```
// effettua una visita in profondità del grafo, esaminando i nodi
   nell'ordine inverso di tempo di fine della prima visita.
int[ ] scc(Graph G)
    Stack S ← topSort(G)  // otteniamo i nodi in ordine decrescente
     di fine
    G^T ← transpose(G)  // inverte il senso degli archi
    ritorna cc(G^T, S) // esegue una visita dfs sul grafo trasposto

// parte iterativa
// cc modificato in 88/101
int[ ] cc(Graph G, Stack S)
    int[ ] id ← new int[1...G.size]

    per ciascun u ∈ G.V fai
        id[u] ← 0

    int counter ← 0
    finché not S.isEmpty fai
        u ← S.pop  // estrazione in tempo inverso di tempo di fine

        se id[u] == 0 allora
            counter++
            ccdfs(G, counter, n, id)

    ritorna id

// parte ricorsiva
ccdfs(Graph G, int counter, Node n, int[ ] id)
    id[u] ← counter  // assegna il contatore a tutti i nodi che incontro

    per ciascun u ∈ G.adj(u) fai
        se id[u] == 0 allora
            ccdfs(G, counter, u, id)
```