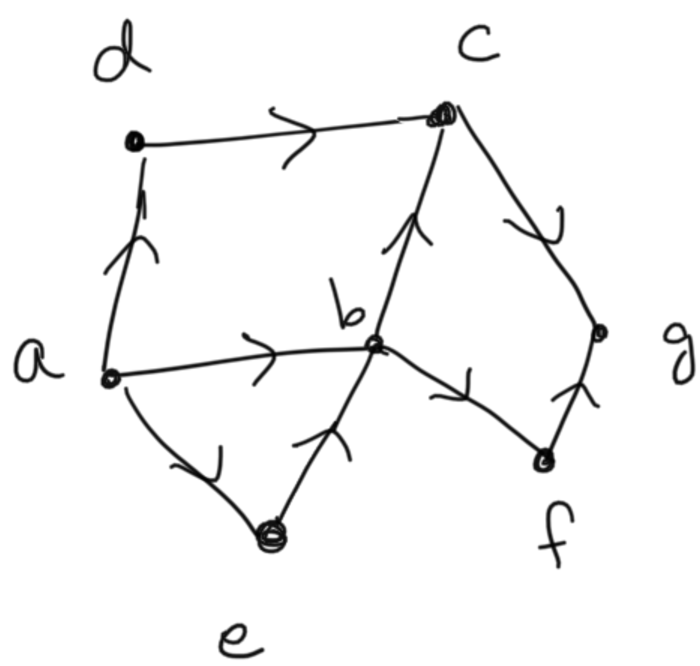
Sorting DAG

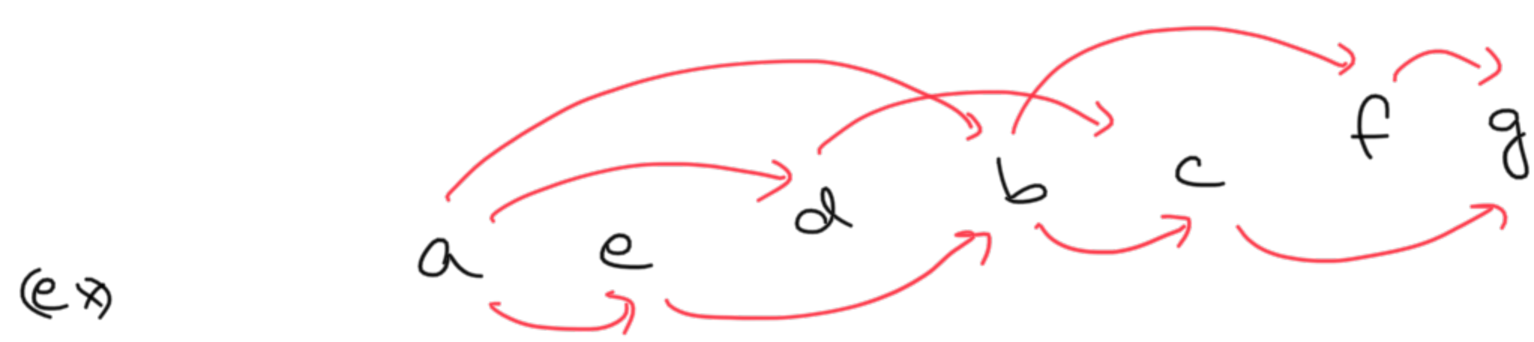
DAG – Directed acyclic graph



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, ≥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

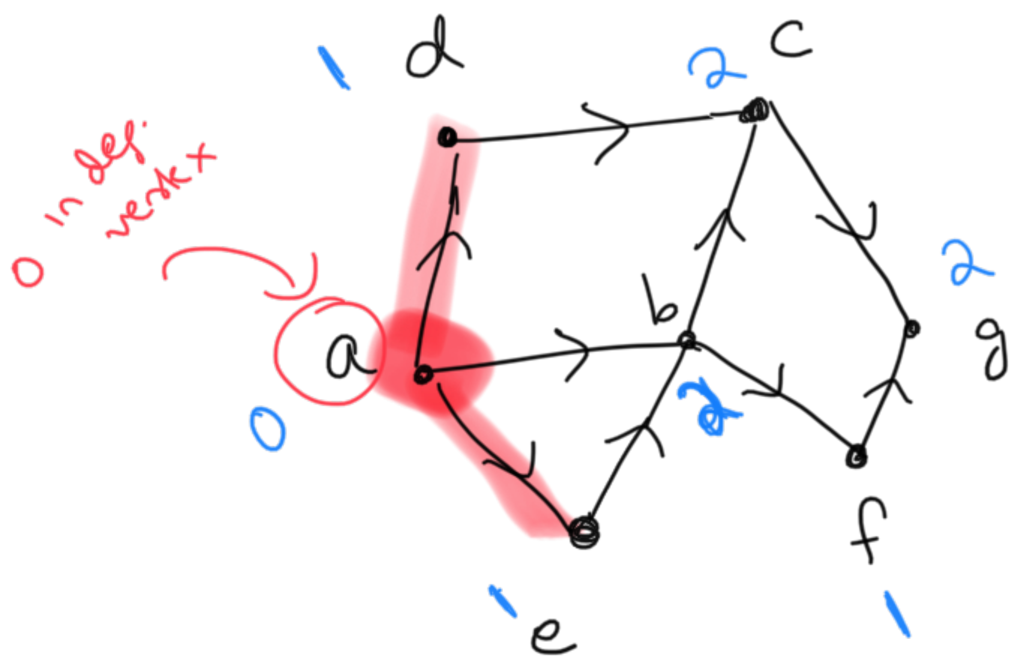
Pf : (by contradiction)

- 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

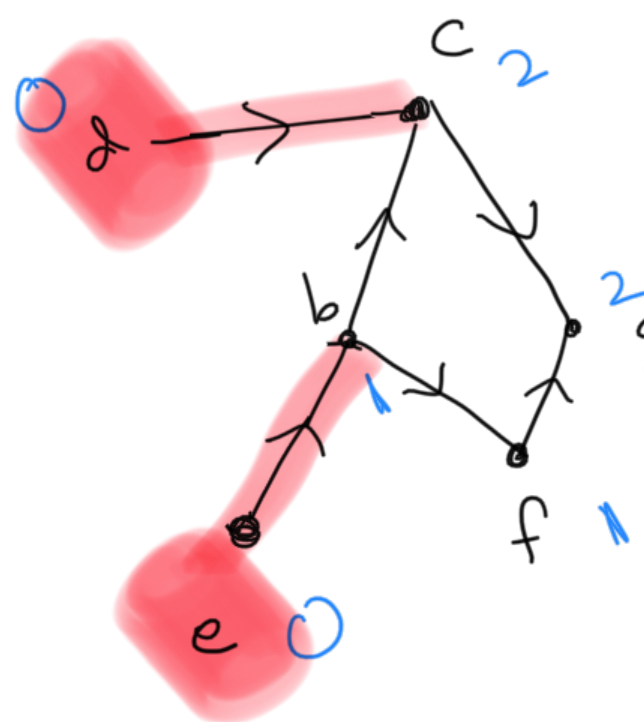
- List these out in any order
- Delete these nodes + edges incident on them from G.



computing in degrees

G

[a]



$G \setminus \{a\}$

Recompute in degrees

(If for edge $a \to x$

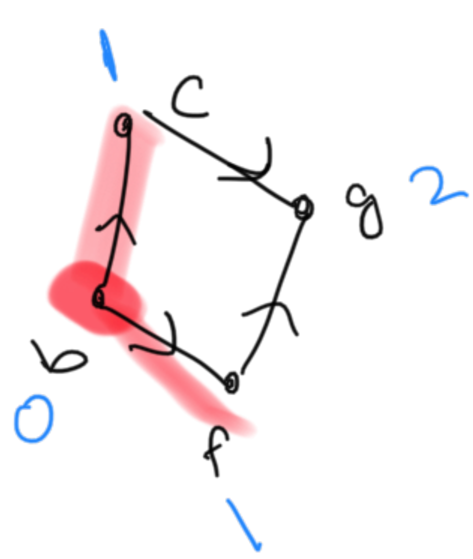
a = source

then indeg (x) = 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]

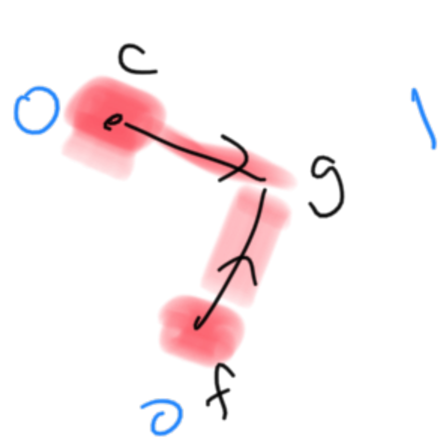


Recompute in degrees

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

$\setminus \{b\}$

[a, d, e, b]



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

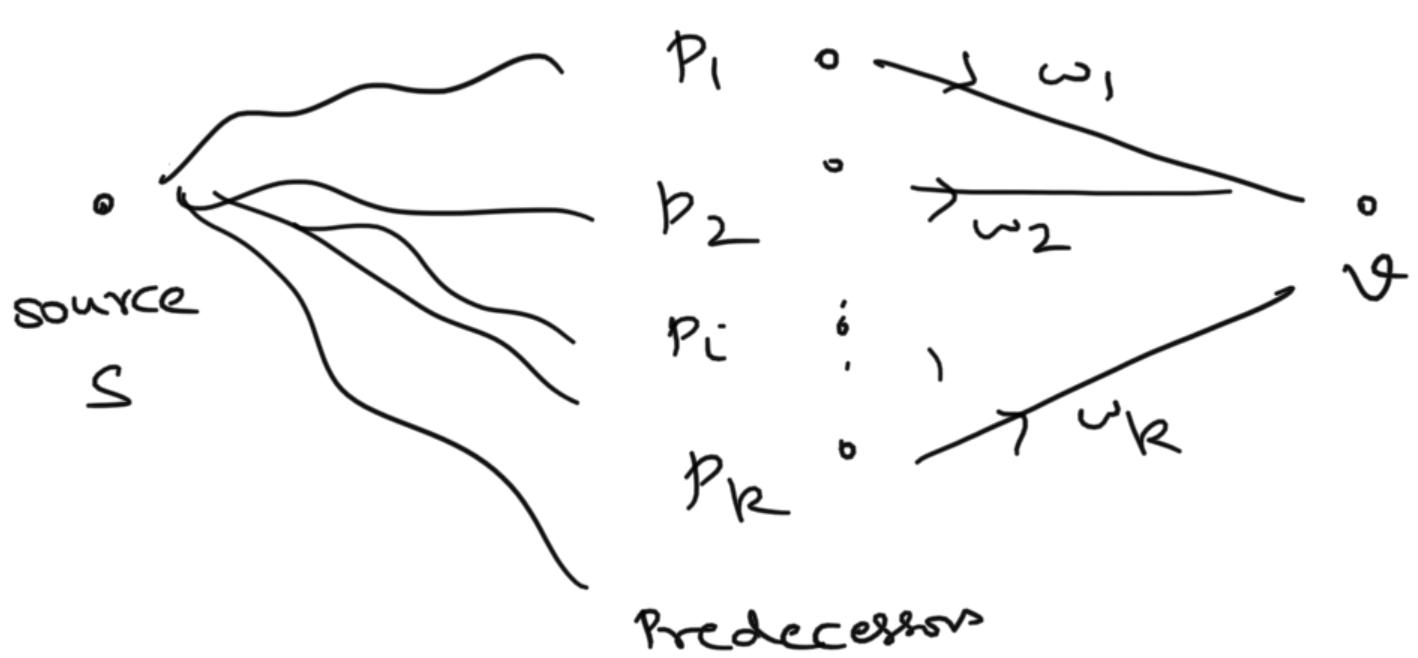
0

g

G → {a}
→ {d, e}
→ {b}
→ {c, f}

[a, d, e, b, c, f, g] ← Top sort

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



$$d(s, v) = \min_i \left[d(s, p_i) + w_i \right]$$

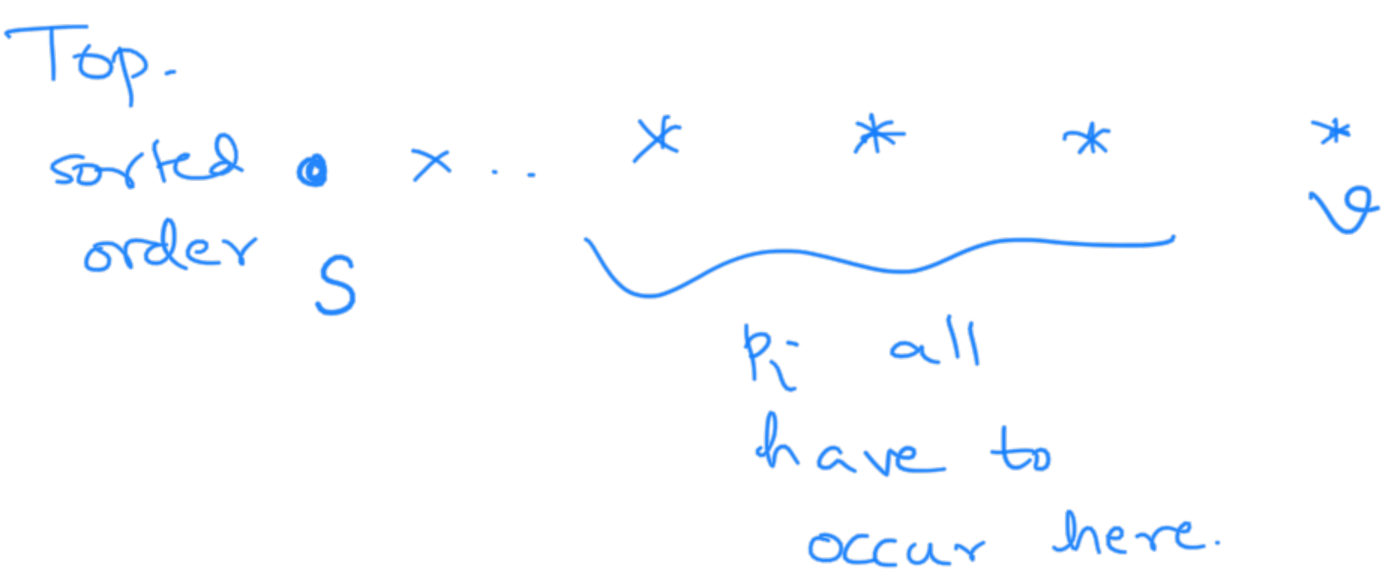
so only thing is,

in order to compute $d(s, v)$

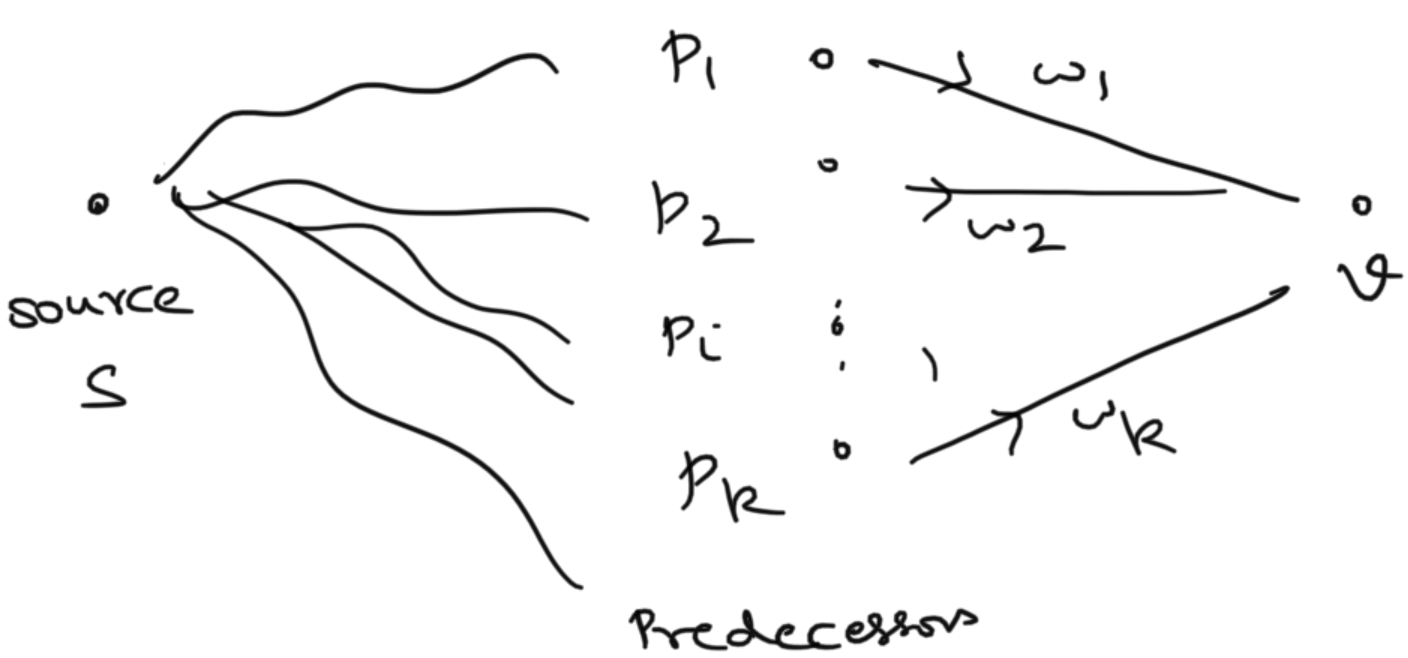
we must have already computed $d(s, p_i)$ $\forall i$

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



Longest path in DAG from source to any vertex v



$$D(s, v) = \max_i \left[D(s, p_i) + w_i \right]$$

Processing the vertices in the topological sorted order ensures that we would have already solved $D(p_i, v)$ $\forall i$ before solving

$D(s, v)$

Top. sorted order

s ... v

p_i all have to occur here.

---

* Longest path in DAG has to originate from source if graph is connected.

source → u → v

adjoin source to u

if longest path

Top sorted order

s - - - u - - - v

path goes forward only.

Complexity

→ Initialize indegree ∀ vertices : $O(|E|)$

→ Find vertex with indeg 0 : potentially $O(|V|)$

I v_0

if naively storing indegrees in unsorted list

III

→ Reduce indeg of all vertices adjacent to picked vertex of indeg 0 : $O(\text{outdeg}(v_0))$

have to do

$\forall\ v \in V,$

step II $O(|V|)$

step III $O(\text{outdeg}(v_0))$

$\leadsto\ O(|V|^2) + O\left(\sum_{v \in V} \text{out deg}(v_i)\right)$

$\leadsto\ O(|V|^2 + |E|)$

In a directed graph,

each edge → +1 outdeg for some vertex

$\therefore\ |E| = \sum_{v_i \in V} \text{out deg}(v_i)$

Better implementation of step II :

→ maintain a queue of indeg 0 vertices.

→ so finding a 0 deg vertex is $O(1)$ operation

So doing $\forall v \in V$, doing step II $\rightsquigarrow O(1) \times |V|$

$= O(|V|)$ time

* while doing step III, updating indegrees, add any indeg 0 vertices to queue

So overall $O(|V| + |E|)$