```
NECESARIO "bintree_eda.h"

#include <iostream>
#include <fstream>
#include <string>
#include <sstream>
#include "bintree_eda.h"

using namespace std;

struct tSol{
    bool esB;
    int min, max;
};

template <typename T>
tSol esB(bintree<T> const& arbol) {
    tSol res;


    if (arbol.empty()) {
        res.esB = true;
    }
    else {
        tSol iz = esB(arbol.left());
        tSol dr = esB(arbol.right());


        if (arbol.left().empty()) {
            res.min = arbol.root();
        }
        else {
            res.min = iz.min;
        }

        if (arbol.right().empty()) {
            res.max = arbol.root();
        }
        else {
            res.max = dr.max;
        }


        bool ordenado = (arbol.left().empty() || iz.max < arbol.root()) &&
            (arbol.right().empty() || dr.min > arbol.root());


        res.esB = iz.esB & dr.esB && ordenado;
    }
    return res;
}


void resuelveCaso() {
```

```
    bintree<int> arbol = leerArbol(-1);

    tSol sol = esB(arbol);

    if (sol.esB) {
        cout << "SI" << endl;
    }
    else {
        cout << "NO" << endl;
    }
}


int main() {
    // Para la entrada por fichero.
    // Comentar para acepta el reto
#ifndef DOMJUDGE
    ifstream in("datos.txt");
    auto cinbuf = std::cin.rdbuf(in.rdbuf()); //save old buf and redirect std::cin to
casos.txt
#endif


    int casos;

    cin >> casos;
    for (int i = 0; i < casos; i++) resuelveCaso();


    // Para restablecer entrada. Comentar para acepta el reto
#ifndef DOMJUDGE // para dejar todo como estaba al principio
    std::cin.rdbuf(cinbuf);
    system("PAUSE");
#endif

    return 0;
}
```