```
// RIMOZIONE DI UN NODO
TREE removeNode(TREE T, ITEM k)
    // individuo il nodo da rimuovere
    TREE u ← lookupNode(T, k)

    // se il nodo da rimuovere è presente nell'albero...
    if u ≠ nil then
(1)     // ...e non ha figli
        if u.left == nil and u.right == nil then
            if u.parent ≠ nil then // se esiste il padre
                link(u.parent, nil, k) // rimuovo il puntatore al figlio

            // rimuovo direttamente il nodo
            delete u
(3)     // ...ed ha due figli
        if u.left ≠ nil and u.right ≠ nil then
            TREE s ← successorNode // individuo il successore

            link(s.parent, s.right, s.key) // collego il sottoalbero destro

            // copio il successore
            // nella posizione del nodo rimosso
            u.key ← s.key
            u.value ← s.value

            // rimuovo il successore
            delete s
(2)     // ...ed ha un solo figlio (sinistro)
        if u.left ≠ nil and u.right == nil then
            link(u.parent, u.left, k) // collega il figlio al padre

            if u.parent == nil then // se il padre non esiste
                T == u.right // il figlio diventa la radice
        // ...ed ha un solo figlio (sinistro)
        else
            link(u.parent, u.right, k) // collega il figlio al padre

            if u.parent == nil then // se il padre non esiste
                T == u.right // il figlio diventa la radice

    // restituisco la radice
    return T
```