A 柠檬汽水

时空限制：1s 256MB

文件名

soda.in/soda.out/soda.cpp

题目描述

这是农场上一个炎热的夏日，Farmer John要给他的N头奶牛发柠檬汽水了！所有的N头奶牛（方便起见，编号为$1 \ldots N$）都喜欢柠檬汽水，只是有些喜欢的程度更高一些。具体地说，奶牛i为了获得柠檬汽水最多愿意排在w_i头奶牛之后。现在所有的N头奶牛都在田里，但是只要Farmer John敲响牛铃，这些奶牛就会立刻赶到FJ的柠檬汽水站。她们会在FJ开始分发柠檬汽水之前到达，但是没有两头奶牛会在同一时刻到达。此外，当奶牛i到达时，当且仅当至多有w_i头奶牛在排队时她会来排队。

Farmer John想要提前准备一定量的柠檬汽水，但是他不想浪费。排队的奶牛的数量可能取决于她们到达的顺序。帮助他求出最少可能的排队的奶牛数量。

输入格式

第一行包含N，第二行包含N个用空格分隔的整数$w_1, w_2, \ldots, w_N$。输入保证$1 \leq N \leq 10000$，此外对于每头奶牛i，$0 \leq w_i \leq 10^9$。

输出格式

输出在所有可能的奶牛到达顺序之下，最小可能的排队的奶牛数量。

样例 #1

样例输入 #1

```
5
7 1 400 2 2
```

样例输出 #1

```
3
```

提示

在这个情况下，可能最后仅有三头奶牛在队伍中（这也是最小可能值）。假设$w = 7$和$w = 400$的奶牛先到并等在队伍中。然后$w = 1$的奶牛到达并且会离开，这是由于已经有2头奶牛在队伍中了。然后$w = 2$的两头奶牛到达，一头留下排队，一头离开。

子任务

- 测试点 1-4 满足 $N \leq 500$。
- 测试点 5-10 没有额外限制。

B 捕蛇

时空限制：2s 256MB

文件名

snakes.in/snakes.out/snakes.cpp

题目描述

传说，数千年前圣帕特里克消灭了哞尔兰所有的蛇。然而，蛇们现在卷土重来了！圣帕特里克节是在每年的3月17日，所以Bessie要用彻底清除哞尔兰所有的蛇来纪念圣帕特里克。

Bessie装备了一个捕网，用来捕捉 N 组排成一行的蛇（ $1 \leq N \leq 400$ ）。Bessie必须按照这些组在这一行中出现的顺序捕捉每一组的所有蛇。每当Bessie抓完一组蛇之后，她就会将蛇放在笼子里，然后带着空的捕网开始捕捉下一组。

一个大小为 s 的捕网意味着Bessie可以抓住任意包含 g 条的一组蛇，其中 $g \leq s$ 。然而，每当Bessie用大小为 s 的捕网抓住了一组 g 条蛇，就意味着浪费了 $s-g$ 的空间。Bessie可以任意设定捕网的初始大小，并且她可以改变 K 次捕网大小（ $1 \leq K < N$ ）。

请告诉Bessie她捕捉完所有组的蛇之后可以达到的总浪费空间的最小值。

输入格式

输入的第一行包含 N 和 K 。

第二行包含 N 个整数 $a_1, \ldots, a_N$ ，其中 a_i （ $0 \leq a_i \leq 10^6$ ）为第 i 组蛇的数量。

输出格式

输出一个整数，为Bessie抓住所有蛇的总浪费空间的最小值。

样例 #1

样例输入 #1

```
6 2
7 9 8 2 3 2
```

样例输出 #1

```
3
```

提示

Bessie可以设置她的捕网开始时大小为7。当她抓完第一组蛇之后，她将她的捕网的大小调整为9，保持这个大小直到抓完第4组蛇，再将捕网大小调整为3。总浪费空间为
$(7-7)+(9-9)+(9-8)+(3-2)+(3-3)+(3-2)=3$ 。

子任务

- 测试点 1-4 满足 $N \leq 100$。
- 测试点 5-12 满足 $N \leq 400$。

C 哞网

时空限制：2s 256MB

文件名

network.in/network.out/network.cpp

题目描述

Farmer John 的 N 头奶牛，编号为 $1 \ldots N$ （ $2 \leq N \leq 10^5$ ），拥有一种围绕“哞网”，一些仅在组内互相交流却不与其他组进行交流的奶牛小组，组成的复杂的社交网络。

每头奶牛位于农场的二维地图上的不同位置 (x, y) ，并且我们知道有 M 对奶牛($1 \leq M < 10^5$)会相互哞叫。两头相互哞叫的奶牛属于同一哞网。

为了升级他的农场，Farmer John 想要建造一个四边与 x 轴和 y 轴平行的长方形围栏。Farmer John 想要使得至少一个哞网完全被围栏所包围（在长方形边界上的奶牛计为被包围的）。请帮助 Farmer John 求出满足他的要求的围栏的最小可能周长。有可能出现这一围栏宽为 0 或高为 0 的情况。

输入格式

输入的第一行包含 N 和 M 。以下 N 行每行包含一头奶牛的 x 坐标和 y 坐标（至多 10^8 的非负整数）。以下 M 行每行包含两个整数 a 和 b ，表示奶牛 a 和 b 之间有哞叫关系。每头奶牛都至少存在一个哞叫关系，并且输入中不会出现重复的哞叫关系。

输出格式

输出满足 Farmer John 的要求的围栏的最小周长。

样例 #1

样例输入 #1

```
7 5
0 5
10 5
5 0
5 10
6 7
8 6
8 4
1 2
2 3
3 4
5 6
7 6
```

样例输出 #1

```
10
```

子任务

- 测试点 1-3 满足 $N \leq 100, M \leq 100$。
- 测试点 4-5 满足 $N \leq 10000, M \leq 10000$。
- 测试点 6-10 没有额外限制。

D 冒泡排序

时空限制：1s 256MB

文件名

bubblesort.in/bubblesort.out/bubblesort.cpp

题目描述

留意着农场之外的长期职业生涯的可能性，奶牛Bessie开始在不同的在线编程网站上学习算法。

她到目前为止最喜欢的算法是“冒泡排序”。这是Bessie最初的对长度为N的数组A进行排序的奶牛码实现。

```
sorted = false
while (not sorted):
   sorted = true
   moo
   for i = 0 to N-2:
      if A[i+1] < A[i]:
         swap A[i], A[i+1]
         sorted = false
```

显然，奶牛码中的“moo”指令的作用只是输出“moo”。奇怪的是，Bessie看上去执着于在她的代码中的不同位置使用这个语句。

在用若干个数组测试了她的代码之后，Bessie得到一个有趣的观察现象：大的元素很快就会被拉到数组末尾，然而小的元素需要很长时间“冒泡”到数组的开头（她怀疑这就是为什么这个算法得名的原因）。为了实验和缓解这一问题，Bessie试着修改了她的代码，使代码在每次循环中向前再向后各扫描一次，从而无论是大的元素还是小的元素在每一次循环中都有机会被拉较长的一段距离。她的代码现在是这样的：

```
sorted = false
while (not sorted):
   sorted = true
   moo
   for i = 0 to N-2:
      if A[i+1] < A[i]:
         swap A[i], A[i+1]
   for i = N-2 downto 0:
      if A[i+1] < A[i]:
         swap A[i], A[i+1]
   for i = 0 to N-2:
      if A[i+1] < A[i]:
         sorted = false
```

给定一个输入数组，请预测Bessie修改后的代码会输出多少次“moo”。

输入格式

输入的第一行包含N（$1 \leq N \leq 100,000$）。接下来N行描述了$A[0] \ldots A[N-1]$，每个数都是一个范围为$0 \ldots 10^9$的整数。输入数据不保证各不相同。

输出格式

输出“moo”被输出的次数。

样例 #1

样例输入 #1

```
5
1
8
5
3
2
```

样例输出 #1

```
2
```

子任务

- 测试点 1-3 满足 $N \leq 10000$。
- 测试点 4-10 满足 $N \leq 10^5$。