Depth-first search [edit]

An alternative algorithm for topological sorting is based on depth-first search. The algorithm loops through each node of the graph, in an arbitrary order, initiating a depth-first search that terminates when it hits any node that has already been visited since the beginning of the topological sort or the node has no outgoing edges (i.e. a leaf node):

```
L ← Empty list that will contain the sorted nodes
while there are unmarked nodes do
    select an unmarked node n
    visit(n)
```

```
function visit(node n)
    if n has a permanent mark then return
    if n has a temporary mark then stop (not a DAG)
    if n is not marked (i.e. has not been visited yet) then
        mark n temporarily
        for each node m with an edge from n to m do
            visit(m)
        mark n permanently
        add n to head of L
```

Each node *n* gets *prepended* to the output list L only after considering all other nodes which depend on *n* (all descendants of *n* in the graph). Specifically, when the algorithm adds node *n*, we are guaranteed that all nodes which depend on *n* are already in the output list L: they were added to L either by the recursive call to visit() which ended before the call to visit *n*, or by a call to visit() which started even before the call to visit *n*. Since each edge and node is visited once, the algorithm runs in linear time. This depth-first-search-based algorithm is the one described by Cormen et al. (2001); it seems to have been first described in print by Tarjan (1976).