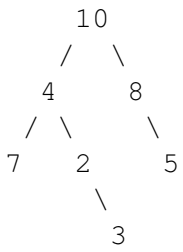
Mètode de BinaryTree que calcula el mínim de cada profunditatS39962_ca

Donada la classe `BinaryTree<T>`, implementa el mètode públic **RECURSIU**:

```
void minAtDepth(vector<T> &l) const;
```

Aquest mètode guarda en un vector el mínim dels valors dels nodes de l'arbre de cada nivell de profunditat tenint en compte que l'arrel es troba a profunditat 0. El vector hauria d'emmagatzemar a la casella 0 el mínim dels nodes a profunditat 0, a la casella 1 el mínim de profunditat 1, etc. El vector on es retorna el resultat inicialment és buit.

Exemple:



k	Nodes	`minAtDepth(k)`
0	10	10
1	4, 8	4
2	7, 2, 5	2
3	3	3

Què has de lliurar

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:

```
template <typename T>
void BinaryTree<T>::minAtDepth(vector<T>& l) const {
}
```

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

També pots trobar entre els fitxers que s'adjunten a l'exercici el fitxer `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 `minAtDepth`.**

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

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

Observacions

- No es pot usar cap contenidor auxiliar de la STL (`stack`, `queue`, `list`, ...) llevat de `vector`.
- Has de trobar una solució **RECURSIVA** i eficient del problema. En particular, no hi hauria d'haver cap bucle en cap de les funcions/accions que implementis.
- Recorda que tots els mètodes que implementis han de tenir les corresponents **Precondició** (Pre) i **Postcondició** (Post).
- En les crides recursives inclou la **hipòtesi d'inducció** (HI) i la **funció de fita** (FF).

Entrada

Una seqüència d'arbres binaris.

Sortida

Per a cada arbre binari s'escriurà el resultat del mètode `minAtDepth`.

El programa principal que t'oferim ja s'encarrega de llegir la seqüència d'arbres binaris i fer les crides corresponents al mètode de `BinaryTree` que se't demana d'implementar. Només cal que facis les modificacions abans esmentades dins el fitxer `BinaryTree.hpp`.

Per més detalls de com és l'entrada i la sortida consulta els jocs de proves públics.

Exemple d'entrada 1

```
()
0
1
2
5(3,)
5(,3)
5(1,)
5(,10)
5(,10(7,))
5(,20(,30))
7(2,15)
4(2(,3),12)
10(3,14(13,20))
3(1,30(4,52))
20(5(2,12),)
20(5(31,12),50(52,2))
20(15(12(6(3,),),),),)
10(4(7,2(()),3)),8((),5))
1(2(4,6),5(3,7))
10(,8(,6(,4(,2))))
10(8(6(4(2,),),),),)
```

Exemple de sortida 1

```
() --> []
0 --> [0]
1 --> [1]
2 --> [2]
5(3,) --> [5, 3]
5(,3) --> [5, 3]
5(1,) --> [5, 1]
5(,10) --> [5, 10]
5(,10(7,)) --> [5, 10, 7]
5(,20(,30)) --> [5, 20, 30]
7(2,15) --> [7, 2]
4(2(,3),12) --> [4, 2, 3]
10(3,14(13,20)) --> [10, 3, 13]
3(1,30(4,52)) --> [3, 1, 4]
20(5(2,12),) --> [20, 5, 2]
20(5(31,12),50(52,2)) --> [20, 5, 2]
20(15(12(6(3,),),),),) --> [20, 15, 12, 6, 3]
10(4(7,2),) --> [10, 4, 2]
1(2(4,6),5(3,7)) --> [1, 2, 3]
10(,8(,6(,4(,2)))) --> [10, 8, 6, 4, 2]
10(8(6(4(2,),),),),) --> [10, 8, 6, 4, 2]
```

Exemple d'entrada 2

```
-47(44,-15(-19(-22(,-33(,-37)),),36(,19)))
-21(14(27,-47),21(-25(41(,33),39),19(,3)))
-7(-15(-31(-23(,47(-37,-39)),-2(,-38)),-5
40(-42(-45(34(,-21),48(-13,)),-40),-38(-2,-15))
35(-16(,39(37,)),32(-41(27,31),-29(18(,43),)))
```

```
48(49(-43(-21(-46,),),-10((),1(-16,))),-42(-23,22(,41(
24(1(-4(-22(-33((),15(13,-39)),46),-44(-36,-31)),30),-
17(-18(20,),-49(,37(42(-36,18(,46)),-16(48(32(,-7),-13
64(-71(-93(,90(-29(-37,-42(-64(89,-73),)),)),74(40(-77,
(-6(27,),)),-17(-45,))
```

Exemple de sortida 2

```
-47(44,-15(-19(-22(,-33(,-37)),),36(,19))) --> [47, 15, -19, -22, -33, -37]
-21(14(27,-47),21(-25(41(,33),39),19(,3))) --> [21, 14, 47, 3, 33]
-7(-15(-31(-23(,47(-37,-39)),-2(,-38)),-5(-6(27,),)),-17(-45,)) --> [-7, -17, -45, -23, -38, -39
40(-42(-45(34(,-21),48(-13,)),-40),-38(-2,-15)) --> [40, -42, -45, 34, -21]
35(-16(,39(37,)),32(-41(27,31),-29(18(,43),))) --> [35,
48(49(-43(-21(-46,),),-10),) --> [48, 49, -43, -21, -46
24(1(-4(-22(-33,),),),) --> [24, 1, -4, -22, -33]
17(-18(20,),-49(37(42(-36,18(,46)),-16(48(32(,-7),-13
64(-71(-93(,90(-29(37,-42(-64(89,-73),)),)),74(40(-77,
```

Informació del problema

Autoria: Bernardino Casas

Generació: 2026-06-11T20:22:05.699Z