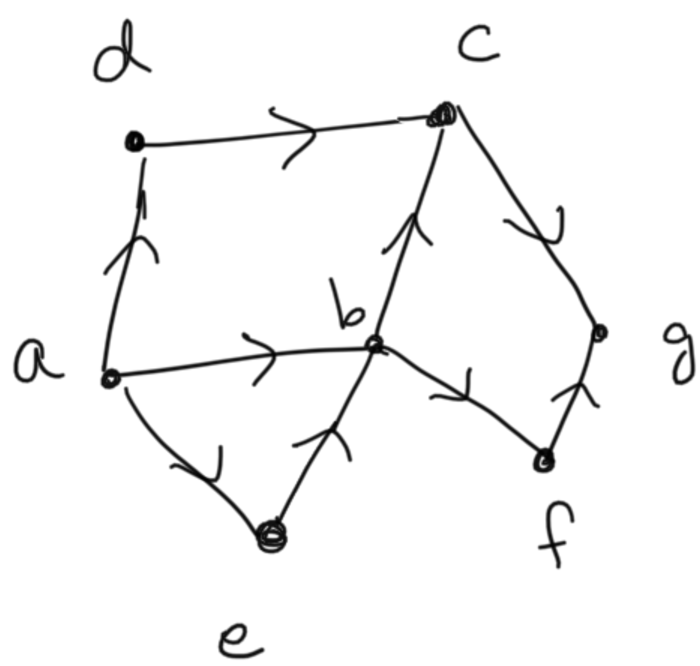
# Sorting DAG

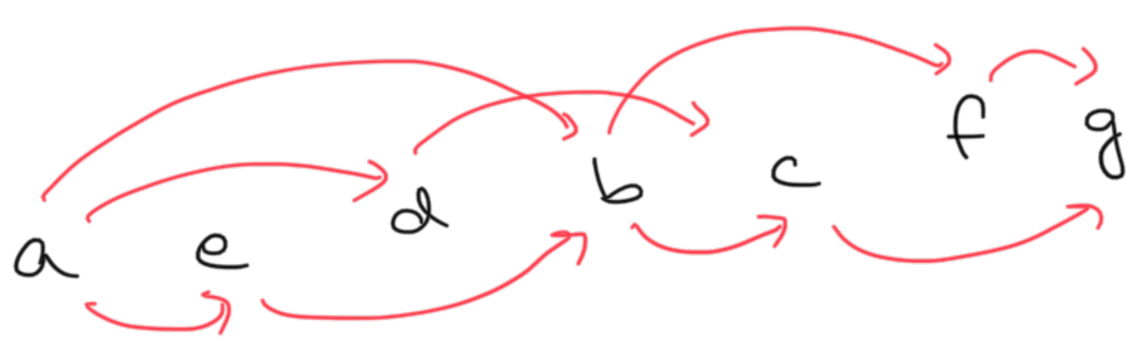
**DAG** - Directed acyclic graph




**example**
DAG G

List out vertices in a seq so that no dag edge going from any vertex in seq to an earlier vertex in sequence

(ex)




[only forward edges, no backward edges]

- "Topological sort" of dag
- In general, $\geq 1$ such sorted sequences

## Algorithm

For each node, compute indegree [node]

- scan all edges once $O(|E|)$

Claim: $\exists$ atleast one vertex of indegree 0

Pf : (by contradiction)

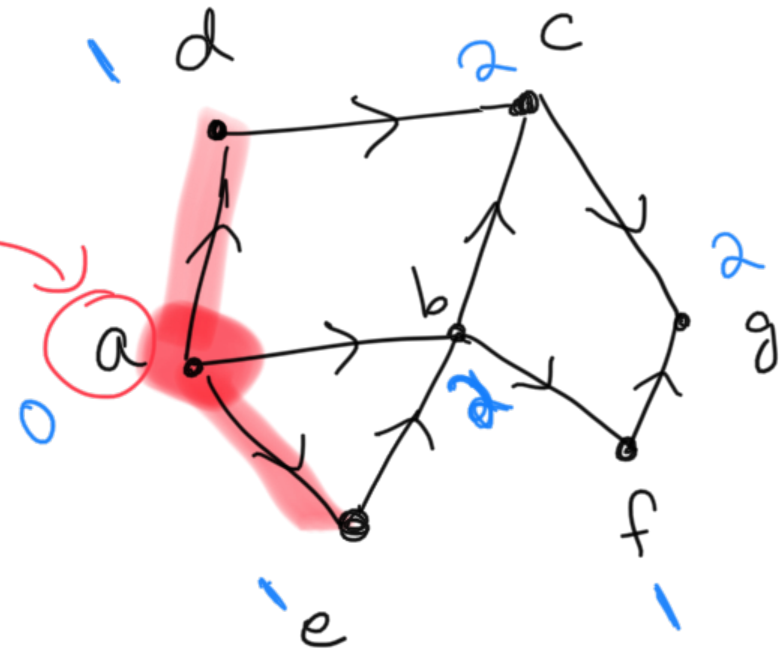
- Assume all nodes have indegree > 0
- Order vertices $v_0, v_1, \ldots, v_{n-1}$
- Reverse all arrows. (still DAG)
- So each node now has outdegree > 0
- Pick $v_0$. out degree > 0
- Pick edge and go to $v_{i(1)}$
  - out degree > 0. Pick edge to go to $v_{i(2)}$. Since DAG, no edge to previously visited vertices ...
  - But after visiting all vertices, can still take edge out! [as last vertex visited has outdeg > 0]

$\rightarrow\leftarrow$

- So $\exists$ indegree 0 nodes in original graph

- List these out in any order
- Delete these nodes + edges incident on them from G.

0 in deg. vertex

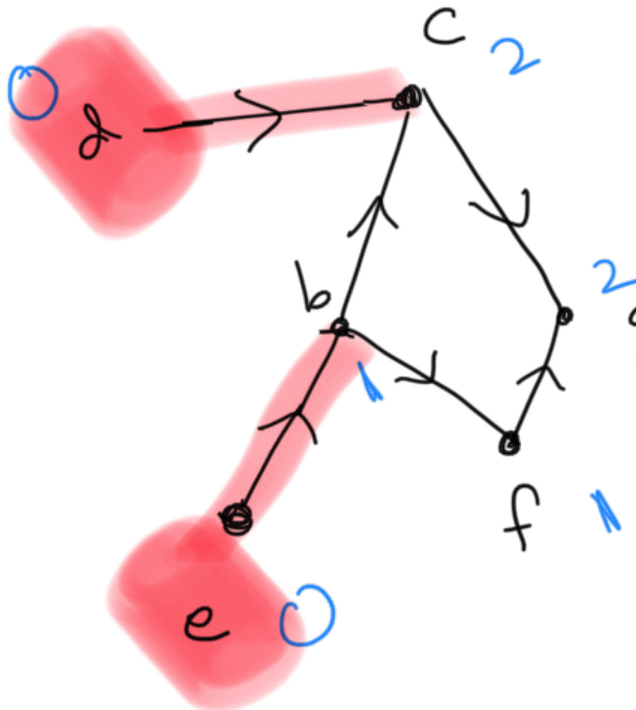



$O(|E|)$

computing in degrees

$G$

[a]




$O(|E|)$

$G \setminus \{a\}$

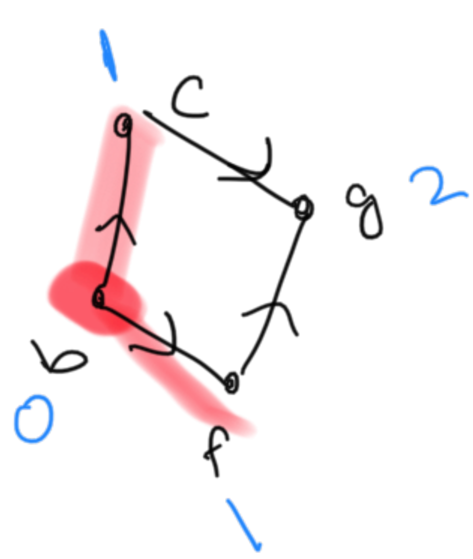
Recompute in degrees

(If for edge $a \to x$, $a$ = source then $\mathrm{indeg}(x) = \mathrm{indeg}(x) - 1$)

$\{d, e\}$ ← new 0 in deg. vertices will exist as $G \setminus \{a\}$ is DAG again.

$G \setminus \{a\} \setminus \{d,e\}$

[a, d, e]

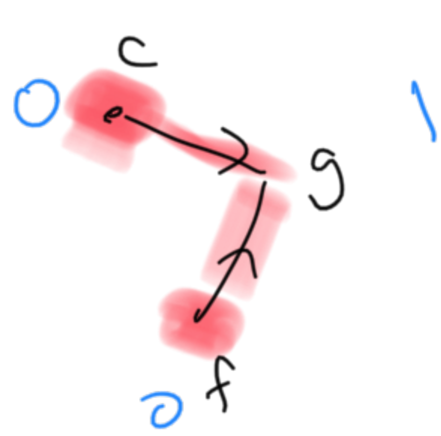



$O(|E|)$

Recompute in degrees

$G \setminus \{a\} \setminus \{d,e\}$ $\setminus \{b\}$

[a, d, e, b]




$O(|E|)$

[a, d, e, b, c, f]   0 g

G ~ {a}
~ {d, e}
~ {b}
~ {c, f}

$O(|E|)$

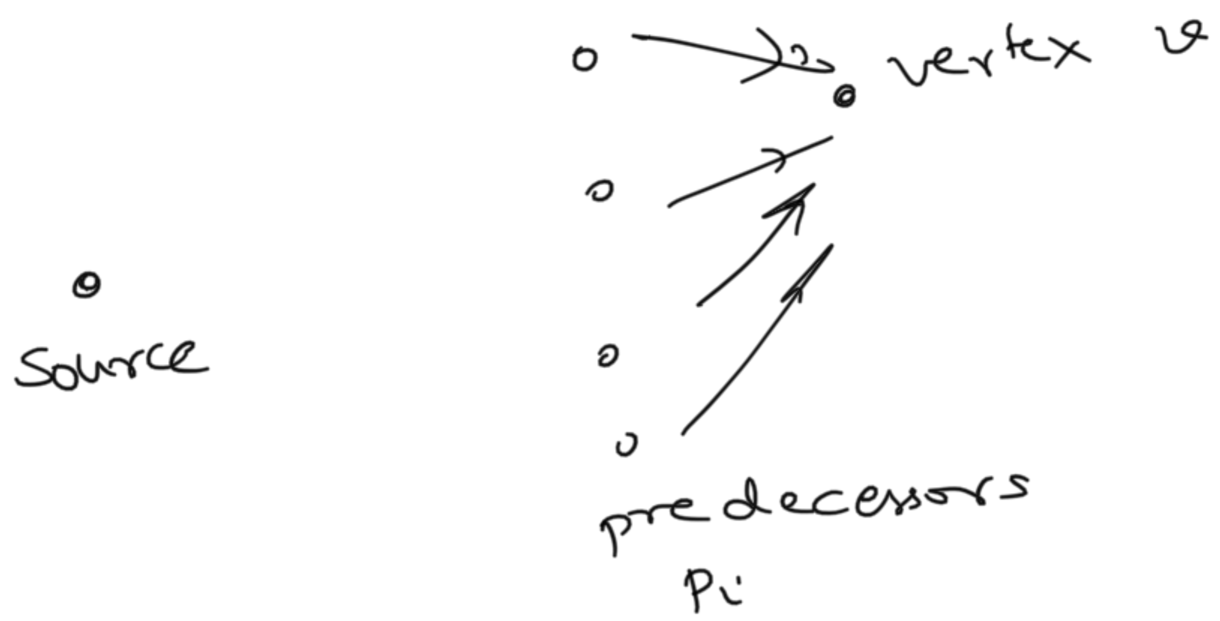
[a, d, e, b, c, f, g] ← Top sort

$\leq |V|$ passes, each pass, $O(|E|)$ scan of edge list.

$$O(|V||E|)$$

---

Shortest path in a dag with source vertex [index 0] to all other vertices




$$d(\text{source}, v) = \min_i \left[ d(p_i, v) + \text{weight}[p_i - v] \right]$$

So only thing is,

in order to compute $d(\text{source}, v)$,

we must have already computed

$d(\text{source}, p_i)\ \forall i$

Processing the vertices in the topological sorted order ensures this as at any point we see a vertex v, we are assured we have already seen all its predecessors

Top. sorted order

$$\times \ldots \underbrace{* \quad * \quad *}_{p_i \text{ all have to occur here.}} \quad \underset{v}{*} \quad * \quad * \quad * \quad *$$