5.4 Обхід вершин графу

Основні алгоритми обходу вершин графу (або пошуку у графі):

— пошук вшир;

— пошук вглиб.

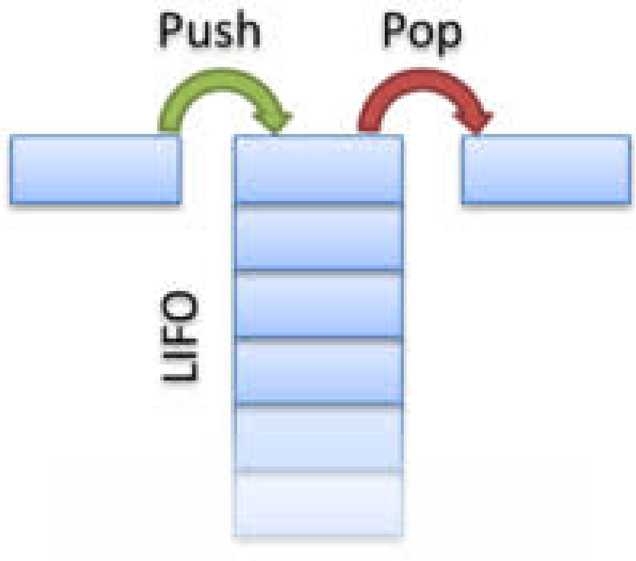
5.4.1 Пошук вглиб або DFS-метод (depth first search)

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).

DFS-номер вершини x позначають $\mathrm{DFS}(x)$.

У ході роботи алгоритму використовують структуру даних для збереження множин, яку називають *стеком*.



Зі стеку можна вилучити тільки той елемент, який було додано до нього останнім: «останнім прийшов — першим вийшов» (*last in*, *first out* — LIFO).

Алгоритм пошуку вглиб у простому зв'язному графі

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.

Алгоритм пошуку вглиб у простому зв'язному графі

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.

Алгоритм пошуку вглиб у простому зв’язному графі

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

Щоб результат виконання алгоритму був однозначним, вершини, які суміжні з вершиною v, аналізують за зростанням їх порядкових номерів (або в алфавітному порядку).

Динаміку роботи алгоритму зручно відображати за допомогою таблиці з трьома стовпцями:

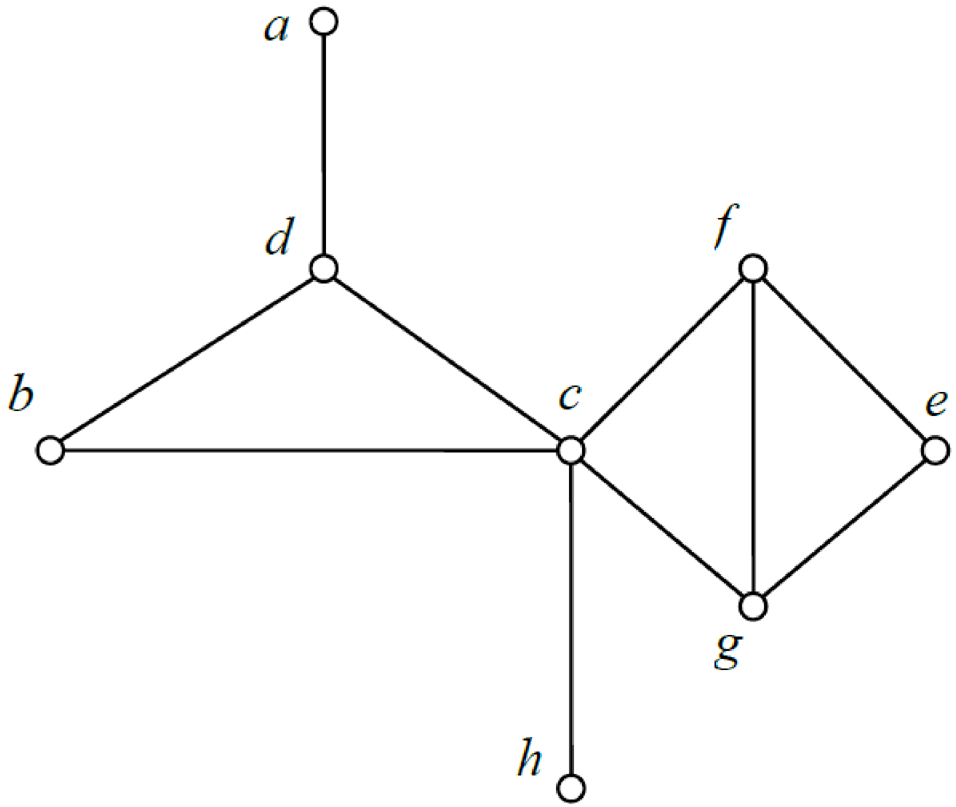
— вершина;

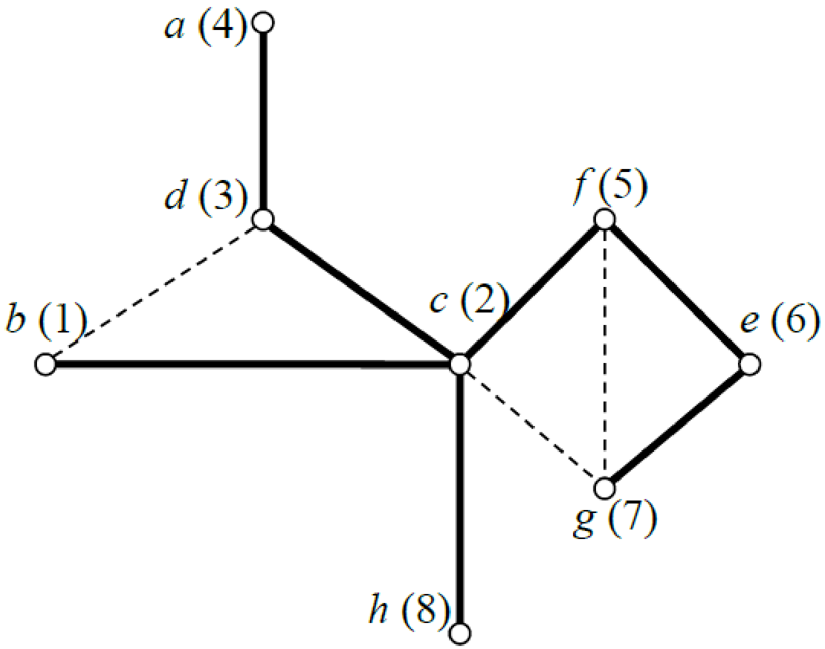
— DFS-номер;

— вміст стеку.

Цю таблицю називають **протоколом обходу** графу пошуком вглиб.

Приклад. Виконати обхід пошуком вглиб даного графу, починаючи з вершини b





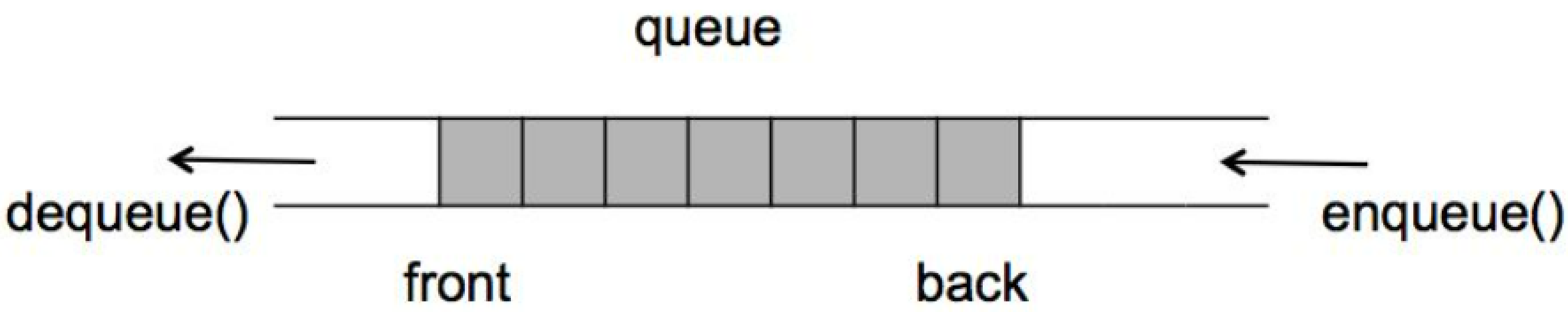
Вершина	**DFS-номер**	**Вміст стеку**
b	1	b
c	2	bc
d	3	bcd
a	4	$bcda$
-	-	bcd
-	-	bc
f	5	bcf
e	6	$bcfe$
g	7	$bcfeg$
-	-	$bcfe$
-	-	bcf
-	-	bc
h	8	bch
-	-	bc
-	-	b
-	-	$\varnothing$

5.4.2 Пошук вшир або BFS-метод (breadth first search)

У ході реалізації алгоритму пошуку вшир використовують структуру даних для збереження множин, яку називають *чергою*.

З черги можна вилучити тільки той елемент, який було додано до неї першим: «першим прийшов — першим вийшов» (*first in*, *first out* — скорочено FIFO).

Елемент включається у *хвіст* черги, а виключається з її *голови*.



Алгоритм пошуку вшир у простому зв’язному графі

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, вже визначено BFS-номери, то перейти до кроку 4, інакше — до кроку 3.

Алгоритм пошуку вшир у простому зв’язному графі

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 зручно відображати за допомогою таблиці з трьома стовпцями:

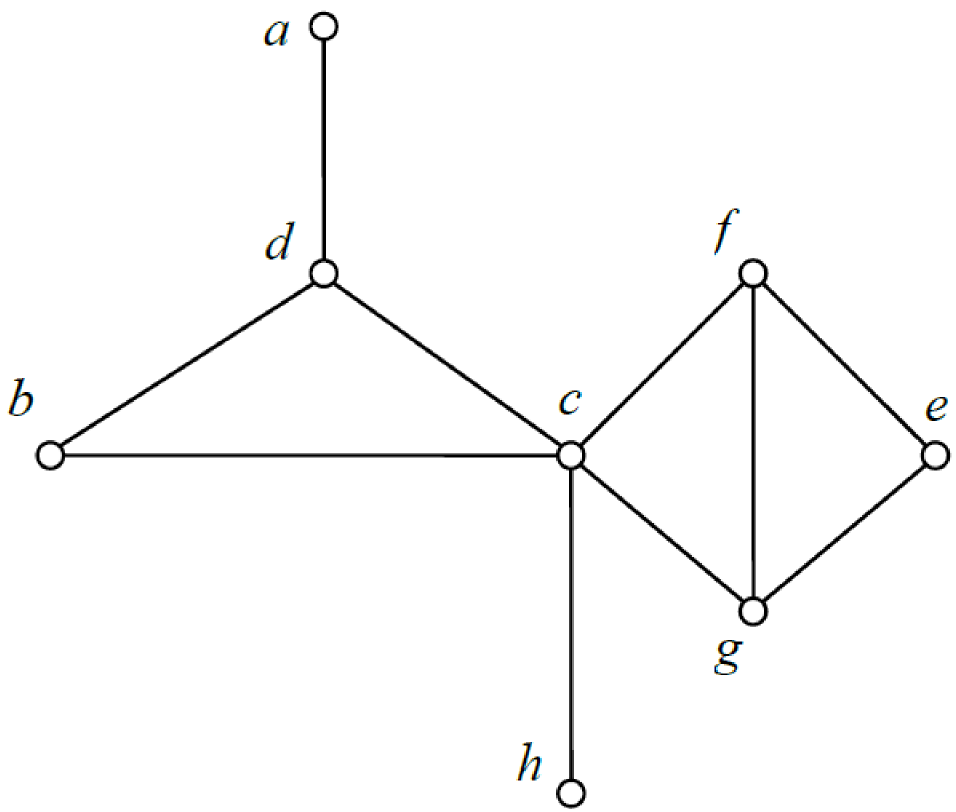
— вершина;

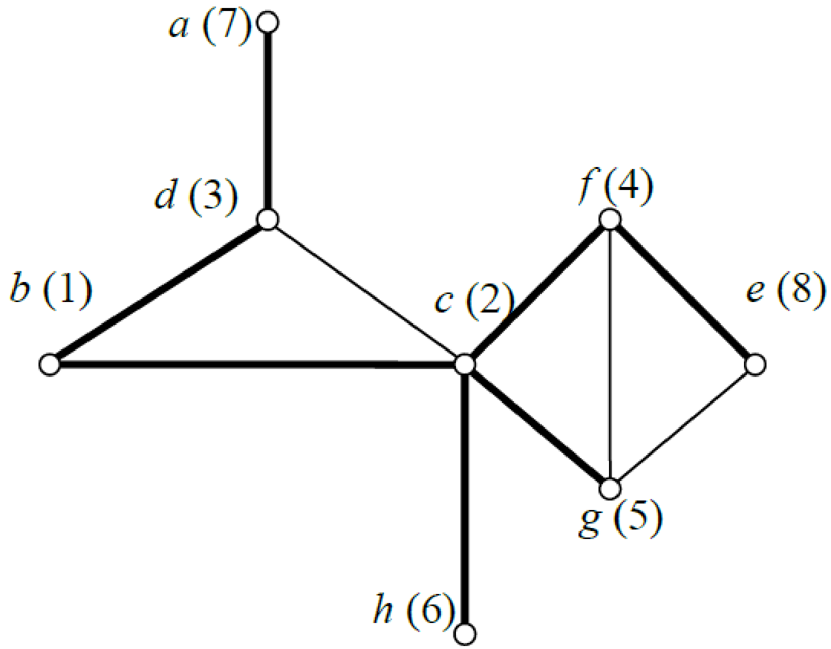
— BFS-номер;

— вміст черги.

Цю таблицю називають **протоколом обходу** графу пошуком вшир.

Приклад. Виконати обхід пошуком вшир даного графу, починаючи з вершини *b*





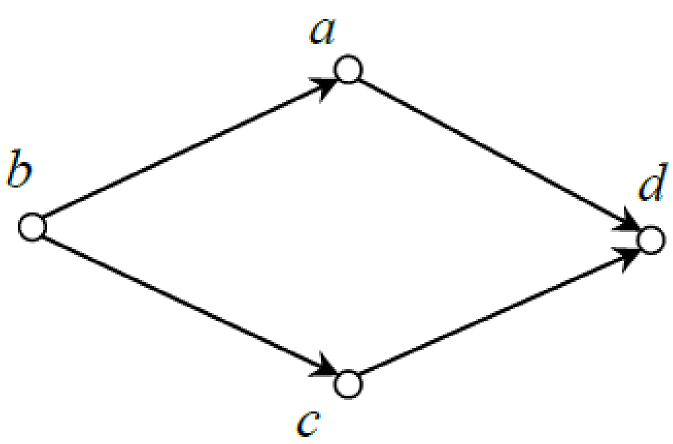
Вершина	BFS-номер	Вміст черги
b	1	*b*
c	2	*bc*
d	3	*bcd*
-	-	*cd*
f	4	*cdf*
g	5	*cdfg*
h	6	*cdfgh*
-	-	*dfgh*
a	7	*dfgha*
-	-	*fgha*
8	*e*	*fghae*
-	-	*ghae*
-	-	*hae*
-	-	*ae*
-	-	*e*
-	-	∅

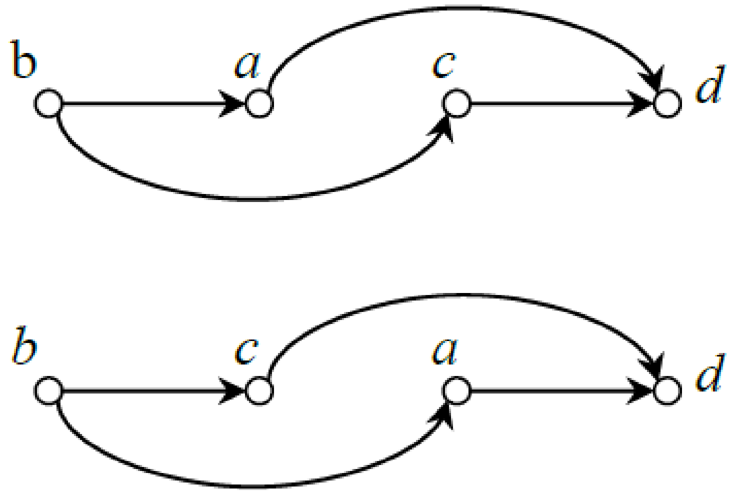
5.4.3 Топологічне сортування

Топологічним сортуванням орієнтованого ациклічного графу $G = (V, E)$ називається таке лінійне впорядкування всіх його вершин, що якщо граф містить ребро (v, u), то вершина v в такому впорядкуванні розташовується до вершини u.

Зауваження. Для виконання топологічного сортування граф повинен бути орієнтованим ациклічним графом (directed acyclic graph — DAG).

Приклад





Застосування топологічного сортування: планування послідовності робіт або завдань на основі їх залежностей.

Роботи представлені вершинами, і є ребро від x до y, якщо завдання x має бути завершено до того, як можна буде розпочати роботу y. Тоді топологічне сортування дає порядок виконання завдань.

Ідея алгоритму Кана:

На кожному кроці з множини вершин графу вилучається вершина, яка не має попередників, і поміщається у результуючу впорядковану множину, при цьому з множини ребер E вилучаються усі ребра, що починаються в цій вершині.

Схема алгоритму топологічного сортування Кана

1 **Вхід:** множина вершин $V = \{1, \dots, n\}$
множина ребер $E = \{(v, u)\}$

2 **Вихід:** L — впорядкована послідовність вершин графу (черга), що буде містити топологічно відсортовані вершини

3 $L := (\,)$, $V' := V$, $E' := E$

4 Визначити множину $S \subseteq V'$ вершин, які не мають вхідних ребер

5 **while** множина V' не пуста

6 В множині S **обрати** вершину v

7 **Помістити** вершину v в кінець черги $L := (L, v)$

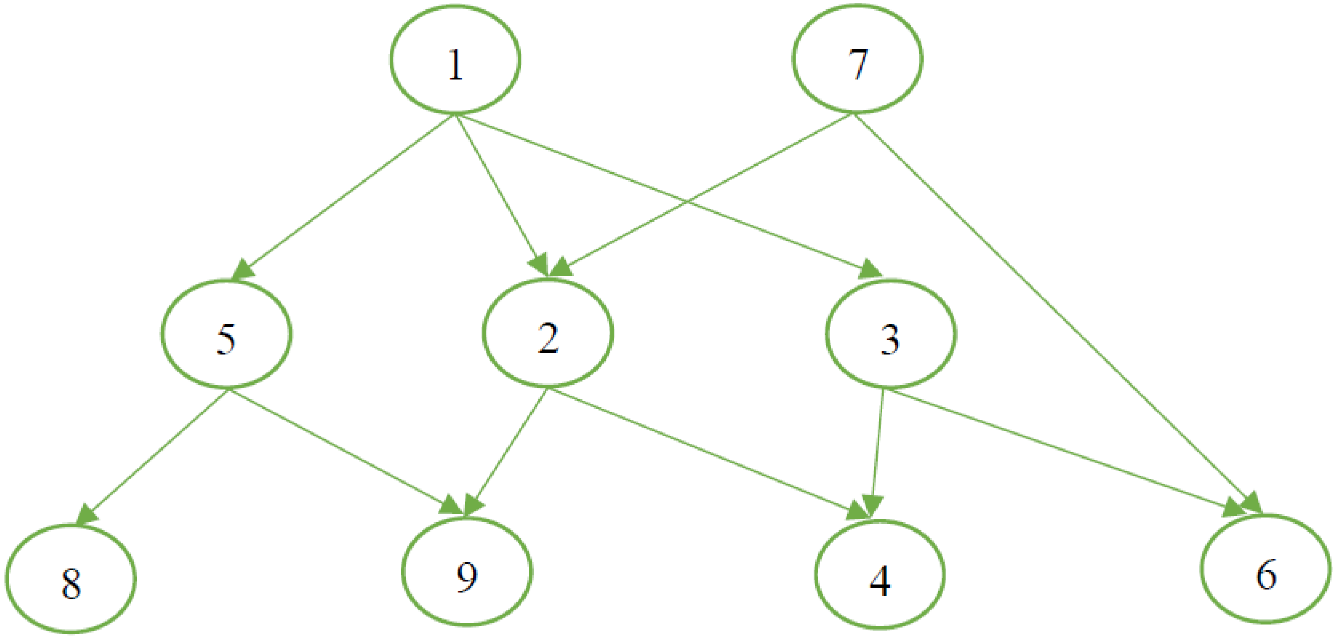
8 **Вилучити** вершину v з V': $V' := V' \backslash v$

9 **Вилучити** з множини E' усі ребра, що починаються в вершині v:
$E' := E' \backslash \{(v, u) \mid \forall\, u\ (v, u) \in E'\}$

10 **Оновити** множину S (додати до неї усі вершини, попередники яких вилучені з розгляду)

11 **end while**

Приклад

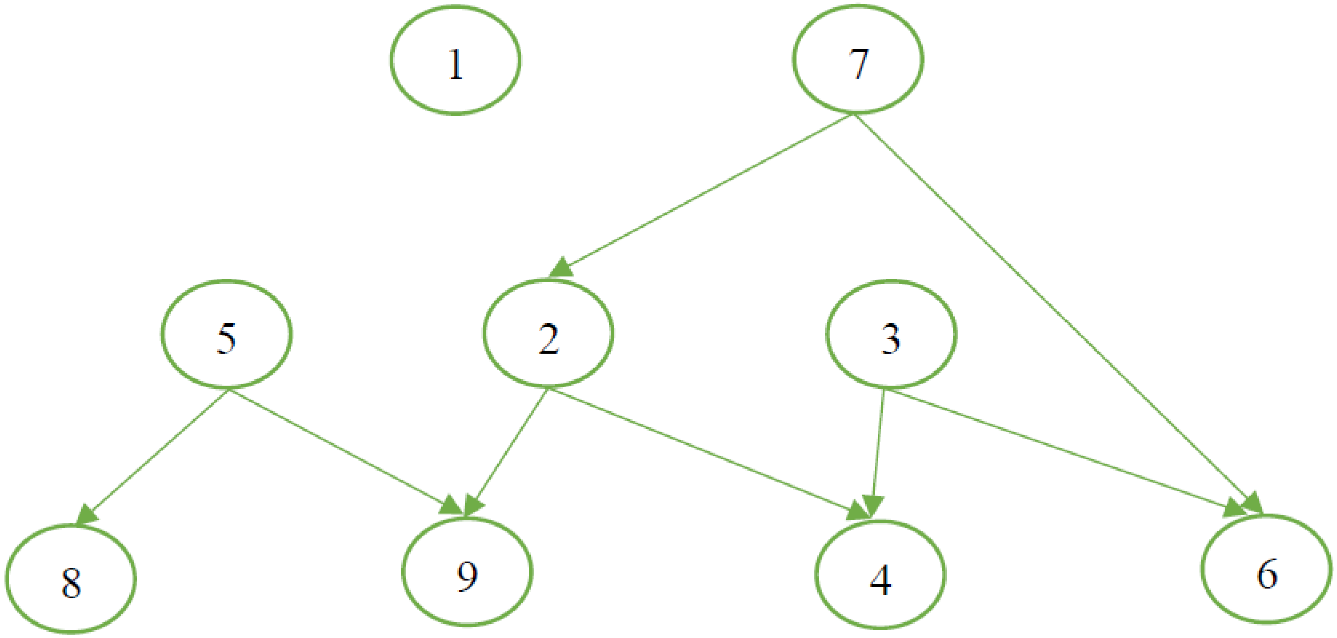


$L = (\ ),\ \ S = \{1, 7\}.$

Вершину 1 помістимо в кінець черги множини L та вилучимо з множини S:

$$L = \{1\},\ \ S = \{7\}.$$

Вилучимо всі ребра, що починаються у вершині 1:



Оновимо множину S

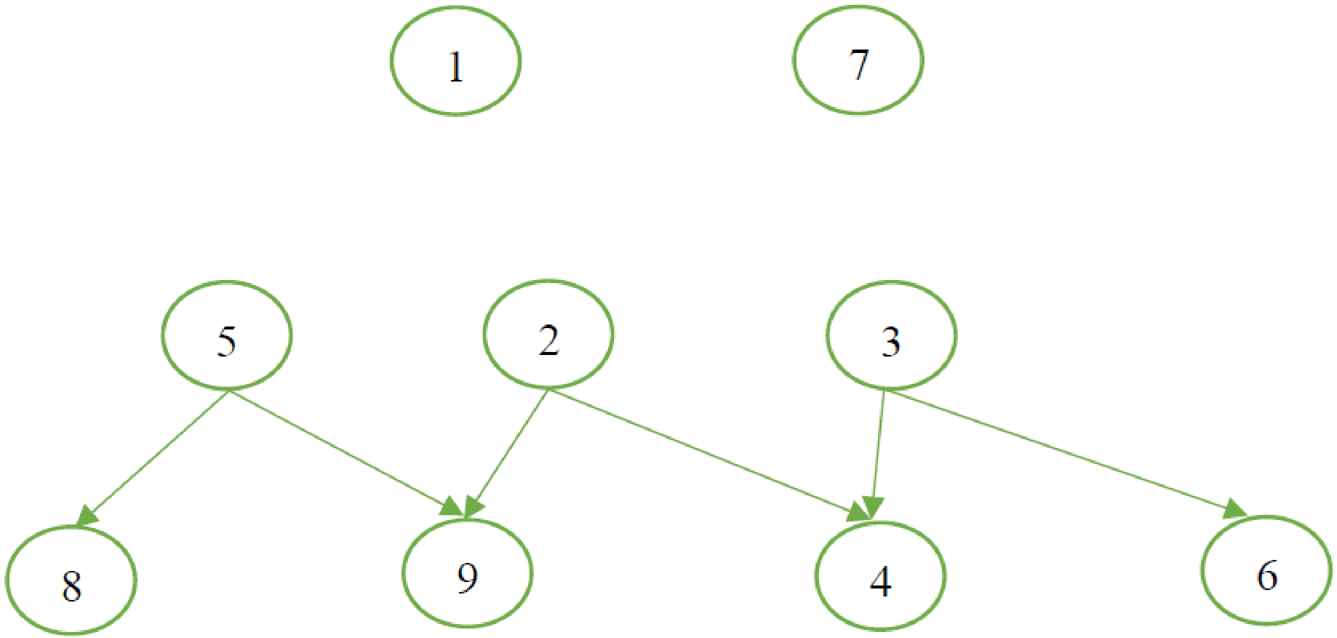
додамо до неї вершини 5 та 3:

$$L = \{1\}, \;\; S = \{7, 5, 3\}.$$

Вершину 7 помістимо в кінець черги множини L та вилучимо з множини S:

$$L = \{1, 7\},\ S = \{5, 3\}.$$

Вилучимо всі ребра, що починаються у вершині 7:



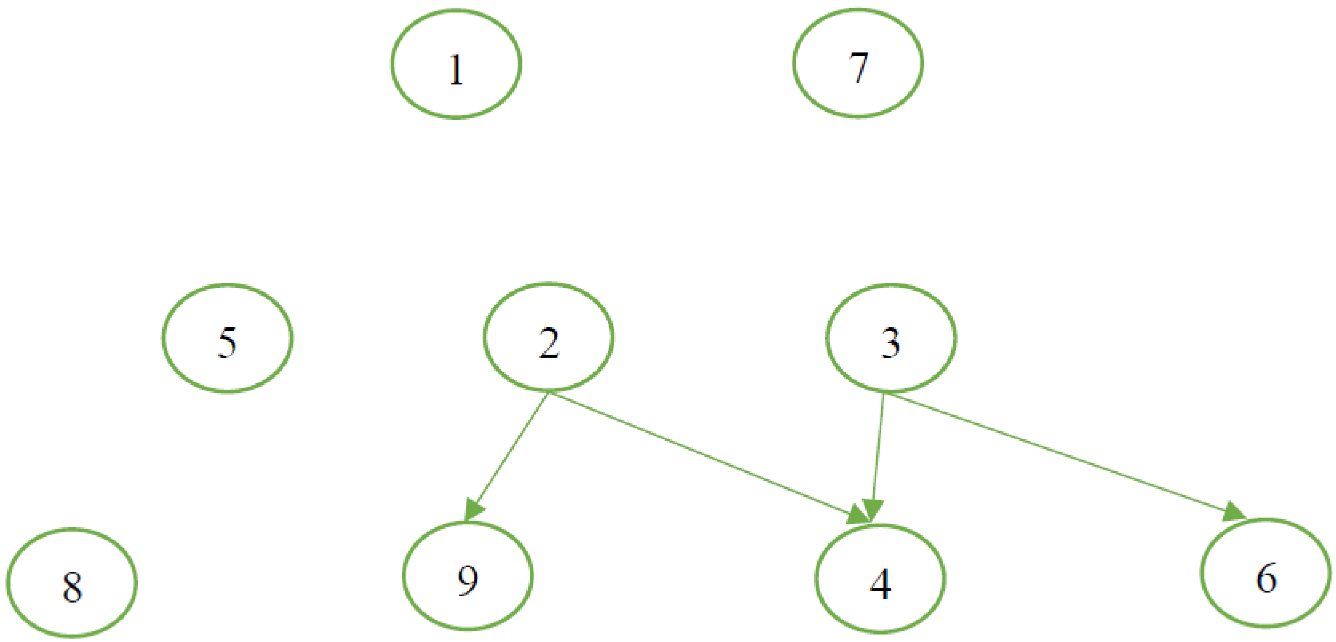
Додамо вершину 2 до множини S:

$$L = \{1, 7\},\ \ S = \{5, 3, 2\}.$$

Аналогічно на наступних кроках роботи алгоритму отримаємо:

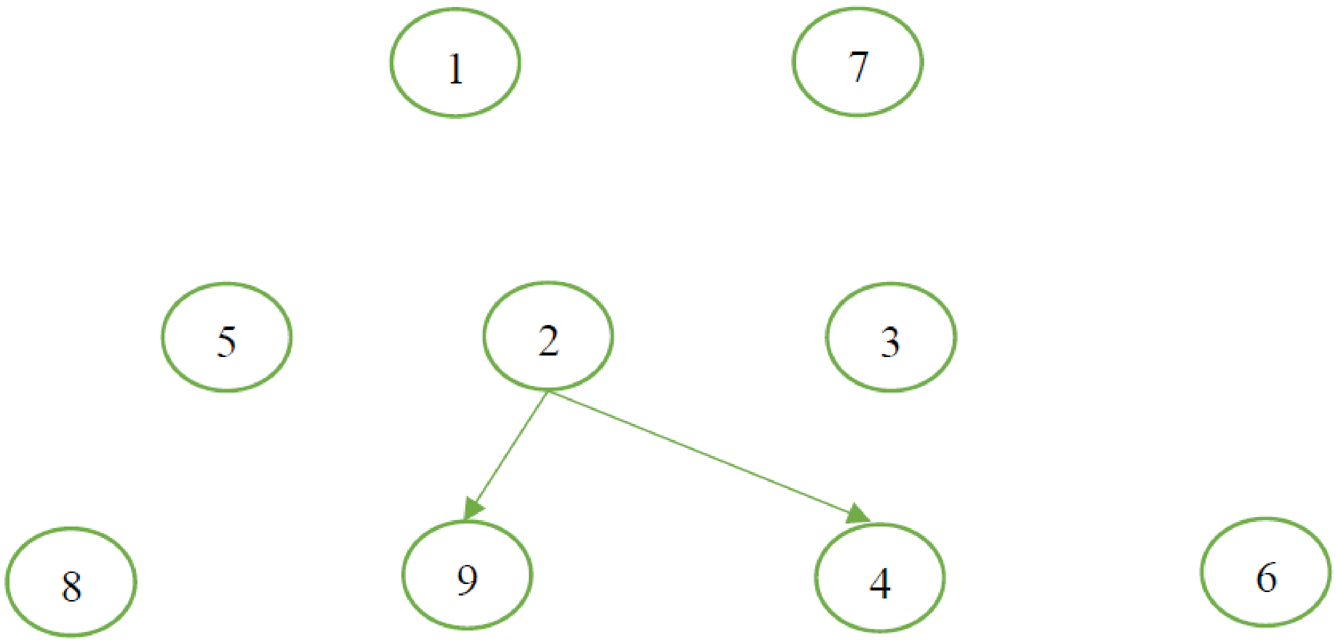
$L = \{1, 7, 5\}, \quad S = \{3, 2\};$

$L = \{1, 7, 5\}, \; S = \{3, 2, 8\}.$



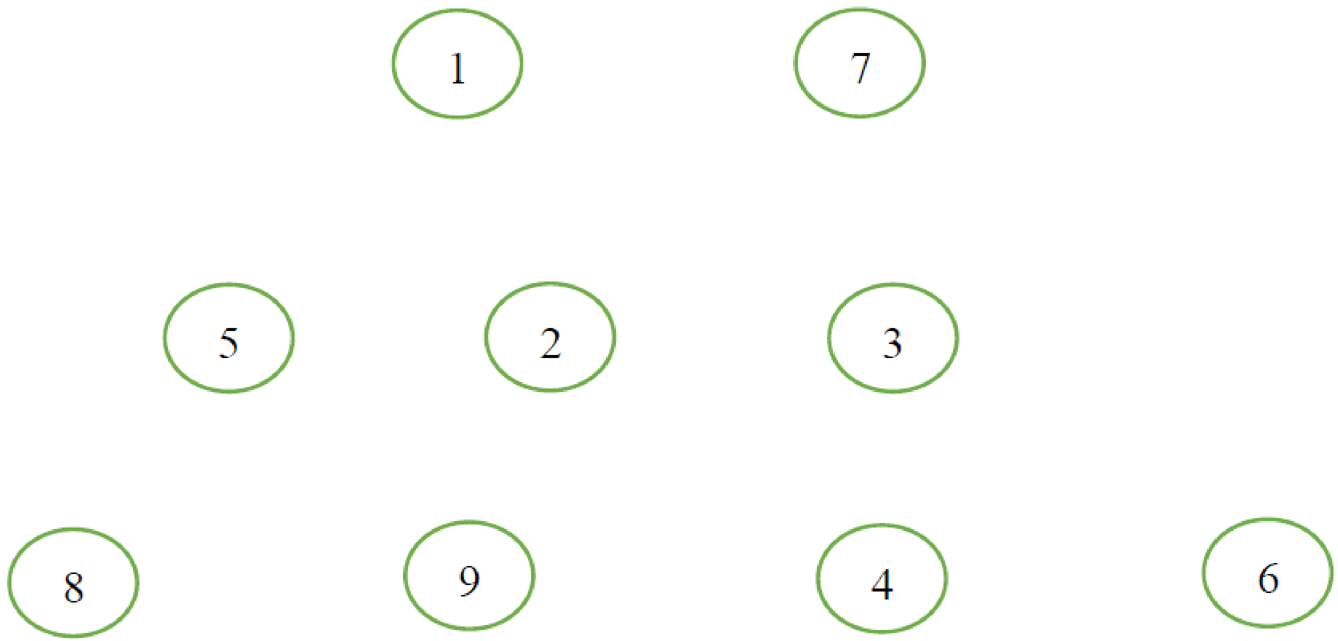
$$L = \{1, 7, 5, 3\},\ \ S = \{2, 8\};$$

$$L = \{1, 7, 5, 3\},\ \ S = \{2, 8, 6\}.$$



$$L = \{1, 7, 5, 3, 2\},\ S = \{8, 6\};$$

$$L = \{1, 7, 5, 3, 2\},\ S = \{8, 6, 9, 4\}.$$



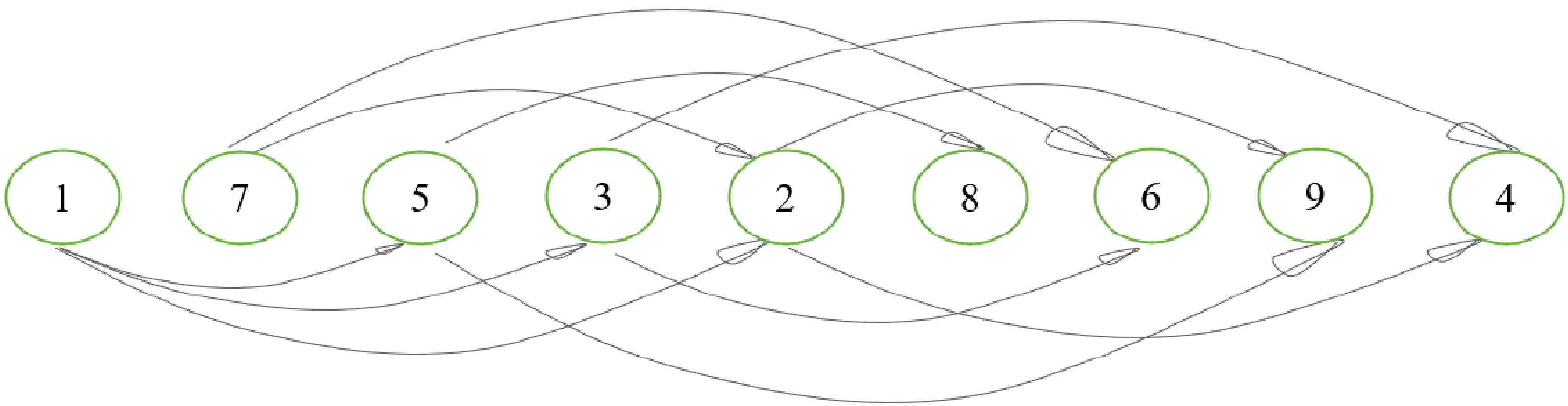
$$L = \{1, 7, 5, 3, 2, 8\}, \;\; S = \{6, 9, 4\}.$$

$$L = \{1, 7, 5, 3, 2, 8, 6\}, \;\; S = \{9, 4\}.$$

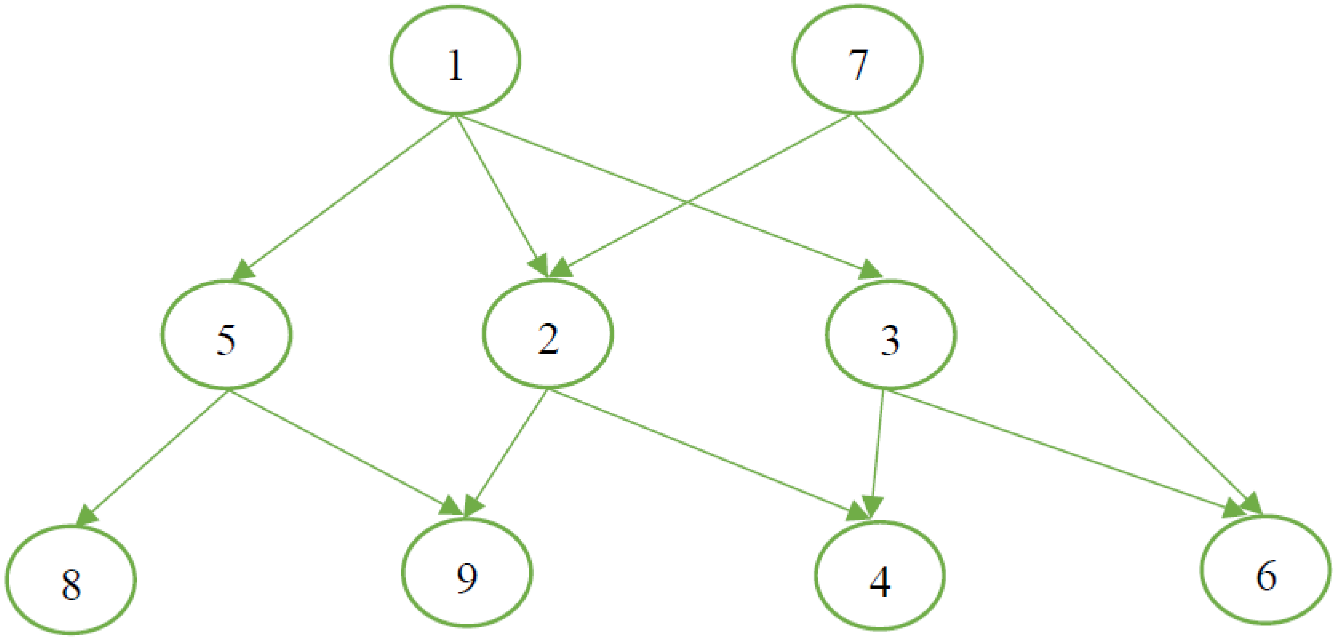
$$L = \{1, 7, 5, 3, 2, 8, 6, 9\}, \;\; S = \{4\}.$$

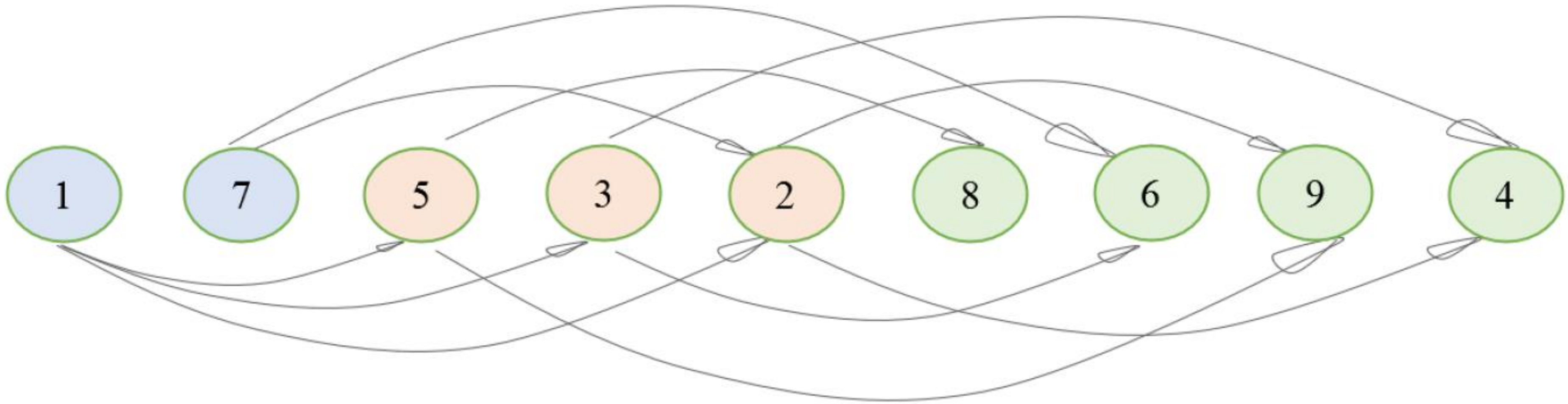
$$L = \{1, 7, 5, 3, 2, 8, 6, 9, 4\}, \;\; S = \{\varnothing\}.$$

Топологічне сортування $L = \{1, 7, 5, 3, 2, 8, 6, 9, 4\}$



В алгоритмі Кана вибір вершини v, яка на поточний момент не має вхідних ребер, не є однозначним. Можливі варіанти правил вибору вершин з множини S: FIFO, LIFO, RANDOM, за кількістю підпорядкованих вершин тощо.


1
7
5
2
3
8
9
4
6


1
7
5
3
2
8
6
9
4

В загальному випадку для графа може існувати декілька топологічних сортувань.

Для графу, що розглядається, існує 2!3!4!=144 топологічних сортування. Наприклад,