## El Árbol Escorado T36888_es

Dado un árbol binario *a*, con números enteros positivos como valores de los nodos, queremos obtener su equivalente *escorado*. Veamos qué significa que un árbol esté *escorado*.
Empecemos definiendo el *peso* y el *tamaño* de un árbol binario: Los números en los nodos representan pesos. El *peso* de un árbol es la suma de los valores en sus nodos. El *tamaño* de un árbol es el número de nodos.
Queremos *escorar* el árbol de tal modo que, para todo nodo, el hijo izquierdo siempre pese menos que el hijo derecho. Así, si el peso del hijo izquierdo es superior al peso del hijo derecho, el árbol escorado será el árbol con la misma raíz, el hijo derecho como hijo izquierdo, y el hijo izquierdo como hijo derecho (se intercambian de lado a los hijos). En caso de que los dos hijos pesen lo mismo, los intercambiaremos si el tamaño del hijo izquierdo es mayor que el tamaño del hijo derecho.
Nótese que en el árbol habrá los mismos nodos antes y después de escorar, pero colocados de manera diferente, y también están los mismos caminos antes y después pero dispuestos de forma distinta.
Se pide, pues, escribir una función **`escorar(a)`** y añadirla al archivo **`code.py`**, tal y como se especifica en las Observaciones.

**Parámetros y retorno**

El parámetro de la función **`escorar(a)`** es una instancia de la clase `ArbreBinari` (que ya conocemos).
El valor en cada nodo del árbol es un número entero positivo.
La función **`escorar(a)`** debe retornar *tres cosas*: el árbol escorado (instancia de `ArbreBinari`), el peso de el árbol (un número entero) y el tamaño del árbol (un número entero).

### Entrada

La entrada en el programa será el preorden del árbol binario, con marca -1 para indicar los árboles vacíos (ya conocemos este formato de los ejercicios en clase de laboratorio).
Vea los ejemplos del juego de pruebas público.

### Salida

El programa debe escribir el preorden y el inorden del árbol escorado, con los valores de los nodos separado por ',', sin marcas para representar a los árboles vacíos. El preorden y el inorden deben estar en líneas separadas
Vea los ejemplos del juego de pruebas público.

### Observaciones

Debe descargar el archivo **`code.py`** (icono de la serpiente). Este archivo es un programa con **todo** lo necesario para ejecutar los juegos de prueba públicos. Sólo falta, claro, la función que se pide en el enunciado. Este archivo debe completarse con el código que falta, y eso, **todo**, es lo que ha de enviarse al Jutge como solución.

Dentro del archivo **`code.py`** hay la clase **`ArbreBinari`** que hemos trabajado en las clases de teoría. Hemos eliminado algunos métodos que seguro no tienen ninguna utilidad para

resolver este problema. No será necesario que su solución haga ningún *import* ni nada. Todo el código que necesita está dentro de **`code.py`**.

La eficiencia y calidad de la solución se tendrán en cuenta en la corrección manual.

### Ejemplo de entrada 1

```
11 2 3 -1 -1 10 -1 -1 3 5 -1 -1 4 -1 -1
```

### Ejemplo de salida 1

```
11,3,4,5,2,3,10
4,3,5,11,3,2,10
```

### Ejemplo de entrada 2

```
5 2 10 -1 -1 8 -1 -1 20 -1 -1
```

### Ejemplo de salida 2

```
5,20,2,8,10
20,5,8,2,10
```

### Ejemplo de entrada 3

```
5 2 -1 -1 -1
```

### Ejemplo de salida 3

```
5,2
5,2
```

### Ejemplo de entrada 4

```
10 2 3 -1 -1 5 -1 -1 2 3 -1 -1 5 -1 -1
```

### Ejemplo de salida 4

```
10,2,3,5,2,3,5
3,2,5,10,3,2,5
```

### Ejemplo de entrada 5

```
10 -1 -1
```

### Ejemplo de salida 5

```
10
10
```

### Información del problema

Autoría: Jordi Delgado (amb agraïments als professors de PRO2-GEI)

Generación: 2026-01-25T12:52:21.051Z