

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 пошуку в ширину та його паралельна реалізація мовою програмування Java

|  |  |
| --- | --- |
| **Керівник**:  асп. Дифучина Олександра Юріївна  «Допущено до захисту»  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2023 р.  Захищено з оцінкою  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Члени комісії:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | **Виконавець**:  Середюк Валентин Васильович  студент групи ІП-02  залікова книжка № ІП-0222  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  «23» травня 2023 р.  Інна СТЕЦЕНКО  Олександра ДИФУЧИНА |

**Київ – 2023**

**ЗАВДАННЯ**

1. Виконати огляд існуючих реалізацій алгоритму пошуку в ширину, послідовних та паралельних, з відповідними посиланнями на джерела інформації (статті, книги, електронні ресурси). Зробити висновок про актуальність дослідження.
2. Виконати розробку послідовного алгоритму пошуку в ширину та реалізувати мовою програмування Java. Дослідити швидкодію алгоритму при зростанні складності обчислень та зробити висновки про необхідність паралельної реалізації алгоритму.
3. Виконати розробку паралельного алгоритму пошуку в ширину та реалізувати мовою програмування Java. Забезпечити зручне введення даних для початку обчислень.
4. Виконати тестування алгоритму пошуку в ширину, що доводить коректність результатів обчислень.
5. Виконати дослідження швидкодії алгоритму пошуку в ширину при зростанні кількості даних для обчислень.
6. Виконати експериментальне дослідження прискорення розробленого алгоритму пошуку в ширину 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 пошуку в ширину та програмних засобів, які використовувались.

**АНОТАЦІЯ**

У даній роботі досліджується алгоритм пошуку в ширину (BFS) та його паралельна реалізація мовою програмування Java. Метою роботи є оцінка швидкодії паралельної версії алгоритму порівняно з послідовною. Буде проводиться огляд алгоритму BFS, його принципу роботи. Далі розглядатиметься паралельна реалізація BFS, визначаться основні кроки та способи розділення простору пошуку між потоками. Алгоритм BFS буде представлено мовою програмування Java. Також буде демонструватися використання потоків або пулу потоків у Java для покращення продуктивності BFS.

Буде проводитися оцінка продуктивності та оптимізація паралельної версії алгоритму. Вимірюватиметься час виконання паралельної версії у порівнянні з послідовною, будуть наведені графіки порівняння двох алгоритмів. Очікується, що паралельна реалізація дозволить прискорити час роботи в порівнянні з послідовною реалізацією мінімум в 1,2 раза.

**ЗМІСТ**

[ВСТУП 5](#_Toc135567388)

[1 ОПИС ПОСЛІДОВНОГО АЛГОРИТМУ ПОШУКУ В ШИРИНУ ТА ЙОГО ВІДОМИХ ПАРАЛЕЛЬНИХ РЕАЛІЗАЦІЙ 6](#_Toc135567389)

[**1.1 Назва підрозділу** 6](#_Toc135567390)

[**1.2 Назва підрозділу** 6](#_Toc135567391)

[2 РОЗРОБКА ПОСЛІДОВНОГО АЛГОРИТМУ ТА АНАЛІЗ ЙОГО ШВИДКОДІЇ 7](#_Toc135567392)

[3 ВИБІР ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗРОБКИ ПАРАЛЕЛЬНИХ ОБЧИСЛЕНЬ ТА ЙОГО КОРОТКИЙ ОПИС 8](#_Toc135567393)

[4 РОЗРОБКА ПАРАЛЕЛЬНОГО АЛГОРИТМУ З ВИКОРИСТАННЯМ ОБРАНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ: ПРОЕКТУВАННЯ, РЕАЛІЗАЦІЯ, ТЕСТУВАННЯ 9](#_Toc135567394)

[5 ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ ПАРАЛЕЛЬНИХ ОБЧИСЛЕНЬ АЛГОРИТМУ 10](#_Toc135567395)

[ВИСНОВКИ 11](#_Toc135567396)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 12](#_Toc135567397)

[ДОДАТКИ 13](#_Toc135567398)

[**Додаток А. Нзава додатку** 13](#_Toc135567399)

[**Додаток Б. Назва додатку** 14](#_Toc135567400)

[**Додаток В. Назва додатку** 15](#_Toc135567401)

# **ВСТУП**

У сучасному світі, де обчислювальні завдання стають все більш складними і об'ємними, паралельні обчислення набувають особливої актуальності. З кожним роком технології розвиваються швидкими темпами, дозволяючи реалізувати паралельність на все більш широкій шкалі. Такі області, як штучний інтелект, обробка великих об'ємів даних, комп'ютерне зору, наукові дослідження та інші, потребують швидкого та ефективного вирішення складних завдань. Однією з основних тенденцій у розвитку паралельних обчислень є зростання кількості ядер та потоків в процесорах. Замість того, щоб прискорювати окремі ядра до надзвичайно високих швидкостей, виробники процесорів почали ставити на акцент на збільшення кількості ядер, що дозволяє виконувати більше завдань паралельно. Це відкриває шлях до ефективного використання паралельних обчислень для прискорення обчислювальних процесів.

Одним з найбільш популярних алгоритмів, що використовуються в різних областях обчислювальної науки, є алгоритм пошуку в ширину (BFS – Breadth-First Search). Він використовується для пошуку найкоротшого шляху між двома вузлами графа, перебираючи сусідні вузли на кожному рівні глибини перед переходом до наступного рівня. Завдяки зростанню потужності сучасних обчислювальних систем, реалізація алгоритму BFS у паралельній середовищі стає важливою задачею. Паралельні обчислення дозволяють розподілити завдання між багатьма процесорами або ядрами, що дозволяє прискорити обчислення та забезпечити ефективне використання обчислювальних ресурсів.

Мова програмування Java, з її широким спектром бібліотек та засобів, є однією з найпоширеніших мов для розробки паралельних програм. Вона надає розширені можливості для створення багатопоточних додатків та ефективного використання паралельних обчислень.

# **1 ОПИС ПОСЛІДОВНОГО АЛГОРИТМУ ПОШУКУ В ШИРИНУ ТА ЙОГО ВІДОМИХ ПАРАЛЕЛЬНИХ РЕАЛІЗАЦІЙ**

## **Загальна інформація**

Алгоритм пошуку в ширину (BFS, Breadth-First Search) є одним з базових алгоритмів графового пошуку. Він дозволяє обходити всі вершини графа, починаючи з заданої стартової вершини, і відшукати найкоротші шляхи до всіх досяжних вершин. Алгоритм гарантує знаходження найкоротшого шляху до будь-якої досяжної вершини, при умові, що всі ребра мають однакову вагу. Він також може бути використаний для пошуку найкоротшого шляху між двома вершинами.

Складність алгоритму пошуку в ширину залежить від кількості вершин (n) та ребер (m) у графі. У найгіршому випадку, коли граф є повністю зв'язним, складність алгоритму можна оцінити як O(n + m), оскільки кожна вершина та кожне ребро відвідується рівно один раз. У випадку розріджених графів, де кількість ребер (m) набагато менша за кількість вершин (n), складність алгоритму можна приблизно оцінити як O(n), оскільки n >> m.

Алгоритм пошуку в ширину є ефективним і широко використовується у багатьох областях, включаючи графові структури даних, алгоритми маршрутизації в комп'ютерних мережах, графічні системи та штучний інтелект.

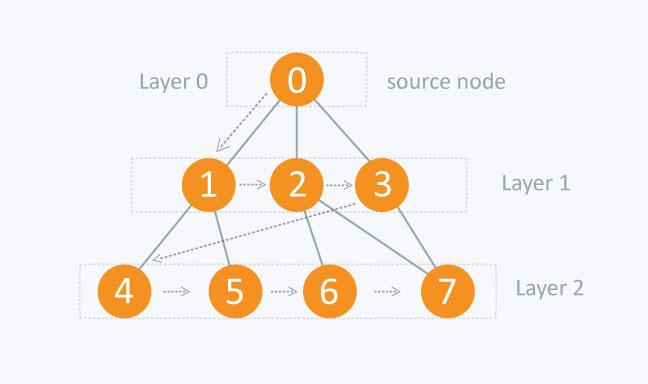
* 1. **Реалізація послідовного алгоритму**

Алгоритм пошуку в ширину (BFS) працює на основі принципу "перебування у шарах". Він починає з заданої стартової вершини і розглядає всі сусідні вершини цієї стартової вершини перед переходом до сусідніх вершин наступного шару.

Опис роботи алгоритму:

1. Початкова вершина ставиться у чергу.
2. Позначаємо початкову вершину як відвідану.
3. Поки черга не порожня:
   1. Беремо вершину з початку черги.
   2. Перевіряємо всіх сусідніх вершин для поточної вершини, які ще не відвідані.
   3. Додаємо сусідні вершини до кінця черги та позначайте їх як відвідані.
4. Повторюємо крок 3 до тих пір, поки черга не порожня.

Алгоритм продовжує просуватися шар за шаром по графу, знаходячи всі вершини на поточному шарі, перш ніж переходити до наступного шару (рис 1.1).

  
Рисунок 1.1 – Приклад обходу графа за допомогою BFS

Це гарантує знаходження найкоротшого шляху від початкової вершини до всіх досяжних вершин. Процес продовжується до тих пір, поки не будуть відвідані всі досяжні вершини або досягнутий певний критерій завершення. Наприклад, відвідані всі вершини, знайдена шукана вершина або досягнуто певної глибини пошуку.

* 1. **Реалізація послідовного алгоритму**

Для кращого розуміння, зобразимо роботу алгоритму у вигляді блок-схеми (рис 1.2).

Зображення, що містить текст, квитанція, чорно-білий, схема

Автоматично згенерований опис  
Рисунок 1.2 – Блок-схема роботи послідовного алгоритму

# **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**