Alberi

Cammino radice–discendente crescente

Dato un albero binario contenente interi, scrivere un algoritmo che restituisca la lunghezza del più lungo cammino monotono crescente radice-discendente, dove:

- il discendente non è necessariamente foglia;
- con lunghezza si intende il numero totale di *archi* attraversati;
- con monotona crescente si intende che i valori contenuti nei nodi della sequenza devono essere ordinati in senso crescente da radice a discendente.

Discuterne correttezza e complessità.

Alberi – Larghezza albero

La larghezza di un albero ordinato è il numero massimo di nodi che stanno tutti al medesimo livello. Si fornisca una funzione che calcoli in tempo ottimo la larghezza di un albero ordinato T di n nodi.

Spoiler alert!

Alberi – Percorso cammino-discendente

```
integer monotone(TREE T)
integer maxl ← 0
integer maxr ← 0
if T ≠ nil then
    if T.left() ≠ Nil and T.left().value > T.value then
        maxl ← 1 + monotone(T.left())
    if T.right() ≠ Nil and T.right().value > T.value then
        maxr ← 1 + monotone(T.right())
return max(maxl, maxr)
```

Larghezza (1)

integer larghezza(TREE t)

```
integer larghezza ← 1
integer level ← 1
integer count ← 1
QUEUE Q ← Queue()
Q.enqueue(t)
t.level ← 0
while not Q.isEmpty() do
    TREE u ← Q.dequeue()
    if u.level ≠ level then
        level ← u.level
        count ← 0
    count ← count + 1
    larghezza ←
     max(larghezza, count)
    TREE v ← u.leftmostChild()
    while v ≠ nil do
        v.level ← u.level + 1
        Q.enqueue(v)
        v ← v.rightSibling()
return larghezza
```

Larghezza (2)

```
integer larghezza(Tree t)
integer count ← 1          % # nodi nel livello corrente da visitare; radice
integer larghezza ← 1      % Massimo larghezza trovata finora; radice
Queue Q ← Queue()
Q.enqueue(t)
while not Q.isEmpty() do
    Tree u ← Q.dequeue()
    Tree v ← u.leftmostChild()
    while v ≠ nil do
        Q.enqueue(v)
        v ← v.rightSibling()
    count ← count − 1
    if count = 0 then                          % Nuovo livello
        count = Q.size()
        larghezza ← max(larghezza, count)
return larghezza
```

Larghezza (3) – Alberi binari

integer larghezza(TREE t)

VECTOR $count \leftarrow$ **new** VECTOR
larghezza($t, count, 0$)
return max($count$)

larghezza(TREE t, VECTOR $count$, **integer** $level$)

if $t \neq$ **nil then**
    $count[level] \leftarrow count[level] + 1$
    larghezza($t.left, count, level + 1$)
    larghezza($t.right, count, level + 1$)