

```
// 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
```

```
     $G^T \leftarrow \text{transpose}(G)$  // 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  $u \in G.V$  fai
```

```
         $id[u] \leftarrow 0$ 
```

```
    int counter ← 0
```

```
    finché not S.isEmpty fai
```

```
         $u \leftarrow S.pop$  // 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)
```

```
     $id[u] \leftarrow counter$  // assegna il contatore a tutti i nodi che incontro
```

```
    per ciascun  $u \in G.adj(u)$  fai
```

```
        se  $id[u] == 0$  allora
```

```
            ccdfs(G, counter, u, id)
```