**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

|  |  |
| --- | --- |
| Подп. и дата |  |
| Инв. № дубл. |  |
| Взам. инв. № |  |
| Подп. и дата |  |
| Инв. №подл |  |

СОГЛАСОВАНО

Доцент департамента программной инженерии, кандидат технических наук

Р.З. Ахметсафина

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

УТВЕРЖДАЮ

Академический руководитель образовательной программы «Программная инженерия», к.т.н., профессор ДПИ ФКН

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В. В. Шилов

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

**ВЕБ ПРИЛОЖЕНИЕ ДЛЯ ГЕНЕРАЦИИ**

**ЗАДАЧ ПО АЛГЕБРЕ**

**Текст программы**

**ЛИСТ УТВЕРЖДЕНИЯ**

**RU.17701729.05.15-01 12 01-1-ЛУ 1**

**Исполнитель**

Студент группы БПИ199

\_\_\_\_\_\_\_\_\_\_\_\_ / К. Пике /

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

**Москва 2021**

**УТВЕРЖДЁН**

**RU.17701729.05.15-01 12 01-1 ЛУ**

|  |  |
| --- | --- |
| Подп. и дата |  |
| Инв. № дубл. |  |
| Взам. инв. № |  |
| Подп. и дата |  |
| Инв. №подл |  |

**ВЕБ ПРИЛОЖЕНИЕ ДЛЯ ГЕНЕРАЦИИ**

**ЗАДАЧ ПО АЛГЕБРЕ**

**Текст программы**

**RU.17701729.05.15-01 12 01-1**

**Листов 44**

**Москва 2021**

[1 ТЕКСТ ПРОГРАММЫ 3](#__RefHeading___Toc5482_1933433957)

[1.1 TasksController.java 3](#__RefHeading___Toc5484_1933433957)

[1.2 Column.java 9](#__RefHeading___Toc5361_3298813295)

[1.3 Matrix.java 12](#__RefHeading___Toc5486_1933433957)

[1.4 Row.java 26](#__RefHeading___Toc4482_423323679)

[1.5 DimKer.java 30](#__RefHeading___Toc4484_423323679)

[1.6 FindEigenvalues.java 33](#__RefHeading___Toc4486_423323679)

[1.7 InverseMatrix.java 35](#__RefHeading___Toc4488_423323679)

[1.8 JordanCanonical.java 37](#__RefHeading___Toc4490_423323679)

[1.9 MatrixAddSubMult.java 40](#__RefHeading___Toc4492_423323679)

[1.10 MatrixPowerN.java 43](#__RefHeading___Toc4494_423323679)

[1.11 MatrixProblem.java 45](#__RefHeading___Toc4496_423323679)

[1.12 MatrixProblemFactory.java 47](#__RefHeading___Toc4498_423323679)

[1.13 MatrixRank.java 49](#__RefHeading___Toc4500_423323679)

[1.14 OrthogonalDiag.java 51](#__RefHeading___Toc4502_423323679)

[1.15 ProblemInterface.java 54](#__RefHeading___Toc4504_423323679)

[1.16 QRdeomposition.java 55](#__RefHeading___Toc4506_423323679)

[1.17 RotationAroundAxis.java 58](#__RefHeading___Toc4508_423323679)

[1.18 SVDdecomposition.java 64](#__RefHeading___Toc4510_423323679)

[1.19 ExprUtils.java 68](#__RefHeading___Toc4512_423323679)

[1.20 AlgebraGeneratorApplication.java 69](#__RefHeading___Toc4514_423323679)

[1.21 TasksDocument.java 70](#__RefHeading___Toc4516_423323679)

[1.22 common.ftlh 73](#__RefHeading___Toc4518_423323679)

[1.23 navbar.ftlh 76](#__RefHeading___Toc4520_423323679)

[1.24 greeting.ftlh 80](#__RefHeading___Toc4522_423323679)

[1.25 problems.ftlh 81](#__RefHeading___Toc4524_423323679)

[1.26 application-dev.properties 82](#__RefHeading___Toc4526_423323679)

[1.27 application.properties 83](#__RefHeading___Toc4528_423323679)

# **ТЕКСТ ПРОГРАММЫ**

## TasksController.java

package com.schuyweiz.algebragenerator.controllers;

import com.schuyweiz.algebragenerator.TasksDocument;

import com.schuyweiz.algebragenerator.tasks.MatrixAddSubMul;

import com.schuyweiz.algebragenerator.tasks.MatrixProblem;

import com.schuyweiz.algebragenerator.tasks.MatrixProblemFactory;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.ContentDisposition;

import org.springframework.http.HttpHeaders;

import org.springframework.http.MediaType;

import org.springframework.http.ResponseEntity;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.\*;

import java.io.File;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.InputStream;

import java.nio.file.Files;

import java.nio.file.Path;

import java.util.\*;

import java.util.zip.ZipEntry;

import java.util.zip.ZipOutputStream;

@Controller

public class TasksController {

@Autowired

private TasksDocument document = new TasksDocument();

private MatrixProblem currentProblem = new MatrixAddSubMul((int) System.currentTimeMillis());

String currentType = "addsubmult";

@GetMapping("/")

public String greeting(

Map<String, Object> model

){

String problemContent = currentProblem.getProblemContent();

String answerContent = currentProblem.getAnswerContent();

String problemText = currentProblem.getProblemText();

model.put("problem", problemContent);

model.put("answer", answerContent);

model.put("problemtext", problemText);

model.put("items", document.getSize());

model.put("size",document.getSize());

return "redirect:/problems?addsubmult";

}

@GetMapping("/problems")

public String problem(

Map<String,Object> model, @RequestParam(name="type", defaultValue = "none") String type

)throws Exception{

int seed = new Random((int) System.currentTimeMillis()).nextInt();

MatrixProblem problem;

if (!type.equals("none")){

currentType = type;

}

if (type.equals("none")){

problem = MatrixProblemFactory.typeof(currentType,seed);

}

else

problem = MatrixProblemFactory.typeof(type,seed);

//TODO:track amount of roblems, empty the problems after download

String problemContent = problem.getProblemContent();

String answerContent = problem.getAnswerContent();

String problemText = problem.getProblemText();

model.put("problem", problemContent);

model.put("answer", answerContent);

model.put("problemtext", problemText);

model.put("items", document.getSize());

model.put("size",document.getSize());

currentProblem = problem;

return "/problems";

}

@PostMapping("/problems/add")

public String add()

{

document.addTask(currentProblem);

return "redirect:/problems";

}

@GetMapping(value = "/download")

public @ResponseBody

ResponseEntity<byte[]> download()

throws IOException, InterruptedException {

Random rand = new Random(System.currentTimeMillis());

String tname = getRandomName(rand);

String aname = getRandomName(rand);

String tasksName = "tasks" + tname + ".tex";

String answersName = "answers" + aname + ".tex";

String pathTasks = "src/main/resources/files/"+tasksName;

String pathAnswers = "src/main/resources/files/"+ answersName;

document.createTasksTex(pathTasks,pathAnswers);

ProcessBuilder pb = new ProcessBuilder(

"/usr/bin/pdflatex",

"-output-directory=src/main/resources/files",

"-interaction=nonstopmode",

pathTasks

);

ProcessBuilder pb2 = new ProcessBuilder(

"/usr/bin/pdflatex",

"-output-directory=src/main/resources/files",

"-interaction=nonstopmode",

pathAnswers

);

pb.redirectErrorStream(true);

Process p = pb.start();

InputStream is = p.getInputStream();

int in=-1;

while((in=is.read())!=-1){

System.out.print((char)in);

}

int exitWith = p.waitFor();

System.out.println("\nExited with " + exitWith);

Process p2 = pb2.start();

p2.waitFor();

String zipName = getRandomName(rand) + ".zip";

String zipPath = "src/main/resources/files/"+zipName;

createZip(

zipPath,

new ArrayList<>(List.of(

pathTasks,

pathAnswers,

"src/main/resources/files/tasks" + tname + ".pdf",

"src/main/resources/files/answers" + aname + ".pdf")),

new ArrayList<>(

List.of(

"/tasks.tex",

"/answers.tex",

"/tasks.pdf",

"/answers.pdf"

)

)

);

var content = Files.readAllBytes(Path.of(zipPath));

HttpHeaders httpHeaders = new HttpHeaders();

httpHeaders.set(HttpHeaders.CONTENT\_TYPE, MediaType.APPLICATION\_OCTET\_STREAM\_VALUE); // (3) Content-Type: application/octet-stream

httpHeaders.set(HttpHeaders.CONTENT\_DISPOSITION, ContentDisposition.attachment().filename("algebrator.zip").build().toString());

File dir = new File("src/main/resources/files");

for(File file: Objects.requireNonNull(dir.listFiles()))

if (!file.isDirectory())

file.delete();

this.document = new TasksDocument();

return ResponseEntity.ok().headers(httpHeaders).body(content);

}

private void createZip(String zipPath, ArrayList<String> contentPaths, ArrayList<String> zipPaths) throws IOException {

FileOutputStream fout = new FileOutputStream(zipPath);

ZipOutputStream zout = new ZipOutputStream(fout);

for(int i=0;i<contentPaths.size();i++)

{

ZipEntry ze = new ZipEntry(zipPaths.get(i));

zout.putNextEntry(ze);

var content = Files.readAllBytes(Path.of(contentPaths.get(i)));

zout.write(content);

zout.closeEntry();

}

zout.close();

}

private String getRandomName(Random rand){

StringBuilder sb = new StringBuilder();

for (int i = 0; i < 16; i++) {

if (rand.nextBoolean())

sb.append('A' + rand.nextInt(26));

else

sb.append(rand.nextInt(10));

}

return sb.toString();

}

@PostMapping (value = "/reset")

public String endSession(){

this.document = new TasksDocument();

return "redirect:/";

}

@GetMapping(value = "/reset")

public String reset(){

this.document = new TasksDocument();

return "redirect:/";

}

}

## **Column.java**

package com.schuyweiz.algebragenerator.matrix;

import java.io.StringWriter;

import java.util.ArrayList;

import org.matheclipse.core.eval.EvalEngine;

import org.matheclipse.core.eval.TeXUtilities;

import org.matheclipse.core.expression.IntegerSym;

import org.matheclipse.core.interfaces.IExpr;

public class Column {

private ArrayList<IExpr> content;

private int size;

public boolean isWeak(){

int zeroes = 0;

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

if (content.get(i).equals(IntegerSym.valueOf(0)))

zeroes++;

if (zeroes==size-1)

return true;

}

return false;

}

public Column(ArrayList<IExpr> content){

this.content = content;

this.size = content.size();

}

public Column mult(IExpr times){

ArrayList<IExpr> col = new ArrayList<>();

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

//this.content.set(i,this.content.get(i)\*times);

var newNum = this.content.get(i).multiply(times);

col.add(newNum);

}

return new Column(col);

}

public Column add(Column anotherCol, IExpr coef) throws Exception {

ArrayList<IExpr> col = new ArrayList<>();

var tempRow = anotherCol.mult(coef);

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

//this.content.set(i,this.content.get(i)\*times);

var newNum = this.content.get(i).add(tempRow.get(i));

col.add(newNum);

}

return new Column(col);

}

public int getSize() {

return size;

}

public ArrayList<IExpr> getContent() {

return content;

}

public void set(int i, IExpr newValue){

this.content.set(i, newValue);

}

public IExpr get(int at) {

return content.get(at);

}

@Override

public String toString(){

StringWriter wr = new StringWriter();

TeXUtilities tu = new TeXUtilities(new EvalEngine(),false);

ArrayList<String> arr = new ArrayList<>();

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

//arr.add(ExprUtils.getExpression(util.eval(content.get(i))));

tu.toTeX(content.get(i),wr);

arr.add(wr.toString());

wr.getBuffer().setLength(0);

wr.getBuffer().trimToSize();

}

return String.join("&", arr);

}

public Row toRow(){

return new Row(this.content);

}

}

## Matrix.java

package com.schuyweiz.algebragenerator.matrix;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Random;

import com.schuyweiz.algebragenerator.utility.ExprUtils;

import org.matheclipse.core.expression.F;

import org.matheclipse.core.expression.FractionSym;

import org.matheclipse.core.expression.IntegerSym;

import org.matheclipse.core.interfaces.IExpr;

public class Matrix implements Cloneable{

//region elementary operations on rows

public void multRow(int rowId, IExpr coef){

var temp = this.rows.get(rowId).mult(coef);

rows.set(rowId,temp);

this.updateCols(rowId);

}

public void addRow(int fromId, int toId, IExpr coef) {

var temp = this.rows.get(toId).add(this.rows.get(fromId), coef);

rows.set(toId,temp);

this.updateCols(toId);

}

public void swapRow(int fromId, int toId){

Row temp = this.rows.get(fromId);

this.rows.set(fromId,this.rows.get(toId));

this.rows.set(toId,temp);

updateCols(fromId);

updateCols(toId);

}

//endregion

//endregion

//region matrix basic operations

public Matrix sub(Matrix another) {

ArrayList<Row> newRows = new ArrayList<>();

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

newRows.add(getRows().get(i).sub(another.rows.get(i)));

}

return new Matrix(newRows);

}

public Matrix add(Matrix another) {

ArrayList<Row> newRows = new ArrayList<>();

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

newRows.add(getRows().get(i).add(another.rows.get(i)));

}

return new Matrix(newRows);

}

public Matrix mult(Matrix matrix) {

matrix.convertRowsToCols();

ArrayList<Row> newRows = new ArrayList<>();

for (Row row:rows){

ArrayList<IExpr> newRow = new ArrayList<>();

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

newRow.add(row.mult(matrix.cols.get(i)));

}

newRows.add(new Row(newRow));

}

return new Matrix(newRows);

}

//endregion

//region static matrix generators

public static Matrix randomMatrix(Random rand, int boundL, int boundR, int height, int width){

ArrayList<Row> rows = new ArrayList<>();

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

ArrayList<IExpr> row = new ArrayList<>();

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

row.add(ExprUtils.getPositiveRandom(rand,boundL,boundR));

}

rows.add(new Row(row));

}

return new Matrix(rows);

}

public static Matrix identity(int width){

ArrayList<Row> newMatrix = new ArrayList<>();

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

ArrayList<IExpr> newRow = new ArrayList<>();

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

newRow.add(i==j? IntegerSym.valueOf(1):IntegerSym.valueOf(0));

}

newMatrix.add(new Row(newRow));

}

return new Matrix(newMatrix);

}

public static Matrix diag(int width, ArrayList<IExpr> diagonal){

Matrix matrix = identity(width);

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

var temp = matrix.rows.get(i).mult(diagonal.get(i));

matrix.rows.set(i,temp);

}

return matrix;

}

public static Matrix randDiag(int width, Random rand){

ArrayList<IExpr> diagonal = new ArrayList<>();

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

diagonal.add(ExprUtils.getPositiveRandom(rand,-3,3));

}

return Matrix.diag(width,diagonal);

}

public static Matrix orthogonal(Random rand, ArrayList<Integer> order, int n){

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

Collections.swap(order, rand.nextInt(order.size()), rand.nextInt(order.size()));

}

int a = order.get(0);

int b = order.get(1);

int c = order.get(2);

int d = order.get(3);

ArrayList<Row> rows = new ArrayList<Row>();

ArrayList<IExpr> row1 = new ArrayList<>();

row1.add(FractionSym.valueOf(a\*a+b\*b-c\*c-d\*d,n));

row1.add(FractionSym.valueOf(2\*(-a\*d+b\*c),n));

row1.add(FractionSym.valueOf(2\*(a\*c+b\*d),n));

ArrayList<IExpr> row2 = new ArrayList<>();

row2.add(FractionSym.valueOf(2\*(a\*d+b\*c),n));

row2.add(FractionSym.valueOf(a\*a-b\*b+c\*c-d\*d,n));

row2.add(FractionSym.valueOf(2\*(-a\*b + c\*d),n));

ArrayList<IExpr> row3 = new ArrayList<>();

row3.add(FractionSym.valueOf(2\*(-a\*c+b\*d),n));

row3.add(FractionSym.valueOf(2\*(a\*b+c\*d),n));

row3.add(FractionSym.valueOf(a\*a-b\*b-c\*c+d\*d,n));

rows.add(new Row(row1));

rows.add(new Row(row2));

rows.add(new Row(row3));

return new Matrix(rows);

}

public static Matrix ofRank(int width,int height, int rank, Random rand){

ArrayList<Row> rows = new ArrayList<>();

int indent = 0;

int remains = rank;

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

ArrayList<IExpr> exprs = new ArrayList<>();

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

if (j>=indent && i<rank)

exprs.add(ExprUtils.getRandomNonNull(rand,1,5));

else

exprs.add(IntegerSym.valueOf(0));

}

remains--;

rows.add(new Row(exprs));

indent += (rand.nextInt(height-indent-remains)+1);

}

return new Matrix(rows);

}

//endregion

public Matrix transpose(){

ArrayList<Row> rows = new ArrayList<>();

for (Column col: cols){

rows.add(col.toRow());

}

return new Matrix(rows);

}

public int getHeight() {

return height;

}

public ArrayList<Row> getRows() {

return rows;

}

public IExpr get(int row, int col){

return this.rows.get(row).get(col);

}

public Matrix(ArrayList<Row> rows) {

this.height = rows.size();

this.width = rows.get(0).getSize();

this.rows = new ArrayList<>();

for (Row row:rows){

try {

this.rows.add((Row) row.clone());

} catch (CloneNotSupportedException e) {

e.printStackTrace();

}

}

convertRowsToCols();

}

private void convertRowsToCols(){

ArrayList<Column> cols = new ArrayList<>();

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

ArrayList<IExpr> newCol = new ArrayList<>();

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

newCol.add(rows.get(j).get(i));

}

cols.add(new Column(newCol));

}

this.cols = cols;

}

private void setCol(int i, int j, IExpr newValue){

this.cols.get(j).set(i,newValue);

}

private void updateCols(int at){

for (int colId = 0; colId < cols.size(); colId++) {

setCol(at,colId,

rows.get(at).get(colId));

}

}

@Override

public String toString(){

StringBuilder s= new StringBuilder();

for (Row row:rows){

s.append(row.toString()).append("\n");

}

return s.toString();

}

public Matrix strongShuffle(Random rand, int left, int right, int cycles){

var inverseMatrix = new Matrix(this.rows);

ArrayList<Integer> order = new ArrayList<>();

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

order.add(i);

}

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

Collections.swap(order, rand.nextInt(height), rand.nextInt(height));

}

int first = order.get(0);

for (int i = 0; i < height - 2; i++) {

IExpr coef = ExprUtils.getRandomNonNull(rand,left, right);

this.multCurrent(

elementaryOpAdd(first, order.get(i+1),coef,width)

);

inverseMatrix = elementaryOpAdd(first, order.get(i+1),coef.negative(),width).mult(inverseMatrix);

}

int second = order.get(height-1);

for (int i = 0; i < height - 2; i++) {

IExpr coef = ExprUtils.getRandomNonNull(rand,left, right);

this.multCurrent(

elementaryOpAdd(second, order.get(i),coef,width)

);

inverseMatrix = elementaryOpAdd(second, order.get(i),coef.negative(),width).mult(inverseMatrix);

}

int nonZero = 0;

for (Column col : cols) {

if (col.isWeak()) {

for (int j = 0; j < col.getSize(); j++) {

if (!col.get(j).equals(IntegerSym.valueOf(0))) {

nonZero = j;

break;

}

}

}

}

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

if (i!=nonZero){

IExpr tempCoef = IntegerSym.valueOf(rand.nextBoolean()?1:-1);

this.multCurrent(

elementaryOpAdd(i,nonZero,tempCoef,width)

);

inverseMatrix = elementaryOpAdd(i, nonZero, tempCoef.negative(),width).mult(inverseMatrix);

}

}

for (int i = 0; i < height\*cycles; i++) {

int id = getRandomId(height, i,rand);

IExpr coef = ExprUtils.getRandomNonNull(rand,left, right);

this.multCurrent(

elementaryOpAdd(i%height,id, coef,width)

);

inverseMatrix = elementaryOpAdd(i%height, id, coef.negative(),width).mult(inverseMatrix);

}

return inverseMatrix;

}

public Matrix weakShuffle(Random rand, int left, int right){

var inverseMatrix = new Matrix(this.rows);

ArrayList<Integer> order = new ArrayList<>();

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

order.add(i);

}

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

Collections.swap(order, rand.nextInt(height), rand.nextInt(height));

}

IExpr coef = ExprUtils.getRandomNonNull(rand,left, right);

this.multCurrent(

elementaryOpAdd(order.get(0), order.get(1),coef,width)

);

inverseMatrix = elementaryOpAdd(order.get(0), order.get(1),coef.negative(),width).mult(inverseMatrix);

coef = ExprUtils.getRandomNonNull(rand,left, right);

this.multCurrent(

elementaryOpAdd(order.get(order.size()-1), order.get(order.size()-2),coef,width)

);

inverseMatrix = elementaryOpAdd(order.size()-1, order.size()-2,coef.negative(),width).mult(inverseMatrix);

return inverseMatrix;

}

public void simpleShuffle(Random rand, int left, int right){

ArrayList<Integer> order = new ArrayList<>();

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

order.add(i);

}

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

Collections.swap(order, rand.nextInt(height), rand.nextInt(height));

}

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

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

if (i!=j){

IExpr coef = ExprUtils.getRandomNonNull(rand,left, right);

addRow(order.get(i), order.get(j), coef);

}

}

}

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

int id = getRandomId(height, i,rand);

swapRow(order.get(i), id);

}

}

private Matrix elementaryOpAdd(int from, int to, IExpr coef, int size){

var identity = Matrix.identity(size);

identity.addRow(from, to, coef);

return identity;

}

private int getRandomId(int size, int current, Random random){

return (current + random.nextInt(size-1)+1)%size;

}

private void multCurrent(Matrix matrix){

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

ArrayList<IExpr> row = new ArrayList<>();

for (Column col : matrix.cols) {

IExpr val = rows.get(i).mult(col);

row.add(val);

}

this.rows.set(i,new Row(row));

}

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

updateCols(i);

}

}

private final ArrayList<Row> rows;

private ArrayList<Column> cols;

private final int width;

private final int height;

}

## Row.java

package com.schuyweiz.algebragenerator.matrix;

import java.io.StringWriter;

import java.util.ArrayList;

import org.matheclipse.core.eval.EvalEngine;

import org.matheclipse.core.eval.TeXUtilities;

import org.matheclipse.core.expression.IntegerSym;

import org.matheclipse.core.interfaces.IExpr;

public class Row implements Cloneable {

@Override

protected Object clone() throws CloneNotSupportedException{

Row row = new Row();

row.content = new ArrayList<>();

for (IExpr num: this.content){

row.content.add(num);

}

row.size = this.size;

return row;

}

public Row(){}

public IExpr get(int id){

return content.get(id);

}

public void set(int id, IExpr newValue){

this.content.set(id, newValue);

}

public Row(ArrayList<IExpr> content){

this.content = content;

this.size = content.size();

}

public Row mult(IExpr times){

ArrayList<IExpr> row = new ArrayList<>();

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

//this.content.set(i,this.content.get(i)\*times);

var newNum = this.content.get(i).multiply(times);

row.add(newNum);

}

return new Row(row);

}

public Row add(Row anotherRow, IExpr coef){

ArrayList<IExpr> row = new ArrayList<>();

var tempRow = anotherRow.mult(coef);

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

//this.content.set(i,this.content.get(i)\*times);

var newNum = this.content.get(i).add(tempRow.get(i));

row.add(newNum);

}

return new Row(row);

}

public Row add(Row anotherRow){

return this.add(anotherRow,IntegerSym.valueOf(1));

}

public Row sub(Row anotherRow) {

ArrayList<IExpr> row = new ArrayList<>();

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

//this.content.set(i,this.content.get(i)\*times);

var newNum = this.content.get(i).subtract(anotherRow.get(i));

row.add(newNum);

}

return new Row(row);

}

public IExpr mult(Column column) {

IExpr result = IntegerSym.valueOf(0);

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

result = result.add(this.content.get(i).multiply(column.getContent().get(i)));

}

return result;

}

public int getSize() {

return size;

}

@Override

public String toString(){

StringWriter wr = new StringWriter();

TeXUtilities tu = new TeXUtilities(new EvalEngine(),false);

ArrayList<String> arr = new ArrayList<>();

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

//arr.add(ExprUtils.getExpression(util.eval(content.get(i))));

tu.toTeX(content.get(i),wr);

arr.add(wr.toString());

wr.getBuffer().setLength(0);

wr.getBuffer().trimToSize();

}

var temp = String.join("&", arr);

return temp+"\\\\";

}

private ArrayList<IExpr> content;

private int size;

}

## DimKer.java

package com.schuyweiz.algebragenerator.tasks;

import com.schuyweiz.algebragenerator.matrix.Matrix;

import java.util.Random;

public class DimKer extends MatrixProblem{

private Matrix A;

private Matrix B;

private Matrix X;

private int rank;

private String problemText = "Найти ранг ядра и отображения\n матрицы перехода из А в В";

public DimKer(int seed){

this.rand = new Random(seed);

this.rank = rand.nextInt(3)+1;

this.A = initA();

this.X = initX();

this.B = initB();

}

private Matrix initX(){

return Matrix.ofRank(4,4,rank,rand);

}

private Matrix initA(){

var matrix = Matrix.randDiag(4,rand);

return matrix.strongShuffle(rand,-1,2,1);

}

private Matrix initB(){

return A.mult(X);

}

@Override

public String getProblemText() {

return problemText;

}

@Override

public String getAnswerText() {

return null;

}

@Override

public String getProblemContent() {

return texExpression(

String.format(

"A = %s\n" +

"B = %s",

getMatrixValues(A),

getMatrixValues(B)

)

);

}

@Override

public String getAnswerContent() {

return String.format(

"%s\n" +

"rank = %s",

texExpression("X = " + getMatrixValues(X)),

rank

);

}

}

## FindEigenvalues.java

package com.schuyweiz.algebragenerator.tasks;

import com.schuyweiz.algebragenerator.matrix.Matrix;

import com.schuyweiz.algebragenerator.utility.ExprUtils;

import org.matheclipse.core.interfaces.IExpr;

import java.util.ArrayList;

import java.util.Random;

import java.util.stream.Collectors;

public class FindEigenvalues extends MatrixProblem{

ArrayList<IExpr> eigenvalues;

Matrix matrix;

Matrix A;

Matrix invA;

private final String problemText = "Найдите все собственные значения матрицы: ";

private final String answerText = "Собственные значения матрицы: ";

public FindEigenvalues(int seed) throws Exception {

this.rand = new Random(seed);

initEigenvalues();

this.matrix = Matrix.diag(3,eigenvalues);

for (int i = 0; i < 1; i++) {

A = Matrix.identity(3);

invA = A.strongShuffle(rand,-3,3,1);

matrix = invA.mult(matrix).mult(A);

}

}

private void initEigenvalues(){

ArrayList<IExpr> values = new ArrayList<>();

for (int i = 0; i < 3; i++) {

values.add(ExprUtils.getRandom(rand,-5,5));

}

eigenvalues = values;

}

@Override

public String getProblemText() {

return problemText;

}

@Override

public String getAnswerText() {

return answerText;

}

@Override

public String getProblemContent() {

return texExpression(

String.format("%s", getMatrixValues(matrix))

);

}

//TODO: modify answers format

@Override

public String getAnswerContent() {

return eigenvalues

.stream()

.map(val->val.toString())

.collect(Collectors.joining(" ,"));

}

}

## InverseMatrix.java

package com.schuyweiz.algebragenerator.tasks;

import com.schuyweiz.algebragenerator.matrix.Matrix;

import org.matheclipse.core.expression.IntegerSym;

import java.util.Random;

public class InverseMatrix extends MatrixProblem{

private final Matrix matrix;

private final Matrix inverse;

private boolean isInvertible =true;

public InverseMatrix(int seed) throws Exception {

this.rand = new Random(seed+1);

int width = 3;

if (rand.nextBoolean()){

int rank = rand.nextInt(width-1)+1;

matrix = Matrix.ofRank(width,width,rank,rand);

isInvertible = false;

}

else{

matrix = Matrix.identity(width);

}

inverse = matrix.strongShuffle(rand,-2,2,3);

}

@Override

public String getProblemText() {

return "Найти обратную матрицу";

}

@Override

public String getAnswerText() {

return null;

}

@Override

public String getProblemContent() {

String matrixString = getMatrixValues(matrix);

return texExpression(matrixString);

}

@Override

public String getAnswerContent() {

String reverseString = getMatrixValues(this.inverse);

if (!isInvertible)

return "Матрица не обратима";

return texExpression(reverseString);

}

}

## JordanCanonical.java

package com.schuyweiz.algebragenerator.tasks;

import com.schuyweiz.algebragenerator.matrix.Matrix;

import com.schuyweiz.algebragenerator.matrix.Row;

import com.schuyweiz.algebragenerator.utility.ExprUtils;

import org.matheclipse.core.expression.IntegerSym;

import org.matheclipse.core.interfaces.IExpr;

import java.util.ArrayList;

import java.util.Random;

public class JordanCanonical extends MatrixProblem {

private final int size;

private final Matrix jordan;

private final Matrix problemMatrix;

public JordanCanonical(int seed) {

this.rand = new Random(seed);

size = 4;

jordan = generateJordan();

Matrix A = Matrix.identity(size);

Matrix invA = A.strongShuffle(rand, -2, 2,1);

problemMatrix = A.mult(jordan).mult(invA);

}

private Matrix generateJordan() {

Matrix jordan;

ArrayList<IExpr> eigenvalues = new ArrayList<>();

IExpr currentVal = ExprUtils.getRandomNonNull(rand, -5, 5);

int i = 0;

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

if (rand.nextInt(3) == 2) {

break;

}

eigenvalues.add(currentVal);

}

currentVal = ExprUtils.getRandomNonNull(rand, -5, 5);

for (; i < size; i++) {

eigenvalues.add(currentVal);

}

ArrayList<Row> rows = new ArrayList<Row>();

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

ArrayList<IExpr> row = new ArrayList<>();

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

if (k == j)

row.add(eigenvalues.get(j));

else

row.add(IntegerSym.valueOf(0));

}

rows.add(new Row(row));

}

for (int j = 0; j < size - 1; j++) {

if (rows.get(j).get(j).equals(rows.get(j + 1).get(j + 1))) {

if (rand.nextBoolean()) {

rows.get(j).set(j+1, IntegerSym.valueOf(1));

}

}

}

jordan = new Matrix(rows);

return jordan;

}

@Override

public String getProblemText() {

return "Применить алгоритм нахождения жордановой нормальной матрицы: ";

}

@Override

public String getAnswerText() {

return null;

}

@Override

public String getProblemContent() {

return this.texExpression(

String.format("%s", getMatrixValues(problemMatrix))

);

}

@Override

public String getAnswerContent() {

return this.texExpression(

String.format("%s", getMatrixValues(jordan))

);

}

}

## MatrixAddSubMult.java

package com.schuyweiz.algebragenerator.tasks;

import com.schuyweiz.algebragenerator.matrix.Matrix;

import java.util.Random;

public class MatrixAddSubMul extends MatrixProblem{

private final String sign;

public MatrixAddSubMul(int randomSeed) {

this.rand = new Random(randomSeed);

String sign = generateSign();

int height = rand.nextInt(2)+3;

int width = rand.nextInt(2)+3;

this.firstTerm = Matrix.randomMatrix(this.rand, -5,5,height,width);

if (sign.equals("\\times")){

int secondWidth = rand.nextInt(2)+2;

secondTerm = Matrix.randomMatrix(this.rand, -5,5,width,secondWidth);

}

else

this.secondTerm = Matrix.randomMatrix(this.rand, -5,5,height,width);

if (sign.equals("+"))

this.answer = firstTerm.add(secondTerm);

if (sign.equals("-"))

this.answer = firstTerm.sub(secondTerm);

if (sign.equals("\\times"))

this.answer = firstTerm.mult(secondTerm);

this.sign = " " + sign + " ";

}

private String generateSign(){

int value = rand.nextInt(3);

return value==0? "+"

: value == 1? "-"

: "\\times";

}

@Override

public String getProblemText() {

return "Вычислить результат выражения: ";

}

@Override

public String getAnswerText() {

return null;

}

@Override

public String getProblemContent() {

String firstTermString = getMatrixValues(this.firstTerm);

String secondTermString = getMatrixValues(this.secondTerm);

return texExpression(String.format("%s",firstTermString + sign + secondTermString));

}

@Override

public String getAnswerContent() {

return texExpression(getMatrixValues(answer));

}

}

## MatrixPowerN.java

package com.schuyweiz.algebragenerator.tasks;

import com.schuyweiz.algebragenerator.matrix.Matrix;

import com.schuyweiz.algebragenerator.utility.ExprUtils;

import org.matheclipse.core.expression.F;

import org.matheclipse.core.interfaces.IExpr;

import java.util.\*;

public class MatrixPowerN extends MatrixProblem {

//для создания задач такого типа будем пользоваться формой A = P Q P^-1

private Matrix P;

private Matrix invP;

private Matrix Q;

private Matrix Qn;

private Matrix A;

private final int width;

public MatrixPowerN(int seed) throws Exception {

this.rand = new Random(seed);

this.width = rand.nextInt(2)+3;

this.P = Matrix.randDiag(width,rand);

this.invP = P.weakShuffle(rand,-2,2);

createMatrixQ();

createMatrixA();

createMatrixQn();

}

private void createMatrixQ(){

ArrayList<IExpr> diagonal = new ArrayList<>();

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

diagonal.add(ExprUtils.getPositiveRandom(rand,-3,3));

}

this.Q = Matrix.diag(width,diagonal);

}

private void createMatrixQn(){

ArrayList<IExpr> diagonal = new ArrayList<>();

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

diagonal.add(F.Power(Q.get(i,i),F.Dummy("n")));

}

this.Qn = Matrix.diag(width,diagonal);

}

private void createMatrixA() throws Exception {

this.A = this.P.mult(Q).mult(invP);

}

@Override

public String getProblemText() {

return "Найти n-ую степень матрциы";

}

@Override

public String getAnswerText() {

return null;

}

@Override

public String getProblemContent() {

String aString = getMatrixValues(this.A);

return String.format("\\(%s^{%s}\\)",aString,"n");

}

@Override

public String getAnswerContent() {

Matrix m = this.P.mult(this.Qn).mult(this.invP);

String pInvString = getMatrixValues(m);

return String.format("\\( %s%s%s \\\\= %s\\)",

getMatrixValues(P),getMatrixValues(Qn),getMatrixValues(invP),pInvString);

}

}

## MatrixProblem.java

package com.schuyweiz.algebragenerator.tasks;

import com.schuyweiz.algebragenerator.matrix.Matrix;

import org.springframework.stereotype.Component;

import java.util.Random;

@Component

public abstract class MatrixProblem implements ProblemInterface {

protected Matrix firstTerm;

protected Matrix secondTerm;

protected Matrix answer;

protected Random rand;

protected String getMatrixValues(Matrix matrix){

int rows = matrix.getHeight();

StringBuilder sb = new StringBuilder();

sb.append("\\begin{pmatrix}\n");

for (int i=0;i<rows-1;i++){

sb.append(matrix.getRows().get(i).toString())

.append("\\\\\n");

}

sb.append(matrix.getRows().get(rows-1))

.append("\n")

.append("\\end{pmatrix}");

return sb.toString();

}

protected String texExpression(String expr){

return String.format(

"$$ %s $$", expr

);

}

}

## MatrixProblemFactory.java

package com.schuyweiz.algebragenerator.tasks;

import com.schuyweiz.algebragenerator.matrix.Matrix;

import java.util.ArrayList;

import java.util.List;

import java.util.Random;

public class MatrixProblemFactory {

public static MatrixProblem typeof(String type, int seed) throws Exception {

switch (type){

case "qr":

return new QRdecomposition(seed);

case "orthdiag":

return new OrthgonalDiag(seed);

case "eigenvalues":

return new FindEigenvalues(seed);

case "inverse":

return new InverseMatrix(seed);

case "addsubmult":

return new MatrixAddSubMul(seed);

case "powern":

return new MatrixPowerN(seed);

case "rank":

return new MatrixRank(seed);

case "jordan":

return new JordanCanonical(seed);

case "svd":

return new SVDdecomposition(seed);

case "rot":

return new RotationAroundAxis(seed);

case "dimker":

return new DimKer(seed);

}

return null;

}

public static MatrixProblem getRandomProblem(Random rand, int seed) throws Exception {

ArrayList<String> vars = new ArrayList<>(

List.of(

"qr",

"orthdiag",

"eigenvalues",

"inverse",

"addsubmult",

"powern",

"rank",

"jordan",

"svd",

"rot",

"dimker"

)

);

return typeof(

vars.get(rand.nextInt(vars.size())),

seed

);

}

}

## MatrixRank.java

package com.schuyweiz.algebragenerator.tasks;

import com.schuyweiz.algebragenerator.matrix.Matrix;

import java.util.Random;

public class MatrixRank extends MatrixProblem {

public MatrixRank(int seed) {

this.rand = new Random(seed);

int width = rand.nextInt(4)+3;

int height = rand.nextInt(2)+3;

rank = rand.nextInt(height-1)+1;

this.matrix = Matrix.ofRank(width,height,rank,rand);

initialMatrix = getMatrixValues(matrix);

matrix.simpleShuffle(rand,-2,2);

}

@Override

public String getProblemText() {

return "Найти ранг матрицы ";

}

@Override

public String getAnswerText() {

return "Ранг матрицы равен ";

}

@Override

public String getProblemContent() {

String matrixString = getMatrixValues(matrix);

return texExpression(matrixString);

}

@Override

public String getAnswerContent() {

return

texExpression( String.format("%s rank = %s", initialMatrix, rank));

}

private Matrix matrix;

private final String initialMatrix;

private final int rank;

}

## OrthogonalDiag.java

package com.schuyweiz.algebragenerator.tasks;

import com.schuyweiz.algebragenerator.matrix.Matrix;

import com.schuyweiz.algebragenerator.utility.ExprUtils;

import org.matheclipse.core.expression.IntegerSym;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

import java.util.Random;

public class OrthgonalDiag extends MatrixProblem {

private String problemText = "Найти ортогональное преобразование для приведения \n" +

"к диагональному виду и вычислить диагональный вид:";

private String answerText = "";

private Matrix T;

private Matrix invT;

private Matrix D;

private Matrix A;

private int n;

public OrthgonalDiag(int seed){

this.rand = new Random(seed);

this.T = createT();

this.invT = T.transpose();

this.D = createD();

this.A = createA();

}

private Matrix createT(){

int a = rand.nextInt(2)\*2+1;

int b =rand.nextInt(2)\*2;

int c =rand.nextInt(2)\*2;

int d = rand.nextInt(2)\*2;

n = a\*a+ b\*b + c\*c + d\*d;

var order = new ArrayList<Integer>(

List.of(

a,b,c,d

));

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

Collections.swap(order, rand.nextInt(order.size()), rand.nextInt(order.size()));

}

return Matrix.orthogonal(rand, order,n);

}

private Matrix createD(){

var diag = new ArrayList<>(List.of(

ExprUtils.getRandom(rand, -5,5).times(IntegerSym.valueOf(n\*n)),

ExprUtils.getRandom(rand,-5,5).times(IntegerSym.valueOf(n\*n)),

ExprUtils.getRandom(rand,-5,5).times(IntegerSym.valueOf(n\*n))

));

return Matrix.diag(3,diag);

}

private Matrix createA(){

return this.T.mult(this.D).mult(this.invT);

}

@Override

public String getProblemText() {

return problemText;

}

@Override

public String getAnswerText() {

return answerText;

}

@Override

public String getProblemContent() {

return texExpression(

String.format(

"%s", getMatrixValues(A)

)

);

}

@Override

public String getAnswerContent() {

return texExpression(

String.format(

"%s \\times %s \\times %s = %s",

getMatrixValues(T),

getMatrixValues(D),

getMatrixValues(invT),

getMatrixValues(T.mult(D).mult(invT))

)

);

}

}

## ProblemInterface.java

package com.schuyweiz.algebragenerator.tasks;

public interface ProblemInterface {

String getProblemText();

String getAnswerText();

String getProblemContent();

String getAnswerContent();

}

## QRdeomposition.java

package com.schuyweiz.algebragenerator.tasks;

import com.schuyweiz.algebragenerator.matrix.Matrix;

import com.schuyweiz.algebragenerator.matrix.Row;

import com.schuyweiz.algebragenerator.utility.ExprUtils;

import org.matheclipse.core.expression.IntegerSym;

import org.matheclipse.core.interfaces.IExpr;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

import java.util.Random;

public class QRdecomposition extends MatrixProblem {

private Matrix Q;

private Matrix R;

private Matrix A;

private int n;

public QRdecomposition(int seed){

this.rand = new Random(seed);

Q = createQ();

R = createR();

A = Q.mult(R);

}

private Matrix createR(){

var rows= Q.getRows();

ArrayList<Row> rowsR = new ArrayList<>();

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

ArrayList<IExpr> row = new ArrayList<>();

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

row.add(IntegerSym.valueOf(0));

}

row.add(ExprUtils.getPositiveRandom(rand,1,4).times(IntegerSym.valueOf(n)));

for (int j = i+1; j < rows.get(i).getSize(); j++) {

row.add(ExprUtils.getRandomNonNull(rand,-3,3).times(IntegerSym.valueOf(n)));

}

rowsR.add(new Row(row));

}

return new Matrix(rowsR);

}

private Matrix createQ(){

int a = rand.nextInt(2)\*2+1;

int b =rand.nextInt(2)\*2;

int c =rand.nextInt(2)\*2;

int d = rand.nextInt(2)\*2;

n = a\*a+ b\*b + c\*c + d\*d;

var order = new ArrayList<Integer>(

List.of(

a,b,c,d

));

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

Collections.swap(order, rand.nextInt(order.size()), rand.nextInt(order.size()));

}

return Matrix.orthogonal(rand, order, n);

}

@Override

public String getProblemText() {

return "Найти QR разложение для матрицы";

}

@Override

public String getAnswerText() {

return null;

}

@Override

public String getProblemContent() {

return this.texExpression(

String.format("%s", getMatrixValues(A))

);

}

@Override

public String getAnswerContent() {

return this.texExpression(

String.format("Q = %s R = %s", getMatrixValues(Q), getMatrixValues(R))

);

}

}

## RotationAroundAxis.java

package com.schuyweiz.algebragenerator.tasks;

import com.schuyweiz.algebragenerator.matrix.Matrix;

import com.schuyweiz.algebragenerator.matrix.Row;

import com.schuyweiz.algebragenerator.utility.ExprUtils;

import org.matheclipse.core.eval.ExprEvaluator;

import org.matheclipse.core.expression.F;

import org.matheclipse.core.expression.FractionSym;

import org.matheclipse.core.expression.IntegerSym;

import org.matheclipse.core.interfaces.IExpr;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

import java.util.Random;

import java.util.stream.Collectors;

public class RotationAroundAxis extends MatrixProblem{

private String problemText = "Найти угол поворота матрицы оператора и ось поворота: ";

private final ArrayList<IExpr> possibleFractions = new ArrayList<>(

List.of(

FractionSym.valueOf(1,4),

FractionSym.valueOf(1,3),

FractionSym.valueOf(2,3),

FractionSym.valueOf(1,2),

FractionSym.valueOf(3,4),

FractionSym.valueOf(1,6),

FractionSym.valueOf(5,6),

FractionSym.ONE

)

);

private IExpr arg;

private Matrix A;

private FractionSym frac;

private ArrayList<IExpr> axis;

public RotationAroundAxis(int seed){

this.rand = new Random(seed);

this.arg = initArg();

this.axis = initAxis();

this.A = initA();

}

private ArrayList<IExpr> initAxis() {

ArrayList<IExpr> list = new ArrayList<>();

int zeroes = rand.nextInt(2)+1;

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

list.add(IntegerSym.valueOf(0));

}

int axisSize = 3;

IExpr expr = ExprUtils.getRandomNonNull(rand,-2,2);

for (int i = 0; i < axisSize-zeroes; i++) {

list.add(expr);

expr = ExprUtils.getRandomNonNull(rand,-2,2);

}

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

Collections.swap(list, rand.nextInt(list.size()), rand.nextInt(list.size()));

}

return list;

}

private IExpr initArg() {

ExprEvaluator util = new ExprEvaluator(false, (short) 100);

var choice = rand.nextInt(possibleFractions.size());

var expr = possibleFractions.get(choice).times(F.Pi);

frac = (FractionSym) possibleFractions.get(choice);

if (rand.nextBoolean()) {

return util.eval(expr);

}

else {

return util.eval(expr.negative());

}

}

private Matrix initA(){

var x = axis.get(0);

var y = axis.get(1);

var z = axis.get(2);

ExprEvaluator util = new ExprEvaluator(false, (short) 100);

var cos = util.eval(F.Cos(arg));

var sin = util.eval(F.Sin(arg));

ArrayList<IExpr> row1 = new ArrayList<>(

List.of(

cos.plus(IntegerSym.valueOf(1).minus(cos).times(x.times(x))),

x.times(y).times(IntegerSym.valueOf(1).minus(cos)).minus(sin.times(z)),

x.times(z).times(IntegerSym.valueOf(1).minus(cos)).plus(sin.times(y))

)

);

ArrayList<IExpr> row2 = new ArrayList<>(

List.of(

x.times(y).times(IntegerSym.valueOf(1).minus(cos)).plus(sin.times(z)),

cos.plus(IntegerSym.valueOf(1).minus(cos).times(y.times(y))),

y.times(z).times(IntegerSym.valueOf(1).minus(cos)).minus(sin.times(x))

)

);

ArrayList<IExpr> row3 = new ArrayList<>(

List.of(

x.times(z).times(IntegerSym.valueOf(1).minus(cos)).minus(sin.times(y)),

y.times(z).times(IntegerSym.valueOf(1).minus(cos)).plus(sin.times(x)),

cos.plus(IntegerSym.valueOf(1).minus(cos).times(z.times(z)))

)

);

return new Matrix(new ArrayList<Row>(

List.of(

new Row(row1),

new Row(row2),

new Row(row3)

)

));

}

@Override

public String getProblemText() {

return problemText;

}

@Override

public String getAnswerText() {

//TODO: write [roper answer text

return "Найти угол поворота и ось вращения: ";

}

@Override

public String getProblemContent() {

return texExpression(

String.format(

"%s", getMatrixValues(A)

)

);

}

@Override

public String getAnswerContent() {

return String.format(

"Угол поворота θ = %s\n" +

"Ось вращения v = %s",

texExpression(

String.format("%s\\times \\frac{%s}{%s}", "\\pi",frac.numerator(),frac.denominator())

),

texExpression(axis.stream()

.map(s->s.toString())

.collect(Collectors.joining(",")))

);

}

}

## SVDdecomposition.java

ackage com.schuyweiz.algebragenerator.tasks;

import com.schuyweiz.algebragenerator.matrix.Matrix;

import com.schuyweiz.algebragenerator.utility.ExprUtils;

import org.matheclipse.core.expression.IntegerSym;

import org.matheclipse.core.interfaces.IExpr;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

import java.util.Random;

public class SVDdecomposition extends MatrixProblem {

private Matrix U;

private Matrix A;

private Matrix V;

private Matrix D;

private int nU;

private int nV;

//U^T A V = D

private String problemText = "Найти сингулярное разложение следующей матрицы: ";

public SVDdecomposition(int seed){

this.rand = new Random(seed);

this.U = createU();

this.V = createV();

this.D = createD();

this.A = U.mult(D).mult(V.transpose());

}

private Matrix createD(){

var list = new ArrayList<IExpr>(

List.of(

ExprUtils.getPositiveRandom(rand,1,3),

ExprUtils.getPositiveRandom(rand,1,3),

ExprUtils.getPositiveRandom(rand,1,3)

)

);

return Matrix.diag(3,list).mult(Matrix.diag(3,

new ArrayList<>(

List.of(

IntegerSym.valueOf(nU\*nV),

IntegerSym.valueOf(nU\*nV),

IntegerSym.valueOf(nU\*nV)

)

)));

}

private Matrix createU(){

int a = rand.nextInt(2)\*2+1;

int b =rand.nextInt(2)\*2;

int c =rand.nextInt(2)\*2;

int d = rand.nextInt(2)\*2;

nU = a\*a+ b\*b + c\*c + d\*d;

var order = new ArrayList<Integer>(

List.of(

a,b,c,d

));

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

Collections.swap(order, rand.nextInt(order.size()), rand.nextInt(order.size()));

}

return Matrix.orthogonal(rand,order,nU);

}

private Matrix createV(){

int a = rand.nextInt(2)\*2+1;

int b =rand.nextInt(2)\*2;

int c =rand.nextInt(2)\*2;

int d = rand.nextInt(2)\*2;

nV = a\*a+ b\*b + c\*c + d\*d;

var order = new ArrayList<Integer>(

List.of(

a,b,c,d

));

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

Collections.swap(order, rand.nextInt(order.size()), rand.nextInt(order.size()));

}

return Matrix.orthogonal(rand,order,nV);

}

@Override

public String getProblemText() {

return problemText;

}

@Override

public String getAnswerText() {

return "";

}

@Override

public String getProblemContent() {

return texExpression(

String.format(

"%s",

getMatrixValues(A)

)

);

}

@Override

public String getAnswerContent() {

return "Один из возможных ответов: " + texExpression(

String.format(

"A = %s = U \\times \\Sigma \\times V^{-1} = %s \\times %s \\times %s",

getMatrixValues(A),

getMatrixValues(U),

getMatrixValues(D),

getMatrixValues(V.transpose())

)

);

}

}

## ExprUtils.java

package com.schuyweiz.algebragenerator.utility;

import org.matheclipse.core.expression.\*;

import org.matheclipse.core.interfaces.IExpr;

import java.util.Random;

public class ExprUtils {

public static IExpr getPositiveRandom(Random rand, int left, int right){

int num = left + rand.nextInt(right - left+1);

return IntegerSym.valueOf(num);

}

public static IExpr getRandom(Random rand, int left, int right){

if (rand.nextBoolean()){

return IntegerSym.valueOf(-rand.nextInt(-left));

}

return IntegerSym.valueOf(rand.nextInt(right));

}

public static IExpr getRandomNonNull(Random rand, int left, int right){

IExpr rhs;

if (left<0){

rhs = getPositiveRandom(rand, 1,right);

}

else if (left!=0){

return getPositiveRandom(rand, left, right);

}

else{

return getPositiveRandom(rand,1,right);

}

boolean isPositive = rand.nextBoolean();

if (isPositive){

return rhs;

}

else{

return rhs.times(IntegerSym.valueOf(-1));

}

}

}

## AlgebraGeneratorApplication.java

package com.schuyweiz.algebragenerator;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.context.properties.EnableConfigurationProperties;

@SpringBootApplication

public class AlgebraGeneratorApplication {

public static void main(String[] args) {

SpringApplication.run(AlgebraGeneratorApplication.class, args);

}

}

## TasksDocument.java

package com.schuyweiz.algebragenerator;

import com.schuyweiz.algebragenerator.tasks.MatrixProblem;

import org.springframework.beans.factory.config.ConfigurableBeanFactory;

import org.springframework.context.annotation.Scope;

import org.springframework.stereotype.Component;

import java.io.FileWriter;

import java.io.IOException;

import java.util.ArrayList;

@Component

@Scope(value = ConfigurableBeanFactory.SCOPE\_PROTOTYPE)

public class TasksDocument {

ArrayList<String> text = new ArrayList<>();

ArrayList<String> content = new ArrayList<>();

ArrayList<String> answers = new ArrayList<>();

public TasksDocument(){

}

public void addTask(MatrixProblem problem){

this.text.add(problem.getProblemText());

this.content.add(problem.getProblemContent());

this.answers.add(problem.getAnswerContent());

}

public String createTex(String content) throws IOException {

return String.format(

"\\documentclass{article}\n" +

"\\usepackage[utf8]{inputenc}\n" +

"\\usepackage{mathtools} \n" +

"\\usepackage[english,russian]{babel}\n" +

"\\begin{document}\n" +

"%s\n" +

"\\end{document}",

content

);

}

public void createTasksTex(String qpath, String apath) throws IOException {

String contentTasks = createTex(documentTasks());

FileWriter fw = new FileWriter( qpath);

fw.write(contentTasks);

fw.close();

String contentAnswer = createTex(documentSolutions());

FileWriter fw2 = new FileWriter( apath );

fw2.write(contentAnswer);

fw2.close();

}

public int getSize(){

return this.answers.size();

}

private String documentTasks(){

StringBuilder sb = new StringBuilder();

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

sb.append(String.format("\\section\*{Задача № %s}\n",i+1));

sb.append(text.get(i)).append("\n");

sb.append(content.get(i)).append("\n");

}

return sb.toString();

}

private String documentSolutions(){

StringBuilder sb = new StringBuilder();

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

sb.append(String.format("\\section\*{Задача № %s}\n",i+1));

sb.append(text.get(i)).append("\n");

sb.append(content.get(i)).append("\n");

sb.append(String.format("\\section\*{Ответ № %s}\n",i+1));

sb.append(answers.get(i)).append("\n");

}

return sb.toString();

}

## common.ftlh

<#macro page>

<!DOCTYPE html>

<head>

<title>Here is the title</title>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1">

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/bootstrap.min.css"

rel="stylesheet" integrity="sha384-eOJMYsd53ii+scO/bJGFsiCZc+5NDVN2yr8+0RDqr0Ql0h+rP48ckxlpbzKgwra6"

crossorigin="anonymous">

</head>

<body>

<div class = "container">

<#nested>

</div>

</body>

</html>

</#macro>

//TODO: add an argument of a page title

<#macro mjpage>

<!DOCTYPE html>

<html>

<head>

<title>Algebrator</title>

<script src="https://polyfill.io/v3/polyfill.min.js?features=es6"></script>

<script type="text/javascript" id="MathJax-script" async

src="https://cdn.jsdelivr.net/npm/mathjax@3/es5/tex-chtml.js">

</script>

<script type="text/javascript" src="https://yastatic.net/jquery/3.3.1/jquery.min.js"></script>

<meta name="viewport" content="width=device-width, initial-scale=1">

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/js/bootstrap.bundle.min.js"

integrity="sha384-JEW9xMcG8R+pH31jmWH6WWP0WintQrMb4s7ZOdauHnUtxwoG2vI5DkLtS3qm9Ekf"

crossorigin="anonymous"></script>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/bootstrap.min.css"

rel="stylesheet" integrity="sha384-eOJMYsd53ii+scO/bJGFsiCZc+5NDVN2yr8+0RDqr0Ql0h+rP48ckxlpbzKgwra6"

crossorigin="anonymous">

</head>

<body>

<#include "navbar.ftlh">

<div class="container-fluid mt-5">

<#nested>

</div>

<script>

window.onunload = function() {

console.log("Ya ttut");

$.ajax({

url: "/reset",

method: 'POST',

success: function() {

console.log("я получил и жив");

}

});

}

</script>

<script>

window.

</script>

</body>

</html>

</#macro>

<#macro ans answer>

<p>

<button class="btn btn-primary btn-dark" type="button" data-bs-toggle="collapse" data-bs-target="#collapseExample" aria-expanded="false" aria-controls="collapseExample">

Показать ответ

</button>

</p>

<div class="collapse" id="collapseExample">

<div class="card card-body">

${answer}

</div>

</div>

</#macro>

## navbar.ftlh

<nav class="navbar navbar-dark bg-dark navbar-expand-lg bg-light">

<div class="container-fluid">

<a class="navbar-brand me-auto mb-3 mb-lg-2" href="#">Алгебратор</a>

<div class="dropdown" id="dropdownMenu">

<ul class="navbar-nav me-auto mb-3 mb-lg-2">

<#if size?? && size gt 0>

<li class="nav-item">

<a class="nav-link active nav-link-dark" aria-current="page" href="/reset">

Сбросить текущий выбор

</a>

</li>

</#if>

<li class="nav-item dropdown"

aria-current="page">

<a class="nav-link dropdown-toggle active"

href="#"

id="navbarDarkDropdownMenuLink"

role="button"

data-bs-toggle="dropdown"

aria-expanded="true">

Сгенерировать задачу

</a>

<ul class="dropdown-menu dropdown-menu-dark" aria-labelledby="navbarDropdown">

<li><a class="dropdown-item" href="/problems?type=addsubmult">Сложение/Вычитание/Умножение матриц</a></li>

<li><a class="dropdown-item" href="/problems?type=rank">Поиск ранга матрицы</a></li>

<li><a class="dropdown-item" href="/problems?type=powern">Поиск n-ой степени матрицы</a></li>

<li><a class="dropdown-item" href="/problems?type=inverse">Определить обратимость матрицы</a></li>

<li><a class="dropdown-item" href="/problems?type=eigenvalues">Найти собственные значения</a></li>

<li><a class="dropdown-item" href="/problems?type=jordan">Привести к форме Жордана</a></li>

<li><a class="dropdown-item" href="/problems?type=qr">Найти QR разложение матрицы</a></li>

<li><a class="dropdown-item" href="/problems?type=orthdiag">Найти даигональную матрицу и ортогональное преобразование</a></li>

<li><a class="dropdown-item" href="/problems?type=svd">Провести сингулярное разложение над матрицей</a></li>

<li><a class="dropdown-item" href="/problems?type=rot">Найти угол и ось вращения ортогонального оператора</a></li>

<li><a class="dropdown-item" href="/problems?type=dimker">Найти размерность ядра и отображения матрицы перехода</a></li>

</ul>

</li>

<li class="nav-item">

<#if size??>

<#if size gt 0>

<a class="nav-link active nav-link-dark " aria-current="page" href="/download" onclick="reloadPage()">

Импортировать задачи и ответы

<#if items??>

<span class="badge badge-light">${items}</span>

<#else>

</#if>

</a>

<#else>

<a class="nav-link disabled nav-link-dark" aria-current="page" >

Импортировать задачи и ответы

<#if items??>

<span class="badge badge-light">${items}</span>

<#else>

</#if>

</a>

</#if>

<#else >

<a class="nav-link disabled nav-link-dark" aria-current="page" >

Импортировать задачи и ответы

<#if items??>

<span class="badge badge-light">${items}</span>

<#else>

<span class="badge badge-light">0</span>

</#if>

</a>

</#if>

</li>

</ul>

</div>

</div>

</nav>

<script>

function func() {

console.log("Ya ttut");

$.ajax({

url: "/reset",

method: 'POST',

success: function() {

console.warn("warning1");

console.debug("im here");

}

});

}

</script>

<script>

async function reloadPage(){

await new Promise(r => setTimeout(r, 1000));

window.location.reload();

}

</script>

## greeting.ftlh

<!DOCTYPE html>

<html>

<head>

<title>Algebrator</title>

<meta name="viewport" content="width=device-width, initial-scale=1">

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/js/bootstrap.bundle.min.js"

integrity="sha384-JEW9xMcG8R+pH31jmWH6WWP0WintQrMb4s7ZOdauHnUtxwoG2vI5DkLtS3qm9Ekf"

crossorigin="anonymous"></script>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/bootstrap.min.css"

rel="stylesheet" integrity="sha384-eOJMYsd53ii+scO/bJGFsiCZc+5NDVN2yr8+0RDqr0Ql0h+rP48ckxlpbzKgwra6"

crossorigin="anonymous">

</head>

<body>

<#include "parts/navbar.ftlh">

</body>

</html>

## problems.ftlh

<#import "parts/common.ftlh" as c>

<@c.mjpage>

<button type="button" class="btn btn-dark btn-lg btn-block btn-primary" onclick="refreshPage()">

Другая задача

</button>

<form action="/problems/add" method="post">

<button type="submit" class="btn btn-dark btn-lg btn-block" >

Добавить задачу

</button>

</form>

<#if problemtext??>

<div class="container">

<div class="row" align="center">

<p>${problemtext}</p>

<p>${problem}</p>

</div>

<div class="row" align="center">

<@c.ans "${answer}"/>

</div>

</div>

</#if>

<script>

function refreshPage(){

window.location.reload();

}

</script>

</@c.mjpage>

## application-dev.properties

spring.freemarker.expose-request-attributes=true

## MULTIPART (MultipartProperties)

# Enable multipart uploads

spring.servlet.multipart.enabled=true

# Threshold after which files are written to disk.

spring.servlet.multipart.file-size-threshold=2KB

# Max file size.

spring.servlet.multipart.max-file-size=200MB

# Max Request Size

spring.servlet.multipart.max-request-size=215MB

logging.level.org.springframework.web= DEBUG

logging.level.web= DEBUG

logging.level.com.schuyweiz.algebragenerator=DEBUG

## File Storage Properties

# All files uploaded through the REST API will be stored in this directory

file.upload-dir=/resources/files/

server.servlet.session.persistent=false

server.tomcat.connection-timeout=60s

server.servlet.session.cookie.max-age=-1

server.port=8080

## application.properties

spring.freemarker.expose-request-attributes=true

## MULTIPART (MultipartProperties)

# Enable multipart uploads

spring.servlet.multipart.enabled=true

# Threshold after which files are written to disk.

spring.servlet.multipart.file-size-threshold=2KB

# Max file size.

spring.servlet.multipart.max-file-size=200MB

# Max Request Size

spring.servlet.multipart.max-request-size=215MB

logging.level.org.springframework.web= DEBUG

logging.level.web= DEBUG

## File Storage Properties

# All files uploaded through the REST API will be stored in this directory

file.upload-dir=/resources/

server.servlet.session.persistent=false

server.tomcat.connection-timeout=60s

server.servlet.session.cookie.max-age=-1

hostname=37.228.117.48

server.port=80

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Изм. | Номера листов (страниц) | | | | Всего листов (страниц) в документе | № документа | Входящий № сопроводительного документа и дата | Подпись | Дата |
| изменённых | заменённых | новых | аннулированных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |