

Examen Arboles 01/08/22

Construye una función que, dado un Árbol Binario, devuelva true si es un AVL y false en el caso contrario.

\*Nota: definir tipos de datos y prototipos de las operaciones de los TADs usados.

## Solución:

```

1  #include "abin.hpp"
2  #include "abin_E-S.h"
3  #include <iostream>
4
5  using namespace std;
6
7  typedef int tElto;
8  const tElto fin = -1;
9  bool EsAbbRec(Abin<tElto>& A, Abin<tElto>::nodo n);
10 bool EsAvlRec(Abin<tElto>& A, Abin<tElto>::nodo n);
11 int altura(Abin<tElto>& A, Abin<tElto>::nodo n);
12
13 //-----|| Aqui comienza el examen ||-----
14
15 /*
16  | EsAvl -> Recibe un arbol binario, y devuelve true si es un Arbol Binario de Busqueda Equilibrado (AVL) o false si no
17  */
18 bool EsAvl(Abin<tElto>& A)
19 {
20     bool flag = EsAbbRec(A, A.raiz()); //un AVL debe ser un ABB
21     flag = flag && EsAvlRec(A, A.raiz()); //y cumplir la condicion de estar equilibrado :)
22     return flag;
23 }
24
25 /*
26  | EsAbbRec -> Recibe un arbol binario y un nodo de este, y devuelve true si es un Arbol Binario de Busqueda (ABB) o false si no (Version auxiliar recursiva)
27  */
28 bool EsAbbRec(Abin<tElto>& A, Abin<tElto>::nodo n)
29 {
30     if(A.hijoIzqdo(n) != Abin<tElto>::NODO_NULO && A.hijoDrcho(n) != Abin<tElto>::NODO_NULO)
31     {
32         return A.elemento(n) > A.elemento(A.hijoIzqdo(n)) && A.elemento(n) < A.elemento(A.hijoDrcho(n)) && EsAbbRec(A, A.hijoIzqdo(n)) && EsAbbRec(A, A.hijoDrcho(n));
33     }
34
35     if(A.hijoIzqdo(n) != Abin<tElto>::NODO_NULO && A.hijoDrcho(n) == Abin<tElto>::NODO_NULO)
36     {
37         return A.elemento(n) > A.elemento(A.hijoIzqdo(n)) && EsAbbRec(A, A.hijoIzqdo(n));
38     }
39
40     if(A.hijoIzqdo(n) == Abin<tElto>::NODO_NULO && A.hijoDrcho(n) != Abin<tElto>::NODO_NULO)
41     {
42         return A.elemento(n) < A.elemento(A.hijoDrcho(n)) && EsAbbRec(A, A.hijoDrcho(n));
43     }
44
45     if(A.hijoIzqdo(n) == Abin<tElto>::NODO_NULO && A.hijoDrcho(n) == Abin<tElto>::NODO_NULO)
46     {
47         return true;
48     }
49 }
50
51 /*
52  | EsAvlRec -> Recibe un arbol binario y un nodo de este, y devuelve true si es un Arbol Binario de Busqueda Equilibrado (AVL) o false si no (version recursiva)
53  */
54 bool EsAvlRec(Abin<tElto>& A, Abin<tElto>::nodo n)
55 {
56     if(A.hijoIzqdo(n) != Abin<tElto>::NODO_NULO && A.hijoDrcho(n) != Abin<tElto>::NODO_NULO)
57     {
58         return abs(altura(A, A.hijoDrcho(n)) - altura(A, A.hijoIzqdo(n))) <= 1 && EsAvlRec(A, A.hijoIzqdo(n)) && EsAvlRec(A, A.hijoDrcho(n));
59     }
60
61     if(A.hijoIzqdo(n) != Abin<tElto>::NODO_NULO && A.hijoDrcho(n) == Abin<tElto>::NODO_NULO)
62     {
63         return abs(altura(A, A.hijoDrcho(n)) - altura(A, A.hijoIzqdo(n))) <= 1 && EsAvlRec(A, A.hijoIzqdo(n));
64     }
65
66     if(A.hijoIzqdo(n) == Abin<tElto>::NODO_NULO && A.hijoDrcho(n) != Abin<tElto>::NODO_NULO)
67     {
68         return abs(altura(A, A.hijoDrcho(n)) - altura(A, A.hijoIzqdo(n))) <= 1 && EsAvlRec(A, A.hijoDrcho(n));
69     }
70
71     if(A.hijoIzqdo(n) == Abin<tElto>::NODO_NULO && A.hijoDrcho(n) == Abin<tElto>::NODO_NULO)
72     {
73         return true;
74     }
75 }
76
77 /*
78  | altura -> Recibe un arbol binario y un nodo de este, y devuelve la altura de dicho nodo en el arbol
79  */
80 int altura(Abin<tElto>& A, Abin<tElto>::nodo n)
81 {
82     if(n == Abin<tElto>::NODO_NULO)
83     {
84         return 0;
85     }
86     else
87     {
88         return 1 + max(altura(A, A.hijoIzqdo(n)), altura(A, A.hijoDrcho(n)));
89     }
90 }
91
92

```

```
93  /*
94  DEFINICIONES:
95  - A.raiz() -> Devuelve el nodo raiz del arbol (en caso de no tener devolveria NODO_NULO)
96  - A.hijoIzqdo(n) -> Devuelve el nodo hijo izquierdo del nodo n del arbol (en caso de no tener devolveria NODO_NULO)
97  - A.hijoDrcho(n) -> Devuelve el nodo hijo derecho del nodo n del arbol (en caso de no tener devolveria NODO_NULO)
98  - A.elemento(n) -> Devuelve el elemento que posee el nodo n del arbol (Debe ser un nodo del arbol)
99  */
100
101  //-----|| Aquí termina el examen ||-----
102
103  //Main con el objetivo de probar el programa y comprobar su funcionamiento
104  int main()
105  {
106      Abin<Elto> A;
107
108      cout << "*** Lectura del arbol binario A ***\n";
109
110      rellenarAbin(A, fin); // Desde std::cin
111
112      bool flag = EsAvl(A);
113
114      if(flag)
115      {
116          cout << "El arbol es AVL" << endl;
117      }
118      else
119      {
120          cout << "El arbol no es AVL" << endl;
121      }
122
123      return 0;
124  }
```