Львівський національний університет імені Івана Франка

Факультет прикладної математики та інформатики

**Паралельні та розподілені обчислення**

**Лабораторна робота №6**

Виконав:

Студент групи ПМі-33

Тимчишин Ярема

Львів 2023

**Тема:** Алгоритм Дейкстри.

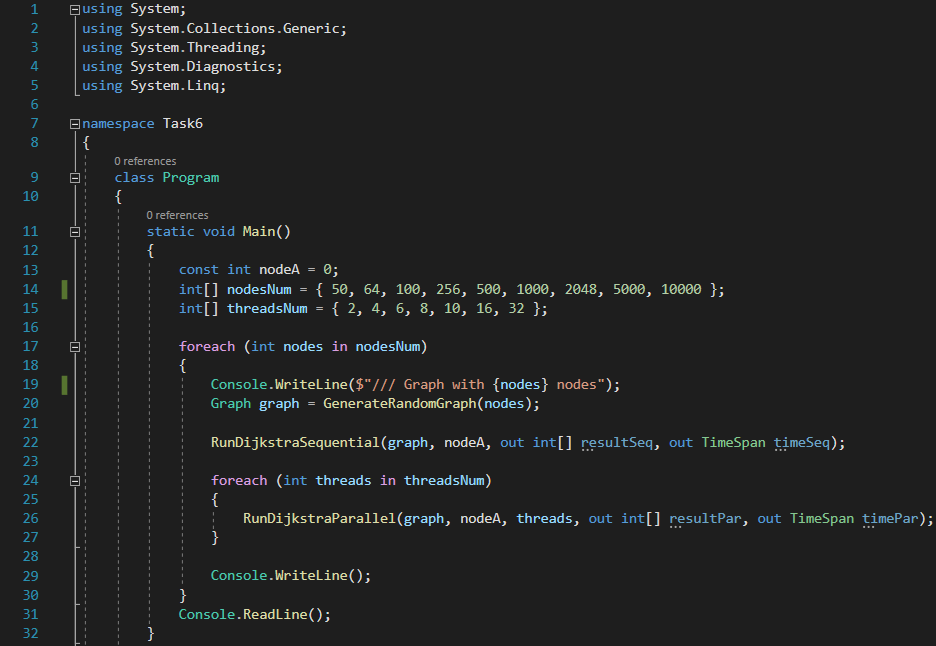
**Завдання:** Для зваженого графа G(V,F), де V={a0, a1,…an} – множина вершин (n –велике число), а F множина ребер між вершинами, використовуючи алгоритм Дейкстри, знайти найкоротший шлях між заданою вершиною a та усіма іншими.

Для різної розмірності графів та довільного вузла a порахувати час виконання програми без потоків та при заданих k потоках розпаралелення.

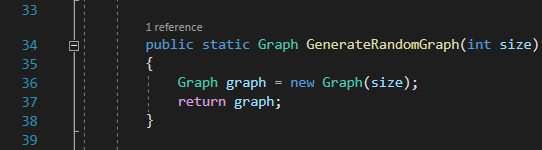
**Хід роботи**

Використовуючи мову програмування C# написав програму для знаходження найкоротшого шляху між заданою вершиною a та усіма іншими за допомогою розпаралелення методу Дейкстри.

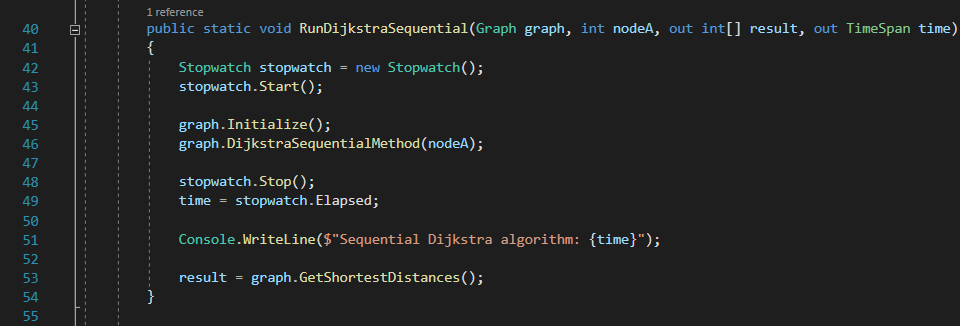
1. Метод **Main()**. Оголошення всіх змінних, виклик методу для генерації графу та виклик двох основних методів RunDijkstraSequential() та RunDijkstraParallel().



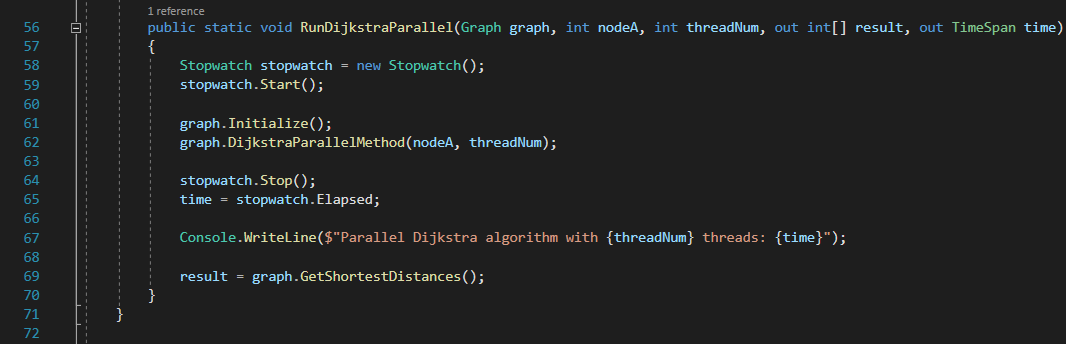
1. Метод **GenerateRandomGraph()**. Цей метод отримує кількість вершин, яка має бути у графі, після чого створює екземпляр класу Graph, який відповідатиме за представлення та роботу з графом, і потім повертає цей граф.



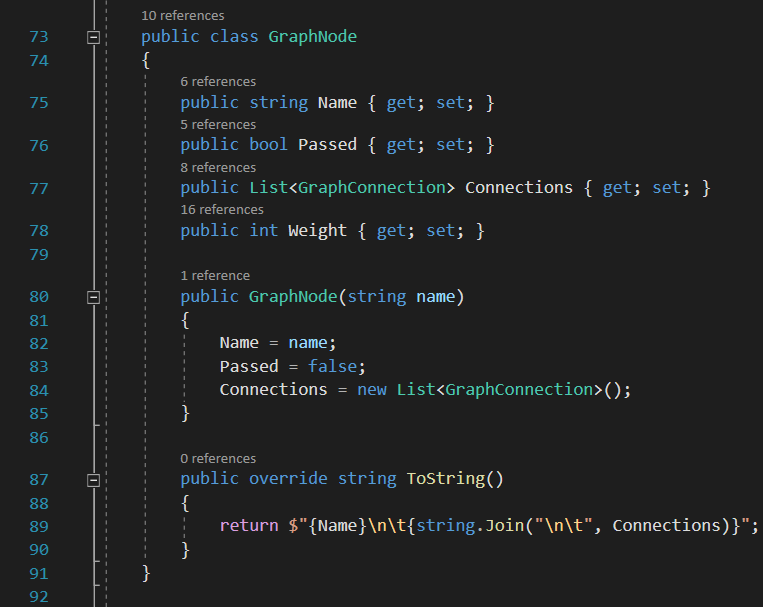
1. Метод **RunDijkstraSequential()**. Цей метод викликає послідовний алгоритм Дейкстри для заданого графа та для заданої вершини і виводить час виконання алгоритму.



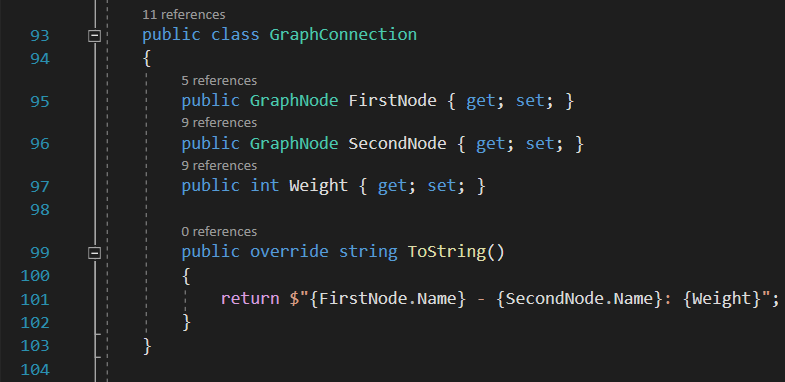
1. Метод **RunDijkstraParallel()**. Цей метод викликає паралельний алгоритм Дейкстри з різною кількістю потоків для заданого графа та для заданої вершини і виводить час виконання алгоритму з різною кількістю потоків.



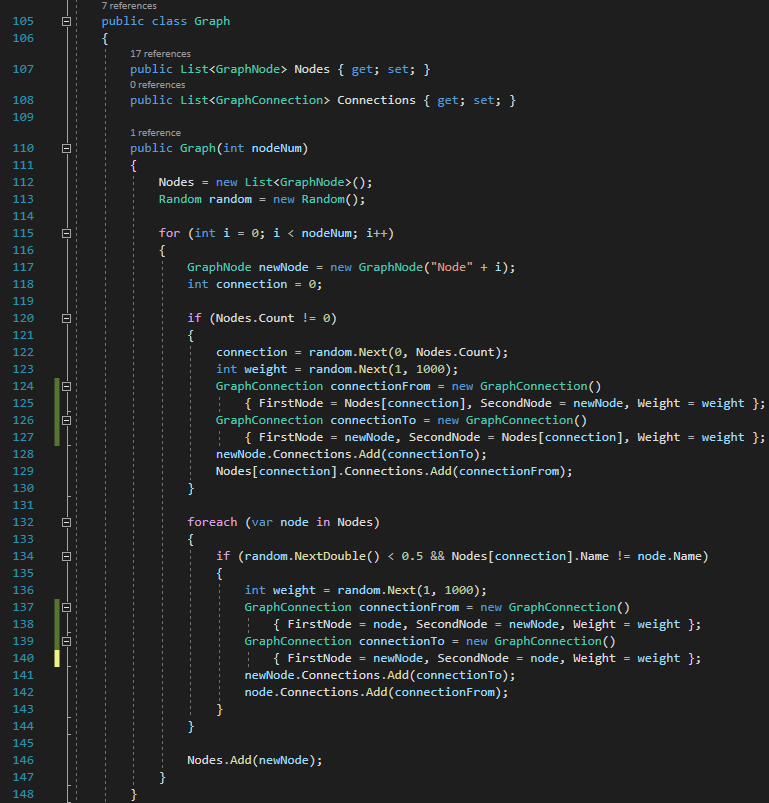
1. Клас **GraphNode**. Цей клас представляє вузол у графі. Детальніше розглянемо властивості та методи цього класу:
   * Властивість **Name**. Ця властивість представляє ім'я або ідентифікатор вершини графа.
   * Властивість **Passed**. Ця властивість є булевим прапорцем, який використовується в алгоритмах обходу графа для позначення того, чи була вершина відвідана під час обходу.
   * Властивість **Connections**. Ця властивість є списком об'єктів GraphConnection, що представляють зв'язки або ребра, які з'єднують цю вершину з іншими вершинами графа.
   * Властивість **Weight**. Ця властивість представляє вагу, пов'язану з вершиною.
   * Конструктор **GraphNode**. Цей конструктор приймає параметр name та ініціалізує властивості об'єкта GraphNode. Він встановлює надане ім'я у Name, ініціалізує Passed у false і створює порожній список для Connections.
   * Метод **ToString()**. Цей метод надає користувацьке рядкове представлення об'єкта GraphNode. Він повертає рядок, що містить ім'я вузла та відформатований список його з'єднань.



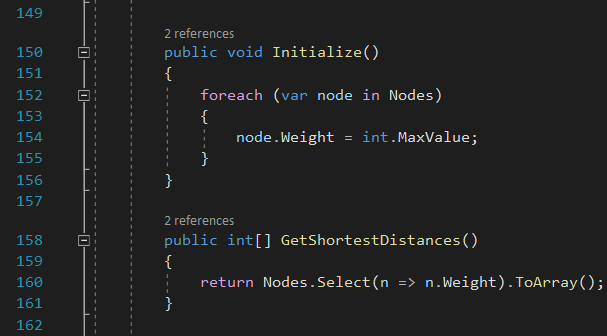
1. Клас **GraphConnection**. Цей клас представляє зв'язок або ребро між двома вершинами графа. Детальніше розглянемо властивості та методи цього класу:
   * Властивість **FirstNode**. Ця властивість представляє перший із з'єднаних вузлів.
   * Властивість **SecondNode**. Ця властивість представляє другий підключений вузол.
   * Властивість **Weight**. Ця властивість представляє вартість проходження ребра між двома вузлами.
   * Метод **ToString()**. Цей метод надає користувацьке рядкове представлення об'єкта GraphConnection. Він повертає рядок, який містить імена першої та другої вершин, з'єднаних цим ребром, а також вагу, пов'язану зі з'єднанням.



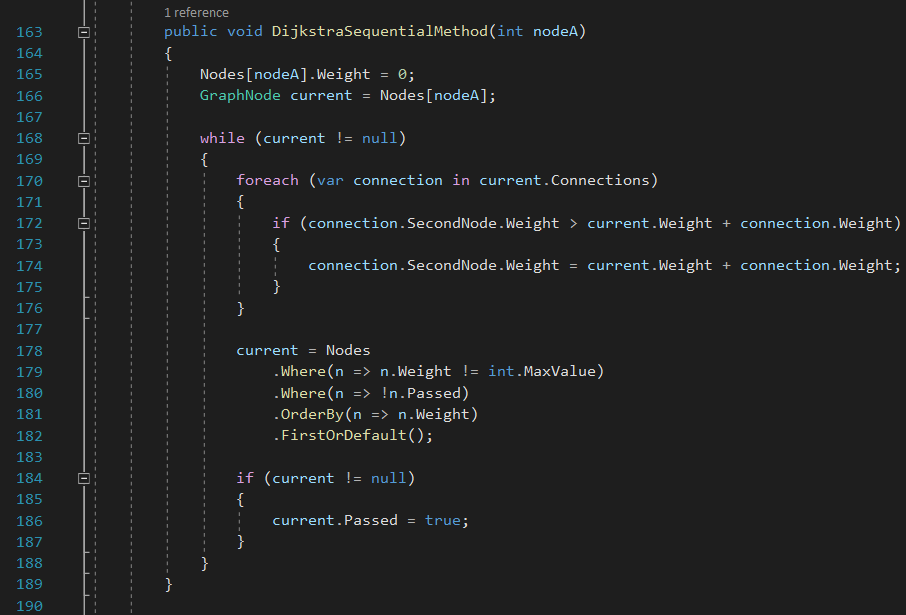
1. Клас **Graph**. Цей клас представляє структуру даних графа і містить методи для ініціалізації графа, знаходження найкоротших відстаней між вершинами за алгоритмом Дейкстри, а також виконання цього обчислення як послідовно, так і паралельно. Детальніше розглянемо властивості та методи цього класу:
   * Властивість **Nodes**. Ця властивість є списком, який містить екземпляри GraphNode, що представляють вузли графу.
   * Властивість **Connections**. Ця властивість є списком, що містить екземпляри GraphConnection, які представляють зв'язки або ребра між вершинами графа.
   * Конструктор **Graph**. Цей конструктор ініціалізує новий об'єкт Graph. Він приймає ціле число nodeNum як параметр, який визначає кількість вершин у графі. Всередині конструктора генерується випадковий граф з вказаною кількістю вершин. Граф будується шляхом створення вершин і випадкових зв'язків між ними.



* + Метод **Initialize()**. Цей метод відповідає за ініціалізацію властивості Weight кожної вершини графа максимальним значенням. Мета цього методу - підготувати граф для алгоритму пошуку найкоротших шляхів між вершинами. Встановивши для всіх ваг вершин максимальне значення, алгоритм може почати роботу з чистого аркуша, вважаючи всі вершини невідвіданими, а відстані - невідомими.
  + Метод **GetShortestDistances()**. Цей метод повертає масив цілих чисел, що містить найкоротші відстані від початкової заданої вершини до всіх інших вершин графа. Він робить це, вибираючи властивість Weight кожної вершини GraphNode у списку Nodes і перетворюючи їх у масив.



* + Метод **DijkstraSequentialMethod()**. Цей метод послідовно застосовує алгоритм Дейкстри для пошуку найкоротших шляхів від заданої початкової вершини до всіх інших вершин графа. Він оновлює ваги вершин і встановлює властивість Passed для позначення відвіданих вершин. Наприкінці цього методу властивість Weight кожної вершини графа буде містити найкоротшу відстань від заданої початкової вершини до цієї вершини, за умови, що між ними існує шлях.



* + Метод **DijkstraParallelMethod()**. Цей метод застосовує алгоритм Дейкстри з використанням заданої кількості потоків (threadNum). Він використовує пул потоків для паралельного виконання алгоритму, оновлюючи ваги вузлів та обробляючи синхронізацію між потоками за допомогою події CountdownEvent.



Результат роботи програми

