```
TREE huffman(int[ ] c, int[ ] f, int n)
    // c[ ]in:  caratteri dell'alfabeto
    // f[1...n]:  frequenze dei caratteri
    // n:  dimensione dell'alfabeto
    PRIORITYQUEUE Q ← MinPriorityQueue
    from i ← 1 until n do //                                        O(n)
        Q.inserisci(f[i], Tree(f[i], c[i])) //                      O(log n)
    from i ← 1 until n − 1 do // n:  radice                         O(n)
        // estraggo i 2 caratteri meno frequenti
        z1 ← Q.deleteMin
        z2 ← Q.deleteMin
        // Creo un nuovo nodo
        z ← Tree(z1.f + z2.f, nil)
        z.left ← z1
        z.right ← z2
        // Lo inserisco nella coda
        Q.inserisci(z.f, z)
    return Q.deleteMin
```