**Національний університет „Львівська політехніка”**

Кафедра ЕОМ



**Звіт**

з лабораторної роботи №2

з дисципліни: “ Паралельні та розподілені обчислення ”

Варіант 9

Виконав: ст.гр. КІ-33

Лобай Р.І

Перевірив: викладач

Козак Н.Б

Львів 2020

**Мета:** 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:
   1. Кадри.
   2. Програми з одноразовим присвоєнням.
   3. Рекурсивні рівняння.
   4. Графи залежностей.

**Завдання:** Запропонувати та реалізувати локально-рекурсивний алгоритм обчислення виразу:

 ,

де А та В матриці з елементами  та , відповідно(), тобто:

 () .

Тип вхідних послідовностей визначається згідно варіанту.

Матриця А задається однозначно і залежить лише від розмірності даних.

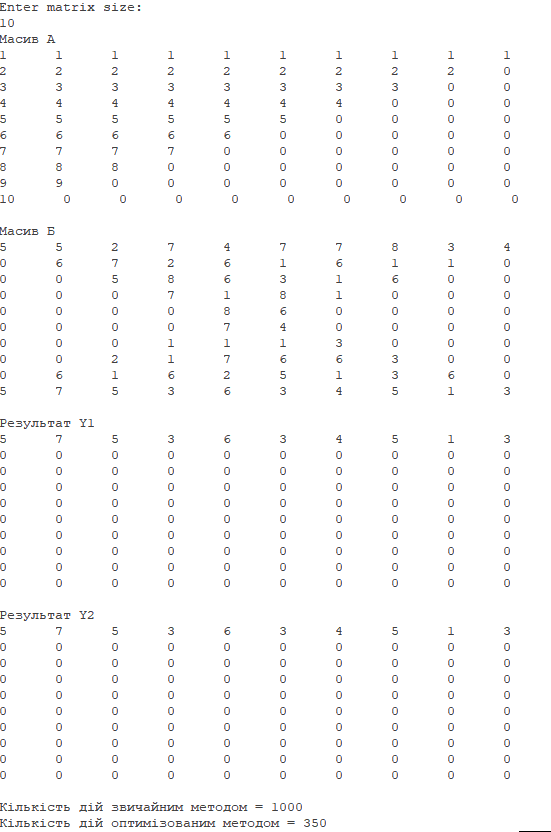
Для матриці В: заштрихована область – довільні цілі числа, відмінні від нуля, а не заштрихована область – нулі.

|  |  |  |
| --- | --- | --- |
| **варіант**  **№** | **Тип матриці А** | **Тип матриці В** |
| **9** | 111…111  222…220  333…300  …  n000…00 |  |

**Графи залежностей (n = 4)**

|  |  |
| --- | --- |
| **Локалізований граф залежностей** | **Оптимізований граф залежностей** |
|  |  |

Приклад роботи з розміром 10:



Як бачимо з результату роботи використовуючи оптимізований метод ми зменшуємо кількість виконуваниї дій.

Код програми

using System;

namespace Lab2

{

class Program

{

static int countsimple = 0;

static int countoptimiz = 0;

static void Main(string[] args)

{

Console.OutputEncoding = System.Text.Encoding.Unicode;

string N;

Console.WriteLine("Enter matrix size: ");

N = Console.ReadLine();

int n = System.Convert.ToInt32(N);

int[,] arrA = new int[n,n];

int[,] arrB = new int[n, n];

//генерація матриці А

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

{

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

{

if (i+j >=n)

{

arrA[i, j] = 0;

} else

{

arrA[i, j] = i + 1;

}

}

}

//генерація матриці B

Random rand = new Random();

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

{

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

{

if ((i == j) || (i < j) && (i + j < n) || (i > j) && (i + j > n - 2))

{

arrB[i, j] = rand.Next(1,9);

}else

{

arrB[i, j] = 0;

}

}

}

int[,] ressimple = MulMatrixSimple(arrA,arrB,n);

int[,] resoptimiz = MulMatrixOptimiz(arrA, arrB, n);

PrintArray("Масив А",arrA, n);

PrintArray("Масив Б",arrB, n);

PrintArray("Результат Y1 ",ressimple,n);

PrintArray("Результат Y2",resoptimiz, n);

Console.WriteLine("Кількість дій звичайним методом = " + countsimple);

Console.WriteLine("Кількість дій оптимізованим методом = " + countoptimiz);

Console.ReadKey();

}

static int[,] MulMatrixSimple(int [,]A, int [,]B, int N)

{

int[,] res = new int[N,N];

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

{

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

{

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

{

res[i, j] = A[i, k] \* B[k, j];

countsimple += 1;

}

}

}return res;

}

static int[,] MulMatrixOptimiz(int[,] A, int[,] B, int N)

{

int[,] res = new int[N, N];

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

{

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

{

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

{

if ((i + j < N) && ((i == j) || (i < j) && (i + j < N) || (i > j) && (i + j > N - 2)))

{

res[i, j] = A[i, k] \* B[k, j];

countoptimiz += 1;

}

}

}

}

return res;

}

static void PrintArray(string name, int [,]arr,int N)

{

Console.WriteLine(name);

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

{

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

{

Console.Write(arr[i, j] + " ");

}

Console.WriteLine();

}

Console.WriteLine();

}

}

}

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