

# 图灵编程 10 月 13 日普及组模拟赛

时间：2024 年 10 月 13 日 18:00 ~ 21:30

题目名称	计数	自由择数	卡牌策略	水题
题目类型	传统型	传统型	传统型	传统型
可执行文件名	count.exe	range.exe	card.exe	water.exe
输入文件名	count.in	range.in	card.in	water.in
输出文件名	count.out	range.out	card.out	water.out
每个测试点时限	1.0 秒	1.0 秒	1.0 秒	1.0 秒
内存限制	512 MiB	512 MiB	512 MiB	512 MiB
测试点数目	5	20	5	10
测试点是否等分	是	是	否	是

提交源程序文件名

对于 C++ 语言	count.cpp	range.cpp	card.cpp	water.cpp
-----------	-----------	-----------	----------	-----------

编译选项

对于 C++ 语言	-O2 -std=c++14 -Wl,--stack=536870912
-----------	--------------------------------------

## 计数 (count)

### 【题目背景】

你会数学吗？答案是显而易见的，你当然会。只是程序的数学，和数学课的数学，毕竟是两回事情，因而数学方法和编程方法并不具有完全的参考性质。

### 【题目描述】

就来一个具有比较明显做法的问题吧，要求很简单，只需要计算在  $[a, b]$  之间有多少对  $(i, j)$  满足： $i + j = c$  就可以了。

这个任务，简单吗？但是如果要求以  $O(1)$  的做法，又该怎么做呢？

不过你不需要以  $O(1)$  的方法实现这个任务，不过，作为代价，你还需要与此同时，分别计算出：

- $[a, b]$  之间有多少对  $(i, j)$  满足： $i + j = c$
- $[a, b]$  之间有多少对  $(i, j)$  满足： $i - j = c$
- $[a, b]$  之间有多少对  $(i, j)$  满足： $i \times j = c$
- $[a, b]$  之间有多少对  $(i, j)$  满足： $i / j = c$

### 【输入格式】

从文件 `count.in` 中读入数据。

本题存在多组输入

第一行会输入一个正整数  $T$ ，代表一共有  $T$  组数据

接下来一共  $T$  行，每行有三个正整数，分别是  $a, b, c$ ，三个数的定义已经写在上面

### 【输出格式】

输出到文件 `count.out` 中。

输出  $T$  行，每行输出 4 个数字，分别是  $i + j = c$  的数量， $i - j = c$  的数量， $i \times j = c$ ， $i / j = c$  的数量

### 【样例 1 输入】

```
1 6
2 1 1 2
3 2 2 4
4 1 3 2
5 1 10 3
6 3 1000 56
```

7 1024 4096 2048

### 【样例 1 输出】

```

1 1 0 0 0
2 1 0 1 0
3 1 1 2 1
4 2 7 2 3
5 51 942 4 15
6 1 1025 0 0

```

### 【样例 1 解释】

对于第二组数据，存在：

$$(2 + 2 = 4), (2 * 2 = 4)$$

对于第三组数据，存在：

$$(1 + 1 = 2), (3 - 1 = 2), (1 \times 2 = 2), (2 \times 1 = 2), (2 / 1 = 2)$$

### 【样例 2】

见选手目录下的 *count/count2.in* 与 *count/count2.ans*。

### 【子任务】

数据点	T	a	b	c
1,2	$T \leq 100$	$a \leq 100$	$a \leq b \leq 100$	$\leq 1000$
3		$a \leq 10^5$	$a \leq b \leq 10^5$	$\leq 10^5$
4		$a = 1$	$a \leq b \leq 10^{10}$	$\leq 10^{10}$
5	$T \leq 200$	$a \leq 10^{10}$		

## 自由择数 (range)

### 【题目背景】

你会编程吗？答案是显而易见的，你当然会。只是落于实处，问某个题目是否会做，确实也难以回答，毕竟会编程和写各种算法，说到底，是两回事情，重点是想到方法，而代码却是偏轻的。

### 【题目描述】

对于大家来说，给你一个一维数组，取得其中的最大的数，是一件很容易的事情。

如果是二维数组呢？取得其中最大的数，也是一件很容易的事情。

将问题再升级一下，取得二维数组每一行的最大值，也是很容易的。

现在将问题再次升级，你要在二维数组  $A$  的每一行中都取出一个数字。

当然，取出来的方法有很多种，你只需要找到让取出来的数中最大的数和最小的数之间的差值最小的一种取数方法即可。

### 【输入格式】

从文件 *range.in* 中读入数据。

第一行一共两个正整数， $n$  和  $m$ ，分别表示二维数组的行数和列数

接下来一共  $n$  行，每行  $m$  个整数，其中第  $i$  行第  $j$  列的元素为  $A_{i,j}$

### 【输出格式】

输出到文件 *range.out* 中。

你的取数方法下取出来的数中最大的数和最小的数之间的最小差值

### 【样例 1 输入】

```
1 3 4
2 1 3 5 7
3 2 10 -5 -100
4 50 1000 3 -6
```

### 【样例 1 输出】

```
1 1
```

## 【样例 1 解释】

选择第一行的第二个，第二行的第一个，第三行的第三个。

选出的数字为  $\{3, 2, 3\}$ ，最大的数为 3，最小的数为 2，差值为 1

可以证明这是最优解

## 【样例 2 输入】

```
1 5 2
2 1 6
3 9 12
4 3 7
5 6 8
6 5 3
```

## 【样例 2 输出】

```
1 4
```

一种选数的方法为  $\{6, 9, 7, 6, 5\}$ ，最大的数和最小的数的差值为  $9 - 5 = 4$

## 【样例 3】

见选手目录下的 *range/range3.in* 与 *range/range3.ans*。

## 【子任务】

数据点	n	m	$A_{i,j}$
1-4	$n = 2$	$m \leq 100$	$ A_{i,j}  \leq 10^5$
5-7	$n = 3$		
8-10		$m \leq 2000$	
11-14	$n \leq 100$		$ A_{i,j}  \leq 1$
15-20			$ A_{i,j}  \leq 10^5$

## 卡牌策略 (card)

### 【题目背景】

你会贪心吗？答案是显而易见的，你当然会，只是贪心是多变的，贪心是难以证明的，直接看到的做法并不保证其是正解，还是需要多加斟酌，确定正解。一瞬的灵感和真实，不可同论，是两回事情。

### 【题目描述】

就来 WDG 和你来一场游戏吧，WDG 和你进行，你先手。

场上平摊有  $n$  张牌，每张牌都有一个对应的分数  $A_i$ ，取得到这张牌，会让你的分数  $+A_i$ ，你的初始分数为 0，在游戏结束时你要让自己取得的分数尽可能高

在每回合中，你可以选择：

- 取走场上的任意一张牌，然后 WDG 取走任意一张牌
- 取走场上的任意  $x$  张牌，然后 WDG 取走任意  $y$  张牌

双方拿完牌后，这回合结束。

在游戏的过程中，可能会出现没牌可取的情况，一旦双方出现需要取的牌的和的数量大于场上剩余牌的数量的情况，则游戏立刻结束。

WDG 对于胜负其实无所谓，他不需要在比赛中取得高分，他只希望让你的分数尽可能低，他知道什么用方法可以让你的分数尽可能低。

所以，你最多可以拿到多少分呢？

### 【输入格式】

从文件 **card.in** 中读入数据。

第一行输入三个正整数  $n, x, y$

第二行输入  $n$  个整数，第  $i$  个数为  $A_i$

### 【输出格式】

输出到文件 **card.out** 中。

你最多可以取得的分数

### 【样例 1 输入】

```
1 8 2 3
2 1000 100 2 3 7 10 -8 - 3
```

**【样例 1 输出】**

```
1 1102
```

**【样例 1 解释】**

第一轮：你拿走 1000,100，然后 WDG 拿走 10,7,3，

第二轮：你拿走 2，WDG 拿走 -3，

接下来你要拿 2 张牌，拿不了了，游戏结束。

你获得 1102 分

**【样例 2 输入】**

```
1 7 2 4
2 9 4 5 9 1 9 2
```

**【样例 2 输出】**

```
1 22
```

**【样例 2 解释】**

第一轮：你拿走 9，WDG 拿走 9，

第二轮：你拿走 9,WDG 拿走 5，

第三轮：你拿走 4,WDG 拿走 2，

虽然场上还剩下一张牌，但是无论怎么分配不够给两个人，拿不了了，游戏结束。

你获得 22 分

**【样例 3】**

见选手目录下的 *card/card3.in* 与 *card/card3.ans*。

**【子任务 0】**

本题采取捆绑测试

Subtask 1(25pts):  $n \leq 20$

Subtask 2(20pts):  $n \leq 1000$

Subtask 3(15pts):  $A_i > 0, x \geq y$

Subtask 4(15pts):  $A_i > 0, x = 1$

Subtask 5(25pts): 无特殊限制

对于 100% 的数据,  $x + y \leq n \leq 5 \times 10^5, 2 \leq n, |A_i| \leq 10^6$



## 水题 (water)

### 【题目背景】

当然，前文的几点，其实又有些片面了，真的会存在某个万能的方法，可以解决万事万物，或者是解决某个类型的题目？也许有高级的做法来解决简单的题目（也就是牛刀杀鸡），不过学无止境，方法善变，你还是需要自行寻找一切。

### 【题目描述】

正如前文所说，只需要简单的变形，就可以让题目变难，例如走迷宫，求最短路，很简单吧，相信大家都会做。

现在给你一个  $n \times m$  的迷宫，你每秒可以向上下左右四个方向移动 1 格，要求从  $(x_a, y_a)$  出发到达  $(x_b, y_b)$  的最快时间。每个点都有高度  $A_{i,j}$ 。其中  $A_{i,j} = 0$  的是水池，你不能从水池走过，也不能呆在水池上

到了这里都很简单，我们给它添加一个条件：

- 现在开始洪水泛滥，一开始水平面的高度为 0，每秒水平面高度上升一格
- 因此当与水池相邻的地块的高度小于等于水平面时，洪水将淹没这个地块，让这个地块也变成水池

那么，基于这样的前提下，最快的时间是多少呢？

### 【输入格式】

从文件 `water.in` 中读入数据。

第一行输入六个正整数  $n, m, x_a, y_a, x_b, y_b$

接下来  $n$  行，每行  $m$  个非负整数，第  $i$  行第  $j$  个数为  $A_{i,j}$

### 【输出格式】

输出到文件 `water.out` 中。

$(x_a, y_a)$  出发到达  $(x_b, y_b)$  的最快时间，若到达不了，输出  $-1$

### 【样例 1 输入】

```
1 4 4 1 1 4 4
2 1 0 0 0
3 2 3 4 5
4 0 6 0 6
5 0 7 3 7
```

**【样例 1 输出】**

1 6

路径为:  $(1,1) \rightarrow (2,1) \rightarrow (2,2) \rightarrow (2,3) \rightarrow (2,4) \rightarrow (3,4) \rightarrow (4,4)$

**【样例 1 解释】****【样例 2 输入】**

```
1 4 4 1 1 4 4
2 1 0 0 0
3 1 2 4 5
4 0 6 0 6
5 0 7 3 7
```

**【样例 2 输出】**

1 -1

现在当你走到  $(2,1)$  时, 水会淹没当前位置, 导致实际上不能走到  $(2,1)$ , 因此无法走到终点

**【样例 3】**

见选手目录下的 *water/water3.in* 与 *water/water3.ans*。

**【子任务】**

对于 30% 的数据,  $n, m \leq 10$

另有 20% 的数据, 保证不会出现“洼地” ( $A_{i,j} > 0$  且直接相连的四个位置都大于  $A_{i,j}$ ),  $n, m \leq 100$

另有 20% 的数据,  $A_{i,j} \leq 10$

对于 100% 的数据,  $n, m \leq 1000, 0 \leq A_{i,j} \leq 10^5, x_a, x_b, y_a, y_b$  在迷宫范围内

本题数据比较大, 建议选手使用较快的读入方式