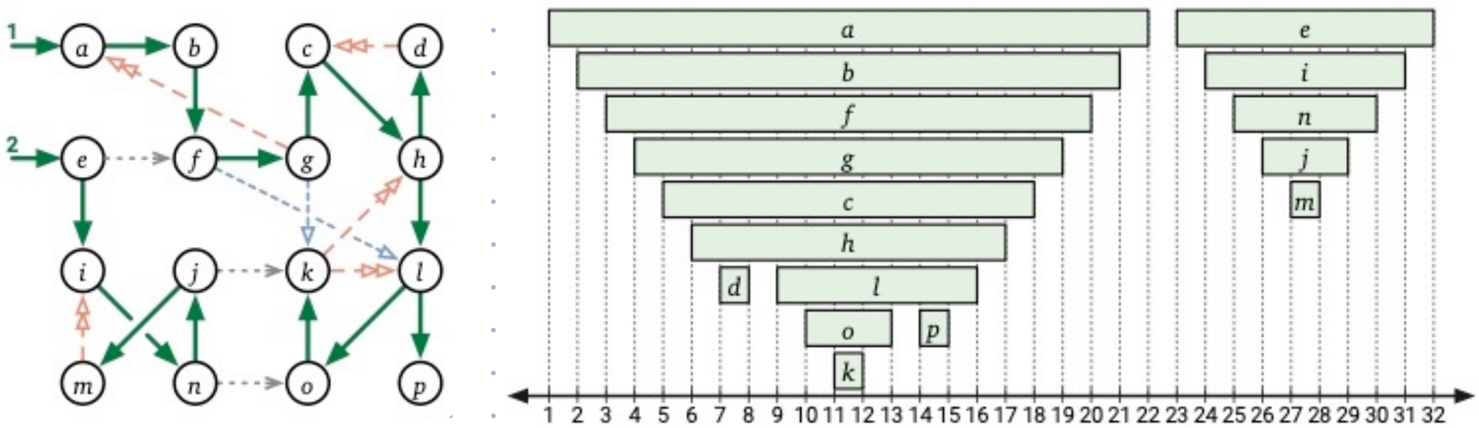
~~1 Modify the graph, apply algorithms as black boxes~~

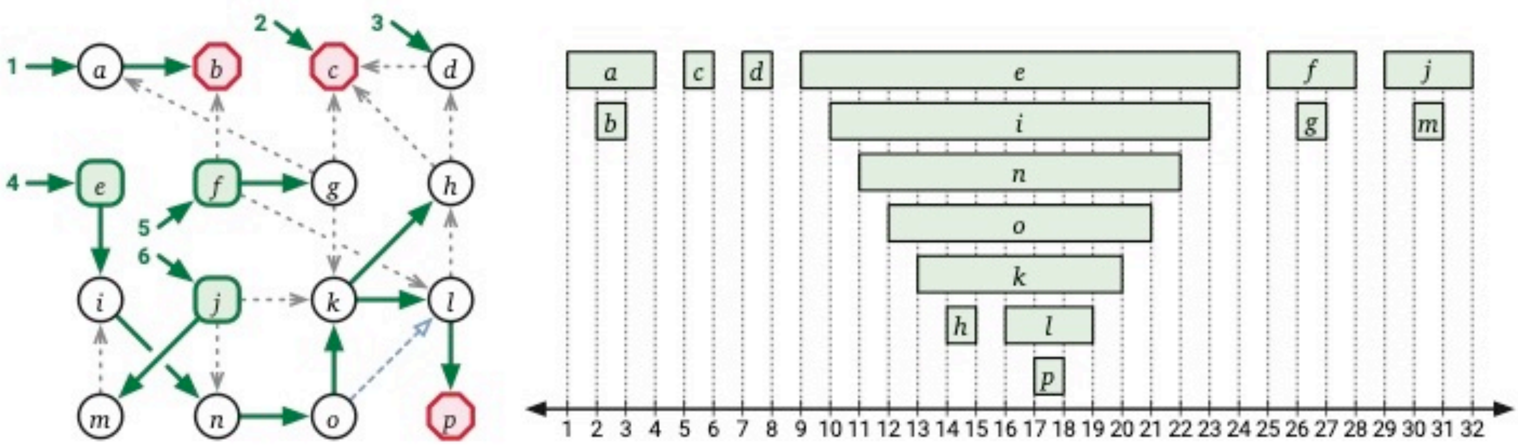
2 Modify algorithms, apply them to original graph !!

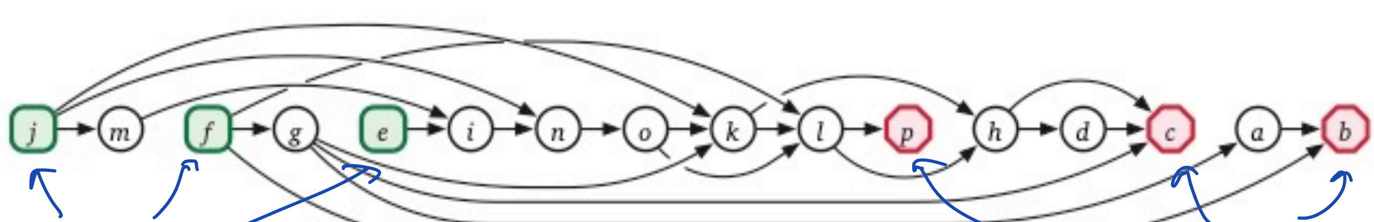
Do this!!

Depth first search



Topological Sort $O(V+E)$





Sources

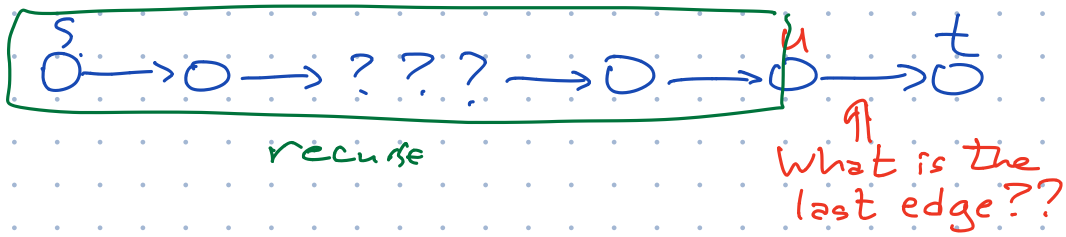
sinks

```
TopologicalSort(G):
    for all vertices v
        v.status ← New
    clock ← V
    for all vertices v
        if v.status = New
            clock ← TopSortDFS(v, clock)
    return S[1..V]
```

```
TopSortDFS(v, clock):
    v.status ← Active
    for each edge v→w
        if w.status = New
            clock ← TopSortDFS(v, clock)
        else if w.status = Active
            fail gracefully
    v.status ← Finished
    S[clock] ← v
    clock ← clock − 1
    return clock
```

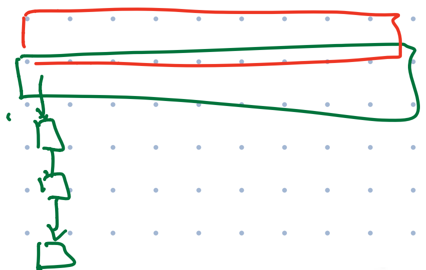
Figure 6.9. Explicit topological sort

Longest Path in a DAG from s to t



$LSP(u)$ = length of ~~longest~~ shortest path from s to u

$$LSP(u) = \begin{cases} 0 & \text{if } u = s \\ \cancel{\max}\ \min\limits_{v \to u} \left(w(v\to u) + LLP(v) \right) & \text{if } u \neq s \end{cases}$$



Memoize into the graph itself
Evaluate in topological order

$$\text{Time} = O\left(\sum_u (1 + \text{indeg}(u))\right) = \boxed{O(V+E)}$$

```
PostProcessDAG(G):
    for all vertices v in postorder
        Process(v)
```

O(V+E)

```
PostProcessDAG(G):
    for all vertices v
        unmark v
    for all vertices v
        if v is unmarked
            PostProcessDagDFS(s)
```

```
PostProcessDagDFS(v):
    mark v
    for each edge v→w
        if w is unmarked
            PostProcessDagDFS(w)
    Process(v)
```

Different function!

Length of the longest path from v to t

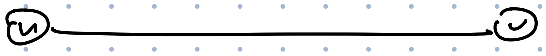
$$LLP(v) = \begin{cases} 0 & \text{if } v = t, \\ \max\left\{\ell(v\to w) + LLP(w) \;\middle|\; v\to w \in E\right\} & \text{otherwise,} \end{cases}$$

```
LONGESTPATH(v, t):
  if v = t
      return 0
  if v.LLP is undefined
      v.LLP ← −∞
      for each edge v→w
          v.LLP ← max{v.LLP, ℓ(v→w) + LONGESTPATH(w, t)}
      return v.LLP
```

These are the same algorithm!!

```
LONGESTPATH(s, t):
  for each node v in postorder
      if v = t
          v.LLP ← 0
      else
          v.LLP ← −∞
          for each edge v→w
              v.LLP ← max{v.LLP, ℓ(v→w) + w.LLP}
  return s.LLP
```

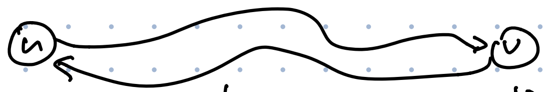
undir:



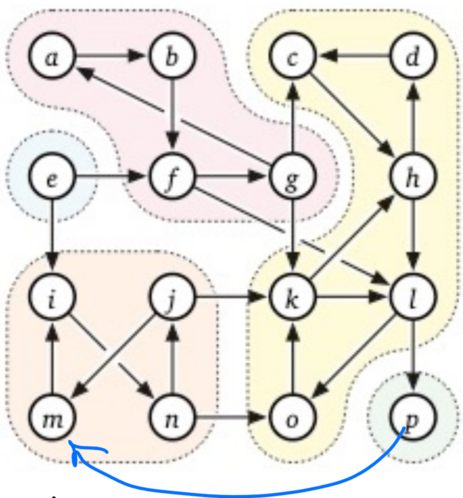
connectivity is symmetrice

directed:

reachability is <u>not</u> sym.

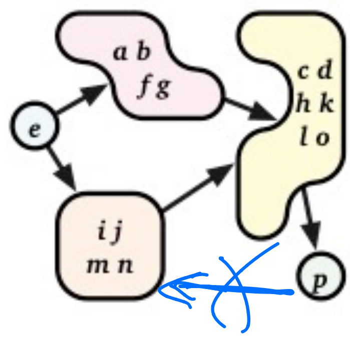


<u>strong</u> connectivity is sym.



strong components

O(V+E) time



Condensation
metagraph
scc(G)

DAG

Strongly connected $\Leftrightarrow$ every vertex can reach every other vertex.

Shortest Paths:

Greedy • Unweighted — BFS $O(V+E)$

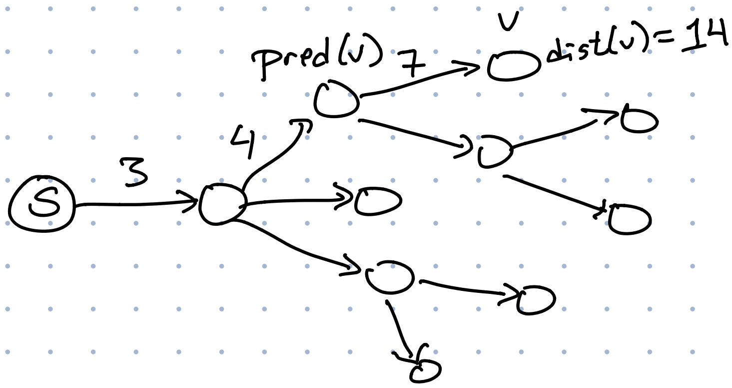
DP • DAG — DFS/top sort $O(V+E)$

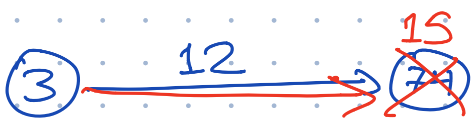
Greedy • Weighted, no neg edges — Dijkstra $O(E \log V)$

DP • Weighted — Bellman-Ford $O(EV)$

Generic strategy Ford (1956)

InitSSSP(s):
  $dist(s) \leftarrow 0$
  $pred(s) \leftarrow$ NULL
  for all vertices $v \neq s$
    $dist(v) \leftarrow \infty$
    $pred(v) \leftarrow$ NULL





Always: $dist(v) \geq$ shortest path from s to v

Relax($u \rightarrow v$):
  $dist(v) \leftarrow dist(u) + w(u \rightarrow v)$
  $pred(v) \leftarrow u$

$u \rightarrow v$ is tense
if $dist(u) + w(u \rightarrow v) < dist(v)$

FordSSSP(s):
  InitSSSP(s)
  while there is at least one tense edge
    Relax any tense edge

When no edges are tense, all dist are correct