

```
// 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
```

```
    finché ( $t \neq T$ ) and (t.color == BLACK) fai
```

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

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

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

```
    se f.color == RED allora
```

```
        p.color ← RED
```

```
        f.color ← BLACK
```

```
        rotateLeft(p)
```

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

```
    altrimenti
```

```
        se ns.color == nd.color == BLACK allora
```

```
            f.color == RED
```

```
            t ← p // passo il problema al padre
```

```
        altrimenti se (ns.color == RED) and (nd.color == BLACK) allora
```

```
            ns.color ← BLACK
```

```
            f.color ← RED
```

```
            rightRotation(f)
```

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

```
        altrimenti se nd.color == RED allora
```

```
            f.color == p.color
```

```
            p.color ← BLACK
```

```
            nd.color ← BLACK
```

```
            leftRotation(p)
```

```
            t ← T
```