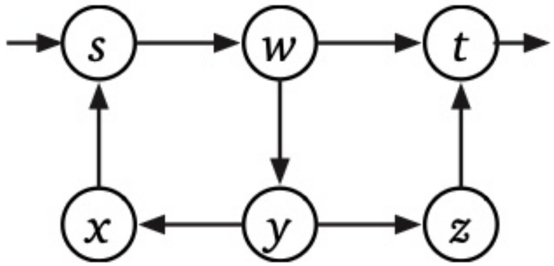
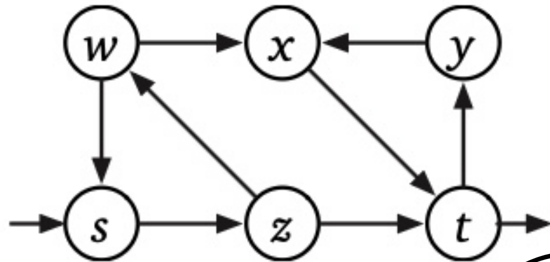
Given <u>directed</u> $G=(V,E)$ + two vertices s + t.

Is there a walk of length $0 \bmod 3$?



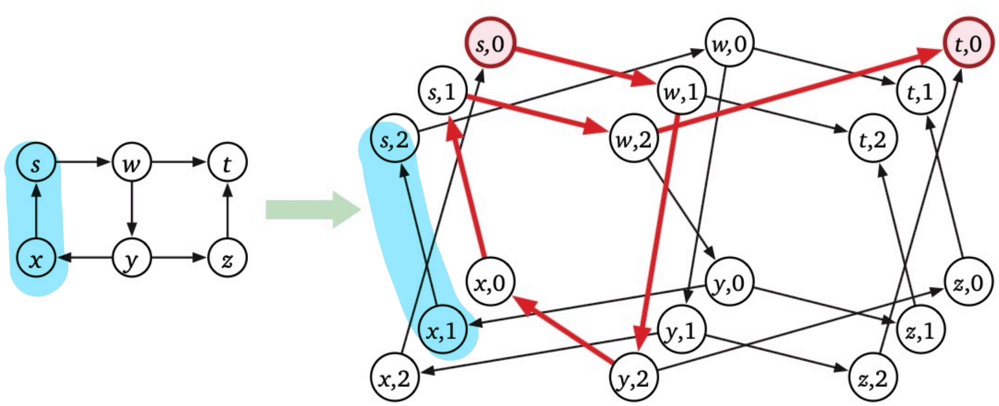


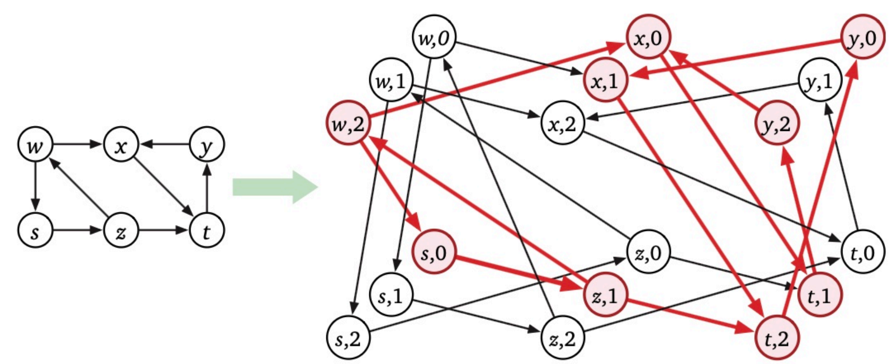
There is a (u,v)-walk of length $\ell \bmod 3$ in G iff $((u,i),(v,i+\ell \bmod 3))$-walk in G'.

Construct $G'=(V',E')$

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

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





$\Rightarrow$ want to know if there is an $(s,0)$ to $(t,0)$ walk in G'.

Call WhateverFirstSearch$((s,0))$ + return whether $(t,0)$ is marked.

Analysis: Build G' in $O(3V+3E)$ time

WFS in $O(3V+3E)$

Total $O(3V+3E) = O(V+E)$

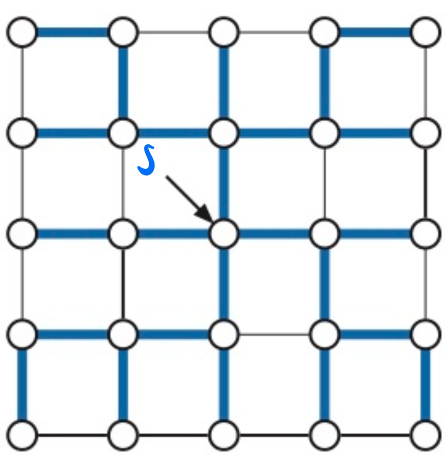
```
WhateverFirstSearch(s):
  put (∅, s) in bag
  while the bag is not empty
      take (p, v) from the bag          (⋆)
      if v is unmarked
          mark v
          v.parent ← p
          for each edge vw              (†)
              put (v, w) into the bag   (⋆⋆)
```

WFS(s): The pairs $(v, v.parent)$ form a rooted tree over all vertices reachable from s.

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

Time is $O(V+ET)$ where T is the time per op on bag.

If bag is... queue.



breadth-first search (BFS)

shortest unweighted paths from s

$O(V+E)$

priority queue...

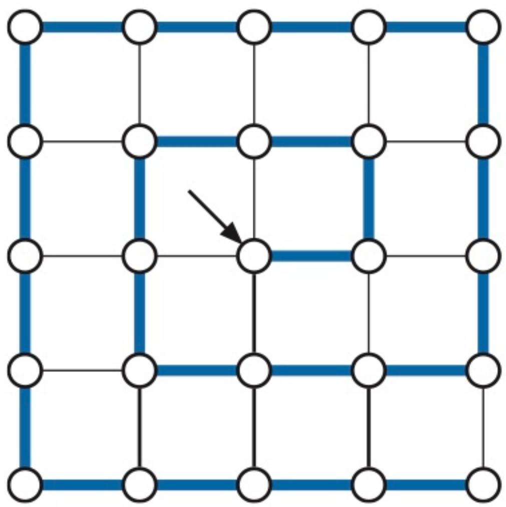
If binary heap...

$O(V+E\log E)$ time.

Priorities are edge weights + G is undir.

MST via Prims

If bag is a .. stack.



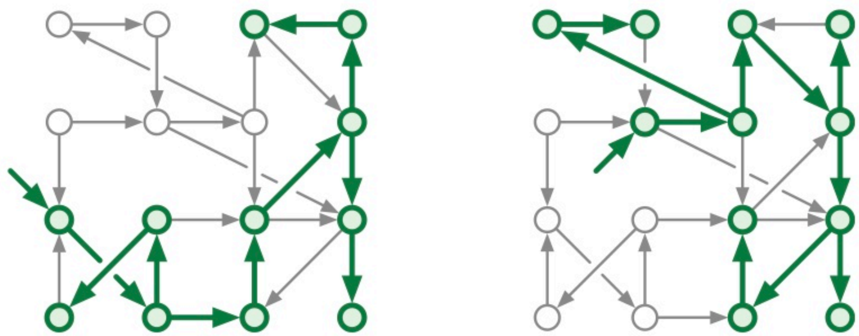
depth-first search (DFS)

O(V + E)

Priorities are length of s to u path + w(u → v)

Dijkstra for weighted shortest paths.

Depth-first search:

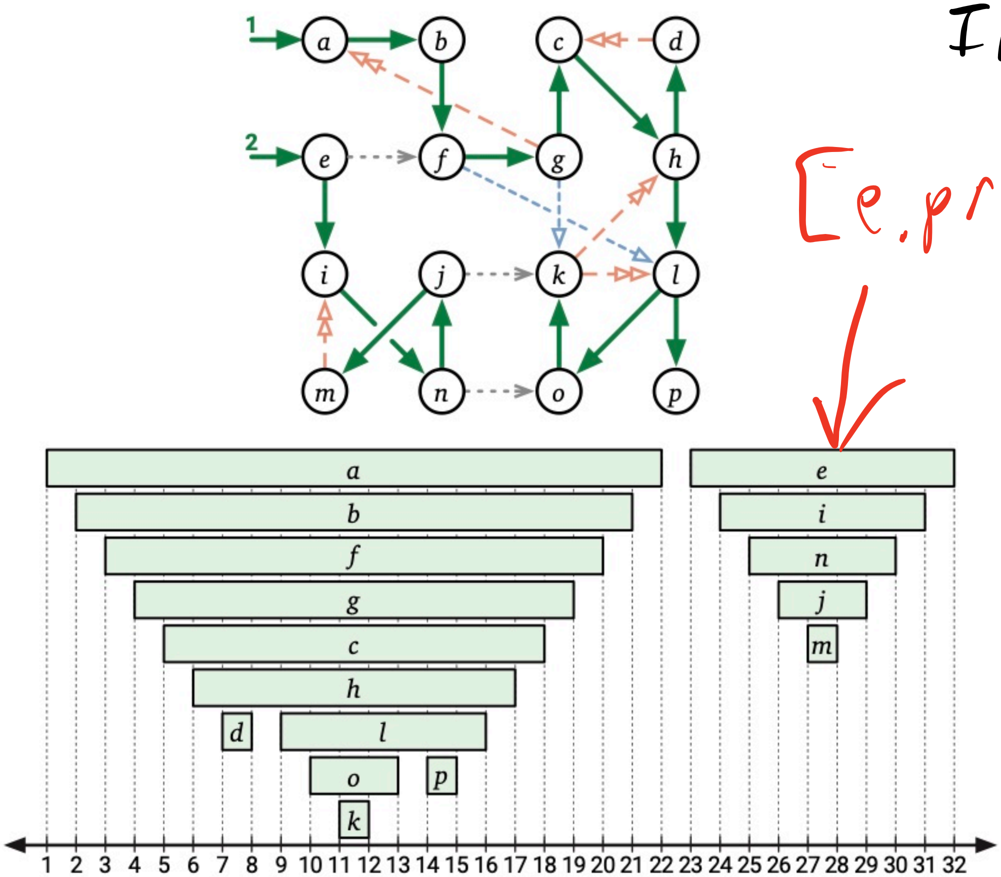


```
DFS(v):
    mark v
    PreVisit(v)
    for each edge v→w
        if w is unmarked
            parent(w) ← v
            DFS(w)
    PostVisit(v)
```

```
DFSAll(G):
    Preprocess(G)
    for all vertices v
        unmark v
    for all vertices v
        if v is unmarked
            DFS(v)
```

```
DFSAll(G):
    clock ← 0
    for all vertices v
        unmark v
    for all vertices v
        if v is unmarked
            clock ← DFS(v, clock)
```

```
DFS(v, clock):
    mark v
    clock ← clock + 1;  v.pre ← clock
    for each edge v→w
        if w is unmarked
            w.parent ← v
            clock ← DFS(w, clock)
    clock ← clock + 1;  v.post ← clock
    return clock
```



Intervals are disjoint or overlap.

[e.pre, e.post]

_.pre gives a preorder

_.post gives a postorder

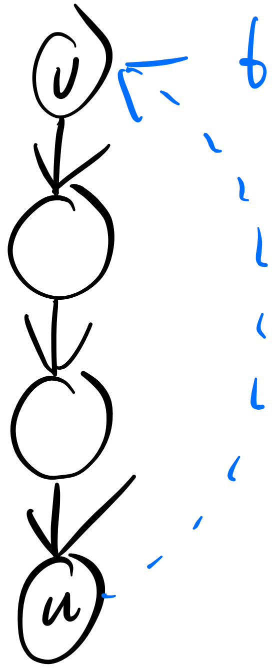
Pre: a b f g c h d l o k p e i n j m

Post: d k o p l h c g f b a m j n i e

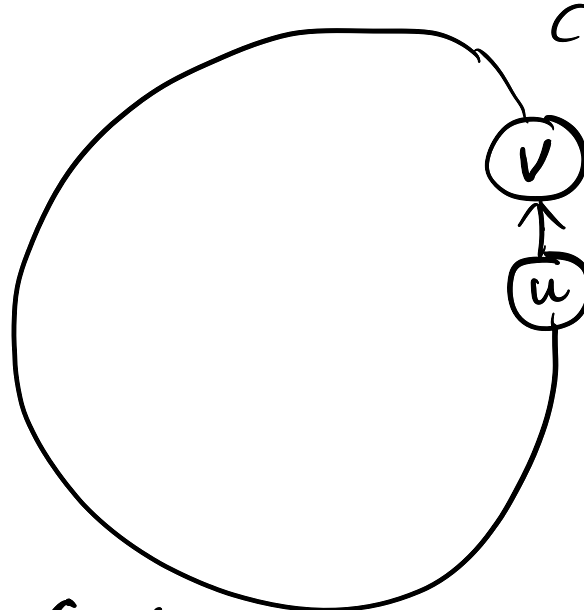
Directed acyclic graph (dag): directed graph with no directed cycle.

How to tell if G is a dag?

"back edge" => cycle



cycle C => back edge



← first vertex visited by DFSAll on C

DFSAll finds a back edge iff there is a directed cycle.

$u \to v$ is a back edge iff $u.post < v.post$.

$\Rightarrow$ Lemma: G has a cycle iff for some edge $u \to v$, $u.post < v.post$.

Algo for cycle detection on G.

Run DFSAll(G).

Return if $u.post < v.post$ on <u>any</u> $u \to v$.

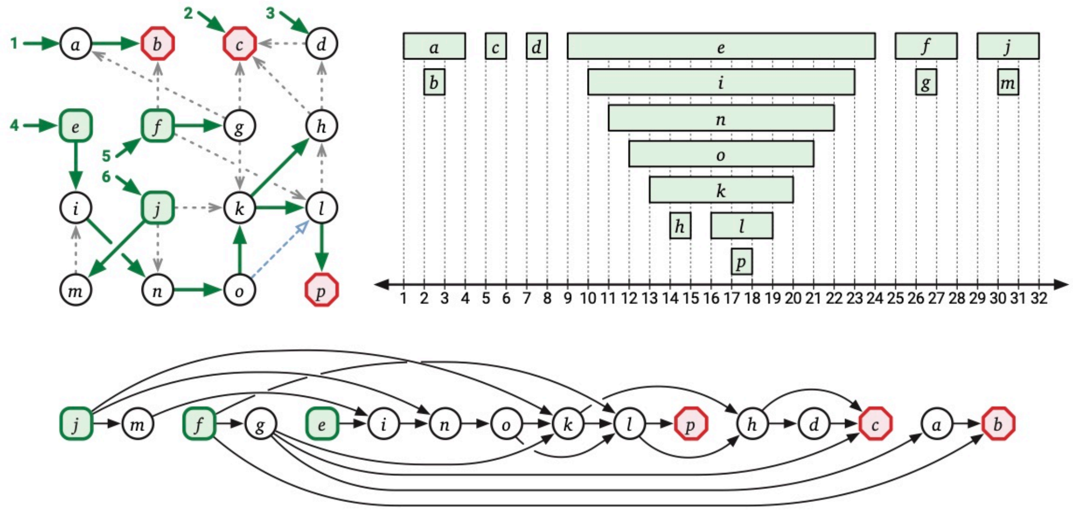
Analysis: $O(V+E)$

Topological ordering: order V so that $u \prec v$ for all $u \to v$.

If G has a cycle: No topological order.

o.w., $u.post > v.post$ for every edge $u \to v$.

$\Rightarrow$ reverse post order is a topological order



TopologicalSort(G):

$\quad clock \leftarrow V$

$\quad$ for all vertices v in postorder

$\qquad S[clock] \leftarrow v$

$\qquad clock \leftarrow clock - 1$

$\quad$ return $S[1\,..\,V]$