**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”**

кафедра ЕОМ



***Лабораторна робота №2***

**з дисципліни “ Паралельні та розподілені обчислення “**

**на тему:**

**“Паралельне представлення алгоритмів”**

Виконав: студент .гр. КІ-33

Харченко М.В.

Прийняв: асистент каф. ЕОМ

Козак Н.Б.

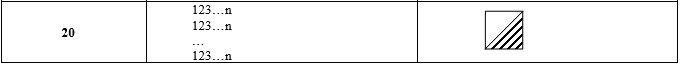
Львів – 2020

**Мета:** Вивчити можливості паралельного представлення алгоритмів. Набути навиків такого представлення.

**Завдання:**

Запропонувати та реалізувати локально-рекурсивний алгоритм обчислення виразу: , де *А* та *В* матриці з елементами  та , відповідно . Матриця *А* задається однозначно і залежить лише від розмірності даних. Для матриці *В*: заштрихована область — довільні цілі числа, відмінні від нуля, а незаштрихована область - нулі.

№ A B



**Послідовність виконання роботи:**

1. Програма з одноразовим присвоюванням.

void ArrayMul(int[,] arrA, int[,] arrB, int N)

{

int[,] arrC = new int[N, N + 1];

int n = N - 1;

for (int i = 0; i < N; i++)

{

arrC[i, 0] = 0;

for (int j = n; j < N; j++)

{

arrC[i, j + 1] = arrC[i, j] + arrA[0, j]\* arrB[j, i];

Count += 2;

}

n--;

}

Show(arrC, N, "З одноразовим присвоєнням", Count);

}

Всі рядки матриці А є однаковими і немає потреби множення матриці на матрицю оскільки у результуючій матриці всі рядки також будуть однаковими. Тому я перемножив рядок-вектор матриці А на матрицю В, отримав рядок-вектор С і клонував його на всі інші рядки матриці С.

1. Рекурсивні рівняння:

*k* – індекс рекурсії.

1. Граф залежностей(n=4):



1. Оптимізований граф залежностей(n=4):



1. Аналітичні оцінки кількості арифметичних операцій та їх порівняння.

В безпосередньому графі залежностей кількість операцій рівна N2(2N-1) де n кількість стовпців чи рядків матриці.(Для обчислення кожного з N2 елементів необхідно 2N-1 операцій).

За оптимізованим графом залежностей кількість операцій рівна ((N+1)/2)\*2N=(N+1)\*N

Для N=4 за безпосереднім графом кількість операцій дорівнює 112, а за оптимізованим 20 що у 5.6 разів швидше.

Для N=10 за безпосереднім графом 1900 операцій, а за оптимізованим 110 що у 17.27 разів швидше.

1. Текст програми, що реалізовує оптимізований локально-рекурсивний алгоритм:

private void localAlgorutm(int[,] arrA, int[,] arrB, int N)

{

int q = 0;

int[,] TempA = new int[N + 1, N];

int[,] arrC = new int[N, N + 1];

int n = N - 2;

for (int j = 0; j < N; j++)

TempA[N, j] = arrA[0, j];

for (int i = 0; i < N; i++)

{

arrC[N - 1 - i, 0] = 0;

for (int j = i; j < N; j++)

{

TempA[N - 1 - i, j] = TempA[N - i, j];

arrC[N - i - 1, j + 1] = arrC[N - 1 - i, j] + TempA[N - 1 - i, j] \* arrB[j, N - 1 - i];

q += 2;

}

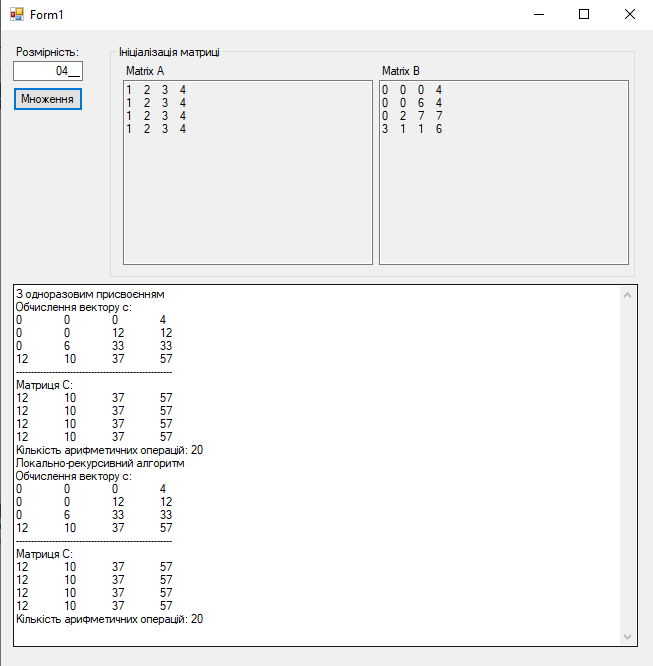
n--;

}

Show(arrC, N, "Локально-рекурсивний алгоритм", q);

}

Результат виконання програми для n = 4;



Висновок: виконуючи лабораторну роботу, я отримав навики паралельного обчислення алгоритмів на базі множення матриць, також навчився реалізовувати алгоритм з одноразовим присвоєнням і локально-рекурсивний алгоритм, який працює на базі локалізованого графа.