'єднання, не може мати гамільтонового циклу). Приклад графа, зображеного на рис.5, свідчить, що двозв'язності недостатньо для наявності гамільтонового циклу. Незважаючи на зовнішнісню подібність формулювань задач про існування ейлерового й гамільтонового циклів, ці задачі принципово різні. Використовуючи результати попереднього підрозділу, легко виявити, чи має цей граф ейлерів цикл, і, якщо має, то побудувати його.

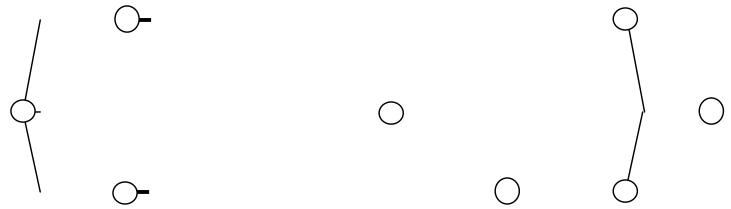

Рис. 5

Ситуація для гамільтонового циклу істотно інша. Відповісти на питання, чи має граф гамільтонів цикл, зазвичай, дуже важко. Вивчення достатніх умов наявності в графі гамільтонового циклу — один із важливих напрямім у теорії графів. Інтуїтивно зрозуміло, що граф із багатьма ребрами, достатньо рівномірно росподілений, з великою ймовірності має гамільтонів цикл. Доведемо одну з теорем такого типу.

Як знайти гамільтонів цикл або переконатись, що його немає? Очевидний алгоритм, який можна застосувати, - це повний перебір усіх можливостей, тобто $n!$перестановок усіх вершин графа й перевірок. Зрозуміло, що такий спосіб не практичний. Більш практичним є спосіб алгоритму бектрекінга(пошук з поверненнями).

Граф, який містить гамільтонів цикл, часто називають *гамільтоновим графом*.

Також існують інші усови існування гамільтонового графу окрім умови Дірака, зазначеної вище, такі як:

Умова Оре(1960р.). Якщо р— кількість вершин в даному графі. Якщо для будь-якої пари несуміжних вершин *x, y* виконано нерівність $d(x) + d(y) \geqslant p$то граф називаєтся *графом Оре* (тобто сума степеней будь-яких двох несуміжних вершин не менша загального числа вершин в графі). Граф Оре — гамільтонів.

Умова Бонді-Хватала. Теорема Бонді-Хватала узагальнює твердження Дірака і Оре. Спочатку визначимо замикання графа. Нехай у графа Gє p вершин. Тоді замикання $cl(G)$однозначно будується по Gдодаванням для всіх несуміжних вершин xі y, у яких виконується $d(x) + d(y) \geqslant p$нового ребра uv. Граф є гамільтоновим тоді і тільки тоді, коли його замикання – гамільтонів граф.

Також варто зазначити, що:

- Будь-який повний граф є гамільтоновим.
- Усі цикли є гамільтоновими графами.
- Усі правильні багатогранники є гамільтоновими графами.

Зважені графи й алгоритми пошуку найкоротших шляхів

У реальних задачах на графах часто потрібно брати до уваги додаткову інформацію — фактичну віддаль між окремими пунктами, вартість проїзду, час проїзду тощо. Для цього використовують поняття зваженого графа.

Зваженим називають простий граф, кожному ребру є якого приписано дійсне число ω(e). Це

число називають вагою ребра є. Аналогічно означають зважений орієнтований граф: це такий орієнтований граф, кожній дузі є якого приписано дійсне число ω(e), називане вагою дуги.

Розглянемо три способи зберігання зваженого графа G = (V, E) в пам'яті комп'ютера.

Нехай |V| = n, |E| = m.

Перший — подання графа матрицею ваг W, яка являє собою аналог матриці суміжності, її елемент $w_y = w(v_v, v_j)$, якщо ребро $(v_v, v_j) \in$ E (у разі орієнтованого графа — дуга $(v_v, v_j) \in$ E). Якщо ж ребро $(v_v, v_j) \notin$E(або дуга $(v_v, v_j) \notin$E), то $w_y = 0$ чи $w_y = \infty$ залежно від розв'язуваної задачі.

Другий спосіб — подання графа списком ребер. Для зваженого графа під кожний елемент списку E можна відвести три комірки — дві для ребра й одну для його ваги, тобто всього потрібно 3m комірок. Третій спосіб — подання графа списками суміжності. Для зваженого графа кожний список Adj[u] містить окрім покажчиків на всі вершини υ; множини Γ(υ)ще й числа $\omega(u, \upsilon)$.

Довжиною шляху в зваженому графі називають суму ваг ребер (дуг), які утворюють цей шлях. Якщо граф не зважений, то вагу кожного ребра (кожної дуги) уважають рівною 1 й отримують раніше введене поняття довжини шляху як кількості ребер (дуг) у ньому.

Задача про найкоротший шлях полягає в знаходженні найкоротшого шляху від заданої початкової вершини a до заданої вершини z. Наступні дві задачі — безпосередні узагальнення сформульованої задачі про найкоротший шлях.

1. Для заданої початкової вершини a знайти найкоротші шляхи від a до всіх інших вершин.

2. Знайти найкоротші шляхи між усіма парами вершин.

Найефективніший алгоритм визначення довжини найкоротшого шляху від фіксованої вершини до будь-якої іншої запропонував 1959 р. датський математик E. Дейкстра (E. Dijkstra). Цей алгоритм застосовний лише тоді, коли вага кожного ребра (дуги) додатна.

Алгоритм Дейкстри

1. Присвоювання початкових значень. Виконати l(a) = 0 та вважати цю мітку постійною. Виконати $l(v) = \infty$ для всіх $v \neq a$ й вважати ці мітки тимчасовими. Виконати $x = a, M = a.$

2.Оновлення міток. Для кожної вершини $v \in \Gamma(x) \setminus$ M замінити мітки: $l(v) = minl(v), l(x) + w(x, v)$, тобто оновлювати тимчасові мітки вершин, у які з вершини x іде дуга.

3. Перетворення мітки в постійну. Серед усіх вершин із тимчасовими мітками знайти вершину з мінімальною міткою, тобто знайти вершину v* з умови l(v*) = $minl(v),\ v \in T$, де T= V\ M.

4. Вважати мітку вершини v* постійною й виконати M= M ∪ {v*}; x = v* (вершину v* включено в множину M).

5. а) Для пошуку шляху від a до z: якщо $x = z$, то $l(z)$– довжина найкоротшого шляху від a до z, зупинитись; якщо $a \neq z$, то перейти до кроку 2.

б) Для пошуку шляхів від a до всіх вершин: якщо всі вершини отримали постійні мітки (включені в множину M), то ці мітки дорівнюють довжинам найкоротших шляхів, зупинитись; якщо деякі вершини мають тимчасові мітки, то перейти до кроку 2.

Якщо граф подано матрицею суміжності, складність алгоритму Дейкстри становить $O(n^2)$. Коли кількість дуг значно менша ніж n^2, то найкраще подавати орієнтований граф списками суміжності. Тоді алгоритм можна реалізувати зі складністю $O(mlgn)$, що в цьому разі істотно менше ніж $O(n^2)$.

Алгоритм Дейкстри дає змогу обчислити довжину найкоротшого шляху від початкової вершини a до заданої вершини z. Для знаходження самого шляху потрібно лише збільшувати вектор вершин, з яких найкоротший шлях безпосередньо потрапляє в дану вершину.

Алгоритм Флойда-Уоршола

1. Присвоювання початкових значень. Пронумерувати вершини графа G цілими числами від 1 до n. Побудувати матрицю $W^{(n)}$, задавши кожний її (i, j)-й елемент таким, що дорівнює вазі дуги, котра з'єднує вершину i з вершиною j. Якщо в графі G ці вершини не з'єднано дугою, то виконати $\omega_{ij}^{(0)} = \infty$.Крім того, для всіх i виконати $\omega_{ij}^{(0)} = 0$.

2. Цикл по всіх k, що послідовно набуває значення 1, 2, …, n, визначити за елементами матриці $W^{(k-1)}$ елементи матриці $W^{(k)}$, використовуючи рекурентне співвідношення (30.1).

Після закінчення цієї процедури (i, j)-й елемент матриці $W^{(n)}$дорівнює довжині найкоротшого шляху з вершини i у вершину j. Якщо під час роботи алгоритму для якихось k та i виявиться, що $\omega_{ii}^{(k)} < 0$, то в графі G існує цикл із від'ємною довжиною, який містить вершину i. Тоді роботу алгоритму потрібно припинити.

Алгоритм Белмана-Форда

Для розв'язку задачі знаходження найкоротших шляхів у графах з негативними вагами ребер може використовуватись алгоритм Белмана-Форда. Цей алгоритм був запропонований незалежно Річардом Белманом та Лестером Фордом у 1958 та 1956 роках. Він дозволяє знаходити найкоротший шлях від заданої вершини до всіх інших вершин графу у випадку існування в графі ребер з негативними вагами. Єдиним обмеженням застосування цього алгоритму є відсутність негативних циклів у графі. Дійсно, якщо в графі є хоча б один цикл, сумарна вага ребер якого є від'ємною, то це дозволить нам знайти маршрут з якою завгодно малою довжиною між двома вершинами в графі. Алгоритм Белмана-Форда у випадку існування негативних циклів в графі дозволяє визначити цей факт та коректно завершити свою роботу.

Обхід графів.

Існує багато алгоритмів на графах, які грунтуються на систематичному переборі їх вершин або обході вершин, під час якого кожна вершина одержує унікальний порядковий номер. Алгоритми обходу вершин графа називають методами пошуку.

Пошук углиб у простому зв'язному графі

Опишемо метод пошуку в простому зв'язному графі, який являє собою одну з основних методик проектування алгоритмів, пов'язаних із графами. Цей метод називають пошуком у глиб, або DFS-методом (від англ. Depth first search).

Нехай G=(V,E) — простий зв'язний граф, усі вершини якого позначено попарно різними символами. У процесі пошуку вглиб вершинам графа G надають номери (DFS-номери) та певним способом позначають ребра. У ході роботи алгоритму використовують структуру даних для збереження множин, яку називають стеком (англ. stack — стіг). Зі стеку можна вилучити тільки той елемент, котрий було додано до нього останнім: стек працює за принципом „останнім прийшов — першим вийшов" (last in, first out — скорочено LIFO). Інакше кажучи, додавання й вилучення елементів у стеку відбувається з одного кінця, який називають верхівкою стеку. DFS-номер вершини x позначають DFS(x).

Алгоритм пошуку вглиб у простому зв'язному графі

Наведемо кроки алгоритму.

Крок 1. Почати з довільної вершини v_s. Виконати $DFS(v_s) := 1$. Включити цю вершину

в стек.

Крок 2. Розглянути вершину у верхівці стеку: нехай це вершина x . Якщо всі ребра, інцидентні вершині x, позначено, то перейти до кроку 4, інакше — до кроку 3.

Крок 3. Нехай {x, y}– непозначене ребро. Якщо DFS(y) уже визначено, то позначити ребро {x, y}штриховою лінією та перейти до кроку 2. Якщо DFS(y) не визначено, то позначити ребро {x, y}потовщеною суцільною лінією, визначити DFS(y) як черговий DFS-номер, включити цю вершину в стек і перейти до кроку 2.

Крок 4. Виключити вершину xзі стеку. Якщо стек порожній, то зупинитись, інакше — перейти до кроку 2.

Щоб вибір номерів був однозначний, доцільно домовитись, що вершини, суміжні з тією, яка вже отримала DFS-номер, аналізують за зростанням їх порядкових номерів (або в алфавітному порядку). Динаміку роботи алгоритму зручно зображати за допомогою таблиці з трьома стовпцями: вершина, DFS-номер. уміст стеку. Її називають протоколом обходу графа пошуком вглиб.

Пошук вшир у простому зв'язному графі

У процесі пошуку вшир вершини графа проглядають в іншій послідовності, ніж у методі пошуку вглиб, і їм надають BFS-номери (від англ. breadth first search). BFS-номер вершини x позначають BFS(x). Під час пошуку рухаються вшир, а не вглиб: спочатку проглядають усі сусідці вершини, після цього — сусіди сусідів і так далі.

У ході реалізації алгоритму використовують структуру даних для збереження множин, яку називають чергою (англ. queue). З черги можна вилучити тільки той елемент, який перебував у ній найдовше: працює принцип „першим прийшов — першим вийшов" (first in. first out — скорочено FIFO). Елемент включається у хвіст черги, а виключається з її голови. Пошук ушир, узагалі кажучи відрізняється від пошуку вглиб заміною стеку на чергу. Після такої модифікації що раніше відвідується вершина (включається в чергу), то раніше вона використовується (і виключається з черги). Використання вершини полягає в перегляді одразу всіх іще не відвіданих її сусідів. Усю процедуру подано нижче.

Алгоритм пошуку ушир у простому зв'язному графі

Наведемо кроки алгоритму.

Крок 1. Почати з довільної вершини v_s. Виконати BFS $v_s := 1$. Включити вершину v_s у чергу.

Крок 2. Розглянути вершину, яка перебуває на початку черги; нехай це буде вершина x. Якщо для всіх вершин, суміжних із вершиною x, уже визначено BFS-номери, то перейти до кроку 4, інакше — до кроку 3.

Крок 3. Нехай {x;y} — ребро, у якому номер BFS(y) не визначено. Позначити це ребро потовщеною суцільною лінією, визначити BFS(y) як черговий BFS-номер, включити вершину y у чергу й перейти до кроку 2.

Крок 4. Виключити вершину x із черги. Якщо черга порожня, то зупинитись, інакше – перейти до кроку 2.

Щоб результат виконання алгоритму був однозначним, вершини, які суміжні з вершиною x, аналізують за зростанням їх порядкових номерів (або в алфавітному порядку). Динаміку роботи алгоритму пошуку вшир також зручно зображати за допомогою протоколу обходу.

Обчислювальна складність обох алгоритмів обходу однакова й у разі подання графа списками суміжності становить $O(n + m)$, де m— кількість ребер, а n— кількість вершин графа.

Планарні графи.

Планарний граф – це граф, який можна зобразити на площині без перетину ребер (на рис.1 зображено приклад планарного графа). Також кажуть, що граф є планарний тоді, коли він ізоморфний деякому плоскому графу. Поняття планарного графа є тісно повязане з поняттям плоского графа.

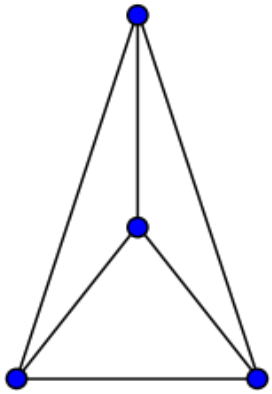

Рис.1

Що таке плоский граф? Розлянемо неорієнтовані графи. В більшості випадків не має значення, як зобразити граф на рисунку, адже ізоморфні графи несуть одну і ту ж інформацію. Але часто є важливо, чи можна подати граф на площині так, щоб його зображення задовольняло певні вимоги. Наприклад, у радіоелектроніці в процесі виготовлення мікросхем, електричні ланцюги наносять на поверхню ізоляційного матералу. Оскільки провідники не ізольовані, то вони не повинні перетинатись. Також при проектуванні доріг або залізничних шляхів виникає проблема перехресть та переїздів, що є небажаним. Так виникає поняття “плоский граф”.

Плоский граф – граф, який зображений на площині так, що ніякі два його ребра геометрично не перетинаються ніде, окрім інцидентних їм вершин.

Області, на які плоский граф розбиває площину є його гранями. Одна з граней плоского графа є необмежена. Така грань називається зовнішньою гранню.

Критерії непланарності графа:

- достатня умова: якщо граф містить дводольний підграф $K_{3,3}$або повний підграф K_5— то він є непланарний.
- необхідна умова: якщо граф не планарний, то він повинен містити більше 4 вершин,степінь яких більше 3, або більше 5 вершин степеня 2.

Два графи називаються гомеоморфними, якщо їх можна отримати з одного графа додаванням до його ребер нових вершин степеня 2.

Властивості планарних графів:

- довільний планарний граф може бути зображений на площині так, щоб усі його ребра були прямими відрізками
- вершини довільного планарного графа можна розфарбувати в чотири кольори так, щоб усі суміжні вершини мали різні кольори.

Планарний граф називають зовнішнім планарним, якщо існує грань, яка містить всі його вершини. Такі графи можна зобразити на площині так, що усі вершини будуть розміщені на колі. Усі дерева є планарними зовнішніми планарними графами.

Властивості зовнішніх планарних графів:

- довільний зовнішній планарний граф маж вершину щонайменше другого

степеня.

- вершини довільного зовнішнього планарного графа можна розфарбувати в три кольори так, щоб усі суміжні вершини були різного кольору.

Максимальним планарним графом називається планарний граф, який при додаванні до нього будь-якого ребра перестає бути планарним. Плоский зв'язний граф, кожна грань якого обмежена трикутником, називається тріангуляцією.

Для довільного $n \geq 3$ існує принаймні одна тріангуляція з n вершинами.

Кожен плоский граф, який має не менше 3 вершин і не є тріангуляцією, можна доповнити ребрами до тріангуляції.

Серед всіх плоских графів з n вершинами найбільшу кількість граней має тріангуляція.