VIDEO EXPLICATIVO VIDEO

CONSIGNA Escribir una función

```
void partition(tree<int> &T,int n,map<int,string> &M);
```

que, dado un arbol `T` genera una correspondencia `M` que indica para cada nodo del árbol (es decir, para su etiqueta) el tipo de nodo que es, con respecto al nodo `n` (derecha, izquierda, antecesor propio, descendiente propio).

EJEMPLOS VER VIDEO:

```
T:(9 (3 (7 19 2) 0) (12 (5 (18 13)) 8)),n:5,
    => M:[0->IZQ,2->IZQ,3->IZQ,5->NOD,7->IZQ,8->DER,9->ANT,12->ANT,13->DES,18->DES,19->IZQ]
T:(9 (3 (7 19 2) 0) (12 (5 (18 13)) 8)),n:0,
    => M:[0->NOD,2->IZQ,3->ANT,5->DER,7->IZQ,8->DER,9->ANT,12->DER,13->DER,18->DER,19->IZQ]
```

Con respecto a los valores de la imagen de la correspondencia:

- NOD: indica el nodo `n` en cuestión
- IZQ: indica que el nodo está a la izquierda de `n`
- DER: indica que el nodo está a la derecha de `n`
- DES: indica el nodo es descendiente propio de `n`
- ANT: indica el nodo es antecesor propio de `n`

NOTAS

- Notar que el valor de la correspondencia es un string, indicando el tipo de nodo (con respecto a `n`)
- Los valores nodales son únicos, es decir que no puede haber dos nodos con la misma etiqueta (valor nodal).

AYUDA

- La forma de clasificar los nodos se va a basar en generar los órdenes previo y posterior del arbol y comparar su ubicacion con respecto a `n`.

- Notemos, por ejemplo, que los nodos antecesores propios de `n` están antes que `n` en orden previo y después en orden posterior.
- Los nodos descendientes propios de `n` están después que `n` en orden previo y antes en orden posterior.
- Los nodos de la izquierda de `n` están antes que `n` tanto en orden previo como en orden posterior.
- Los nodos de la derecha de `n` están después de `n` tanto en orden previo como en orden posterior.

Entonces el algoritmo consiste en lo siguiente:

- Recorrer el ábol en orden previo e ir almacenando los órdenes en un `map<int,int> ordpre` que indica para cada etiqueta su número de orden, en orden previo.
- Por ejemplo, si `T=(4 7 (9 8 6) 2)` entonces el listado en orden previo es `[4 7 9 8 6 2]` y entonces `ordpre=[2->5,4->0,6->4,7->1,8->3,9->2]`.
- Similarmente, construir una correspondencia `map<int,int> ordpost` con el orden posterior. El orden posterior es `[7 8 6 9 2 4]` y entonces `ordpost=[2->4,4->5,6->2,7->0,8->1,9->3]`.
- Ahora recorrer todas los valores nodales (puede ser recorriendo las claves de `ordpre` u `ordpost` (son las mismas, es decir las etiquetas nodales). Para cada valor nodal `m` comparar el su orden previo previo y posterior con el de `n`.
- Por ejemplo si `ordpre[m]<ordpre[n]` y `ordpost[m]<ordpost[n]` entonces `m` está a la izquierda de `n`, y así siguiendo.

ZIP Enlace al zip