C08-2: Método `count_interval`

Estructuras de Datos
Facultad de Informática - UCM

Esta semana se proponen dos ejercicios en el juez: `C08-1` y `C08-2`. Solamente es necesario realizar **uno de ellos**. La fecha tope de entrega es el **19 de abril de 2020**.

La entrega consiste en un único fichero `.cpp` que se subirá a *DOMjudge*. Puedes subir tantos intentos como quieras. Se tendrá en cuenta el último intento con el veredicto `CORRECT` que se haya realizado antes de la hora de entrega por parte de alguno de los miembros del grupo.

No olvides poner el nombre de los componentes del grupo en el fichero `.cpp`. Solo es necesario que uno de los componentes del grupo realice la entrega.

Evaluación: Este ejercicio se puntuará de 0 a 10. Para poder obtener una calificación superior a 0 es necesario obtener un veredicto `CORRECT`. Si se entregan los dos ejercicios (`C08-1` y `C08-2`), se contabilizará aquel con una calificación mayor.

Este ejercicio consiste en añadir un método a la clase `SetTree`. Recordemos que esta clase implementa el TAD Conjunto mediante árboles binarios de búsqueda. El método a añadir es el siguiente:

```
template <typename T>
class SetTree {
public:
  // ...
  int count_interval(const T &lower, const T &upper) const;
};
```

El método `count_interval` devuelve un número que indica cuántos elementos del conjunto `this` están comprendidos en el intervalo cerrado [`lower`, `upper`]. Por ejemplo, si `s` es una instancia de `SetTree` que denota el conjunto $\{1, 3, 5, 8, 13\}$, la llamada `s.count_interval(3, 9)` devuelve 3, ya que hay tres elementos en el conjunto `s` comprendidos entre 3 y 9.

Procede del siguiente modo:

1. En la plantilla que se proporciona en el Campus Virtual, implementa el método `count_interval` pedido.
2. Determina el coste en tiempo de la función anterior, en función del número de elementos del conjunto de entrada.
3. Comprueba la corrección de tu solución subiéndola al problema *DOMjudge* con identificador `C08-2`.

Entrada

La entrada consta de una serie de casos de prueba, cada uno de ellos consistente en un conjunto de N números enteros y una serie de M intervalos sobre los que llamar sucesivamente al método `count_interval`.

Cada caso de prueba comienza los números N (cardinal del conjunto) y M (número de intervalos). A continuación vienen N números enteros, que son los elementos del conjunto, seguidos de M pares de números, cada uno de ellos representando un intervalo. Se garantiza que el primer elemento de cada par es menor o igual que el segundo elemento.

Salida

Para cada intervalo del caso de prueba se escribirá una línea con un número de entero, que indica el número de elementos comprendidos en dicho intervalo. Tras procesar todos los intervalos del caso de prueba, se escribirá una línea con tres guiones (---).

Entrada de ejemplo

```
5 3
10 43 54 41 34
6 12
13 45
35 39
3 2
20 17 15
15 19
17 20
0 1
10 20
```

Salida de ejemplo

```
1
3
0
---
2
2
---
0
---
```