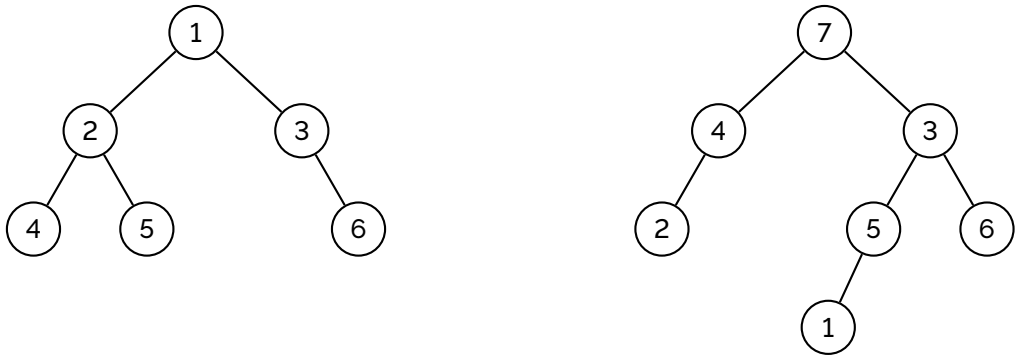
Árboles equilibrados estables

Decimos que un árbol binario equilibrado es *estable* si, al quitar cualquiera de sus hojas, el árbol mantiene su equilibrio independientemente de la hoja que hayamos quitado. Por ejemplo:



En esta figura ambos árboles son equilibrados. El de la izquierda es estable, ya que si quitamos cualquiera de las hojas (4, 5 o 6), el árbol sigue estando equilibrado. Por el contrario, el árbol de la derecha no es estable, ya que al quitar la hoja que contiene el 2 (por ejemplo), el árbol deja de estar equilibrado.

Por convenio, consideramos que el árbol vacío es estable.

Se pide:

1. Implementar una función `es_estable` con la siguiente cabecera:

   ```
   bool es_estable(const BinTree<T> &arbol);
   ```

   Esta función recibe como parámetro un árbol binario equilibrado y debe devolver `true` si el árbol es estable, o `false` en caso contrario. Puedes crear las funciones auxiliares que sean necesarias.

2. Indica el coste de la función anterior, en función del número de nodos del árbol de entrada.

Entrada

La entrada comienza con un número que indica el número de casos de prueba que vienen a continuación. Cada caso de prueba consiste en una línea con la descripción de un árbol binario mediante la notación vista en clase. El árbol vacío se representa mediante `.` y el árbol no vacío mediante `(iz x dr)`, siendo `x` la raíz, e `iz` y `dr` las representaciones de ambos hijos. Puedes suponer que todos los árboles son equilibrados.

Salida

Para cada caso de prueba debe imprimirse `SI` (sin tilde) si el árbol es estable, o `NO` en caso contrario.

Entrada de ejemplo

```
6
(((. 4 .) 2 (. 5 .)) 1 (. 3 (. 6 .)))
((((. 4 .) 3 .) 7 (. 9 (. 8 .))) 1 ((. 27 .) 14 (. 3 .)))
(((. 2 .) 4 .) 7 (((. 1 .) 5 .) 3 (. 6 .)))
(((. 6 .) 4 (. 7 .)) 3 (. 5 .))
(. 4 .)
.
```

Salida de ejemplo

```
SI
SI
NO
NO
SI
SI
```

Autor

Manuel Montenegro