

---

**Haskell — Arbres AVL****P79515\_ca**

---

Si ja heu fet el problema dels Arbres Binaris de Cerca, ara us podeu lluir extenent la vostra implementació a Arbres AVL. En aquest problema es demana que definiu operacions per inserir en AVLs i per comprovar-ne la correctesa.

Per als AVLs, es defineix el tipus següent:

```
data AVL a
  = E | N a Int (AVL a) (AVL a)    -- l'enter guarda l'alçada
  deriving (Show)
```

Sobre aquest tipus cal crear les operacions següents:

- **insert :: Ord a => AVL a -> a -> AVL a**  
Retorna el resultat d'inserir un element en un arbre AVL. Si l'element ja hi era, el resultat és l'arbre original.
- **create :: Ord a => [a] -> AVL a**  
Retorna un arbre AVL tot inserint un rera l'altre la llista d'elements donats.
- **check :: AVL a -> (Bool, Int)**  
Donat un arbre, si és AVL, retorna cert i la seva alçada; si no és AVL, retorna fals i -99.  
Els arbres buits es consideren d'alçada -1, les fulles es consideren d'alçada zero, ...);

**Exemple d'entrada 1**

```
create [1..3]
create [1..5]
check $ create [1..3]
check $ create [1..5]
check (E :: AVL Int)
check (N 1 2 (N 2 1 (N 3 0 E E) E) E)
```

**Exemple de sortida 1**

```
N 2 1 (N 1 0 E E) (N 3 0 E E)
N 2 2 (N 1 0 E E) (N 4 1 (N 3 0 E E) (N 5 0 E E))
(True, 1)
(True, 2)
(True, -1)
(False, -99)
```

**Informació del problema**

Autoria: Jordi Petit

Generació: 2026-02-03T16:59:46.937Z

© Jutge.org, 2006–2026.

<https://jutge.org>