Troceos Iguales W75823_es

Sea `V` un vector de enteros. Un `Trozo` del vector `V` es un subvector **máximo** en el que todos los elementos son iguales. Por ejemplo, el vector `V = 5 5 5 5 6 4 4 5 5 5 7 7` tiene 5 trozos:

```
5 5 5 5     6     4 4     5 5 5     7 7
```

Debes hacer una **función** que devuelva los trozos que tiene el vector `V`. Esto quiere decir que para cada trozo hay que devolver: el valor que contiene, la posición donde empieza y la posición donde acaba. Además, debes devolverlos (en un vector) según el orden del tamaño del trozo: de menor a mayor. En caso de empate en el tamaño, irá primero el trozo que tenga el valor más pequeño. En caso de empate en el tamaño y en el valor, irá primero el trozo que empiece en la posición más pequeña.

Por ejemplo, para el vector `V = 5 5 5 5 6 4 4 5 5 5 7 7` hay que devolver un vector que contenga esto (y en este orden):

6	4	4
4	5	6
7	10	11
5	7	9
5	0	3

Cada fila corresponde a un trozo. En primera posición está el trozo más pequeño: el que tiene un 6 y va de la posición 4 a la posición 4 (tiene tamaño 1). El siguiente es el que tiene un 4 y va de la posición 5 a la posición 6 (tiene tamaño 2), etcétera hasta el último, que es el que tiene un 5 y va de la posición 0 a la posición 3 (tiene tamaño 4). Fíjate en que el segundo trozo más pequeño (4 5 6) y el tercero (7 10 11) tienen el mismo tamaño, pero (4 5 6) va antes que (7 10 11) porque $4 \leq 7$.
Haz una **función** `bocins` con la siguiente declaración:

```
/*
   Devuelve un vector con el valor, inicio y final de los trozos
   del vector v, ordenados por tamaño, de menor a mayor.
   En caso de empate en el tamaño, va primero el trozo con el valor más pequeño
   En caso de empate en el tamaño y en el valor,
   irá primero el trozo que empiece en la posición más pequeña.
*/

vector<Boci> bocins(const vector<int>& v)
```

Para que tu función compile, deberás usar la tupla `Boci`. **Es necesario añadir** el siguiente código al fichero que enviarás al juez:

```
#ifndef BOCI
#define BOCI
```

```
struct Boci
{
    int valor;
    int inici;
    int final;
};
#endif
```

Te recomendamos que copies y pegues este código del fichero `main.cc` que te damos.

Observación

Debes enviar un fichero que contenga **únicamente**:

1. la función que te pedimos.
2. las funciones auxiliares que hayas declarado (si las hay).
3. los `include` necesarios.
4. el código para declarar la tupla `Boci` (que se encuentra en el `main.cc`).

No debes poner el `main` en el fichero que enviarás, porque si lo haces, el `juez` te dará error.
No se puede usar la ordenación de C++: `std::sort`. Si quieres ordenar un vector, debes implementarlo tú (puedes usar cualquier algoritmo).
Si lo consideras conveniente, en este ejercicio se pueden usar: el método `push_back()` de la clase `vector`, `min`, `max` o `swap`.

Entrada

Un vector de enteros. El vector tiene al menos dos elementos.

Salida

El valor, inicio y final de los trozos del vector v, ordenados por tamaño, de menor a mayor. En caso de empate, el valor más pequeño va primero. En caso de empate en el tamaño y en el valor, irá primero el trozo que empiece en la posición más pequeña.

Ejemplo de entrada 1

```
12

5 5 5 5 6 4 4 5 5 5 7 7

10

1 2 3 4 5 6 7 8 9 10

5
1 1 1 1 1

4
4 3 2 1
```

Ejemplo de salida 1

```
6 4 4
4 5 6
7 10 11
5 7 9
5 0 3


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

```
9 8 8
10 9 9


1 0 4
```

```
1 3 3
2 2 2
3 1 1
4 0 0
```

Información del problema

Autoría: PRO1

Generación: 2026-01-25T13:20:06.978Z