ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ІВАНА ФРАНКА

Факультет прикладної математики та інформатики

Кафедра програмування

Паралельні та розподілені обчислення

ЛАБОРАТОРНА РОБОТА №5

Тема: «Алгоритм Флойда»

Виконав:

Ст. *Лук'янчук Денис*

Група ПМІ-33

2025

Тема: «Алгоритм Флойда»

Мета роботи: Ознайомитися з принципами багатопоточності та синхронізації потоків, навчитися вимірювати час виконання послідовних та паралельних обчислень, розраховувати прискорення та ефективність роботи алгоритму для аналізу взаємодії потоків.

Теоретичний матеріал

Орієнтований зважений граф $G(V, F)$складається з:

- **V = {a0, a1, …, an}** – множина вершин,
- **F** – множина орієнтованих ребер, де кожне ребро має вагу $w(i, j)$.

Вага ребра $w(i, j)$показує «довжину» або «вартість» переходу від вершини i до вершини j. Якщо ребро відсутнє, його вага приймається як нескінченність (∞).

Алгоритм Флойда дозволяє знайти найкоротші шляхи між усіма парами вершин. Основна ідея алгоритму: для кожної проміжної вершини k перевіряємо, чи можна скоротити шлях між вершинами i та j через k:

$$\text{dist}[i, j] = \min(\text{dist}[i, j], \text{dist}[i, k] + \text{dist}[k, j])$$

- Алгоритм має кубічну складність $O(n^3)$.
- Можлива реалізація як **послідовно**, так і **паралельно**, розподіляючи обчислення по рядках матриці суміжності.

Паралельна реалізація через потоки:

- Матриця відстаней розбивається на частини, кожен потік обробляє свою частину рядків.
- Після завершення обробки всіх потоків відбувається синхронізація (Join()).
- Прискорення та ефективність визначаються формулами:

$$S_k = \frac{T_{\text{посл}}}{T_{\text{парал}}}, E_k = \frac{S_k}{k} \times 100\%$$

де k – кількість потоків.

Хід роботи

Під час виконання роботи я реалізував алгоритм Флойда двома способами: послідовно та паралельно через потоки.

Спершу я згенерував граф випадковим чином, при цьому ваги ребер задавалися випадковими числами, а відсутні ребра позначалися нескінченністю.

У послідовній версії алгоритму я обчислював відстані між вершинами по черзі, проходячи всі проміжні вершини і оновлюючи значення відстаней.

У паралельній версії я розділив матрицю відстаней на частини, і кожен потік обробляв свій блок рядків. Після обробки всіх потоків відбувалася синхронізація, після чого алгоритм переходив до наступної проміжної вершини.

Щоб перевірити продуктивність, я вимірював час виконання обох версій алгоритму, а також розраховував прискорення та ефективність паралельного виконання.

Для перевірки коректності я обрав певні вершини та обчислив найкоротший шлях між ними, порівнюючи результати між послідовною та паралельною версією.

Результати експерименту

- Для невеликого графа (n=100) послідовний алгоритм працює швидше, ніж паралельний через накладні витрати на потоки. А ефективність з прискорення зменшуються зі зростанням кількості задіяних потоків.

```
Граф 100x100
Вершини для пошуку шляху: a=12, b=78

Послідовно: 5,42 мс
Найкоротший шлях від 12 до 78: 91

Паралельно (4 потоків): 42,24 мс
Найкоротший шлях від 12 до 78: 91

Показники продуктивності
Прискорення (S4) = 0,128
Ефективність (E4) = 3,21%
```

```
Граф 100x100
Вершини для пошуку шляху: a=12, b=78

Послідовно: 5,83 мс
Найкоротший шлях від 12 до 78: 91

Паралельно (8 потоків): 72,15 мс
Найкоротший шлях від 12 до 78: 91

Показники продуктивності
Прискорення (S8) = 0,081
Ефективність (E8) = 1,01%
```

```
Граф 100x100
Вершини для пошуку шляху: a=12, b=78

Послідовно: 5,31 мс
Найкоротший шлях від 12 до 78: 91

Паралельно (16 потоків): 147,53 мс
Найкоротший шлях від 12 до 78: 91

Показники продуктивності
Прискорення (S16) = 0,036
Ефективність (E16) = 0,22%
```

- При збільшенні розміру графа до n=1000 паралельна версія з стала швидшою за послідовну. А ефективність з прискорення, аналогічно з першим прикладом, зменшуються зі зростанням кількості задіяних потоків.

```
Граф 1000x1000
Вершини для пошуку шляху: a=12, b=78

Послідовно: 4951,76 мс
Найкоротший шлях від 12 до 78: 22

Паралельно (4 потоків): 2361,95 мс
Найкоротший шлях від 12 до 78: 22

Показники продуктивності
Прискорення (S4) = 2,096
Ефективність (E4) = 52,41%
```

```
Граф 1000x1000
Вершини для пошуку шляху: a=12, b=78

Послідовно: 4687,71 мс
Найкоротший шлях від 12 до 78: 22

Паралельно (8 потоків): 2304,13 мс
Найкоротший шлях від 12 до 78: 22

Показники продуктивності
Прискорення (S8) = 2,034
Ефективність (E8) = 25,43%
```

```
Граф 1000x1000
Вершини для пошуку шляху: a=12, b=78

Послідовно: 4828,73 мс
Найкоротший шлях від 12 до 78: 22

Паралельно (16 потоків): 2720,27 мс
Найкоротший шлях від 12 до 78: 22

Показники продуктивності
Прискорення (S16) = 1,775
Ефективність (E16) = 11,09%
```

- Збільшення кількості потоків понад фізично доступні ядра призводить до зниження ефективності через накладні витрати на синхронізацію.

- Найефективнішим виявився варіант з помірною кількістю потоків, що відповідає кількості фізичних ядер.

Висновок: У лабораторній роботі я ознайомився з принципами багатопоточності та синхронізації потоків на прикладі алгоритму Флойда для знаходження найкоротших шляхів. Було реалізовано послідовне та паралельне обчислення, що дозволило оцінити прискорення та ефективність для графів різного розміру.

Потоки обробляли блоки рядків матриці відстаней і синхронізувалися після кожної проміжної вершини, що дало змогу побачити їхню взаємодію та накладні витрати. Аналіз роботи з різною кількістю потоків (4, 8, 16) показав, що для малих графів послідовне виконання ефективніше, а для більших оптимальна кількість потоків відповідає фізичним ядрам процесора.