Topological Sort

\((V, E)\) is a directed graph. A cycle is a sequence of vertices \(v_0, v_1, \ldots, v_m, v_i \in V\) and \(m \geq 1\) so that \((v_i, v_{i+1}) \in E\) for \(0 \leq i \leq m-1\), and \((v_m, v_0) \in E\).

\((V, E)\) is a directed acyclic graph (DAG), if it has no cycle.

To produce a topological sort of \((V, E)\) we need to order vertices that is compatible with \(E\). Formally, a sequence of vertices \(v_1, v_2, \ldots, v_n\) is a topological sort so that \(i < j\) if \((v_i, v_j) \in E\).

In other words, \(i > j\) only if \((v_i, v_j) \in E\).

Ex.

\[a, b, h, i, g, f, c, d, e\] is a topological sort.
A vertex with no incoming edge is a source, with no outgoing edge is a sink.

Claim \((V, E)\) is acyclic if and only if DFS search yields no back edge.

The idea is to use DFS to produce a topological sort. We report a vertex when it is finished. It produces a topological sort back to front. It is justified by the following.

Claim If \((u, v) \in E\) then \(V[u].f > V[v].f\).

Proof Consider the time \((u, v)\) is explored by DFS.

Case 1. \(v\) is an ancestor of \(u\) (contradiction to no back edge claim)

Case 2. \(v\) is yet unexplored.
Then \(v\) becomes descendant of \(u\).
So, \(V[u].f > V[v].f\)

Case 3. \(v\) is already finished.
Then \(V[u].f > V[v].f\) automatically