

Міністерство Освіти і Науки України
Національний Університет “Львівська Політехніка”



Кафедра ЕОМ

ПАРАЛЕЛЬНЕ ПРЕДСТАВЛЕННЯ АЛГОРИТМІВ.

Методичні вказівки

до лабораторної роботи з курсу “Паралельні та розподілені обчислення”
для студентів спеціальності 123 - “Комп’ютерна інженерія”

Затверджено
на засіданні кафедри
”Електронні обчислювальні машини”
Протокол № _ від 2021 року

Львів – 2021

МЕТА РОБОТИ. Вивчити можливості паралельного представлення алгоритмів. Набути навиків такого представлення.

ТЕОРЕТИЧНІ ВІДОМОСТІ.

Можливі два підходи до побудови паралельного представлення алгоритму:

1. Векторизація алгоритму представленого послідовно.
2. Безпосередньо паралельне представлення:
 - 2.1. Кадри.
 - 2.2. Програми з одноразовим присвоєнням.
 - 2.3. Рекурсивні рівняння.
 - 2.4. Графи залежностей

1. Векторизація

Це процес генерації паралельних машинних кодів на основі послідовного алгоритму, записаного на деякій мові програмування. Вона виконується, як правило векторизуючим компілятором (автоматично) і полягає у виявленні та аналізі залежностей між операторами з метою паралельного виконання незалежних, невпорядкованих дій.

2. Пряме представлення паралельних алгоритмів

2.1. Кадр – опис обчислювальних дій в конкретний момент часу. Кадри є найбільш природнім засобом представлення алгоритму, за допомогою якого розробник може перевірити певний математичний алгоритм.

2.2. Програма з одноразовим присвоєнням – це форма, в якій кожній змінній присвоюється лише одне значення при виконанні алгоритму.

Приклад.

Розглянемо задачу множення матриці на вектор, яка описується формулою:

$$c_i = \sum_{j=1}^N A_{ij} b_j \quad (1)$$

Безпосередня реалізація на послідовній мові програмування (в даному випадку – на мові C) має вигляд:

```
for (i=0; i<N; i++)  
{  
    c[i]=0;  
    for (j=0; j<N; j++)  
        c[i]=c[i]+A[i][j]*b[j];  
}
```

В цій програмі $c[i]$ переписується багато разів з метою економії пам'яті. Таким чином, значення $c[i]$ присвоюється більше одного разу. При перетворенні цієї ж програми в програму з одноразовим присвоєнням кількість індексів вектора c – зросте:

```
for (i=0; i<N; i++)  
{  
    c[i][0]=0;
```

```

    for (j=0; j<N; j++)
    c[i][j+1]=c[i][j]+A[i][j]*b[j];
    }

```

Тепер, кожному елементу вектора c буде присвоєно лише одне значення, а остаточні значення будуть отримані на останньому кроці ітерації.

2.3. Рекурсивний алгоритм – це алгоритм, який визначається за допомогою правила одноразового присвоювання і є стислим представленням багатьох алгоритмів. Побудова рекурсивного алгоритму зводиться до виведення рекурсивних рівнянь. Дії паралельних алгоритмів адекватно описуються в рекурсивних рівняннях з просторово-часовими індексами якщо один індекс використовується для часу, а інші – для простору (надалі – індексний простір).

Приклад.

Для випадку множення матриці на вектор, рекурсивне рівняння буде мати вигляд:

$$c_i^{(j+1)} = c_i^{(j)} + A_i^{(j)} * b_i^{(j)} \quad (2)$$

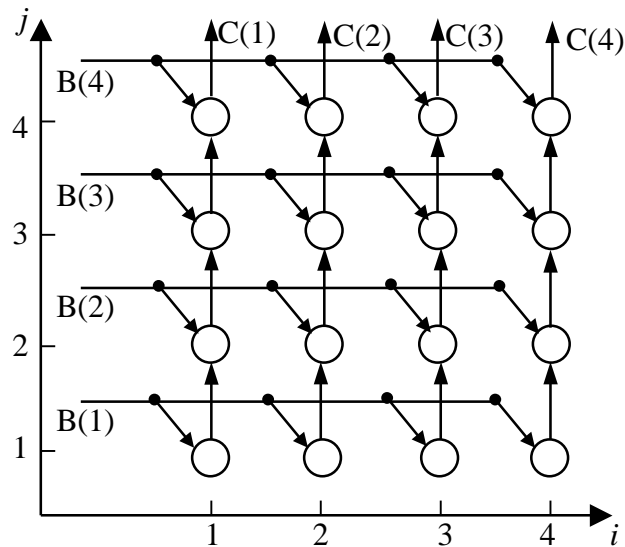
$$a_i^{(j)} = A[i][j];$$

$$b_i^{(j)} = b[j]$$

j – індекс рекурсії.

2.4. Граф залежностей (ГЗ) - це граф, який описує залежність обчислень в алгоритмі. ГЗ може розглядатися як графічне представлення алгоритму з одноразовим присвоєнням. ГЗ називається повним, якщо він визначає всі залежності між всіма змінними в індексному просторі. Переважно, операції в вузлах графу не розкриваються, оскільки будуть виконуватися незалежними обчислювальними засобами (часто – процесорними елементами) і граф є скороченим..

Приклад. Для обчислення (1), виходячи з наведеного алгоритму з одноразовим присвоєнням очевидно, що $c[i][j+1]$ безпосередньо залежить від $c[i][j]$, $A[i][j]$, $b[j]$. Представивши кожну залежність у вигляді дуги між відповідними змінними, що розташовані в індексному просторі, можна отримати ГЗ:



ГЗ для множення матриці на вектор (для $N=4$) з глобальним зв'язком.

З нього видно, що значення $b[j]$ кожного елемента вектора b має бути розповсюджене у всі індексні точки, що мають однаковий індекс j . Цей тип даних називається “поширюваними” даними. Це означає, що має бути глобальний зв'язок, що не завжди прийнятна умова для обчислювальної системи.

Можна стверджувати, що алгоритм є зчисленням, якщо його повний граф не містить петель і циклів

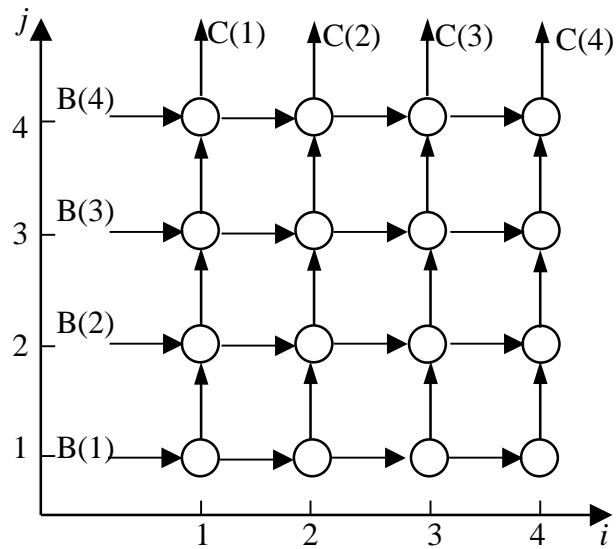
Локалізований Граф Залежностей. Алгоритм є локалізований, якщо всі змінні безпосередньо залежать лише від змінних в сусідніх вузлах. Дані, що пересилаються незмінними до всіх вершин графу називаються передаваними, в іншому випадку – це непередані дані.

Приклад. Програма для локалізованого алгоритму має вигляд ($b[0][j]=b[j]$):

```
for (i=0; i<N; i++)
{
    c[i][0]=0;
    for (j=0; j<N; j++)
        b[i+1][j]=b[i][j]
        c[i][j+1]=c[i][j]+A[i][j]*b[i][j];
}
```

в ній $b[i+1][j]$ безпосередньо залежить від $b[i][j]$, а $c[i][j+1]$ від $c[i][j]$, $A[i][j]$, $b[i][j]$.

Відповідний граф залежностей лише з локальними зв'язками має вигляд:



ГЗ для множення матриці на вектор (для $N=4$) з локальним зв'язком.

Виходячи з локалізованого ГЗ можна дати означення:

Локально-рекурсивний алгоритм – це алгоритм, відповідний ГЗ якого має лише локальні залежності, тобто розмір задачі не впливає на довжину кожної дуги і більшість вузлів ГЗ складається з операцій одного типу.

ЗАВДАННЯ.

Запропонувати та реалізувати локально-рекурсивний алгоритм обчислення виразу:

$$Y = A \times B ,$$

де A та B матриці з елементами a_{ij} та b_{ij} , відповідно ($i, j = 1 \dots N$), тобто:

$$y_{ij} = \sum_{k=1}^N a_{ik} b_{kj} \quad (k = 1 \dots N) .$$

Тип вхідних послідовностей визначається згідно варіанту.

ПОРЯДОК ВИКОНАННЯ РОБОТИ

1. Написати програму з одноразовим присвоюванням.
2. Знайти рекурсивні рівняння (тобто рекурсивний алгоритм).
3. Побудувати граф залежностей та виходячи з нього локалізований граф залежностей.
4. Оптимізувати граф залежностей, врахувавши тип вхідних даних, тобто усунути зайві фрагменти обчислень.

5. Визначити та порівняти кількість арифметичних операцій, що потрібно здійснити при обчисленні виразу за безпосереднім та оптимізованим графами залежностей.

6. Написати програму, що реалізовує локально-рекурсивний алгоритм.

7. Зробити висновок про ефективність обчислень виходячи з графу залежностей.

ЗМІСТ ЗВІТУ

1. Тема, мета, аналіз завдання.

2. Результати виконання роботи, тобто:

- текст програми з одноразовим присвоюванням;
- рекурсивні рівняння;
- локалізований граф залежностей;
- оптимізований граф залежностей;
- аналітичні оцінки кількості арифметичних операцій та їх порівняння;
- текст програми, що реалізовує оптимізований локально-рекурсивний алгоритм;
- результат роботи програми на довільному наборі вхідних даних, для розмірності $n \geq 3$.

3. Висновки.

ЛІТЕРАТУРА











С.Немногин О.Стесик “Параллельное программирование для многопроцессорных систем” Петербург “БХВ-Петербург”, 2002

Томас Бройнль “Параллельне програмування. Початковий курс”Київ "Вища школа, 1997

ВАРІАНТИ ЗАВДАНЬ

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.

варіант №	Тип матриці А	Тип матриці В
1	$\begin{matrix} n & 0 & \dots & 0 \\ 0 & n-1 & \dots & 0 \\ & & \dots & \\ 0 & \dots & & 1 \end{matrix}$	
2	$\begin{matrix} 1*2 & 0 & \dots & 0 \\ 0 & 2*3 & \dots & 0 \\ & & \dots & \\ 0 & \dots & & n(n+1) \end{matrix}$	
3	$\begin{matrix} 11\dots 11 \\ 1 \dots 1 \\ \cdot \ 0 \cdot \\ 1 \dots 1 \\ 11\dots 11 \end{matrix}$	
4	$\begin{matrix} 111\dots 111 \\ 011\dots 110 \\ \dots \\ 011\dots 110 \\ 111\dots 111 \end{matrix}$	
5	$\begin{matrix} 100\dots 001 \\ 110\dots 011 \\ \dots\dots\dots \\ 110\dots 011 \\ 100\dots 001 \end{matrix}$	
6	$\begin{matrix} n & 0 & \dots & 0 & \dots & 0 \\ n-1 & n & & 0 & \dots & 0 \\ n-2 & n-1 & n & \dots & & \\ \dots & & & & & \\ 1 & 2 & 3 & \dots & n & \end{matrix}$	
7	$\begin{matrix} 1 & 2 & 3 & \dots & n \\ \dots & & & & \\ n-2 & n-1 & n & \dots & \\ n-1 & n & 0 & \dots & 0 \\ n & 0 & \dots & 0 & \end{matrix}$	
8	$\begin{matrix} 1 & 2 & 3 & \dots & n-1 & n \\ 2 & 1 & 2 & \dots & n-2 & n-1 \\ 3 & 2 & 1 & \dots & n-3 & n-2 \\ \dots & & & & & \\ n-1 & n-2 & n-3 & \dots & 1 & 2 \\ n & n-1 & n-2 & \dots & 2 & 1 \end{matrix}$	
9	$\begin{matrix} 111\dots 111 \\ 222\dots 220 \\ 333\dots 300 \\ \dots \\ n000\dots 00 \end{matrix}$	
10	$\begin{matrix} 123\dots n \\ 123\dots n \\ \dots \\ 123\dots n \end{matrix}$	

ВАРІАНТИ ЗАВДАНЬ

Матриця А задається однозначно і залежить лише від розмірності даних.

Для матриці В: заштрихована область – довільні цілі числа, відмінні від нуля, а не заштрихована область – нулі.

варіант №	Тип матриці А	Тип матриці В
11	$\begin{matrix} n & 0 & \dots & 0 \\ 0 & n-1 & \dots & 0 \\ & & \dots & \\ 0 & \dots & & 1 \end{matrix}$	
12	$\begin{matrix} 1*2 & 0 & \dots & 0 \\ 0 & 2*3 & \dots & 0 \\ & & \dots & \\ 0 & \dots & & n(n+1) \end{matrix}$	
13	$\begin{matrix} 11\dots11 \\ 1\dots1 \\ .\ 0\ . \\ 1\dots1 \\ 11\dots11 \end{matrix}$	
14	$\begin{matrix} 111\dots111 \\ 011\dots110 \\ \dots \\ 011\dots110 \\ 111\dots111 \end{matrix}$	
15	$\begin{matrix} 100\dots001 \\ 110\dots011 \\ \dots\dots\dots \\ 110\dots011 \\ 100\dots001 \end{matrix}$	
16	$\begin{matrix} n & 0 & .0 & \dots & 0 \\ n-1 & n & 0 & \dots & 0 \\ n-2 & n-1 & n & \dots & \\ \dots & & & & \\ 1 & 2 & 3 & \dots & n \end{matrix}$	
17	$\begin{matrix} 1 & 2 & 3 & \dots & n \\ & & & \dots & \\ n-2 & n-1 & n & \dots & \\ n-1 & n & 0 & \dots & 0 \\ n & 0 & .0 & \dots & 0 \end{matrix}$	
18	$\begin{matrix} 1 & 2 & 3 & \dots & n-1 & n \\ 2 & 1 & 2 & \dots & n-2 & n-1 \\ 3 & 2 & 1 & \dots & n-3 & n-2 \\ \dots & & & & & \\ n-1 & n-2 & n-3 & \dots & 1 & 2 \\ n & n-1 & n-2 & \dots & 2 & 1 \end{matrix}$	
19	$\begin{matrix} 111\dots111 \\ 222\dots220 \\ 333\dots300 \\ \dots \\ n000\dots00 \end{matrix}$	
20	$\begin{matrix} 123\dots n \\ 123\dots n \\ \dots \\ 123\dots n \end{matrix}$	

ВАРІАНТИ ЗАВДАНЬ

Матриця А задається однозначно і залежить лише від розмірності даних.

Для матриці В: заштрихована область – довільні цілі числа, відмінні від нуля, а не заштрихована область – нулі.

варіант №	Тип матриці А	Тип матриці В
21	$\begin{matrix} n & 0 & \dots & 0 \\ 0 & n-1 & \dots & 0 \\ & & \dots & \\ 0 & \dots & 1 \end{matrix}$	
22	$\begin{matrix} 1*2 & 0 & \dots & 0 \\ 0 & 2*3 & \dots & 0 \\ & & \dots & \\ 0 & \dots & n(n+1) \end{matrix}$	
23	$\begin{matrix} 11\dots11 \\ 1\dots1 \\ \cdot & 0 & \cdot \\ 1\dots1 \\ 11\dots11 \end{matrix}$	
24	$\begin{matrix} 111\dots111 \\ 011\dots110 \\ \dots \\ 011\dots110 \\ 111\dots111 \end{matrix}$	
25	$\begin{matrix} 100\dots001 \\ 110\dots011 \\ \dots\dots\dots \\ 110\dots011 \\ 100\dots001 \end{matrix}$	
26	$\begin{matrix} n & 0 & \cdot 0 & \dots & 0 \\ n-1 & n & 0 & \dots & 0 \\ n-2 & n-1 & n & \dots & \\ \dots & & & & \\ 1 & 2 & 3 & \dots & n \end{matrix}$	
27	$\begin{matrix} 1 & 2 & 3 & \dots & n \\ & & & \dots & \\ n-2 & n-1 & n & \dots & \\ n-1 & n & 0 & \dots & 0 \\ n & 0 & \cdot 0 & \dots & 0 \end{matrix}$	
28	$\begin{matrix} 1 & 2 & 3 & \dots & n-1 & n \\ 2 & 1 & 2 & \dots & n-2 & n-1 \\ 3 & 2 & 1 & \dots & n-3 & n-2 \\ \dots & & & & & \\ n-1 & n-2 & n-3 & \dots & 1 & 2 \\ n & n-1 & n-2 & \dots & 2 & 1 \end{matrix}$	
29	$\begin{matrix} 111\dots111 \\ 222\dots220 \\ 333\dots300 \\ \dots \\ n000\dots00 \end{matrix}$	
30	$\begin{matrix} 123\dots n \\ 123\dots n \\ \dots \\ 123\dots n \end{matrix}$	

Приклад виконання роботи

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.

Варіант завдання:

1 2 3 ... n-1 n	
2 1 2 ... n-2 n-1	
3 2 1 ... n-3 n-2	
...	
n-1 n-2 n-3 ... 1 2	
n n-1 n-2 ... 2 1	
...	
...	
...	
...	

1. Рекурсивні рівняння:

$C_{ij}^{(k+1)} = C_{ij}^{(k)} + A_{ik}^{(j)} * B_{kj}^{(i)}$, де $A_{ij}^{(k)} = A[i, k, j]$, $B_{ij}^{(k)} = B[k, j, i]$
 k, i, j - індекси рекурсії.

Цикл Загального алгоритму множення матриць.

```
for (int k = 0; k < N; k++)
{
    for (int l = 0; l < N; l++)
    {
        for (int s = 0; s < N; s++)
        {
            MatrixR[k, l] += MatrixA[k, s] * MatrixB[s, l];
        }
    }
}
```

Як бачимо, в цій програмі MatrixR[k,l] переписується багато разів з метою економії пам'яті. Таким чином, значення MatrixR[k,l] присвоюється більше одного разу. І так як цей алгоритм не оптимізований, кількість операцій над даними більша ніж в оптимізованому локально-рекурсивному алгоритмі, результати порівнянь яких можна побачити далі в таблиці і на графіку.

Цикл алгоритму з одноразовим присвоюванням.

```
for (int i = 0; i < N; i++)
    for (int j = 0; j < N; j++)
        for (int k = 0; k < N; k++)
        {
            MatrixY[i, j, k + 1] = MatrixY[i, j, k] + MatrixA[i, k] * MatrixB[k, j];
        }
```

Кожному елементу матриці с буде присвоєно лише одне значення, а остаточні значення будуть отримані на останній грані.

Оптимізований Локально-рекурсивний алгоритм.

```
if (i < N)
{
    if (j < N)
    {
        if (k < N)
        {
            MA[i, k, j + 1] = MA[i, k, j];
        }
    }
}
```

```

    MB[k, j, i + 1] = MB[k, j, i];
    if (MA[i, k, 0] != 0 && MB[k, j, 0] != 0)
    {

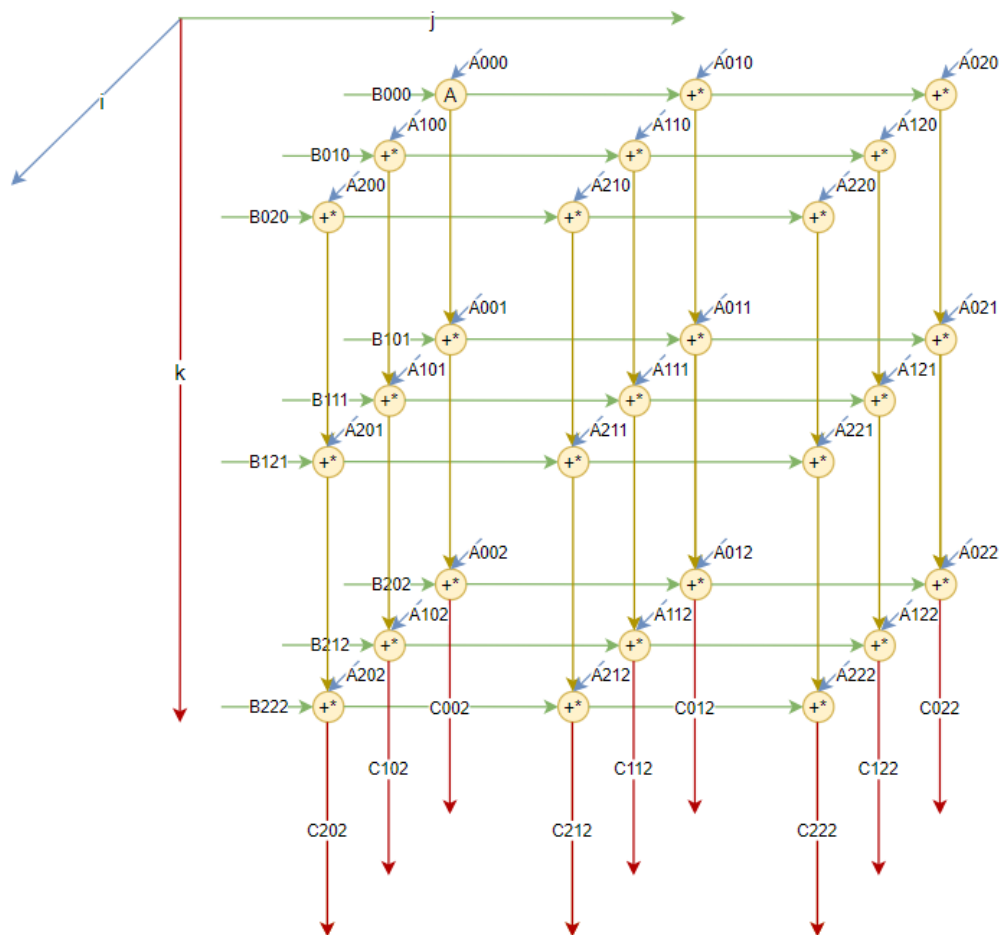
        MY[i, j, k + 1] = MY[i, j, k] + MA[i, k, j] * MB[k, j, i];

    }
    else
    {
        MY[i, j, k + 1] = MY[i, j, k];
    }

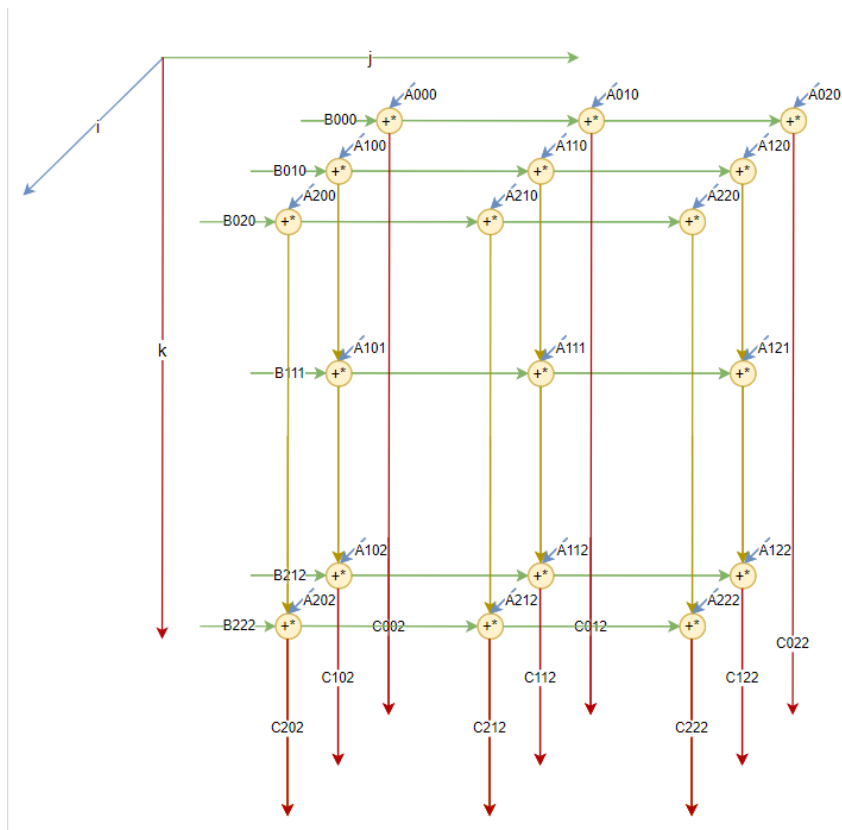
    k++;
    MulMatrixes2();
}
k = 0;
j++;
MulMatrixes2();
}
j = 0;
i++;
MulMatrixes2();
}
}
}

```

Даний алгоритм має меншу кількість операцій через те, що він не виконує операцій над елементами матриць які рівні нулю.



Локалізований ГЗ для множення 2 матриць (для N=3).



Оптимізований Локалізований ГЗ для множення 2 матриць (для N=4).

Скріни роботи програми:

Matrix A	
1	2
2	1

Значення N:

☐ Локально-рекурсивно

Кількість операцій додавання:
4

Кількість операцій множення:
8

Загальна кількість операцій:
12

Matrix B	Result A * B
4 7	4 23
0 8	8 22

Грань2:

7	16	7	12
5	8	5	9
9	16	9	16
13	24	13	23

Грань3:

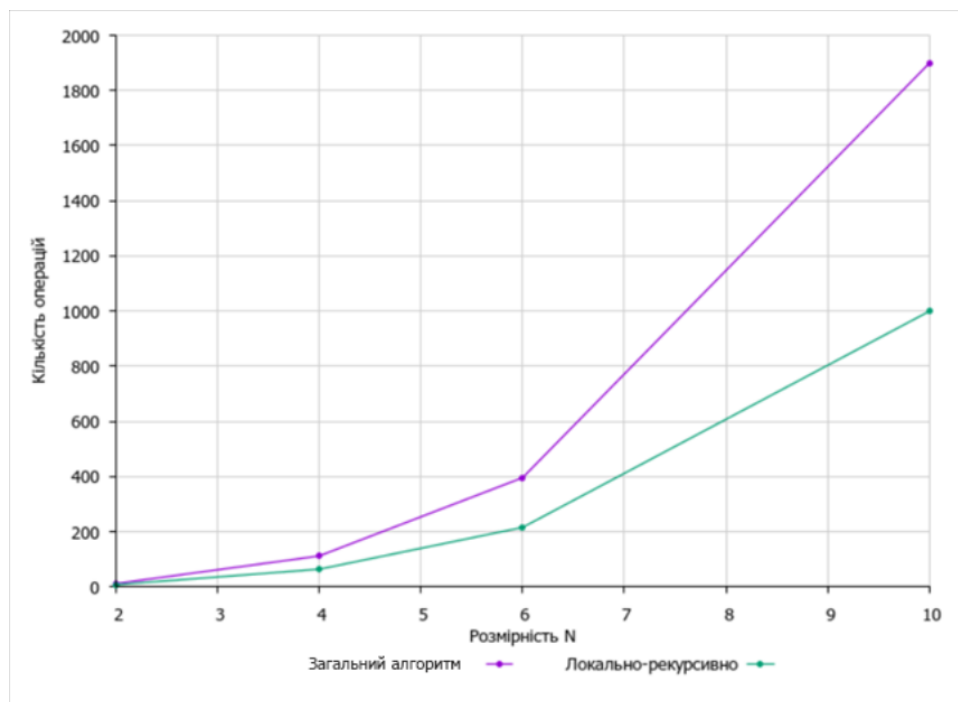
7	16	19	12
5	8	13	9
9	16	13	16
13	24	21	23

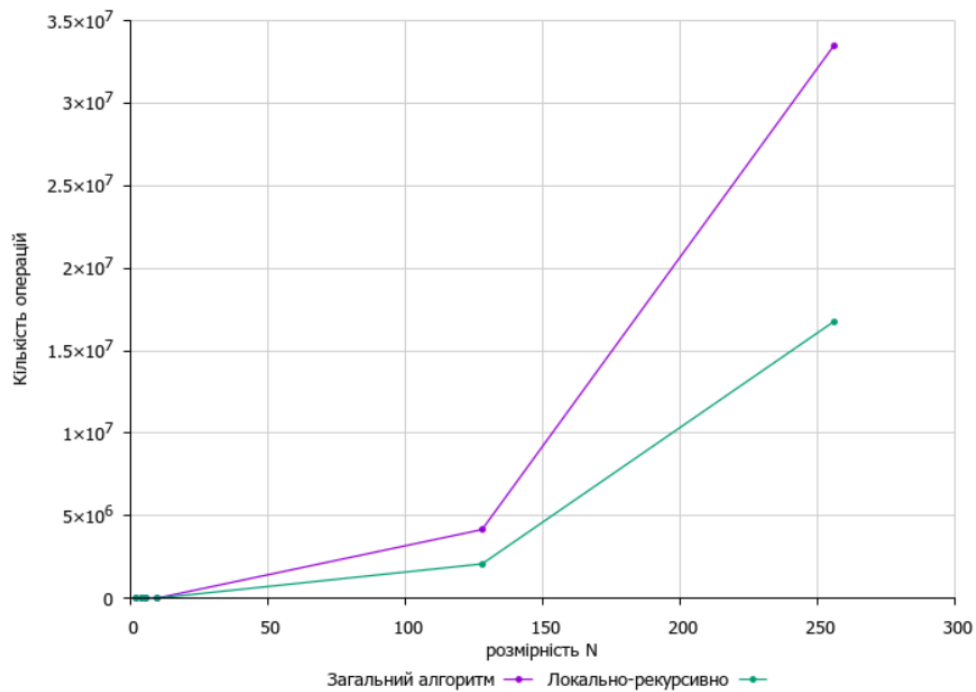
Грань4:

7	16	23	24
5	8	16	18
9	16	15	22
13	24	22	26

Дослідження ефективності створеного алгоритму (за кількістю операцій)

Розмірність	Кількість операцій	
	Загальний алгоритм	Локально-рекурсивно
2	12	8
4	112	64
6	396	216
10	1900	1000
128	4177920	2097152
256	33488896	16777216





Висновки. Як видно з графіків, програма з використанням локально-рекурсивного алгоритму має меншу кількість операцій, і чим більша розмірність N, тим більша різниця між кількістю операцій які виконує той чи інший алгоритм.

Найбільш відомими способами паралельного програмування є :

1. Threads / Processes
2. OpenMP
3. MPI

Також існують і інші маловідомі, застарілі або вузькоспеціалізовані програми, такі як GlobalArrays, PVM і т.д.

Бібліотека MPI надає примітиви для синхронізації і обміну даними.

Основними недоліками MPI є те, що одразу задається кількість процесорів, ця кількість є фіксованою і задається при запуску програми. Також до недоліків можна віднести складність розробки і відносно високі затрати на синхронізацію і обмін даними.

Додаток А. Код проекту.