题目描述

一个图像有n个像素点，存储在一个长度为n的数组img里，每个像素点的取值范围[0,255]的正整数。
请你给图像每个像素点值加上一个整数k（可以是负数），得到新图newImg，使得新图newImg的所有像素平均值最接近中位值128。
请输出这个整数k。

输入描述

n个整数，中间用空格分开

输出描述

一个整数k

备注

- 1 <= n <= 100
- 如有多个整数k都满足，输出小的那个k；
- 新图的像素值会自动截取到[0,255]范围。当新像素值<0，其值会更改为0；当新像素值>255，其值会更改为255；

例如newImg=”-1 -2 256”,会自动更改为”0 0 255”

用例

输入	0 0 0 0
输出	128
说明	四个像素值都为0

输入	129 130 129 130
输出	-2
说明	-1的均值128.5，-2的均值为127.5，输出较小的数-2

题目解析

本题如果用暴力法求解的话思路如下：

首先输入的老图片的像素值，应该是符合要求的，即像素点应都在[0,255]范围内，因此像素点最小值为0，最大值为255。

那么如果我们想让0接近中位值128，则需要加上128。

如果我们想让255接近中位值128，则需要减去127。

因此，k的取值范围应该在-127到128之间，这样的话，就可以保证每一个点都能接近到中位值。

那么我们就从-127遍历到128，然后将遍历值加到老图片的每一个像素值上，然后 求平均值 。

需要注意的是，如果新图片的像素点值低于0，则取0，高于255，则取255

```
function getResult(arr) {
  const ans = [];
  for (let k = -127; k <= 128; k++) {
    const res =
      arr
        .map((num) => {
          const newNum = num + k;
          return newNum < 0 ? 0 : newNum > 255 ? 255 : newNum;
        })
        .reduce((p, c) => p + c) / arr.length;
    ans.push([k, res]);
  }
  console.table(ans);
}

getResult([10, 20, 100, 200, 250]);
```

复制

题目解析

本题如果用暴力法求解的话思路如下：

首先输入的老图片的像素值，应该是符合要求的，即像素点应都在[0,255]范围内，因此像素点最小值为0，最大值为255。

那么如果我们想让0接近中位值128，则需要加上128。

如果我们想让255接近中位值128，则需要减去127。

因此，k的取值范围应该在-127到128之间，这样的话，就可以保证每一个点都能接近到中位值。

那么我们就从-127遍历到128，然后将遍历值加到老图片的每一个像素值上，然后 求平均值 。

需要注意的是，如果新图片的像素点值低于0，则取0，高于255，则取255

```
function getResult(arr) {
  const ans = [];
  for (let k = -127; k <= 128; k++) {
    const res =
      arr
        .map((num) => {
          const newNum = num + k;
          return newNum < 0 ? 0 : newNum > 255 ? 255 : newNum;
        })
        .reduce((p, c) => p + c) / arr.length;
    ans.push([k, res]);
  }
  console.table(ans);
}

getResult([10, 20, 100, 200, 250]);
```

如上面代码中，我们可以得到各种k加到图片像素点上后的图片像素平均值

上面测试代码结果显示，k取14时，老图片像素点的平均值为128.2，最接近中位值128。

上面 算法时间复杂度 是外层256，内层100（1 <= n <= 100），差不多两三万次循环，可以接受。

JavaScript算法源码

```
/* 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(Number);
  console.log(getResult(arr));
});

function getResult(arr) {
  const len = arr.length;
  let minDiff = Infinity;
  let ans;

  for (let k = -127; k <= 128; k++) {
    let sum = 0;
    for (let j = 0; j < len; j++) {
      let newVal = arr[j] + k;
      // 新图的像素值会自动截取到[0,255]范围。当新像素值<0，其值会更改为0；当新像素值>255，其值会更改为255；
      newVal = newVal < 0 ? 0 : newVal > 255 ? 255 : newVal;
      sum += newVal;
    }

    const diff = Math.abs(sum / len - 128);

    if (diff < minDiff) {
      minDiff = diff;
      ans = k;
    } else if (diff === minDiff) {
      // 如有多个整数k都满足，输出小的那个k
      ans = Math.min(ans, k);
    }
  }

  return ans;
}
```

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(Number);
  console.log(getResult(arr));
});

function getResult(arr) {
  const len = arr.length;
  let minDiff = Infinity;
  let ans;

  for (let k = -127; k <= 128; k++) {
    let sum = 0;
    for (let j = 0; j < len; j++) {
      let newVal = arr[j] + k;
      // 新图的像素值会自动截取到[0,255]范围，当新像素值<0，其值会更改为0；当新像素值>255，其值会更改为255；
      newVal = newVal < 0 ? 0 : newVal > 255 ? 255 : newVal;
      sum += newVal;
    }

    const diff = Math.abs(sum / len - 128);

    if (diff < minDiff) {
      minDiff = diff;
      ans = k;
    } else if (diff === minDiff) {
      // 如有多个整数k都满足，输出小的那个k
      ans = Math.min(ans, k);
    }
  }

  return ans;
}
```

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 str = sc.nextLine();
    Integer[] arr = Arrays.stream(str.split(" ")).map(Integer::parseInt).toArray(Integer[]::new);

    System.out.println(getResult(arr));
  }

  public static int getResult(Integer[] arr) {
    int len = arr.length;
    double minDiff = Integer.MAX_VALUE;
    Integer ans = null;

    for (int k = -127; k <= 128; k++) {
      double sum = 0;
      for (Integer val : arr) {
        int newVal = val + k;
        // 新图的像素值会自动截取到[0,255]范围，当新像素值<0，其值会更改为0；当新像素值>255，其值会更改为255；
        newVal = Math.max(0, Math.min(newVal, 255));
        sum += newVal;
      }

      double diff = Math.abs(sum / len - 128);

      if (diff < minDiff) {
        minDiff = diff;
        ans = k;
      } else if (diff == minDiff && ans != null) {
        // 如有多个整数k都满足，输出小的那个k
        ans = Math.min(ans, k);
      }
    }

    return ans;
  }
}
```

Python算法源码

```python
import sys
```

伏城之外 已关注 0 8 7 专栏目录 已订阅

Python算法源码

```python
import sys

# 输入获取
arr = list(map(int, input().split()))


# 算法入口
def getResult(arr):
    minDiff = sys.maxsize
    ans = None

    for k in range(-127, 129):
        sum = 0
        for j in range(len(arr)):
            # 新图的像素值会自动截取到[0,255]范围。当新像素值<0，其值会更改为0；当新像素值>255，其值会更改为255；
            newVal = min(max(0, arr[j] + k), 255)
            sum += newVal

        diff = abs(sum / len(arr) - 128)

        if diff < minDiff:
            minDiff = diff
            ans = k
        elif diff == minDiff and ans is not None:
            # 如有多个整数k都满足，输出小的那个k
            ans = min(ans, k)

    return ans


# 算法调用
print(getResult(arr))
```