МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
РАДІОЕЛЕКТРОНІКИ

Кафедра Штучного Інтелекту

Звіт

про виконання лабораторної роботи №1

«Створення багатопотокових програмних застосунків під платформу .NET»

з дисципліни «Програмування під .NET Core»

Виконав:
ст. гр. ІТШІ-20-2
Науменко А.С.

Прийняв:
Бібічков І.Є.

Харків – 2023

1. НАЗВА РОБОТИ:

Створення багатопотокових програмних застосунків під платформу .NET.

2. МЕТА РОБОТИ:

Вивчення особливостей створення багатопотокових програмних застосунків під платформу .NET. Вивчення особливостей розпаралелювання математичних алгоритмів.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ:

16**. Знаходження мінімаксу запропонованої платіжної матриці.

3. ПОРЯДОК ВИКОНАННЯ РОБОТИ:

1. Узгодити з викладачем завдання.

2. Реалізувати запропоноване завдання у вигляді однопотокового застосунку.

3. Здійснити розпаралелювання алгоритму для його подальшої програмної реалізації у вигляді багатопотокового застосунку.

4. Реалізувати розроблений алгоритм як багатопотоковий застосунок. Для реалізації необхідно вибрати .NET технологію створення багатопотокових застосунків, що передбачає явне створення потоків. При цьому в застосунку має бути можливість завдання кількості потоків, що породжуються. Кількість потоків, що породжуються, може залежати, наприклад, від числа ядер процесору, але не повинна прямо залежати від розмірності завдання.

5. Рішення обох варіантів реалізації алгоритму мають бути об'єктно-орієнтованими.

6. На тестових прикладах різної розмірності і з різною кількістю потоків, що породжуються, оцінити величину прискорення, що дається багатопотоковою реалізацією алгоритму. Прискорення необхідно розраховувати як відношення часу виконання однопотокової версії алгоритму до часу виконання багатопотокової його версії. Виявлені закономірності відобразити у висновках.

7. Завдання, не відзначені зірочками, оцінюються з максимуму в 18 балів. Позначені однією зірочкою – з максимуму в 19 балів. Позначені двома та трьома зірочками – з максимуму в 20 балів. За вибір завдання, відзначеного трьома зірочками,

надається компенсаційний бал. Вітається пропонування власних оригінальних варіантів завдання.

8. Підготувати звіт з лабораторної роботи.

4. ОПИС ОДНОПОТОКОВОГО ВАРІАНТУ АЛГОРИТМУ:

Алгоритм знаходження мінімаксу для платіжної матриці працює полягає у переборі кожного рядка матриці та визначення максимального значення в кожному з них. Потім алгоритм знаходить мінімум серед отриманих максимальних значень у рядках. Це і є мінімакс.

5. ВИХІДНІ ТЕКСТИ ОДНОПОТОКОВОЇ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ АЛГОРИТМУ:

6. КОРОТКИЙ ОПИС ВИБРАНОЇ .NET ТЕХНОЛОГІЇ СТВОРЕННЯ БАГАТОПОТОКОВИХ ЗАСТОСУНКІВ:

Для роботи з паралельними потоками я використовував клас Thread, який входить до простору імен System.Threading. Цей клас дозволяє створювати та керувати асинхронним виконанням коду в межах програми. Клас Thread має різноманітні методи та властивості для управління життєвим циклом та виконанням потоку.

7. ОПИС БАГАТОПОТОКОВОГО ВАРІАНТУ АЛГОРИТМУ:

Відмінність багатопотокового алгоритму заключається в тому, що для пошуку максимума в рядках було розподілено подекілька або одному рядку на кожен поток. Так само потім і для пошуку мінімума.

8. ВИХІДНІ ТЕКСТИ БАГАТОПОТОКОВОЇ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ:

MatrixAlgorithm.cs

```
using System.Diagnostics;

namespace Minimax;

static class MinimaxAlgorithm
{
    public static (double, long) FindMinimax(double[,] matrix)
    {
        var rows = matrix.GetLength(0);
        var cols = matrix.GetLength(1);

        Stopwatch stopwatch = new();
        stopwatch.Start();

        double[] rowMax = new double[rows];
        for (int i = 0; i < rows; i++)
        {
            double maxInRow = double.MinValue;
            for (int j = 0; j < cols; j++)
            {
                if (matrix[i, j] > maxInRow)
                {
                    maxInRow = matrix[i, j];
                }
            }
            rowMax[i] = maxInRow;
        }

        double minimax = rowMax[0];
        for (int i = 1; i < rows; i++)
        {
            if (rowMax[i] < minimax)
            {
                minimax = rowMax[i];
            }
        }

        stopwatch.Stop();
        return (minimax, stopwatch.ElapsedTicks);
    }

    public static (double, long, int) FindMinimaxParallel(double[,] matrix, int
threadsAmount)
    {
        var rows = matrix.GetLength(0);
        var cols = matrix.GetLength(1);

        if (threadsAmount <= 0)
        {
            threadsAmount = Environment.ProcessorCount * 12 / 7;
        }

        if (threadsAmount > rows)
        {
            threadsAmount = rows;
        }
```

```
var rowsPerOneThread = rows / threadsAmount;

var allThreads = new List<int>();
for (int i = 0; i < threadsAmount; i++)
{
    allThreads.Add(rowsPerOneThread);
}
allThreads[^1] += rows % threadsAmount;

var startedThreadsAmount = 0;
var resetEvent = new ManualResetEvent(false);

Stopwatch stopwatch = new();
stopwatch.Start();

double[] rowMax = new double[rows];
for (int threadI = 0; threadI < threadsAmount; threadI++)
{
    var currentI = threadI;
    var thread = new Thread(() =>
    {
        for (int i = currentI * rowsPerOneThread;
           i < currentI * rowsPerOneThread + allThreads[currentI];
           i++)
        {
            double maxInRow = double.MinValue;
            for (int j = 0; j < cols; j++)
            {
                if (matrix[i, j] > maxInRow)
                {
                    maxInRow = matrix[i, j];
                }
            }
            rowMax[i] = maxInRow;
        }


        if (Interlocked.Increment(ref startedThreadsAmount) ==
                    threadsAmount)
        {
            resetEvent.Set();
        }
    });

    thread.Start();
}

resetEvent.WaitOne();
resetEvent.Reset();
startedThreadsAmount = 0;
var lockObject = new object();

double minimax = rowMax[0];
for (int i = 0; i < threadsAmount; i++)
{
    var currentI = i;
    var thread = new Thread(() =>
    {
        for (int j = currentI * rowsPerOneThread;
           j < currentI * rowsPerOneThread + allThreads[currentI];
           j++)
        {
            if (rowMax[j] < minimax)
            {
                lock (lockObject)
                {
                    minimax = rowMax[j];
```

```
                        }
                    }
                }

                if (Interlocked.Increment(ref startedThreadsAmount) ==
                        threadsAmount)
                {
                    resetEvent.Set();
                }
            });

            thread.Start();
        }

        resetEvent.WaitOne();
        stopwatch.Stop();

        return (minimax, stopwatch.ElapsedTicks, threadsAmount);
    }
}
```

Program.cs

```
using Minimax;


var parametersList = new List<Parameters>
{
    new(10, 10, 5),
    new(10, 10, 10),

    new(900, 900, 100),
    new(900, 900, 200),
    new(900, 900, 900),

    new(9000, 9000, 100),
    new(9000, 9000, 300),
    new(9000, 9000, 1000),
    new(9000, 9000, 3000),

    new(900000, 1000, 500),
    new(900000, 1000, 1000),
    new(900000, 1000, 2000),
};


var random = new Random();
foreach (var parameters in parametersList)
{
    var matrix = new double[parameters.RowsAmount,parameters.ColumnsAmount];
    for (int i = 0; i < parameters.RowsAmount; i++)
    {
        for (int j = 0; j < parameters.ColumnsAmount; j++)
        {
            matrix[i, j] = random.Next(-999, 999);
        }
    }

    var result1 = MinimaxAlgorithm.FindMinimax(matrix);
    var result2 = MinimaxAlgorithm.FindMinimaxParallel(matrix,
parameters.ThreadsAmount);

    Console.WriteLine();
    Console.WriteLine("Для матриці " + parameters.RowsAmount + "X" +
parameters.ColumnsAmount);
```

```
    Console.WriteLine("Мінімакс однопотокового: " + result1.Item1);
    Console.WriteLine("Мінімакс багатопотокового: " + result2.Item1);
    Console.WriteLine("Час виконання однопотокового алгоритму: " + result1.Item2 + "
тіків");
    Console.WriteLine("Час виконання багатопотокового алгоритму: " + result2.Item2 + "
тіків");
    Console.WriteLine("Потоків задіяно: " + result2.Item3);
    Console.WriteLine("Величина прискорення: " + result1.Item2 /
(double)result2.Item2);
}


class Parameters
{
    public Parameters(int rowsAmount, int columnsAmount, int threadsAmount)
    {
        RowsAmount = rowsAmount;
        ColumnsAmount = columnsAmount;
        ThreadsAmount = threadsAmount;
    }

    public int RowsAmount { get; }
    public int ColumnsAmount { get; }
    public int ThreadsAmount { get; }
}
```

9. СКРІНИ РЕЗУЛЬТАТІВ РОБОТИ КОЖНОЇ ПРОГРАМИ І ЗНАЧЕННЯ ОЦІНОК ПРИСКОРЕННЯ:

```
Для матриці 10X10
Мінімакс однопотокового: 55
Мінімакс багатопотокового: 55
Час виконання однопотокового алгоритму: 4335 тіків
Час виконання багатопотокового алгоритму: 66964 тіків
Потоків задіяно: 5
Величина прискорення: 0.06473627620811181

Для матриці 10X10
Мінімакс однопотокового: 665
Мінімакс багатопотокового: 665
Час виконання однопотокового алгоритму: 14 тіків
Час виконання багатопотокового алгоритму: 154806 тіків
Потоків задіяно: 10
Величина прискорення: 9.043577122333759E-05

Для матриці 900X900
Мінімакс однопотокового: 978
Мінімакс багатопотокового: 978
Час виконання однопотокового алгоритму: 35183 тіків
Час виконання багатопотокового алгоритму: 1001322 тіків
Потоків задіяно: 100
Величина прискорення: 0.035136549481585344
```

```
Для матриці 900X900
Мінімакс однопотокового: 984
Мінімакс багатопотокового: 984
Час виконання однопотокового алгоритму: 64581 тіків
Час виконання багатопотокового алгоритму: 1900710 тіків
Потоків задіяно: 200
Величина прискорення: 0.03397730321827107

Для матриці 900X900
Мінімакс однопотокового: 982
Мінімакс багатопотокового: 982
Час виконання однопотокового алгоритму: 42537 тіків
Час виконання багатопотокового алгоритму: 6327082 тіків
Потоків задіяно: 900
Величина прискорення: 0.006723004380218243

Для матриці 9000X9000
Мінімакс однопотокового: 997
Мінімакс багатопотокового: 997
Час виконання однопотокового алгоритму: 2876990 тіків
Час виконання багатопотокового алгоритму: 1266707 тіків
Потоків задіяно: 100
Величина прискорення: 2.2712355738146233
```

```
Для матриці 9000X9000
Мінімакс однопотокового: 996
Мінімакс багатопотокового: 996
Час виконання однопотокового алгоритму: 2691743 тіків
Час виконання багатопотокового алгоритму: 2345016 тіків
Потоків задіяно: 300
Величина прискорення: 1.1478569869032877

Для матриці 9000X9000
Мінімакс однопотокового: 996
Мінімакс багатопотокового: 996
Час виконання однопотокового алгоритму: 4052444 тіків
Час виконання багатопотокового алгоритму: 9045750 тіків
Потоків задіяно: 1000
Величина прискорення: 0.4479942514440483

Для матриці 9000X9000
Мінімакс однопотокового: 996
Мінімакс багатопотокового: 996
Час виконання однопотокового алгоритму: 3972786 тіків
Час виконання багатопотокового алгоритму: 19330973 тіків
Потоків задіяно: 3000
Величина прискорення: 0.2055140214618271
```

```
Для матриці 900000X1000
Мінімакс однопотокового: 969
Мінімакс багатопотокового: 969
Час виконання однопотокового алгоритму: 242999206 тіків
Час виконання багатопотокового алгоритму: 10825935 тіків
Потоків задіяно: 500
Величина прискорення: 22.446024846814616

Для матриці 900000X1000
Мінімакс однопотокового: 971
Мінімакс багатопотокового: 971
Час виконання однопотокового алгоритму: 64840374 тіків
Час виконання багатопотокового алгоритму: 15395459 тіків
Потоків задіяно: 1000
Величина прискорення: 4.211655787592952

Для матриці 900000X1000
Мінімакс однопотокового: 970
Мінімакс багатопотокового: 970
Час виконання однопотокового алгоритму: 130438367 тіків
Час виконання багатопотокового алгоритму: 19309229 тіків
Потоків задіяно: 2000
Величина прискорення: 6.755234349336268
```

ВИСНОВКИ:

Під час лабораторної роботи було вивчено особливості створення багатопотокових програмних застосунків під платформу .NET та вивчено особливості розпаралелювання математичних алгоритмів. Було розроблено однопотоковий та багатопотоковий алгоритми. В результаті аналізу виявлено, що при рівній чи близкій кількості потоків та стопців матриці більш ефективним є однопотоковий алгоритм, але якщо рядків суттєво більше, ніж потоків, то багатопотоковий алгоритм працює значно швидше звичайного.