给定一组不等式，判断是否成立并输出不等式的最大差(输出浮点数的整数部分)

要求:

1. 不等式系数为 double类型，是一个二维数组
2. 不等式的变量为 int类型，是一维数组;
3. 不等式的目标值为 double类型，是一维数组
4. 不等式约束为字符串数组，只能是:“>”,“>=”,“<”,“<=”,“=”,

例如，不等式组:

a11x1+a12x2+a13x3+a14x4+a15x5<=b1;

a21x1+a22x2+a23x3+a24x4+a25x5<=b2;

a31x1+a32x2+a33x3+a34x4+a35x5<=b3;

最大差 = max{(a11x1+a12x2+a13x3+a14x4+a15x5-b1),(a21x1+a22x2+a23x3+a24x4+ a25x5-b2),(a31x1+a32x2+a33x3+a34x4+a35x5-b3)},

类型为整数(输出浮点数的整数部分)

输入描述

a11,a12,a13,a14,a15,a21,a22,a23,a24,a25, a31,a32,a33,a34,a35,x1,x2,x3,x4,x5,b1,b2,b3,<=,<=,<=

1)不等式组系数(double类型):

a11,a12,a13,a14,a15

a21,a22,a23,a24,a25

a31,a32,a33,a34,a35

2)不等式变量(int类型):x1,x2,x3,x4,x5

3)不等式目标值(double类型):b1,b2,b3

4)不等式约束(字符串类型):<=,<=,<=

输出描述

true或者 false，最大差

用例

输入	2.3,3,5.6,7,6;11,3,8.6,25,1;0.3,9,5.3,66,7.8;1,3,2,7,5;340,670,80.6;<=,<=,<=
输出	false,458
说明	无

输入	2.36,3,6,7.1,6;1,30,8.6,2.5,21;0.3,69,5.3,6.6,7.8;1,13,2,17,5;340,67,300.6;<=,>=,<=
输出	false 758
说明	无

题目解析

这题是个纸老虎，大家不要怕。

答案都在题目里。

JavaScript算法源码

```javascript
/* JavaScript Node ACM模式 控制台输入获取 */
const readline = require("readline");

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});

rl.on("line", (line) => {
  const arr = line.split(";").map((str) => str.split(","));

  const [a11, a12, a13, a14, a15] = arr[0].map(Number);
  const [a21, a22, a23, a24, a25] = arr[1].map(Number);
  const [a31, a32, a33, a34, a35] = arr[2].map(Number);
  const [x1, x2, x3, x4, x5] = arr[3].map(Number);
  const [b1, b2, b3] = arr[4].map(Number);
  const [y1, y2, y3] = arr[5];

  let diff1 = a11 * x1 + a12 * x2 + a13 * x3 + a14 * x4 + a15 * x5 - b1;
  let diff2 = a21 * x1 + a22 * x2 + a23 * x3 + a24 * x4 + a25 * x5 - b2;
  let diff3 = a31 * x1 + a32 * x2 + a33 * x3 + a34 * x4 + a35 * x5 - b3;

  const flag =
    compareWithZero(diff1, y1) &&
    compareWithZero(diff2, y2) &&
    compareWithZero(diff3, y3);

  const maxDiff = Math.max(diff1, diff2, diff3);

  console.log(`${flag} ${Math.floor(maxDiff)}`);
});

function compareWithZero(val, constraint) {
  let flag;
  switch (constraint) {
    case ">":
      flag = val > 0;
      break;
    case ">=":
      flag = val >= 0;
      break;
    case "<":
      flag = val < 0;
      break;
    case "<=":
      flag = val <= 0;
      break;
    case "=":
      flag = val === 0;
      break;
  }
  return flag;
}
```

Java算法源码

```java
import java.util.Arrays;
import java.util.Scanner;

public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    String[][] arr =
        Arrays.stream(sc.nextLine().split(";")).map(s -> s.split(",")).toArray(String[][]::new);

    Double[] a1 = Arrays.stream(arr[0]).map(Double::parseDouble).toArray(Double[]::new);
    Double[] a2 = Arrays.stream(arr[1]).map(Double::parseDouble).toArray(Double[]::new);
    Double[] a3 = Arrays.stream(arr[2]).map(Double::parseDouble).toArray(Double[]::new);
    Double[] x = Arrays.stream(arr[3]).map(Double::parseDouble).toArray(Double[]::new);
    Double[] b = Arrays.stream(arr[4]).map(Double::parseDouble).toArray(Double[]::new);
    String[] y = arr[5];

    double diff1 = a1[0] * x[0] + a1[1] * x[1] + a1[2] * x[2] + a1[3] * x[3] + a1[4] * x[4] - b[0];
    double diff2 = a2[0] * x[0] + a2[1] * x[1] + a2[2] * x[2] + a2[3] * x[3] + a2[4] * x[4] - b[1];
    double diff3 = a3[0] * x[0] + a3[1] * x[1] + a3[2] * x[2] + a3[3] * x[3] + a3[4] * x[4] - b[2];

    boolean flag =
        compareWithZero(diff1, y[0])
            && compareWithZero(diff2, y[1])
            && compareWithZero(diff3, y[2]);

    double maxDiff = Math.max(Math.max(diff1, diff2), diff3);

    System.out.println(flag + " " + (int) Math.floor(maxDiff));
  }

  public static boolean compareWithZero(double val, String constraint) {
    boolean flag = false;

    switch (constraint) {
      case ">":
        flag = val > 0;
        break;
      case ">=":
        flag = val >= 0;
        break;
      case "<":
        flag = val < 0;
        break;
      case "<=":
        flag = val <= 0;
        break;
      case "=":
        flag = val == 0;
        break;
    }

    return flag;
  }
}
```

Python算法源码

```python
# 输入获取
arr = list(map(lambda s: s.split(","), input().split(";")))


def compareWithZero(val, constraint):
    if constraint == ">":
        return val > 0
    elif constraint == ">=":
        return val >= 0
    elif constraint == "<":
        return val < 0
    elif constraint == "<=":
        return val <= 0
    elif constraint == "=":
        return val == 0
    else:
        return False


# 算法入口
def getResult(arr):
    a11, a12, a13, a14, a15 = map(float, arr[0])
    a21, a22, a23, a24, a25 = map(float, arr[1])
    a31, a32, a33, a34, a35 = map(float, arr[2])
    x1, x2, x3, x4, x5 = map(float, arr[3])
    b1, b2, b3 = map(float, arr[4])
    y1, y2, y3 = arr[5]

    diff1 = a11 * x1 + a12 * x2 + a13 * x3 + a14 * x4 + a15 * x5 - b1
    diff2 = a21 * x1 + a22 * x2 + a23 * x3 + a24 * x4 + a25 * x5 - b2
    diff3 = a31 * x1 + a32 * x2 + a33 * x3 + a34 * x4 + a35 * x5 - b3

    flag = compareWithZero(diff1, y1) and compareWithZero(diff2, y2) and compareWithZero(diff3, y3)

    maxDiff = max(diff1, diff2, diff3)

    print(f"{flag} {int(maxDiff)}".lower())


# 算法调用
getResult(arr)
```