ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ІВАНА ФРАНКА
Факультет прикладної математики та інформатики

# Паралельні та розподілені обчислення
# ЛАБОРАТОРНА РОБОТА №6

**Тема: «Алгоритм Дейкстри».**

Виконала:
Ст. *Пелещак Вероніка*
ПМІ-35с

2025

**Тема:** Алгоритм Дейкстри

**Завдання:** Для зваженого графа G(V,F), де V={a0, a1,…an} – множина вершин (n –велике число), а F множина ребер між вершинами, використовуючи алгоритм Дейкстри, знайти найкоротший шлях між заданою вершиною a та усіма іншими. Для різної розмірності графів та довільного вузла a порахувати час виконання програми без потоків та при заданих k потоках розпаралелення.

## Опис програми:

### 1. generateRandomGraph:

Генерує випадковий неорієнтований зважений граф заданої кількості вершин та щільності.

- Формує список суміжності для графа.
- Для кожної пари вершин із певною ймовірністю (визначеною параметром щільності) створює ребро.
- Вага кожного ребра генерується випадково у межах від **1 до maxWeight**.
- Граф є неорієнтованим, тому ребра додаються в обидві сторони.

### 2. sequentialDijkstra:

Реалізує класичний послідовний алгоритм Дейкстри.

- Ініціалізує всі відстані нескінченністю, крім початкової вершини.
- На кожній ітерації вибирає непереглянуту вершину з мінімальною відстанню.

- Оновлює відстані до суміжних вершин, якщо знайдено коротший шлях.
- Алгоритм завершується, коли всі вершини відвідано або немає досяжних вершин.

### 3. parallelDijkstra:

Реалізує паралельну версію алгоритму Дейкстри з використанням **OpenMP**.

- Визначає кількість потоків для паралельного виконання.
- Пошук вершини з мінімальною відстанню виконується паралельно між потоками.
- Оновлення відстаней до суміжних вершин також розподіляється між потоками.
- Використовуються директиви **OpenMP**:
  - **#pragma omp parallel** — створення паралельної області;
  - **#pragma omp for** — розподіл ітерацій циклу між потоками;
  - **#pragma omp critical** — захист спільних змінних від одночасного доступу.
- Здійснюється синхронізація для уникнення конфліктів при оновленні масиву відстаней.

### 4. runExperiments:

Проводить серію експериментів для оцінки продуктивності алгоритмів.

- Генерує випадковий граф з певною кількістю вершин, щільністю та максимальною вагою.

- Виконує послідовний алгоритм і фіксує час виконання.
- Запускає паралельний алгоритм для різної кількості потоків **(2, 4, 8)**.
- Обчислює показники:
  - **Speedup** — прискорення відносно послідовного виконання;
  - **Efficiency** — ефективність використання потоків.
- Виводить результати для кожного експерименту.

**5. main:**

Є головною точкою входу програми.

Задає параметри експериментів: кількість вершин, щільність, максимальна вага.

- Викликає функцію **runExperiments()** для різних варіантів:
  - кількість вершин: **5000, 10000, 20000**;
  - щільність графа: **0.1, 0.5, 0.9**.
- Виводить результати часу виконання, прискорення та ефективності.

# Результати:

```
--- Граф з 5000 вузлами та щільністю 0.5 ---
Послідовний Дейкстра: 0.744117 сек
Паралельний Дейкстра (2 потоки): 0.712796 сек | Speedup = 1.04394 | Efficiency = 0.521971
Паралельний Дейкстра (4 потоки): 0.649513 сек | Speedup = 1.14565 | Efficiency = 0.286413
Паралельний Дейкстра (8 потоки): 0.78019 сек | Speedup = 0.953764 | Efficiency = 0.11922

--- Граф з 10000 вузлами та щільністю 0.5 ---
Послідовний Дейкстра: 2.95626 сек
Паралельний Дейкстра (2 потоки): 2.17133 сек | Speedup = 1.3615 | Efficiency = 0.680748
Паралельний Дейкстра (4 потоки): 1.91767 сек | Speedup = 1.54159 | Efficiency = 0.385397
Паралельний Дейкстра (8 потоки): 1.98287 сек | Speedup = 1.4909 | Efficiency = 0.186363

--- Граф з 20000 вузлами та щільністю 0.5 ---
Послідовний Дейкстра: 13.5079 сек
Паралельний Дейкстра (2 потоки): 8.5505 сек | Speedup = 1.57978 | Efficiency = 0.789888
Паралельний Дейкстра (4 потоки): 6.07209 сек | Speedup = 2.22458 | Efficiency = 0.556146
Паралельний Дейкстра (8 потоки): 5.79889 сек | Speedup = 2.32939 | Efficiency = 0.291173

--- Граф з 5000 вузлами та щільністю 0.1 ---
Послідовний Дейкстра: 0.573742 сек
Паралельний Дейкстра (2 потоки): 0.459319 сек | Speedup = 1.24911 | Efficiency = 0.624557
Паралельний Дейкстра (4 потоки): 0.568535 сек | Speedup = 1.00916 | Efficiency = 0.25229
Паралельний Дейкстра (8 потоки): 0.720587 сек | Speedup = 0.796215 | Efficiency = 0.0995268

--- Граф з 10000 вузлами та щільністю 0.1 ---
Послідовний Дейкстра: 2.17253 сек
Паралельний Дейкстра (2 потоки): 1.5304 сек | Speedup = 1.41959 | Efficiency = 0.709794
Паралельний Дейкстра (4 потоки): 1.56466 сек | Speedup = 1.3885 | Efficiency = 0.347125
Паралельний Дейкстра (8 потоки): 1.77672 сек | Speedup = 1.22278 | Efficiency = 0.152847
```

```
--- Граф з 20000 вузлами та щільністю 0.1 ---
Послідовний Дейкстра: 9.58484 сек
Паралельний Дейкстра (2 потоки): 6.58643 сек | Speedup = 1.45524 | Efficiency = 0.72762
Паралельний Дейкстра (4 потоки): 4.81958 сек | Speedup = 1.98873 | Efficiency = 0.497183
Паралельний Дейкстра (8 потоки): 4.81417 сек | Speedup = 1.99097 | Efficiency = 0.248871

--- Граф з 5000 вузлами та щільністю 0.9 ---
Послідовний Дейкстра: 1.02053 сек
Паралельний Дейкстра (2 потоки): 0.835152 сек | Speedup = 1.22197 | Efficiency = 0.610986
Паралельний Дейкстра (4 потоки): 0.769733 сек | Speedup = 1.32583 | Efficiency = 0.331457
Паралельний Дейкстра (8 потоки): 0.85474 сек | Speedup = 1.19397 | Efficiency = 0.149246

--- Граф з 10000 вузлами та щільністю 0.9 ---
Послідовний Дейкстра: 4.39275 сек
Паралельний Дейкстра (2 потоки): 2.95116 сек | Speedup = 1.48848 | Efficiency = 0.744242
Паралельний Дейкстра (4 потоки): 2.40451 сек | Speedup = 1.82688 | Efficiency = 0.456721
Паралельний Дейкстра (8 потоки): 2.33216 сек | Speedup = 1.88356 | Efficiency = 0.235445

--- Граф з 20000 вузлами та щільністю 0.9 ---
Послідовний Дейкстра: 15.9484 сек
Паралельний Дейкстра (2 потоки): 9.10683 сек | Speedup = 1.75126 | Efficiency = 0.875628
Паралельний Дейкстра (4 потоки): 6.43962 сек | Speedup = 2.47661 | Efficiency = 0.619151
Паралельний Дейкстра (8 потоки): 6.30528 сек | Speedup = 2.52937 | Efficiency = 0.316171
```