Suma màxima d'un camí entre dos nodes qualsevol de l'arbre binari X61481_ca

Implementeu una funció **RECURSIVA** que, donat un arbre binari d'enters, obtingues la suma màxima d'un camí entre dos nodes qualsevol de l'arbre binari. Aquesta és la capcelera:

```
// Pre: t és un arbre binari no buit que conté nombres enters.
// Post: Retorna la suma màxima possible d'un camí entre dos nodes
//       qualsevol de l'arbre donat. El camí de suma màxima pot
//       passar o no per l'arrel.
int maxSumPath(BinaryTree<int> t);
```

Aquí teniu dos exemples d'entrada de la funció i les seves corresponents sortides. El primer, el camí de suma màxima passa per l'arrel i el segon no passa per l'arrel:

```
2(4(7,6),5(3,4))
=>
22 -> 7+4+2+5+4
1(1(2,3),5(9,9))
=>
23 -> 5+9+9
```

Fixeu-vos que l'enunciat d'aquest exercici ja ofereix uns fitxers que haureu d'utilitzar per a compilar: `Makefile`, `program.cpp`, `BinaryTree.hpp`, `maxSumPath.hpp`. Us falta crear el fitxer `maxSumPath.cpp` amb els corresponents `includes` i implementar-hi la funció anterior. Quan pugeu la vostra solució al JUTGE, només cal que pugeu un **TAR** construït així:

```
tar cf solution.tar maxSumPath.cpp
```

Entrada

L'entrada té un nombre arbitrari de casos. Cada cas consisteix en una línia amb un string describint un arbre binari d'enters. Fixeu-vos en que el programa que us oferim ja s'encarrega de llegir aquestes entrades. Només cal que implementeu la funció abans esmentada.

Sortida

Per a cada cas, cal escriure la suma màxima d'un camí entre dos nodes qualsevol de l'arbre binari. Fixeu-vos en que el programa que us oferim ja s'encarrega d'escriure aquesta sortida. Només cal que implementeu la funció abans esmentada.

Exemple d'entrada 1

```
1(2(,-4),3(5(7,8),6))
1(2,3)
5(-1(11(-1,2),),8(13,7(,-1)))
3(4(-10,4),5)
-15(5(-8(2,-3),1),6(3,9(,0(4,-1(10,)))))
-10(9,20(15,7))
1(3,7(9,))
2(1(3,7(9,)),3)
```

Exemple de sortida 1

```
22
6
38
```

```
16
27
42
20
22
```

Exemple d'entrada 2

```
3(1(4(0,2),2(1,0)),2(1(4,5),3(5,-1)))
0(1(1,2),-1(4,1))
4(4(5(0(4,4),-1(2,4)),2(1(0,-1),1(1,0))),3(4(-1(5,2),5(1,1)),0(2(4,-1),0(0,1))))
3(0(2(1(5,3),4(2,4)),4(3(2,-1),4(2,1))),2(4(1(3,-1),1(-1,-1)),2(4(-1,3),3(-1,1))))
0(4(3(4(-1,-1),0(3,5)),3(-1(3,-1),2(-1,0))),4(-1(5(2,-1),5(0,3)),1(5(5,-1),5(5,0))))
2(2(3(2,3),2(4,0)),4(3(2,4),4(0,-1)))
0(3(4(-1,2),3(3,2)),3(2(2,2),2(2,1)))
3(3(4,5),4(3,-1))
0(1(-1(5(2,-1),5(4,1)),3(2(2,0),-1(-1,2))),0(2(-1(1,3),-1(5,1)),1(4(0,-1),1(-1,3))))
1(2(4,4),2(3,0))
```

Exemple de sortida 2

```
20
6
24
27
21
16
18
16
12
```

Observació

La vostra funció i subfuncions que creeu han de treballar només amb arbres. Heu de trobar una solució **RECURSIVA** del problema. En les crides recursives, incloeu la hipòtesi d'inducció, és a dir una explicació del que es cumpleix després de la crida, i també la funció de fita/decreixement o una justificació de perquè la funció recursiva acaba.
Una solució directa superarà els jocs de proves públics i us permetrà obtenir una nota raonable. Però molt possiblement serà lenta, i necessitareu crear alguna funció recursiva auxiliar per a produïr una solució més eficient capaç de superar tots els jocs de proves.
Avaluació sobre 10 punts:

- Solució lenta: 7 punts.
- Solució lenta + justificació: 8 punts.
- solució ràpida: 9 punts.
- solució ràpida + justificació: 10 punts.

Entenem com a solució lenta una que és correcta i capaç de superar els jocs de proves públics. Entenem com a solució ràpida una que és correcta i capaç de superar els jocs de proves públics i privats. La justificació val 1 punt i consisteix en definir correctament les PRE/POST de les funcions auxiliars que afegiu i en definir correctament les hipòtesis d'inducció i funcions de fita.

Informació del problema

Autoria: STUDENTS PRO1

Generació: 2026-01-25T21:19:01.238Z