МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ МИХАЙЛА ОСТРОГРАДСЬКОГО

Навчалньно-науковий інститут електричної інженерії та інформаційних технологій

Кафедра автоматизації та інформаційних систем

**ПРАКТИЧНА РОБОТА**

Виконав: студент групи КІ-23-1

Бобров Євген

Перевірив:

Сидоренко Валерій Миколайович

м. Кременчук

2024 рік

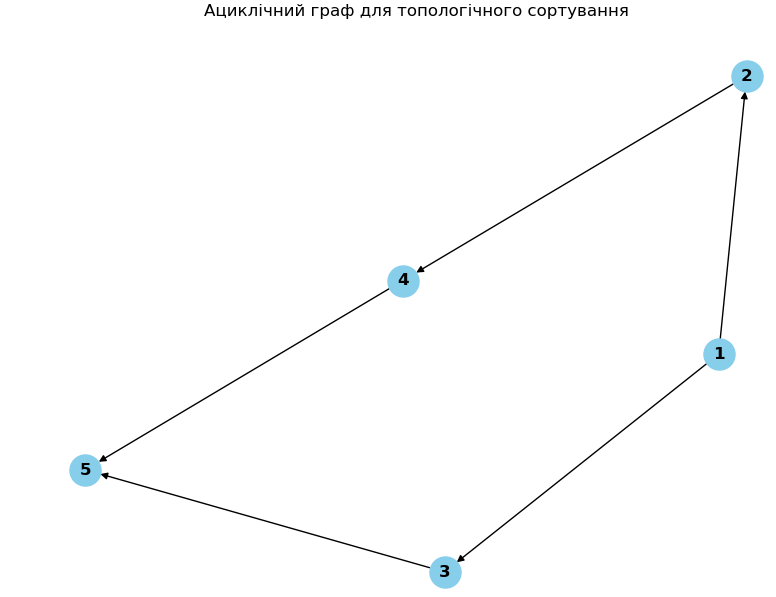
**Практична робота № 5**

**Тема**. **Графи. Ациклічні графи**

**Мета:** набути практичних навичок розв’язання задач топографічного сортування та оцінювання їх асимптотичної складності.

**Хід роботи**

2. Задано ациклічний граф: {1,2,3,4,5}{(1,2),(1,3),(2,4),(3,5),(4,5)}. Побудувати граф і розв’язати задачу топологічного сортування за допомогою алгоритму DFS.



from collections import defaultdict

# Реалізація графа

graph = {

1: [2, 3],

2: [4],

3: [5],

4: [5],

5: []

}

# Алгоритм DFS для топологічного сортування

def dfs\_topological\_sort(graph):

visited = set() # Множина відвіданих вершин

result = [] # Список для зберігання результату

def dfs(node):

if node in visited:

return

visited.add(node)

for neighbor in graph[node]:

dfs(neighbor)

result.append(node)

for vertex in graph:

if vertex not in visited:

dfs(vertex)

return result[::-1] # Зворотній порядок списку

# Викликаємо функцію для сортування

topological\_order = dfs\_topological\_sort(graph)

print("Топологічне сортування:", topological\_order)

Топологічне сортування: [1, 3, 2, 4, 5]

**Контрольні питання**

1. Переваги та недоліки алгоритмів Кана і DFS

Переваги алгоритму Кана:

* Легка реалізація: Простий у розумінні та реалізації, оскільки використовує лише чергу без потреби у рекурсії.
* Чітка робота з пріоритетами: Зручний для завдань, де потрібно враховувати порядок обробки вершин за мінімумом або максимумом.
* Менший ризик помилок: Відсутність рекурсії робить його більш надійним для обробки великих графів, уникнувши переповнення стека.

Недоліки алгоритму Кана:

* Додаткові ресурси пам’яті: Потребує використання черги, що може стати проблемою для великих графів.

Переваги алгоритму DFS:

* Мінімум пам’яті: Використовує рекурсію, що дозволяє уникнути додаткових структур даних, якщо оптимізувати стек викликів.
* Гнучкість: Легко модифікується для виконання додаткових задач, таких як пошук шляхів або виявлення циклів.

Недоліки алгоритму DFS:

* Складність реалізації: Вимагає акуратності у врахуванні рекурсії, станів вершин і роботи зі стеком.
* Переповнення стека: Може виникати в дуже великих графах із глибокими деревами.

2. Складність часу і пам’яті для кожного алгоритму

Алгоритм Кана:

* Часова складність:
  + Найгірший випадок: O(V + E), де VVV — кількість вершин, EEE — кількість ребер. Необхідно обробити всі вершини та ребра.
  + Найкращий випадок: O(V + E), адже навіть при мінімальній кількості операцій потрібно пройти всі вершини та ребра.
* Просторова складність:
  + O(V) — для збереження черги та масиву, який відстежує ступінь входу для кожної вершини.

Алгоритм DFS:

* Часова складність:
  + Найгірший випадок: O(V + E), оскільки необхідно відвідати кожну вершину та ребро.
  + Найкращий випадок: O(V + E), оскільки всі елементи обробляються у будь-якому випадку.
* Просторова складність:
  + O(V) — для збереження стану відвіданих вершин і рекурсивних викликів (глибина рекурсії може досягати VVV).

3. Чи можна застосовувати алгоритм Кана до графів із вагами на ребрах?

Алгоритм Кана не підтримує обробку графів із вагами на ребрах, оскільки орієнтований лише на ступені входу вершин. Для задач, які враховують ваги (наприклад, пошук найкоротшого шляху), потрібні модифікації алгоритму.

Для DFS топологічне сортування також не враховує ваги ребер. Проте алгоритм може бути модифікований для задач, які враховують ваги, таких як пошук критичного шляху.

4. Вплив структури графа на швидкість роботи алгоритмів

* Алгоритм Кана:  
  Працює швидше в графах із великою кількістю вершин із нульовим ступенем входу, оскільки черга буде швидше очищатися.
* DFS:  
  Ефективніший для графів із великою кількістю глибоких зв'язків, особливо якщо необхідні додаткові операції після обходу (наприклад, пошук компонент зв’язності).

5. Обмеження використання кожного алгоритму

Алгоритм Кана:

* Непридатний для графів із циклами, оскільки цикли унеможливлюють топологічне сортування.
* Призначений для орієнтованих ациклічних графів (DAG).

Алгоритм DFS:

* Не працює для графів із циклами під час виконання топологічного сортування (може використовуватися для виявлення циклів).
* Також орієнтований на DAG, але більш універсальний для інших завдань.

6. Варіанти оптимізації кожного алгоритму

Алгоритм Кана:

* Використання пріоритетної черги для обробки вершин із найменшою кількістю залежностей.
* Оптимізація структур даних для зменшення часу обробки ступенів входу та ребер.

Алгоритм DFS:

* Використання ітеративної заміни рекурсії (стек замість рекурсії) для уникнення переповнення стека.
* Раннє виявлення циклів і завершення обходу у разі їх знаходження.