Práctica 2 – Estructuras de datos y algoritmos

Ejercicio 1

En clase hemos visto los Árboles (Binarios, BST, red-black). Investigar si se pueden utilizar directamente a través de las STL. Justificar en caso contrario.

Tras haber estado investigando, la respuesta es **no**. No se pueden utilizar árboles en C++ directamente a través de las STL. ¿Por qué no se puede? Muy sencillo, piensa en los vectores y las listas: ¿qué tienen en común? Pueden verse directamente como "secuencias". Y las secuencias (entre un `begin()` y un `end()`) son todo lo que los algoritmos de la STL tratan.

Un árbol, si no se utiliza para un propósito específico, no es una estructura monodimensional, por lo que no encaja en la lógica de los algoritmos secuenciales. Pero, hasta que no se define un propósito, el número de formas en que se puede ver y manipular es tan grande que escribir una estructura eficiente se convierte en una tarea difícil, ya que lo que es eficiente para un propósito no lo es para otro.

Dicho esto, los árboles están dentro de la biblioteca estándar como implementación interna de `std::map` y `std::set`, pero son árboles muy particulares. Son árboles binarios, utilizados para implementar secuencias rápidas auto-ordenadas por lo que no sólo mantienen los valores, sino también alguna información parcial de ordenación que los hace eficientes para su propio propósito, es decir ordenar.

De todos modos, se puede obtener una estructura de árbol genérica mediante la reutilización de un contenedor en un nodo de datos, como:

```
1. template<class T, template<class...> class Container>
2. strut tree { T value; Container<tree> children; };
```

Así puedes tener `tree<int,std::list>` o `tree<double,std::vector>` y similares.

Por supuesto, se tienen que desarrollar los iteradores para los distintos tipos de *caminos* y definir algunos métodos estándar para que el algoritmo estándar pueda trabajar con la estructura.

Pero hacer esto, significa definir un *orden* para los elementos, por lo que nuestro árbol se convertirá en una estructura de datos para la auto-ordenación. Y eso no es más que `std::set` o `std::map`.