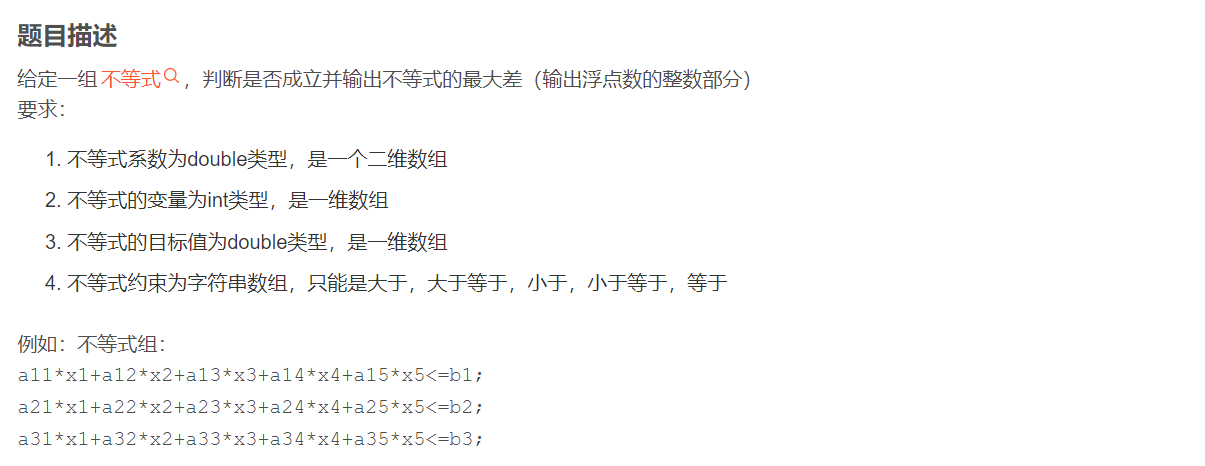
# **E卷-判断一组不等式是否满足约束[100分]（ Java | Python3 | C++ | C语言 | JsNode | Go ）**



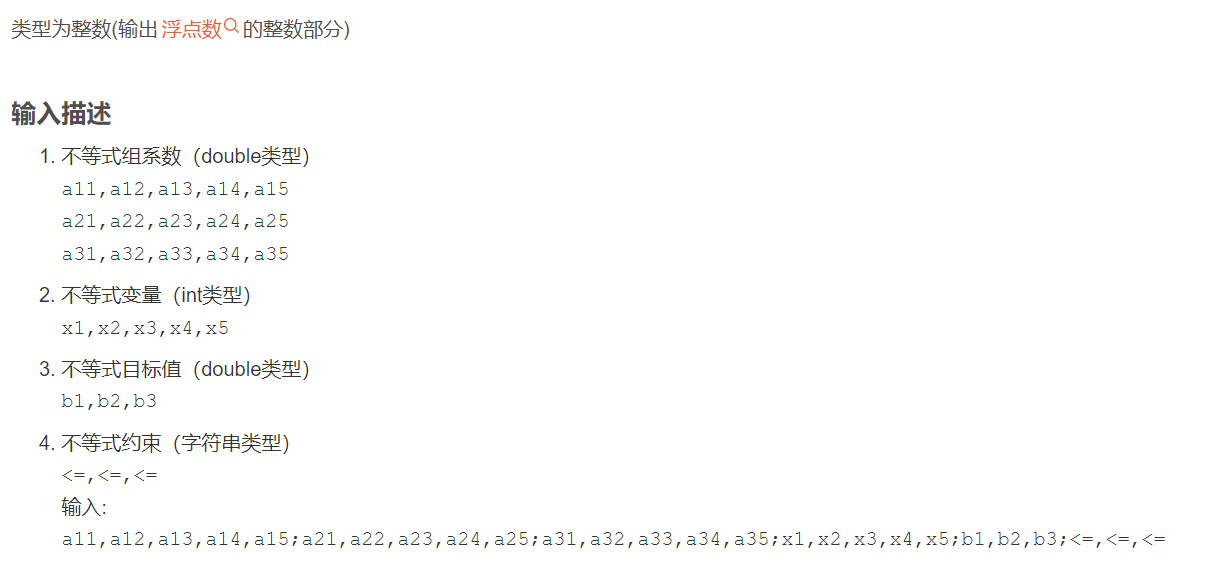
最大差 = max{

(a11\*x1+a12\*x2+a13\*x3+a14\*x4+a15\*x5-b1),

(a21\*x1+a22\*x2+a23\*x3+a24\*x4+a25\*x5-b2),

(a31\*x1+a32\*x2+a33\*x3+a34\*x4+a35\*x5-b3)

},

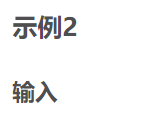




2.3,3,5.6,7,6;11,3,8.6,25,1;0.3,9,5.3,66,7.8;1,3,2,7,5;340,670,80.6;<=,<=,<=



false 458



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;<=,>=,<=



false 758



import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class Main {

// 根据传入的字符串返回对应的状态码

static int state(String s) {

// 如果字符串为">"，返回状态码 1

if (s.equals(">")) return 1;

// 如果字符串为">="，返回状态码 2

else if (s.equals(">=")) return 2;

// 如果字符串为"<"，返回状态码 3

else if (s.equals("<")) return 3;

// 如果字符串为"<="，返回状态码 4

else if (s.equals("<=")) return 4;

// 如果字符串为"="，返回状态码 5

return 5;

}

// 根据传入的两个数和状态码判断是否满足条件

static boolean judge(double a, double b, int state) {

// 根据状态码进行不同的判断

if (state == 1) return a > b;

else if (state == 2) return a >= b;

else if (state == 3) return a < b;

else if (state == 4) return a <= b;

else return a == b;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

// 读取一行输入

String s = sc.nextLine();

// 以";"分割输入字符串，得到各个部分

String[] p = s.split(";");

int Len = p.length;

// 最后一部分以","分割，得到每个约束条件的符号

String[] ys = p[Len - 1].split(",");

int n = ys.length;

// 第一部分以","分割，得到变量的个数

int m = p[0].split(",").length;

// 创建二维数组 A，用于存储约束条件中的系数

double[][] A = new double[n][m];

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

// 对每一部分以","分割，得到当前约束条件的系数

String[] now = p[i].split(",");

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

// 将系数转换为 double 类型并存入数组 A

A[i][j] = Double.parseDouble(now[j]);

}

// 创建数组 X，用于存储变量的值

int[] X = new int[m];

// 读取变量的值

String[] tmpX = p[n].split(",");

for (int i = 0; i < m; i++)

X[i] = Integer.parseInt(tmpX[i]);

// 创建数组 B，用于存储约束条件的右侧值

double[] B = new double[n];

// 读取约束条件的右侧值

String[] tmpB = p[n + 1].split(",");

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

B[i] = Double.parseDouble(tmpB[i]);

// 创建数组 ans，用于存储每个约束条件的差值

double[] ans = new double[n];

int flag = 0;

double MaxC = -1e18;

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

double sum = 0;

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

// 计算当前约束条件左侧的值

sum += A[i][j] \* X[j];

// 判断当前约束条件是否满足

if (!judge(sum, B[i], state(ys[i]))) flag = 1;

// 计算差值并存入数组 ans

ans[i] = sum - B[i];

// 更新最大差值

MaxC = Math.max(MaxC, ans[i]);

}

// 如果所有约束条件都满足，输出 true，否则输出 false

if (flag == 0) System.out.print("true ");

else System.out.print("false ");

// 输出最大差值

System.out.println((int) MaxC);

}

}



def state(s):

# 根据输入的字符串确定状态码

if s == ">":

return 1

elif s == ">=":

return 2

elif s == "<":

return 3

elif s == "<=":

return 4

else:

return 5

def judge(a, b, state\_value):

# 根据状态码判断两个数的关系是否满足条件

if state\_value == 1:

return a > b

elif state\_value == 2:

return a >= b

elif state\_value == 3:

return a < b

elif state\_value == 4:

return a <= b

else:

return a == b

s = input()

p = s.split(';')

Len = len(p)

ys = p[Len - 1].split(',')

n = len(ys)

m = len(p[0].split(','))

A = [[0] \* m for \_ in range(n)]

# 初始化二维数组 A，用于存储约束条件中的系数

for i in range(n):

now = p[i].split(',')

for j in range(m):

A[i][j] = float(now[j])

X = list(map(int, p[n].split(',')))

# 读取变量的值并存入列表 X

B = list(map(float, p[n+1].split(',')))

# 读取约束条件的右侧值并存入列表 B

ans = [0] \* n

flag = 0

MaxC = float('-inf')

# 初始化最大差值为负无穷大

for i in range(n):

sum\_val = 0

for j in range(m):

sum\_val += A[i][j] \* X[j]

# 判断当前约束条件是否满足

if not judge(sum\_val, B[i], state(ys[i])):

flag = 1

ans[i] = sum\_val - B[i]

# 更新最大差值

MaxC = max(MaxC, ans[i])

if flag == 0:

print("true", end=" ")

else:

print("false", end=" ")

print(int(MaxC))



#include <iostream>

#include <sstream>

#include <vector>

#include <climits>

#include <cmath>

int state(std::string s) {

// 根据输入字符串确定状态码

if (s == ">") return 1;

else if (s == ">=") return 2;

else if (s == "<") return 3;

else if (s == "<=") return 4;

return 5;

}

bool judge(double a, double b, int state\_value) {

// 根据状态码判断两个数的关系是否满足条件

if (state\_value == 1) return a > b;

else if (state\_value == 2) return a >= b;

else if (state\_value == 3) return a < b;

else if (state\_value == 4) return a <= b;

else return a == b;

}

int main() {

std::string s;

std::getline(std::cin, s);

std::vector<std::string> p;

std::stringstream ss(s);

std::string item;

// 将输入字符串按分号分割存入向量 p

while (std::getline(ss, item, ';')) {

p.push\_back(item);

}

int Len = p.size();

std::vector<std::string> ys;

std::stringstream yss(p[Len - 1]);

// 将最后一部分按逗号分割，得到每个约束条件的符号存入向量 ys

while (std::getline(yss, item, ',')) {

ys.push\_back(item);

}

int n = ys.size();

std::vector<std::string> first\_split;

std::stringstream fss(p[0]);

// 将第一部分按逗号分割，确定变量个数

while (std::getline(fss, item, ',')) {

first\_split.push\_back(item);

}

int m = first\_split.size();

std::vector<std::vector<double>> A(n, std::vector<double>(m));

// 初始化二维向量 A，用于存储约束条件中的系数

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

std::vector<std::string> now\_split;

std::stringstream nowss(p[i]);

while (std::getline(nowss, item, ',')) {

now\_split.push\_back(item);

}

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

A[i][j] = std::stod(now\_split[j]);

}

}

std::vector<int> X;

std::vector<std::string> X\_split;

std::stringstream Xss(p[n]);

// 读取变量的值存入向量 X

while (std::getline(Xss, item, ',')) {

X\_split.push\_back(item);

}

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

X.push\_back(std::stoi(X\_split[i]));

}

std::vector<double> B;

std::vector<std::string> B\_split;

std::stringstream Bss(p[n + 1]);

// 读取约束条件的右侧值存入向量 B

while (std::getline(Bss, item, ',')) {

B\_split.push\_back(item);

}

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

B.push\_back(std::stod(B\_split[i]));

}

std::vector<double> ans(n);

int flag = 0;

double MaxC = -1e18;

// 初始化最大差值为一个极小值

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

double sum = 0;

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

sum += A[i][j] \* X[j];

}

// 判断当前约束条件是否满足

if (!judge(sum, B[i], state(ys[i]))) flag = 1;

ans[i] = sum - B[i];

MaxC = std::max(MaxC, ans[i]);

}

if (flag == 0) std::cout << "true ";

else std::cout << "false ";

std::cout << (int)MaxC << std::endl;

return 0;

}



#include <stdio.h>

#include <stdlib.h>

#include <string.h>

// 根据传入的字符串返回对应的状态码

int state(char\* s) {

// 如果字符串为">"，返回状态码 1

if (strcmp(s, ">") == 0) return 1;

// 如果字符串为">="，返回状态码 2

else if (strcmp(s, ">=") == 0) return 2;

// 如果字符串为"<"，返回状态码 3

else if (strcmp(s, "<") == 0) return 3;

// 如果字符串为"<="，返回状态码 4

else if (strcmp(s, "<=") == 0) return 4;

// 如果字符串为"="，返回状态码 5

return 5;

}

// 根据传入的两个数和状态码判断是否满足条件

int judge(double a, double b, int st) {

// 根据状态码进行不同的判断

if (st == 1) return a > b;

else if (st == 2) return a >= b;

else if (st == 3) return a < b;

else if (st == 4) return a <= b;

else return a == b;

}

int main() {

// 读取输入行

char s[1024];

fgets(s, 1024, stdin);

// 以 ";" 分割输入字符串，得到各个部分

char\* p[1024];

int len = 0;

char\* token = strtok(s, ";");

while (token != NULL) {

p[len++] = token;

token = strtok(NULL, ";");

}

// 最后一部分以 "," 分割，得到每个约束条件的符号

int n, i, j, m;

char\* ys[1024];

n = 0;

token = strtok(p[len - 1], ",");

while (token != NULL) {

ys[n++] = token;

token = strtok(NULL, ",");

}

// 第一部分以 "," 分割，得到变量的个数

char\* vars[1024];

m = 0;

token = strtok(p[0], ",");

while (token != NULL) {

vars[m++] = token;

token = strtok(NULL, ",");

}

// 创建二维数组 A，用于存储约束条件中的系数

double A[100][100];

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

j = 0;

token = strtok(p[i], ",");

while (token != NULL) {

A[i][j++] = atof(token);

token = strtok(NULL, ",");

}

}

// 创建数组 X，用于存储变量的值

int X[100];

i = 0;

token = strtok(p[n], ",");

while (token != NULL) {

X[i++] = atoi(token);

token = strtok(NULL, ",");

}

// 创建数组 B，用于存储约束条件的右侧值

double B[100];

i = 0;

token = strtok(p[n + 1], ",");

while (token != NULL) {

B[i++] = atof(token);

token = strtok(NULL, ",");

}

// 创建数组 ans，用于存储每个约束条件的差值

double ans[100];

int flag = 0;

double MaxC = -1e18;

// 遍历每个约束条件

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

double sum = 0;

// 计算当前约束条件左侧的值

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

sum += A[i][j] \* X[j];

}

// 判断当前约束条件是否满足

if (!judge(sum, B[i], state(ys[i]))) flag = 1;

// 计算差值并存入数组 ans

ans[i] = sum - B[i];

// 更新最大差值

MaxC = MaxC > ans[i] ? MaxC : ans[i];

}

// 如果所有约束条件都满足，输出 true，否则输出 false

if (flag == 0) printf("true ");

else printf("false ");

// 输出最大差值

printf("%d\n", (int) MaxC);

return 0;

}



const readline = require('readline');

const rl = readline.createInterface({

input: process.stdin,

output: process.stdout

});

rl.question('', (s) => {

const p = s.split(';');

const Len = p.length;

const ys = p[Len - 1].split(',');

const n = ys.length;

const m = p[0].split(',').length;

const A = new Array(n).fill().map(() => new Array(m).fill(0));

// 初始化二维数组 A，用于存储约束条件中的系数

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

const now = p[i].split(',');

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

A[i][j] = parseFloat(now[j]);

}

}

const X = p[n].split(',').map(Number);

// 读取变量的值存入数组 X

const B = p[n + 1].split(',').map(Number);

// 读取约束条件的右侧值存入数组 B

const ans = new Array(n).fill(0);

let flag = 0;

let MaxC = -Infinity;

// 初始化最大差值为负无穷大

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

let sum = 0;

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

sum += A[i][j] \* X[j];

}

// 判断当前约束条件是否满足

if (!judge(sum, B[i], state(ys[i]))) flag = 1;

ans[i] = sum - B[i];

MaxC = Math.max(MaxC, ans[i]);

}

s = ""

if (flag === 0) s += "true ";

else s += "false ";

s += Math.floor(MaxC);

console.log(s);

rl.close();

});

function state(s) {

// 根据输入字符串确定状态码

if (s === ">") return 1;

else if (s === ">=") return 2;

else if (s === "<") return 3;

else if (s === "<=") return 4;

return 5;

}

function judge(a, b, state\_value) {

// 根据状态码判断两个数的关系是否满足条件

if (state\_value === 1) return a > b;

else if (state\_value === 2) return a >= b;

else if (state\_value === 3) return a < b;

else if (state\_value === 4) return a <= b;

else return a === b;

}



package main

import (

"fmt"

"strconv"

"strings"

)

func state(s string) int {

// 根据输入字符串确定状态码

if s == ">" {

return 1

} else if s == ">=" {

return 2

} else if s == "<" {

return 3

} else if s == "<=" {

return 4

}

return 5

}

func judge(a, b float64, stateValue int) bool {

// 根据状态码判断两个数的关系是否满足条件

if stateValue == 1 {

return a > b

} else if stateValue == 2 {

return a >= b

} else if stateValue == 3 {

return a < b

} else if stateValue == 4 {

return a <= b

} else {

return a == b

}

}

func main() {

var s string

fmt.Scanln(&s)

p := strings.Split(s, ";")

Len := len(p)

ys := strings.Split(p[Len-1], ",")

n := len(ys)

m := len(strings.Split(p[0], ","))

A := make([][]float64, n)

// 初始化二维切片 A，用于存储约束条件中的系数

for i := range A {

A[i] = make([]float64, m)

}

for i := 0; i < n; i++ {

now := strings.Split(p[i], ",")

for j := 0; j < m; j++ {

A[i][j], \_ = strconv.ParseFloat(now[j], 64)

}

}

X := make([]int, m)

tmpX := strings.Split(p[n], ",")

// 读取变量的值存入切片 X

for i := 0; i < m; i++ {

X[i], \_ = strconv.Atoi(tmpX[i])

}

B := make([]float64, n)

tmpB := strings.Split(p[n+1], ",")

// 读取约束条件的右侧值存入切片 B

for i := 0; i < n; i++ {

B[i], \_ = strconv.ParseFloat(tmpB[i], 64)

}

ans := make([]float64, n)

flag := 0

MaxC := -1e18

// 初始化最大差值为一个极小值

for i := 0; i < n; i++ {

sum := 0.0

for j := 0; j < m; j++ {

sum += A[i][j] \* float64(X[j])

}

if!judge(sum, B[i], state(ys[i])) {

flag = 1

}

ans[i] = sum - B[i]

MaxC = max(MaxC, ans[i])

}

if flag == 0 {

fmt.Print("true ")

} else {

fmt.Print("false ")

}

fmt.Println(int(MaxC))

}

func max(a, b float64) float64 {

// 返回两个数中的较大值

if a > b {

return a

}

return b

}