```
// bilanciamento di un RED-BLACK TREE in seguito all'inserimento di un nodo RED
balanceInsert(TREE t)
    t.color ← RED  // coloro il nodo da inserire di rosso

    // t==nil è la condizione di fino ciclo
    finché t≠nil fai
        TREE p ← t.parent                                          // riferimento al padre
        TREE n ← iif(p≠nil, p.parent, nil)                         // riferimento al nonno
        TREE z ← iif(n==nil, nil, iif(n.left, n.right, n.left))    // riferimento allo zio
(1)     se p==nil allora
            t.color ← BLACK
            t ← nil  // fine
(2)     se p.color==BLACK allora
            t ← nil  // fine
(3)     se z.color==RED allora
            p.color ← z.color ← BLACK
            n.color ← RED
            t ← n  // passo il problema al nonno
        altrimenti
(4a)        se (t==p.right) and (p==n.left) allora
                rotateLeft(p)
                t ← p  // passo il problema al padre
(4b)        se (t==p.left) and (p==n.right) allora
                rotateRight(p)
                t ← p  // passo il problema al padre
            altrimenti
(5a)            se (t==p.left) and (p==n.left) allora
                    rotateRight(n)
(5b)            altrimenti se (t==p.right) and (p==n.right) allora
                    rotateLeft(n)

                p.color ← BLACK
                n.color ← RED
                t ← nil  // fine
```