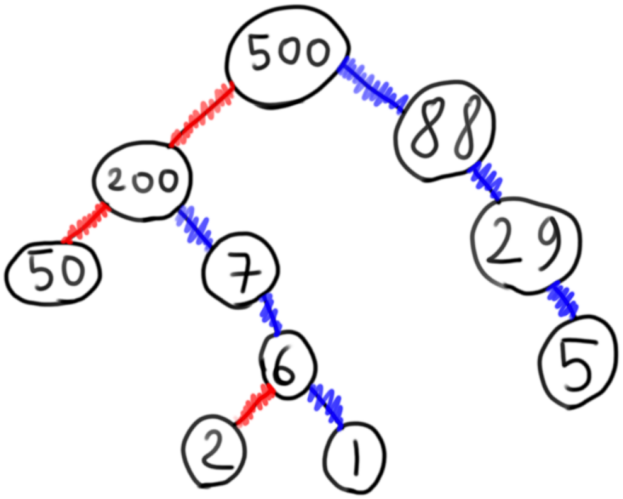
Recuperando el preorder

Contribución de Agustín Santiago Gutiérrez

Descripción del problema

Una noción importante en computación es la de *árbol binario*. En este problema, un árbol binario tiene *n nodos*, cada uno de los cuales puede tener opcionalmente un *hijo izquierdo* (rojo en la figura) y/o un *hijo derecho* (azul en la figura). Si existe, el hijo izquierdo/derecho de un nodo es otro nodo diferente.

La estructura del árbol es tal que existe un único nodo *raíz*, desde el cual es posible alcanzar todos los nodos directa o indirectamente bajando por hijos izquierdos y/o derechos. Además, ningún nodo puede ser hijo de más de un nodo. La raíz es el único nodo que no es hijo de ningún otro nodo.



En cada nodo del árbol hay escrito un número entero positivo. Hay dos maneras importantes de ordenar en una lista estos *n* números: Pre-order e in-order. En ambos casos, se inicia con la lista vacía y se procesan los nodos empezando por el nodo raíz. Luego en cada nodo:

- Pre-order: Para procesar un nodo, primero se agrega el número del nodo a la lista, luego se procesa de manera pre-order el hijo izquierdo (si existe), y finalmente se procesa de manera pre-order el hijo derecho (si existe).
- In-order: Para procesar un nodo, primero se procesa de manera in-order el hijo izquierdo (si existe), luego se agrega el número del nodo a la lista, y finalmente se procesa de manera in-order el hijo derecho (si existe).

Por ejemplo en el árbol de la figura anterior, el ordenamiento pre-order es $500, 200, 50, 7, 6, 2, 1, 88, 29, 5$. En cambio, el ordenamiento in-order es $50, 200, 7, 2, 6, 1, 500, 88, 29, 5$.

Otra noción importante en computación es la de *parva o heap*: Un árbol tiene la propiedad de heap, si en todos los nodos se cumple que su número es **mayor** que los números de sus hijos (cuando existen). Equivalentemente, en un árbol con la propiedad de heap **nunca puede haber un hijo con un número mayor o igual al de su padre**.

Debes implementar una función que, dado el ordenamiento in-order de un árbol binario que tiene la propiedad de heap, calcule el ordenamiento pre-order de ese árbol. **Se sabe que los *n* números son todos diferentes**.

Detalles de implementación

Se debe implementar la función:

`presort(inorder)`, que recibe en `inorder` un arreglo de *n* enteros distintos, que contienen los números de un árbol binario con la propiedad de heap, listados de acuerdo al ordenamiento in-order.

Debe devolver un arreglo con los mismos *n* enteros de la entrada, pero ahora según el ordenamiento pre-order.

Evaluador local

El evaluador local lee de la entrada estándar:

- Una línea con un entero n
- Una línea con n enteros, los valores del arreglo `inorder`

El evaluador escribe a la salida estándar una única línea, con el resultado retornado por la función `presort(inorder)`.

Restricciones

- $1 \leq n \leq 100.000$
- $1 \leq inorder[i] \leq 1.000.000.000$
- $inorder[i] \neq inorder[j]$ si $i \neq j$

Ejemplos

Si el evaluador local recibe la siguiente entrada:

```
10
50 200 7 2 6 1 500 88 29 5
```

Con una solución correcta escribe:

```
500 200 50 7 6 2 1 88 29 5
```

Subtareas

1. $n \leq 2$ (4 ptos.)
2. $n \leq 10$ (15 ptos.)
3. $n \leq 250$ (20 ptos.)
4. $n \leq 2000$ (29 ptos.)
5. Los números de `inorder` están ordenados, aunque no se sabe si en forma ascendente o descendente. (12 ptos.)
6. Sin más restricción (20 ptos.)