

**Звіт**  
**Лабораторна робота №3а,б з ООП**

**“Паралельне мультипоточне програмування”**

виконав студент групи ІПС-21  
Тесленко Назар

Для реалізації послідовних та паралельних алгоритмів для даної роботи була використана лаборторна робота з предмету “Алгоритми та складність” на тему:

### **Алгоритм Джонсона для розріджених графів (включає алгоритми Беллмана-Форда і Дейкстри).**

Повна умова завдання: Алгоритм Джонсона для розріджених графів (включає алгоритми Беллмана-Форда і Дейкстри). В алгоритмі Дейкстри використовуйте чергу пріоритетів. Ваги дуг задаються дійсними числами.

Мова реалізації: C++

#### **Алгоритм Беллмана-Форда**

Алгоритм Беллмана-Форда використовується для пошуку найкоротших шляхів від однієї стартової вершини до всіх інших у зваженому графі, який може містити ребра з від’ємною вагою. Він ітерує  $V-1$  раз (де  $V$  — кількість вершин), поступово оновлюючи відстані.

Складність:  $O(V \cdot E)$

#### **Алгоритм Дейкстри**

Алгоритм Дейкстри знаходить найкоротші шляхи від однієї вершини до всіх інших, але лише у графах без від’ємних ребер. Для ефективності використовує чергу пріоритетів.

Складність:  $O((V+E) \log V)$  для черги пріоритетів

#### **Алгоритм Джонсона**

Алгоритм Джонсона дозволяє знайти найкоротші шляхи між усіма парами вершин у зваженому графі, який може містити від’ємні ваги, але не має циклів від’ємної довжини.

Послідовність виконання:

1. Додає нову вершину, з’єднану нульовими ребрами з усіма іншими.

2. Застосовує алгоритм Беллмана-Форда, щоб знайти потенціали  $h(v)$  для перерахунку ваг.
3. Перераховує ваги ребер:  $\omega'(u, v) = \omega(u, v) + h(u) - h(v)$ .  
Тепер усі ваги невід'ємні.
4. Для кожної вершини запускає алгоритм Дейкстри на графі з новими вагами.

Складність:  $O(V * E + V^2 \log V)$

Чому Джонсон має використовуватись саме на розріджених графах:

У розріджених графах кількість ребер  $E$  набагато менша за максимум  $V^2$ .

- Загальна складність Джонсона з пріоритетною чергою:
  1. 1 раз Беллман-Форд:  $O(V * E)$
  2.  $V$  запусків Дейкстри: Кожен має складність  $O((V+E) \log V)$ ,  
тому загалом -  $O(V(V+E) \log V)$
  3. Отже складність:  $O(VE \log V + V^2 \log V)$

**Проаналізуємо для різних типів графів:**

- Розріджений граф ( $E \approx V$ )
  1. Беллман-Форд:  $O(V^2)$
  2. Дейкстра:  $O(V^2 \log V)$
  3. Загальна складність:  $O(V^2 \log V)$   
Що є ефективнішим за  $O(V^3)$  у Флойда-Воршелла
- Щільний граф ( $E \approx V^2$ )
  1. Беллман-Форд:  $O(V^3)$
  2. Дейкстра:  $O(V^3 \log V)$
  3. Загальна складність:  $O(V^3 \log V)$   
Майже те саме, що у Флойда-Воршелла  $O(V^3)$ , але складніше в реалізації.

Отже, Джонсон з пріоритетною чергою є оптимальним для розріджених графів, де кількість ребер значно менша за  $V^2$ , оскільки в такому випадку часова складність залишається квадратично-логарифмічною  $O(V^2 \log V)$

Проведемо тестування для перевірки ефективності паралелізації алгоритму Джонсона:

Створюватимемо розріджені графи для порівняння

Кількість вершин: 25

Кількість ребер: 100

	Мультипоточний, ms	Послідовний, ms
2	30	20
4	27	20
6	28	23
8	31	25

Кількість вершин: 150

Кількість ребер: 750

	Мультипоточний, ms	Послідовний, ms
2	152	151
4	217	157
6	265	149
8	302	155

При тестуванні на графах з невеликою кількістю вершин та ребер спостерігається варіативність результатів вимірювання часу виконання. Це зумовлено тим, що при малих об'ємах обчислень значну частку загального часу займають допоміжні операції, зокрема:

- ініціалізація структур даних;
- створення об'єктів;
- накладні витрати, пов'язані з системними викликами тощо.

У випадку паралельної реалізації ситуація ускладнюється: додаткові витрати на створення, запуск та синхронізацію потоків можуть перевищувати фактичний виграш від паралелізації. Через це при роботі з невеликими графами паралельна версія алгоритму може демонструвати нижчу або аналогічну продуктивність порівняно з послідовною.

## Протестуємо для більш великих графів

Кількість вершин: 1500

Кількість ребер: 2000

	Мультипоточний, ms	Послідовний, ms
2	313	346
4	431	361
6	550	363
8	611	355

Початкове покращення часу при використанні 2 потоків демонструє потенціал паралельного підходу на розріджених графах. Проте подальше збільшення кількості потоків не приводить до лінійного прискорення, а навпаки — викликає погіршення через системні обмеження та зростання витрат на синхронізацію.

Кількість вершин: 10000

Кількість ребер: 11000

	Мультипоточний, ms	Послідовний, ms
2	7874	9861
4	8456	9840
6	7942	9847
8	8641	9793

Кількість вершин: 15000

Кількість ребер: 15000

	Мультипоточний, ms	Послідовний, ms
2	10661	12926
4	7657	11812
6	6946	12421
8	7593	13521

### **Висновок:**

Було реалізовано алгоритм Джонсона для пошуку найкоротших шляхів між усіма парами вершин у зважених графах, включаючи як послідовну, так і паралельну версії. Проведене тестування на графах різного розміру та щільності дозволяє зробити наступні висновки:

- Для малих графів паралельна реалізація не демонструє приросту продуктивності через значні накладні витрати на створення і синхронізацію потоків.
- Для середніх графів вииграш від паралельності незначний або відсутній через співставну вартість допоміжних операцій і самого обчислення.
- Для великих розріджених графів мультипоточкова реалізація демонструє помітне покращення продуктивності, особливо при використанні 2–4 потоків. Подальше збільшення кількості потоків призводить до втрати ефективності через системні обмеження і зростання витрат на синхронізацію.