DAG: directed acyclic graph

no (directed) cycles

not a DAG
$V_1 \xrightarrow{} V_2 \xrightarrow{} V_3$

no info about $V_2$ vs $V_3$

DFS tree from $V_1$

"flattened" & pointing L $\rightarrow$ R
Topological Sort (on a DAG)

DFS tree from $V_1$

"flattened" & pointing L $\rightarrow$ R

"Sort" all vertices (place in line) s.t. all directed edges are $\rightarrow$

$V_1 \rightarrow V_2 \rightarrow V_3$ no info about $V_2$ vs $V_3$
Topological Sort
(on a DAG)

DFS tree from $V_1$

"flattened" & pointing $L \rightarrow R$

"Sort" all vertices (place in line)
  s.t. all directed edges are $\rightarrow$

$V_1 \rightarrow V_2 \rightarrow V_3$
no info about $V_2$ vs $V_3$

$V_4 \rightarrow V_5 \rightarrow V_1 \rightarrow V_3 \rightarrow V_2$

implied
Topological sort (on a DAG)

DFS tree from $V_1$

"flattened" & pointing L→R

"Sort" all vertices (place in line) s.t. all directed edges are $\rightarrow$

$V_1 \rightarrow V_2 \rightarrow V_3$ no info about $V_2$ vs $V_3$

$V_4 \rightarrow V_1 \rightarrow V_3 \rightarrow V_2 \rightarrow V_5$

implied
$V_1 \leftarrow V_2 \rightarrow V_3$

tree from $V_1$

no info about $V_2$ vs $V_3$

notice, we visited $V_2$ before $V_3$

Otherwise
We need this order: \( V_1 \rightarrow V_3 \rightarrow V_2 \)

no info about \( v_2 \) vs \( v_3 \)

tree from \( v_1 \)

notice, we visited \( v_2 \) before \( v_3 \)

Otherwise
Rule: sort/output by finish time.

- \(v_2\) finishes first. Then \(v_3\). Then \(v_1\).

- No info about \(v_2\) vs \(v_3\).

- Notice, we visited \(v_2\) before \(v_3\).

Otherwise:

- Tree from \(v_1\).
OR

\[
\begin{align*}
V_1 & \xrightarrow{\text{search } V_4, V_5} V_2 \\
V_4 & \xrightarrow{\text{search } V_4, V_5} V_5
\end{align*}
\]
OR

\[ V_1 \xrightarrow{V_3} V_2 \text{ group 1} \]

\[ \text{Continue DFS: search } v_4, v_5 \]

\[ v_4 \rightarrow v_5 \text{ group 2} \]

group 1 finished before group 2
OR

\[
\begin{align*}
V_1 & \xrightarrow{V_3} V_2 & \text{group 1} \\
\text{Continue DFS: search } V_4, V_5 \\
V_4 & \xrightarrow{} V_5 & \text{group 2}
\end{align*}
\]

\text{group 1 finished before group 2, so: } V_4 \xrightarrow{} V_5 \quad V_1 \xleftarrow{} V_3 \xrightarrow{} V_2
OR

\[ V_1 \rightarrow V_3 \rightarrow V_2 \quad \text{group 1} \]

\[ V_1 \rightarrow V_3 \rightarrow V_2 \]

\[ V_4 \rightarrow V_5 \quad \text{group 2} \]

Continue DFS: search \( v_4, v_5 \)

\[ V_4 \rightarrow V_5 \]

\( V_4 \) finished before group 2, so:

\[ V_4 \rightarrow V_5 \rightarrow V_1 \leftarrow V_3 \rightarrow V_2 \]

Can add these; \( V_4 \) found \( v_3, v_1 \), but they were marked.
We could have had other groups or DFS trees, but each would give a valid topological sort.

Continue DFS: search \( v_4, v_5 \)

\( v_4 \rightarrow v_5 \) group 2

\( v_1 \leftarrow v_3 \rightarrow v_2 \) group 1

\( v_4 \rightarrow v_5 \)

\( v_1 \leftarrow v_3 \rightarrow v_2 \)

Group 1 finished before group 2, so:
Intuition

DFS tree from \( v \).
Intuition

DFS tree from $v$.

If there is an $x \rightarrow y$ relation not in tree, then $x$ was explored after $y$. So $y$ finished first & we output correctly.
Intuition

(another example)

DFS tree from v.

If there is an x→y relation not in tree, then x was explored after y.

So y finished first & we output correctly.
Intuition

DFS tree from $v$.

$x \rightarrow y$ is implied in the tree.

$y$ explored after $x$.

So $y$ finished first, and we output correctly.
Intuition

(possible)

DFS tree from v.
Intuition

DFS tree from v.

similar... if search started at v then it will be done before it gets discovered
Intuition

DFS tree from \( v \).

Conclusion: if \( x \leadsto \leadsto \leadsto y \) then \( y \) finishes before \( x \).

Similar... if search started at \( v \) then it will be done before it gets discovered.
Summary

Run DFS in any order.
Run DFS in any order.

When a vertex \( v \) has been processed entirely, add it to a list.
Summary

Run DFS in any order.

When a vertex $v$ has been processed entirely, add it to a list.

Every vertex reachable from $v$ will be done before $v$, so it will already be in the list.
Run DFS in any order.

When a vertex $v$ has been processed entirely, add it to a list.

Every vertex reachable from $v$ will be done before $v$, so it will already be in the list.

Similarly, $v$ will be ahead of any vertex that can reach it.