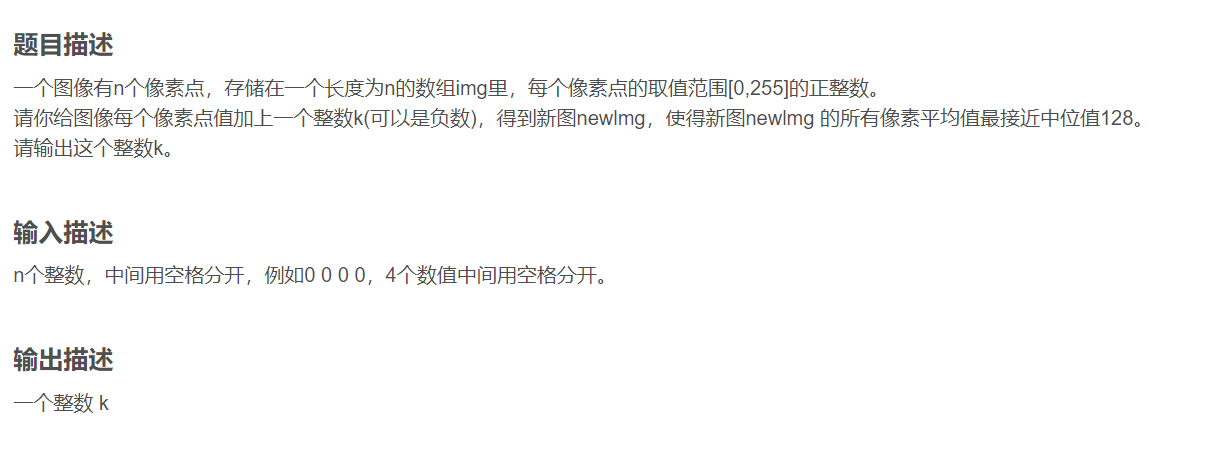
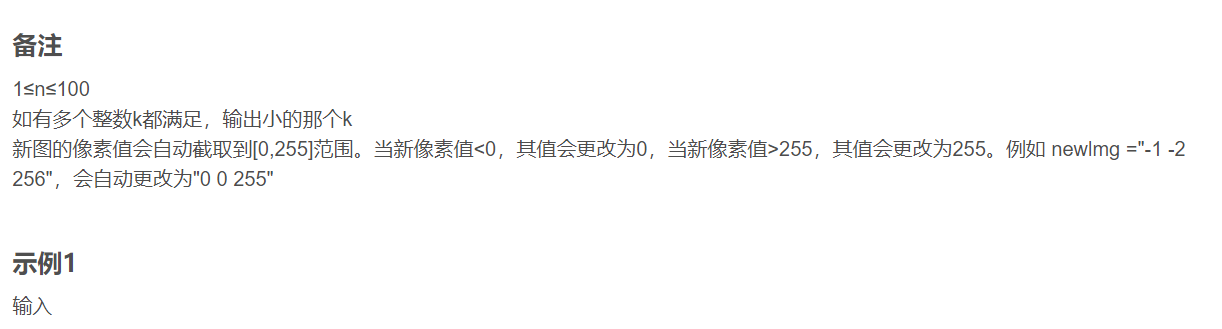
# **E卷-简单的自动曝光[100分]（ Java | Python3 | C++ | C语言 | JsNode | Go ）**





0 0 0 0



128



129 130 129 130



-2





import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in); // 创建一个扫描器从标准输入读取数据

List<Integer> img = new ArrayList<>(); // 创建一个列表用于存储n个像素点的值

// 读取输入的像素点值，假设它们在一行中用空格分隔

while (sc.hasNextInt()) {

img.add(sc.nextInt()); // 将每个输入的整数像素值添加到列表中

}

sc.close(); // 关闭扫描器

int n = img.size(); // 获取像素点的个数

// 定义变量maxx用于存储最大可能的平均值差异，初始值设为一个很大的整数

int maxx = Integer.MAX\_VALUE;

int index = -1; // 存储结果，即使新图像平均值最接近128的整数k

// 遍历从-255到255的可能k值

for (int i = -255; i <= 255; i++) {

int sum = 0; // 初始化累加器

// 遍历每个像素点

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

int q = img.get(j) + i; // 将当前k值添加到像素点上

if (q < 0) q = 0; // 若新值小于0，则截取为0

if (q > 255) q = 255; // 若新值大于255，则截取为255

sum += q; // 将新值累加到sum中

}

// 比较当前k值产生的平均值与目标值128的差异

int diff = Math.abs(sum - 128 \* n);

if (diff < maxx) {

maxx = diff; // 更新最大差异

index = i; // 更新最优k值

}

}

System.out.println(index); // 输出结果k值

}

}



# 导入 sys 库读取标准输入

import sys

# 从标准输入读取所有输入数据，假设输入的像素点值在一行中用空格分隔

input\_data = sys.stdin.read()

# 将输入的数据按空格分开，并转换为整数列表

img = list(map(int, input\_data.split()))

n = len(img) # 获取像素点的总数

max\_diff = float("inf") # 初始化最大差异为一个很大的值

best\_k = 0 # 存储结果，使新图像平均值最接近128的整数 k

# 遍历从 -255 到 255 的每个可能的 k 值

for k in range(-255, 256):

total = 0 # 用于累加每个像素点调整后的值

# 遍历每个像素点

for value in img:

adjusted\_value = value + k # 将当前 k 值添加到像素点上

if adjusted\_value < 0:

adjusted\_value = 0 # 若新值小于0，则截取为0

elif adjusted\_value > 255:

adjusted\_value = 255 # 若新值大于255，则截取为255

total += adjusted\_value # 累加到总和中

# 计算调整后的图像平均值与目标值128的差异

diff = abs(total - 128 \* n)

# 如果当前差异小于已记录的最小差异

if diff < max\_diff:

max\_diff = diff # 更新最小差异值

best\_k = k # 记录当前的 k 值为最优结果

# 输出使新图像平均值最接近128的 k 值

print(best\_k)



#include <iostream>

#include <vector>

#include <climits>

#include <cmath>

using namespace std;

int main() {

vector<int> img; // 初始化一个向量用于存储像素点值

int value;

// 从标准输入读取像素点值，假设值在一行中用空格分隔

while (cin >> value) {

img.push\_back(value); // 将每个读取的像素点值加入向量

}

int n = img.size(); // 获取像素点的总数

int max\_diff = INT\_MAX; // 初始化最小差异为一个很大的整数

int best\_k = 0; // 存储结果，使新图像平均值最接近128的整数 k

// 遍历从 -255 到 255 的每个可能的 k 值

for (int k = -255; k <= 255; k++) {

int total = 0; // 用于累加每个像素点调整后的值

// 遍历每个像素点

for (int val : img) {

int adjusted\_value = val + k; // 将当前 k 值添加到像素点上

if (adjusted\_value < 0) adjusted\_value = 0; // 若新值小于0，则截取为0

if (adjusted\_value > 255) adjusted\_value =

255; // 若新值大于255，则截取为255

total += adjusted\_value; // 累加到总和中

}

// 计算调整后的图像平均值与目标值128的差异

int diff = abs(total - 128 \* n);

// 如果当前差异小于已记录的最大差异

if (diff < max\_diff) {

max\_diff = diff; // 更新最大差异值

best\_k = k; // 记录当前的 k 值为最优结果

}

}

// 输出使新图像平均值最接近128的 k 值

cout << best\_k << endl;

return 0;

}



#include <stdio.h>

#include <stdlib.h>

#include <limits.h>

int main() {

int img[100]; // 初始化一个数组用于存储最多100个像素点值

int n = 0; // 用于记录像素点的数量

int value;

// 从标准输入读取像素点值，假设值在一行中用空格分隔

while (scanf("%d", &value) != EOF) {

img[n++] = value; // 将每个读取的像素点值加入数组

}

int max\_diff = INT\_MAX; // 初始化最小差异值为一个很大的整数

int best\_k = 0; // 存储结果，使新图像平均值最接近128的整数 k

// 遍历从 -255 到 255 的每个可能的 k 值

for (int k = -255; k <= 255; k++) {

int total = 0; // 用于累加每个像素点调整后的值

// 遍历每个像素点

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

int adjusted\_value = img[i] + k; // 将当前 k 值添加到像素点上

if (adjusted\_value < 0)

adjusted\_value = 0; // 若新值小于0，则截取为0

if (adjusted\_value > 255)

adjusted\_value = 255; // 若新值大于255，则截取为255

total += adjusted\_value; // 累加到总和中

}

// 计算调整后的图像平均值与目标值128的差异

int diff = abs(total - 128 \* n);

// 如果当前差异小于已记录的最大差异

if (diff < max\_diff) {

max\_diff = diff; // 更新最大差异值

best\_k = k; // 记录当前的 k 值为最优结果

}

}

// 输出使新图像平均值最接近128的 k 值

printf("%d\n", best\_k);

return 0;

}



const readline = require('readline');

// 创建接口用于读取标准输入

const rl = readline.createInterface({

input: process.stdin,

output: process.stdout

});

const img = []; // 初始化一个数组用于存储像素点值

// 读取标准输入数据，假设输入的像素点值在一行中用空格分隔

rl.on('line', function(line) {

// 将每个像素点值从字符串转换为整数并添加到数组中

img.push(...line.split(' ').map(Number));

rl.close();

});

rl.on('close', function() {

const n = img.length; // 获取像素点的总数

let max\_diff = Number.MAX\_SAFE\_INTEGER; // 初始化最大差异为一个很大的值

let best\_k = 0; // 存储结果，使新图像平均值最接近128的整数 k

// 遍历从 -255 到 255 的每个可能的 k 值

for (let k = -255; k <= 255; k++) {

let total = 0; // 用于累加每个像素点调整后的值

// 遍历每个像素点

for (let val of img) {

let adjusted\_value = val + k; // 将当前 k 值添加到像素点上

if (adjusted\_value < 0) adjusted\_value = 0; // 若新值小于0，则截取为0

if (adjusted\_value > 255) adjusted\_value = 255; // 若新值大于255，则截取为255

total += adjusted\_value; // 累加到总和中

}

// 计算调整后的图像平均值与目标值128的差异

let diff = Math.abs(total - 128 \* n);

// 如果当前差异小于已记录的最大差异

if (diff < max\_diff) {

max\_diff = diff; // 更新最大差异值

best\_k = k; // 记录当前的 k 值为最优结果

}

}

// 输出使新图像平均值最接近128的 k 值

console.log(best\_k);

});



package main

import (

"bufio"

"fmt"

"math"

"os"

"strconv"

"strings"

)

func main() {

reader := bufio.NewReader(os.Stdin) // 创建一个读者从标准输入读取数据

input, \_ := reader.ReadString('\n') // 读取输入数据

input = strings.TrimSpace(input) // 移除首尾空格

strValues := strings.Split(input, " ") // 按空格分隔输入数据

var img []int // 初始化一个切片用于存储像素点值

for \_, str := range strValues {

value, \_ := strconv.Atoi(str) // 将字符串转换为整数

img = append(img, value) // 将整数加入切片

}

n := len(img) // 获取像素点的总数

maxDiff := math.MaxInt32 // 初始化最大差异为一个很大的值

bestK := 0 // 存储结果，使新图像平均值最接近128的整数 k

// 遍历从 -255 到 255 的每个可能的 k 值

for k := -255; k <= 255; k++ {

total := 0 // 用于累加每个像素点调整后的值

// 遍历每个像素点

for \_, val := range img {

adjustedValue := val + k // 将当前 k 值添加到像素点上

if adjustedValue < 0 {

adjustedValue = 0 // 若新值小于0，则截取为0

} else if adjustedValue > 255 {

adjustedValue = 255 // 若新值大于255，则截取为255

}

total += adjustedValue // 累加到总和中

}

// 计算调整后的图像平均值与目标值128的差异

diff := int(math.Abs(float64(total - 128\*n)))

// 如果当前差异小于已记录的最大差异

if diff < maxDiff {

maxDiff = diff // 更新最大差异值

bestK = k // 记录当前的 k 值为最优结果

}

}

// 输出使新图像平均值最接近128的 k 值

fmt.Println(bestK)

}