Mètode de la classe BinaryTree amb un ABC que genera el camí entre l'arrel i l'element indicat

U64796_ca

Implementa un nou mètode RECURSIU de la classe `BinaryTree` que genera una llista amb el camí entre l'arrel i l'element indicat. Aquest arbre binari emmagatzema un **Arbre Binari de Cerca** (ABC, o BST en anglès).

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: p.i. emmagatzema un arbre binari de cerca sense elements repetits,
//    res és una llista buida
// Post: Torna a res el camí entre l'arrel i x. Si x no es troba en el p.i.
//    llavors res serà buit.
template <typename T>
void BinaryTree<T>::route_bst(const T &x, list<T> &res) 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 `route_bst`**.

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

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

Observació

Recorda que si crees funcions auxiliars, has d'afegir-hi les corresponents **Precondició** (Pre) i **Postcondició** (Post). 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. En les crides recursives inclou la **hipòtesi d'inducció** (HI) i la **funció de fita** (FF).

Entrada

Una seqüència d'arbres binaris de cerca.

Sortida

Per a cada arbre binari de cerca s'escriurà el resultat del mètode `route_bst`.

El programa principal que t'oferim ja s'encarrega de llegir la seqüència d'arbres binaris i fer les crides als 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 ()
0 0
1 0
2 0
1 1
2 1
1 2
2 2

5 5(,10)
10 5(,10)
2 5(,10)
7 5(,10)
17 5(,10)

1 5(1,)
5 5(1,)
0 5(1,)
3 5(1,)
7 5(1,)

5 5(,10(7,))
10 5(,10(7,))
7 5(,10(7,))
1 5(,10(7,))
8 5(,10(7,))
11 5(,10(7,))

5 5(,20(,30))
20 5(,20(,30))
30 5(,20(,30))
1 5(,20(,30))
10 5(,20(,30))
25 5(,20(,30))
40 5(,20(,30))

7 7(2,15)
2 7(2,15)
15 7(2,15)
1 7(2,15)
3 7(2,15)
10 7(2,15)
20 7(2,15)

4 4(2(,3),12)
2 4(2(,3),12)
3 4(2(,3),12)
12 4(2(,3),12)
1 4(2(,3),12)
5 4(2(,3),12)
13 4(2(,3),12)

10 10(3,14(13,20))
3 10(3,14(13,20))
14 10(3,14(13,20))
13 10(3,14(13,20))
20 10(3,14(13,20))
1 10(3,14(13,20))
4 10(3,14(13,20))
11 10(3,14(13,20))
15 10(3,14(13,20))
21 10(3,14(13,20))
100 10(3,14(13,20))

3 3(1,30(4,52))
1 3(1,30(4,52))
30 3(1,30(4,52))
4 3(1,30(4,52))
52 3(1,30(4,52))
0 3(1,30(4,52))
2 3(1,30(4,52))
5 3(1,30(4,52))
15 3(1,30(4,52))
35 3(1,30(4,52))
55 3(1,30(4,52))

20 20(5(2,12),)
5 20(5(2,12),)
2 20(5(2,12),)
12 20(5(2,12),)
1 20(5(2,12),)
3 20(5(2,12),)
6 20(5(2,12),)
13 20(5(2,12),)
25 20(5(2,12),)

20 20(5(2,12),50(31,52))
5 20(5(2,12),50(31,52))
2 20(5(2,12),50(31,52))
12 20(5(2,12),50(31,52))
50 20(5(2,12),50(31,52))
31 20(5(2,12),50(31,52))
52 20(5(2,12),50(31,52))
1 20(5(2,12),50(31,52))
3 20(5(2,12),50(31,52))
6 20(5(2,12),50(31,52))
13 20(5(2,12),50(31,52))
21 20(5(2,12),50(31,52))
34 20(5(2,12),50(31,52))
```

```
51 20(5(2,12),50(31,52))
60 20(5(2,12),50(31,52))
100 20(5(2,12),50(31,52))

20 20(15(12(6(3,),),),)
15 20(15(12(6(3,),),),)
12 20(15(12(6(3,),),),)
6 20(15(12(6(3,),),),)
3 20(15(12(6(3,),),),)
1 20(15(12(6(3,),),),)
5 20(15(12(6(3,),),),)
7 20(15(12(6(3,),),),)
11 20(15(12(6(3,),),),)
19 20(15(12(6(3,),),),)
25 20(15(12(6(3,),),),)
```

Exemple de sortida 1

```
0 () --> []
1 () --> []
0 0 --> [0]
1 0 --> []
2 0 --> []
1 1 --> [1]
2 1 --> []
1 2 --> []
2 2 --> [2]
5 5(,10) --> [5]
10 5(,10) --> [5, 10]
2 5(,10) --> []
7 5(,10) --> []
17 5(,10) --> []
1 5(1,) --> [5, 1]
5 5(1,) --> [5]
0 5(1,) --> []
3 5(1,) --> []
7 5(1,) --> []
5 5(,10(7,)) --> [5]
10 5(,10(7,)) --> [5, 10]
7 5(,10(7,)) --> [5, 10, 7]
1 5(,10(7,)) --> []
8 5(,10(7,)) --> []
11 5(,10(7,)) --> []
5 5(,20(,30)) --> [5]
20 5(,20(,30)) --> [5, 20]
30 5(,20(,30)) --> [5, 20, 30]
1 5(,20(,30)) --> []
10 5(,20(,30)) --> []
25 5(,20(,30)) --> []
40 5(,20(,30)) --> []
7 7(2,15) --> [7]
2 7(2,15) --> [7, 2]
15 7(2,15) --> [7, 15]
1 7(2,15) --> []
3 7(2,15) --> []
10 7(2,15) --> []
20 7(2,15) --> []
4 4(2(,3),12) --> [4]
2 4(2(,3),12) --> [4, 2]
3 4(2(,3),12) --> [4, 2, 3]
12 4(2(,3),12) --> [4, 12]
1 4(2(,3),12) --> []
5 4(2(,3),12) --> []
13 4(2(,3),12) --> []
10 10(3,14(13,20)) --> [10]
3 10(3,14(13,20)) --> [10, 3]
14 10(3,14(13,20)) --> [10, 14]
13 10(3,14(13,20)) --> [10, 14, 13]
20 10(3,14(13,20)) --> [10, 14, 20]
1 10(3,14(13,20)) --> []
4 10(3,14(13,20)) --> []
11 10(3,14(13,20)) --> []
15 10(3,14(13,20)) --> []
21 10(3,14(13,20)) --> []
100 10(3,14(13,20)) --> []
3 3(1,30(4,52)) --> [3]
1 3(1,30(4,52)) --> [3, 1]
```

```
30 3(1,30(4,52)) --> [3, 30]
4 3(1,30(4,52)) --> [3, 30, 4]
52 3(1,30(4,52)) --> [3, 30, 52]
0 3(1,30(4,52)) --> []
2 3(1,30(4,52)) --> []
5 3(1,30(4,52)) --> []
15 3(1,30(4,52)) --> []
35 3(1,30(4,52)) --> []
55 3(1,30(4,52)) --> []
20 20(5(2,12),) --> [20]
5 20(5(2,12),) --> [20, 5]
2 20(5(2,12),) --> [20, 5, 2]
12 20(5(2,12),) --> [20, 5, 12]
1 20(5(2,12),) --> []
3 20(5(2,12),) --> []
6 20(5(2,12),) --> []
13 20(5(2,12),) --> []
25 20(5(2,12),) --> []
20 20(5(2,12),50(31,52)) --> [20]
5 20(5(2,12),50(31,52)) --> [20, 5]
2 20(5(2,12),50(31,52)) --> [20, 5, 2]
12 20(5(2,12),50(31,52)) --> [20, 5, 12]
50 20(5(2,12),50(31,52)) --> [20, 50]
31 20(5(2,12),50(31,52)) --> [20, 50, 31]
52 20(5(2,12),50(31,52)) --> [20, 50, 52]
1 20(5(2,12),50(31,52)) --> []
3 20(5(2,12),50(31,52)) --> []
6 20(5(2,12),50(31,52)) --> []
13 20(5(2,12),50(31,52)) --> []
21 20(5(2,12),50(31,52)) --> []
34 20(5(2,12),50(31,52)) --> []
51 20(5(2,12),50(31,52)) --> []
60 20(5(2,12),50(31,52)) --> []
100 20(5(2,12),50(31,52)) --> []
20 20(15(12(6(3,),),),) --> [20]
15 20(15(12(6(3,),),),) --> [20, 15]
12 20(15(12(6(3,),),),) --> [20, 15, 12]
6 20(15(12(6(3,),),),) --> [20, 15, 12, 6]
3 20(15(12(6(3,),),),) --> [20, 15, 12, 6, 3]
1 20(15(12(6(3,),),),) --> []
5 20(15(12(6(3,),),),) --> []
7 20(15(12(6(3,),),),) --> []
11 20(15(12(6(3,),),),) --> []
19 20(15(12(6(3,),),),) --> []
25 20(15(12(6(3,),),),) --> []
```

Informació del problema

Autoria: Bernardino Casas

Generació: 2026-01-25T20:34:54.356Z