
Arbre de xifres**S66867_ca**

Implementeu una funció **RECURSIVA** que, donat un arbre binari no buit de nombres naturals i un dígit (0,1,2,3,4,5,6,7,8 o 9), retorna `true` si qualsevol dels nodes de l'arbre conté el dígit.

Aquesta és la capçalera de la funció:

```
bool treeContainsDigit(BinaryTree<int>& t, int d);
```

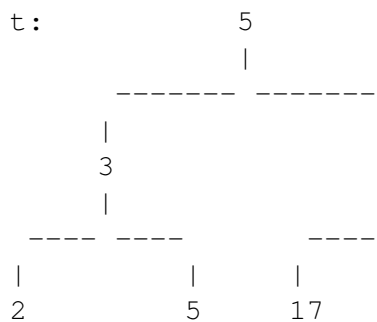
Aquesta funció principal ha de fer servir una funció auxiliar **ITERATIVA**, que també cal que implementeu, per comprovar si un nombre natural conté un dígit. Tingueu en compte que no es pot fer servir cap funció que converteixi el nombre a cadena.

La funció auxiliar ha de tenir la següent capçalera:

```
bool numberContainsDigit(int n, int d);
```

Aquí tenim un exemple de paràmetres d'entrada de la funció principal i la corresponent sortida, que és `true` degut a que l'arbre conté la xifra 7 dins del node 17:

d: 7



=>

`true`

Fixeu-vos que l'enunciat d'aquest exercici ja ofereix uns fitxers que haureu d'utilitzar per a compilar: `Makefile`, `program.cpp`, `BinaryTree.hpp`, `digitsTree.hpp`. Us falta crear el fitxer `digitsTree.cpp` amb els corresponents `includes` i implementar-hi les dues funcions anteriors. Quan pugueu la vostra solució al jutge, només cal que pugueu un tar construït així:

```
tar cf solution.tar digitsTree.cpp
```

Entrada

La primera línia de l'entrada conté el dígit que es busca a tots els arbres. La segona descriu el format en el que es descriuen els arbres, o bé `INLINEFORMAT` o bé `VISUALFORMAT`. Després venen un nombre arbitrari de casos. Cada cas consisteix en una descripció d'un arbre binari no buit de nombres naturals majors o igual a zero. Fixeu-vos en que el programa que us oferim ja s'encarrega de llegir aquestes entrades. Només cal que implementeu les funcions abans esmentades.

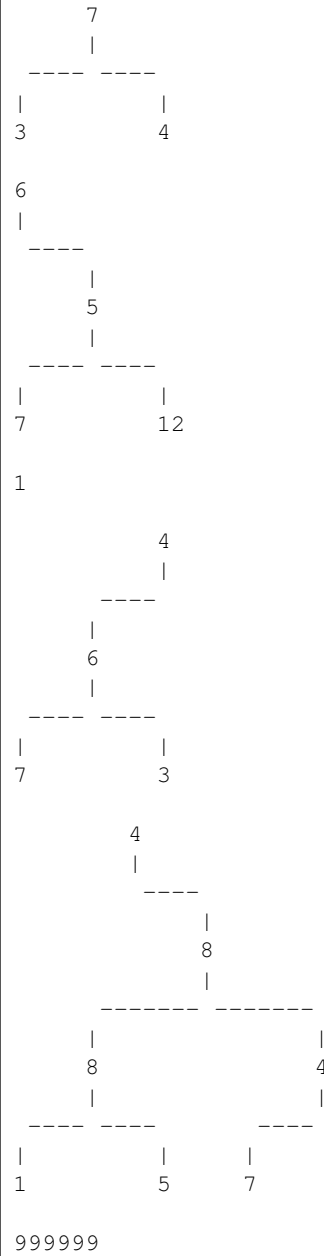
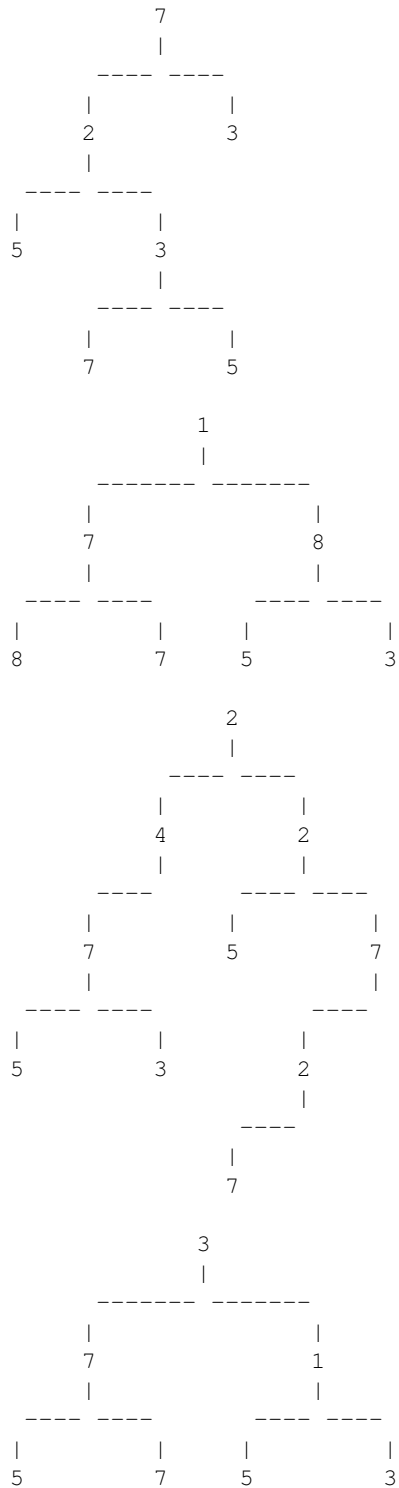
Sortida

Per a cada cas, la sortida és SI o NO. Fixeu-vos en que el programa que us oferim ja s'encarrega d'escriure aquest resultat. Només cal que implementeu les funcions abans esmentades.

Exemple d'entrada 1

1

VISUALFORMAT



999999

Exemple de sortida 1

NO
SI
NO
SI

NO
SI
SI
NO
SI
NO

Exemple d'entrada 2

```
1
INLINEFORMAT
6 (2, 3)
0 (55 (29, 3 (13, 5)), 17)
0 (55 (29, 3 (2266, 5)), 9878977)
9 (2 (3 (4 (5 (6 (7 (8 (9 (99))))))))))
44 (52 (33, 4), 35 (46 (71 (85, ), 569 (20, 900)), ))
1 (2 (3 (4 (5 (6 (7 (8 (9 (10 (11, 12))))))))))
7 (4 (3 (5), 6 (7 (7 (7 (33))))), 6)
15 (0, 2)
22 (33 (44, 55), 66 (77, 88 (99)))
54851863
```

Exemple de sortida 2

NO
SI
NO
NO
SI
SI
NO
NO
SI

Observació

- Les funcions/accions que implementis han de treballar només amb arbres binaris i han de tenir les corresponents **PRE** i **POST**.
- En les crides recursives, inclou la **hipòtesi d'inducció**, és a dir una explicació del que es compleix després de la crida, i també la **funció de fita/decreixement**.
- En els bucles inclou l'**invariant del bucle** i la **funció de fita**.
- Finalment, afegeix un comentari al final de tot del teu codi amb la **justificació informal** de la correctesa de la funció recursiva.

Informació del problema

Autoria: Alfonso da Silva

Generació: 2026-01-25T20:33:56.231Z

© Jutge.org, 2006–2026.

<https://jutge.org>