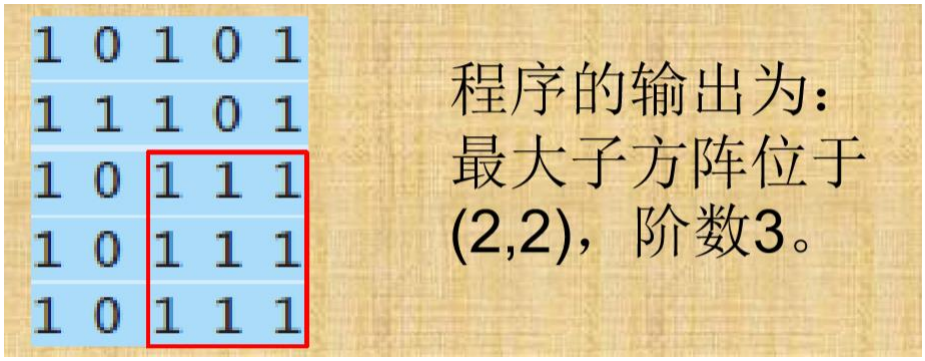
JAVA 实验报告

本文是 Java 实验的实验报告。

一、最大子方阵

题目描述：给定一个由 0，1 组成的 $n \times n$ 方阵（n 在运行时提醒用户输入），判断其中由全 1 组成的最大子方阵的左上角位置和阶数。例如用户输入 n 为 5，随机产生的方阵如下：



要求编写方法实现上述功能，返回值是一个包含 3 个元素的数组，依次表示行下标、列下标、阶数。

方法原型：`public static int[] findLargestBlock(int[][] m)`

解题思路：从 1 一直到 n，遍历每一种可能的子矩阵。

判断是否全为 1：如果该子矩阵的阶数比之前大，更新；

否则继续考察其它子矩阵。

源代码：

```
import java.util.Scanner;

public class Test1 {
    public static int[] findLargestBlock(int[][] m) {
        int[] maxSub = {0, 0, 0};

        for(int order = 1; order <= m.length; ++order) {
            for(int i = 0; i <= m.length - order; ++i) {
                for(int j = 0;j <= m[0].length - order; ++j) {
                    boolean isOne = true;
                    for(int r = i; r < i + order && isOne == true; ++r) {
                        for(int c = j; c < j + order && isOne == true; ++c) {
                            if(m[r][c] == 0)
                                isOne = false;
                        }
                    }
                    if(isOne == true && order > maxSub[0]) {
```

```
                        maxSub[0] = order;
                        maxSub[1] = i;
                        maxSub[2] = j;
                    }
                }
            }
        }

        return maxSub;
    }

    public static void main(String[] args) {
        int n = 0;
        Scanner input = new Scanner(System.in);

        System.out.print("Please input n: ");
        n = input.nextInt();

        int[][] m = new int[n][n];
        for(int i = 0; i < n; ++i) {
            for(int j = 0; j < n; ++j) {
                m[i][j] = (int)(Math.random() * 2);
            }
        }

        System.out.println("The matrix is: ");
        for(int i = 0; i < n; ++i) {
            for(int j = 0; j < n; ++j)
                System.out.print(m[i][j] + " ");
            System.out.print("\n");
        }

        int subMatrix[] = new int[3];
        subMatrix = findLargestBlock(m);

        System.out.println("The max submatrix is " + subMatrix[0] + " order, in
(" + subMatrix[1] + ", " +  subMatrix[2] + ")");
    }
}
```

测试样例：

```
Please input n: 5
The matrix is:
1 1 1 0 1
1 0 1 1 0
```

```
0 1 1 1 0
0 1 0 1 1
1 0 1 1 0
The max submatrix is 2 order, in (1, 2)
```

二、二次方程类

题目描述：设计一个二次方程类 QuadraticEquation，用于处理形如 $ax^2 + bx + c = 0$ ($a \neq 0$)的二次方程，成员如下：

- 私有成员 a, b, c 用于存储系数；
- 含三个参数的构造方法，用于传入 a, b, c；
- 三个方法 getA(), getB(), getC()，用于传出系数；
- 一个方法 getDiscriminant()用于传出 b^2 - 4ac 的值；
- 两个方法 getRoot1()和 getRoot2()用来返回方程的两个根。注意方程可能没有实根，所以返回值定义为 String。

请提供一个测试类，测试上述所有方法。

解题思路：自定义类 QuadraticEquation，包括私有成员 a, b, c；

函数：读入、读出数据；计算判别式；求根函数。

在测试类 Test 中，要分别考虑有不同实根，相同实根，无实根情况。

源代码：

```
import java.util.Scanner;

class QuadraticEquation {
    double a, b, c;
    void updateA(double newA) {
        a = newA;
    }
    void updateB(double newB) {
        b = newB;
    }
    void updateC(double newC) {
        c = newC;
    }

    double getA() {
        return a;
    }
    double getB() {
        return b;
    }
    double getC() {
      return c;
    }
    double getDiscriminant() {
```

```
        return b * b - 4 * a * c;
    }

    String getRoot1() {
        String root1;

        if(getDiscriminant() < 0) {
            root1 = "No real root";
        }
        else {
            root1 = Double.toString((-b + Math.sqrt(getDiscriminant())) / (2 *
a));
        }

        return root1;
    }

    String getRoot2() {
        String root2;

        if(getDiscriminant() < 0) {
            root2 = "No real root";
        }
        else {
            root2 = Double.toString((-b - Math.sqrt(getDiscriminant())) / (2 *
a));
        }

        return root2;
    }
}

public class Test {

    public static void main(String[] args) {
        double a, b, c;
        QuadraticEquation eq = new QuadraticEquation();
        Scanner input = new Scanner(System.in);

        System.out.print("Please input the coefficients: ");
        a = input.nextDouble();
        b = input.nextDouble();
        c = input.nextDouble();
        eq.updateA(a);
```

```
        eq.updateB(b);
        eq.updateC(c);

        System.out.println("The coefficients are: " + eq.getA() + " "+ eq.getB()
+ " " + eq.getC());

        System.out.println("The discriminant is: " + eq.getDiscriminant());

        System.out.println("Root1: " + eq.getRoot1());
        System.out.println("Root2: " + eq.getRoot2());
    }
}
```

测试样例 1：

```
Please input the coefficients: 1 3 2
The coefficients are: 1.0 3.0 2.0
The discriminant is: 1.0
Root1: -1.0
Root2: -2.0
```

测试样例 2：

```
Please input the coefficients: 1 1 1
The coefficients are: 1.0 1.0 1.0
The discriminant is: -3.0
Root1: No real root
Root2: No real root
```

测试样例 3：

```
Please input the coefficients: 1 4 4
The coefficients are: 1.0 4.0 4.0
The discriminant is: 0.0
Root1: -2.0
Root2: -2.0
```

三、 2*2 线性方程组

题目描述：设计一个类 LinearEquation 用于处理如下的 2*2 线性方程组，成员包含：

- 私有成员 a, b, c, d, e, f;
- 一个 6 参数构造方法，用于传入 a, b, c, d, e, f;
- 6 个 getter 用于返回 a, b, c, d, e, f，例如 getA(), getB(), …;
- 一个方法 isSolvable()用于判定方程是否有解，有则返回 true，否则 false;
- 方法 getX()和 getY()返回一组解。

请提供一个测试类，测试上述所有方法。

$$\begin{aligned} ax + by &= e \\ cx + dy &= f \end{aligned} \quad x = \frac{ed - bf}{ad - bc} \quad y = \frac{af - ec}{ad - bc}$$

解题思路：自定义类 LinearEquation

包括了题目要求方法。

在测试类 Test 中进行测试。

源代码：

```
import java.util.Scanner;

class LinearEquation {
    double a, b, c, d, e, f;

    LinearEquation(double newA, double newB, double newC, double newD, double newE,
double newF) {
        a = newA; b = newB; c = newC; d = newD; e = newE; f = newF;
    }

    double getA() {
        return a;
    }
    double getB() {
        return b;
    }
    double getC() {
        return c;
    }
    double getD() {
        return d;
    }
    double getE() {
        return e;
    }
    double getF() {
        return f;
    }

    boolean isSolvable() {
        return (a * d - b * c != 0);
    }

    String getX() {
        if(isSolvable() == true) {
            double x = (e * d - b * f) / (a * d - b * c);
            return Double.toString(x);
```

```
        }
        else {
            return "The equation has no solution";
        }
    }

    String getY() {
        if(isSolvable() == true) {
            double y = (a * f - e * c) / (a * d - b * c);
            return Double.toString(y);
        }
        else {
            return "The equation has no solution";
        }
    }
}

public class Test {
    public static void main(String[] args) {
        double a, b, c, d, e, f;
        Scanner input = new Scanner(System.in);

        System.out.print("Please input the coefficients: ");
        a = input.nextDouble();
        b = input.nextDouble();
        c = input.nextDouble();
        d = input.nextDouble();
        e = input.nextDouble();
        f = input.nextDouble();

        LinearEquation eq = new LinearEquation(a, b, c, d, e, f);

        System.out.println("The coefficients are: " +
                eq.getA() + " " + eq.getB() + " " + eq.getC() + " " + eq.getD() + "
" + eq.getE() + " " + eq.getF() + " ");

        System.out.println("The equation can be solved: " + eq.isSolvable());

        System.out.println("Root X = " + eq.getX());
        System.out.println("Root Y = " + eq.getY());
    }
}
```

测试样例：

样例 1：

```
Please input the coefficients: 1 1 1 1 1 1
The coefficients are: 1.0 1.0 1.0 1.0 1.0 1.0
The equation can be solved: false
Root X = The equation has no solution
Root Y = The equation has no solution
```

样例 2：

```
Please input the coefficients: 1 1 3 1 -1 2
The coefficients are: 1.0 1.0 3.0 1.0 -1.0 2.0
The equation can be solved: true
Root X = 1.5
Root Y = -2.5
```

四、 Location 类

题目描述：定义一个 Location 类，用于搜索二维数组的最大元素出现的位置和值。位置用公有的整型成员变量 row, col 表示，最大值用公有的浮点型成员变量 maxValue 表示。一个成员方法用来求解二维数组的最大元素及其位置，原型如下：

```
public static Location locateLargest(double[][] a)
```

例如数组为{{1,2,3},{8,9,9,5},{4,3,5,7,8}}，最大元素为 9，位置是(1,1)。注意最大值不止一个的时候，只记录第一次出现的位置。

请提供一个测试类，测试上述方法。

解题思路：逐一扫描矩阵中的各个元素，找到最大值并记录其下标。

源代码：

```
import java.util.Scanner;

class Location {
    int row;
    int column;
    double maxValue;

    public static Location locateLargest(double[][] a) {
        Location loc = new Location();
        loc.row = 0;
        loc.column = 0;
        loc.maxValue = Double.MIN_VALUE;

        for(int i = 0;i < a.length; ++i) {
            for(int j = 0;j < a[0].length; ++j) {
                if(a[i][j] > loc.maxValue) {
                    loc.maxValue = a[i][j];
                    loc.row = i;
                    loc.column = j;
                }
            }
```

```
        }
        return loc;
    }
}

public class Test {
    public static void main(String[] args) {
        final int M = 10;
        final int N = 10;

        double[][] m = new double[M][N];
        Scanner input = new Scanner(System.in);
        int row, column;

        System.out.print("Enter the number of rows and columns of the array: ");
        row = input.nextInt();
        column = input.nextInt();

        System.out.println("Enter the array: ");
        for(int i = 0;i < row;++i) {
            for(int j = 0;j < column;++j) {
                m[i][j] = input.nextDouble();
            }
        }

        Location loc;
        loc = Location.locateLargest(m);
        System.out.println("The location of the largest element is " +
loc.maxValue +
                " at (" + loc.row + ", " + loc.column + ")");
    }
}
```

测试样例：

```
Enter the number of rows and columns of the array: 3 5
Enter the array:
1 2 3 3 9 9 5 4 3 5 7 8 1 1 1
The location of the largest element is 9.0 at (0, 4)
```