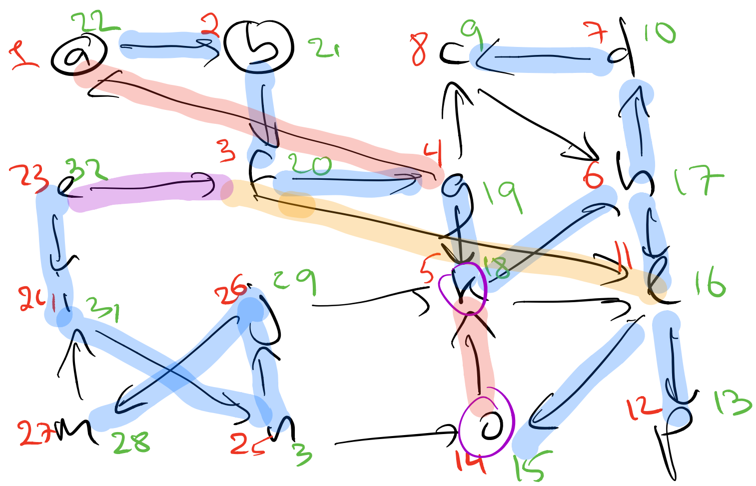
TODAY
- DFS & DAGs
- topo sort
- graph decomposition
- DP on DAGs

```
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
```

Figure 6.3. Defining preorder and postorder via depth-first search.



$u \longrightarrow v$

v is new when DFS(v) begins

u.pre < v.pre < v.post < u.post

tree edge

if v.parent = u

forward edge

v is active when DFS(u) is called

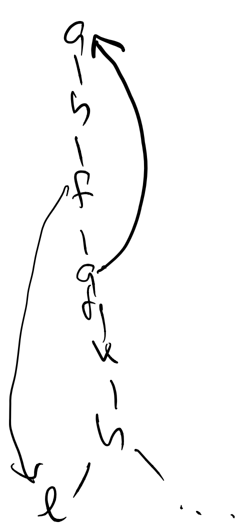
v.pre < u.pre < u.post < v.post

backwards edge

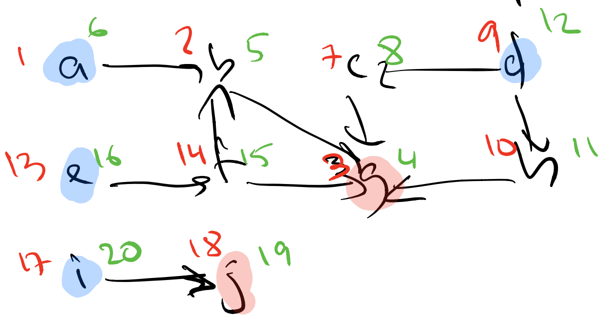
if v is finished when u begins

v.pre < v.post < u.pre < u.post

cross edge



Directed Acyclic Graph (DAG)



source node
→ indegree 0

sink node
→ out degree 0

finding cycles

G has a cycle iff ∃ edge $u \to v$ with $u.post < v.post$

1. Do DFS all to compute post order ($O(V+E)$)
2. For each edge $u \to v$ check if $u.post < v.post$ $O(E)$

$O(V+E)$ time

Topological sort

Compute a total ordering $u < v$ on vertices such that for every edge $u \to v$, $u \nless v$

[impossible if ∃ a cycle]

For a dag for any edge $u \to v$

$u.post > v.post$

⇒ post-ordering is a reverse topo sort

⇒ can topo sort in $O(V+E)$

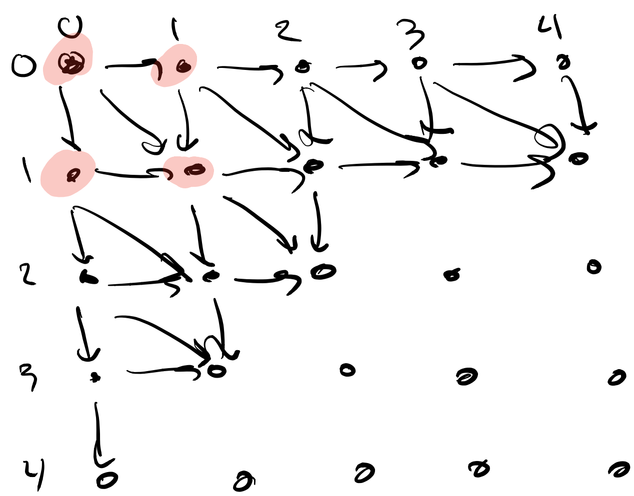
a : b c

b : d e g

d: f g d e f g

g: g

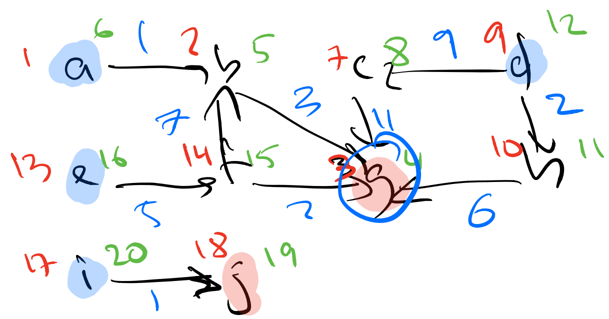
$$LCS[i,j] = \max \begin{cases} LCS[i-1,j] \\ LCS[i,j-1] \\ LCS[i-1,j-1] + \mathbb{1}[S[i]=T[j]] \end{cases}$$



DP on DAGs

Length of the Longest Path t

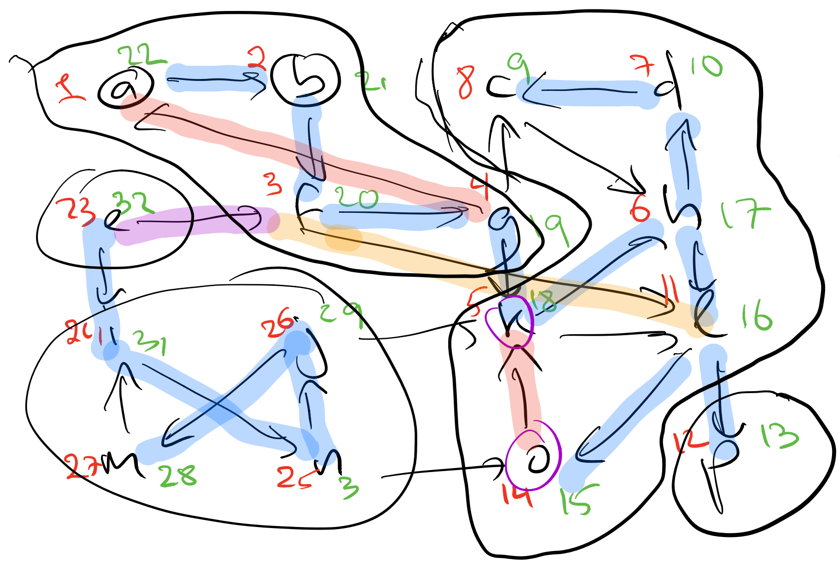
- we have DAG G
- G has weighted edges $\ell(w \to v)$
- target t
- for any vertex s find length of longest (by weight) path to t



target is g

$$LLP(v) = \begin{cases} 0 & \text{if } v = t \\ \max\limits_{v \to w} \ell(v \to w) + LLP(w) & \\ \infty & \text{if } v \neq t \text{ and } \deg(v) = 0 \end{cases}$$

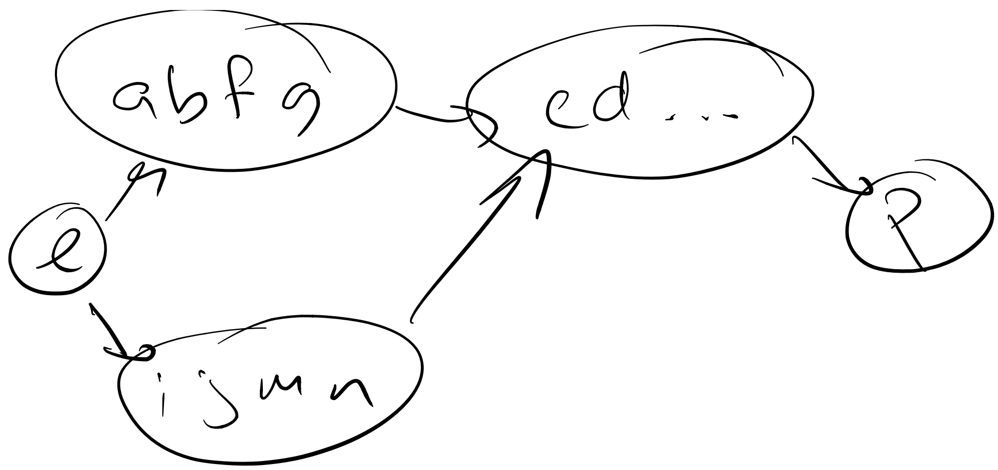
for v in topo sort of G
  compute $LLP(v)$



strongly connected component
SCC
$\forall u, v \in S$
$u \rightsquigarrow v$
$v \rightsquigarrow u$

$$SCC(v) = reach(v, G) + reach(v, G^{-1})$$

$$SCC(G) = \begin{cases} V' = \text{SCC in } G \\ E' = S_1' \to S_2' \\ \quad \text{if } \exists\, v_1 \in S_1 \;\; v_2 \in S_2 \\ \quad v_1 \to v_2 \end{cases}$$


abfg
cd ...
e
ijmn
p