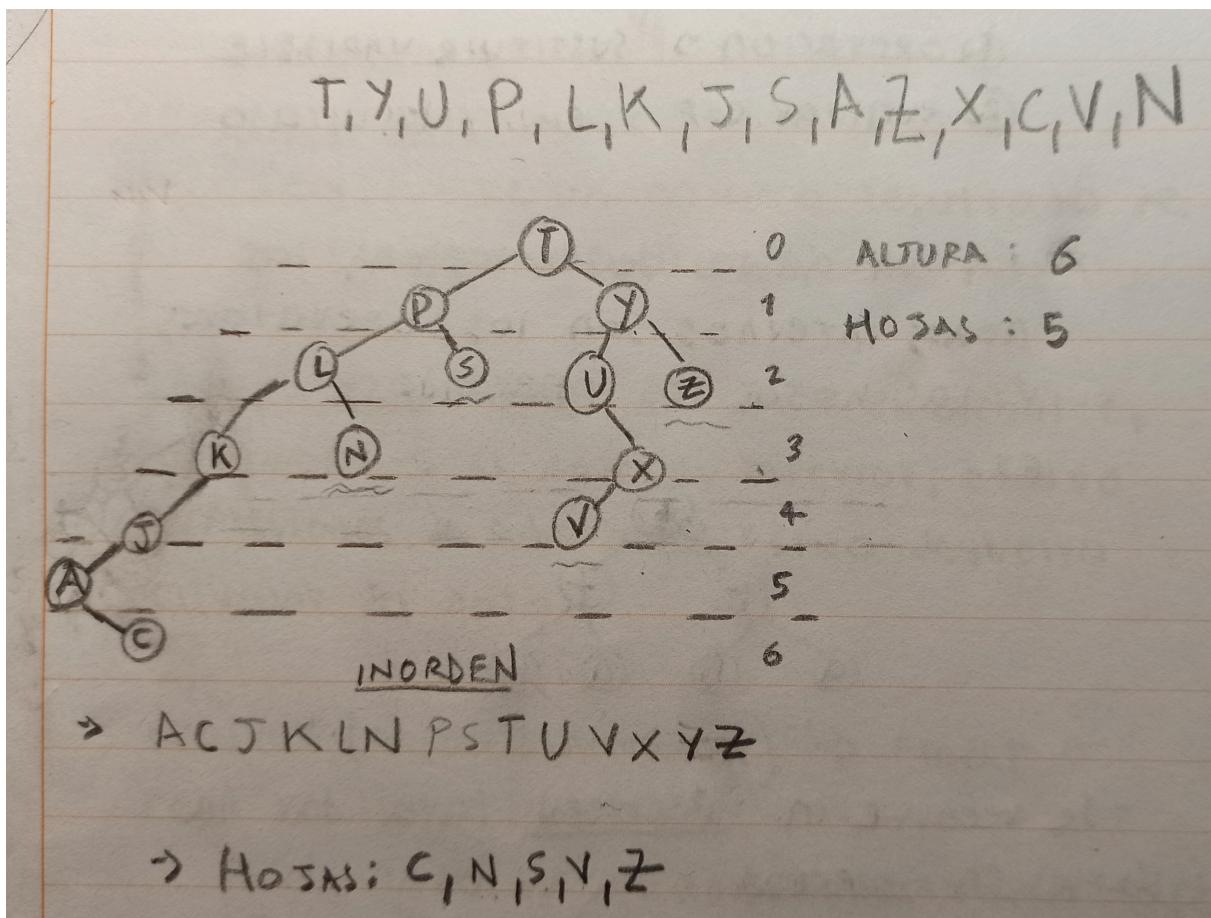


UT4_PD1

Ejercicio #1



¿Cuál de las siguientes afirmaciones es correcta?

a) El árbol tiene altura 7. **FALSO**

b) "Y" es ancestro de "C".

FALSO

c) "X" es descendiente de "P".

FALSO

d) "S" es una hoja.

VERDADERO

En el árbol resultante:

- a) El árbol tiene cinco hojas. **VERDADERO - Se responde en imagen de arriba**
b) "V" es descendiente de "S".

FALSO

- c) "K" es hermano de "J".

FALSO

- d) "A" y "X" están al mismo nivel.

FALSO

Para imprimir las claves en orden lexicográfico basta con

- a) recorrer el árbol en PREORDEN.

FALSO

→ Una bobada, recorriendo **nodo → izquierda → derecha** ocurre que se imprime primero el nodo en cuestión y luego sus hijos. Ya de inicio empieza incrustando la **T** como primero, lo que resulta en una barrabasada de primera línea.

- b) recorrer el árbol en POSTORDEN. **FALSO**

→ Otra bobada más. Recorriendo **izquierda → derecha → nodo** se procesan los dos hijos antes que el padre, por eso, en un principio el recorrido empezaría por la **C**.

- c) recorrer el árbol en INORDEN. **VERDADERO**

→ Primera vez en el día que no escuchó una barrabasada. Proceso primero el hijo izquierdo, luego el nodo en cuestión, y luego el hijo derecho. Así es como queda escrito en la imagen arriba.

- d) debe recorrerse el árbol POR NIVELES. **FALSO**

→ Evidentemente usted tiene una deformidad genética en el hipotalámo, que le impide obrar libre y reflexivamente. En un recorrido por niveles se visitan los nodos nivel por nivel (de arriba a abajo, izquierda a derecha). Claramente no ocurre de ese modo.

Ejercicio #2

MiFunción : devuelve un tipo entero

Comienzo

A ← -1; B ← -1 // O(1)

```

Si HijoIzquierdo <> nulo entonces
    A ← HijoIzquierdo.MiFunción // O(n)
Fin Si
Si HijoDerecho <> nulo entonces
    B ← HijoDerecho.MiFunción // O(n)
Fin Si
Devolver máximo(A,B) +1 // O(1)
Fin

```

- Este algoritmo implementa un recorrido **postorden** para calcular la altura de un árbol binario. → Se sabe que la altura de un árbol binario se define recursivamente como la altura máxima entre sus subárboles izquierdo y derecho.

Si existe un hijo izquierdo, calcula la altura del hijo izquierdo. Si existe un derecho, calcula su altura igualmente.

El caso base está en que, si un nodo es hoja, A y B se mantienen en -1 y la operación `maximo(-1,-1) + 1` devuelve 0, que es la altura de un nodo hoja.

- Tiempo de ejecución de O(n) porque visita cada nodo una vez.

Ejercicio #3

EJERCICIO 3.1

UT4

PD1 - ES #3

NIV

0

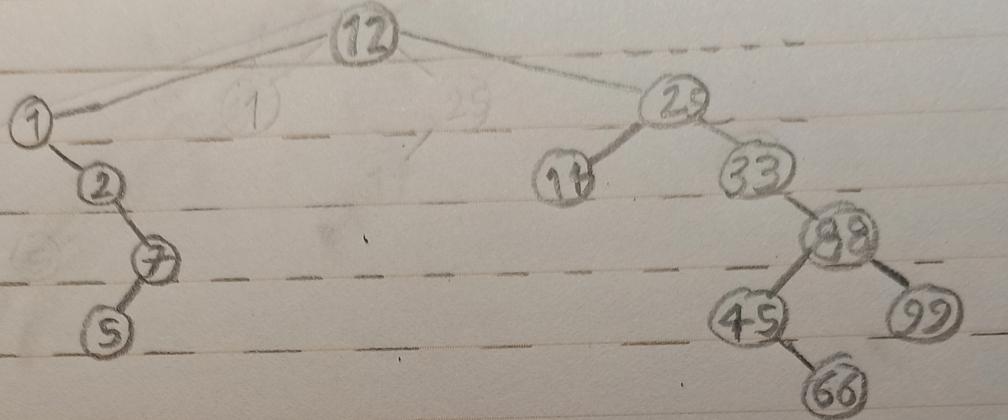
1

2

3

4

5



POSTORDEN

5, 7, 2, 1, 14, 66, 45, 29, 88, 33, 25, 12

PREFORDEN

12, 1, 2, 7, 5, 25, 14, 33, 88, 45, 66, 29

INORDEN

1, 2, 5, 7, 12, 14, 25, 33, 45, 66, 88, 99

EJERCICIO 3.2

Elimino 99 → Simplemente arranco el 99. Imagínese que el nodo 99 se derrite.

```
12
 / \
1  25
 \ / \
2 14 33
 \ / \
7 88 66
```

/ /
5 45

Elimino 2 → Reemplazo por hijo derecho porque es lo único que tiene, es decir, 7.

12
/ \
1 25
\ / \
7 14 33
/ / \
5 88 66
/
45

Elimino 12 → Hay que buscar el **sucesor inorden** (mínimo del subárbol derecho) o **predecesor** (máximo del izquierdo). En este caso tomamos el predecesor 7.

7
/ \
1 25
\ / \
5 14 33
/ \
88 66
/
45

Elimino 33 → Cambio por el **predecesor inorden**, que sería el 45.

7
/ \
1 25
\ / \
5 14 45

/ \
88 66

FINALMENTE

- Preorden → 7, 1, 5, 25, 14, 66, 88, 45
- Postorden → 5, 1, 14, 88, 66, 45, 25, 7
- Inorden → 1, 5, 7, 14, 25, 88, 45, 66