Міністерство освіти і науки

Національний університет “Львівська політехніка”

Кафедра ЕОМ



**Звіт**

з лабораторної роботи № 2

з дисципліни: “Паралельні та розподілені обчислення”

на тему: “Паралельне представлення алгоритмів”

Виконала:

ст. гр. КІ-34

Шугай В. В.

Перевірив:

Козак Н. Б.

Львів – 2020

**Мета лабораторної роботи**

Вивчити можливості паралельного представлення алгоритмів. Набути навиків такого представлення.

**Завдання**

Запропонувати та реалізувати локально-рекурсивний алгоритм обчислення виразу:

,

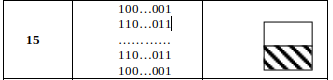
де А та В матриці з елементами та , відповідно(), тобто:

() .

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.



**Графи залежностей (n = 4)**

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

**Текст програми**

**Main.java**

**package com.ki34.shuhai.lab7;**

import com.ki34.shuhai.lab7.controller.Calculation;

import com.ki34.shuhai.lab7.view.Menu;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner input = new Scanner(System.*in*);

System.*out*.println("Input N:");

Calculation calculation = new Calculation(input.nextInt());

Menu menu = new Menu(calculation);

menu.show();

}

}

**Menu.java**

**package com.ki34.shuhai.lab7.view;**

import com.ki34.shuhai.lab7.controller.Calculation;

import java.util.Arrays;

import java.util.LinkedHashMap;

import java.util.Map;

import java.util.Scanner;

public class Menu {

private Scanner input = new Scanner(System.*in*);

private Calculation calculation;

private Map<String, String> mainMenu;

private Map<String, Printable> methodsMenu;

public Menu(Calculation calculation) {

this.calculation = calculation;

mainMenu = new LinkedHashMap<>();

mainMenu.put("1","1 - Input matrix B");

mainMenu.put("2","2 - Random matrix B");

mainMenu.put("Q","Q - Exit");

methodsMenu = new LinkedHashMap<>();

methodsMenu.put("1", this::inputMatrix);

methodsMenu.put("2", this::randomMatrix);

}

private void printMenu(){

System.*out*.println("Menu:");

for (String str: mainMenu.values()){

System.*out*.println(str);

}

}

public void show(){

String keyMenu;

do {

printMenu();

keyMenu = input.nextLine().toUpperCase();

try {

methodsMenu.get(keyMenu).print();

} catch (Exception e){

System.*out*.println("input exception");

}

}while (!keyMenu.equals("Q"));

}

private void inputMatrix(){

System.*out*.println("Matrix B:");

calculation.inputMatrixB();

calculation.showMatrix(calculation.getMatrixB());

printMatrixAAndResult();

}

private void randomMatrix(){

System.*out*.println("Matrix B:");

calculation.randomFillMatrixB();

calculation.showMatrix(calculation.getMatrixB());

printMatrixAAndResult();

}

private void printMatrixAAndResult(){

System.*out*.println("Matrix A:");

calculation.fillMatrixA();

calculation.showMatrix(calculation.getMatrixA());

System.*out*.println("Result:");

calculation.mulMatrix();

calculation.showMatrix(calculation.getMatrixC());

}

}

**Printable.java**

**package com.ki34.shuhai.lab7.view;**

@FunctionalInterface

public interface Printable {

void print();

}

**Calculation.java**

**package com.ki34.shuhai.lab7.controller;**

public class Calculation {

private int[][] matrixA;

private int[][] matrixB;

private int[][] matrixC;

private int n;

public Calculation(int n) {

this.n=n;

this.matrixA = new int[n][n];

this.matrixB = new int[n][n];

this.matrixC = new int[n][n];

}

private FillMatrix fillMatrix = new FillMatrix();

public int[][] getMatrixB() {

return matrixB;

}

public void fillMatrixA(){

fillMatrix.matrixA(matrixA,n);

}

public void randomFillMatrixB(){

fillMatrix.randomMatrixB(matrixB,n);

}

public void inputMatrixB(){

fillMatrix.inputMatrixB(matrixB,n);

}

public void showMatrix(int [][] matrix){

for (int[] ints : matrix) {

for (int j = 0; j < matrix.length; j++) {

System.*out*.print(ints[j] + " ");

}

System.*out*.println();

}

}

public int[][] getMatrixC() {

return matrixC;

}

public void mulMatrix(){

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

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

matrixC[i][j] = 0;

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

matrixC[i][j] = matrixC[i][j] + matrixA[i][k] \* matrixB[k][j];

}

}

}

}

public int[][] getMatrixA() {

return matrixA;

}

}

**FillMatrix.java**

**package com.ki34.shuhai.lab7.controller;**

import java.util.Scanner;

import java.util.SplittableRandom;

public class FillMatrix {

public void matrixA(int [][] matrix,int size){

int temp = 0;

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

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

if (temp >= j)

{

matrix[i][j] = 1;

}

else if (j>=size-temp-1){

matrix[i][j] = 1;

}

else{

matrix[i][j] = 0;

}

}

boolean isTopHalf = i+1 < Math.*round*(size / 2.0);

if (isTopHalf)

{

temp++;

}

else if (!(i + 1 <= size / 2))

{

temp--;

}

// temp++;

}

}

public void randomMatrixB(int [][]matrix,int size){

SplittableRandom random = new SplittableRandom();

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

boolean isTopHalf = i+1 < Math.*round*(size / 2.0);

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

if (isTopHalf){

matrix[i][j]=0;

}

else if (!(i + 1 <= size / 2)) {

matrix[i][j]= random.nextInt(0,100);

}

}

}

}

public void inputMatrixB(int [][]matrix,int size){

Scanner input = new Scanner(System.*in*);

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

boolean isTopHalf = i+1 < Math.*round*(size / 2.0);

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

if (isTopHalf){

matrix[i][j]=0;

}

else if (!(i + 1 <= size / 2)) {

System.*out*.print("b["+i+"]["+j+"] = ");

matrix[i][j]= input.nextInt();

}

}

}

}

}

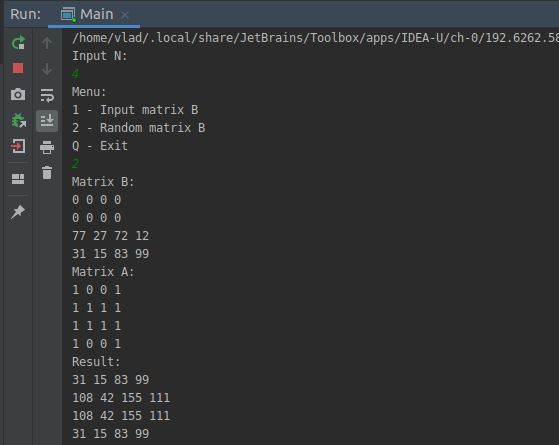
**Результат роботи програми**

Рис. 1 Результат виконання програми

**Висновок**

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