МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ МИХАЙЛА ОСТРОГРАДСЬКОГО

Кафедра комп'ютерної інженерії та електроніки

ЗВІТ З ПРАКТИЧНИХ РОБІТ

з навчальної дисципліни
«Алгоритми та методи обчислень»

Тема «Графи. Ациклічні графи»

Студент гр. КІ-23-1 ПІБ Кобець О. О.

Кременчук 2024

Практична робота № 5

Тема. Графи. Ациклічні графи

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

Завдання

8. Задано ациклічний граф: {1,2,3,4,5,6,7,8,9}{(1,2),(1,3),(2,4),(3,5),(4,5),(4,6),(6,7),(7,8),(8,9)}. Побудувати граф і розв’язати задачу топологічного сортування за допомогою алгоритму DFS.

{1,2,3,4,5,6,7,8,9}

{(1,2),(1,3),(2,4),(3,5),(4,5),(4,6),(6,7),(7,8),(8,9)}

1: [2, 3]

2: [4]

3: [5]

4: [5, 6]

5: []

6: [7]

7: [8]

8: [9]

9: []

```python
def topological_sort_dfs(graph):  1 usage
    visited = set()
    stack = []

    def dfs(node):
        if node in visited:
            return
        visited.add(node)
        for neighbor in graph[node]:
            dfs(neighbor)
        stack.append(node)

    for node in graph:
        if node not in visited:
            dfs(node)

    stack.reverse()
    return stack

graph = {
    1: [2, 3],
    2: [4],
    3: [5],
    4: [5, 6],
    5: [],
    6: [7],
    7: [8],
    8: [9],
    9: []
}

topological_order = topological_sort_dfs(graph)
print("Топологічний порядок:", topological_order)
```

```
Топологічний порядок: [1, 3, 2, 4, 6, 7, 8, 9, 5]

Process finished with exit code 0
```

Контрольні питання

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

Алгоритм Кана:

- **Переваги:**
  - Прямолінійність: легко зрозуміти та реалізувати.
  - Придатний для обробки графа під час побудови: можна використовувати для послідовної обробки вершин без рекурсії.
- **Недоліки:**
  - Використовує додаткову пам'ять для зберігання списку з нульовим ступенем входження та черги.
  - Вимагає перевірки і оновлення ступенів входження для всіх сусідніх вершин.

Алгоритм DFS:

- **Переваги:**
  - Рекурсивна природа: легко інтегрується в рекурсивні програми.
  - Ефективність в плані пам'яті: використовує стек викликів для зберігання станів.
- **Недоліки:**
  - Складність розуміння: може бути складнішим для розуміння та налагодження.
  - Може призвести до переповнення стека при великій глибині рекурсії.

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

Алгоритм Кана:

- **Часова складність:** O(V+E), де V - кількість вершин, E - кількість ребер.
- **Просторова складність:** O(V), оскільки необхідно зберігати ступені входження для всіх вершин і додаткову чергу.

Алгоритм DFS:

- **Часова складність:** O(V+E).
- **Просторова складність:** O(V) для стека викликів і додатково O(V) для зберігання відвіданих вершин.

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

Алгоритм Кана:

- Можна застосовувати до графів з вагами на ребрах, оскільки ваги не впливають на топологічне сортування. Алгоритм працює тільки з структурою графа, а не з вагами.

Алгоритм DFS:

- Також можна застосовувати до графів з вагами на ребрах, оскільки ваги не впливають на хід алгоритму. DFS також розглядає тільки зв'язки між вершинами.

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

Для обох алгоритмів (Кана і DFS), складність визначається кількістю вершин і ребер, тому структурні характеристики графа впливають на швидкість роботи однаково.

У графах з великою кількістю ребер обидва алгоритми можуть виконуватися повільніше, але загальна часова складність залишається O(V+E).

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

Алгоритм Кана:

- Працює тільки для орієнтованих ациклічних графів (DAG).
- Не підходить для графів з циклічними залежностями.

Алгоритм DFS:

- Також працює тільки для орієнтованих ациклічних графів (DAG).
- Може бути неефективним для дуже глибоких графів через можливе переповнення стека.

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

Алгоритм Кана:

- Використання динамічних структур даних, таких як deque, для черги, щоб прискорити вставку та видалення.
- Паралельна обробка вузлів з нульовим ступенем входження, якщо це можливо.

Алгоритм DFS:

- Використання ітеративного підходу замість рекурсивного для уникнення переповнення стека.
- Оптимізація перевірки відвідуваних вузлів за допомогою бітових масок або інших ефективних структур даних для зменшення накладних витрат на пам'ять.