DAG: directed acyclic graph

Left: no (directed) cycles

Right: not a DAG
Topological Sort (on a DAG)

"Sort" all vertices (place in line) s.t. all directed edges are implied.
Topological Sort (on a DAG)

DFS tree from $V_1$: $V_1 \rightarrow V_2 \rightarrow V_3$

directly gives us some info, but...

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

$V_4 \rightarrow V_1 \rightarrow V_3 \rightarrow V_2 \rightarrow V_5$
DFS tree from $V_1$:

$V_1 \rightarrow V_2 \rightarrow V_3$

→ directly gives us some info, but...

Notice, we visited $V_2$ before $V_3$

Otherwise

Rule: sort/output by finish time.

$V_2$ finishes first. Then $V_3$. Then $V_1$
group 1 finished before group 2, so:

We could have had other groups or DFS trees, but each would give a valid topological sort.

---

OR

\[
\begin{align*}
&V_1 &\rightarrow &V_3 &\rightarrow &V_2 &\quad \text{group 1} \\
&V_4 &\rightarrow &V_5 &\quad \text{group 2} \\
&\text{Continue DFS: search } v_4, v_5
\end{align*}
\]

---

We could have added these; \( v_4 \) found \( v_3, v_1 \), but they were marked.
If $x \rightarrow y$ is implied in a DFS tree then $y$ was explored after $x$. So $y$ finished first.

If $x \rightarrow z$ not in tree & not implied, then $x$ was explored after $z$. (subtree) (otherwise we would have $x \rightarrow z$) So $z$ finished first.

In both cases, we output correctly.
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.