

МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

КАФЕДРА ІНФОРМАТИКИ ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ

Курсова робота

з дисципліни «Технології паралельних обчислень»

Тема: Розробка мультипотокового Алгоритму Minimax та порівняння з його базовою версією

|  |  |
| --- | --- |
| **Керівник**:  Дифучина О.Ю.  «Допущено до захисту»  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2023 р.  Захищено з оцінкою  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Члени комісії:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | **Виконавець**:  Терешкович М. О.  студент групи ІТ–02  залікова книжка № ІТ–0223  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  «23» травня 2023 р.  Інна СТЕЦЕНКО  Олександра Дифучина |

**Київ – 2023**

**Міністерство освіти та науки України**

**Національний технічний університет України**

**«Київський політехнічний інститут імені Ігоря Сікорського»**

Кафедра Інформатики та програмної інженерії

Дисципліна Технології паралельних обчислень

Спеціальність Інженерія програмного забезпечення

Курс ІII Група IT-02 Семестр 6

**ЗАВДАННЯ**

**на курсову роботу студента**

Терешковича Максима Олександровича

(прізвище, ім’я, по батькові)

**1.** **Тема роботи** Розробка мультипотокового Алгоритму Minimax та порівняння з його базовою версією реалізований мовою С#.

**2.** **Термін здачі студентом закінченої роботи** « 23 » травня 2023 р.

**4. Зміст розрахунково-пояснювальної записки** (перелік питань, які підлягають розробці): зміст, реферат, ключові слова, вступ, розділ 1 (опис алгоритму та його відомих паралельних реалізацій), розділ 2 (розробка послідовного алгоритму та аналіз його швидкодії), розділ 3 (вибір програмного забезпечення для розробки паралельних обчислень та його короткий опис), розділ 4 (розробка паралельних обчислень алгоритму з використанням обраного програмного забезпечення: проектування, реалізація, тестування), розділ 5 (дослідження ефективності паралельних обчислень алгоритму (порівняльний аналіз швидкості обчислень)), висновки, список використаних джерел, додатки (результати тестування послідовного алгоритму, повний лістинг програми, результати тестування паралельного алгоритму).

**5. Перелік графічного матеріалу** (з точним зазначенням обов’язкових креслень)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**РЕФЕРАТ**

В рамках курсової роботи був спроектований та реалізований базовий алгоритм Мінімакс. Базуючись на базовому алгоритмі була розроблена модифікація даного алгоритму використовуючи механізм мультипотокового програмування.

Було виконане тестування паралельного алгоритму щоб довести коректність результатів обчислень. Було виконане дослідження швидкодії алгоритму при зростанні кількості даних для обчислень. А також в рамках роботи виконане експериментальне дослідження прискорення розробленого алгоритму при зростанні кількості даних для обчислень.

Для реалізації алгоритму була обрана мова С#. Вона є потужною інструментальною платформою, яка надає розробникам доступ до багатьох інструментів та бібліотек для створення високоякісних додатків. C# є однією з найбільш популярних мов програмування для розробки ігор, тому вона може бути хорошим вибором для реалізації алгоритму Мінімакс для ігор.

Переваги мови C# при реалізації алгоритму Мінімакс можуть включати:

1. Об'єктно-орієнтований підхід: C# є мовою, що підтримує об'єктно-орієнтований підхід, що дозволяє більш ефективно працювати зі складними об'єктами, які використовуються в алгоритмі Мінімакс.
2. Бібліотеки: C# має вбудовані бібліотеки, такі як System.Collections.Generic, які дозволяють працювати зі списками, стеками та іншими колекціями даних, що можуть бути корисними для реалізації алгоритму Мінімакс.
3. Простота: C# має простий синтаксис, який дозволяє швидко розробляти програми. Це може бути особливо корисним, коли потрібно розробити алгоритм Мінімакс для невеликих ігор.

Курсова робото проводилася на ноутбуці Asus TUF Gaming A17 FA706IU-H7023 (90NR03K1-M02580) Fortress Gray. Основні характеристики:

* Процесор - Восьмиядерний AMD Ryzen 9 4900H (3.3 - 4.4 ГГц)
* Об'єм оперативної пам'яті - 16 ГБ
* Тип оперативної пам'яті - DDR4
* Характеристики оперативної пам'яті - DDR4-3200 МГц

**ЗМІСТ**

[**ВСТУП 5**](#_gjdgxs)

[**1 ОПИС ПОСЛІДОВНОГО АЛГОРИТМУ ТА ЙОГО ВІДОМИХ ПАРАЛЕЛЬНИХ РЕАЛІЗАЦІЙ 7**](#_30j0zll)

[**1.1 Цілі та загальна характеристика 7**](#_1fob9te)

[**1.2 Розрахунок характеристик 8**](#_n1dl8ll0b77n)

[**1.3 Паралельна версія алгоритму 9**](#_bxiguhtld67l)

[**2 РОЗРОБКА ПОСЛІДОВНОГО АЛГОРИТМУ ТА АНАЛІЗ ЙОГО ШВИДКОДІЇ 10**](#_2et92p0)

[**3 ВИБІР ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗРОБКИ ПАРАЛЕЛЬНИХ ОБЧИСЛЕНЬ ТА ЙОГО КОРОТКИЙ ОПИС 11**](#_tyjcwt)

[**4 РОЗРОБКА ПАРАЛЕЛЬНОГО АЛГОРИТМУ З ВИКОРИСТАННЯМ ОБРАНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ: ПРОЕКТУВАННЯ, РЕАЛІЗАЦІЯ, ТЕСТУВАННЯ 12**](#_3dy6vkm)

[**5 ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ ПАРАЛЕЛЬНИХ ОБЧИСЛЕНЬ АЛГОРИТМУ 13**](#_1t3h5sf)

[**ВИСНОВКИ 14**](#_4d34og8)

[**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 15**](#_2s8eyo1)

[**ДОДАТКИ 16**](#_17dp8vu)

[**Додаток А. Нзава додатку 16**](#_3rdcrjn)

[**Додаток Б. Назва додатку 17**](#_26in1rg)

[**Додаток В. Назва додатку 18**](#_lnxbz9)

# **ВСТУП**

Розробка ефективних алгоритмів для вирішення складних завдань є однією з ключових задач сучасної інформаційної технології. Алгоритм Мінімакс - це один з найбільш популярних алгоритмів, який використовується для прийняття рішень в іграх, таких як шахи, го, хрестики нулики та інші.

Однак, якщо обчислювати всі можливі ходи та їх наслідки в грі, це може зайняти дуже багато часу, тому розробка ефективного алгоритму для прийняття рішень є важливим завданням для інформаційних технологій.

В цьому рефераті розглядається розробка та порівняння базового та мультипотокового алгоритму Мінімакс на мові програмування С#. Базовий алгоритм Мінімакс буде реалізовано без використання мультипотоковості, а мультипотоковий алгоритм буде розроблено з використанням паралельного програмування.

Розглянемо різні підходи до реалізації алгоритму Мінімакс та порівняємо їх ефективність. Основним критерієм оцінки роботи алгоритма та успішності виконаної роботи буде «Прискорення». Мультипотокова версія повинна бути мінімум в 1.2 швидша за базову але ,як відомо, при невеликому об’ємі даних базовий алгоритм завжди буде швидшим – тому потрібно буде визначити основні параметри та значення при яких дана умова буде виконуватися успішно.

Метою цього дослідження є розробка більш ефективного алгоритму Мінімакс з використанням мультипотоковості та порівняння його з базовим алгоритмом без мультипотоковості для визначення його переваг та недоліків

Основні завдання, які слід виконати в курсовій роботі для розробки та порівняння базового та мультипотокового алгоритму Мінімакс:

1. Розробити базовий алгоритм Мінімакс на мові С#.
2. Розробити мультипотоковий алгоритм Мінімакс на мові С# з використанням паралельного програмування.
3. Провести тестування розроблених алгоритмів на різних даних для порівняння їх ефективності.
4. Дати висновки про переваги та недоліки кожного алгоритму та порівняти їх ефективність.

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

## **Цілі та загальна характеристика**

Основною метою послідовного алгоритму Мінімакс є знаходження найкращого ходу у грі з двома гравцями, де кожен гравець старається максимізувати свій виграш, або мінімізувати виграш супротивника.

Загальні характеристики послідовного алгоритму Мінімакс можуть бути наступними:

1. Послідовний алгоритм Мінімакс вимагає перебору всіх можливих ходів у грі, що може зайняти значну кількість часу.
2. Для кожного можливого ходу у грі, алгоритм розраховує очікуваний виграш для гравця, який веде гру, за умови, що обидва гравці грають оптимально.
3. Алгоритм обчислює дерево гри, де кожен вузол представляє можливу позицію у грі, а кожна дуга виходить з вузла і представляє можливий хід гравця.
4. Після розрахунку очікуваного виграшу для кожного можливого ходу, алгоритм вибирає хід, що максимізує очікуваний виграш для гравця, який веде гру.
5. За умови, що гра має скінченну кількість ходів та ігри на дошці не має неперервних стратегій, послідовний алгоритм Мінімакс завжди знайде оптимальний хід для гравця, який веде гру.

Отже, ціллю послідовного алгоритму Мінімакс є знаходження оптимального ходу у грі з двома гравцями, де кожен гравець старається максимізувати свій виграш.

## **Розрахунок характеристик**

Для того, щоб Мінімакс міг приймати рішення про хід у грі, необхідно розрахувати певні характеристики (або оцінки) кожного стану гри. Наприклад, в шахах можна використовувати кількість фігур на дошці, позиційні переваги, розташування короля, тощо. У кожній грі характеристики можуть бути різними.

Розрахунок характеристик зазвичай відбувається на основі евристичних правил, тобто на основі досвіду та інтуїції програміста або експерта в грі. Наприклад, у грі в хрестики-нулики можна використовувати такі правила:

* Кожен вільний кут на дошці збільшує шанс перемоги гравця Х.
* Кожен рядок, колонка або діагональ з двома фігурами гравця Х і вільною клітинкою збільшує його шанс на перемогу.
* Кожен рядок, колонка або діагональ з двома фігурами гравця О і вільною клітинкою зменшує шанси гравця Х на перемогу.

Зазвичай оцінка кожного стану гри залежить від рівня глибини дерева, яке будується Мінімаксом. Чим глибше дерево, тим точніша оцінка стану гри. Проте, підвищення глибини дерева також призводить до збільшення часу виконання алгоритму.

## **1.3 Паралельна версія алгоритму**

Паралельна версія алгоритму Мінімакс заснована на розділенні дерева гри між декількома процесами або потоками, кожен з яких оброблює окрему гілку дерева. Після того, як кожен процес або потік закінчив обробку своєї гілки дерева, результати об'єднуються, і знаходиться оптимальний хід для гравця, який веде гру.

Формула для паралельного алгоритму Мінімакс може бути записана наступним чином:

*minmax(node, depth, is\_maximizing\_player)*

де:

* node - поточний вузол дерева гри
* depth - глибина дерева гри
* is\_maximizing\_player - параметр, який вказує, чи максимізує гравець, який веде гру, свій виграш

У паралельній версії алгоритму Мінімакс кожен процес або потік обробляє свою гілку дерева гри та повертає результат, який містить оптимальний хід для даної гілки. Після того, як всі результати були отримані, об'єднуються, і оптимальний хід для всього дерева гри визначається на основі результатів кожної гілки. Якщо говорити простими словами – то всі ноди корінної ноди запускаються паралельно, з яких потім обирається оптимальний результат.

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

**2.1 Повний перебір**

Припустімо, що ми маємо два масиви. Один містить значення ваги всіх предметів (weights), а другий - їх вартість (values). Також нам дано грузопідйомність рюкзаку (capacity). Визначимо наш метод:

**2.2 Повний перебір**

Алгоритм "brute force" (повний перебір) для розв'язання задачі про рюкзак (knapsack problem) можна описати наступним чином:

1. Створити всі можливі комбінації предметів для розміщення в рюкзаку.
2. Для кожної комбінації перевірити її загальну вагу. Якщо вага перевищує максимальну вмістимість рюкзака, відкинути комбінацію.
3. Для кожної комбінації, яка задовольняє обмеження ваги, обчислити загальну вартість і порівняти її з найкращою відомою вартістю.
4. Зберегти комбінацію з найкращою вартістю.

Реалізацію алгоритму повного перебору ви можете бачити на рисунку 2.?

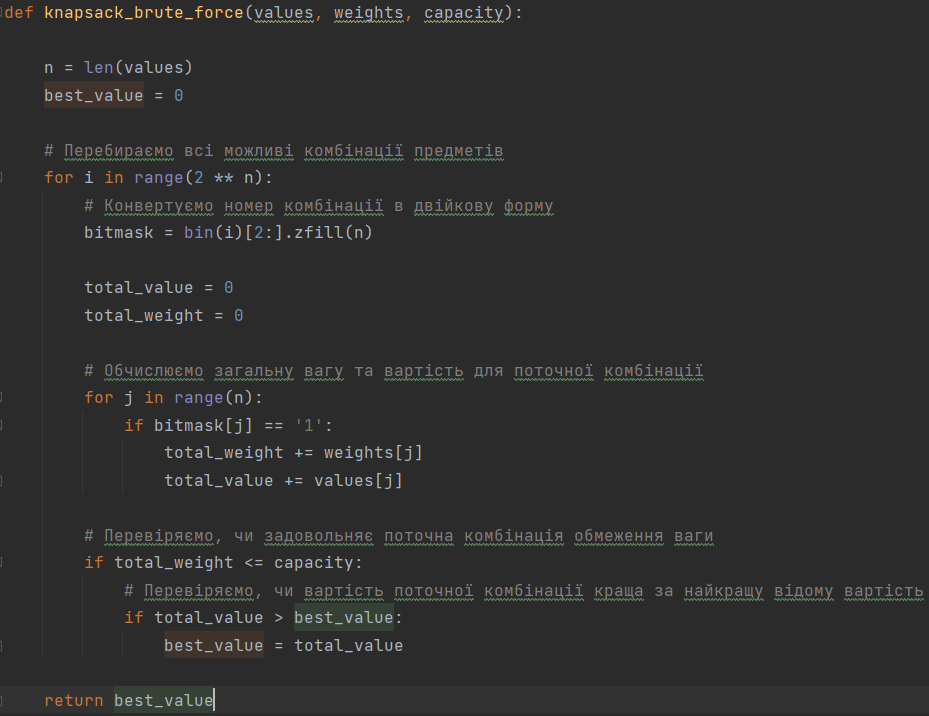


Рис 2.? – Реалізація алгоритму brute force

Тепер протестуємо чи правильно він працює. Задамо наступний тестовий приклад (рис. 2.?)

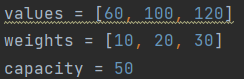


Рисунок 2.? – Тестові дані

Для цього прикладу очікується, що максимальна вартість, яку можна отримати, буде дорівнювати 220.

Застосуємо алгоритм повного перебору до цього прикладу (рис. 2.?)

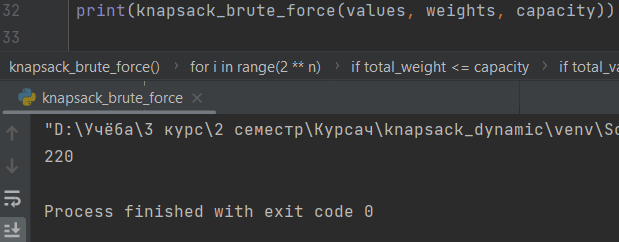


Рисунок 2.? – Застосування алгоритму повного перебору.

Таким чином, алгоритм повного перебору успішно розв'язує задачу про рюкзак для даного прикладу. Однак, слід зазначити, що для великих значень n (кількості предметів), алгоритм повного перебору може бути дуже повільним і непрактичним, оскільки його час виконання зростає експоненційно з кількістю предметів.

**2.3 Жадібний алгоритм**

Жадібний алгоритм для проблеми рюкзака є евристичним алгоритмом, який намагається знайти наближене оптимальне рішення, обираючи крок за кроком найбільш вигідний предмет для додавання до рюкзака.

Жадібний алгоритм працює наступним чином:

1. Розраховується відношення вартості до ваги для кожного предмету.
2. Сортуються всі предмети за спаданням їх відношення вартості до ваги.
3. Предмети додаються до рюкзака, починаючи з найбільш вигідного, до тих пір, поки вмістимість рюкзака не буде досягнуто або всі предмети не будуть взяті.
4. Якщо останній предмет не вміщується повністю, додайте частину його ваги, пропорційну його відношенню вартості до ваги.

Реалізацію жадібного алгоритма ви можете бачити на рисунку 2.?

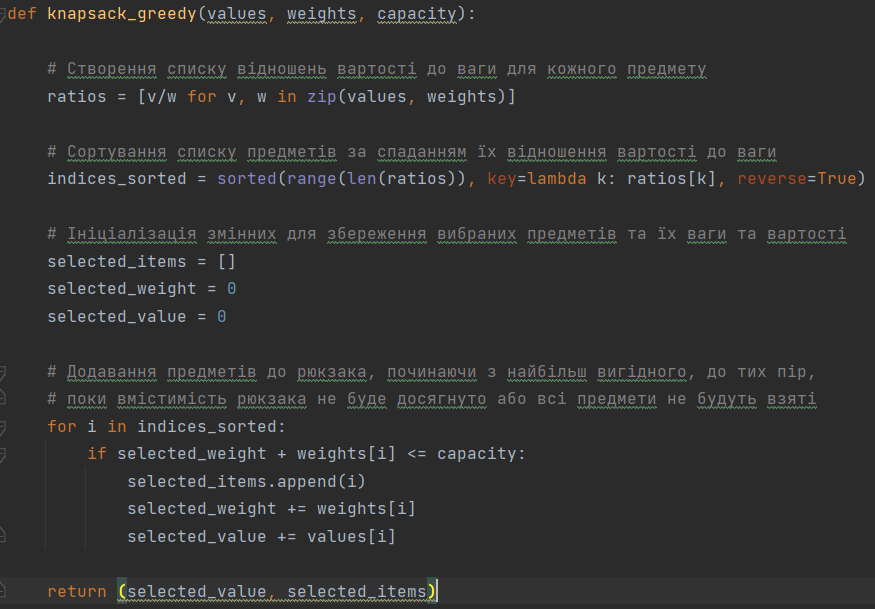


Рисунок 2.?

Тепер протестуємо чи правильно він працює. Задамо наступний тестовий приклад (рис. 2.?)



Рисунок 2.? – Тестові дані

В цьому випадку жадібний алгоритм буде вважати що оптимальний набір – це перший та другий предмет, так як вони мають найбільший VPW (value per weight). Саме тому відповідь алгоритма буде дорівнювати 160 (рис 2.?)

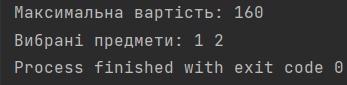


Рисунок 2.? – Рішення жадібного алгоритму

Цей алгоритм працює досить швидко, тому що не вимагає перебору всіх можливих комбінацій предметів. Проте, результат не гарантується бути оптимальним, а лише наближеним до нього.

Жадібний алгоритм для проблеми рюкзака зазвичай використовується як евристика для швидкої оцінки можливих рішень, але для точного розв'язання проблеми рюкзака краще використовувати інші алгоритми, наприклад методи динамічного програмування.

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

3.1 Аналіз існуючих програмних рішень.

Для реалізації курсової роботи була вибрана мова програмування C# її детальні переваги у цьому проекті були наведені на сторінці 3. Але основним критерієм вибору було те, що я більше за все розбираюся в даній мові програмування. Також я обрав Microsoft Visual Studio, яка є хорошим варіантом, оскільки це інтегроване середовище розробки, спеціально призначене для роботи з C# та .NET.

Visual Studio має вбудовані засоби для підтримки паралельного програмування, зокрема, можливість створення мультипотокових програм з використанням бібліотеки Parallel Extensions. Ця бібліотека містить в собі набір інструментів для розпаралелювання обчислень на рівні коду, що дозволяє покращити продуктивність програми.

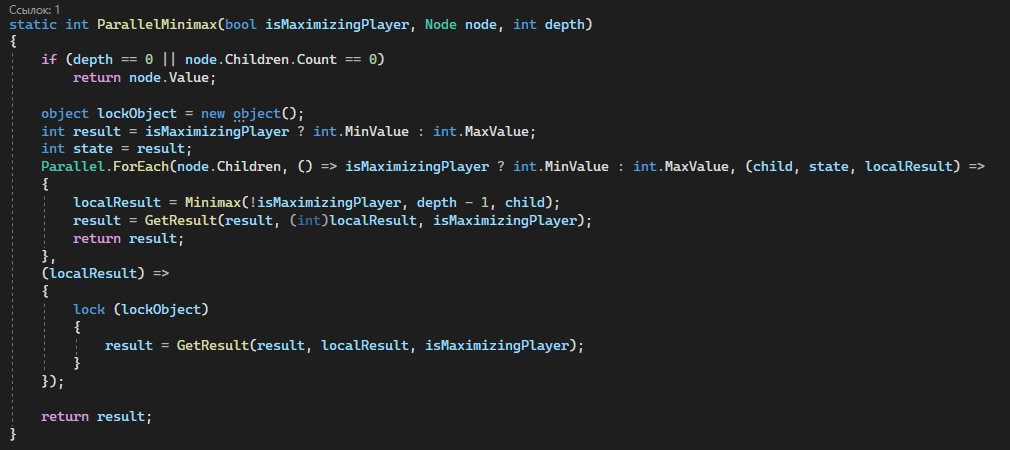
Крім того, Visual Studio має можливості для профілювання програми, що дозволяє виявляти та оптимізувати проблеми або бутленеки в коді, що може бути корисним при розробці паралельних обчислень.

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

## **4.1 Проектування**

Для того, щоб створити паралельну реалізацію алгоритму МініМакс, як i будь-якого іншого алгоритму, спочатку необхідно визначитися, як правильно розбивати задачу на підзадачі, так щоб виконувати ці підзадачі паралельно. Так як у нашого батьківського вузла є список з дочірніх вузлів, можна розбити нашу задачу на підзадачі, де метою кожної з них буде обчислення найбільш неоптимального значення вузла та вже по результатам всіх підзадач вибрати найбільш оптимальний варіант.

## **4.2 Реалізація**

Після того як ми визначилися, що потрібно для створення паралельної версії алгоритму – можна переходити до реалізації. (Рис. 4.1) та (Рис. 4.2)

Graphical user interface, text

Description automatically generatedРис. 4.1 – Паралельна реалізація алгоритму МініМакс

Рис. 4.2 – Реалізація методу GetResult

# **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. Python multiprocessing: https://docs.python.org/3/library/multiprocessing.html
2. …..
3. …..
4. ….
5. …..

# **ДОДАТКИ**

## **Додаток А. Нзава додатку**

## **Додаток Б. Назва додатку**

## **Додаток В. Назва додатку**