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**”

Варіант 6

Виконав: студент .гр. КІ-33

Котик В.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(), тобто:

 () .

Тип вхідних послідовностей визначається згідно варіанту.

|  |  |  |
| --- | --- | --- |
| **6** | n 0 .0 ... 0  n-1 n 0 . ..0  n-2 n-1 n ....  ..,  1 2 3 … n |  |

**ПОРЯДОК ВИКОНАННЯ РОБОТИ**

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





 ;

*j* – індекс рекурсії.



Рис. 1 . Граф залежностей звичайного множення матриць

*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.



Рис. 2 . Граф залежностей з локальним зв'язком

*Для матриці розміром 3х3 при виконанні множення матриці за алгоритмом одноразового присвоєння потрібно виконати 45 арифметичних операцій.*

*А при виконанні множення за локально-рекурсивним алгоритм кількість арифметичних операцій дорівнює 18.*

**Код програми:**

#include <iostream>

#include <vector>

#include <iomanip>

#include <random>

using namespace std;

void PrintArr(vector<vector<int>> x, int n)

{

cout << "-----------------------------" << endl;

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

{

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

{

cout << x[i][j] << "\t";

}

cout << endl;

}

cout << "-----------------------------" << endl;

}

int count1 = 0;

int count2 = 0;

vector<vector<int>> Mul1(vector<vector<int>> x, vector<vector<int>> y, int n)

{

vector<vector<int>> z(n, vector<int>(n));

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

{

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

{

z[i][j] = 0;

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

{

z[i][j] += x[i][k] \* y[k][j];

count1 += 2;

}

count1 -= 1;

}

}

return z;

}

vector<vector<int>> z;

void Mul2(vector<vector<int>> x, vector<vector<int>> y, int n);

int main()

{

srand(time(NULL));

cout << "Input N :";

int n;

cin >> n;

vector<vector<int>> a(n, vector<int>(n));

vector<vector<int>> b(n, vector<int>(n));

z.resize(n);

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

z[i].resize(n);

// заповнення матриці А

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

{

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

{

if (j <= i)

a[i][j] = n - i + j;

else

{

a[i][j] = 0;

}

}

}

cout << "-----------------------------" << endl;

cout << "Output matrix A" << endl;

PrintArr(a, n);

// заповнення матриці B

int end = n - 2;

int one = 1;

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

{

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

{

if (j > end)

b[i][j] = 1;

else

b[i][j] = 1+ rand() % 9;

}

end -= one;

if (end < (n / 2))

one = -one;

}

cout << "Output matrix B" << endl;

PrintArr(b, n);

cout << "Output Multiply 1 " <<endl;

PrintArr(Mul1(a, b, n), n);

cout << "Number of arithmetic operations : " << count1 << endl;

cout << "-----------------------------" << endl;

cout << "Output Multiply 2 " << endl;

Mul2(a, b, n);

PrintArr(z, n);

cout << "Number of arithmetic operations : " << count2 << endl;

}

void Mul2(vector<vector<int>> x, vector<vector<int>> y, int n)

{

static int i = 0, j = 0, k = 0;

if (i < n)

{

if (j < n)

{

if (k < n)

{

z[i][j] += x[i][k] \* y[k][j];

k++;

count2 += 2;

Mul2(x, y, n);

}

count2 -= 1;

k = 0;

j++;

Mul2(x, y, n);

}

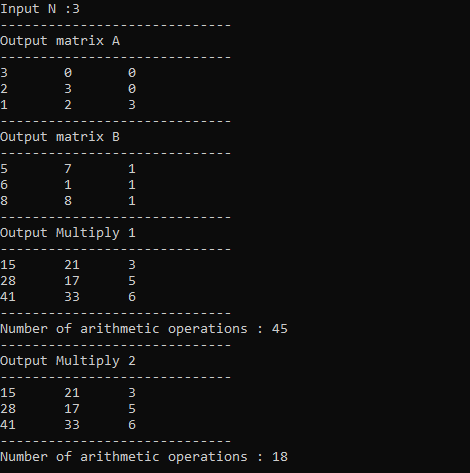
j = 0;

i++;

Mul2(x, y, n);

}

}

****

**Рис.1. Результат виконання програми**

**Висновок:** На цій лабораторній роботі я вивчив можливості паралельного представлення алгоритмів. Набув навиків такого представлення