МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ МИХАЙЛА ОСТРОГРАДСЬКОГО

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕЛЕКТРИЧНОЇ ІНЖЕНЕРІЇ

ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА АВТОМАТИЗАЦІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

НАВЧАЛЬНА ДИСЦИПЛІНА

«АЛГОРИТМИ І СТРУКТУРИ ДАНИХ»

ЗВІТ

З ПРАКТИЧНОЇ РОБОТИ №5

Виконала:

студентка групи КН-24-1

Процко П.Д.

Перевірив:

доцент кафедри АІС

Сидоренко В. М.

Кременчук 2025

Тема: Алгоритми пошуку та їх складність.

Мета: опанувати основні алгоритми сортування та навчитись методам аналізу їх асимптотичної складності.

Хід роботи

# **Постановка завдання**

# **20. Задано ациклічний граф: {1,2,3,4,5}{(1,2),(1,3),(2,4),(3,5),(4,5), (5, 6)}. Побудувати граф і розв’язати задачу топологічного сортування за допомогою алгоритму DFS.Розв'язання задачі**

Задано ациклічний граф:

{1,2,3,4,5,6}{(1,2),(1,3),(2,4),(3,5),(4,5),(5,6)}.

Побудувати граф і розв'язати задачу топологічного сортування за допомогою алгоритму DFS.

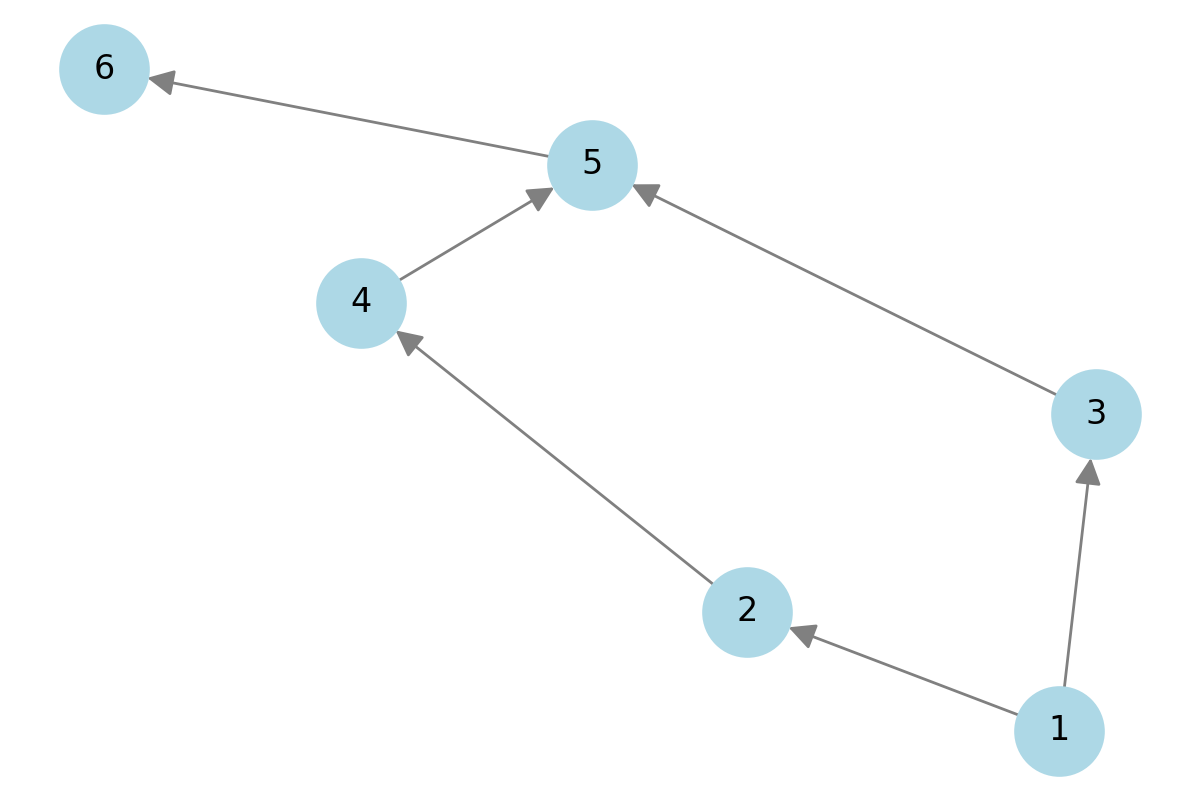


Рисунок 1 - Побудований граф

Розв'язання задачі за допомогою алгоритму DFS:

Для алгоритму DFS визначаємо три стани вершин:

* 0: не відвідана
* 1: в процесі обробки (вершина додана в стек викликів DFS)
* 2: повністю оброблена (вершина і всі її нащадки вже оброблені)

Представлення графа у вигляді списку суміжності:

1: [2, 3]

2: [4]

3: [5]

4: [5]

5: [6]

6: []

### **Крок 1: Запуск DFS з вершини 1:**

* Позначаємо вершину **1** у стані "в обробці" (**1**).
* Рекурсивно викликаємо DFS для сусідів вершини **1**:

#### **а) DFS(2):**

* Позначаємо вершину **2** у стані "в обробці" (**1**).
* Рекурсивно викликаємо DFS для сусідів вершини **2**:

##### **i) DFS(4):**

* Позначаємо вершину **4** у стані "в обробці" (**1**).
* Рекурсивно викликаємо DFS для сусідів вершини **4**:

###### **- DFS(5):**

* Позначаємо вершину **5** у стані "в обробці" (**1**).
* Рекурсивно викликаємо DFS для сусідів вершини **5**:

a) DFS(6):

* Позначаємо вершину **6** у стані "в обробці" (**1**).
* Вершина **6** не має сусідів.
* Позначаємо вершину **6** як "оброблену" (**2**) і додаємо в результат: **L = [6]**.

###### **- Позначаємо вершину 5 як "оброблену" (2) і додаємо в результат: L = [6, 5].**

##### **- Позначаємо вершину 4 як "оброблену" (2) і додаємо в результат: L = [6, 5, 4].**

#### **- Позначаємо вершину 2 як "оброблену" (2) і додаємо в результат: L = [6, 5, 4, 2].**

#### **б) DFS(3):**

* Позначаємо вершину **3** у стані "в обробці" (**1**).
* Рекурсивно викликаємо DFS для сусідів вершини **3**:

##### **i) DFS(5):**

* Вершина **5** вже оброблена, тому пропускаємо.

#### **- Позначаємо вершину 3 як "оброблену" (2) і додаємо в результат: L = [6, 5, 4, 2, 3].**

### **Крок 2: Завершення обходу вершини 1:**

* Позначаємо вершину **1** як "оброблену" (**2**) і додаємо в результат: **L = [6, 5, 4, 2, 3, 1]**.

### **Крок 3: Формування остаточного результату:**

* На виході маємо список у зворотному порядку: **L = [6, 5, 4, 2, 3, 1]**.
* Для отримання правильного топологічного сортування треба перевернути список:

### **Результуюче топологічне сортування:**

**L = [1, 3, 2, 4, 5, 6]** ""

**Відповіді на контрольні питання**

1. Які переваги і недоліки алгоритму Кана порівняно з алгоритмом DFS для топологічного сортування графа?

Алгоритм Кана зручний для виявлення циклів і добре підходить для потокової обробки. Недолік -- потребує додаткових структур (черга, масив ступенів).

DFS простіший у реалізації та не потребує додаткових структур, але складніший у виявленні циклів і потребує обробки після завершення обходу.

1. Яка складність часу і пам'яті для кожного з алгоритмів у найгіршому і найкращому випадках?

Для обох алгоритмів часова складність у найгіршому і найкращому випадках -- O(V + E).

Алгоритм Кана використовує додаткову пам'ять для зберігання ступенів вершин і черги (O(V)).

DFS використовує пам'ять для позначення відвіданих вершин і стек викликів (O(V)).

1. Чи можна застосовувати алгоритм Кана до графів з вагами на ребрах? Як це порівняти з DFS?

Так, обидва можуть працювати з графами, де є ваги на ребрах, але ігнорують самі ваги, бо топологічне сортування не враховує їх --- важливий лише напрям.

1. Як впливає структура графа на швидкість роботи кожного з цих алгоритмів?

У графах з великою кількістю зв'язків (густих) обидва працюють ефективно.

У графах з довгими ланцюгами DFS може бути ефективнішим через менші витрати на структури. Алгоритм Кана вимагає більше обчислень для ступенів.

1. Чи є обмеження використання кожного алгоритму для певних типів графів або завдань?

Обидва алгоритми працюють тільки з ациклічними орієнтованими графами (DAG).

Кан не працює без попередньо обчислених ступенів входу. DFS потребує обережності з порядком додавання у вихід.

1. Які варіанти оптимізації можна застосувати для кожного алгоритму з метою поліпшення його продуктивності

Для Кана можна використати пріоритетну чергу для контролю порядку сортування або ефективніше зберігати ступені.

Для DFS --- можна уникнути рекурсії через ітеративну реалізацію або використовувати бітові масиви для visited.