An algorithm to find cycles in a graph and assign topological numbers

This algorithm assigns a topological numbering to a graph $G = (V, E)$, and returns false if there is no such numbering (i.e., if the graph contains cycles).

The array $T[]$ is an array of n integers ($n = |V|$) and has the following meaning (the colors correspond to our applet from the Depth-First-Search lecture):

T[i]	Contents
0	node is "white" (has not been explored)
-1	node is "green" (has been explored but is not finished)
$1 \ldots n$	node is "black" (done). The number is the topological number of node i

```
boolean $TOPO(V, E)$
    int $T[n]$
    $toponum \leftarrow n$
    **for each**$(v \in V)$
        $T[v] \leftarrow 0$
    **for each**$(v \in V)$
        **if** $(T[v] = 0)$
            **if** $(\neg TOPOrec(V, E, v))$ **return** false
    **return** true
```

```
boolean$TOPOrec(V, E, v)$
    /* mark node $v$ green */
    $T[v] \leftarrow -1$
    **for each**$(w \in V)$ adjacent to $v$
        /* if $w$ is green, we have a cycle */
        **if** $(T[w] = -1)$ **return** $false$
        /* if $w$ is black, we don't have a cycle, but we don't need to do recursion */
        **else if** $(T[w] = 0)$
                **if** $(\neg TOPOrec(V, E, w))$ **return** false
    /* if all is well, we assign a topological number, which implicitly colors the node $v$ black */
    $T[v] \leftarrow toponum$
    $toponum \leftarrow toponum - 1$
    **return** true
```