Breadth First Search (BFS) and Depth First Search (DFS)

There are two main ways to search (or visit) each node in a general graph. You have seen these before in reference to trees, but the algorithms for trees are special cases. These algorithms work for any graph.

Breadth First Search

**Algorithm 1 BFS(V, E, s)**

1: for each vertex \( u \in V \setminus \{s\} \) do  
2: \( u.d = \infty \)  
3: end for  
4: \( s.d = 0 \)  
5: \( Q = \emptyset \)  
6: Enqueue(Q, s)  
7: while \( Q \neq \emptyset \) do  
8: \( u = \text{Dequeue}(Q) \)  
9: for each vertex \( v \in G.\text{Adj}[u] \) do  
10: if \( v.d == \infty \) then  
11: \( v.d = u.d + 1 \)  
12: Enqueue(Q, v)  
13: end if  
14: end for  
15: end while
Depth First Search

**Algorithm 2 DFS(G)**

1: procedure DFS(G)
2:    for each $u \in G.V$ do
3:        $u.color$ = WHITE
4:    end for
5:    time = 0
6:    for each $u \in G.V$ do
7:        if $u.color$ == WHITE then
8:            DFS-Visit(G, $u$)
9:        end if
10:    end for
11: end procedure

12: procedure DFS-Visit(G, $u$)
13:    time = time + 1
14:    $u.d$ = time
15:    $u.color$ = GREY \quad \triangleright \text{discover } u
16:    for each $v \in G.adj[u]$ do
17:        if $v.color$ == WHITE then
18:            DFS-Visit(G, $v$)
19:        end if
20:    end for
21:    $u.color$ = BLACK
22:    time = time + 1
23:    $u.f$ = time \quad \triangleright \text{finish } u
24: end procedure
Topological Sorts

Cormen

This is Cormen’s short algorithm on page 613 to do a topological sort on a Directed Acyclic Graph. If there are any cycles in the graph a topological sort makes no sense and these algorithms might cycle forever.

Algorithm 3 Topological-Sort(DAG)
1: Let $G = DAG$
2: call DFS($G$) to compute finishing times $v.f$ for all $v \in G.V$
3: output vertices in order of decreasing finishing times

Alternate Topological Sort

As far as I know this is not in Cormen.

Algorithm 4 Alternate T-Sort(DAG)
1: Let $G = DAG$
2: Let $Q$ be an empty priority queue with indeg as the key
3: for each $v \in G.V$ do
4: \hspace{1em} ENQUEUE($Q$, $v$)
5: end for
6: while $Q$ is not empty do
7: \hspace{1em} $u = EXTRACT-MIN(Q)$
8: \hspace{2em} if $G$.indeg($u$) == 0 then
9: \hspace{3em} print $u$
10: \hspace{2em} else
11: \hspace{3em} ***ALERT*** There is a cycle in the graph! *ABORT*
12: \hspace{2em} end if
13: \hspace{1em} for each $v \in G$.adj($u$) do
14: \hspace{2em} $G = G - arc(u, v)$
15: \hspace{2em} $G$.indeg($v$) = $G$.indeg($v$) - 1
16: \hspace{1em} end for
17: end while