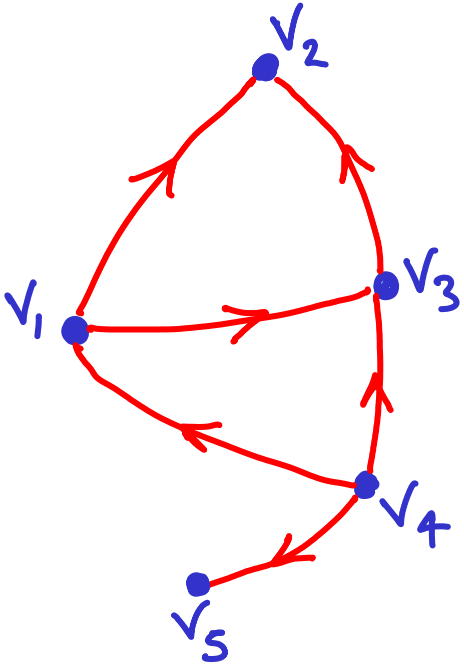
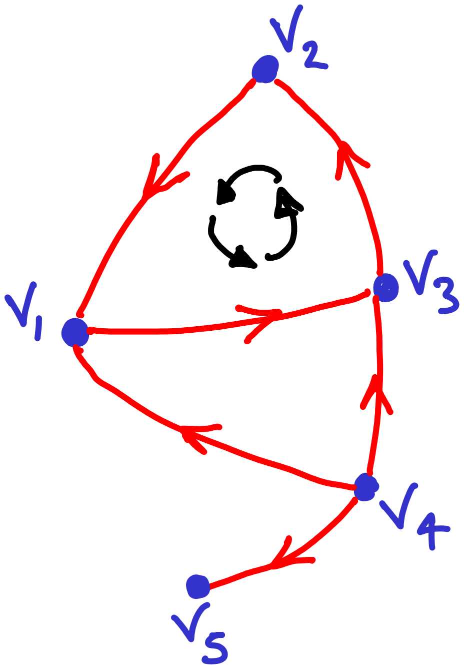
DAG : directed acyclic graph



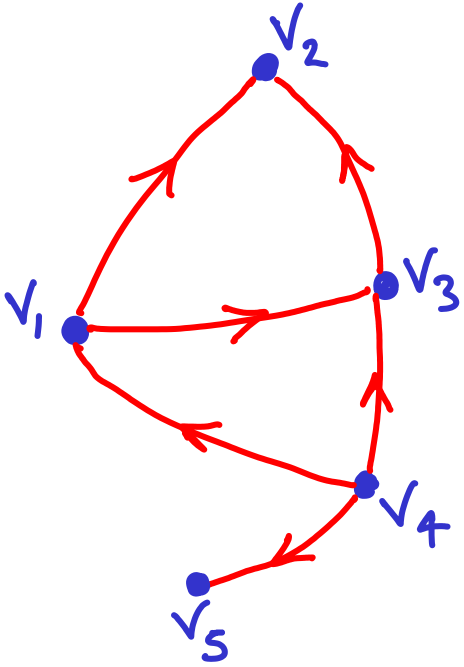
no (directed) cycles

✓



not a DAG

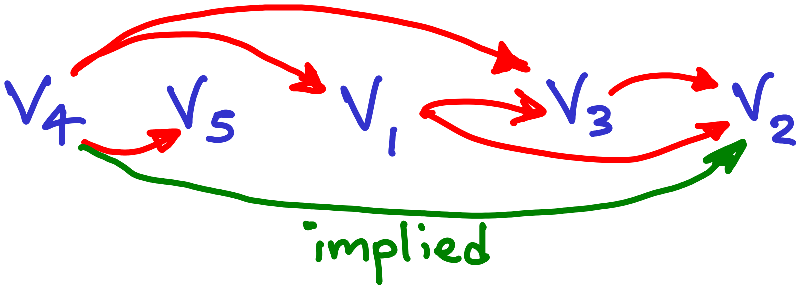
✗

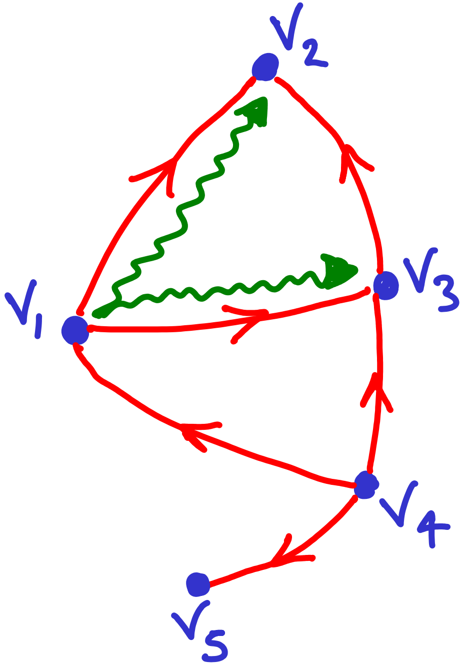


TOPOLOGICAL SORT

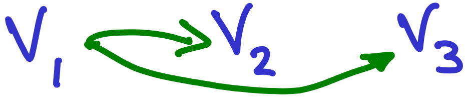
(on a DAG)

"Sort" all vertices (place in line) s.t. all directed edges are →

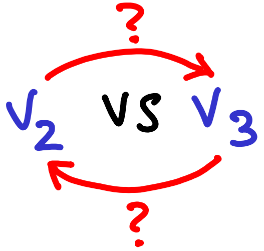




DFS tree from v_1 :



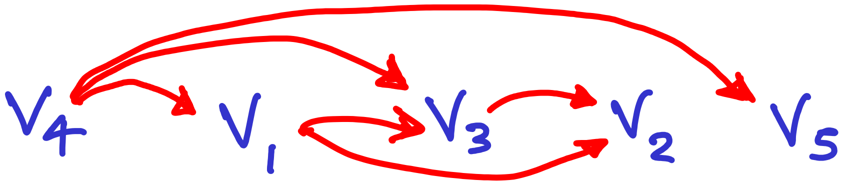
↳ directly gives us some info, but...



TOPOLOGICAL SORT
(on a DAG)

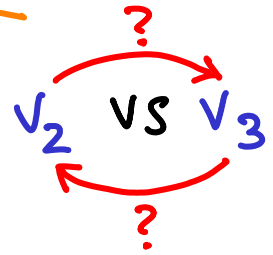
"Sort" all vertices (place in line)

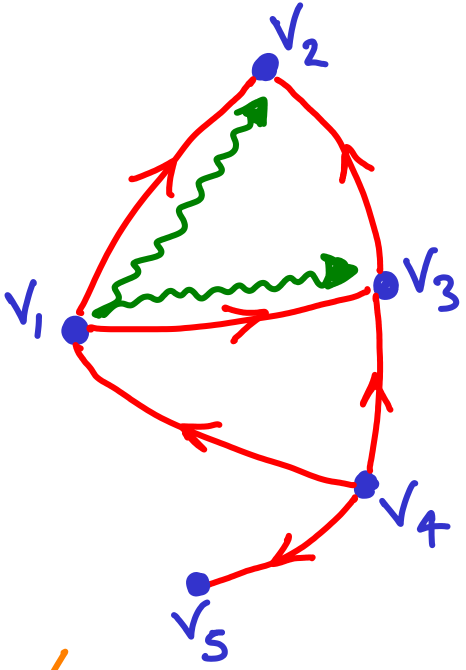
s.t. all directed edges are →



DFS tree from v_1 : ~~$v_1 \to v_2 \to v_3$~~

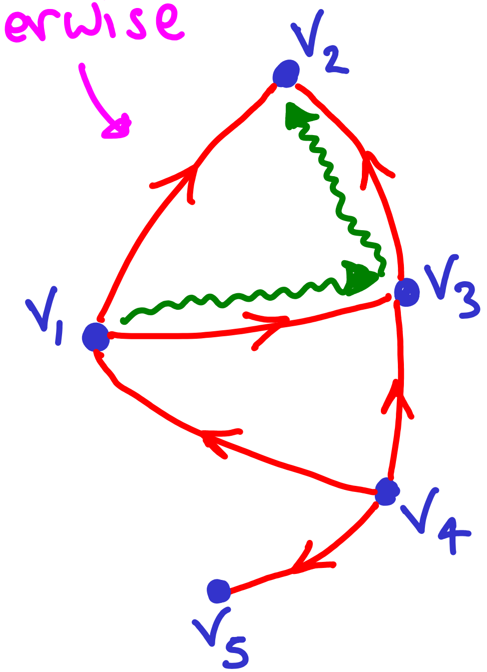
↳ directly gives us some info, but...





notice, we visited v_2 before v_3

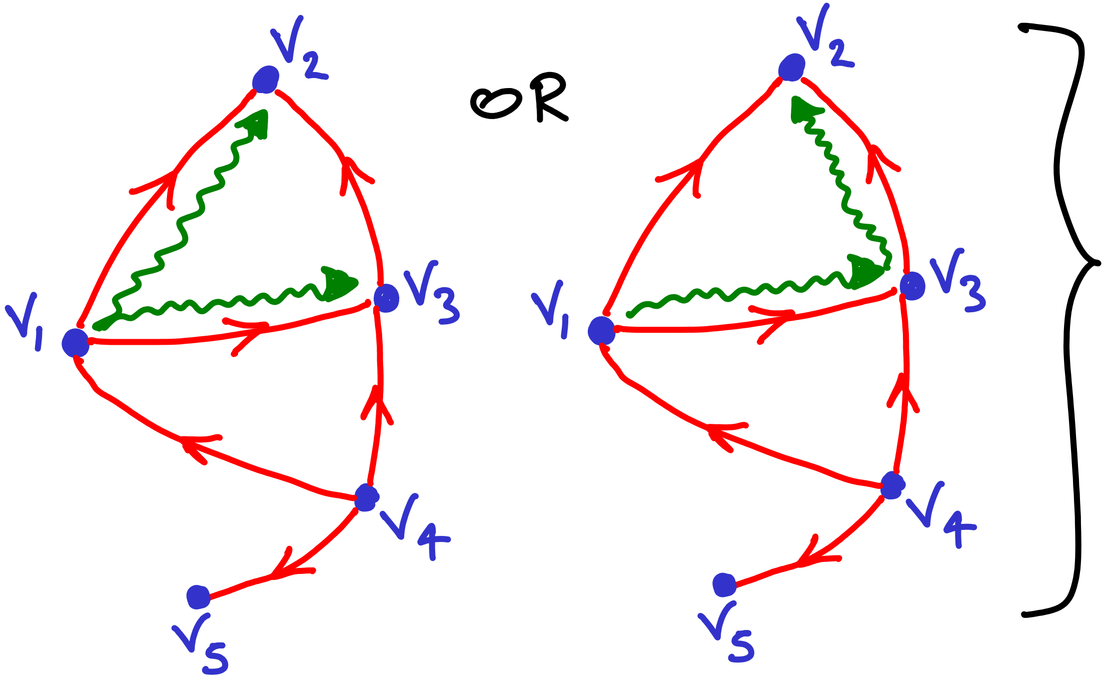
↳ Otherwise

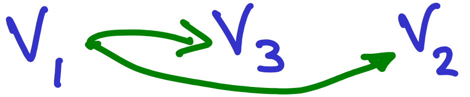


Rule : sort/output by finish time.

v_2 finishes first. Then v_3. Then v_1

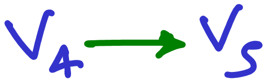
$v_1 \to v_3 \to v_2$





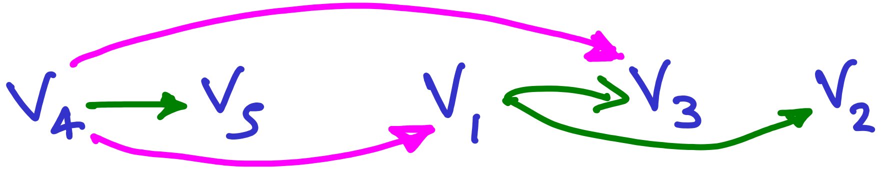
group 1

Continue DFS : search v_4, v_5



group 2

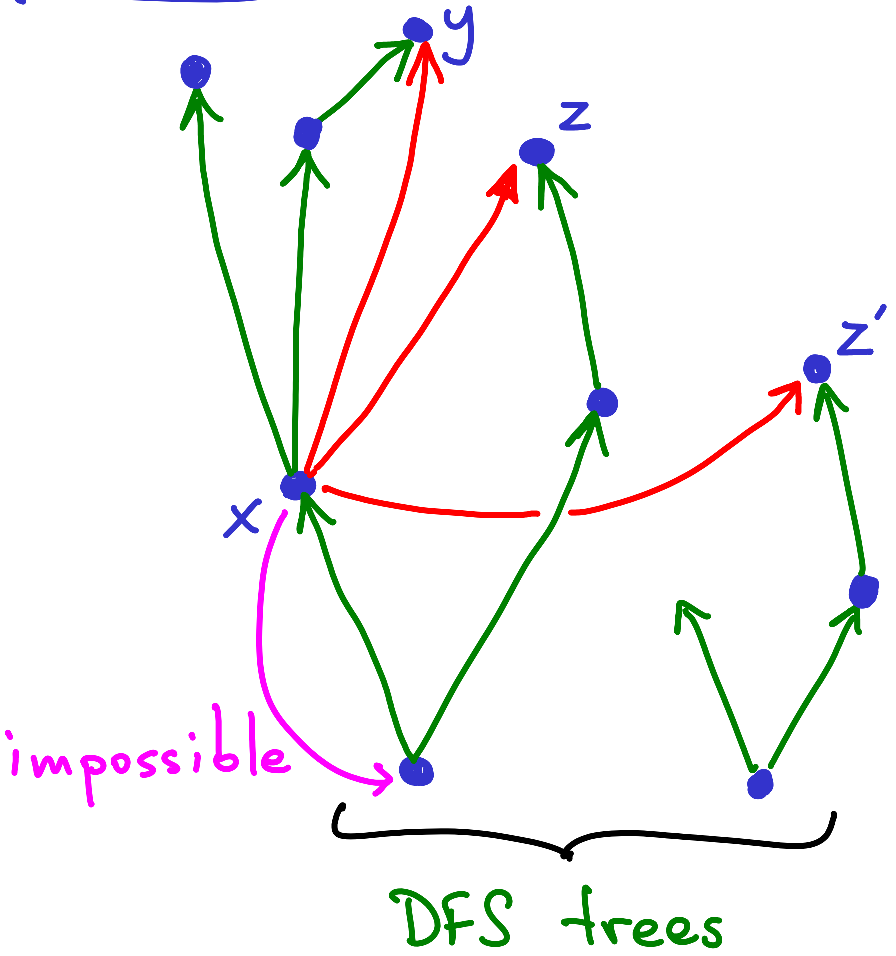
group 1 finished before group 2, so :



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

Intuition



If $x \to y$ is implied in a DFS tree then y was explored after x.

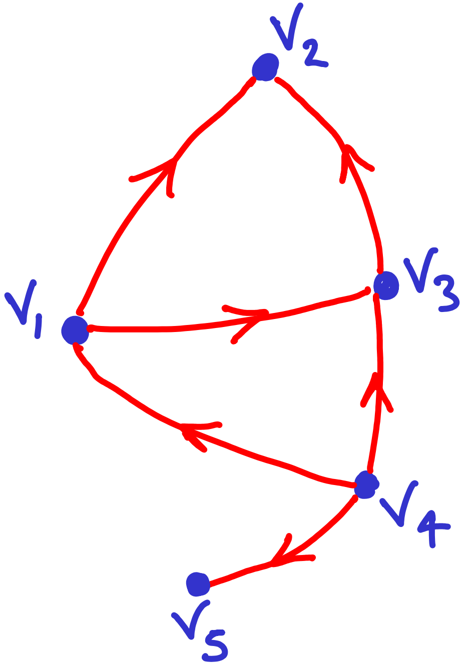
So y finished first

If $x \to z$ not in tree & not implied, then x (subtree) was explored after z (subtree).

(otherwise we would have $x \to z$)

So z finished first

In both cases, we output correctly



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

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