Міністерствоосвіти і науки України

Національний університет „Львівська політехніка”

Кафедра ЕОМ



**Лабораторна робота №1**

З дисципліни:”Паралельні та розподілені обчислення ”

На тему:” ВИКОРИСТАННЯ ФУНКЦІОНАЛЬНОЇ ДЕКОМПОЗИЦІЇ ДЛЯ РОЗВ’ЯЗКУ ОБЧИСЛЮВАЛЬНИХ ЗАДАЧ”

Виконав: ст.гр. КІ-34

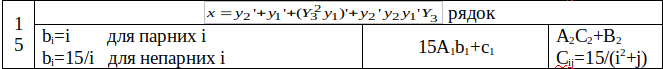
Шугай В. В. Прийняв: Козак Н.Б.

Львів 2020

МЕТА РОБОТИ.

Вивчити методи декомпозицій задач. Набути навиків розв’язування задач з використанням функціональної декомпозиції.

**Варіант 15**



ЗАВДАННЯ.

Використовуючи метод функціональної декомпозиції, розробити алгоритм обчислення запропонованого матрично-векторного виразу, який би враховував можливість паралельного виконання і був оптимальним з точки зору часових затрат.

На основі створеного алгоритму написати програму яка дозволяє обчислити вираз та ілюструє проведену декомпозицію.

***Декомпозиція***. На цьому етапі вихідна задача аналізується, оцінюється можливість її розпаралелювання. Іноді виграш від розпаралелення може бути незначним, а трудоємкість розробки паралельної програми велика. В цьому випадку перший крок розробки алгоритму виявляється і останнім. Якщо ж ситуація відмінна від описаної, то задача та пов’язані з нею дані розділяються на дрібніші частини – підзадачі і фрагменти структур даних. Особливості архітектури конкретної обчислювальної системи на цьому етапі можуть не враховуватися.

Правила знаходження елементів виразу.

1).Задати\* квадратну матрицю А порядку n. Отримати вектор(стовпець) , де b – вектор-стовпець, елементи якого обраховуються за формулою, згідно варіанту.

2).Задати квадратну матрицю А1 порядку n та вектори-стовпці b1 та c1 з n елементами кожен. Отримати вектор згідно формули, що задається варіантом.

3).Задати квадратні матриці А2 та B2 порядку n. Отримати матрицю , яка залежить від А2, B2 та додатково визначеної матриці С2, елементи якої знаходяться за формулою, вказаною варіантом.

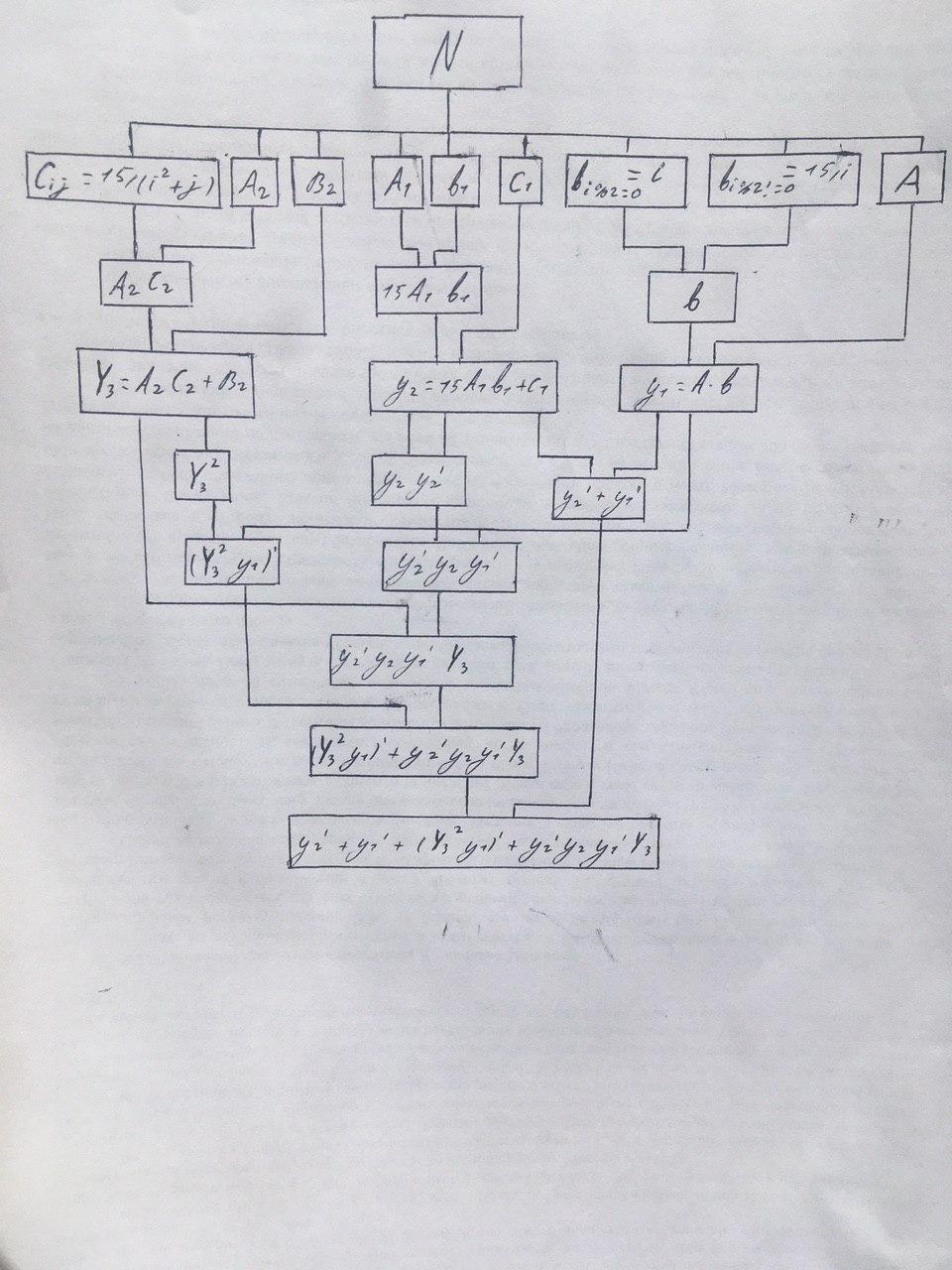
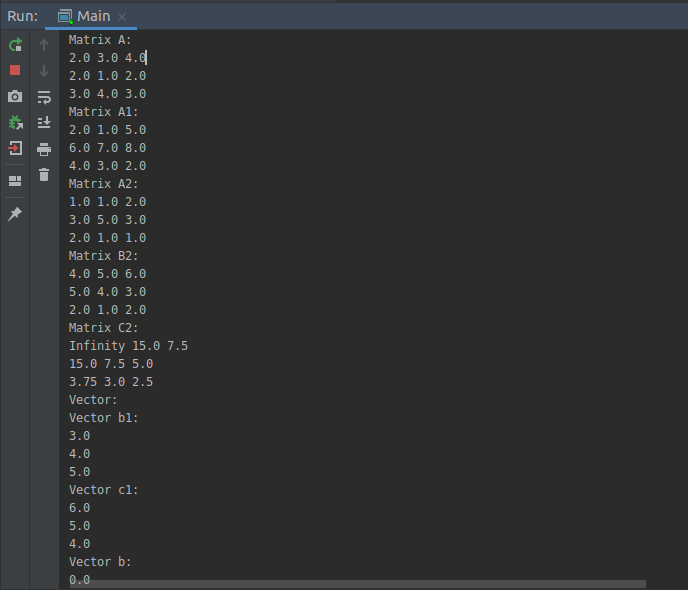


Рис.1. Схема декомпозиції поставленої задачі

Рис. 2 Введені користувачем дані.

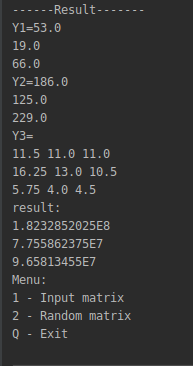


Рис.3. Результат обчислень виразу

Висновок: виконуючи дану лабораторну роботу я вивчив метод декомпозицій задач. Набув навиків розв’язування задач з використанням функціональної декомпозиції.

Лістинги програми:

(Програма написана мовою java)

Main.java

package com.ki34.shuhai.lab7;

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

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*.print("Input N (matrix size):");

Controller controller = new Controller(input.nextInt());

Menu menu = new Menu(controller);

menu.show();

}

}

Menu.java

package com.ki34.shuhai.lab7.view;

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

import java.util.LinkedHashMap;

import java.util.Map;

import java.util.Scanner;

import java.util.SplittableRandom;

public class Menu {

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

private Controller controller;

private Map<String, String> mainMenu;

private Map<String, Printable> methodsMenu;

public Menu(Controller controller) {

this.controller = controller;

mainMenu = new LinkedHashMap<>();

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

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

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.next().toUpperCase();

try {

methodsMenu.get(keyMenu).print();

} catch (Exception e){

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

}

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

}

private void randomMatrix(){

controller.randomMatrix();

printAllMatrix();

controller.allStagesCalculate();

printResult();

}

private void inputMatrix(){

controller.inputMatrix();

printAllMatrix();

controller.allStagesCalculate();

printResult();

}

private void printResult(){

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

System.*out*.print("Y1=");

printMatrix(controller.getY1());

System.*out*.print("Y2=");

printMatrix(controller.getY2());

System.*out*.println("Y3=");

printMatrix(controller.getY3());

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

printMatrix(controller.getTemp\_vector());

}

private void printAllMatrix(){

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

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

printMatrix(controller.getMatrixA());

System.*out*.println("Matrix A1:");

printMatrix(controller.getMatrixA1());

System.*out*.println("Matrix A2:");

printMatrix(controller.getMatrixA2());

System.*out*.println("Matrix B2:");

printMatrix(controller.getMatrixB2());

System.*out*.println("Matrix C2:");

printMatrix(controller.getMatrixC2());

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

System.*out*.println("Vector b1:");

printMatrix(controller.getVectorb1());

System.*out*.println("Vector c1:");

printMatrix(controller.getVectorc1());

System.*out*.println("Vector b:");

printMatrix(controller.getVectorb());

}

private void printMatrix(double[] matrix){

for (double v : matrix) {

System.*out*.println(v + " ");

}

}

private void printMatrix(double[][] matrix){

for (double[] doubles : matrix) {

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

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

}

System.*out*.println();

}

}

}

Printable.java

package com.ki34.shuhai.lab7.view;

@FunctionalInterface

public interface Printable {

void print();

}

Controller.java

package com.ki34.shuhai.lab7.controller;

import java.util.Scanner;

import java.util.SplittableRandom;

public class Controller {

private int n;

private SplittableRandom random = new SplittableRandom();

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

private double[][] temp\_matrix;

private double[][] matrixA;

private double[][] matrixA1;

private double[][] matrixA2;

private double[][] matrixB2;

private double[][] matrixC2;

private double[][] y3;

private double[] temp\_vector;

private double[] vectorb1;

private double[] vectorc1;

private double[] vectorb;

private double[] y1;

private double[] y2;

public Controller(int n) {

this.n = n;

matrixA = new double[n][n];

matrixA1 = new double[n][n];

matrixA2 = new double[n][n];

matrixB2 = new double[n][n];

temp\_matrix = new double[n][n];

y3 = new double[n][n];

matrixC2 = calculationC2();

temp\_vector = new double[n];

vectorb1 = new double[n];

vectorc1 = new double[n];

y1 = new double[n];

y2 = new double[n];

vectorb = calculationVectorB();

}

public double[][] calculationC2() {

double[][] matrix = new double[n][n];

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

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

matrix[i][j] = 15 / (Math.*pow*((double) i, 2) + (double) j);

}

}

return matrix;

}

public double[] calculationVectorB() {

double[] vector = new double[n];

for (int i = 0; i < vector.length; i++) {

vector[i] = i % 2 == 0 ? i : (double) (15 / i);

}

return vector;

}

public void randomMatrix() {

calculationC2();

for (int i = 0; i < matrixA.length; i++) {

vectorb1[i] = random.nextInt(0, 100);

vectorc1[i] = random.nextInt(0, 100);

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

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

matrixA1[i][j] = random.nextInt(0, 100);

matrixA2[i][j] = random.nextInt(0, 100);

matrixB2[i][j] = random.nextInt(0, 100);

}

}

allStagesCalculate();

}

public void firstLevelCalculate() {

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

y1[i] = 0;

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

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

temp\_matrix[i][j] = matrixA2[i][k] \* matrixC2[k][j]; // A2\*C2

}

y1[i] += matrixA[i][j] \* vectorb[j]; //y1=A\*b

temp\_vector[i] = 15 \* matrixA[i][j] \* vectorb1[i]; //15A1\*b1

}

}

}

private void secondLevelCalculate() {

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

y2[i] = 0;

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

y3[i][j] = 0;

y3[i][j] += (temp\_matrix[i][j]) + matrixB2[i][j]; //Y3=A2\*C2+B2

}

y2[i] += temp\_vector[i] + vectorc1[i]; // Y2=15A1\*b1+c1

}

}

private void thirdLevelCalculate() {

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

vectorb1[i] = 0;

temp\_vector[i] = 0;

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

temp\_matrix[i][j] = 0;

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

temp\_matrix[i][j] += y3[i][k] \* y3[k][j]; // temp\_matrix = Y3 \* Y3

}

}

vectorb1[0] += y2[i] \* y2[i]; //vectorb1=y2'\*y2

}

}

public void fourLevelCalculate() {

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

vectorc1[i] = 0;

temp\_vector[i] = 0;

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

temp\_vector[i] += temp\_matrix[i][j] \* y1[i]; // A1 = Y3 \* Y3 \* y1

}

vectorc1[i] += vectorb1[0] \* y1[i]; //vectorc1=y2'\*y2\*y1'

}

}

public void fiveLevelCalculate() {

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

vectorb1[i] = 0;

vectorb[i] = 0;

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

vectorb1[i] += y3[i][j] \* vectorc1[i]; // vectorb1=Y3\*y2'\*y2\*y1'

}

vectorb[i] += y2[i] + y1[i];//y2'+y1'

}

}

public void sixLevelCalculate() {

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

vectorc1[i] = 0;

vectorc1[i] += vectorb1[i] + temp\_vector[i];// vectorb1+temp

}

}

public void sevenLevelCalculate(){

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

temp\_vector[i] = 0;

temp\_vector[i] += vectorc1[i] + vectorb[i];// vectorb1+temp

}

}

public void allStagesCalculate(){

firstLevelCalculate();

secondLevelCalculate();

thirdLevelCalculate();

fourLevelCalculate();

fiveLevelCalculate();

sixLevelCalculate();

sevenLevelCalculate();

}

public void inputMatrix() {

calculationC2();

for (int i = 0; i < matrixA.length; i++) {

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

System.*out*.print("Input [" + i + "][" + j + "] element of matrix A:");

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

}

}

for (int i = 0; i < matrixA1.length; i++) {

for (int j = 0; j < matrixA1.length; j++) {

System.*out*.print("Input [" + i + "][" + j + "] element of matrix A1:");

matrixA1[i][j] = input.nextDouble();

}

}

for (int i = 0; i < matrixA2.length; i++) {

for (int j = 0; j < matrixA2.length; j++) {

System.*out*.print("Input [" + i + "][" + j + "] element of matrix A2:");

matrixA2[i][j] = input.nextDouble();

}

}

for (int i = 0; i < matrixB2.length; i++) {

for (int j = 0; j < matrixB2.length; j++) {

System.*out*.print("Input [" + i + "][" + j + "] element of matrix B2:");

matrixB2[i][j] = input.nextDouble();

}

}

for (int i = 0; i < vectorb1.length; i++) {

System.*out*.println("Input [" + i + "] element of matrix b1:");

vectorb1[i] = input.nextDouble();

}

for (int i = 0; i < vectorc1.length; i++) {

System.*out*.println("Input [" + i + "] element of matrix c1:");

vectorc1[i] = input.nextDouble();

}

}

public double[][] getMatrixA() {

return matrixA;

}

public double[][] getMatrixA1() {

return matrixA1;

}

public double[][] getMatrixA2() {

return matrixA2;

}

public double[][] getMatrixB2() {

return matrixB2;

}

public double[] getVectorb1() {

return vectorb1;

}

public double[] getVectorc1() {

return vectorc1;

}

public double[][] getMatrixC2() {

return matrixC2;

}

public double[] getVectorb() {

return vectorb;

}

public double[][] getY3() {

return y3;

}

public double[] getY1() {

return y1;

}

public double[] getY2() {

return y2;

}

public double[] getTemp\_vector() {

return temp\_vector;

}

}