

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

КАФЕДРА ІНФОРМАТИКИ ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ

Курсова робота з освітнього компоненту

«Технології паралельних обчислень. Курсова робота»

Тема: Алгоритм Крускала та його паралельна реалізація на мові С#

|  |  |
| --- | --- |
| **Керівник**:  Асистент Дифучина О.Ю.  «Допущено до захисту»  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2023 р.  Захищено з оцінкою  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Члени комісії:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | **Виконавець**:  Галько М. В.  студент групи ІП-01  залікова книжка № 0107  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  «23» травня 2023 р.  Інна СТЕЦЕНКО  Ім’я ДИФУЧИНА |

**Київ – 2023**

**ЗАВДАННЯ**

1. Виконати огляд існуючих реалізацій алгоритму, послідовних та паралельних, з відповідними посиланнями на джерела інформації (статті, книги, електронні ресурси). Зробити висновок про актуальність дослідження.
2. Виконати розробку послідовного алгоритму у відповідності до варіанту завдання та обраного програмного забезпечення для реалізації. Дослідити швидкодію алгоритму при зростанні складності обчислень та зробити висновки про необхідність паралельної реалізації алгоритму.
3. Виконати розробку паралельного алгоритму у відповідності до варіанту завдання та обраного програмного забезпечення для реалізації. Забезпечити зручне введення даних для початку обчислень.
4. Виконати тестування алгоритму, що доводить коректність результатів обчислень.
5. Виконати дослідження швидкодії алгоритму при зростанні кількості даних для обчислень.
6. Виконати експериментальне дослідження прискорення розробленого алгоритму при зростанні кількості даних для обчислень. Реалізація алгоритму вважається успішною, якщо прискорення більше 1,2.
7. Зробити висновки про переваги паралельної реалізації обчислень для алгоритму, що розглядається у курсовій роботі, та програмних засобів, які використовувались.

**ЗМІСТ**

[ВСТУП 4](#_Toc136343910)

[1 ОПИС ПОСЛІДОВНОГО АЛГОРИТМУ ТА ЙОГО ВІДОМИХ ПАРАЛЕЛЬНИХ РЕАЛІЗАЦІЙ 5](#_Toc136343911)

[**1.1 Назва підрозділу** 5](#_Toc136343912)

[**1.2 Назва підрозділу** 5](#_Toc136343913)

[2 РОЗРОБКА ПОСЛІДОВНОГО АЛГОРИТМУ ТА АНАЛІЗ ЙОГО ШВИДКОДІЇ 6](#_Toc136343914)

[3 ВИБІР ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗРОБКИ ПАРАЛЕЛЬНИХ ОБЧИСЛЕНЬ ТА ЙОГО КОРОТКИЙ ОПИС 7](#_Toc136343915)

[4 РОЗРОБКА ПАРАЛЕЛЬНОГО АЛГОРИТМУ З ВИКОРИСТАННЯМ ОБРАНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ: ПРОЕКТУВАННЯ, РЕАЛІЗАЦІЯ, ТЕСТУВАННЯ 8](#_Toc136343916)

[5 ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ ПАРАЛЕЛЬНИХ ОБЧИСЛЕНЬ АЛГОРИТМУ 9](#_Toc136343917)

[ВИСНОВКИ 10](#_Toc136343918)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 11](#_Toc136343919)

[ДОДАТКИ 12](#_Toc136343920)

[**Додаток А. Нзава додатку** 12](#_Toc136343921)

[**Додаток Б. Назва додатку** 13](#_Toc136343922)

[**Додаток В. Назва додатку** 14](#_Toc136343923)

# **ВСТУП**

Паралельні обчислення є потужним інструментом, що дозволяє розподілити завдання на багато обчислювальних ресурсів та виконувати їх одночасно, прискорюючи обробку даних та забезпечуючи більш ефективне використання ресурсів. В сучасному світі, де обсяги даних ростуть експоненційно та вимагають швидкого аналізу, паралельні обчислення стають незамінним інструментом для досягнення високої продуктивності та відповіді на складні завдання.

В рамках даної курсової роботи ми розглядаємо алгоритм Крускала, який відноситься до класу алгоритмів мінімального остовного дерева. Його основна мета полягає у знаходженні мінімального остовного дерева в зваженому зв'язаному графі. Цей алгоритм має широкий спектр застосувань, зокрема в телекомунікаціях, транспортних системах, комп'ютерних мережах та інших областях, де важливо знайти оптимальні мережеві з'єднання або шляхи з мінімальною вартістю.

Паралельна реалізація алгоритму Крускала відкриває нові можливості для його ефективного виконання на сучасних багатоядерних архітектурах та розподілених системах. Розпаралелювання обчислень дозволяє використовувати потенціал паралельних обчислювальних ресурсів та зменшує час виконання алгоритму, що особливо важливо для великих графів та задач зі значною обчислювальною складністю.

При розробці паралельної версії алгоритму Крускала важливим аспектом є вибір технологій та підходів до паралельного програмування. Існує багато інструментів та бібліотек для реалізації паралельних обчислень у мові програмування C#, таких як TPL (Task Parallel Library), PLINQ (Parallel LINQ), а також власні розробки, що використовують пряме керування потоками або моделі акторів.

У даному звіті ми проведемо детальний аналіз алгоритму Крускала та його паралельних реалізацій з використанням різних програмних засобів. Описавши послідовний алгоритм та відомі паралельні реалізації, ми перейдемо до розробки власного послідовного алгоритму Крускала та проведемо аналіз його швидкодії. Ми детально проаналізуємо проектування, реалізацію та тестування цього алгоритму. Зосередимося на ефективності та продуктивності паралельних обчислень, проведемо дослідження й оцінку швидкості роботи алгоритму з різною кількістю обчислювальних ресурсів.

На основі проведених досліджень ми зробимо висновки про ефективність та переваги паралельної реалізації алгоритму Крускала.

Цей дослідницький звіт стане цінним джерелом інформації для розробників та дослідників, які цікавляться паралельними обчисленнями та алгоритмом Крускала.

# **1 ОПИС ПОСЛІДОВНОГО АЛГОРИТМУ ТА ЙОГО ВІДОМИХ ПАРАЛЕЛЬНИХ РЕАЛІЗАЦІЙ**

Алгоритм Крускала є ефективним алгоритмом для знаходження мінімального остовного дерева (МОД) в зваженому зв'язаному графі. Остовне дерево є підграфом, що включає всі вершини графу та є деревом (не має циклів), і його вага є найменшою серед всіх можливих остовних дерев графу.

Алгоритм Крускала базується на жадібному підході. Він починає з порожнього остовного дерева і поступово додає ребра до нього у порядку зростання їх ваги, забезпечуючи, щоб додавання кожного ребра не створило цикл у дереві. Алгоритм продовжує додавати ребра до остовного дерева, поки не буде включено всі вершини графу або поки не залишиться лише одне ребро.

Алгоритм Крускала є ефективним завдяки використанню сортування ребер за їх вагою та швидким перевіркам на цикли. Він може бути реалізований як послідовний алгоритм або паралельна версія, що використовує багатопоточність або розподілені обчислення для прискорення обчислень.

## **1.1 Послідовний алгоритм**

Алгоритм Крускала є ефективним способом знаходження мінімального остовного дерева в зваженому зв'язаному графі. Його послідовна реалізація включає такі кроки:

1. Ініціалізація: створення порожнього остовного дерева.
2. Сортування ребер: всі ребра графу сортуються за зростанням їх ваги.
3. Прохід по відсортованим ребрам: для кожного ребра з множини відсортованих ребер проводиться наступна перевірка:
   1. Якщо додавання поточного ребра до остовного дерева не створить цикл, ребро додається до остовного дерева.
   2. Якщо додавання ребра створить цикл, ребро відкидається.
4. Повторення кроку 3 до тих пір, поки остовне дерево не включить усі вершини графа або не залишиться лише одне ребро.
5. Завершення: Отримане остовне дерево є мінімальним остовним деревом графа.

Далі наведено псевдокод до послідовного алгоритму Крускала:

function Kruskal(graph):

create an empty minimum spanning tree (MST)

sort all the edges of the graph in non-decreasing order of their weights

for each vertex v in the graph:

create a disjoint set for v

for each edge (u, v) in the sorted edges:

if the sets containing u and v are different:

add the edge (u, v) to the MST

merge the sets containing u and v

return the minimum spanning tree (MST)

## **1.2 Відомі паралельні реалізації алгоритму**

Алгоритм Крускала також може бути реалізований у паралельному середовищі для прискорення обчислень. Ось дві відомі паралельні реалізації алгоритму Крускала:

Підхід 1: Паралельна сортування ребер

У цьому підході ребра графу сортуються за їх вагою з використанням алгоритму паралельного сортування. Після завершення сортування всіх ребер, можна продовжити виконання алгоритму Крускала за допомогою послідовного підходу.

Підхід 2: Фільтр-Крускал

У цьому підході ребра графу розділяються між різними процесорами або потоками, які паралельно перевіряють свої частини ребер на створення циклів і додають допустимі ребра до остовного дерева. Після завершення перевірки всіх частин ребер, потрібно об'єднати отримані остовні дерева в одне, враховуючи ваги ребер. Цей підхід може забезпечити більш паралельні обчислення, оскільки різні процесори можуть одночасно перевіряти ребра на створення циклів.

Використання паралельних підходів до реалізації алгоритму Крускала може прискорити обчислення та покращити ефективність алгоритму при роботі з великими графами або в розподілених обчислювальних середовищах.

# **2 РОЗРОБКА ПОСЛІДОВНОГО АЛГОРИТМУ ТА АНАЛІЗ ЙОГО ШВИДКОДІЇ**

# **3 ВИБІР ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗРОБКИ ПАРАЛЕЛЬНИХ ОБЧИСЛЕНЬ ТА ЙОГО КОРОТКИЙ ОПИС**

# **4 РОЗРОБКА ПАРАЛЕЛЬНОГО АЛГОРИТМУ З ВИКОРИСТАННЯМ ОБРАНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ: ПРОЕКТУВАННЯ, РЕАЛІЗАЦІЯ, ТЕСТУВАННЯ**

# **5 ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ ПАРАЛЕЛЬНИХ ОБЧИСЛЕНЬ АЛГОРИТМУ**

Таблиця 5.1. – Назва таблиці

|  |  |  |
| --- | --- | --- |
| Кількість елементів | Час паралельного алгоритму, мікросекунд | Час послідовного алгоритму, мікросекунд |
| 1000 | 2477 | 475 |
| 3000 | 2338 | 837 |

Рисунок 5.1. – Назва рисунку

Рисунок 5.3. – Результат виконання програми у консолі

# **ВИСНОВКИ**

# **СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. …...
2. …..
3. …..
4. ….
5. …..

# **ДОДАТКИ**

## **Додаток А. Нзава додатку**

## **Додаток Б. Назва додатку**

## **Додаток В. Назва додатку**