Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра ЕОМ



**Звіт**

з лабораторної роботи №2

з дисципліни: “ Паралельні та розподілені обчислення”

на тему:

“ **ПАРАЛЕЛЬНЕ ПРЕДСТАВЛЕННЯ АЛГОРИТМІВ.**”

Виконав: ст. гр. КІ-34

Гриценко В.В.

Прийняв: викл.

Козак Н.Б.

Львів

2020

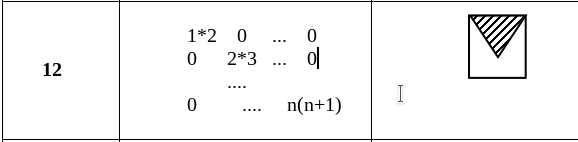
**Тема:** Паралельне представлення алгоритмів.

**Мета роботи:** 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.

А В



3

**Послідовність виконання роботи:**

1. Програма з одноразовим присвоюванням.

Програма об’єднана з програмою реалізації оптимізованого локально-рекурсивного алгоритма, і подана в пункті 6.

1. Рекурсивні рівняння: *Cij(k+1)=Cij(k)+Aij(k)\*Bij(k)*, де *Aij(k)=A[i][j], Bij(k)=B[i][j],* *k* - індекс рекурсії.
2. Граф залежностей(n=3):



Рис 1. Неоптимізований ГЗ

1. Оптимізований граф залежностей(n=3):



Рис. 2 Оптимізований ГЗ

При використанні неоптимізованого локального рекурсивного алгоритму кількість арифметичних операцій приблизно у п’ять разів більша, ніж при роботі алгоритму, де усунуто зайві обчислення. Приріст продуктивності зростає при збільшенні кількості вхідних даних.

**Текст програми:**

#include <iostream>

#include <vector>

using namespace std;

void printMatrix(vector<vector<int>>& matrix);

static int n;

int main()

{

srand(time(nullptr));

cout << "Enter matrices dimension: ";

cin >> n;

vector<vector<int>> A(n, vector<int>(n, 0)),

B(n, vector<int>(n, 0));

// matrices initialization

// A matrix

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

A[i][i] =(i+1) \* (i+2); // due to counting, that starts at 0 value

cout << "A: " << endl;

printMatrix(A);

// B matrix

for (int i = 0; i < n / 2; ++i)

for (int j = i; j < n - i; ++j)

B[i][j] = rand() % 5 + 1; // Bij = [1..10]

cout << endl << "B: " << endl;

printMatrix(B);

int l = 0;

vector<vector<vector<int>>> C(

n,

vector<vector<int>>(

n,

vector<int>(n + 1, 0)

)); // actually there is a C[n][n][n + 1]

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

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

{

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

{

C[i][j][k + 1] = C[i][j][k] + A[i][k] \* B[k][j];

l += 2;

}

l--;

}

cout << endl << "Normal algorithm: " << endl;

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

{

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

{

if (C[i][j][n] / 10 < 1)

{

cout << " " << C[i][j][n];

}

else

{

cout << " " << C[i][j][n];

}

}

cout << endl;

}

cout << endl << "Number of operations: " << l << endl << endl;

// ------------------------------------------------

int p = 0;

// TA[n][n][n + 1]

// TB[n][n][n + 1]

// Co[n][n][n + 1]

vector<vector<vector<int>>>

TA(n,

vector<vector<int>>(

n,

vector<int>(n + 1, 0)

)),

TB(n,

vector<vector<int>>(

n,

vector<int>(n + 1, 0)

)),

Co(n,

vector<vector<int>>(

n,

vector<int>(n + 1, 0)

));

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

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

{

TA[i][j][0] = A[i][j];

TB[i][j][0] = B[i][j];

}

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

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

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

{

TA[i][k][j + 1] = TA[i][k][j];

TB[k][j][i + 1] = TB[k][j][i];

if(TA[i][k][0] != 0 && TB[k][j][0] != 0)

{

Co[i][j][k + 1] = Co[i][j][k] + TA[i][k][j] \* TB[k][j][i];

p += 2;

}

else

Co[i][j][k + 1] = Co[i][j][k];

}

cout << "Optimized algorithm: " << endl;

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

{

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

{

if (Co[i][j][n] / 10 < 1)

{

cout << " " << Co[i][j][n];

}

else

{

cout << " " << Co[i][j][n];

}

}

cout << endl;

}

cout << endl << "Number of operations: " << p << endl << endl;

return(0);

}

void printMatrix(vector<vector<int>>& matrix)

{

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

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

cout << " " << matrix[i][j];

}

cout << endl;

}

}

**Результат виконання програми:**

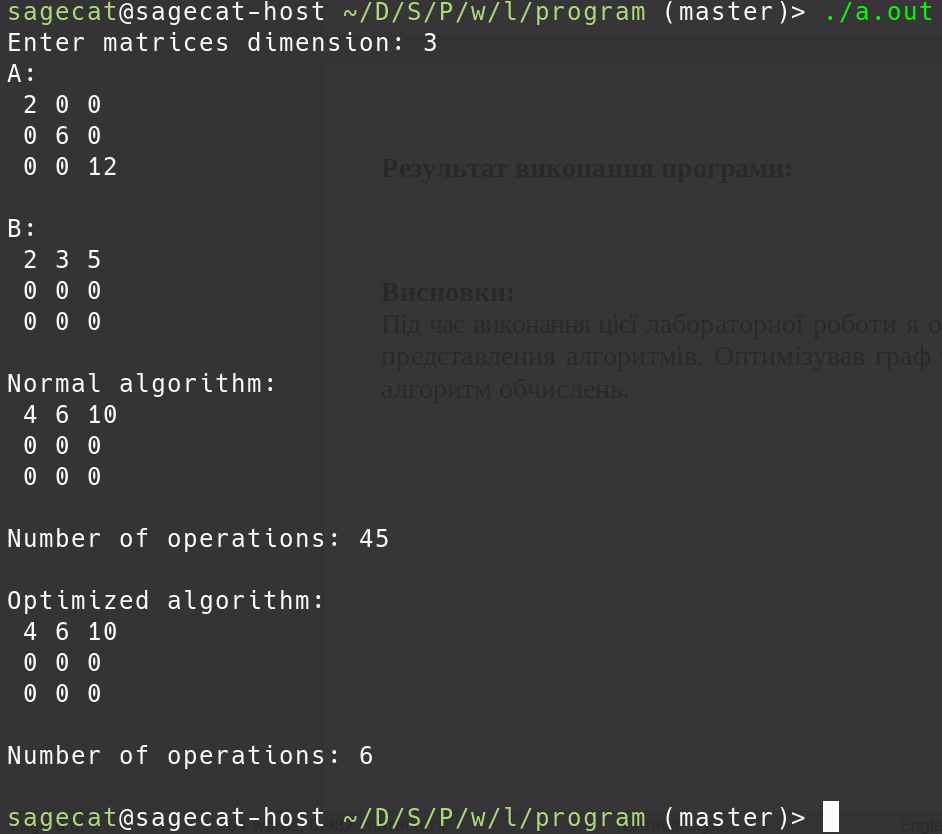
****

Рис 3. Результат роботи програми

**Висновки:**

Під час виконання цієї лабораторної роботи я ознайомився з можливостями паралельного представлення алгоритмів. Оптимізував граф залежностей і отримав високо ефективний алгоритм обчислень.