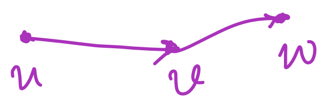
DAGs (contd).

Order on vertices.

- For any edge $\overrightarrow{(u,v)} \in E$, we have $u \leq v$.
- The relation is transitive.

$$u \leq v \text{ and } v \leq w \Rightarrow u \leq w.$$



Want to sort with $\leq$ relation as defined above and $\leq$ is not reflexive, not symmetric but transitive.

Topological sort is a sorting of vertices as per $\leq$.

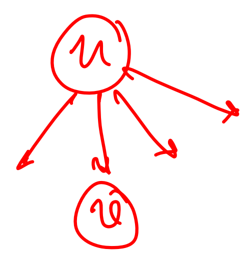
Prereq: Cycle detection:

DFS(s)

Stack: Last In First Out

```
R ← {}
DFS(u):
    Explored[u] ← True
    R ← R ∪ {u}
    For each (u,v) ∈ E :
        If Explored[v] == False:
            DFS(v).
End[u] = current time.
```

Start[u] = current time

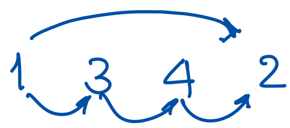


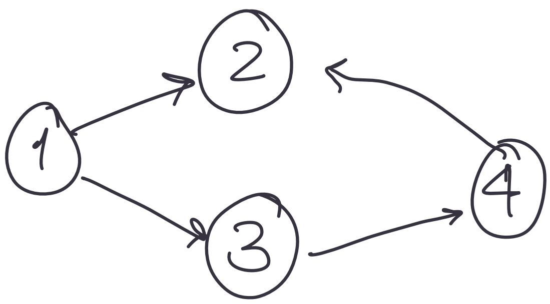
If u was a descendant of v then

- $\text{Start}[v] \leq \text{Start}[u] \leq \text{End}[u] \leq \text{End}[v]$.

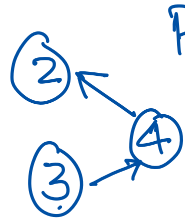
If u and v were un related then $(\text{Start}[u], \text{End}[u])$ and $(\text{start}[v], \text{End}[v])$ are disjoint.

Remark: Back edges $\Leftrightarrow$ Cycles.

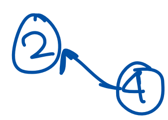




Push 1 into a DS.
Reduce the degree of 2 and 3



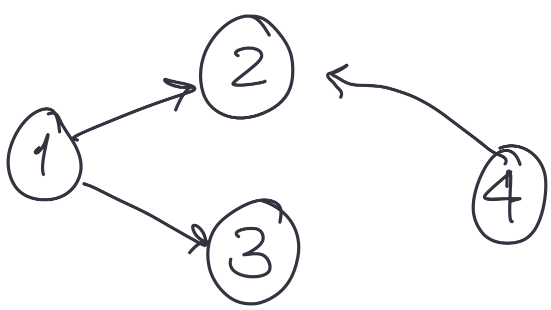
Push 3 into DS
Reduce degree of 4

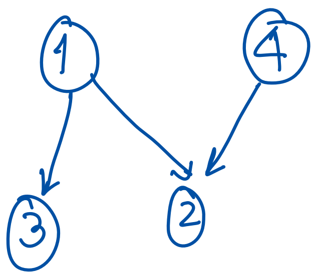


Topological sort (Attempt 1):

Call this set S

- Start from ~~a~~ vertices of in-degree 0 ← Call this s
- Push all elements of S into a queue.





Topological sort (G): // Do cycle existence check.

- Initialize array InDegree [v] $\forall v \in V(G)$.

while $\exists$ a vertex that is not pushed into the DS:

U $\leftarrow$ set of vertices w/ indegree 0. // Use another

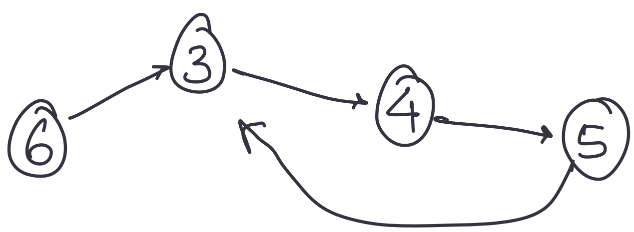
// If U is empty then say "Not DAG". return.

For all $v \in N(U)$:

InDegree [v] = InDegree [v] − 1. // next indegree zero set $\in N(U)$.

DS. append (U).

Claim: When done carefully, the complexity is $O(m+n)$.

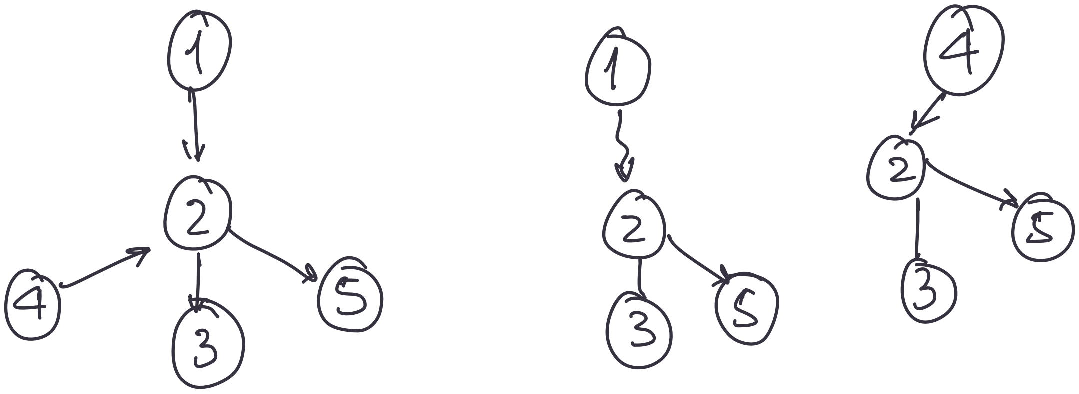


6

Indeg	6	3	4	5
	0	2	1	1
	X	1	1	1

$\leftarrow$ can't get any vertex w/ indeg 0

$\Rightarrow$ G is not a DAG.



1 4 2 3 5 , 4 1 2 3 5

Claim: Topological sort is given by decreasing order of ~~finish~~ times.
End

1 4 2 5 3 , 4 1 2 5 3