CS2023 - Aula de Ejercicios Nº 6

Brenner H. Ojeda Rios

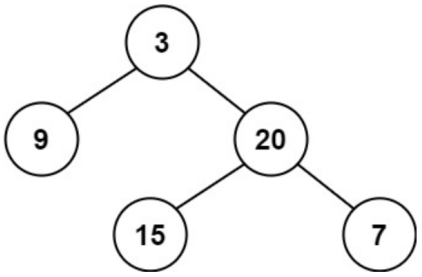
Semestre 2024-1

Se sugiere que cada estudiante trate de resolver los ejercicios de forma **individual** y luego los discuta en grupo. Para la solución de estos ejercicios, **no** debe usar el contenedor **queue** ni **stack** de STL. **Usted, debe crear su propia estructura de datos.**

Ejercicios

1. (5 pts) Dado un árbol binario, determine si tiene altura equilibrada. Un árbol de altura equilibrada es un árbol binario en el que la profundidad de sus subárboles de cada nodo no difiere en mas de uno.

   - **Ejemplo 1:**

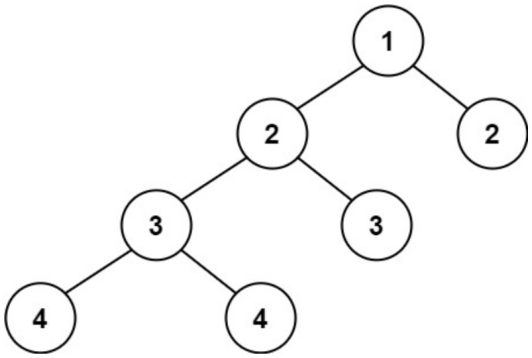
     Input:

     ```
     root = [3,9,20,null,null,15,7]
     ```

     Output:

     ```
     true
     ```

   - **Ejemplo 2:**

     Input:

     ```
     root = root = [1,2,2,3,3,null,null,4,4]
     ```

     Output:

     ```
     false
     ```

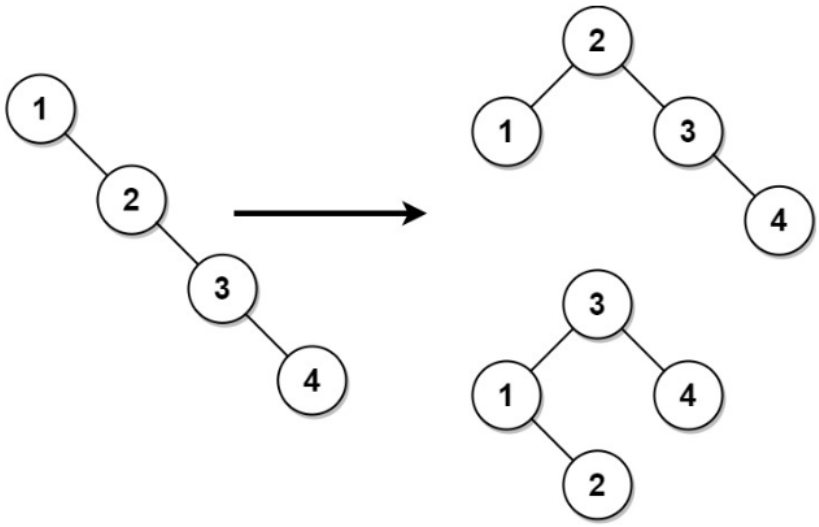
   Restricciones:

   - El número de nodos en el árbol está en el rango [0, 5000].
   - $-10^4 \leq \text{Node.val} \leq 10^4$

2. (6 pts) Dada la raíz de un árbol de búsqueda binario, devuelve un árbol de búsqueda binario balanceado con los mismos valores de nodo. Si hay más de una respuesta, devuelva cualquiera de ellas.

   Un árbol de búsqueda binario está balanceado si la profundidad de los dos subárboles de cada nodo nunca difiere en más de 1.

   - **Ejemplo 1:**



   Input:

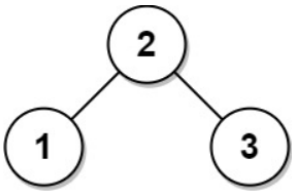
   ```
   root = [1,null,2,null,3,null,4,null,null]
   ```

   Output:

   ```
   [2,1,3,null,null,null,4]
   ```

   Explicación: Esta no es la única respuesta correcta, [3,1,4,null,2] también es correcta.

   - **Ejemplo 2:**



   Input:

   ```
   root = [2,1,3]
   ```

   Output:

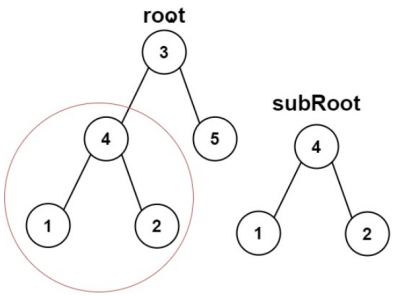
   ```
   [2,1,3]
   ```

Restricciones:

- El número de nodos en el árbol estará en el rango [1, 1000].
- $1 \leq \text{Node.val} \leq 10^5$

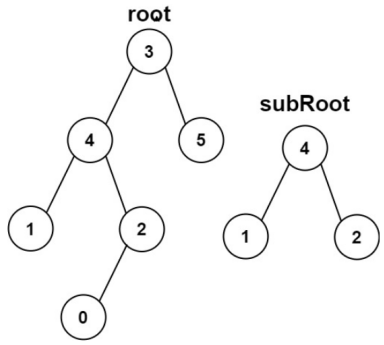
3. (6 pts) Dadas las raíces de dos árboles binarios `root` y `subRoot`, devuelve verdadero si hay un subárbol de `root` con la misma estructura y valores de nodo de `subRoot` y falso en caso contrario. Un subárbol se refiere a una porción de un árbol que, aunque es un árbol por sí mismo, está contenido dentro de otro árbol más grande. El árbol también podría considerarse como un subárbol de sí mismo.

   - **Ejemplo 1:**



     Input: `root = [3,4,5,1,2], subRoot = [4,1,2]`
     Output: true

   - **Ejemplo 2:**



     Input: `root = [3,4,5,1,2,null,null,null,null,0], subRoot = [4,1,2]`
     Output: false

Restricciones:

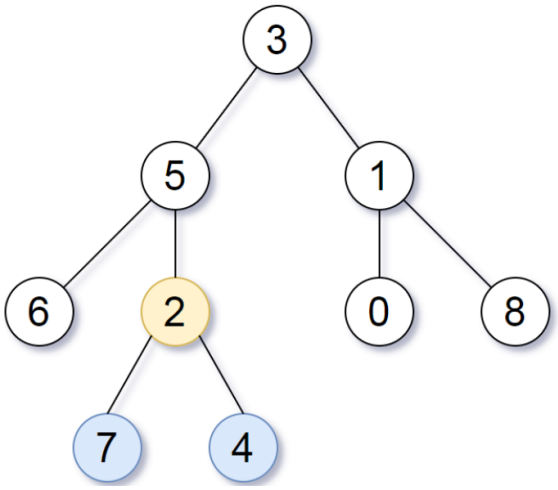
- La cantidad de nodos en el árbol raíz está en el rango de [1, 2000].
- La cantidad de nodos en el subárbol raíz está en el rango de [1, 1000].
- $-10^4 \leq root.val \leq 10^4$.
- $-10^4 \leq subRoot.val \leq 10^4$

4. (7 pts) Dada la raíz de un árbol binario, devuelve el ancestro común más bajo de sus hojas más profundas.

   Recordar que:

   - El nodo de un árbol binario es una hoja si y sólo si no tiene hijos.
   - La profundidad de la raíz del árbol es 0. Si la profundidad de un nodo es d, la profundidad de cada uno de sus hijos es $d + 1$.
   - El ancestro común más bajo de un conjunto S de nodos es el nodo A con la mayor profundidad, de modo que cada nodo en S está en el subárbol con raíz A.

   Ejemplo:



   Input:

```
root = [3,5,1,6,2,0,8,null,null,7,4]
```

   Output:

```
[2,7,4]
```

   Explicación: Devolvemos el nodo con valor 2, coloreado en amarillo en el diagrama.

   Los nodos coloreados en azul son los nodos de las hojas más profundos del árbol. Tenga en cuenta que los nodos 6, 0 y 8 también son nodos hoja, pero su profundidad es 2, pero la profundidad de los nodos 7 y 4 es 3.

   Restricciones:

   - El número de nodos en el árbol estará en el rango [1, 1000].
   - $0 \leq$ Nodo.val ≤ 1000
   - Los valores de los nodos del árbol son únicos.

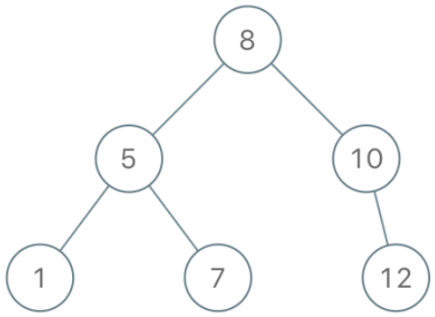
5. (7 pts) Dado un arreglo de enteros `preorder`, que representa el recorrido en preorden de un Árbol de Búsqueda Binaria (BST), construye el árbol y devuelve su raíz.

   Se garantiza que siempre es posible encontrar un árbol de búsqueda binaria con los requisitos dados para los casos de prueba dados.

   Un árbol de búsqueda binaria es un árbol binario donde para cada nodo, cualquier descendiente de `Node.left` tiene un valor estrictamente menor que `Node.val`, y cualquier descendiente de `Node.right` tiene un valor estrictamente mayor que `Node.val`.

   Un recorrido en preorden de un árbol binario muestra primero el valor del nodo, luego recorre `Node.left` y luego recorre `Node.right`.

   - **Ejemplo 1:**
     Input: `preorder = [8,5,1,7,10,12]`
     Output: `[8,5,10,1,7,null,12]`



   - **Ejemplo 2:**
     Input: `preorder = [1,3]`
     Output: `[1,null,3]`

   Restricciones:

   - $1 \leq$ preorder.length ≤ 100.
   - $1 \leq$ preorder[i] ≤ 1000.
   - Todos los valores de preorder son únicos.