Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського» Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

# Дискретна математика

Лабораторна робота №3

«Графи. Способи представлення графів. Остовні дерева.

Пошук найкоротших шляхів».

Київ 2022р.

Виконала: студентка групи ІО-15

Кушнерик Є. О.

Залікова книжка № 1508

Перевірив: Пономаренко А. М.

# Лабораторна робота № 3

**Тема**: «Графи. Способи представлення графів. Остовні дерева. Пошук найкоротших шляхів».

**Мета роботи**: Вивчення властивостей графів, способів їх представлення та основних алгоритмів на графах.

**Завдання**: створити програму, яка реалізує один з алгоритмів на графах.

# Теоретичні основи

* 1. **ОСНОВНІ ОЗНАЧЕННЯ**

Останнім часом теорія графів стала простим, доступним і потужним засобом вирішення завдань, що відносяться до широкого кола проблем. Це проблеми проектування інтегральних схем і схем управління, дослідження автоматів, логічних ланцюгів, блок-схем програм, економіки та статистики, хімії та біології, теорії розкладів і дискретної оптимізації.

**Граф** *G* задають множиною точок або вершин *x1, x2, ..., хn* (яку позначають

через *X*) і множиною ліній або ребер *a1, a2*, ..., (яку позначають символом *А*), що з'єднують між собою всі або частину цих точок. Таким чином, граф *G* повністю задають (і позначають) парою (*X, А*).

**Контуром** (*простим ланцюгом*) називають такий шлях (маршрут), в якому

кожна вершина використовується не більше одного разу. Наприклад, шлях *μ2* є контуром, а шляхи *μ1* і *μ3* – ні. Очевидно, що контур є також ланцюгом, але зворотне твердження невірне. Наприклад, шлях *μ1* є ланцюгом, але не

контуром, шлях *μ2* є ланцюгом і контуром, а шлях *μ3* не є ні ланцюгом, ні контуром.

Аналогічно визначають простий ланцюг у неорієнтованих графах. Так, наприклад, маршрут *μ4* є простим ланцюгом, маршрут *μ5* – ланцюг, а маршрут *μ6* не є ланцюгом.

Шлях або маршрут можна зображати також послідовністю вершин. Наприклад, шлях *μ1* можна представити так: *μ1*={*x2, x5, x4, x3, x5, x6*} і таке представлення часто виявляється більш корисним у тих випадках, коли здійснюється пошук контурів або простих ланцюгів.

# МАТРИЧНІ ПРЕДСТАВЛЕННЯ

* + 1. **МАТРИЦЯ СУМІЖНОСТІ**

Нехай дано граф *G*, його матрицю суміжності позначають через **A***=*[*aij*] і визначають наступним чином: *aij*=1, якщо в *G* існує дуга (*xi*,*xj*), *aij*=0, якщо в *G* немає дуги (*xi*,*xj*).

Матриця суміжності повністю визначає структуру графа. Наприклад, сума всіх елементів рядка *xi* матриці дає напівстепінь виходу вершини *xi*, а сума

елементів стовпця *xi* - напівстепінь входу вершини *xi*. Множина стовпців, які мають 1 у рядку *xi*, є множиною Г(*xi*), а множина рядків, які мають 1 у

стовпчику *xi* , збігається з множиною Г-1(*xi*).

У разі неорієнтованого графа матриця суміжності є симетричною відносно головної діагоналі

# МАТРИЦЯ ІНЦИДЕНТНОСТІ

Нехай дано граф *G* з *n* вершинами і *m* дугами. *Матриця інцидентності* графа *G* позначається через **B**=[*bij*] і є матрицею розмірності *n x m*, яка визначається таким чином: *bij*=1, якщо *xi* є початковою вершиною дуги *aj*; *bij*=-1, якщо *xi* є кінцевою вершиною дуги *aj*; *bij*=0, якщо *xi* не є кінцевою вершиною дуги *aj*.

# НАЙКОРОТШИЙ ОСТОВ ГРАФА

Розглянемо метод прямої побудови *найкоротших остовних дерев* у

зваженому графі (в якому ваги приписані дугам). Найкоротше остовне дерево графа застосовують при прокладці доріг (газопроводів, ліній електропередач і т. д.), коли необхідно пов'язати *n* точок деякою мережею так, щоб загальна довжина "ліній зв'язку" була мінімальною. Якщо точки лежать на евклідовій площині, то їх можна вважати вершинами повного графа *G* з вагами дуг, що дорівнюють відповідним "прямолінійним" відстаням між кінцевими точками дуг.

Якщо "розгалуження" доріг допускається тільки в заданих *n* точках,

найкоротше остовне дерево графа *G* буде якраз необхідною мережею доріг, яка має найменшу вагу.

Розглянемо зважений зв'язний неорієнтовані граф *G=*(*X,А*); вагу ребра (*xi,xj*) позначимо *cij*. З великого числа остовів графа потрібно знайти один, у якого сума ваг ребер найменша. Така задача виникає, наприклад, у тому випадку,

коли вершини є клемами електричної мережі, які повинні бути з'єднані одна з одною за допомогою проводів найменшої загальної довжини (для зменшення рівня наведень). Інший приклад: вершини представляють міста, які потрібно пов'язати мережею трубопроводів; тоді найменшу загальну довжину труб,

яка повинна бути використана для будівництва (за умови, що поза межами міст "розгалуження" трубопроводів не допускаються), визначають як

найкоротший остов відповідного графа.

Завдання побудови найкоротшого остова графа є однією з небагатьох задач теорії графів, які можна вважати повністю вирішеними.

# 3.6.2. АЛГОРИТМ ФОРДА-БЕЛЛМАНА ЗНАХОДЖЕННЯ МІНІМАЛЬНОГО ШЛЯХУ

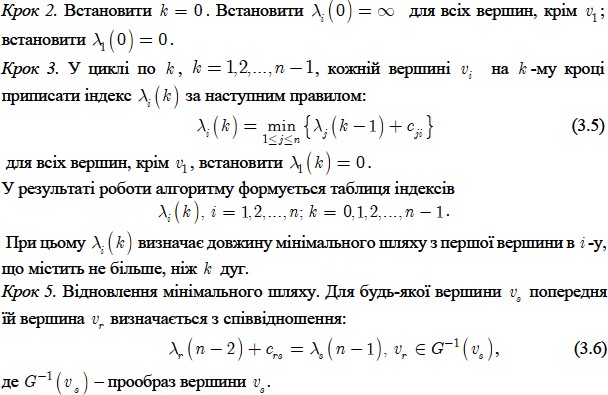
Передбачається, що орієнтований граф не містить контурів від’ємної довжини.

*Алгоритм 1.* (*Алгоритм Форда – Беллмана*) Основними величинами цього алгоритму є величини*λi* (*k* )*,* де *i* = 1,2,...,*n* (*n* – число вершин графа); *k* = 1,2,...,*n* −1. Для фіксованих *i* і *k* величина *λi* (*k* ) дорівнює довжині

мінімального шляху, що веде із заданої початкової вершини *v*1 у вершину *vi* і складається не більше, ніж з *k* дуг.

*Крок 1.* Установка початкових умов. Ввести число вершин графа *n* і матрицю

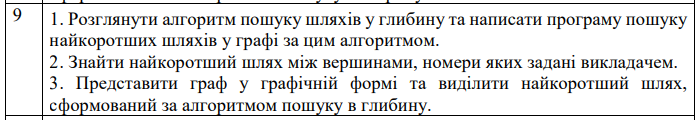
ваг *C* = *cij* .



# 

# Хід роботи

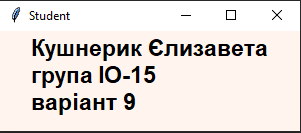
Номер варіанту – 9

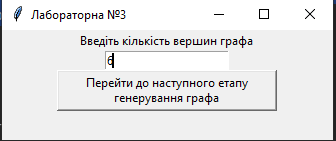


Код програми індивідуального варіанту:

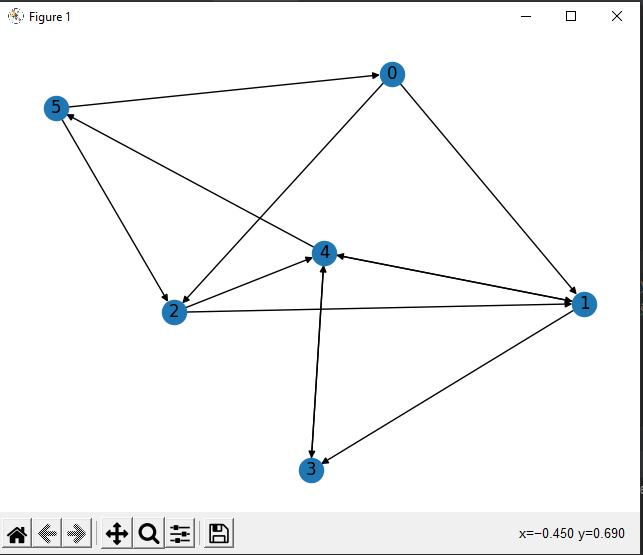
import networkx as nx  
import pylab  
import matplotlib.pyplot as plt  
pos = {0: {1, 2},  
 1: {3, 4},  
 2: {1, 4},  
 3: {4},  
 4: {1, 3, 5},  
 5: {0, 2}}  
  
N = len(pos)  
G = nx.DiGraph()  
a = [(i, j) for i in range(N) for j in pos[i]] # генерация списка рёбер  
  
G.add\_nodes\_from(range(N))  
G.add\_edges\_from(a)  
  
nx.draw(G, with\_labels=True)  
  
pylab.figure()  
plt.show()  
  
def dfs(graph, s, out=0):  
 parents = {v: None for v in graph}  
 level = {v: None for v in graph}  
 level[s] = 0 # уровень начальной вершины  
 queue = [s] # добавляем начальную вершину в очередь  
 while queue: # пока там что-то есть  
 v = queue.pop(-1) # извлекаем вершину  
 for w in graph[v]: # запускаем обход из вершины v  
 if level[w] is None: # проверка на посещенность  
 queue.append(w) # добавление вершины в очередь  
 parents[w] = v  
 level[w] = level[v] + 1 # подсчитываем уровень вершины  
 if out: print(level[w], level, queue)  
 return level, parents  
  
def PATH(end, parents):  
 path = [end]  
 parent = parents[end]  
 while not parent is None:  
 path.append(parent)  
 parent = parents[parent]  
 return path[::-1]  
  
level, parents = dfs(pos, 2, out=0)  
path = PATH(3, parents)  
red\_node = set(path) # вершины маршрута  
red\_edges = [ (path[i],path[i+1]) for i in range(len(path)-1) ] # рёбра маршрута  
  
# разделение по цветам вершин и рёбер  
node\_colours = ['g' if not node in red\_node else 'red' for node in G.nodes()]  
black\_edges = [edge for edge in G.edges() if edge not in red\_edges]  
  
# построение графа  
#p = nx.spring\_layout(G)  
  
p = {0: [ 0.38144628, -0.66882419],  
 1: [0.23970166, 0.49135202],  
 2: [0.41724407, 0.05678197],  
 3: [-0.55966794, 1. ],  
 4: [-0.44016179, 0.07245783],  
 5: [-0.03856228, -0.95176763]}  
  
nx.draw\_networkx\_nodes(G, p, cmap=plt.get\_cmap('jet'), node\_color=node\_colours, node\_size=500)  
nx.draw\_networkx\_labels(G, p)  
nx.draw\_networkx\_edges(G, p, edgelist=black\_edges, width=2.0, edge\_color='k', arrows=True)  
nx.draw\_networkx\_edges(G, p, edgelist=red\_edges, width=3.0, edge\_color='r', arrows=True)  
  
plt.show()

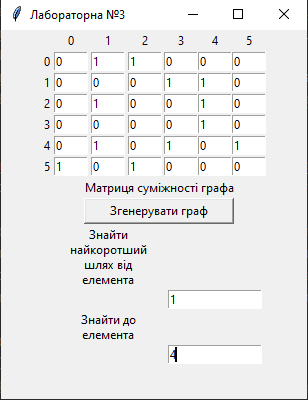
Результати роботи:

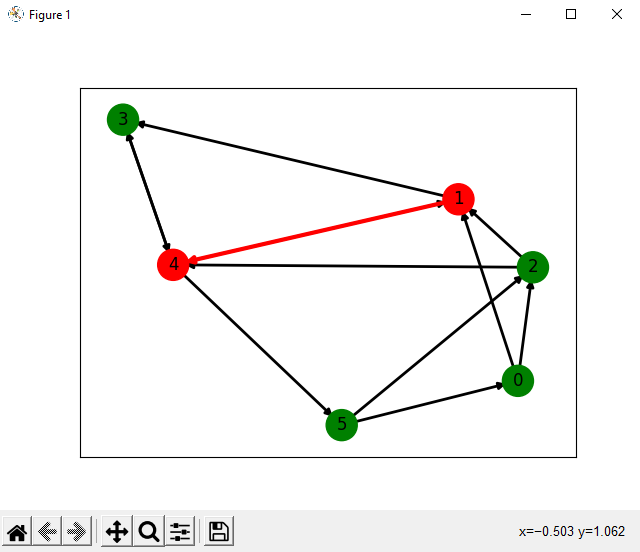


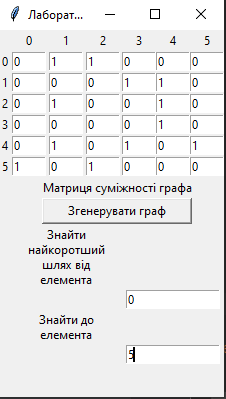


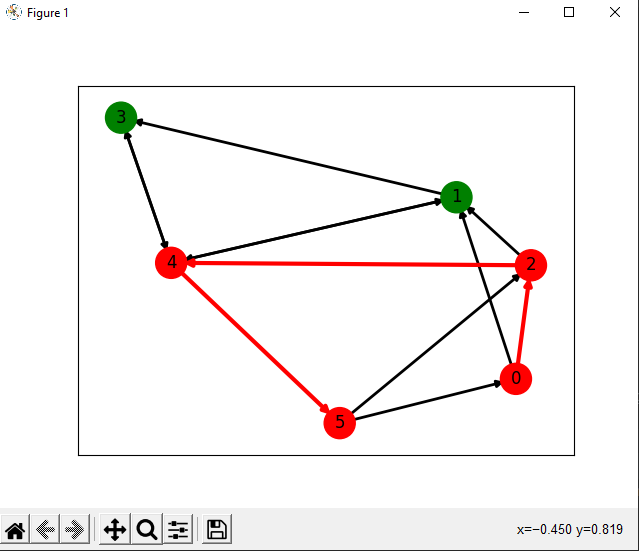
Граф без змін

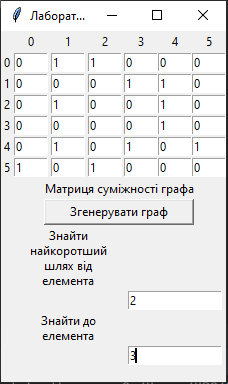


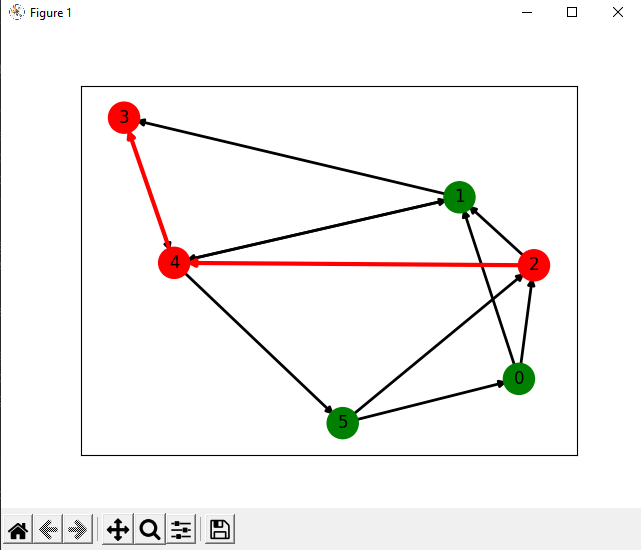










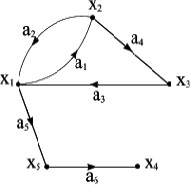


**Висновок:** Протягом виконання роботи я вивчила основні методи роботи з графами, їх види та алгоритми пошуку шляхів, закріпила занання GUI в модулі tkinter в Python та покращила знання у використанні модуля NetworkX.

# Контрольні запитання

1. Назвіть основні способи представлення графів.

Основними представленнями графа є матрицею (матриця інцидентності та суміжності) та списком (список ребер та список суміжності).

1. Покажіть на прикладі пряму і зворотну відповідність для заданої вершини.



1. Чому дорівнює сума степенів усіх вершин неорієнтованого графа? Сума степенів вершин графа дорівнює подвоєній кількості ребер.
2. У чому відмінності матричного представлення орієнтованих і неорієнтованих графів?

В матриці інцидентності для орієнтованого графа *bij*=1, якщо *xi* є початковою вершиною дуги *aj*; *bij*=-1, якщо *xi* є кінцевою вершиною дуги *aj*; *bij*=0, якщо *xi* не є кінцевою вершиною дуги *aj*. Для неорієнтованого - *bij*=1, якщо *xi* є

кінцевою вершиною дуги *aj*; *bij*=0, якщо *xi* не є кінцевою вершиною дуги *aj.*

1. У чому особливості представлення графа матрицею суміжності?

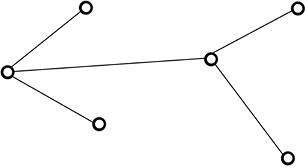
В матриці суміжності елементами є тільки 1 або 0, які показують чи наявна між вершинами дуга.

1. У чому особливості представлення графа матрицею інцидентності?

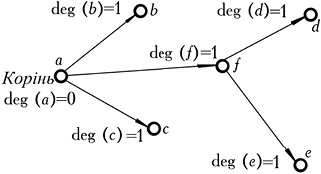
В матриці інцидентності для задавання використовуються вже три елемента – 1, -1 та 0. Через те, що для позначання дуг в цій матриці виділені всі стовпці, що дає можливість працювати з мультиграфами.

1. Дайте визначення дерева; орієнтованого дерева.

*Неорієнтованим деревом* називають зв’язний неорієнтований граф без циклів

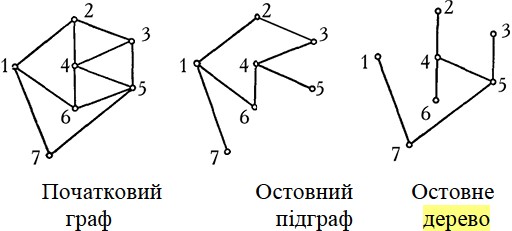


**Орієнтованим деревом** називають зв’язний орієнтований граф без циклів, у якому напівстепінь входу кожної вершини, за винятком кореневої, дорівнює одиниці, а напівстепінь входу кореневої вершини дорівнює 0.



1. Яке дерево називають остовним?

*Остовним деревом* графа *G* називають ***остовний підграф*** графа *G*, який є деревом. *Остовним підграфом* називають такий підграф, у якому множина його вершин збігається з множиною вершин самого графа.



1. Теорема Келі.

*Теорема Келі.* Число різних дерев, які можна побудувати на *n* вершинах, дорівнює *n^(n*2).

1. Що називають коренем дерева?

Корінь дерева – це точка з якої починається будова дерева.

1. Як перетворити неорієнтоване дерево в орієнтоване?

Треба взяти його довільну вершину в якості кореня і ребрам приписати таку орієнтацію, щоб кожна вершина з'єднувалася з коренем (тільки одним)

простим ланцюгом.

1. Скільки ребер містить остовне дерево графа?

n-1 ребро.

1. Завдання знаходження найкоротшого остова графа. Знайти остовне дерево в якому сума всіх ваг найменша.
2. Наведіть практичні приклади знаходження найкоротшого остова графа. Прокладання газопроводу, доріг через міста, тобто логістичні проблеми.
3. Реалізація алгоритму Прима-Краскала для знаходження найкоротшого остова графа.
4. Дайте визначення шляху, маршруту, ланцюга, контура.

Маршрутом або шляхом у графі G (V,E ) називають послідовність вершин і ребер, які чергується



Ланцюг – це шлях через ребра, які не повторюються.

1. Який граф називають зваженим?

Це граф у якому ребра/дуги мають свою вагу, в деяких випадках назначають і вершинам.

1. Як визначають довжину шляху графа? Сумуванням всіх ваг ребер/дуг.
2. Задача знаходження найкоротшого шляху на графі.

Задача про найкоротший шлях полягає в знаходженні такого шляху між двома вершинами (або вузлами) графу, що сума ваг ребер з яких він

складається мінімальна.

1. Реалізація методу динамічного програмування для знаходження найкоротшого шляху на графі.

Пряма ітерація.

Нехай вузли орієнтованого графа пронумеровані так, що

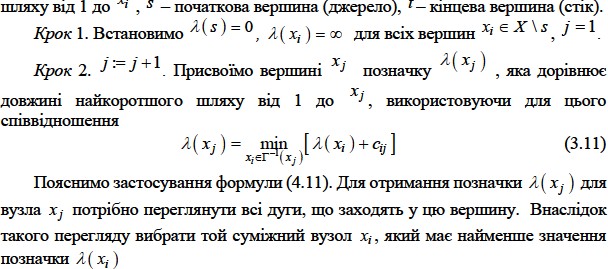
дуга *xi xj*  *E* завжди орієнтована від вузла *xi* до вузла *x j* , що має більший номер. Для ациклічного графа така нумерація завжди можлива і залежить від способу задавання графа.

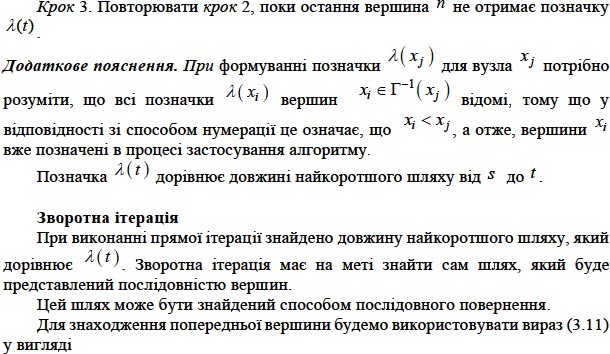
Ациклічними графами будемо називати такі орграфи, що не містять циклів. Приклад орграфа, який містить цикл, показано на рисунку.

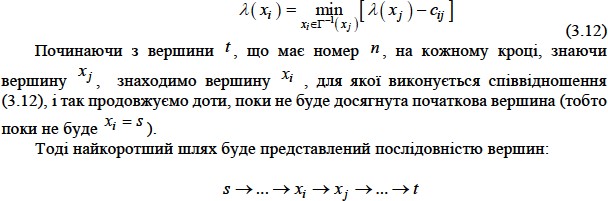
Алгоритм має передбачати перевірку графа, заданого матрицею ваг *C* , на ациклічність.

Така перевірка може бути реалізована методом топологічного сортування, або довільним іншим методом.

Початкова вершина *s* отримує номер 1, а кінцева *t* – номер *n* .

Нехай   *xi*  – позначка вершини *xi* , яка дорівнює довжині найкоротшого

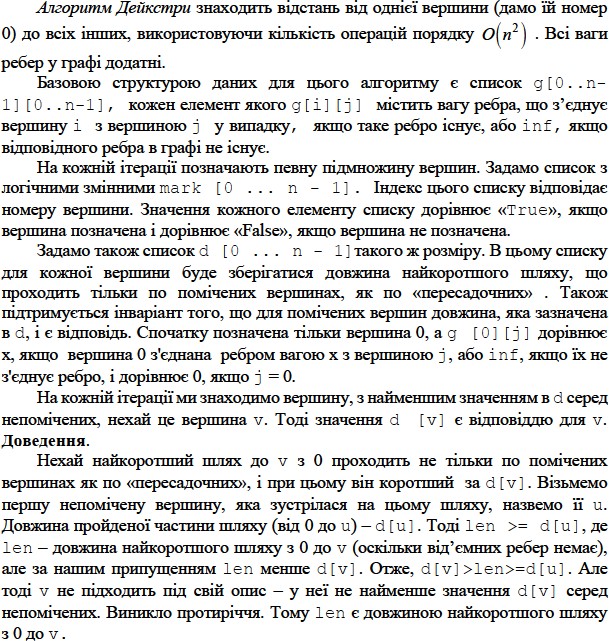


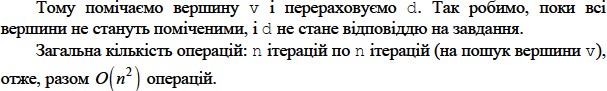


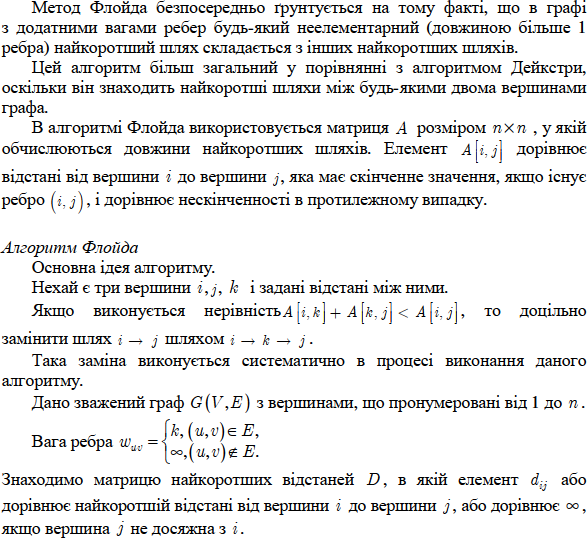
**22.** Що називають правильною нумерацією вершин графа?

Нумерація називається правильної, якщо для будь-якої дуги графа номер початку менше номера кінця.

**24.** Застосування алгоритму Дейкстри для пошуку найкоротших шляхів у графі.





**26.** Алгоритм Флойда-Уоршелла для пошуку найкоротших шляхів у графі.