DAG : directed acyclic graph
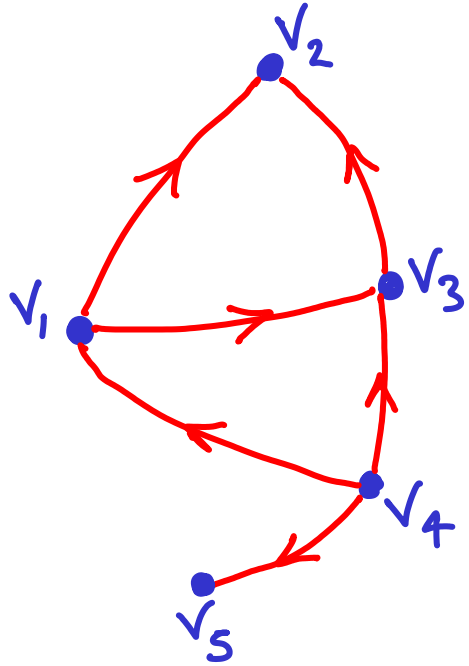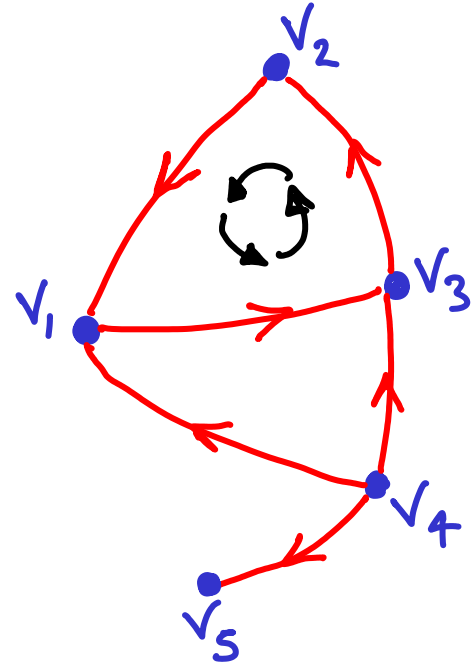

no (directed) cycles
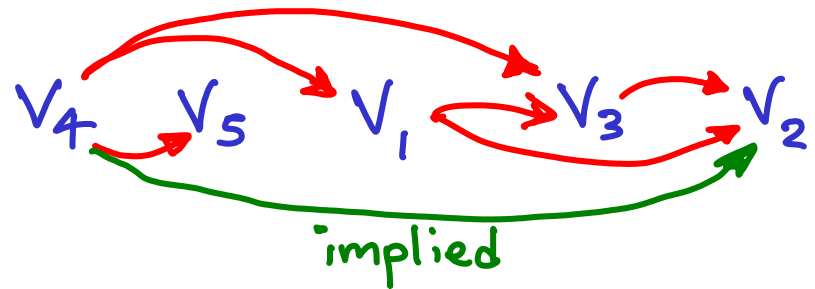
✓

not a DAG
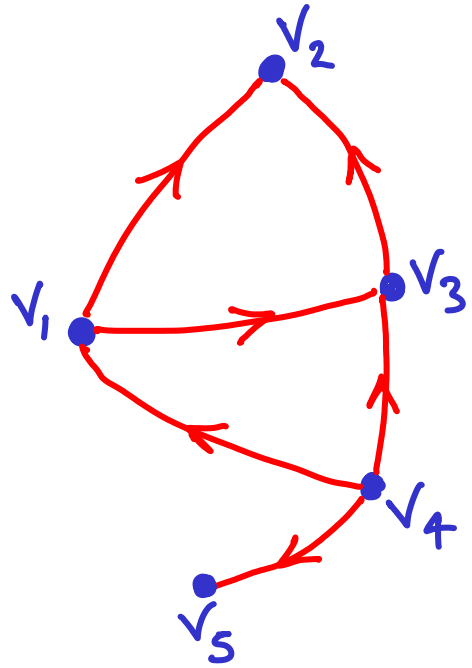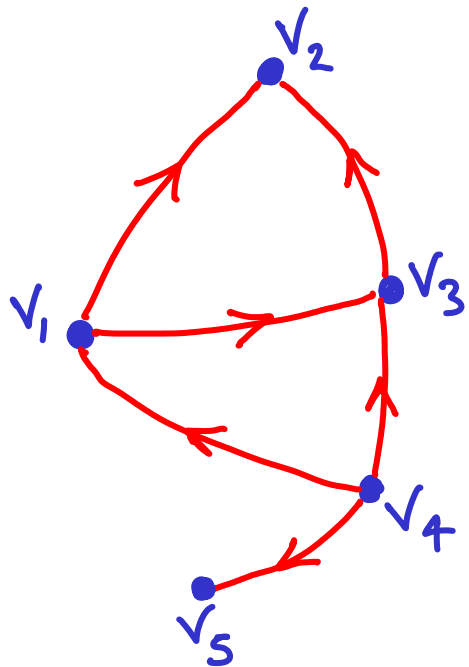
✗

V_2

V_3

V_1

V_4

V_5

TOPOLOGICAL SORT

(on a DAG)

"Sort" all vertices (place in line)

s.t. all directed edges are $\rightarrow$

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

implied

TOPOLOGICAL SORT

(on a DAG)

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

$V_4 \quad V_1 \quad V_3 \quad V_2 \quad V_5$

implied

DFS tree from v_1: $v_1 \rightleftarrows v_2 \rightarrow v_3$
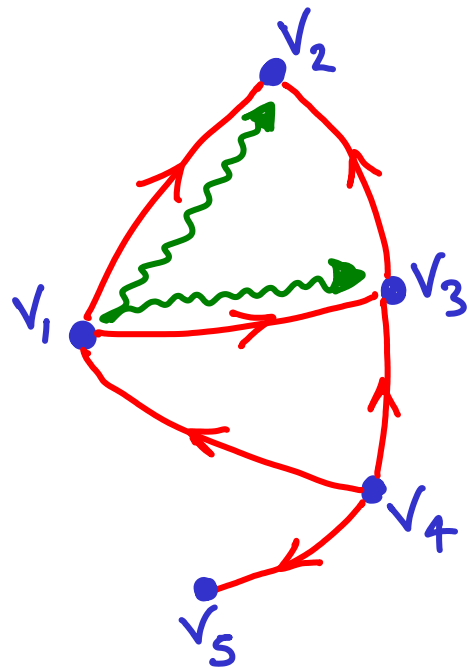
↳ directly gives us some info, but... v_2 vs v_3 ?

TOPOLOGICAL SORT

(on a DAG)

"Sort" all vertices (place in line)

s.t. all directed edges are →

$v_4 \rightarrow v_1 \rightarrow v_3 \rightarrow v_2 \rightarrow v_5$
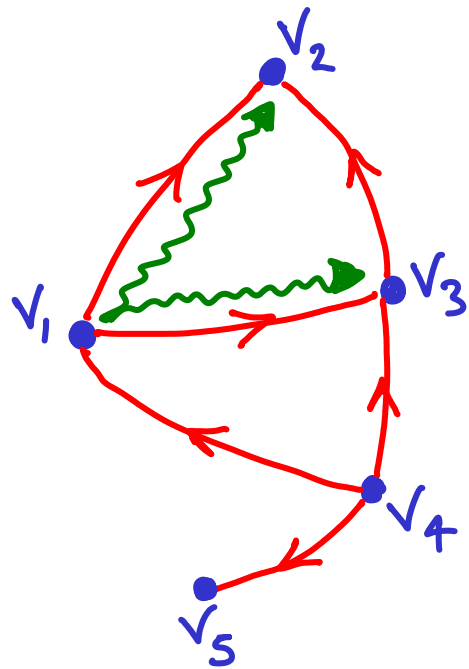
DFS tree from v_1: $v_1 \rightarrow v_2 \rightarrow v_3$

↳ directly gives us some info, but... v_2 vs v_3

? ?
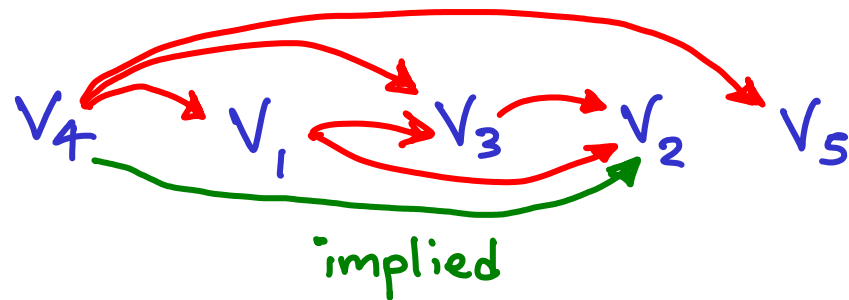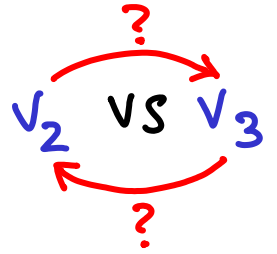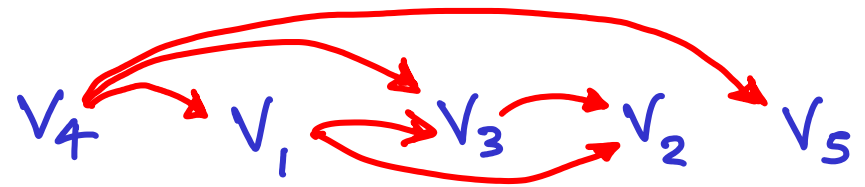
notice, we visited v_2 before v_3

↳ Otherwise

DFS tree from V_1 :   ~~$V_1 \longrightarrow V_2 \longrightarrow V_3$~~

$\hookrightarrow$ directly gives us some info, but... V_2 vs V_3 ?

notice, we visited V_2 before V_3

$\hookrightarrow$ Otherwise

We need this order:
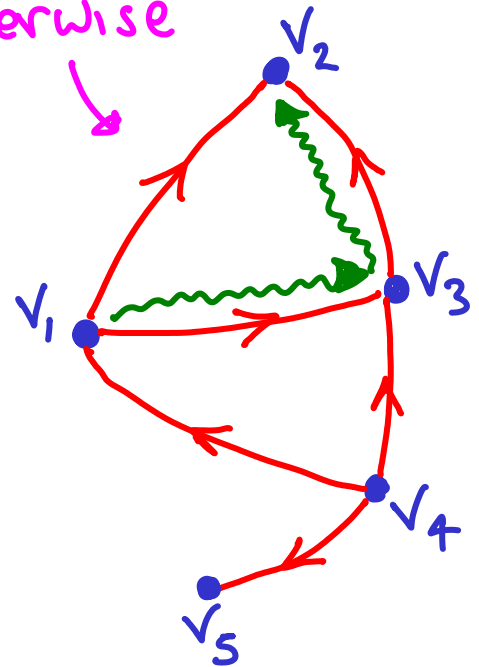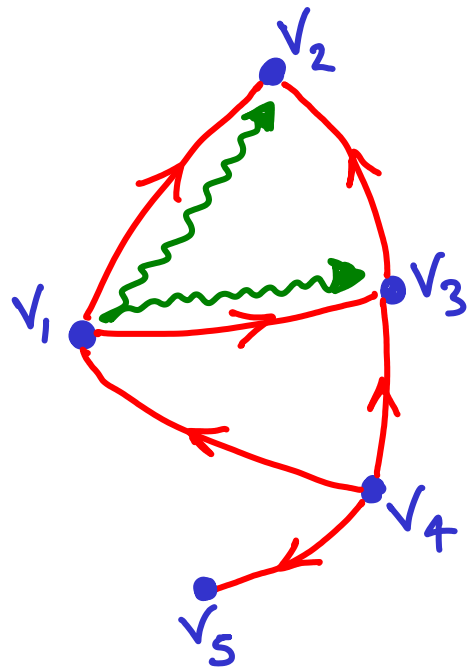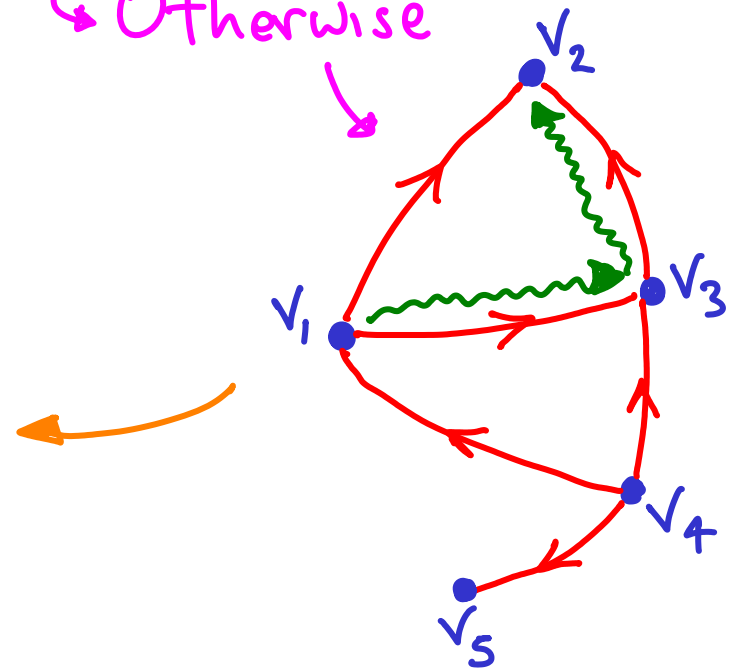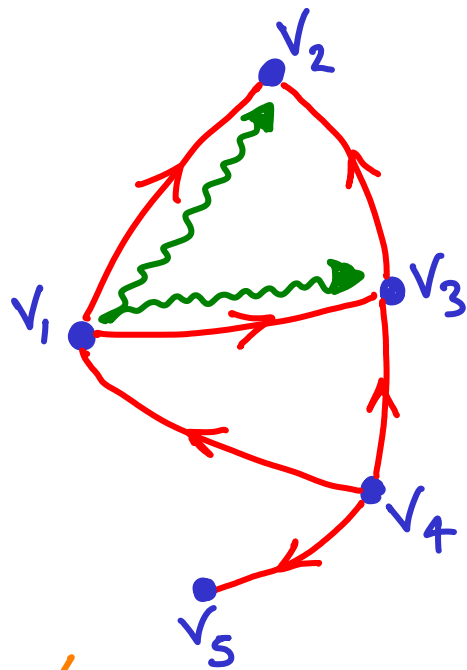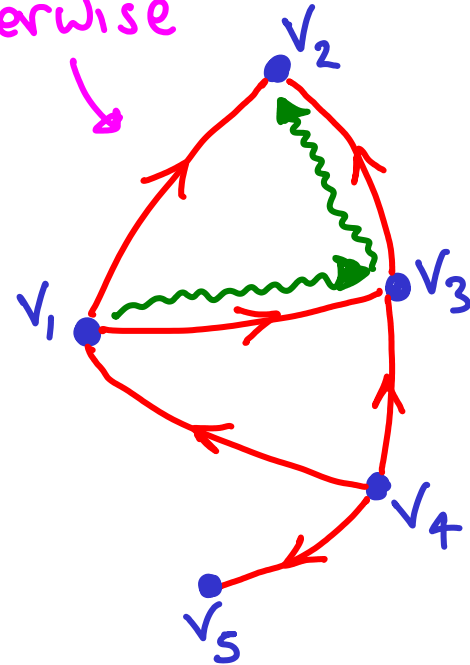
$V_1 \longrightarrow V_3 \longrightarrow V_2$

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

↳ directly gives us some info, but... v_2 vs v_3 ? ?
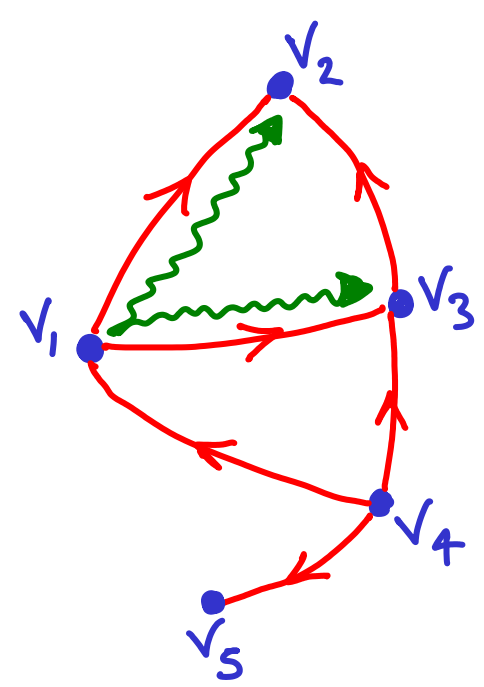
notice, we visited v_2 before v_3

↳ Otherwise

Rule : sort/output by finish time.
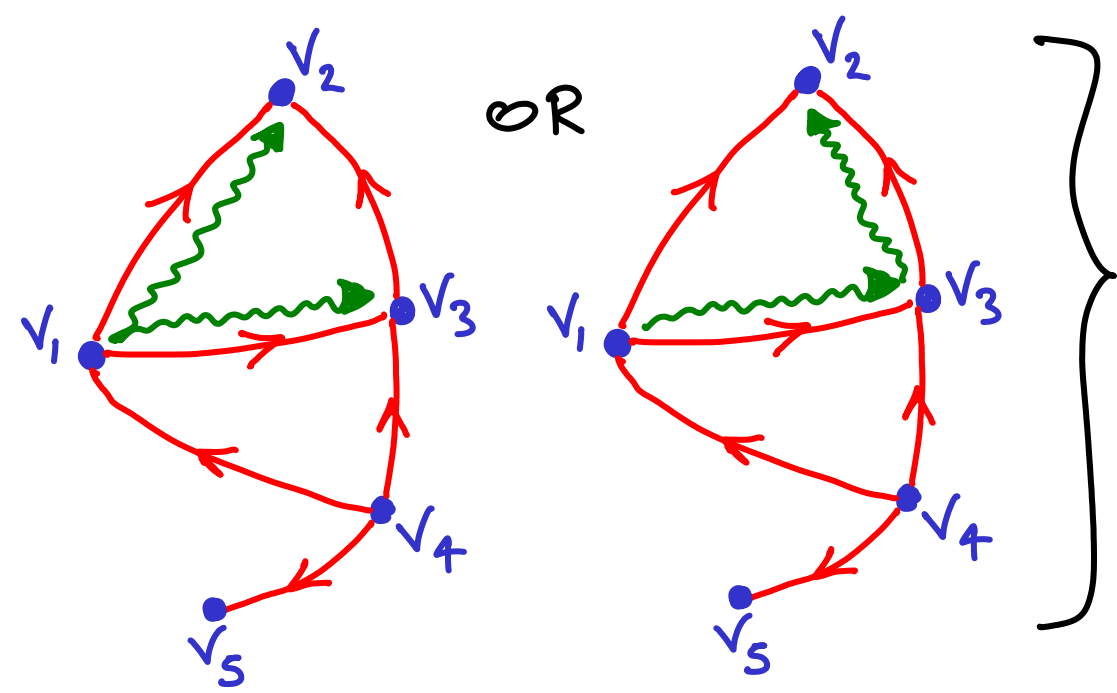v_2 finishes first. Then v_3. Then v_1
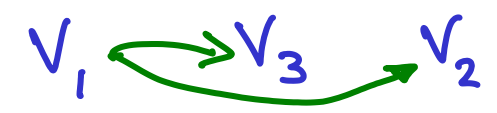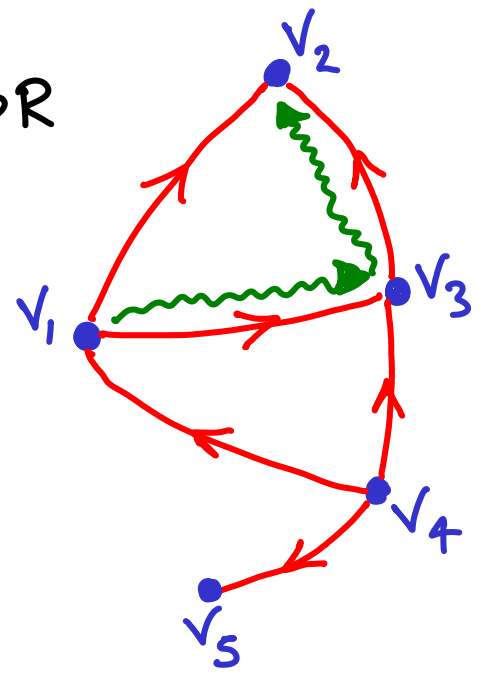
$v_1 \Rightarrow v_3 \to v_2$

OR

$V_1 \longrightarrow V_3 \longrightarrow V_2$

OR

$V_1 \rightarrow V_3 \rightarrow V_2$

Continue DFS : search V_4, V_5

$V_4 \rightarrow V_5$

OR

group 1 finished before group 2

$V_1 \Rightarrow V_3 \rightarrow V_2$ group 1

Continue DFS : search V_4, V_5

$V_4 \rightarrow V_5$ group 2

OR

$V_1 \rightrightarrows V_3 \rightarrow V_2$ group 1

Continue DFS : search V_4, V_5

$V_4 \rightarrow V_5$ group 2

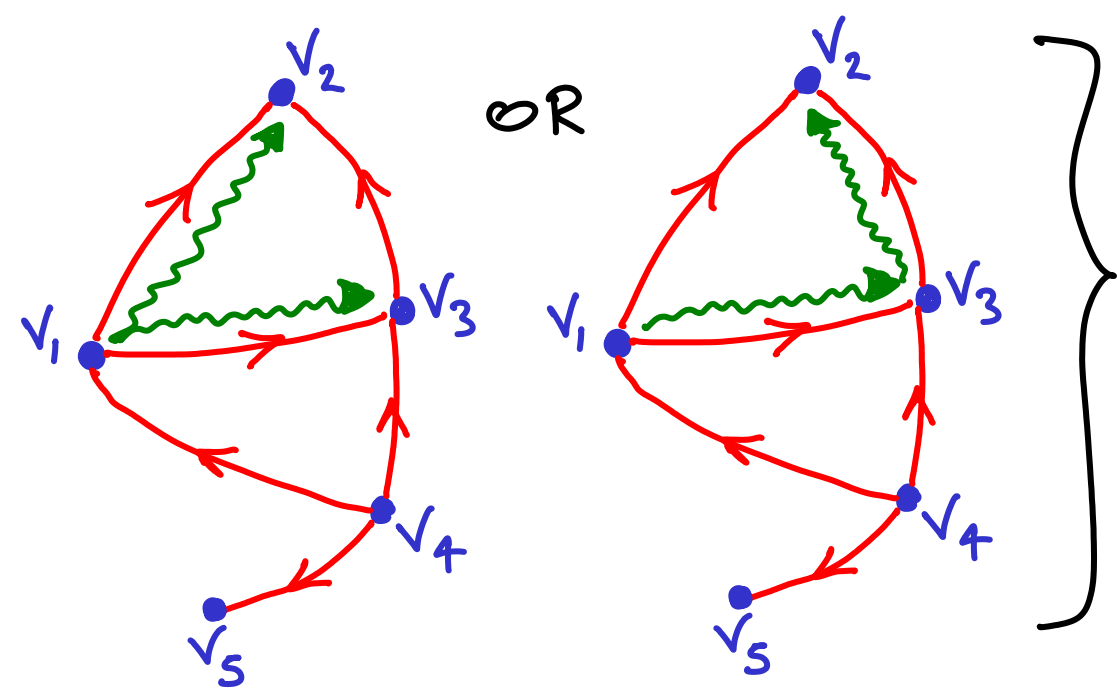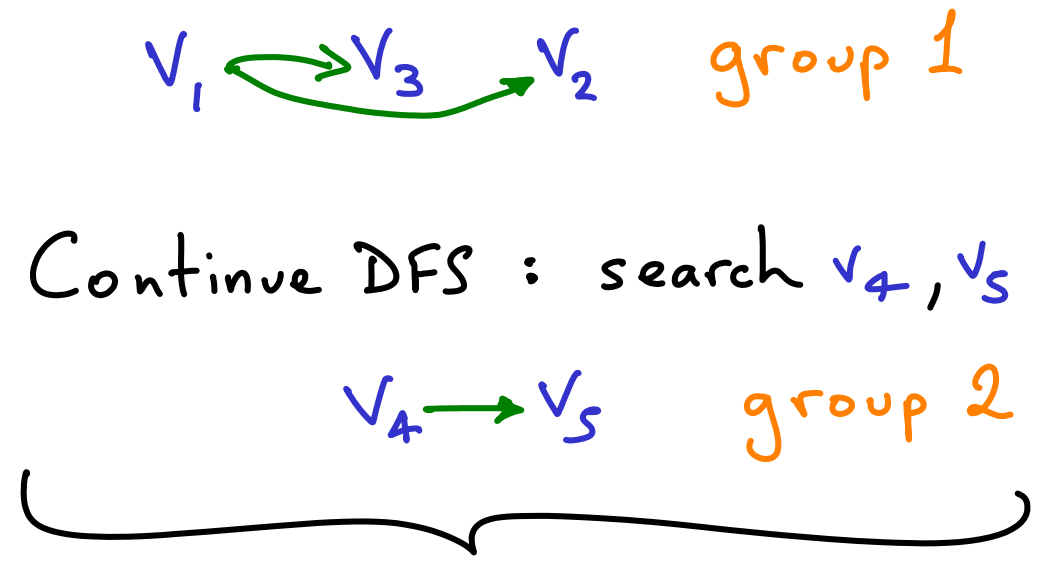group 1 finished before group 2, so : $V_4 \rightarrow V_5$ $V_1 \rightrightarrows V_3 \rightarrow V_2$

OR

$V_1 \rightrightarrows V_3 \rightarrow V_2$ group 1

Continue DFS : search V_4, V_5

$V_4 \rightarrow V_5$ group 2

group 1 finished before group 2, so : $V_4 \rightarrow V_5 \quad V_1 \rightrightarrows V_3 \rightarrow V_2$

can add these; V_4 found V_3, V_1
but they were marked

OR

V_2 V_1 V_3 V_4 V_5
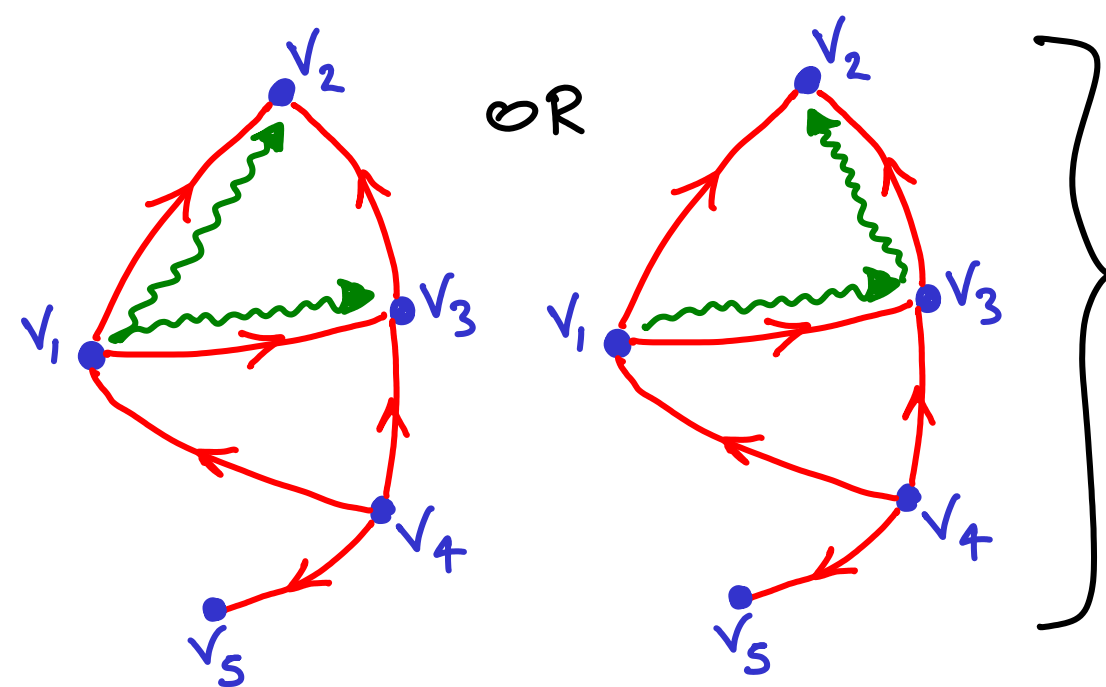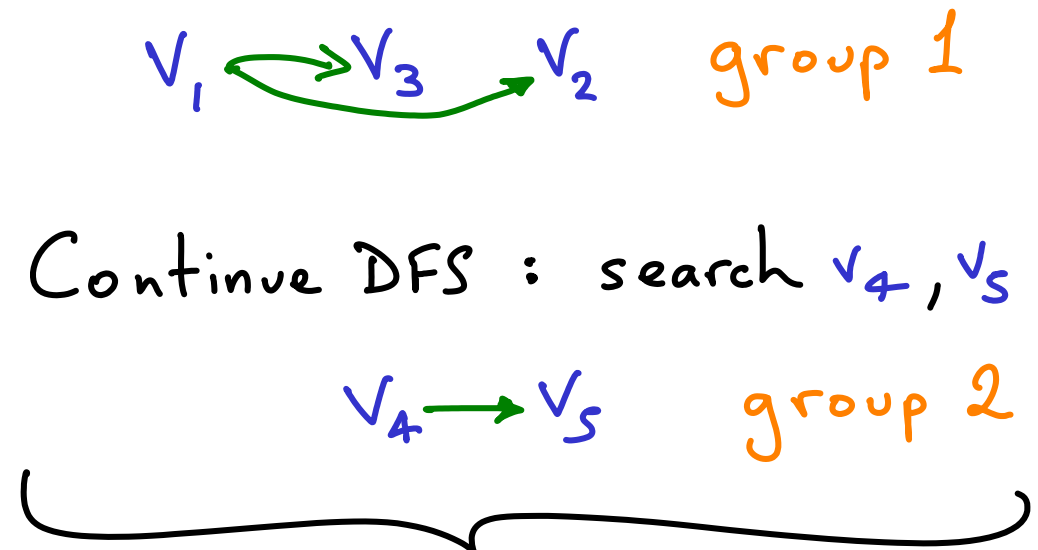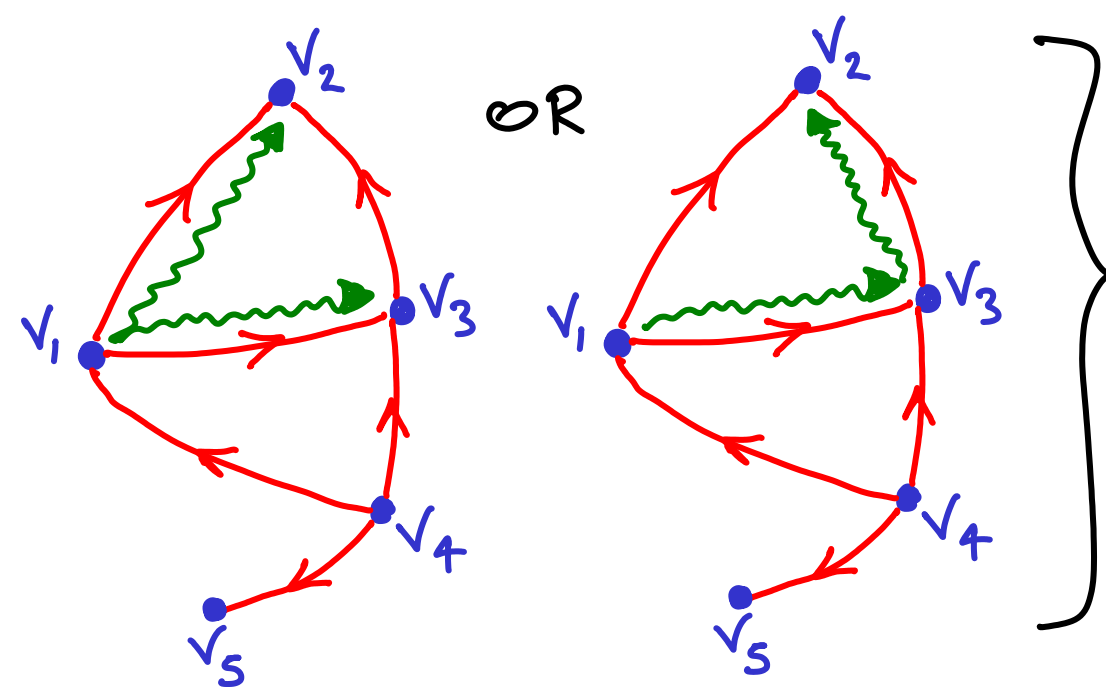
$V_1 \longrightarrow V_3 \rightarrow V_2$ group 1

Continue DFS : search V_4, V_5

$V_4 \longrightarrow V_5$ group 2
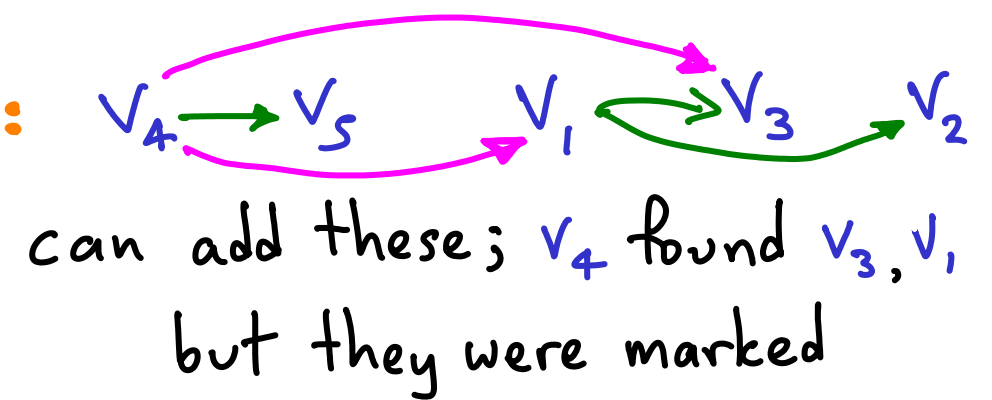
group 1 finished before group 2, so :

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

can add these; V_4 found V_3, V_1
but they were marked
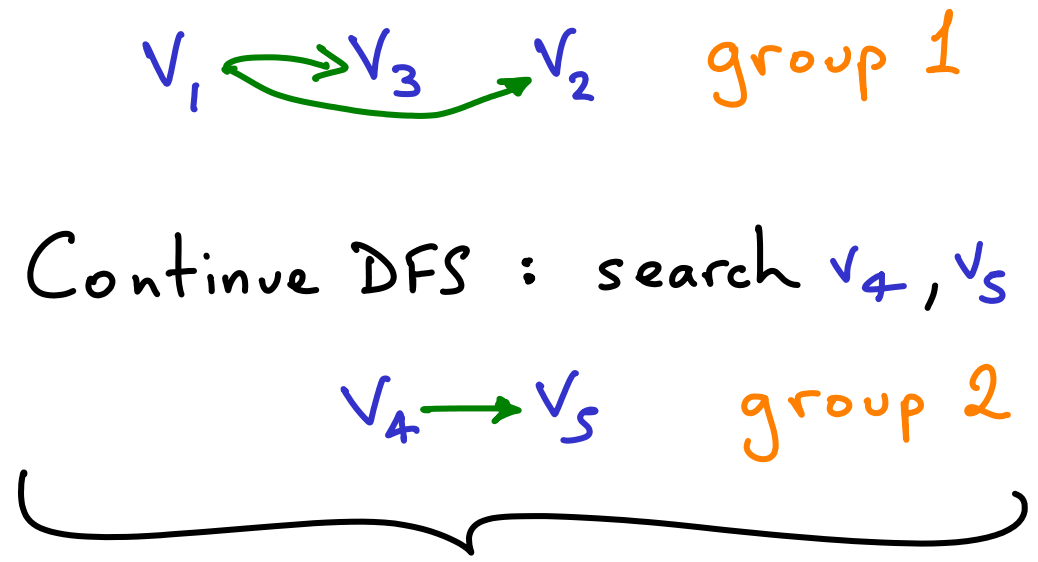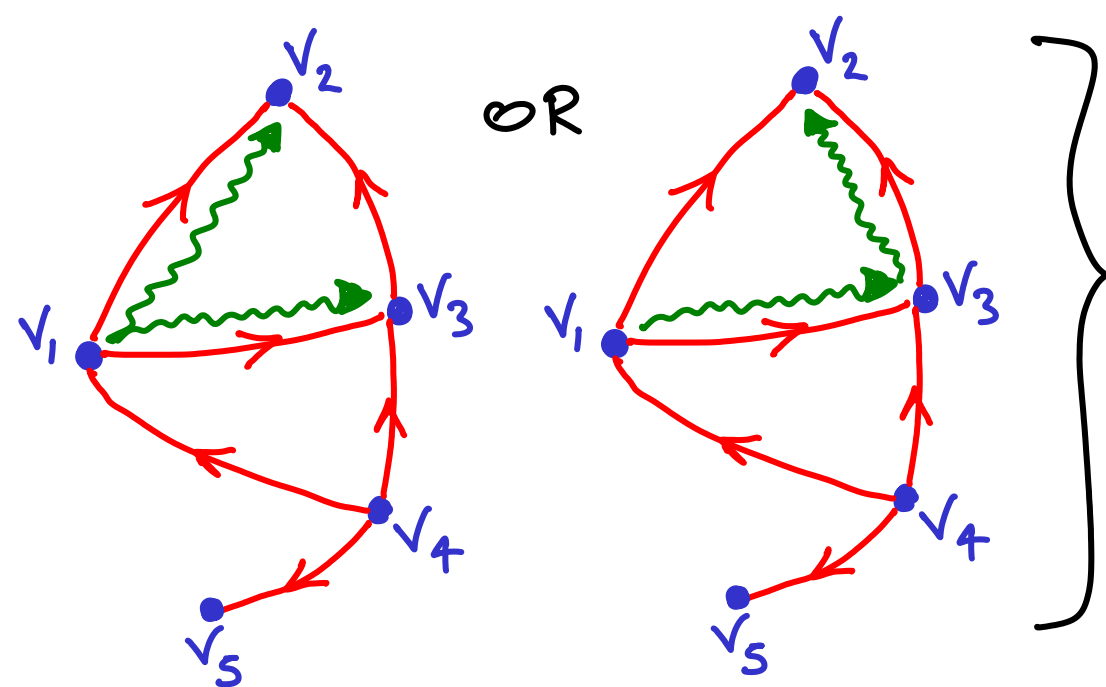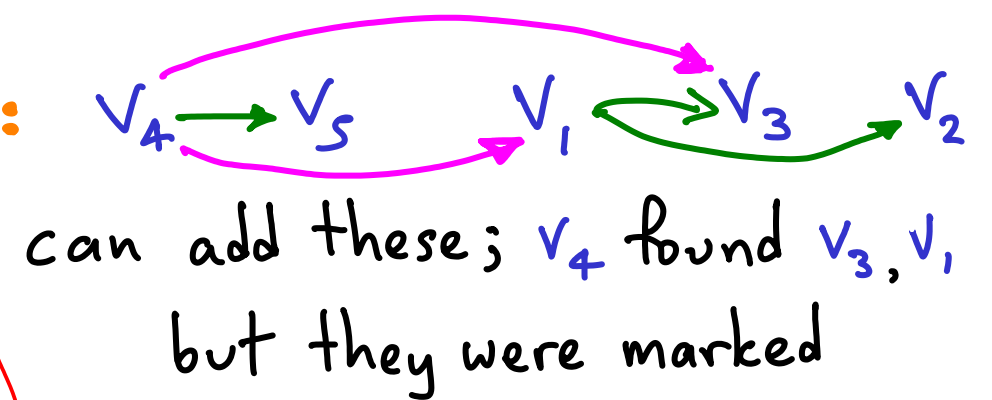
We could have had other groups
or DFS trees, but each would
give a valid topological sort

Intuition
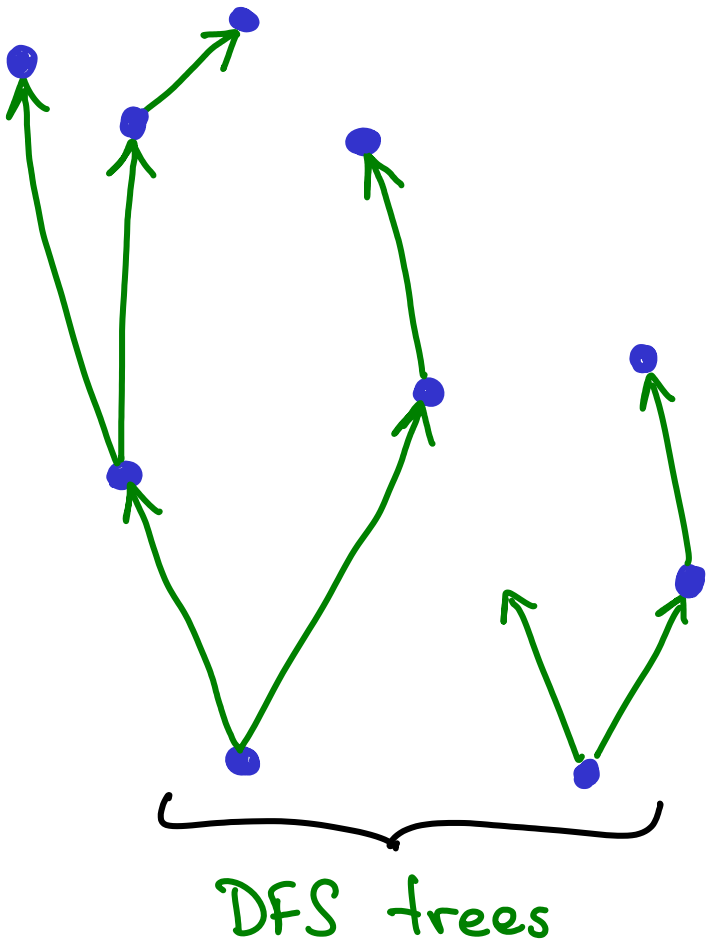
DFS trees

Intuition

impossible
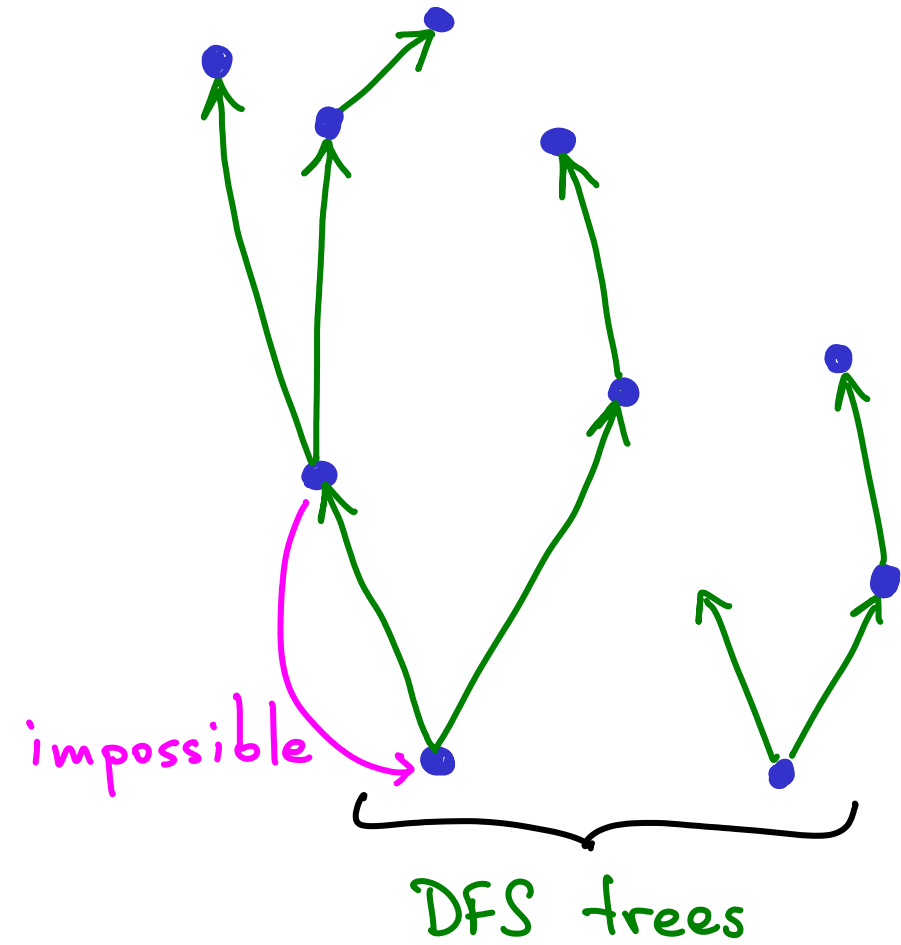
DFS trees

Intuition

DFS trees

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

So y finished first

Intuition

DFS trees

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

So y finished first

we output correctly

Intuition

DFS trees

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

So y finished first
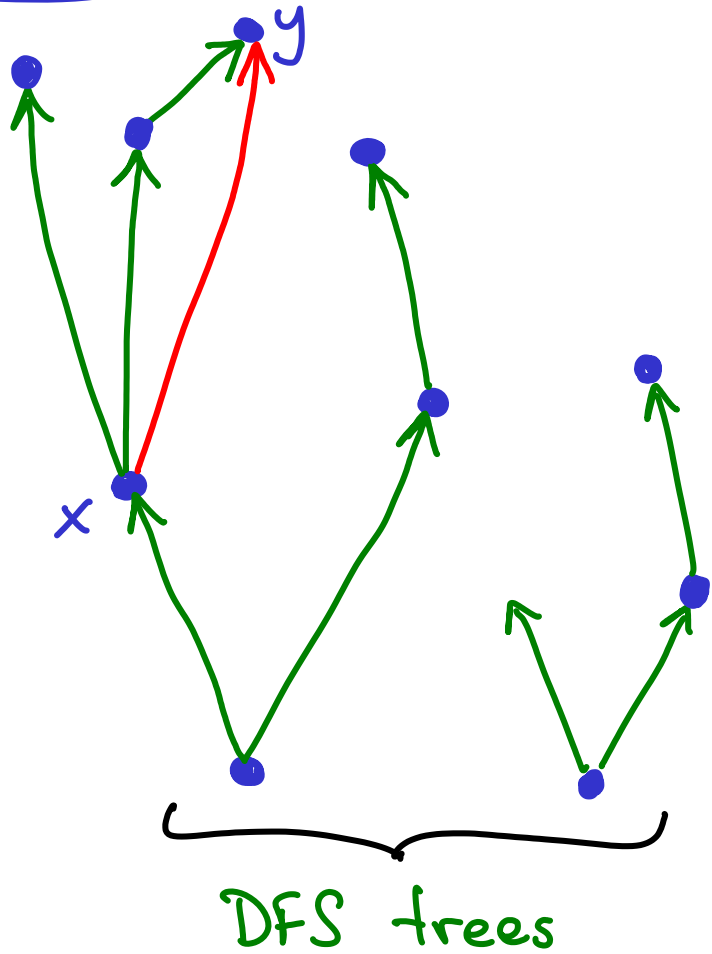
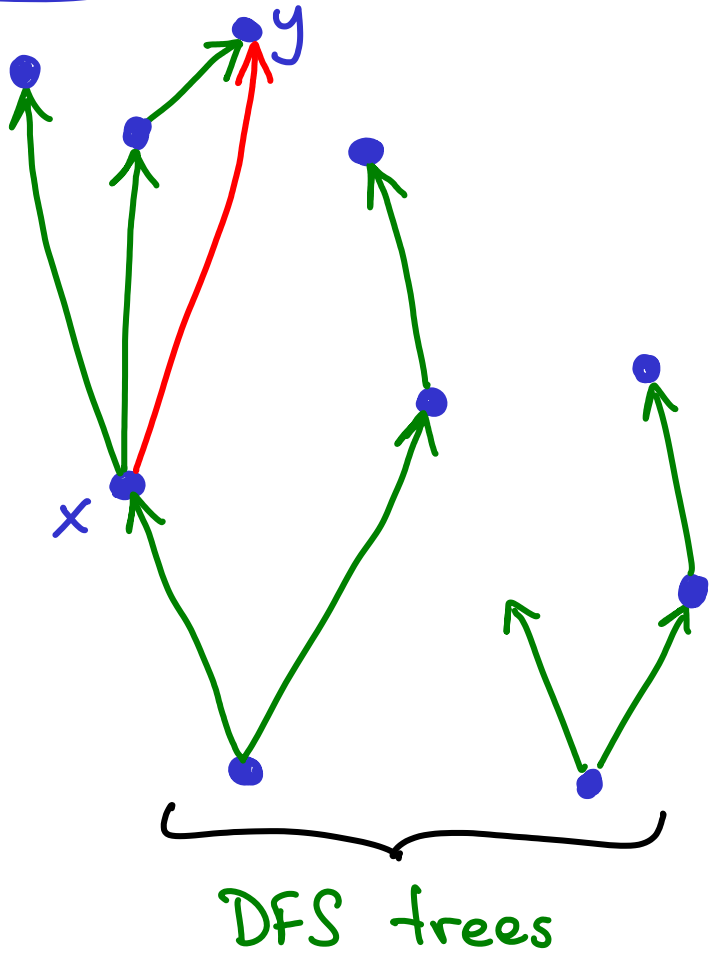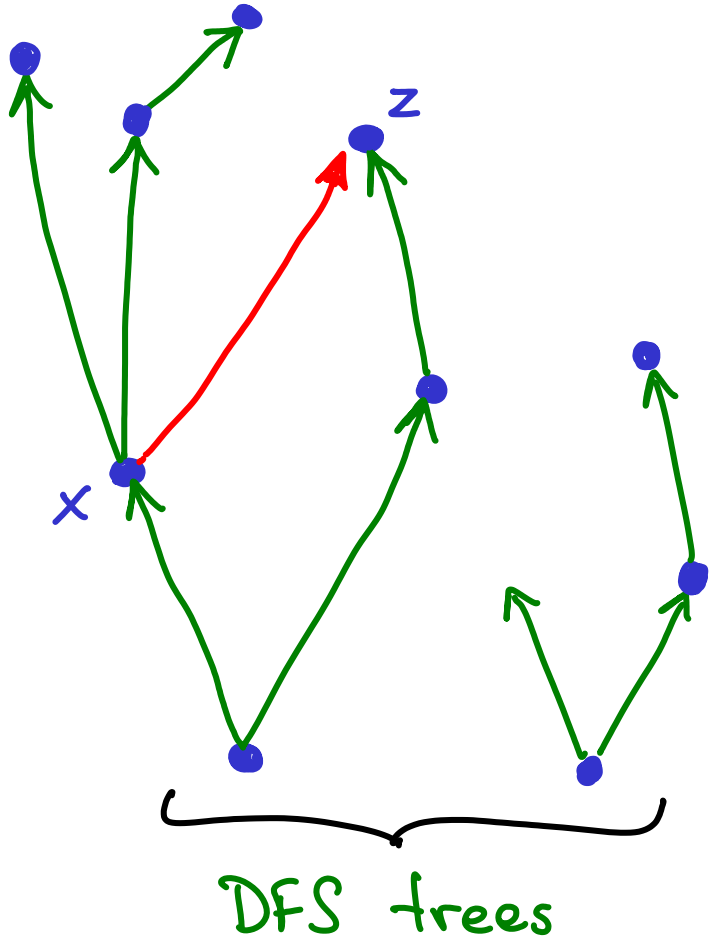If $x \longrightarrow z$ not in tree & not implied, then ?

Intuition

DFS trees

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)   (subtree)

(otherwise we would have $x \longrightarrow z$)
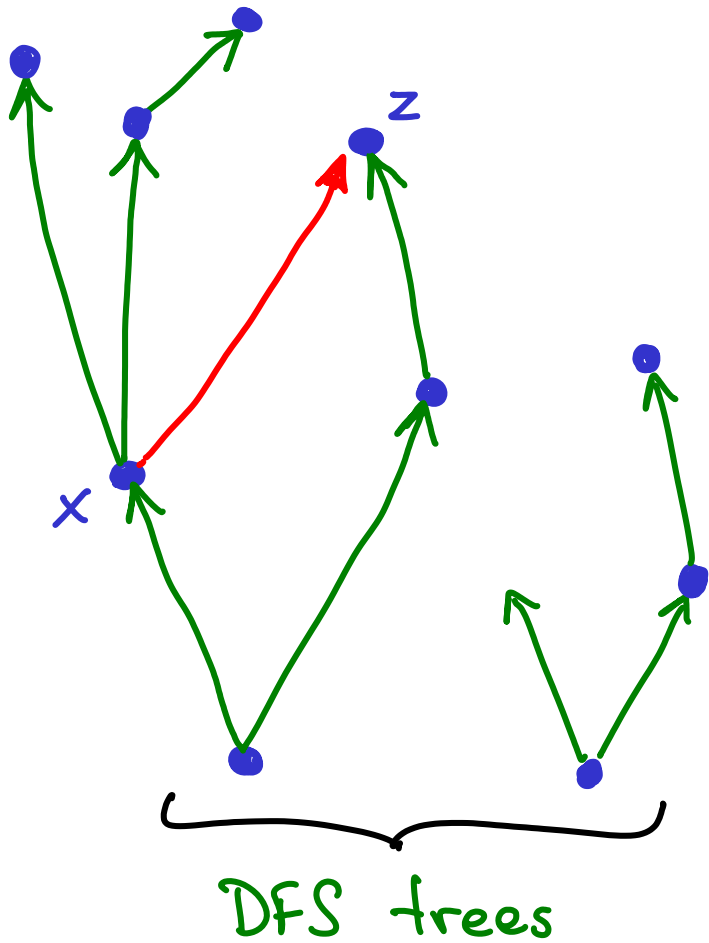
So ?

Intuition

DFS trees

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

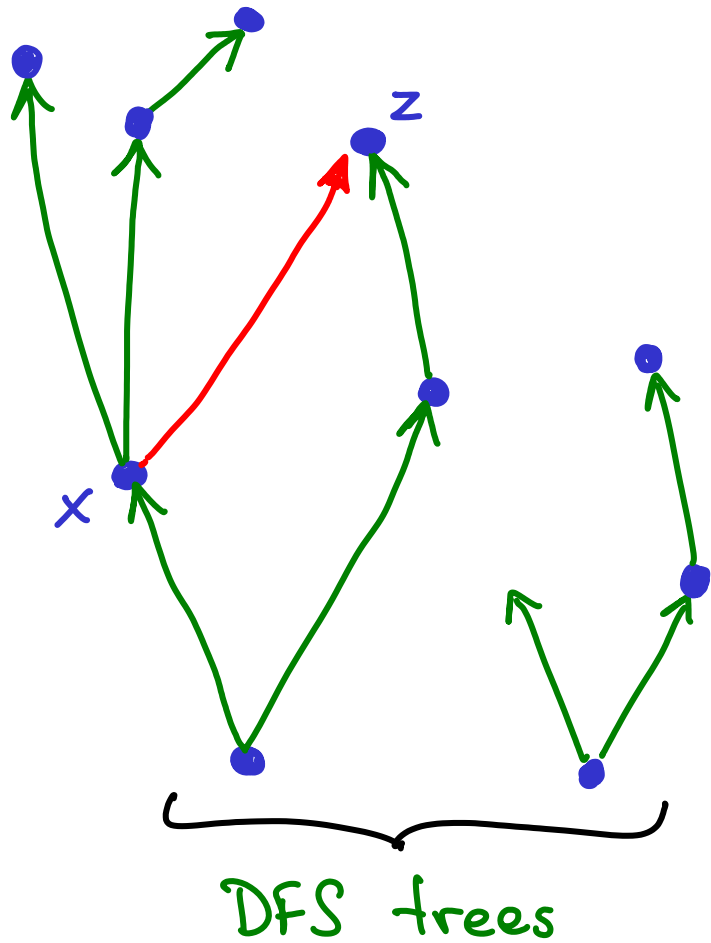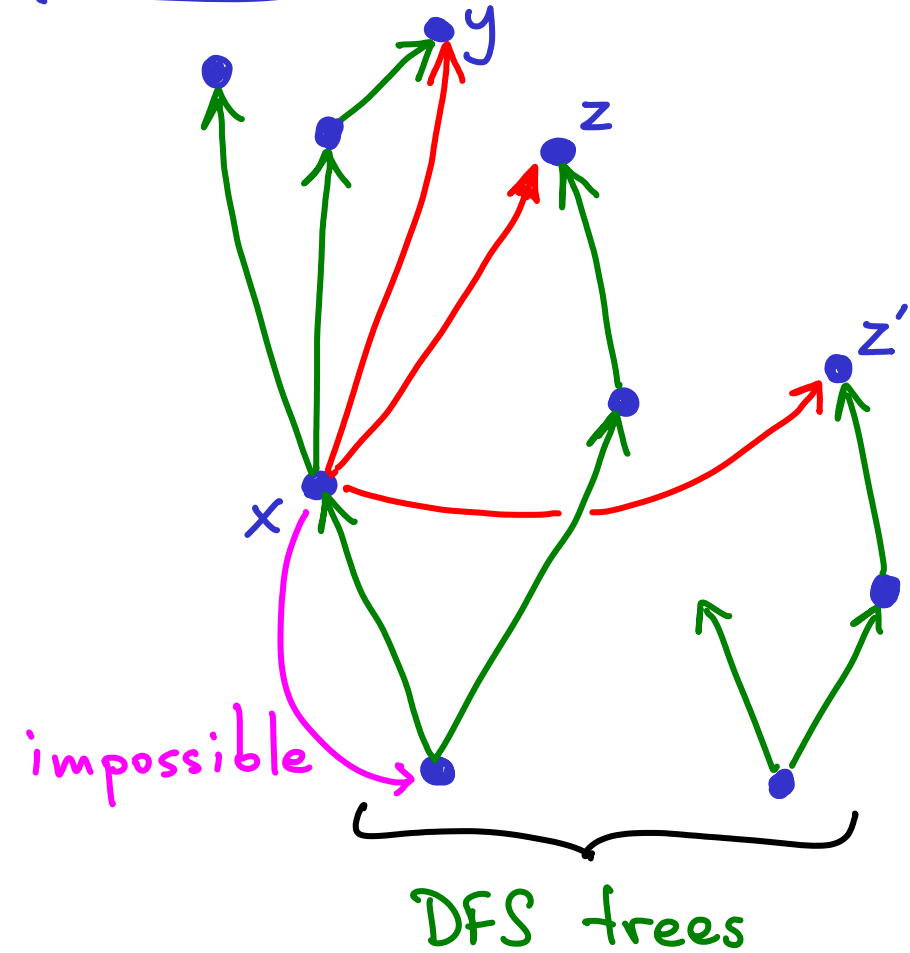So y finished first

---

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

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

So z finished first

↳ we output correctly

Intuition

DFS trees

impossible

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 was explored after z.
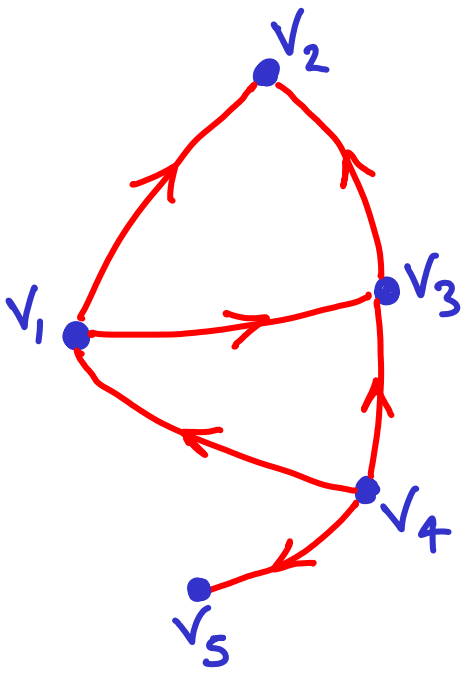(subtree)   (subtree)

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

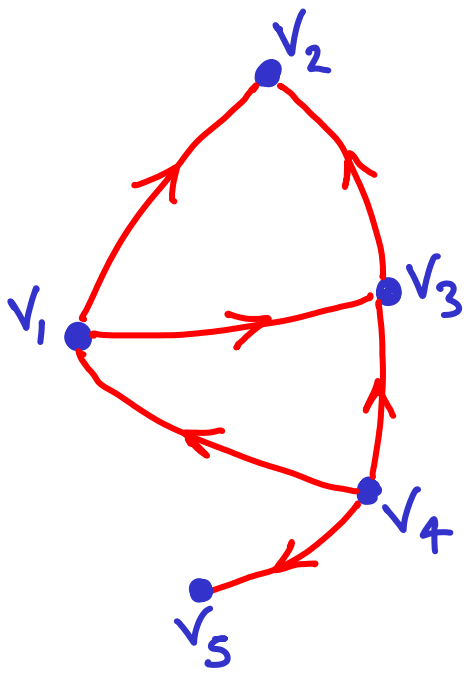So z finished first

---

In both cases, we output correctly

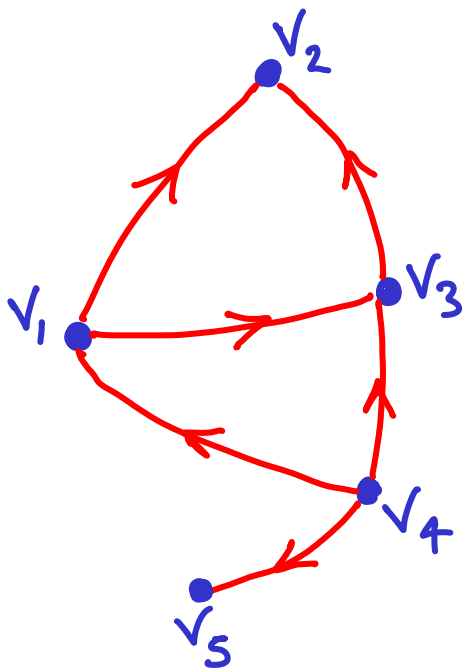## SUMMARY

Run DFS in any order.

SUMMARY

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
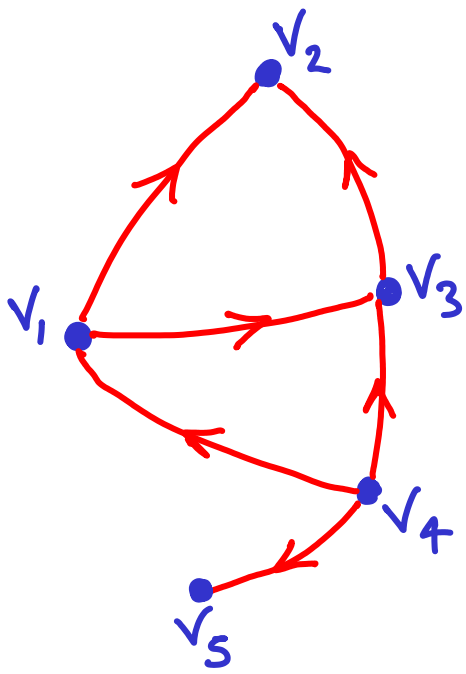
SUMMARY

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.