
Mètode de la classe `BinaryTree` que calcula el diàmetre X23305_ca

El **diàmetre** d'un arbre es defineix com el camí més llarg entre dos nodes fulla qualsevols.

Implementa un nou mètode de la classe `BinaryTree` que torna el diàmetre de l'arbre binari.

D'entre els fitxers que s'adjunten en aquest exercici, trobaràs `BinaryTree.old.hpp`, a on hi ha una implementació de la classe genèrica `BinaryTree`. En primer lloc, hauràs de fer:

```
cp BinaryTree.old.hpp BinaryTree.hpp
```

A continuació si obres el fitxer `BinaryTree.hpp` al final del mateix trobaràs el mètode que has d'implementar:

```
// Pre: cert
// Post: Torna el diàmetre de l'arbre, és a dir, la mida
// del camí més llarg entre dos nodes fulla qualsevol.
int diameter();
```

IMPORTANT: No toquis la resta de la implementació de la classe, excepte si necessites afegir algun mètode auxiliar o atribut a la part privada.

D'entre els fitxers que s'adjunten a l'exercici també hi ha `program.cpp` (programa principal) i `Makefile` per a compilar i generar l'executable. El programa principal que t'oferim ja s'encarrega de llegir els arbres binaris i fer les crides al mètode indicat. **Només cal que implementis el mètode `diameter`.**

Per a pujar la teva solució, has de crear el fitxer `solution.tar` així:

```
tar cf solution.tar BinaryTree.hpp
```

Entrada

El programa principal té una variable d'arbre d'enters `t`, inicialment buida, i llegeix instruccions que, o bé mostren com és `t`, o bé modifiquen algun subarbre de `t` o mostren l'alçada d'algun subarbre de `t`. Les instruccions que mostren `t` són simplement de la forma `<< t`. Les altres instruccions comencen per `t`, seguit d'una seqüència de `.left` o `.right`. Finalment, o bé la instrucció acaba amb `.diameter`, cas en el qual s'escriurà el diàmetre del corresponent subarbre, o ve seguida de `= t'`, on `t'` és un string que representa un arbre, cas en el qual `t'` (com a arbre) serà assignat al corresponent subarbre de `t`. Per exemple:

```
t = 3(4,5(1,2))
<< t
t.diameter
t.left.diameter
t.right.diameter
t.right.left = 8(9,10)
<< t
t.right.diameter
```

La sortida de la seqüència anterior és:

```
3 (4, 5 (1, 2))
4
1
3
3 (4, 5 (8 (9, 10), 2))
5
```

Com pots observar, el `diameter` d'un arbre que està per sobre del que hem assignat també ha estat actualitzat.

Se suposa que la seqüència d'entrada serà correcta (sense accessos fora de l'arbre, tot i que sí que es pot accedir a subarbres buits de l'arbre).

El programa principal que t'ofereim ja s'encarrega de llegir aquestes entrades i fer les crides als corresponents mètodes de la classe `BinaryTree`. Només cal que feu les modificacions abans esmentades dins el fitxer `BinaryTree.hpp`.

Sortida

Per a cada instrucció `<< t`, s'escriurà el contingut actual de l'arbre. Per a cada instrucció acabada en `diameter`, s'escriurà el `diameter` del subarbre indicat. El programa que us oferim ja fa això. Només cal que feu les modificacions abans esmentades dins el fitxer `BinaryTree.hpp`.

Exemple d'entrada

```
t = 7(2,5)
t.diameter
<< t
t = 5(,1)
t.diameter
<< t
t.left = 4(2(,3),2)
t.left.diameter
<< t
t.left.left = 7(3,)
t.left.diameter
<< t
t.right = 5(6(1,),2)
t.diameter
<< t
t.left.right = 5(,8(,3))
t.diameter
<< t
t.left.right.left = 1(3,4(3,2))
t.diameter
<< t
t.right.right = 2(5(2,2),)
t.right.right.diameter
<< t
t.left.left = 6
t.left.left.diameter
<< t
t.right.right.left = 1
t.right.diameter
<< t
```

Exemple de sortida

```
3
7 (2, 5)
2
5 (, 1)
4
5 (4 (2 (, 3), 2), 1)
4
5 (4 (7 (3, ), 2), 1)
7
5 (4 (7 (3, ), 2), 5 (6 (1, ), 2))
8
5 (4 (7 (3, ), 5 (, 8 (, 3))), 5 (6 (1, ), 2))
9
5 (4 (7 (3, ), 5 (1 (3, 4 (3, 2)), 8 (, 3))), 5 (6 (1, ), 2))
3
5 (4 (7 (3, ), 5 (1 (3, 4 (3, 2)), 8 (, 3))), 5 (6 (1, ), 2 (5 (2, 2), )))
1
5 (4 (6, 5 (1 (3, 4 (3, 2)), 8 (, 3))), 5 (6 (1, ), 2 (5 (2, 2), )))
5
5 (4 (6, 5 (1 (3, 4 (3, 2)), 8 (, 3))), 5 (6 (1, ), 2 (1, )))
```

Informació del problema

Autor : Bernardino Casas

Generació : 2024-06-19 15:31:55

© *Jutge.org*, 2006–2024.

<https://jutge.org>