Depth-first search: Keep following edges from current vertex. Backtrack when no edges to unvisited vertices.

DFS tree:

\[ C \]

\[ G \] \[ \Leftrightarrow \text{vertices reachable from } C \]

\[ D \quad F \]
\[ \text{pred}[D] = G \]
\[ \text{pred}[H] = D \]

\[ \text{WHITE: unseen} \]
\[ \text{GRAY: seen but not completed} \]
\[ \text{BLACK: complete (backtracked from)} \]
DFS-VISIT(G, u)
    color[u] <- GRAY
    d[u] <- time++
    for each v adjacent to u
        if (color[v] = WHITE)
            pred[v] = u
            DFS-VISIT(G, v)
    f[u] <- time++
    color[u] <- BLACK
    DFS-VISIT(G, u)

DFS(G)
    for each u in G.V
        color[u] <- WHITE
        time <- 0
    for each u in G.V
        if color[u] = WHITE
            DFS-VISIT(G, u)

T(v) = running time of DFS-VISIT on vertex v
T(v) = ∑ 1 + degree(v)

I call to DFS-VISIT per vertex |V| calls total
each call looks at all adj. vert
eventually all edges examined
Θ(V + E)
Topological Sort

For a directed, acyclic graph $G$, ordering of vertices so edges go from earlier verts in ordering to later verts

![Diagram of a directed graph]

Algorithm:

Do DFS and insert into list as they finish
return verts in order of decreasing finish time

$\Theta(V+E)$

$O(V \log V)$ (sort by $f$)

$O(V \log V + E)$

or use counting sort
$
\Theta(V)$ sort

$\Theta(V+E)$ total