Ricerca i-simo in un ABR

- Scrivere una funzione che riceve un ABR di interi in input e un intero i e restituisce il puntatore al nodo contenente il valore i-simo nell'ordine, NULL se i e' maggiore del numero dei nodi o l'albero e' vuoto o i<=0.

Soluzione 1

```
void trovalesimo(TreePtr t , TreePtr* temp, int* i)
/* postc: restituisce in temp il puntatore all'i-simo
   elemento nell'ordine se i >=0 e minore del numero dei
   nodi, NULL altrimenti*/
{if (!t) return;
trovalesimo(t->left,temp,i);
 if ((*i) == 1) *temp = t;
(*i)--;
if ( (*i) >= 1) trovalesimo (t->right,temp,i);
}
```

Soluzione 2

```
TreePtr trovalesimo2(TreePtr t , int* i)
/* postc: restituisce  il puntatore all'i-simo
   elemento nell'ordine se i >=0 e minore del
   numero dei nodi, NULL altrimenti*/
{TreePtr temp;
if (!t ) return NULL;
temp = trovalesimo2(t->left,i);
if ((*i) == 1) temp = t;
(*i)--;
if ( (*i) >= 1) temp = trovalesimo2 (t->right,i);
return temp;}
```

Verifica bilanciamento nel numero dei nodi: definizioni.

Definizione:

Un albero è bilanciato nel Numero dei Nodi, brevemente n-bilanciato , quando, per ogni sottoalbero t radicato in un suo nodo, il numero dei nodi del sottoalbero sinistro di t, meno il numero dei nodi del sottoalbero destro di t è in valore assoluto al più 1.

Alberi Bilanciati nel numero dei nodi: definizione ricorsiva

Base: l'albero vuoto è n-bilanciato
passo ricorsivo:
un albero t, con sottoalberi t1 e t2, rispettivamente con n1 e n2 nodi, è n-bilanciato sse

t1 e t2 sono n-bilanciati

e

$|n1-n2| \leq 1$.

Verifica bilanciamento nel numero dei nodi. 1°

```
int nBilIneff(TreePtr  tPtr)
/* post: restituisce 1 se l'albero t e' bilanciato nel
numero di nodi, ossia se la differenza tra  il numero dei
nodi nel sottoalbero sinistro e quello nel sottoalbero
destro di ogni nodo è minore di 1 in valore assoluto,
altrimenti restituisce 0. VERSIONE INEFFICIENTE */
 { int nnr, nnl;
   if (!tPtr) return 1;
   if (nBilIneff(tPtr->lPtr))
       if (nBilIneff(tPtr->rPtr))
          { nnl = numNodi(tPtr->lPtr);
            nnr = numNodi(tPtr->rPtr);
            if (abs(nnr - nnl)<=1) return 1;
          }
  return 0;
}
```

Verifica bilanciamento nel numero dei nodi. 2°

```
int nBilEffAus(TreePtr  tPtr)
/* Verifica il bilanciamento di tPtr con un unica scansione
dell'albero. Ogni chiamata restituisce il numero dei nodi del
   sottoalbero radicato nel nodo della chiamata, se questo è
   bilanciato, mentre restituisce -1 altrimenti.
postc: restituisce -1 se l'albero non e' bilanciato nel numero
   dei nodi, un valore positivo altrimenti.*/
{ int nnl, nnr;
  if (!tPtr) return 0;
          else
          { nnl = nBilEffAus(tPtr->lPtr);
            if (nnl >= 0)
                 { nnr = nBilEffAus(tPtr->rPtr);
                   if (nnr>=0 && abs(nnl-nnr)<=1)
                       return (nnl+nnr+1);
                  }
           }
   return -1;
 }
```

Verifica bilanciamento fine

```
int nBilEff(TreePtr  tPtr)
 /* verifica il bilanciamento nel numero di nodi di
   tPtr,
   *postc: restuisce 1 se l'albero è bilanciato, 0
   altrimenti
   */
 { if (nBilEffAus(tPtr) == -1) return 0;
      else return 1;
         }
```