Estrategias de Programación y Estructuras de Datos
Tema 6: Árboles
Ejercicios propuestos

1. Calcular el coste de todas las operaciones públicas de `Tree<E>`, `GTree<E>` y `Btree<E>`.

2. Programar un método:

   `getMax()`

   dentro de la clase `GTree` que devuelva el valor máximo de un árbol general. Consideremos los siguientes supuestos:

   a) Los nodos del árbol no están ordenados según su valor.
   b) El valor de la raíz es, siempre, menor que el de todos sus hijos.

   Calcule el coste asintótico temporal en el caso peor de este método.

3. Programar un método:

   `getMax()`

   dentro de la clase `BTree` que devuelva el valor máximo de un árbol general. Consideremos los siguientes supuestos:

   a) Los nodos del árbol no están ordenados según su valor.
   b) El valor de la raíz es, siempre, menor que el de todos sus hijos.
   c) El valor de la raíz es, siempre, menor que el de todos los nodos contenidos en uno de sus hijos y mayor que el de todos los nodos contenidos en el otro.

4. Debido a la estructura de datos utilizada, no es posible mantener los atributos de tamaño de los árboles (`size`) de manera que podamos utilizar el método `size()` de `Collection` para devolver el tamaño de un árbol (tanto general como binario).

   En este ejercicio proponemos una solución para este problema realizando los siguientes pasos:

   a) Añada un atributo `parent` que apunte al padre del árbol. ¿En qué clase lo añadiría y con qué tipo?

b) Identifique todos los métodos de las clases `GTree` y `BTree` que se vean afectados por la inclusión de este atributo, de manera que siempre se mantenga actualizado.
c) Realice las modificaciones oportunas en estos métodos y compare su coste asintótico temporal en el caso peor con el calculado en el ejercicio 1.

5. Al igual que en el ejercicio anterior, es posible enriquecer la Estructura de Datos con atributos que almacenen la altura del árbol o su fan-out.

   a) Añada un atributo `height` que almacene la altura del árbol. ¿En qué clase lo añadiría? Realice los apartados b y c del ejercicio anterior sobre este atributo, de forma que el método `getHeight()` consista, simplemente, en devolver el valor de ese atributo.
   b) Añada un atributo `fanout` que almacene el fanout del árbol. ¿En qué clase lo añadiría? Realice los apartados b y c del ejercicio anterior sobre este atributo, de forma que el método `getFanOut()` consista, simplemente, en devolver el valor de ese atributo.

6. Dado que los árboles binarios sólo tienen dos hijos y los árboles generales pueden tener cualquier número de hijos, es posible implementar los árboles binarios utilizando un árbol general como estructura de datos.

   Implemente una clase `BTreeGT` que extienda `Collection` e implemente el interfaz `BTreeIF` empleando un árbol general como estructura de datos en lugar de los atributos añadidos por las clases `Tree` y `BTree`. Utilice los métodos de `GTree` para programar los de `BTree`.

7. Aunque en un principio puede resultar chocante, es posible representar un árbol general empleando un árbol binario como estructura de datos. Para ello hay que realizar la siguiente transformación:

   - El hijo izquierdo de un árbol apunta a su primer hijo.
   - El hijo derecho de un árbol apunta a su siguiente hermano.

   Implemente una clase `GTreeBT` que extienda `Collection` e implemente el interfaz `GTreeIF` empleando un árbol binario como estructura de datos en lugar de los atributos añadidos por las clases `Tree` y `GTree`. Utilice los métodos de `BTree` para programar los de `Gtree`.

   A continuación se puede ver una imagen con un ejemplo de representación de un árbol general mediante un árbol binario. Los enlaces en negro representan una relación padre-hijo, mientras que los enlaces en azul representan una relación entre hermanos.

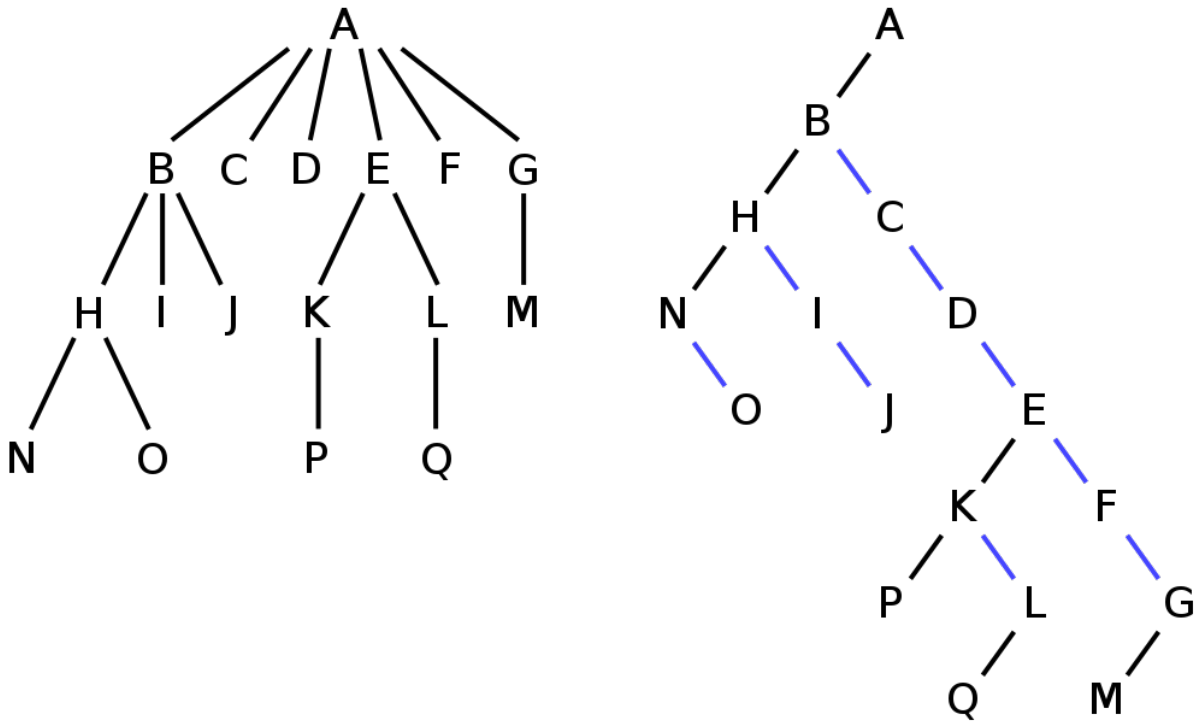

Imagen 1: Imagen extraída de Wikipedia. De CyHawk - Trabajo propio, Dominio público, https://commons.wikimedia.org/w/index.php?curid=4657190