```
STACK topSort(GRAPH G)
    // inizializzazione
    bool[ ] visited ← new bool[1 ... G.n]
    per ciascun u ∈ G.V fai  visited[u] ← falso

    per ciascun u ∈ G.V fai
        // per ogni nodo del grafo
        se not visitato[u] allora
            // se non l'hai visitato
            // effettua una chiamata ricorsiva
            ts-dfs(G, u, visitato, S)

    ritorna STACK

// restituisce l'ordinamento topologico dei nodi di un DAG
// questo algoritmo funziona partendo da qualsiasi nodo
int ts-dfs(GRAPH G, NODE u, bool[ ] visitato, NODE[ ] STACK S)
    visitato[u] ← vero  // imposta il nodo come visitato
    per ciascun v ∈ G.adj(u) fai
        // è una grafo diretto aciclico quindi non ho bisogno di ricordare da dove sono
           venuto
        se not visitato[v] allora
            // effettua una visita in profondità
            i ← ts-dfs(G, u, visitato, S)

    S.push  // aggiungi il nodo in testa alla pila
    // quando ho terminato tutte le chiamate ricorsive l'algoritmo mi restituirà l'ordine
       topologico dei nodi del grafo dato in input, il nodo viene messo in testa in modo tale
       che si trovi prima dei nodi che i suoi archi puntano, ossia i suoi discendenti
```