CS2023 - Aula de Ejercicios Nº 6

Angel Napa
ACL: Joaquín Jordán
Semestre 2024-1

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

IMPORTANTE:

- Enviar en canvas un **único archivo .cpp** (colocar las 3 soluciones en el mismo archivo .cpp).
- **No se revisaran archivos .zip** subidos a canvas.
- **Colocar nombres y apellidos** de los integrantes **al inicio** del archivo .cpp.
- La evaluación continua es grupal (mínimo 2 alumnos y máximo 3 alumnos).

Ejercicios

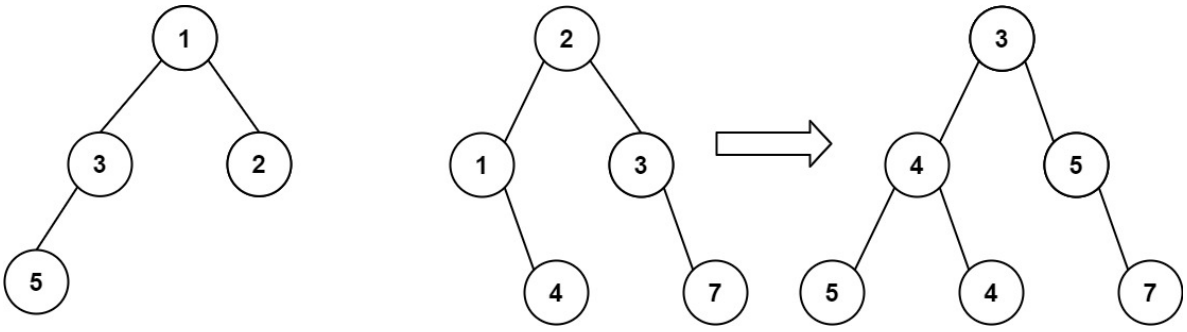
1. (6 pts) Te dan dos árboles binarios root1 y root2.

   Imagina que cuando pones uno de ellos para cubrir el otro, algunos nodos de los dos árboles se superponen mientras que los demás no. Debes fusionar los dos árboles en un nuevo árbol binario. La regla de fusión es que si dos nodos se superponen, se suman los valores de los nodos como el nuevo valor del nodo fusionado. De lo contrario, el nodo NOT null se utilizará como nodo del nuevo árbol.

   Devuelve el árbol fusionado .

   Nota: El proceso de fusión debe comenzar desde los nodos raíz de ambos árboles.

   - **Ejemplo 1:**



   Input:

   ```
   root1 = [1,3,2,5],
   root2 = [2,1,3,null,4,null,7]
   ```

   Output:

   ```
   [3,4,5,5,4,null,7]
   ```

   - **Ejemplo 2:**

   Input:

   ```
   root1 = [1], root2 = [1,2]
   ```

Output:

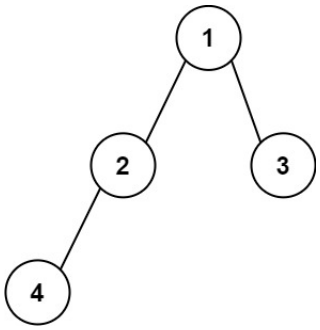
```
[2,2]
```

2. (7 pts) Dado el rootnodo de un árbol binario, su tarea es crear una representación de cadena del árbol siguiendo un conjunto específico de reglas de formato. La representación debe basarse en un recorrido previo al pedido del árbol binario y debe cumplir con las siguientes pautas:

   - Representación de nodos : cada nodo del árbol debe estar representado por su valor entero.
   - Paréntesis para hijos : si un nodo tiene al menos un hijo (ya sea izquierdo o derecho), sus hijos deben representarse entre paréntesis. Específicamente:
     - Si un nodo tiene un hijo izquierdo, el valor del hijo izquierdo debe estar entre paréntesis inmediatamente después del valor del nodo.
     - Si un nodo tiene un hijo correcto, el valor del hijo correcto también debe estar entre paréntesis. Los paréntesis del hijo derecho deben seguir a los del hijo izquierdo.
   - Omitir paréntesis vacíos : cualquier par de paréntesis vacíos (es decir, ()) debe omitirse de la representación de cadena final del árbol, con una excepción específica: cuando un nodo tiene un hijo derecho pero no un hijo izquierdo. En tales casos, debe incluir un par de paréntesis vacíos para indicar la ausencia del niño izquierdo. Esto garantiza que se mantenga la asignación uno a uno entre la representación de cadena y la estructura de árbol binario original.

   En resumen, los pares de paréntesis vacíos deben omitirse cuando a un nodo solo le queda un hijo o ningún hijo. Sin embargo, cuando un nodo tiene un hijo derecho pero ningún hijo izquierdo, un par de paréntesis vacíos debe preceder a la representación del hijo derecho para reflejar la estructura del árbol con precisión.

   - **Ejemplo 1:**



Input:

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

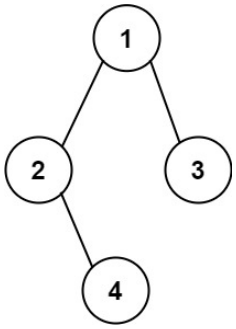
Output:

```
"1(2(4))(3)"
```

Explicación:

```
Originally, it needs to be "1(2(4())(3()())",
but you need to omit all the empty parenthesis pairs.
And it will be "1(2(4))(3)".
```

- **Ejemplo 2:**



Input:

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

Output:

```
"1(2()(4))(3)"
```

Explicación:

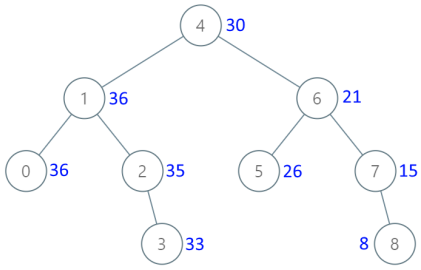
```
Almost the same as the first example,
except the () after 2 is necessary to indicate
the absence of a left child for 2 and the
presence of a right child.
```

3. (7 pts) Dado el root de un árbol de búsqueda binaria (BST), conviértalo en un árbol mayor de modo que cada clave del BST original se cambie a la clave original más la suma de todas las claves mayores que la clave original en BST.

   Como recordatorio, un árbol de búsqueda binario es un árbol que satisface estas restricciones:

   - El subárbol izquierdo de un nodo contiene solo nodos con claves menores que la clave del nodo.
   - El subárbol derecho de un nodo contiene solo nodos con claves mayores que la clave del nodo.
   - Tanto el subárbol izquierdo como el derecho también deben ser árboles de búsqueda binarios.

   - **Ejemplo 1:**



Input:

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

Output:

```
[30,36,21,36,35,26,15,null,null,null,
33,null,null,null,8]
```

- **Ejemplo 2:**
  Input:

```
root = [0,null,1]
```

Output:

```
[1,null,1]
```