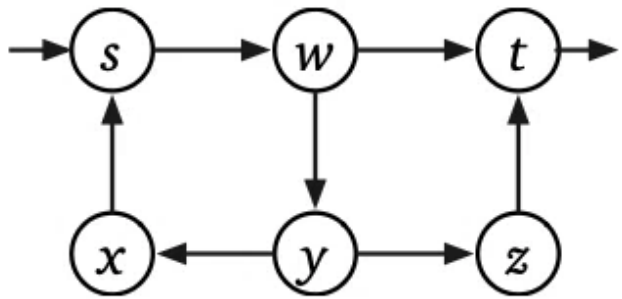


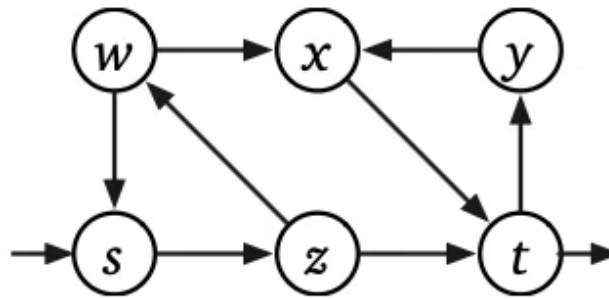
GPS 8 Mon Oct 26

HW 8 Tue Oct 27 (yes, Dijkstra is OK)

Given a directed graph $G = (V, E)$ + two vertices $s, t \in V$. Is there a walk from s to t with length divisible by 3?



yes! G_1



no! G_2

Reduce to standard reachability.

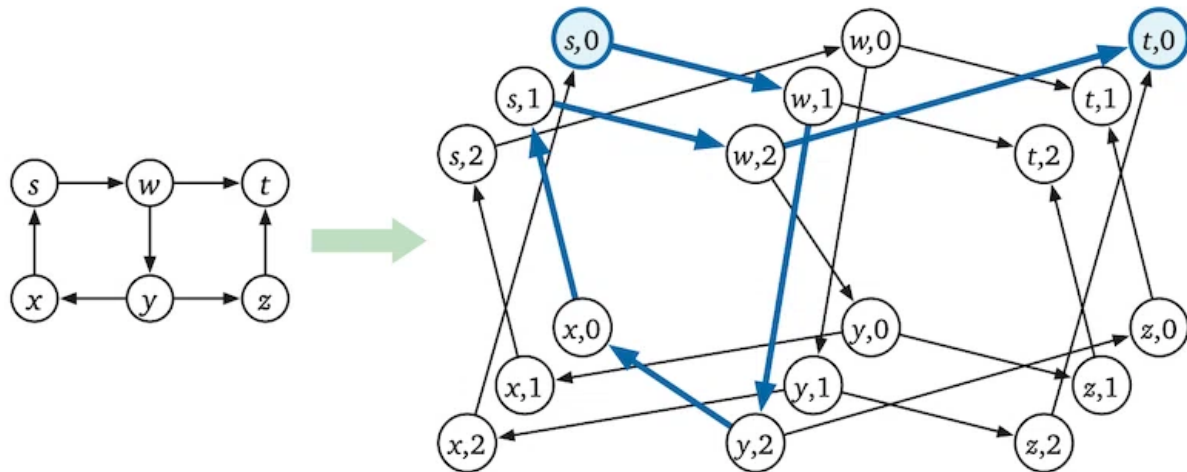
Don't modify WFS directly!

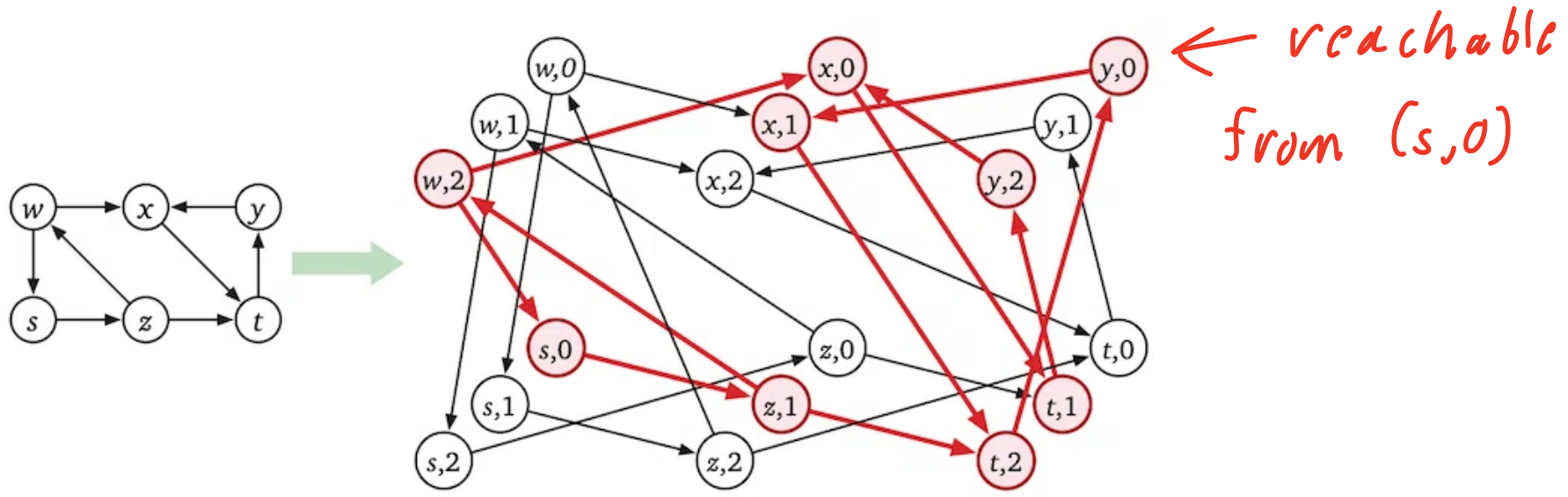
Build a new graph $G' = (V', E')$.

$$V' = \{(v, i) \mid v \in V \text{ and } i \in \{0, 1, 2\}\}$$
$$= V \times \{0, 1, 2\}$$

$$E' = \{(u, i) \rightarrow (v, i+1 \bmod 3) \mid u \rightarrow v \in E, i \in \{0, 1, 2\}\}$$

There is a walk from s to t in G of length div. by 3 iff $(s, 0)$ can reach $(t, 0)$ in G' .





So, call $WFS((s,0))$ & return if $(t,0)$ gets marked.

Time: $O(|V| + |E|) = O(|V| + |E|)$.

Graph layering

WhateverFirstSearch(s):

put $(\emptyset, s)$ into the bag

while the bag is not empty

take (p, u) from the bag

if u is unmarked

mark u

$u.\text{parent} \leftarrow p$

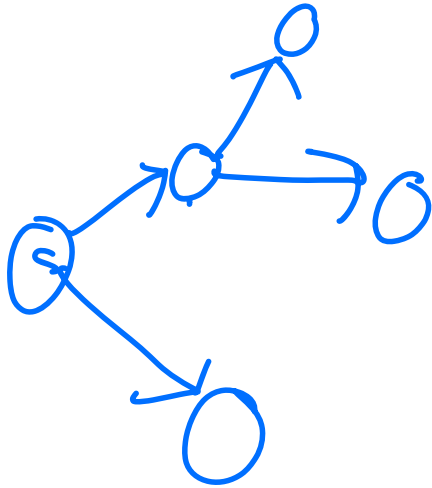
for each edge $u \rightarrow v$

put (u, v) into the bag

For each marked u ,

$u.\text{parent} \rightarrow u$ is
an edge (except for
 s)

These edges form
a tree rooted at
 s .



(If G is undirected, the
tree is a spanning tree of
 s 's connected component.)

Different bag data structures give

useful trees, but change run time.

If bag uses T time for insert & removal,

WFS takes $O(|V| + |E| \cdot T)$ time.

Queue: Breadth-first search (BFS)

shortest path tree from s (no edge weights)

$O(|V| + |E|)$ time.

Priority queue: "Best-first search"

depends on priority...

Suppose we have non-negative weights

$$w: E \rightarrow \mathbb{R}^+$$

Give $u \rightarrow v$ priority = (sum of weights in tree from s to u) + $w(u \rightarrow v)$

Dijkstra's algorithm for shortest paths!

Say G is undirected & (u, v) has priority $w(uv)$.

Prim's algorithm for min spanning trees.

Binary heap takes $O(\log n)$ time

per operation.

If G is simple, total time: $O(|V| + |E| \log |E|)$
 $= O(|V| + |E| \log |V|)$

If G is a stack: depth-first
search.

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

DFS(v):

mark v

for each edge $v \rightarrow w$

if w is unmarked

$w.parent \leftarrow v$

DFS(w)

DFSAll(G):

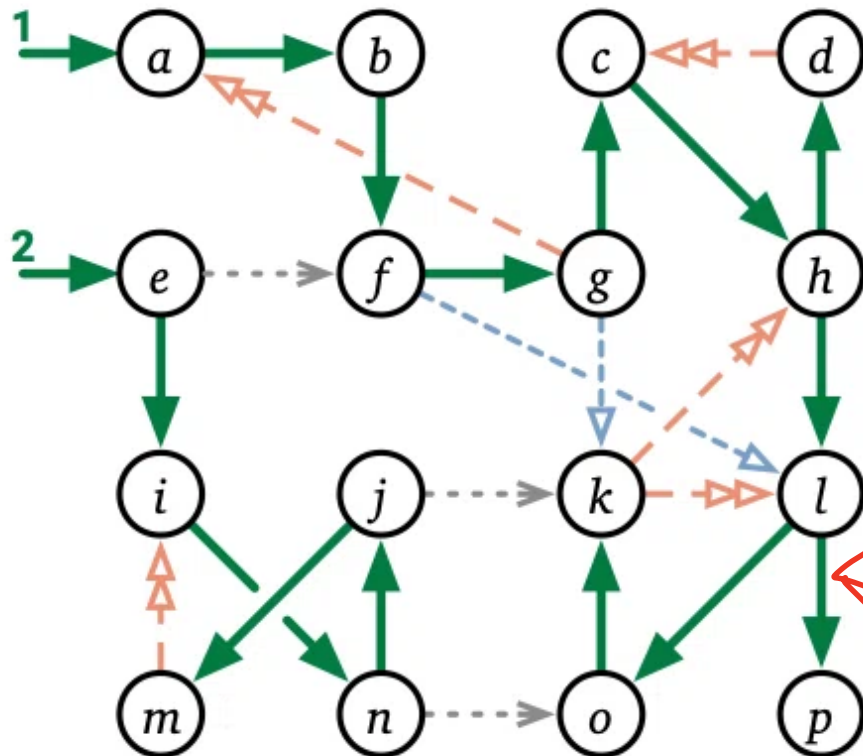
for all vertices v in G

unmark v

for all vertices v in G

if v is unmarked

DFS(v)



we want to learn
about everything, but
in a depth-first way

thick green tree edges
 $O(|V| + |E|)$ for

DFS All.

1. Every vertex is marked. We call $\text{DFS}(v)$ on all vertices v .
2. When we call $\text{DFS}(v)$, all vertices reachable without going through one already marked will be marked before $\text{DFS}(v)$ finishes.
3. If $\text{DFS}(v)$ calls (indirectly) $\text{DFS}(w)$ there is a path from v to w .

DFSAll(G):

$clock \leftarrow 0$

for all vertices v in G

unmark v

for all vertices v in G

if v is unmarked

$clock \leftarrow \text{DFS}(v, clock)$

DFS($v, clock$):

mark v

$clock \leftarrow clock + 1; v.pre \leftarrow clock$

for each edge $v \rightarrow w$

if w is unmarked

$w.parent \leftarrow v$

$clock \leftarrow \text{DFS}(w, clock)$

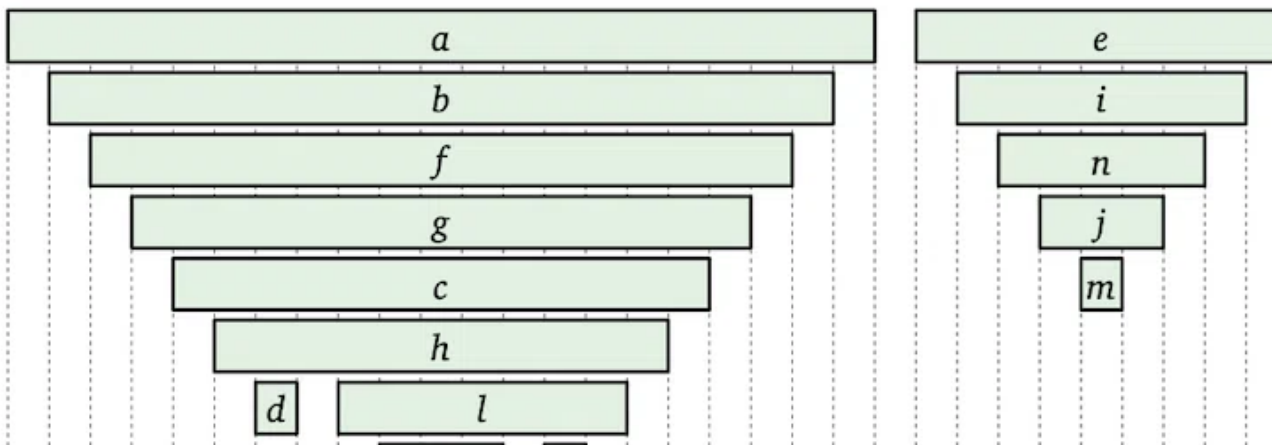
$clock \leftarrow clock + 1; v.post \leftarrow clock$

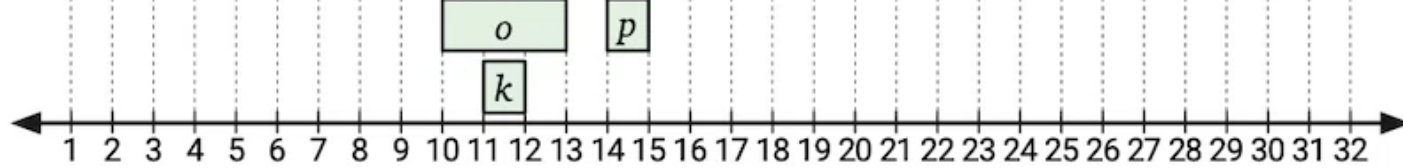
return $clock$

start time

finish time

v 's active interval is $[v.pre, v.post]$



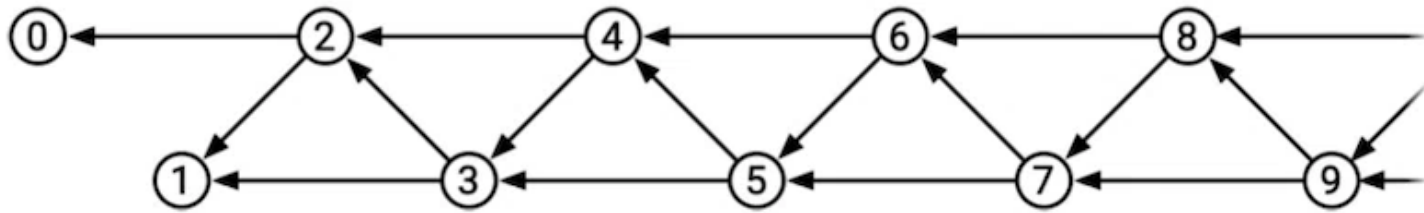


active intervals are disjoint or nested

order by $v.pre$: a preorder (meh)

order by $v.post$: a postorder hmm

Dependency graphs

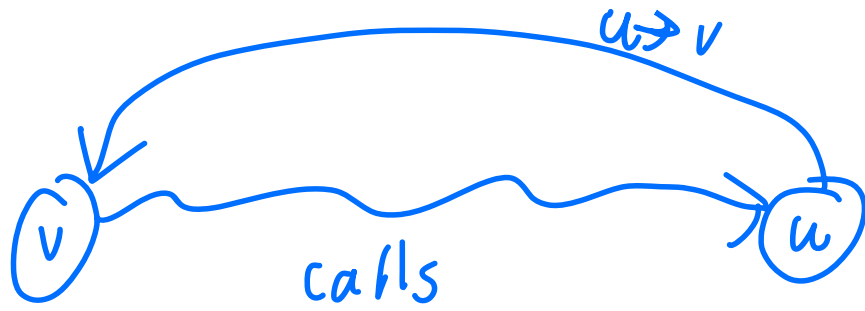


Cannot have a cycle!

Problem: Given directed $G = (V, E)$,
does G have a directed cycle or
is it a directed acyclic graph (dag)?

Run $DFSAll(G)$.

If some $\text{DFS}(v)$ (indirectly) calls $\text{DFS}(u)$ +
there is an edge $u \rightarrow v$, there is a
cycle!



$u \rightarrow v$ is a back edge.

What about the converse?