

// bilanciamento di un RED-BLACK TREE in seguito alla rimozione di un nodo RED

balanceDelete(TREE *t*)

t.color ← RED // colore il nodo da inserire di rosso

while (*t* ≠ *T*) **and** (*t.color* == BLACK) **do**

 TREE *p* ← *t.parent* // riferimento al padre

if *t* == *p.left* **then**

 TREE *f* ← *p.right* // riferimento al fratello

 TREE *ns* ← *f.left* // riferimento al nipote sinistro

 TREE *nd* ← *f.right* // riferimento al nipote destro

if *f.color* == RED **then**

p.color ← RED

f.color ← BLACK

 rotateLeft(*p*)

 // *t* viene lasciato inalterato, quindi si ricade nei casi 2, 3, 4

else

if *ns.color* == *nd.color* == BLACK **then**

f.color == RED

t ← *p* // passo il problema al padre

else if (*ns.color* == RED) **and** (*nd.color* == BLACK) **then**

ns.color ← BLACK

f.color ← RED

 rotateRight(*f*)

 // *t* viene lasciato inalterato, quindi si ricade nel caso 4

else if *nd.color* == RED **then**

f.color == *p.color*

p.color ← BLACK

nd.color ← BLACK

 rotateLeft(*p*)

t ← *T*

else

 // casi speculari