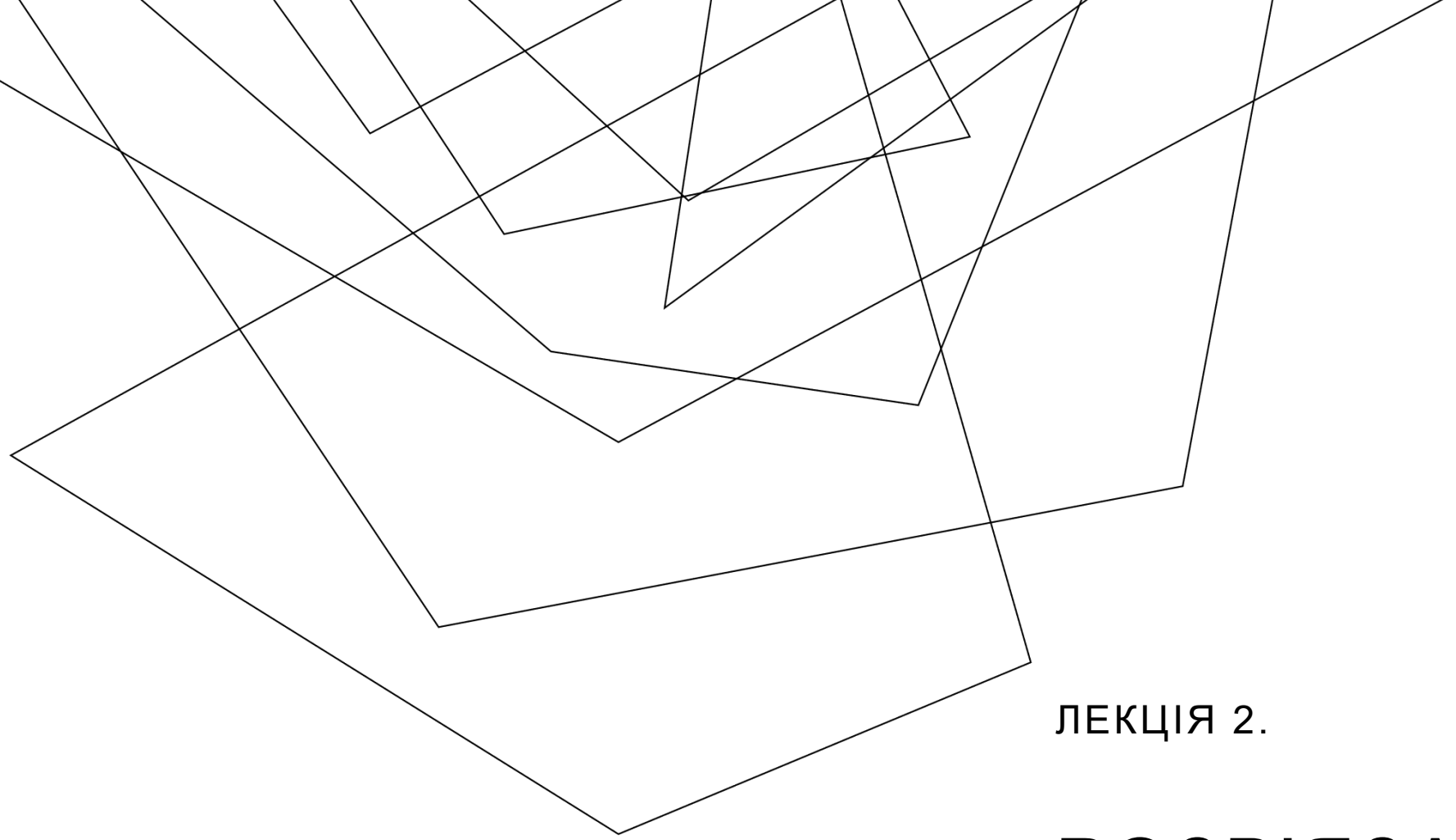
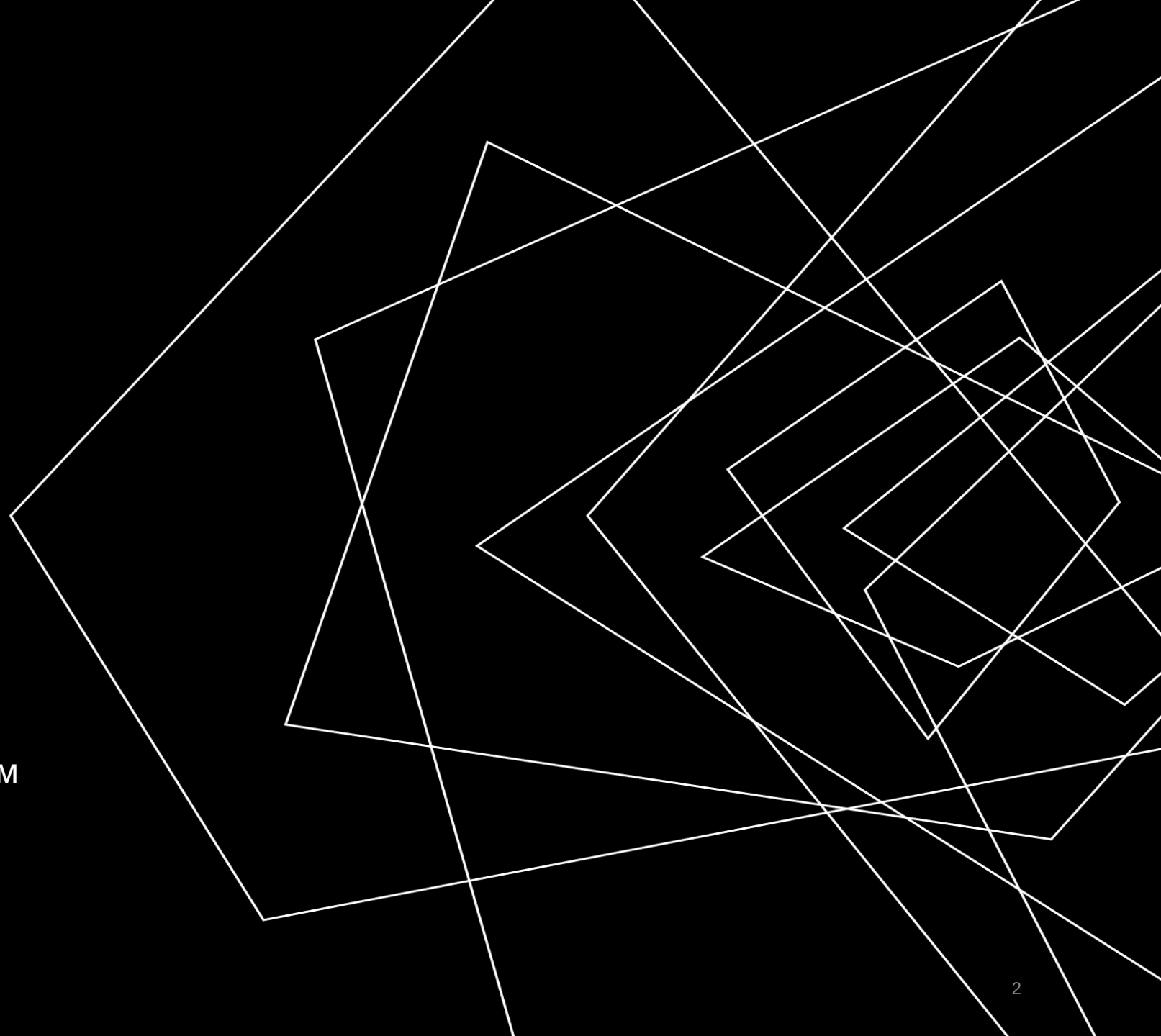


ЛЕКЦІЯ 2.

РОЗВ'ЯЗАННЯ ЗАДАЧ НА ГРАФАХ

ПЛАН

1. Ейлерів та гамільтонів цикли
2. Алгоритми пошуку найкоротших шляхів у зважених графах
3. Планарність та ізоморфізм



ЕЙЛЕРІВ ТА ГАМІЛЬТОНІВ ЦИКЛИ

МАРШРУТИ, ЛАНЦЮГИ ТА ЦИКЛИ

Послідовність вершин та ребер $<v_1,e_1,v_2,e_2..,v_i,..,e_{k-1},v_k>$ (не обов'язково різних) таких, що $e_i=\{v_i,v_{i+1}\} \in E$ для усіх $i=1,2,\ldots,k-1$ називається **маршрутом *довжиною k-1*.** Вершина v_1 – початок маршруту, v_k – його кінець.

Маршрут називається **ланцюгом**, якщо усі його ребра різні та **простим ланцюгом**, якщо усі вершини різні, крім, можливо, початкової та кінцевої вершин.

Якщо початкова і кінцева вершини співпадають, то такий маршрут називається **циклічним**. Циклічний маршрут, який одночасно є ланцюгом, називається **циклом**.

Якщо циклічний маршрут є простим ланцюгом, то його називають **простим циклом**.

ЕЙЛЕРІВ ЦИКЛ

Ейлерів цикл – це шлях у графі, який проходитиме через кожне ребро рівно один раз і повертається у початкову вершину.

Умова існування: Граф має цикл тоді і тільки тоді, коли всі його вершини мають парний ступінь.

Цикл в графі називають **Ейлеровим**, якщо він містить усі ребра графу. Зв’язний граф, в якому існує Ейлеровий цикл називають **Ейлеровим графом**.

ЕЙЛЕРІВ ЦИКЛ

Необхідна і достатня умова існування ейлерового циклу:

- Граф повинен бути зв'язним.
- Всі вершини графа повинні мати парний степінь. Степінь вершини - це кількість ребер, що інцидентні до цієї вершини.

Приклад:

У квадраті кожна вершина має ступінь 2, тому в ньому існує ейлерів цикл.

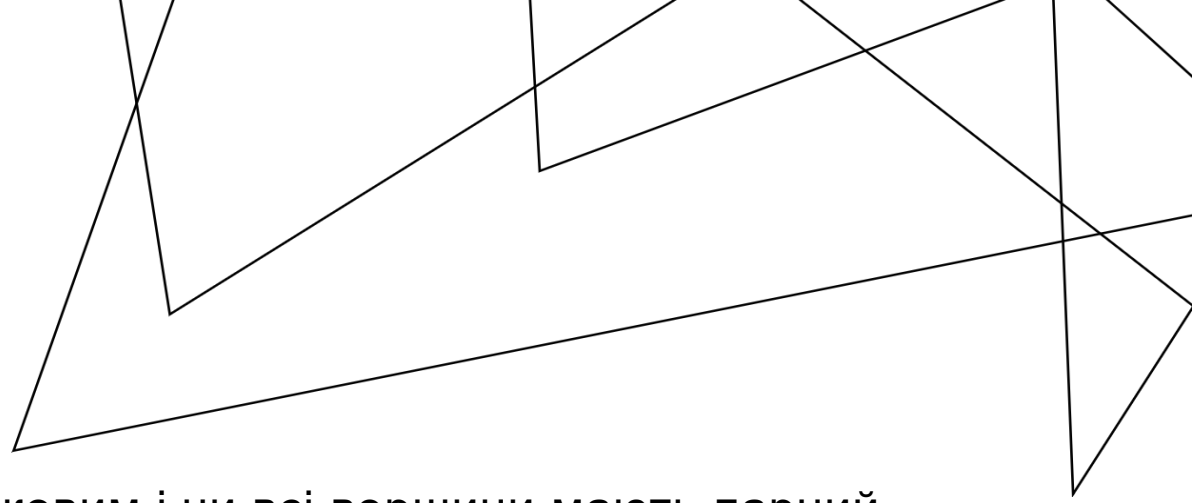
У кубі також кожна вершина має парний ступінь, отже, існує ейлерів цикл.

ЕЙЛЕРІВ ЦИКЛ

Теорема (Ейлер). Скінчений граф є ейлеровим тоді і тільки тоді, коли він зв’язний та степені всіх його вершин парні.

Для побудови ейлерового циклу існує алгоритм Флері. Алгоритм нумерує ребра графу таким чином, що номер ребра вказує, яким по порядку є це ребро у ейлеровому циклі.

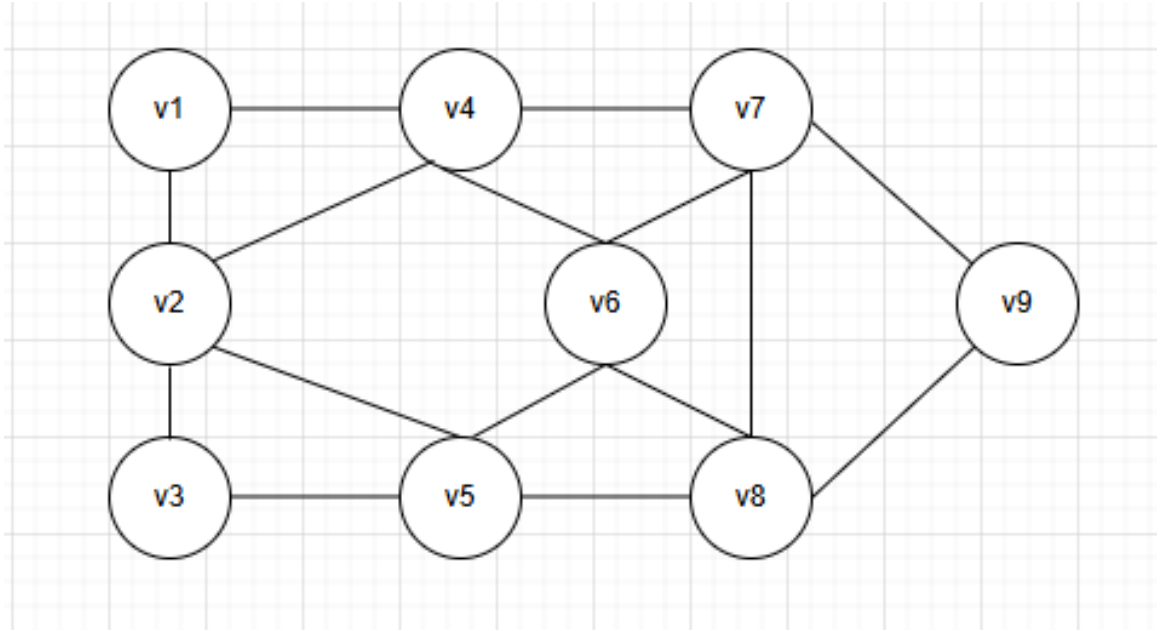
АЛГОРИТМ ПОШУКУ ЕЙЛЕРОВОГО ЦИКЛУ (АЛГОРИТМ ФЛЕРІ)



1. **Перевірка умов.** Перевіряємо, чи є граф зв'язковим і чи всі вершини мають парний ступінь. Якщо умови не виконуються, цикл Ейлера не існує.
2. **Вибір початкової вершини.** Вибираємо будь-яку вершину графа як початкову.
3. **Побудова циклу:**
   - Поки що є невідзначені ребра:
     - Вибираємо будь-яке ребро, інцидентне поточній вершині, так, щоб воно не було мостом (міст – це ребро, видалення якого робить граф незв'язним).
     - Зазначаємо обране ребро як пройдене.
     - Переходимо до вершини, куди веде обране ребро.
4. **Перевірка результату.** Якщо всі ребра були відзначені, то отриманий шлях є циклом Ейлера.

ПРИКЛАД

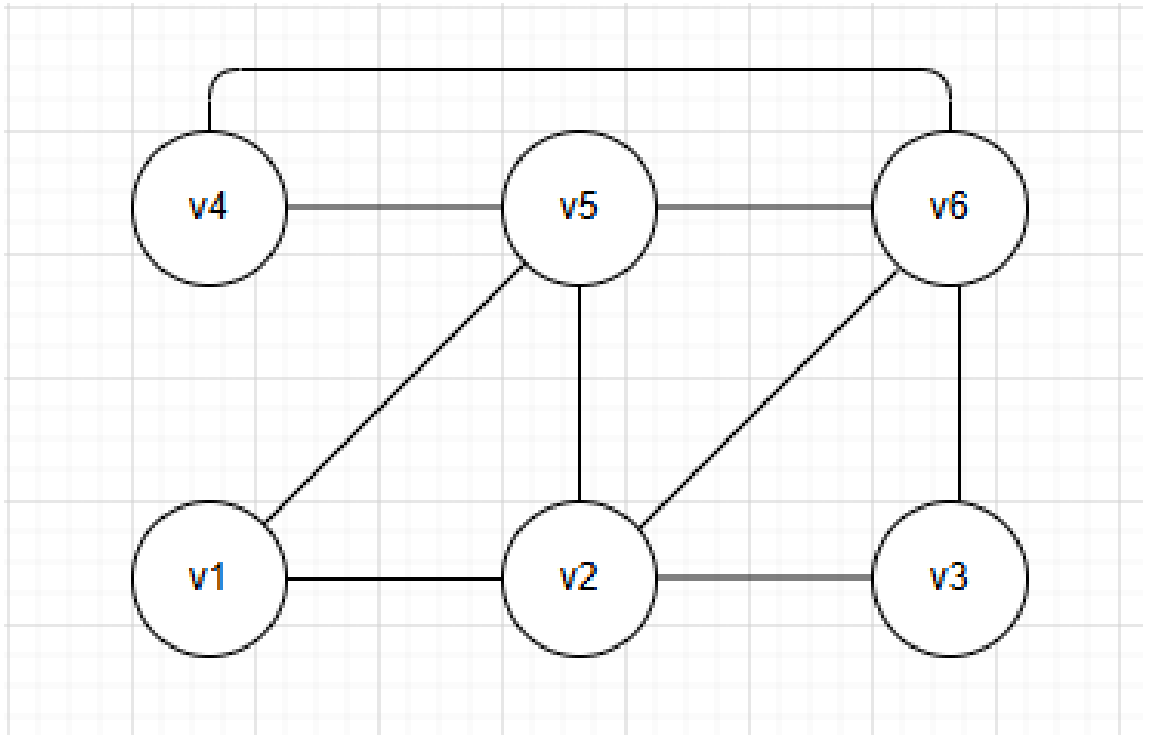
Ейлерів граф та ейлерів цикл у цьому графі , побудований алгоритмом Флері



v1 – v2 – v3 – v5 – v2 – v4 – v6 – v5 – v8 – v6 – v7 – v8 – v9 – v7 – v4 – v1

ПРИКЛАД

Ейлерів граф та ейлерів цикл у цьому графі



Для даного графа результатом виконання алгоритму буде:
v1 > v5 > v2 > v6 > v5 > v4 > v6 > v3 > v2 > v1.

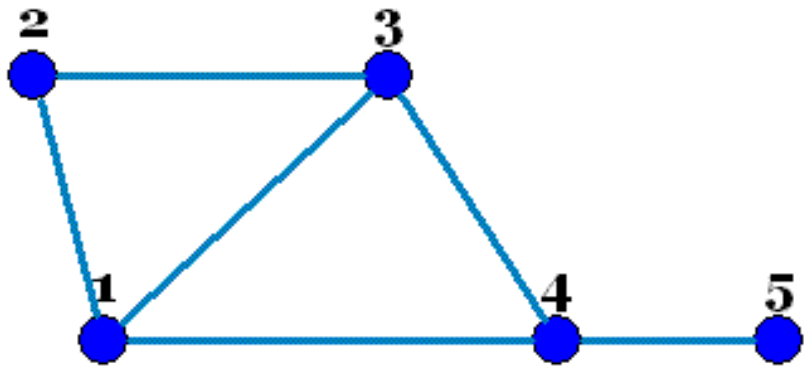
ГАМІЛЬТОНІВ ЦИКЛ

Гамільтоновим циклом називають простий цикл, який містить всі вершини даного графу. Граф, у якому існує такий цикл - **Гамільтонів.**

Теорема (Дірак). Граф без петель з кількістю вершин не менше, ніж 3 є Гамільтоновим, якщо для будь-якої вершини виконується нерівність: , де *n* - кількість вершин, *k(v)* - степінь вершини.

ПРИКЛАД

Задано граф



Він містить декілька гамільтонових шляхів, що починаються у вершині

1: 1 – 2 – 3 – 4 – 5,

2: 2 – 1 – 3 – 4 – 5, 2 – 3 – 1 – 4 – 5,

3: 3 – 2 – 1 – 4 – 5,

Оскільки існує гамільтонів шлях 3 – 2 – 1 – 4 – 5, то шлях, записаний у зворотному порядку вершин, буде також гамільтоновим. Тому маємо ще 4 шляхи Гамільтона, що починаються у вершині **5:**

5 – 4 – 1 – 2 – 3, **5 – 4 – 1 – 3 – 2**,

5 – 4 – 3 – 1 – 2, **5 – 4 – 3 – 2 – 1**

Відмінності між ейлеровим та гамільтоновим циклами

Характеристика	Ейлерів цикл	Гамільтонів цикл
Що оминати	Усі ребра	Усі вершини
Умова існування	Усі вершини мають парний степінь	Немає критерію
Приклади графів	Квадрат, куб	Додекаедр, повний граф

АЛГОРИТМИ ПОШУКУ НАЙКОРОТШИХ ШЛЯХІВ У ЗВАЖЕНИХ ГРАФАХ

АЛГОРИТМИ ПОШУКУ НАЙКОРОТШИХ ШЛЯХІВ У ЗВАЖЕНИХ ГРАФАХ

Алгоритм Дейкстри

Призначення. Знаходить найкоротший шлях від однієї вершини до всіх інших у графі з невід'ємними вагами ребер.

Принцип роботи. Починаючи зі стартової вершини, алгоритм ітеративно розширює область пошуку, завжди вибираючи непомічену вершину з найменшою сумарною вагою шляху до неї.

Обмеження. Не працює з графами, що містять ребра з від'ємними вагами

Алгоритм Беллмана-Форда

Призначення. Знаходить найкоротший шлях від однієї вершини до всіх інших у графі, який може містити ребра з від'ємними вагами.

Принцип роботи. Алгоритм виконує обхід ребер графа кілька разів, поступово покращуючи оцінки відстаней до всіх вершин.

Виявлення від'ємних циклів. Може виявити наявність від'ємних циклів у графі.

АЛГОРИТМИ ПОШУКУ НАЙКОРОТШИХ ШЛЯХІВ У ЗВАЖЕНИХ ГРАФАХ

Алгоритм Флойда-Уоршолла

Призначення. Знаходить найкоротші шляхи між усіма парами вершин у зваженому графі.

Принцип роботи. Алгоритм використовує динамічне програмування для обчислення відстаней.

Обмеження. Може бути неефективним для великих графів

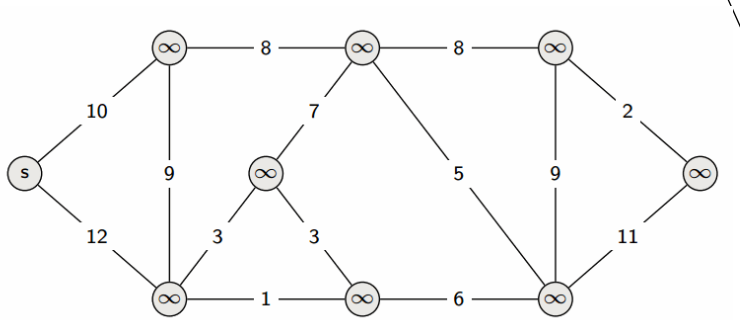
Алгоритм *A**

Призначення. Евристичний алгоритм пошуку, призначений для знаходження найкоротшого шляху в графі з використанням евристичної функції.

Принцип роботи. Алгоритм комбінує інформацію про точну відстань від стартової вершини до поточного вузла та оцінку відстані до цільової вершини.

Обмеження. Зазвичай швидше за алгоритм Дейкстри, особливо для великих графів, але не гарантує знаходження оптимального рішення.

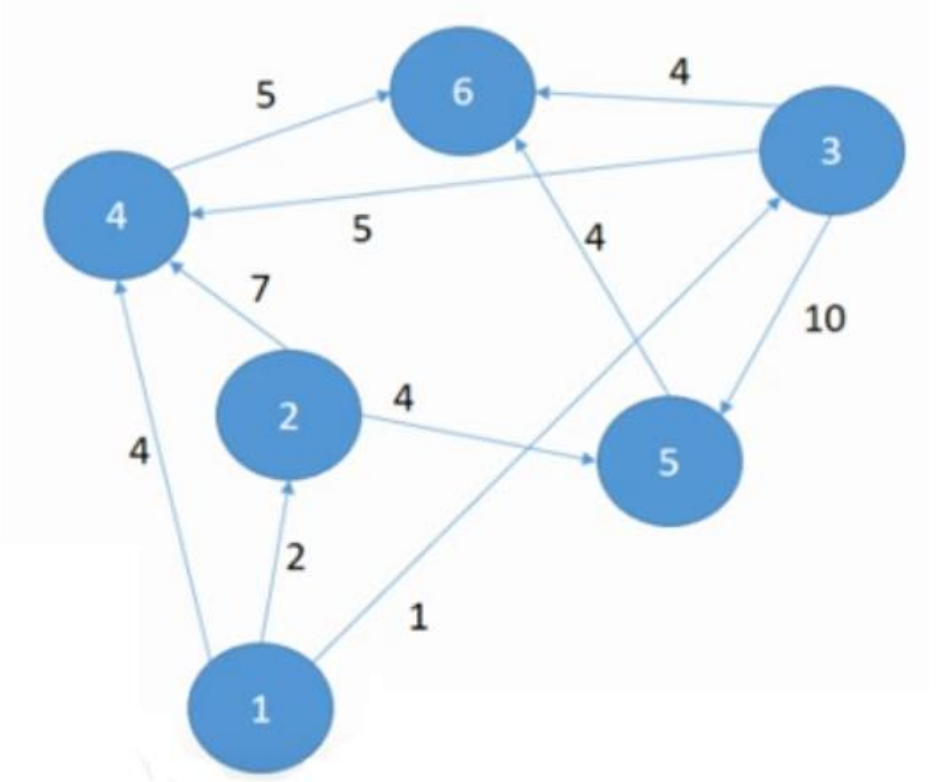
АЛГОРИТМ ДЕЙКСТРИ

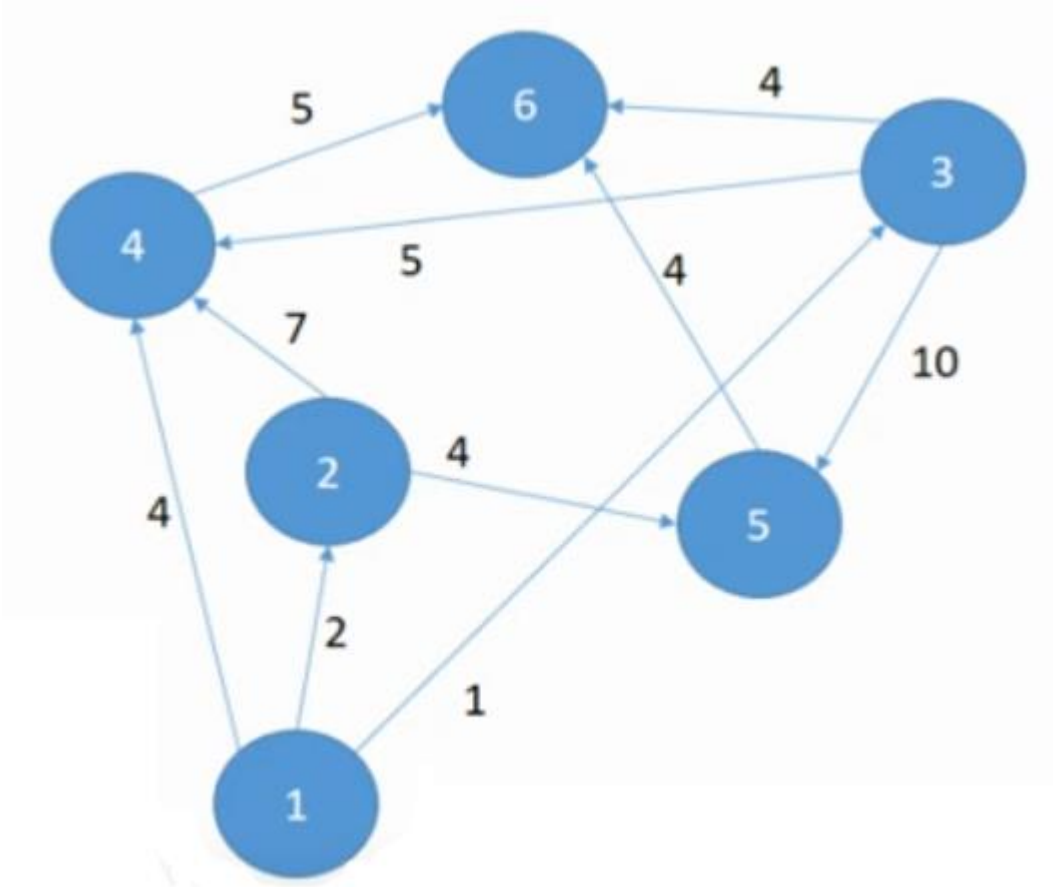


Алгоритм Дейкстри - це потужний інструмент в теорії графів, який дозволяє знаходити найкоротші шляхи від однієї вибраної вершини до всіх інших в зваженому графі. Це означає, що кожне ребро в графі має певну вагу (наприклад, відстань, вартість, час), і алгоритм шукає шлях, сумарна вага якого буде найменшою. Але одне обмеження накладається: *довжини ребер не мають бути невідємними*. Цей алгоритм може бути застосований як в орграфі так і н-графі.

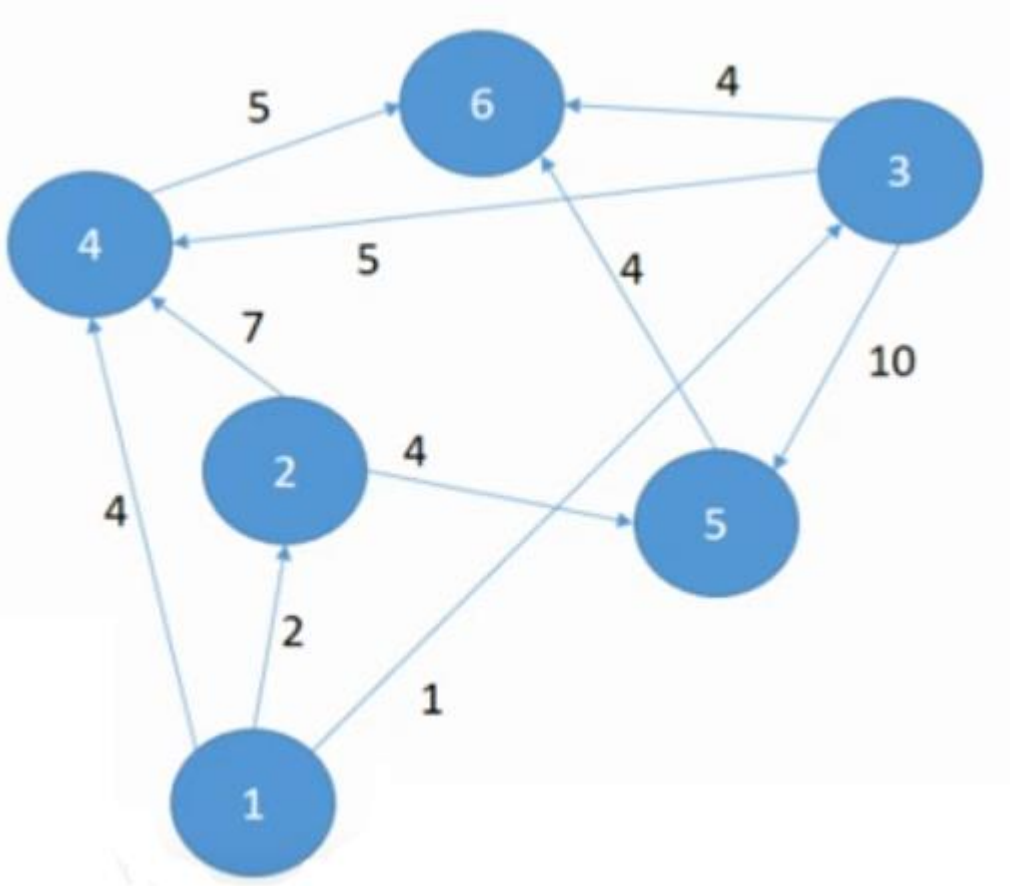
ПРИКЛАД

Знайти найкоротший шлях від вершини 1 до усіх інших

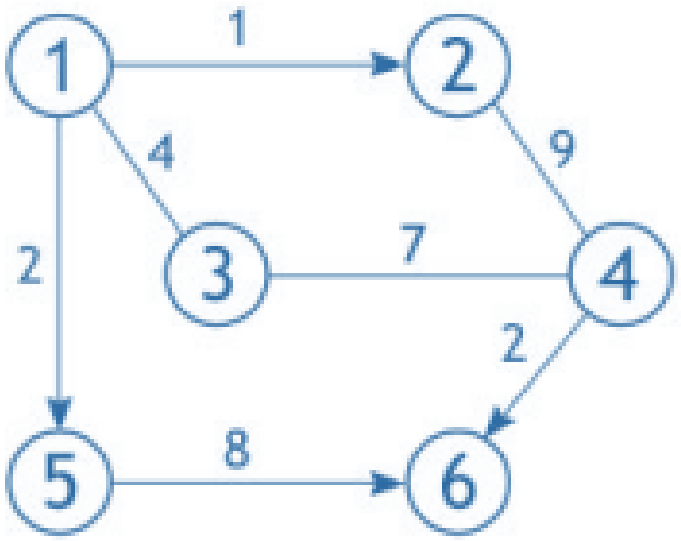




	1	2	3	4	5	6



	1	2	3	4	5	6
	0	∞	∞	∞	∞	∞
1		2	1	4	∞	∞
3		2		4	11	5
2				4	6	5
4					6	5
6					6	
	0	2	1	4	6	5



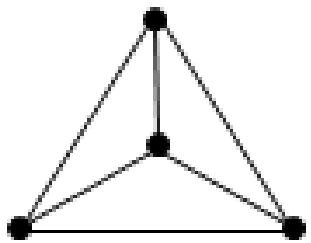
	1	2	3	4	5	6

ПЛАНАРНІСТЬ ТА ІЗОМОРФІЗМ

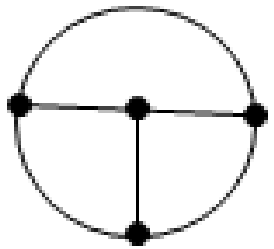
ОСНОВНІ ВИЗНАЧЕННЯ

Нехай *G* та *H* графи, $\varphi: V(G) \to V(H)$ взаємно однозначне відображення множини вершин графу *G* в множину вершин графу *H*.

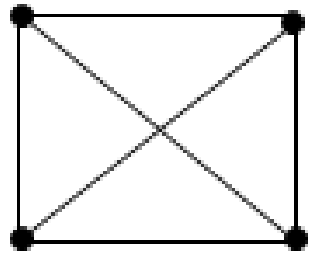
Якщо дві будь-які вершини графу *G* суміжні тоді і тільки тоді, коли їх образи суміжні в *H*, то відображення φ називається ізоморфізмом графу *G* на граф *H*. Такі графи називаються **ізоморфними**.



а)



б)



в)

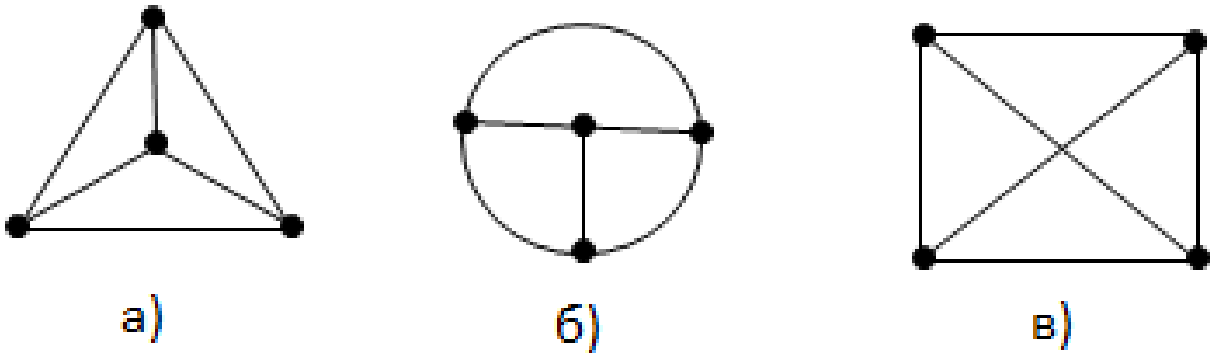
ОСНОВНІ ВИЗНАЧЕННЯ

Плоским називається граф, вершини якого є точками площини, а ребра неперервними плоскими лініями без самоперетинів, причому два ребра не мають спільних точок, окрім, можливо, спільної їх вершини.

Будь-який граф, ізоморфний плоскому графу, називається **планарним**.

Приклад:

Граф в) є планарним, графи а) та б) плоскими.



АЛГОРИТМ УКЛАДКИ ГРАФУ НА ПЛОЩИНІ

Ідея алгоритму полягає у послідовному додаванні до укладеної частини G’ графа *G* нового простого ланцюга, обидва кінці якого належать G’. Цей ланцюг розбиває одну з граней графа на дві. Як вихідний підграф G‘ обирається довільний простий цикл графу *G*. Процес продовжується до тих пір, поки ми не побудуємо плоский граф, ізоморфний вихідному. Якщо на якомусь етапі алгоритму продовжувати процес виявиться неможливим, а плоска укладка ще не одержана, то такий граф не є планарним.

γ-АЛГОРИТМ

0. Оберемо деякий плоский цикл *C* графу *G* і укладемо його на площині.

1. Знайдемо грані графа G', сегменти відносно G'. Якщо множина сегментів порожня, то перейдемо на пункт 7. Розглянемо інші можливі випадки:

a) Існує сегмент для якого немає припустимої грані. У цьому випадку граф непланарний.

b) Для деякого сегмента є лише одна припустима грань. Тоді наступний крок це розміщення довільного α-ланцюга сегменту S у цій грані.

c) $\Gamma(S) \geq 2$ для будь-якого сегменту *S* виникає кілька варіантів продовження побудови. Але справедливе наступне твердження: для продовження γ-алгоритму можна обирати будь-який α-ланцюг і розміщати його в будь-який припустимій грані.

2. Для кожного сегменту *S* визначаємо $\Gamma(S)$.

3. Якщо існує сегмент *S*, для якого $\Gamma(S)=\varnothing$, то граф непланарний. Алгоритм закінчує роботу. Якщо ні, то перехід до пункту 4.

4. Якщо існує сегмент *S*, для якого маємо лише одну припустиму грань Γ, то перейдемо на пункт 6, у протилежному випадку до пункту 5.

5. Для деякого сегменту $S(\Gamma(S)>1)$ обираємо будь-яку припустиму грань.

6. Розмістимо довільний α-ланцюг $L \in S$ до грані Γ. Перейдемо до пункту 1.

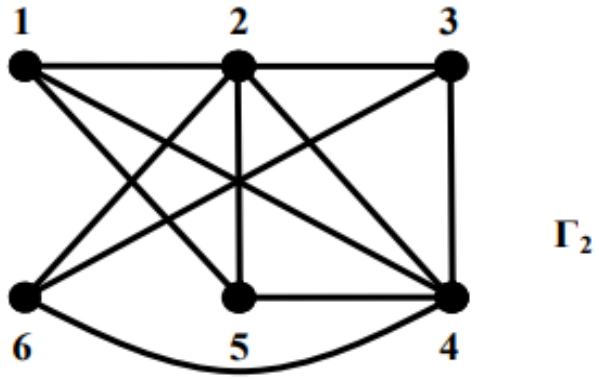
7. Побудована укладка G' графу G на площині. Кінець роботи.

ПРИКЛАД

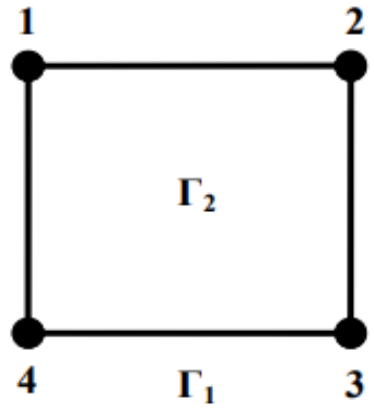
Визначити планарність та побудувати плоску укладку графа G.

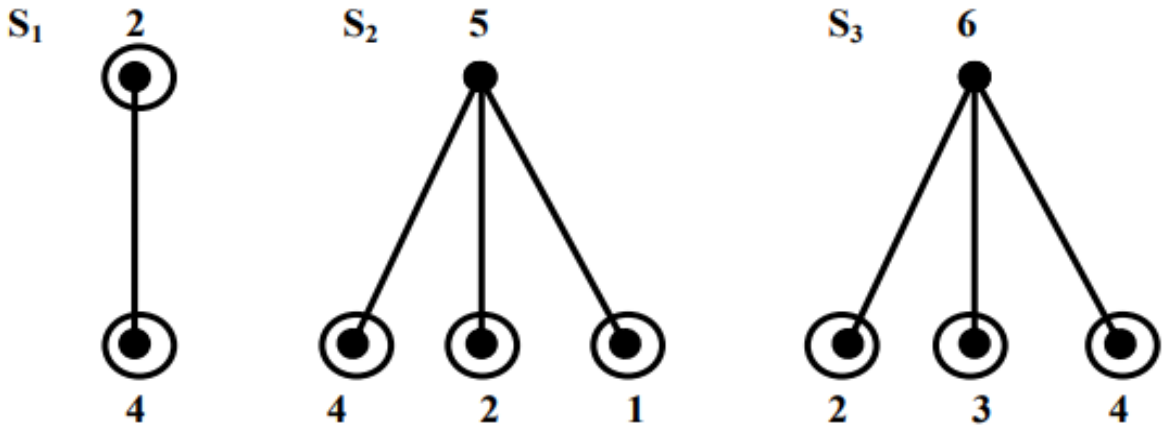
Рішення:

Визначимо множину сегментів відносно поточної плоскої укладки.



Γ_2





ПРИКЛАД

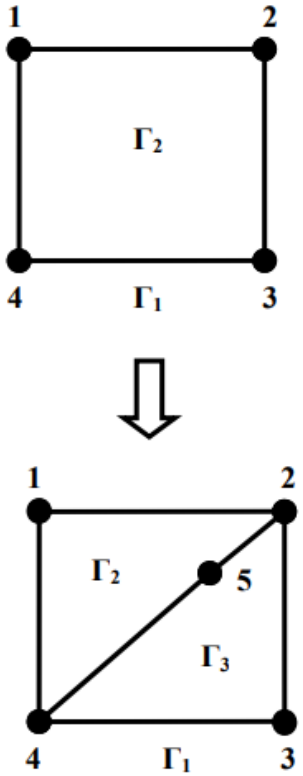
Рішення:

Оберемо в графі довільний простий цикл та укладемо його на площині.

Визначимо для нього множину граней.

Множина сегментів не порожня. Визначимо для кожного сегмента множину допустимих граней: Г(S1)=Г(S2) =Г(S3) = {Г1, Г2}.

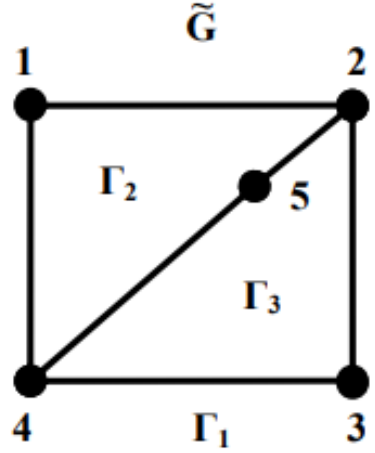
Оберемо сегмент S2 і α- ланцюг = {4, 5, 2}, укладемо його до однієї з допустимих граней, нехай це буде Г2 .

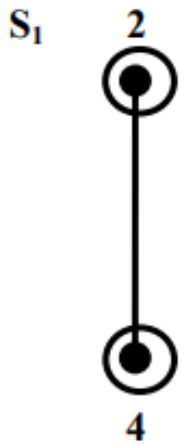


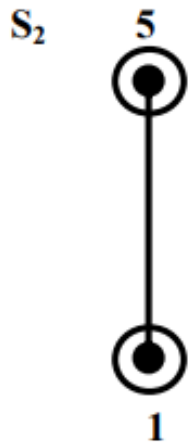
ПРИКЛАД

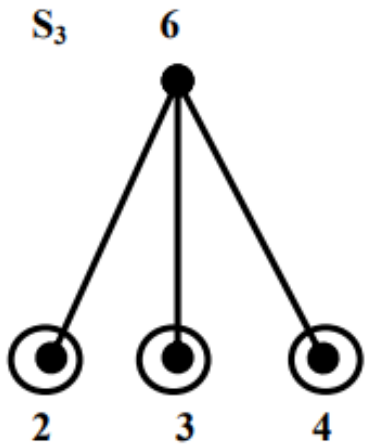
Продовження рішення:

Для нової поточної плоскої укладки визначаємо нові множини граней і сегментів.
Для кожного сегмента визначимо множини допустимих граней:
Г(S1)={Г1, Г2, Г3} ,
Г(S2)={Г2},
Г(S3)= {Г1, Г3}
Обираємо сегмент S2 і укладаємо його в єдину для нього допустиму грань Г2 . Отримуємо поточну часткову плоску укладку графа G .







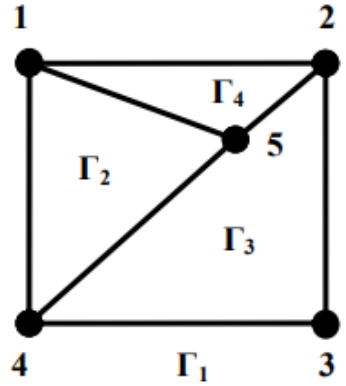


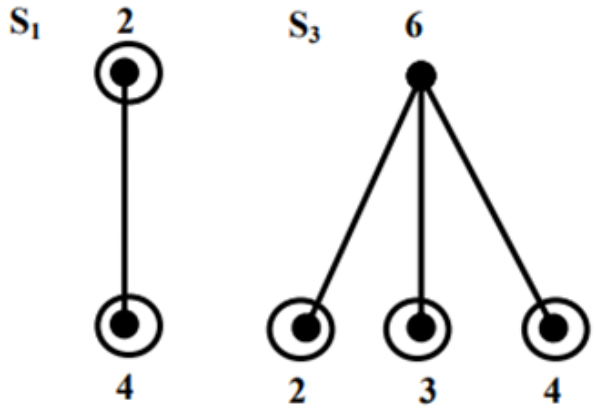
ПРИКЛАД

Продовження рішення:

Для нової поточної часткової укладки G’ визначимо множину граней та сегментів.

Множини допустимих граней для сегментів: Г(S1)=Г(S3)= {Г1, Г3}
Обираємо сегмент S1 та укладаємо до допустимої для нього грані Г3 .





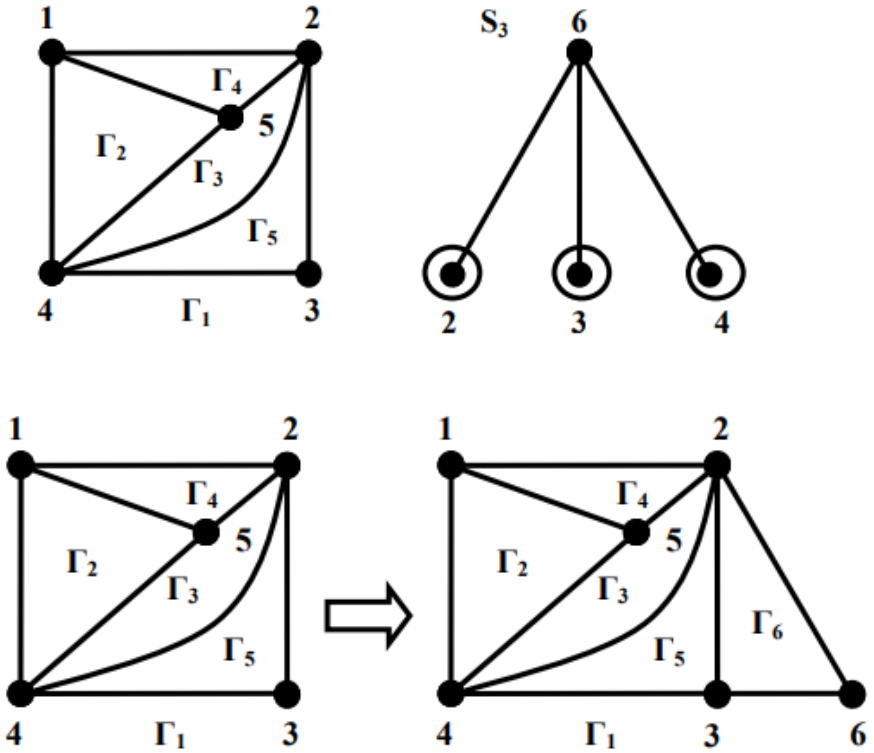
ПРИКЛАД

Продовження рішення:

Отримуємо нову часткову укладку. Визначаємо для неї множину граней та сегментів.

Множина допустимих граней для сегмента Г(S3)={Г1, Г5}

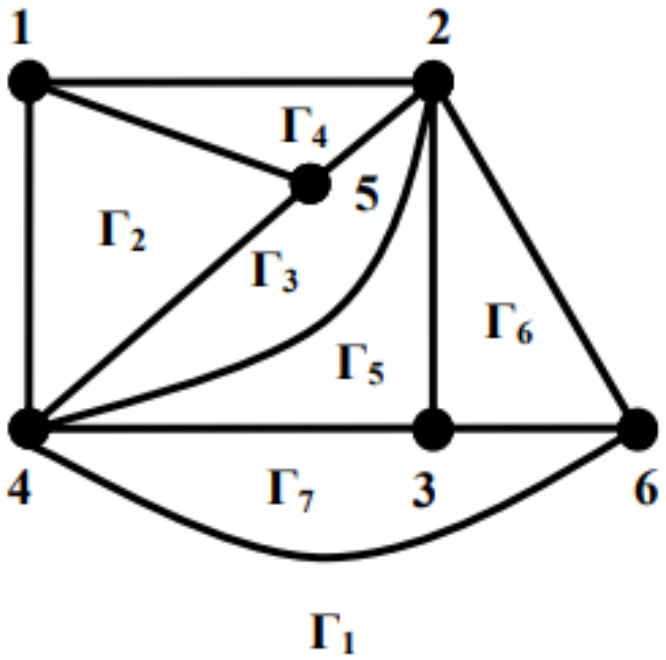
Обираємо грань Г1 і α -ланцюг = {3, 6, 2}, укладаємо його до допустимої грані та отримуємо нову поточну часткову укладку.



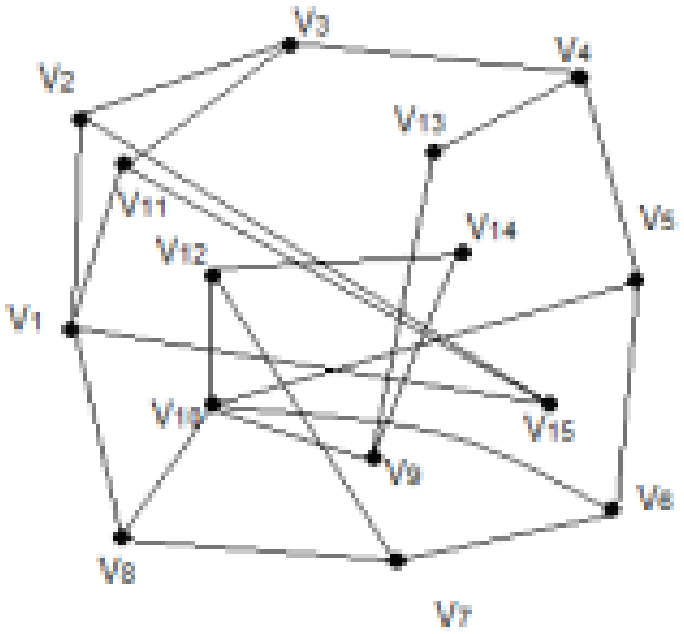
ПРИКЛАД

Продовження рішення:

Для сегмента S3 множина допустимих граней складає: Г(S3)={Г1} .
Укладаємо сегмент S3 до допустимої грані Г1.
Отримуємо нову поточну плоску укладку.
Множина сегментів порожня, отже, алгоритм закінчив роботу. **Вхідний граф – планарний і побудована його плоска укладка.**



γ-АЛГОРИТМ



THANK YOU