CS2023 - Aula de Ejercicios Nº 5

Brenner H. Ojeda Rios

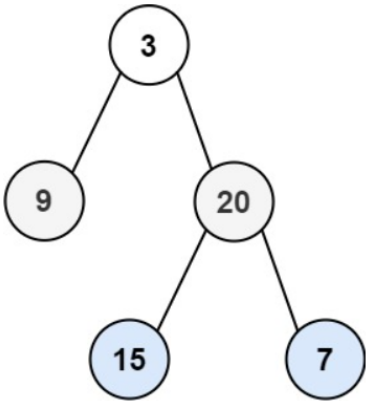
Semestre 2024-0

Se sugiere que cada estudiante trate de resolver los ejercicios de forma **individual** y luego los discuta en grupo.

Ejercicios

1. (5 pts) Dada la raíz de un árbol binario, devuelve el recorrido de orden de nivel de los valores de sus nodos. (es decir, de izquierda a derecha, nivel por nivel).

   Ejemplo:

   Input:

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

   Output:

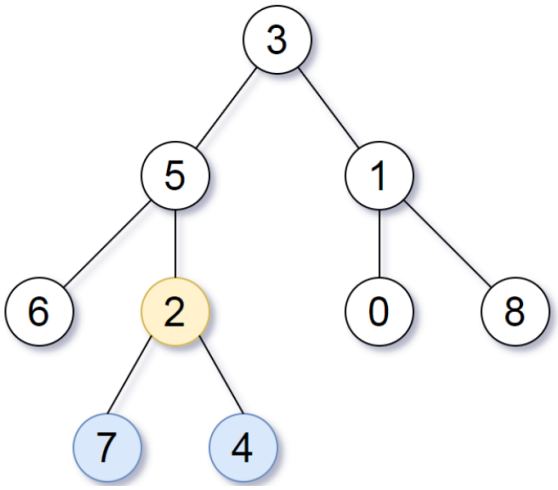
   ```
   [[3],[9,20],[15,7]]
   ```

2. (6 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.

3. (9 pts) Se le proporcionan n nodos raíz BST (árbol de búsqueda binaria) para n BST separados almacenados en una matriz de árboles (indexados 0). Cada BST en los árboles tiene como máximo 3 nodos y no hay dos raíces que tengan el mismo valor. En una sola operación podrás:

   - Seleccione dos índices distintos i y j de modo que el valor almacenado en una de las hojas de los `trees[i]` sea igual al valor de la raíz de los `trees[j]`.
   - Reemplace el nodo hoja en `trees[i]` con `trees[j]`.
   - Retire los `trees[j]` de los árboles.

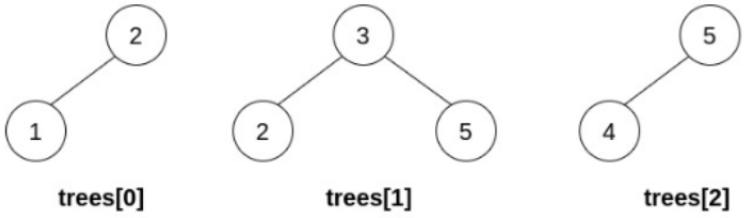
   Devuelve la raíz del BST resultante si es posible formar un BST válido después de realizar $n - 1$ operaciones, o nulo si es imposible crear un BST válido.

   Un BST (árbol de búsqueda binaria) es un árbol binario donde cada nodo satisface la siguiente propiedad:

   - Cada nodo en el subárbol izquierdo del nodo tiene un valor estrictamente menor que el valor del nodo.
   - Cada nodo en el subárbol derecho del nodo tiene un valor estrictamente mayor que el valor del nodo.

   Una hoja es un nodo que no tiene hijos.

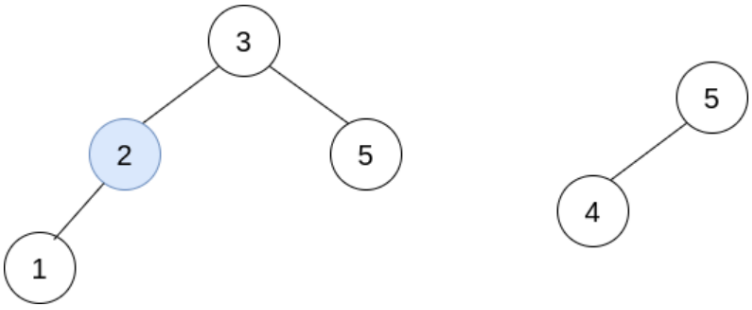
   Ejemplo



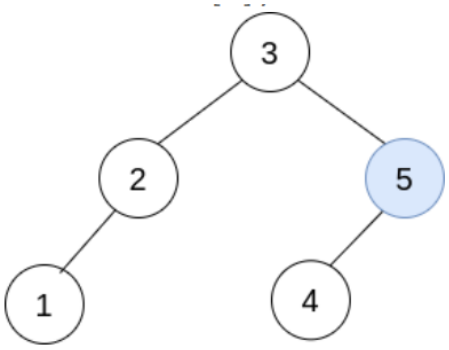
   Input: `trees = [[2,1],[3,2,5],[5,4]]`

   Output: `[3,2,5,1,null,4]`

   Explicación: En la primera operación, elija $i = 1$ y $j = 0$ y combine los `árboles[0]` en los `árboles[1]`. Elimine `árboles[0]`, por lo que `árboles = [[3,2,5,1], [5,4]]`.



   En la segunda operación, elija $i = 0$ y $j = 1$ y combine los `árboles[1]` en los `árboles[0]`. Elimine `árboles[1]`, por lo que `árboles = [[3,2,5,1, null,4]]`.



Restricciones:

- n = trees.length
- $1 \leq n \leq 5 * 10^4$
- El número de nodos en cada árbol está en el rango `[1, 3]`.
- Cada nodo en la entrada puede tener hijos pero no nietos.
- No hay dos raíces de árboles que tengan el mismo valor.
- Todos los árboles en la entrada son BST válidos.
- $1 \leq$ TreeNode.val $\leq 5 * 10^4$.