Árboles zurdos

Estructuras de Datos
Facultad de Informática - UCM

Decimos que un árbol binario es *zurdo* si es vacío, o es una hoja, o es un nodo interno en el que más de la mitad de sus descendientes están en el hijo izquierdo y, además, ambos hijos son zurdos.

Partimos de la siguiente definición de la interfaz de la clase `BinTree<T>`:

```
template <class T> class BinTree {
public:
  // ...
  bool empty() const;
  const T &root() const;
  BinTree left() const;
  BinTree right() const;
private:
  // ...
};
```

Implementa una función `es_zurdo` externa a la clase,

```
template<typename T>
bool es_zurdo(const BinTree<T> &t);
```

que devuelva `true` si el árbol binario pasado como parámetro es zurdo, o `false` en caso contrario.

Puedes definir las funciones auxiliares que necesites.

Solución

Decir que un nodo tiene más de la mitad de sus descendientes en el hijo izquierdo equivale a decir que el hijo izquierdo tiene más nodos que el hijo derecho.

Definimos una función auxiliar recursiva `es_zurdo_aux` que devuelva un valor booleano y un valor entero.

- El valor booleano indica si el árbol es zurdo o no.
- El valor entero indica el número de nodos del árbol. Necesitamos este valor para saber si el hijo izquierdo de un árbol tiene más nodos que el derecho.

```
template <typename T>
pair<bool, int> es_zurdo_aux(const BinTree<T> &t) {
  if (t.empty()) {
    // El arbol vacio es zurdo por definicion, y ademas tiene 0 nodos.
    return {true, 0};
  } else if (t.left().empty() && t.right().empty()){
    // Una hoja es zurda por definicion, y ademas tiene 1 nodo.
    return {true, 1};
  } else {
    // Llamamos recursivamente a la funcion sobre el hijo izquierdo. Nos
    // devuelve dos resultados:
    //   - es_zurdo_iz : Vale true si el hijo izquierdo es zurdo
    //   - num_nodos_iz : Numero de nodos en el hijo izquierdo
    auto [es_zurdo_iz, num_nodos_iz] = es_zurdo_aux(t.left());

    // Llamamos recursivamente a la funcion sobre el hijo derecho. Nos
    // devuelve dos resultados:
    //   - es_zurdo_dr : Vale true si el hijo derecho es zurdo
    //   - num_nodos_dr : Numero de nodos en el hijo derecho
    auto [es_zurdo_dr, num_nodos_dr] = es_zurdo_aux(t.right());

    // A partir de los resultados de las llamadas recursivas, determinamos
    // si el arbol es zurdo, y calculamos su numero de nodos.
    bool es_zurdo = es_zurdo_iz && es_zurdo_dr && num_nodos_iz > num_nodos_dr;
    int num_nodos = 1 + num_nodos_iz + num_nodos_dr;

    // Devolvemos ambas cosas
    return {es_zurdo, num_nodos};
  }
}
```

La función pedida se implementa haciendo la *llamada inicial* a la función recursiva definida anteriormente, y obteniendo el resultado que nos interesa, que es el valor booleano que indica si el árbol es zurdo o no.

```
template <typename T>
bool es_zurdo(const BinTree<T> &t) {
  return es_zurdo_aux(t).first;
}
```

El coste de `es_zurdo_aux` viene dado por la siguiente recurrencia, donde n denota el número de nodos:

$$T(n) = \begin{cases} k_1 & \text{si } n \leq 1 \\ T(n_i) + T(n_d) + k_2 & \text{si } n > 2 \end{cases}$$

En la última ecuación, n_i y n_d denotan el número de nodos de los hijos izquierdo y derecho, respectivamente. Se cumple que $n = n_i + n_d + 1$. Esta recurrencia, independientemente de los valores de n_i y n_d correspondientes a cada nodo, cumple que $T(n) \in O(n)$. Por tanto, el coste de `es_zurdo_aux` es lineal con respecto al número de nodos del árbol pasado como parámetro.