|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика, ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ и системы  
 управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/07 Интеллектуальные системы анализа,**

**обработки и интерпретации больших данных**

**Отчет**

**по лабораторной работе №6**

**Название:** Коллекции

**Дисциплина:** Языки программирования для работы с большими данными

Вариант: 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-22М |  |  | Н.А. Аскерова |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | П.В. Степанов |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

**Вариант 1**

2. Списки (стеки, очереди) I(1..n) и U(1..n) содержат результаты n измерений тока и напряжения на неизвестном сопротивлении R. Найти приближенное число R методом наименьших квадратов. Листинг 1 – Код программы

import java.util.ArrayList;

import java.util.Arrays;

public class main {

public static void main(String[] args) {

ArrayList<Double> I = new ArrayList<>(Arrays.asList(1.2, 2.4, 4.8, 5.6, 6.4)); // current measurements

ArrayList<Double> U = new ArrayList<>(Arrays.asList(2.4, 4.8, 9.6, 11.2, 12.8)); // voltage measurements

int n = I.size();

double sumI = I.stream().mapToDouble(Double::doubleValue).sum();

double sumU = U.stream().mapToDouble(Double::doubleValue).sum();

double sumIU = 0;

double sumI2 = 0;

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

sumIU += I.get(i) \* U.get(i);

sumI2 += Math.pow(I.get(i), 2);

}

double R = (n \* sumIU - sumI \* sumU) / (n \* sumI2 - Math.pow(sumI, 2));

System.out.println("Approximate value of resistance R: " + R);

}

}

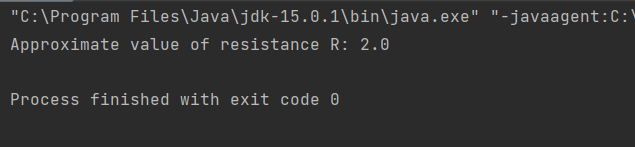


Рисунок 1 – Результат работы программы

3. С использованием множества выполнить попарное суммирование произвольного конечного ряда чисел по следующим правилам: на первом этапе суммируются попарно рядом стоящие числа, на втором этапе суммируются результаты первого этапа и т.д. до тех пор, пока не останется одно число.

Листинг 2 – Код программы

import java.util.HashSet;

import java.util.Set;

public class Main {

public static void main(String[] args) {

Set<Double> numbers = new HashSet<>();

numbers.add(2.0);

numbers.add(3.0);

numbers.add(4.0);

numbers.add(5.0);

while (numbers.size() > 1) {

Set<Double> newNumbers = new HashSet<>();

Double previous = null;

for (Double number : numbers) {

if (previous != null) {

//System.out.println(number);

newNumbers.add(previous + number);

//System.out.println(newNumbers);

previous = null;

} else {

previous = number;

}

}

if (previous != null) {

newNumbers.add(previous);

}

numbers = newNumbers;

}

System.out.println("Result: " + numbers.iterator().next());

}

}

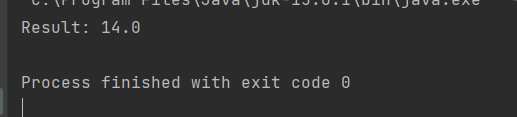


Рисунок 2 – Результат работы программы

**Вариант 2**

2. Реализовать класс, моделирующий работу N-местной автостоянки. Машина подъезжает к определенному месту и едет вправо, пока не встретится свободное место. Класс должен поддерживать методы, обслуживающие приезд и отъезд машины.

Листинг 3 – Код программы

import java.util.Arrays;

class ParkingLot {

private boolean[] spaces;

public ParkingLot(int n) {

spaces = new boolean[n];

Arrays.fill(spaces, false);

}

public int park() {

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

if (!spaces[i]) {

spaces[i] = true;

return i;

}

}

return -1;

}

public boolean leave(int spot) {

if (spot >= 0 && spot < spaces.length && spaces[spot]) {

spaces[spot] = false;

return true;

}

return false;

}

}

public class Main {

public static void main(String[] args) {

// create a parking lot with 5 spaces

ParkingLot lot = new ParkingLot(5);

// park 3 cars

int spot1 = lot.park();

int spot2 = lot.park();

int spot3 = lot.park();

// try to park another car (should fail)

int spot4 = lot.park();

if (spot1 >= 0) {

System.out.println("Car parked in spot " + spot1);

}

if (spot2 >= 0) {

System.out.println("Car parked in spot " + spot2);

}

if (spot3 >= 0) {

System.out.println("Car parked in spot " + spot3);

}

if (spot4 >= 0) {

System.out.println("Car parked in spot " + spot4);

} else {

System.out.println("Parking lot is full");

}

// vacate the first parking space

boolean success = lot.leave(spot1);

if (success) {

System.out.println("Car left spot " + spot1);

} else {

System.out.println("Spot " + spot1 + " is already empty or out of range");

}

// park another car in the first parking space

int spot5 = lot.park();

if (spot5 >= 0) {

System.out.println("Car parked in spot " + spot5);

} else {

System.out.println("Parking lot is full");

}

}

}

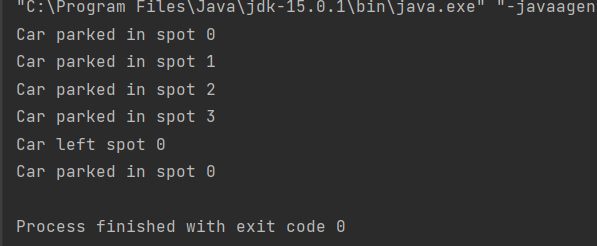


Рисунок 3 – Результат работы программы

3. Во входном файле хранятся две разреженные матрицы А и В. Построить циклически связанные списки СА и СВ, содержащие ненулевые элементы соответственно матриц А и В. Просматривая списки, вычислить: а) сумму S = A + B; б) произведение P = A \* B.  
Листинг 4 – Код программы

import java.util.\*;

class SparseMatrix {

private int n, m;

private Node[] rows;

public SparseMatrix(int[][] data) {

n = data.length;

m = data[0].length;

rows = new Node[n];

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

Node head = new Node(-1, -1, 0);

Node tail = head;

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

if (data[i][j] != 0) {

tail.next = new Node(i, j, data[i][j]);

tail = tail.next;

}

}

rows[i] = head.next;

}

}

public SparseMatrix add(SparseMatrix other) {

if (n != other.n || m != other.m) {

throw new IllegalArgumentException("Matrices must have the same dimensions");

}

int[][] data = new int[n][m];

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

Node p = rows[i], q = other.rows[i];

while (p != null || q != null) {

int j, value;

if (q == null || (p != null && p.j < q.j)) {

j = p.j;

value = p.value;

p = p.next;

} else if (p == null || (q != null && q.j < p.j)) {

j = q.j;

value = q.value;

q = q.next;

} else {

j = p.j;

value = p.value + q.value;

p = p.next;

q = q.next;

}

data[i][j] = value;

}

}

return new SparseMatrix(data);

}

public SparseMatrix multiply(SparseMatrix other) {

if (m != other.n) {

throw new IllegalArgumentException("Matrices must have compatible dimensions");

}

SparseMatrix transposed = other.transpose();

int[][] data = new int[n][other.m];

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

Node p = rows[i];

while (p != null) {

int j = p.j;

Node q = transposed.rows[j];

while (q != null) {

int k = q.j;

data[i][k] += p.value \* q.value;

q = q.next;

}

p = p.next;

}

}

return new SparseMatrix(data);

}

public SparseMatrix transpose() {

int[][] data = new int[m][n];

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

Node p = rows[i];

while (p != null) {

int j = p.j;

data[j][i] = p.value;

p = p.next;

}

}

return new SparseMatrix(data);

}

public void print() {

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

Node p = rows[i];

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

if (p != null && p.j == j) {

System.out.printf("%d ", p.value);

p = p.next;

} else {

System.out.print("0 ");

}

}

System.out.println();

}

}

private static class Node {

int i, j, value;

Node next;

Node(int i, int j, int value) {

this.i = i;

this.j = j;

this.value = value;

}

}

}

class Main {

public static void main(String[] args) {

int[][] aData = {{0, 0, 1}, {0, 2, 0}, {3, 0, 0}};

int[][] bData = {{0, 4, 0}, {0, 0, 5}, {6, 0, 0}};

SparseMatrix a = new SparseMatrix(aData);

SparseMatrix b = new SparseMatrix(bData);

SparseMatrix s = a.add(b);

SparseMatrix p = a.multiply(b);

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

a.print();

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

b.print();

System.out.println("S = A + B:");

s.print();

System.out.println("P = A \* B:");

p.print();

}

}

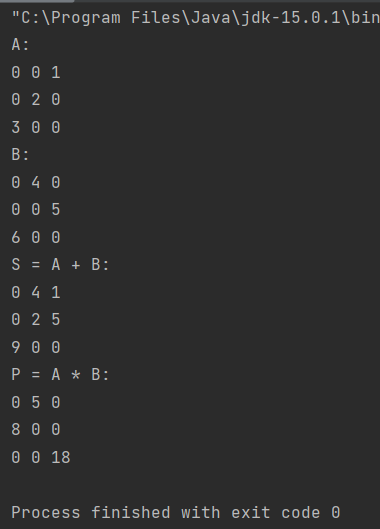


Рисунок 4 – Результат работы программы

**Вывод:** приобретен навык работы с коллекциями.

**Ссылка на репозиторий с программами:** [**https://github.com/nargi3/BigData**](https://github.com/nargi3/BigData)