《Java》课程实验报告

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 专业名称 | 计算机科学与技术 | 年级 | 2023 | 班级 | 01 |
| 学生姓名 | 颜丙超 | 学号 | 202311000415 | 指导教师 | 梁成 |
| 实验题目 | 三维张量相乘 消除数组中重复出现的值 | | | 提交时间 | 2024.10.28 |

1. 实验目的和要求

**实验目的：**1.能够熟练理解多维数组的特性，并通过Java编程实践，掌握三维张量与三维数组结构之间的关联关系。进一步地，要求能够实现三维张量的相乘操作，该操作基于三维数组的相加及相乘等基本操作。

1. 掌握数组的基本操作，包括数组的遍历、搜索、删除等。同时，通过编写一个具体的程序——消除数组中重复出现的值，来深入理解Java中的方法定义、参数传递、递归调用等核心概念。此外，本实验还旨在培养学生的编程思维，提高解决实际问题的能力。

**实验要求：**1.深入理解三维张量的概念及其与三维数组结构之间的关联关系。能够根据三维张量的定义，使用Java编写函数实现两个三维张量的相乘操作。这里的相乘操作是基于特定的张量乘积（T-product）定义，涉及到unfold(), fold()以及bcirc()等运算。编写的程序应能够接收用户输入的两个三维张量的维度和元素数据，然后计算并输出它们的乘积张量。最后，需要根据提供的Matlab代码（tprod.m）对所编写的Java程序结果进行验证，确保程序的正确性和可靠性。

2.理解数组的基本操作，掌握方法定义和参数传递，实现重复消元的功能，编写测试程序，注意代码规范。

1. 实验环境

1.电脑一台，Windows11系统2.eclpise软件运行

三、实验内容及实施

**实验1**

在机器学习和多维数据分析中，张量（Tensor）是一个非常重要的概念，它可以被理解为高维数组。本次实验主要关注三维张量的操作，特别是三维张量的相乘操作。

三维张量可以看作是由多个二维数组（或称为矩阵）堆叠而成的结构，每个二维数组都是张量的一个“切片”（slice）。当固定张量的某一个维度时，可以得到该张量在该维度上的不同切片。

实验要求实现的三维张量相乘操作是基于特定的张量乘积（T-product）定义。这种乘积涉及到对张量的展开（unfold）、块循环矩阵（bcirc）的构建以及折叠（fold）回原张量形状的过程。

实施步骤：理解张量和三维数组：

深入理解张量的概念，特别是三维张量的结构。

掌握三维数组在Java中的表示方法，即使用三维数组来存储三维张量的数据。

实现unfold()函数：

编写一个Java函数，用于将三维张量沿指定维度展开成二维矩阵。这是实现T-product的关键步骤之一。

实现bcirc()函数：

编写一个Java函数，用于构建块循环矩阵。块循环矩阵是将张量的某个切片作为块，然后按循环方式排列形成的矩阵。

实现fold()函数：

编写一个Java函数，用于将二维矩阵折叠回三维张量的形状。这是展开操作的逆过程。

实现T-product函数：

利用上述三个函数，编写一个Java函数来实现两个三维张量的T-product。该函数应首先对两个张量进行适当的展开和块循环矩阵构建，然后进行矩阵相乘，最后再将结果折叠回三维张量的形状。

编写主程序：

编写一个Java主程序，用于接收用户输入的两个三维张量的维度和元素数据。

调用T-product函数计算两个张量的乘积，并输出结果张量的维度和元素数据。

验证结果：

使用Matlab中的tprod.m函数对Java程序的结果进行验证。通过比较Java程序和Matlab程序输出的结果，确保Java程序的正确性和可靠性。

编写验证报告：

根据实验结果，编写一份详细的验证报告，包括实验目的、实验步骤、实验结果和结论等部分。报告中应包含对Java程序和Matlab程序输出结果的比较和分析。

**实验2**

本实验旨在通过Java编程实践，掌握数组的基本操作，特别是如何遍历数组、检查元素是否存在以及构建新的数组来存储不重复的元素。实验的核心是实现一个名为eliminateDuplicates的方法，该方法能够接收一个整数数组作为输入，并返回一个新的数组，其中不包含任何重复的元素。

实施步骤：理解问题需求：

仔细阅读实验要求，明确eliminateDuplicates方法的输入和输出。

理解什么是数组中的重复元素，并思考如何有效地检测它们。

设计算法：

思考如何实现一个算法来遍历输入数组。

设计一个策略来检查每个元素是否已经在结果数组中，并据此决定是否将其添加到结果数组中。

考虑使用嵌套循环或额外的数据结构（如HashSet）来优化性能。

编写代码：

在Java开发环境中创建一个新的Java类。

在类中定义eliminateDuplicates方法，并为其添加必要的注释。

实现算法，包括遍历输入数组、检查重复元素和构建结果数组的逻辑。

测试代码：

编写测试代码来验证eliminateDuplicates方法的正确性。

创建包含不同整数数组的测试用例，包括空数组、所有元素都相同的数组和包含多个不同元素的数组。

运行测试代码，并检查结果是否符合预期。

优化代码：

分析代码的性能，并考虑是否有改进的空间。

如果使用了嵌套循环，可以尝试使用HashSet等数据结构来减少时间复杂度。

重新运行测试代码，以确保优化后的代码仍然正确。

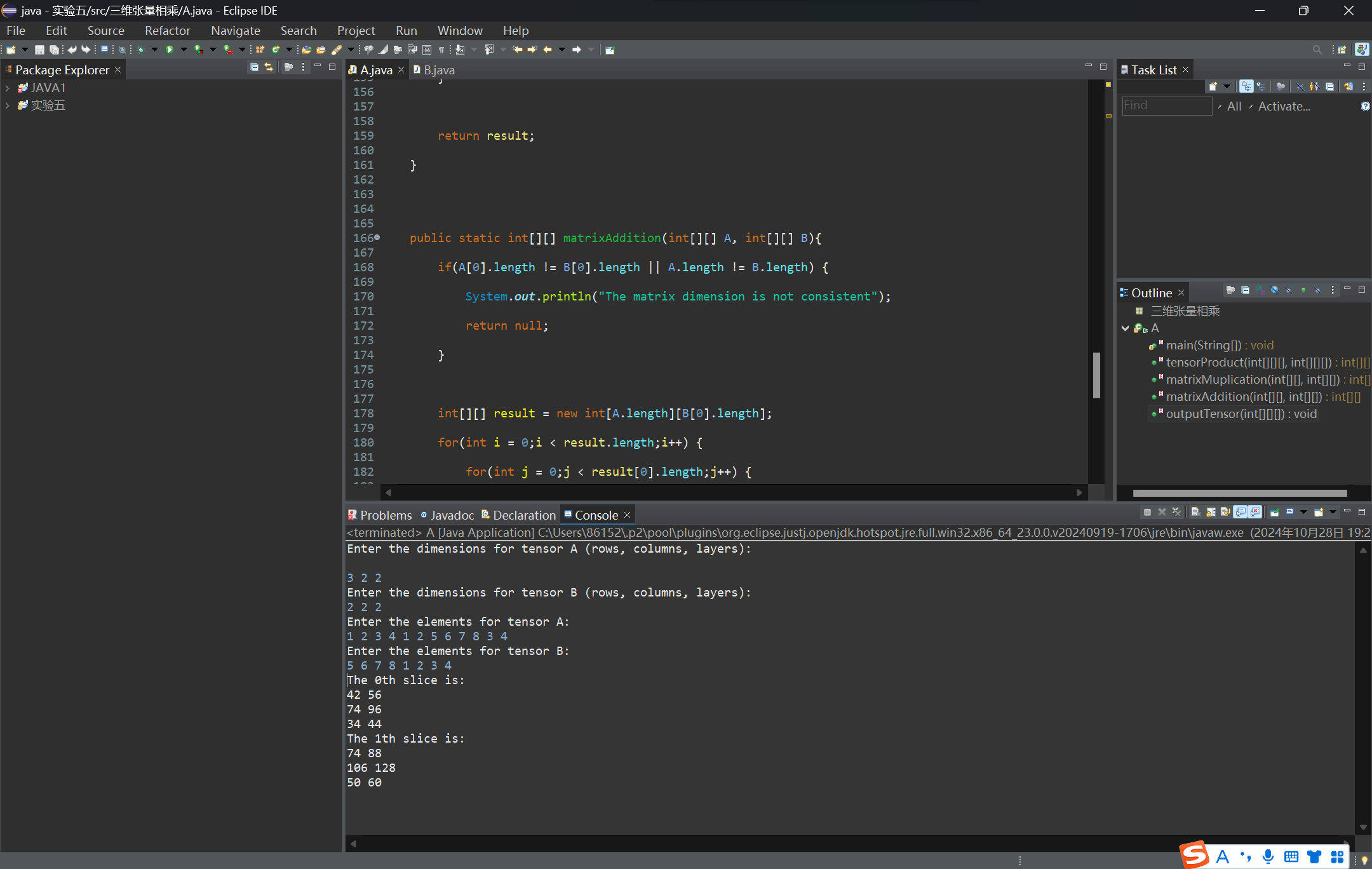
撰写实验报告：

详细描述实验的目的、要求、实施步骤和结果。

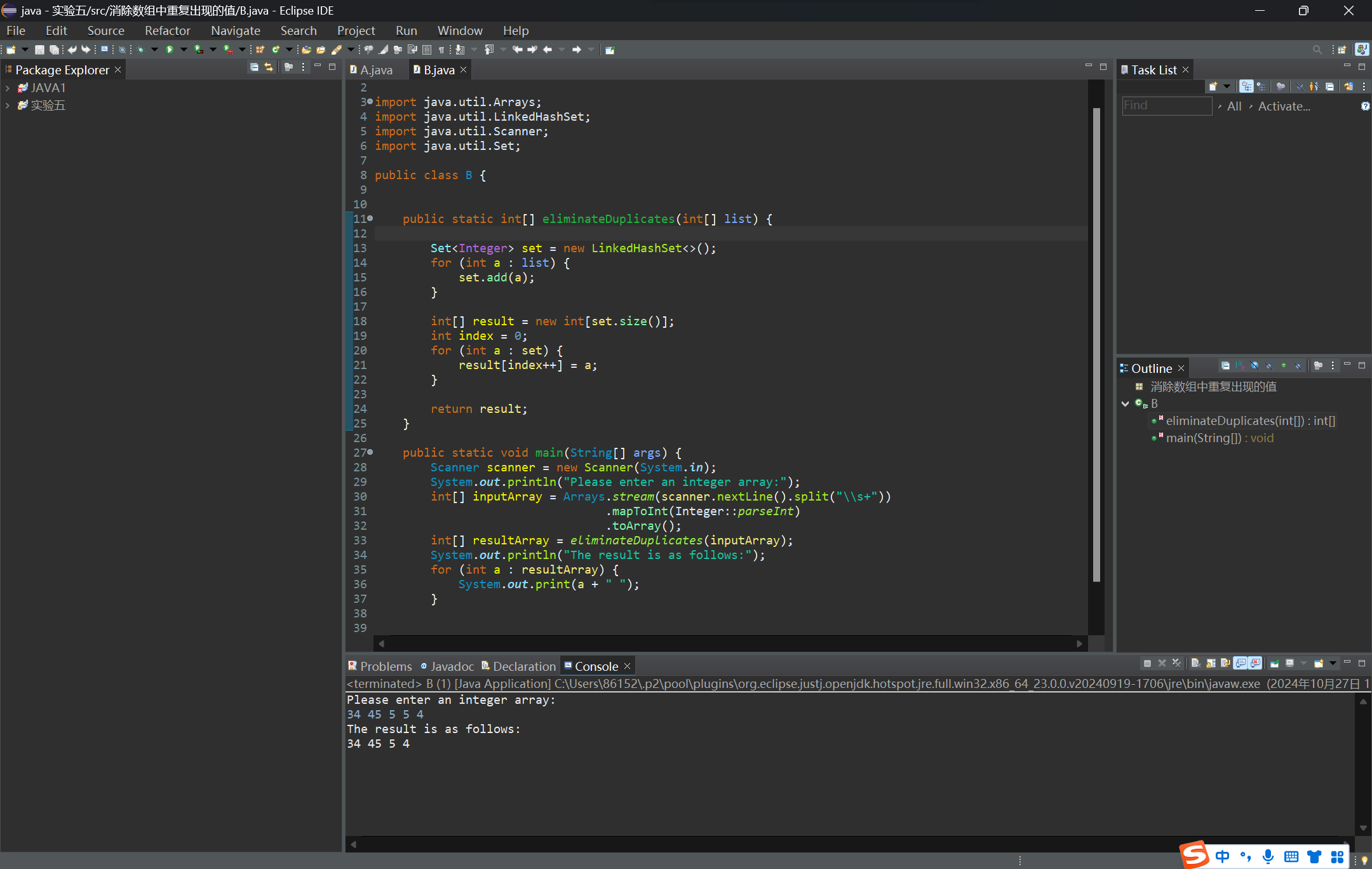
展示测试代码和结果，包括任何异常或错误情况的处理。

**四、实验结果 (程序的执行结果)**

**实验1**



**实验2**



1. **实验讨论（可选）**

通过本次实验，我们深入了解了三维张量的概念及其在多维数据分析中的重要性。实验要求我们能够熟练理解多维数组的特性，特别是三维张量与三维数组结构之间的关联关系。在此基础上，我们需要实现三维张量的相乘操作，该操作是通过三维数组的相加及相乘操作来实现的。

在实验实施过程中，我们首先根据给定的张量维度和元素，构建了相应的三维数组。随后，我们根据T-product的定义，通过实现unfold()、fold()以及bcirc()等函数，完成了两个三维张量的相乘操作。这些函数的实现不仅要求我们深入理解张量的结构和运算规则，还需要我们具备一定的编程能力和算法设计能力。

在实验过程中，我们遇到了一些挑战。例如，在实现unfold()函数时，我们需要将三维张量的某一维度展开成二维矩阵，这需要我们精确地控制数组的索引和维度变化。同样，在实现fold()函数时，我们需要将二维矩阵重新折叠回三维张量，这也需要我们准确地把握数组的维度和元素位置。此外，在实现bcirc()函数时，我们需要构造一个块循环矩阵，这要求我们具备一定的矩阵运算知识和编程技巧。

尽管在实验过程中遇到了一些困难，但我们通过不断尝试和调试，最终成功地实现了三维张量的相乘操作。通过与Matlab代码的比较和验证，我们确认了所编写程序的正确性和可靠性。

在实验结果的讨论中，我们注意到，三维张量的相乘操作具有广泛的应用价值。例如，在机器学习领域中，张量相乘可以用于实现神经网络的权重更新和特征提取；在计算机视觉领域中，张量相乘可以用于图像处理和模式识别等任务。因此，掌握三维张量的相乘操作对于提高我们的编程能力和解决实际问题的能力具有重要意义。

综上所述，本次实验不仅让我们深入了解了三维张量的概念和运算规则，还提高了我们的编程能力和算法设计能力。我们相信，在未来的学习和工作中，这些知识和技能将对我们产生积极的影响和帮助。

在本次实验中，我们深入探索了Java编程中数组操作的一个重要方面：如何有效地删除数组中的重复元素。通过编写和运行Java代码，我们不仅加深了对数组操作的理解，还提高了编程实践和问题解决能力。

在实验过程中，我们首先回顾了数组的基本概念，包括数组的声明、初始化和遍历。随后，我们聚焦于如何检测数组中的重复元素，并思考如何构建一个新的数组来存储这些不重复的元素。为了实现这一目标，我们采用了多种策略，如使用额外的数据结构（如HashSet）来记录已经遇到的元素，或者通过嵌套循环来比较数组中的每个元素。

在编写代码时，我们遇到了几个挑战。首先，我们需要确保算法的正确性，即新数组中不包含任何重复的元素，并且原始数组中的每个非重复元素都出现在新数组中。其次，我们需要考虑算法的效率，特别是当数组很大时，如何减少时间复杂度和空间复杂度。最后，我们还需要确保代码的可读性和可维护性，以便在需要时能够轻松地理解和修改代码。

为了验证算法的正确性，我们编写了一系列的测试用例，包括空数组、所有元素都相同的数组和包含多个不同元素的数组。通过运行这些测试用例，我们成功地验证了算法的正确性，并发现了几个潜在的错误和边界情况。通过不断地调试和修改代码，我们最终得到了一个稳定且高效的算法。

在实验结果的讨论中，我们注意到，删除数组中的重复元素是一个常见的问题，它在许多实际应用中都有出现。例如，在数据处理和分析中，我们经常需要从一个数据集中提取唯一的元素；在编程竞赛和算法挑战中，这个问题也经常出现作为测试编程技能和算法设计能力的题目。因此，掌握删除数组重复元素的技巧和方法对于我们提高编程能力和解决实际问题具有重要意义。

此外，我们还讨论了算法的优化和改进方法。例如，使用HashSet可以显著提高算法的效率，因为它允许我们在常数时间内检查一个元素是否已经存在于集合中。另外，我们还可以考虑使用其他数据结构或算法来进一步优化性能，如使用位运算或并行计算等技术。

综上所述，本次实验不仅让我们深入了解了Java中数组操作的基本概念和技巧，还提高了我们的编程实践和问题解决能力。我们相信，在未来的学习和工作中，这些知识和技能将对我们产生积极的影响和帮助。

实验代码

实验1

01 package 三维张量相乘;

02

03 import java.util.Scanner;

04

05 public class A

06 {

07

08

09

10 public static void main(String[] args)

11 {

12

13

14

15 Scanner scanner = new Scanner(System.in);

16

17

18

19 System.out.println("Enter the dimensions for tensor A (rows, columns, layers):");

20

21 int rowsA = scanner.nextInt();

22

23 int columnsA = scanner.nextInt();

24

25 int layersA = scanner.nextInt();

26

27

28

29 System.out.println("Enter the dimensions for tensor B (rows, columns, layers):");

30

31 int rowsB = scanner.nextInt();

32

33

34

35 int columnsB = scanner.nextInt();

36

37 int layersB = scanner.nextInt();

38

39 int[][][] A = new int[layersA][rowsA][columnsA];

40

41 int[][][] B = new int[layersB][rowsB][columnsB];

42

43

44

45 System.out.println("Enter the elements for tensor A:");

46

47 for (int i = 0; i < layersA; i++)

48 {

49

50 for (int j = 0; j < rowsA; j++)

51 {

52

53 for (int k = 0; k < columnsA; k++)

54 {

55

56 A[i][j][k] = scanner.nextInt();

57

58 }

59

60 }

61

62 }

63

64

65

66 System.out.println("Enter the elements for tensor B:");

67

68 for (int i = 0; i < layersB; i++)

69 {

70

71 for (int j = 0; j < rowsB; j++)

72 {

73

74 for (int k = 0; k < columnsB; k++)

75 {

76

77 B[i][j][k] = scanner.nextInt();

78

79 }

80

81 }

82

83 }

84

85 int[][][] C = tensorProduct(A, B);

86

87

88

89 outputTensor(C);

90

91 }

92

93

94

95 public static int[][][] tensorProduct(int[][][] tA, int[][][] tB)

96 {

97

98 if(tA[0][0].length != tB[0].length)

99 {

100

101 System.out.println("The tensor dimension is not consistent");

102

103 return **null**;

104

105 }

106

107

108

109 int[][][] result = new int[tA.length][][];

110

111

112

113 for(int k = 0; k < result.length; k++)

114 {

115

116 int[][] temp = new int[tA[0].length][tB[0][0].length];

117

118 for(int i = 0; i < result.length; i++)

119 {

120

121 temp = matrixAddition(temp,

122

123 matrixMuplication(tA[(result.length - i + k) % result.length], tB[i]));

124

125 }

126

127 result[k] = temp;

128

129 }

130

131

132

133 return result;

134

135 }

136

137

138

139 public static int[][] matrixMuplication(int[][] A, int[][] B)

140 {

141

142

143

144 if(A[0].length != B.length)

145 {

146

147 System.out.println("The matrix dimension is not consistent");

148

149 return **null**;

150

151 }

152

153

154

155 int[][] result = new int[A.length][B[0].length];

156

157 for(int i = 0; i < result.length; i++)

158 {

159

160 for(int j = 0; j < result[0].length; j++)

161 {

162

163 for(int k = 0; k < B.length; k++)

164 {

165

166 result[i][j] += A[i][k] \* B[k][j];

167

168 }

169

170 }

171

172 }

173

174

175

176 return result;

177

178 }

179

180

181

182

183 public static int[][] matrixAddition(int[][] A, int[][] B)

184 {

185

186 if(A[0].length != B[0].length || A.length != B.length)

187 {

188

189 System.out.println("The matrix dimension is not consistent");

190

191 return **null**;

192

193 }

194

195

196

197 int[][] result = new int[A.length][B[0].length];

198

199 for(int i = 0; i < result.length; i++)

200 {

201

202 for(int j = 0; j < result[0].length; j++)

203 {

204

205 result[i][j] = A[i][j] + B[i][j];

206

207 }

208

209 }

210

211

212

213 return result;

214

215 }

216

217

218

219 public static void outputTensor(int[][][] mat)

220 {

221

222 for(int k = 0; k < mat.length; k++)

223 {

224

225 System.out.println("The " + k + "th slice is:");

226

227 for(int i = 0; i < mat[k].length; i++)

228 {

229

230 for(int j = 0; j < mat[k][i].length; j++)

231 {

232

233 System.out.print(mat[k][i][j] + " ");

234

235 }

236

237 System.out.println();

238

239 }

240

241 }

242

243 }

244

245

246

247 }

实验2

01 package 消除数组中重复出现的值;

02

03 import java.util.Arrays;

04 import java.util.LinkedHashSet;

05 import java.util.Scanner;

06 import java.util.Set;

07

08 public class B

09 {

10

11

12 public static int[] eliminateDuplicates(int[] list)

13 {

14

15 Set<Integer> set = new LinkedHashSet<>();

16 for (int a : list)

17 {

18 set.add(a);

19 }

20

21 int[] result = new int[set.size()];

22 int index = 0;

23 for (int a : set)

24 {

25 result[index++] = a;

26 }

27

28 return result;

29 }

30

31 public static void main(String[] args)

32 {

33 Scanner scanner = new Scanner(System.in);

34 System.out.println("Please enter an integer array:");

35 int[] inputArray = Arrays.stream(scanner.nextLine().split("\\s+"))

36 .mapToInt(Integer::parseInt)

37 .toArray();

38 int[] resultArray = eliminateDuplicates(inputArray);

39 System.out.println("The result is as follows:");

40 for (int a : resultArray)

41 {

42 System.out.print(a + " ");

43 }

44

45

46 scanner.close();

47 }

48 }