ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ІВАНА ФРАНКА

Факультет прикладної математики та інформатики

Кафедра програмування

Паралельні та розподілені обчислення

ЛАБОРАТОРНА РОБОТА №7

Тема: «Алгоритм Прима»

Виконав:

Ст. *Лук'янчук Денис*

Група ПМІ-33

2025

Тема: «Алгоритм Прима»

Мета роботи: Ознайомитися з принципами багатопоточності та синхронізації потоків на прикладі алгоритму Прима для побудови мінімального кістякового дерева. Навчитися вимірювати час виконання послідовних і паралельних обчислень, а також розраховувати прискорення та ефективність алгоритму при різній кількості потоків.

Теоретичний матеріал

Зважений неорієнтований граф G(V, F) складається з:

- $V = \{a_0, a_1, \ldots, a_n\}$ — множина вершин,
- F — множина ребер, кожне з яких має вагу w(i, j).

Мінімальне кістякове дерево (MST) — це підграф зв'язного зваженого неорієнтованого графа, який містить усі вершини та має мінімальну сумарну вагу ребер без утворення циклів.

Метою **алгоритму Прима** є побудова мінімального кістякового дерева (MST), тобто підмножини ребер, які з'єднують усі вершини графа з мінімальною сумарною вагою без утворення циклів.

Алгоритм починається з довільної вершини a і поступово додає нові вершини, обираючи ребро з мінімальною вагою, що з'єднує вже включену вершину з невключеною.
Формально:

$$min(w(u,v)) \text{де} u \in MST,\ v \notin MST$$

Складність алгоритму Прима у базовій реалізації становить $O(V^2)$.

Хід роботи

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

Спершу я згенерував випадковий зважений неорієнтований граф, де ваги ребер задавалися числами від 5 до 100, а відсутні ребра позначалися нескінченністю. Для дослідження було обрано розмір графа n = 1000 і фіксовану кількість потоків k = 8. Початкова вершина a вибиралася випадковим чином.

У послідовній версії алгоритму відстані оновлювалися послідовно — на кожному кроці вибиралася вершина з мінімальною вагою ребра, яка приєднувалася до кістякового дерева, після чого оновлювалися ключі (ваги) для сусідніх вершин.

У паралельній версії оновлення ключів (key[v]) було розподілено між потоками — кожен потік обробляв свій блок вершин. Після завершення роботи всіх потоків

виконувалася синхронізація (Join()). Вибір вершини з мінімальним ключем залишався послідовним, тому повна паралельність алгоритму недосяжна.

Для оцінки продуктивності було виміряно час виконання обох версій, а також обчислено прискорення та ефективність.

Результати експерименту

Граф n = 100:

```
Граф 100×100
Стартова вершина a = 15
Кількість потоків: 4

Послідовно: 0,67 мс
Сума ваг MST = 873,00

Паралельно (4 потоків): 41,73 мс
Сума ваг MST = 873,00

Прискорення (S4) = 0,016
Ефективність (E4) = 0,40%
```

```
Граф 100×100
Стартова вершина a = 15
Кількість потоків: 8

Послідовно: 0,59 мс
Сума ваг MST = 896,00

Паралельно (8 потоків): 68,63 мс
Сума ваг MST = 896,00

Прискорення (S8) = 0,009
Ефективність (E8) = 0,11%
```

```
Граф 100×100
Стартова вершина a = 15
Кількість потоків: 16

Послідовно: 0,51 мс
Сума ваг MST = 961,00

Паралельно (16 потоків): 140,20 мс
Сума ваг MST = 961,00

Прискорення (S16) = 0,004
Ефективність (E16) = 0,02%
```

Послідовне виконання працює швидше, ніж паралельне через накладні витрати на створення потоків і блокування даних. Прискорення практично дорівнює нулю, ефективність падає при збільшенні k.

Граф n = 1000:

```
Граф 1000×1000
Стартова вершина a = 15
Кількість потоків: 4

Послідовно: 6,38 мс
Сума ваг MST = 5072,00

Паралельно (4 потоків): 500,52 мс
Сума ваг MST = 5072,00

Прискорення (S4) = 0,013
Ефективність (E4) = 0,32%
```

```
Граф 1000×1000
Стартова вершина a = 15
Кількість потоків: 8

Послідовно: 4,50 мс
Сума ваг MST = 5073,00

Паралельно (8 потоків): 730,44 мс
Сума ваг MST = 5073,00

Прискорення (S8) = 0,006
Ефективність (E8) = 0,08%
```

```
Граф 1000×1000
Стартова вершина a = 15
Кількість потоків: 16

Послідовно: 4,60 мс
Сума ваг MST = 5079,00

Паралельно (16 потоків): 1525,11 мс
Сума ваг MST = 5079,00

Прискорення (S16) = 0,003
Ефективність (E16) = 0,02%
```

Послідовний алгоритм залишається набагато швидшим. Паралельний варіант зі зростанням потоків демонструє ще більші накладні витрати, прискорення та ефективність продовжують падати.

Граф n = 10000:

```
Граф 10000×10000
Стартова вершина a = 1500
Кількість потоків: 4

Послідовно: 408,96 мс
Сума ваг MST = 49990,00

Паралельно (4 потоків): 4077,18 мс
Сума ваг MST = 49990,00

Прискорення (S4) = 0,100
Ефективність (E4) = 2,51%
```

```
Граф 10000×10000
Стартова вершина a = 1500
Кількість потоків: 8

Послідовно: 404,31 мс
Сума ваг MST = 49990,00

Паралельно (8 потоків): 7517,56 мс
Сума ваг MST = 49990,00

Прискорення (S8) = 0,054
Ефективність (E8) = 0,67%
```

```
Граф 10000×10000
Стартова вершина a = 1500
Кількість потоків: 16

Послідовно: 404,35 мс
Сума ваг MST = 49990,00

Паралельно (16 потоків): 15145,79 мс
Сума ваг MST = 49990,00

Прискорення (S16) = 0,027
Ефективність (E16) = 0,17%
```

Паралельна обробка через потоки показала мінімальне збільшення ефективності порівняно з попередніми прикладами. Прискорення сильно менше 1, швидкість виконання надалі сильно повільніша за послідовний.

Граф n = 25000:

Послідовний алгоритм Прима є швидшим ніж паралельний при n = 25000 кількості вершин графа. Паралельний варіант зі зростанням потоків демонструє ще більші накладні витрати, прискорення та ефективність продовжують падати при збільшені кількості потоків. Но чим більше кількість вершин у графі тим ефективність стає більша у великому графі в порівнянні з малим.

```
Граф 25000×25000
Стартова вершина a = 5500
Кількість потоків: 4

Послідовно: 4724,16 мс
Сума ваг MST = 124990,00

Паралельно (4 потоків): 13306,03 мс
Сума ваг MST = 124990,00

Прискорення (S4) = 0,355
Ефективність (E4) = 8,88%
```

```
Граф 25000×25000
Стартова вершина a = 5500
Кількість потоків: 8

Послідовно: 3001,03 мс
Сума ваг MST = 124990,00

Паралельно (8 потоків): 21405,25 мс
Сума ваг MST = 124990,00

Прискорення (S8) = 0,140
Ефективність (E8) = 1,75%
```

```
Граф 25000×25000
Стартова вершина a = 5500
Кількість потоків: 16

Послідовно: 2928,19 мс
Сума ваг MST = 124990,00

Паралельно (16 потоків): 41817,06 мс
Сума ваг MST = 124990,00

Прискорення (S16) = 0,070
Ефективність (E16) = 0,44%
```

Граф n = 65000:

```
Граф 65000×65000, стартова вершина a = 35348
Кількість потоків: 4

Послідовно: 121201,97 мс
Сума ваг MST = 324990,00

Паралельно (4 потоків): 115832,69 мс
Сума ваг MST = 324990,00

Прискорення (S4) = 1,046
Ефективність (E4) = 26,16%
```

```
Граф 65000×65000, стартова вершина a = 50878
Кількість потоків: 8

Послідовно: 155026,03 мс
Сума ваг MST = 324990,00

Паралельно (8 потоків): 257704,46 мс
Сума ваг MST = 324990,00

Прискорення (S8) = 0,602
Ефективність (E8) = 7,52%
```

```
Граф 65000×65000, стартова вершина a = 64198
Кількість потоків: 16

Послідовно: 116003,65 мс
Сума ваг MST = 324990,00

Паралельно (16 потоків): 206676,84 мс
Сума ваг MST = 324990,00

Прискорення (S16) = 0,561
Ефективність (E16) = 3,51%
```

При великій кількості вершин графа спостерігається, що паралельний метод нарешті перевищує за швидкодією послідовний при використанні 4 потоків. Однак подальше збільшення кількості потоків понад 4 призводить до зниження ефективності та прискорення через зростання накладних витрат на синхронізацію. Таким чином, оптимальною для даної розмірності графа є кількість потоків, що дорівнює чотирьом.

Висновки по потоках:

Для **малих графів (n = 100)** паралельне виконання є неефективним — час роботи зростає через накладні витрати на створення потоків і синхронізацію. Прискорення практично відсутнє, а ефективність зменшується при збільшенні кількості потоків.

При **середніх графах (n = 1000)** ситуація залишається подібною: послідовна версія працює швидше, а паралельна демонструє зростання часу виконання через додаткові витрати на координацію потоків.

Для **великих графів (n = 10000–25000)** спостерігається невелике покращення роботи паралельного методу, проте прискорення залишається меншим за одиницю. Це пояснюється тим, що приріст обчислень компенсується великими витратами на обмін даними між потоками. Водночас із ростом кількості вершин ефективність поступово підвищується порівняно з малими графами.

Лише при **дуже великій кількості вершин (n = 65000)** паралельний метод починає перевищувати послідовний за швидкодією при використанні 4 потоків. Проте подальше збільшення кількості потоків понад 4 призводить до зниження ефективності через зростання накладних витрат на синхронізацію.

Висновок: У лабораторній роботі я ознайомився з принципами багатопоточності та синхронізації потоків на прикладі алгоритму Прима для побудови мінімального кістякового дерева у зваженому неорієнтованому графі. Було реалізовано послідовне та паралельне обчислення, що дозволило оцінити прискорення та ефективність для графів різного розміру.

Результати експериментів показали, що послідовна версія працює швидше для малих і середніх графів, тоді як паралельна реалізація дає незначне покращення лише при великій кількості вершин і обмеженій кількості потоків. Подальше збільшення кількості потоків призводить до падіння ефективності через накладні витрати на синхронізацію.