Programación para la Inteligencia Artificial

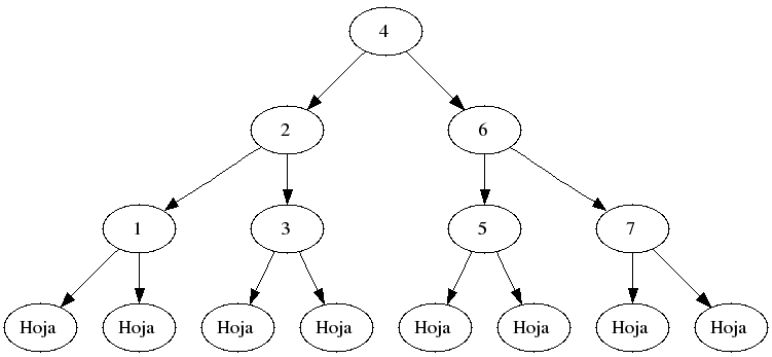
Haskell - Práctica 6 (Parte I)
Árboles y conjuntos

1. Árboles binarios de búsqueda

Consideramos la siguiente definición de **árbol ordenado de búsqueda**, considerando que el resultado obtenido por la función `aplanar` en un árbol de este tipo consiste en una lista ordenada en orden creciente:

```
data ArbolB a = Hoja
    | Rama (ArbolB a) a (ArbolB a)
    deriving Show
```

- Definir el árbol correspondiente a la siguiente figura:



- Definir la función `tama` que devuelva el número de nodos internos.
- Definir la función `aplanar` de manera que `aplanar ab` sea la lista obtenida aplanando el árbol `ab`.
- Definir la función `pertenece` de manera que `pertenece e ab` se verifica si el elemento `e` pertenece al árbol de búsqueda `ab`.

- Definir la función `insertar` de manera que `insertar e ab` añade el elemento `e` al árbol de búsqueda `ab`.
- Definir la función `borrar` de manera que `borrar e ab` elimina el elemento `e` del árbol de búsqueda `ab`.
- Definir la función `crearArbolL` de manera que `crearArbolL l` crea un árbol binario de búsqueda a partir de una lista `l` no necesariamente ordenada.
- Definir la función `ordenarConArbol` que tome como entrada una lista `l`, de manera que `ordenarConArbol l` devuelve una nueva lista como resultado de ordenar `l` mediante un árbol de búsqueda.

2. **Conjuntos con listas**

   Supongamos que estamos usando listas para representar conjuntos.

   - Tomando como entrada dos listas, definir la función `subconjunto` tal que `subconjunto xs ys` se verifica si `xs` es subconjunto de `ys`.
   - Definir la función `igualesC` tal que `igualesC l1 l2` se verifica si las listas `l1` y `l2`, vistas como conjuntos, son iguales.