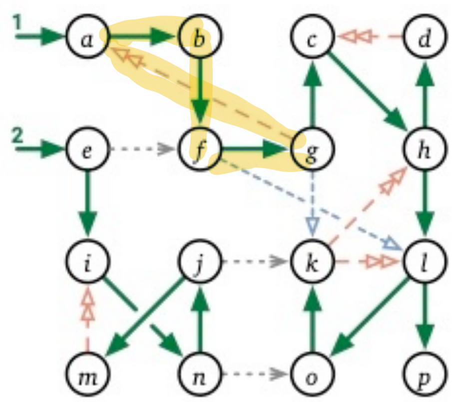
HW9 out — due next Tuesday

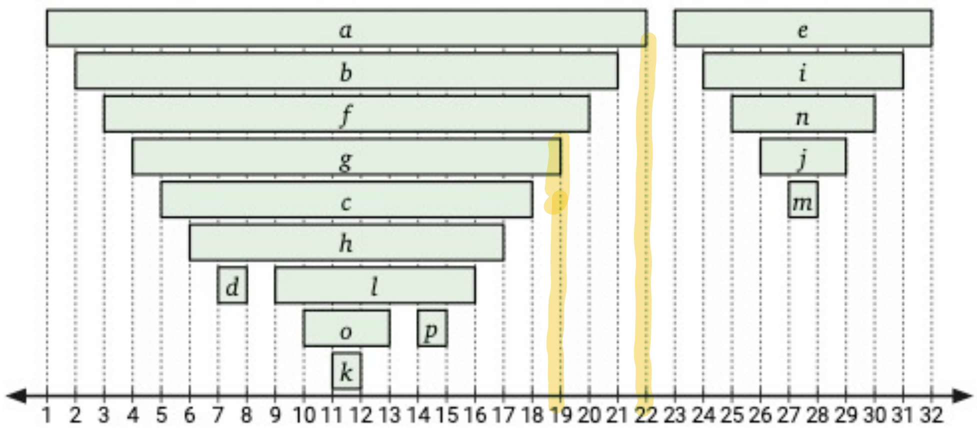
GPS 9 out — due Monday

Midterm 2 in two weeks

Depth-First search

preorder — pushes
postorder — pops

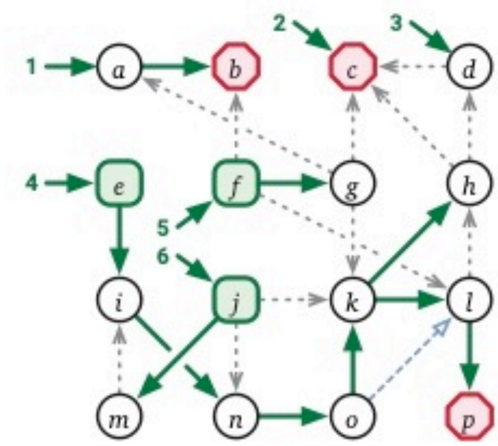


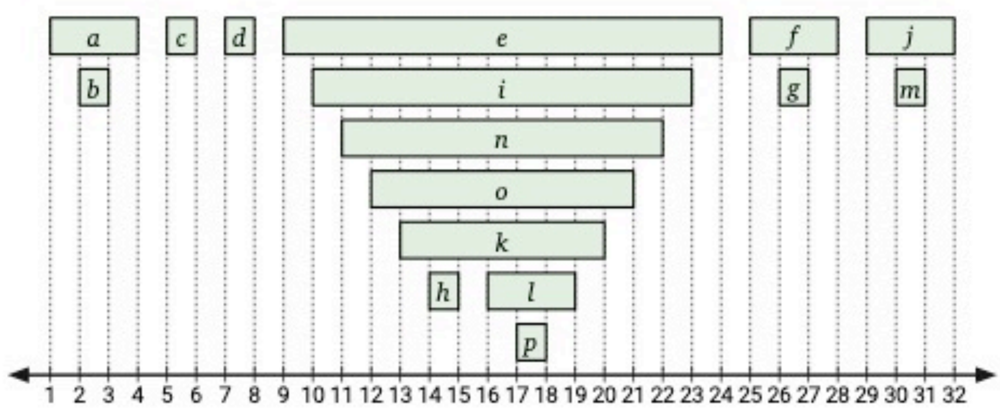


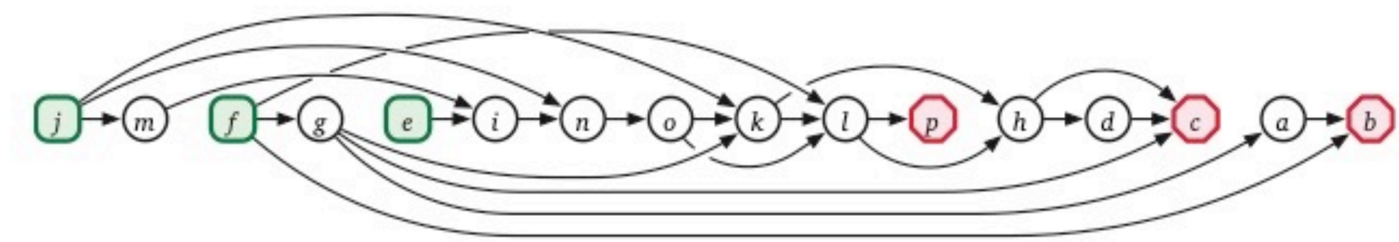
G has a cycle iff $post(u) < post(v)$ for some edge $u \to v$

DAG $\to$ reverse postorder = topological order

$O(V+E)$ time







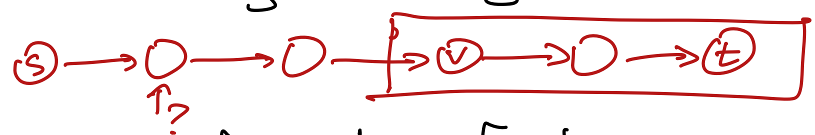
```
TopologicalSort(G):
    for all vertices v
        v.status ← New
    clock ← V
    for all vertices v
        if v.status = New
            clock ← TopSortDFS(v, clock)
    return S[1..V]
```

```
TopSortDFS(v, clock):
    v.status ← Active
    for each edge v→w
        if w.status = New
            clock ← TopSortDFS(w, 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 from s to t in a dag G

Let LLP(v) = length of longest path from v to t

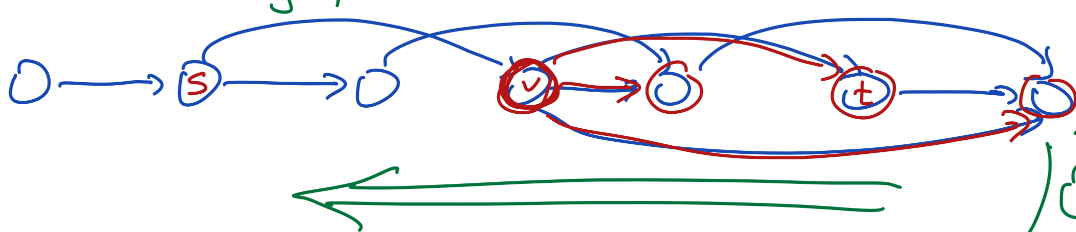


length = sum of lengths of edges

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

max $\emptyset = -\infty$

Memoize into graph v.LLP



$\boxed{O(V+E) \text{ time}}$

```
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
```

Memoized backtracking

SAME?

```
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
```

DP

SAME?

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

```
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)
```

□BOTH|EARTH|AND|SATURN|SPIN|- - -

Define dag $G = (V, E)$

$V = \{0 \dots n\}$

$E = \{ i \to j \mid \text{IsWord}(A[i+1..j]) \}$

Build using O(n^2) calls to IsWord

Is there a path from 0 to n in G?

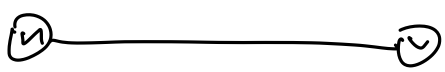
WFS $O(V+E) = O(n^2)$ time

Given a graph G, Find a walk visiting max # unique vertices

DAG $\longrightarrow$ longest path O(V+E) DFS

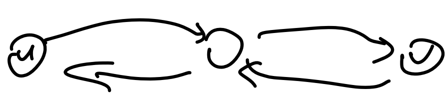
Undirected $\longrightarrow$ largest component O(V+E) WFS

Directed $\longrightarrow$



connected = reach is symmetric

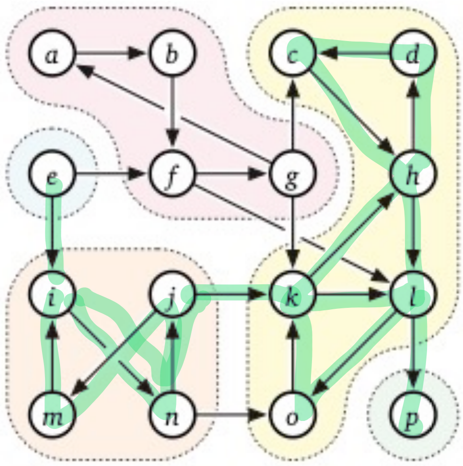
reach is not symmetric

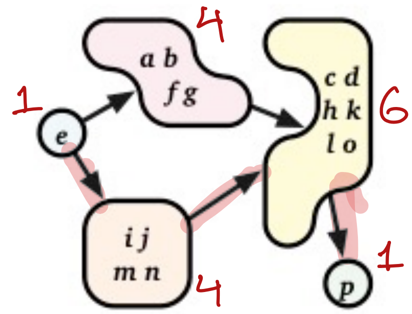


strongly connected is symmetric

Strong components = equiv classes for strong connectivity

Tarjan
Kosaraju-Sharir
$O(V+E)$ time





SCC(G)
condensation
metagraph

DAG!

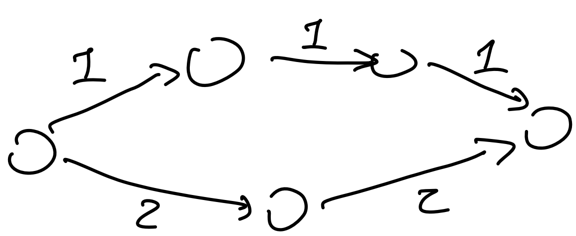
#unique vertices in a walk in G = weight of a path in SCC(G)

max this = max

$O(V+E)$ time

Shortest Paths

Greedy	Unweighted — BFS		$O(V+E)$
DP	DAG — DFS/DP		$O(V+E)$
Greedy	Non-negative weights — Dijkstra		$O(E \log V)$
DP	Arbitrary weights — Bellman-Ford		$O(EV)$



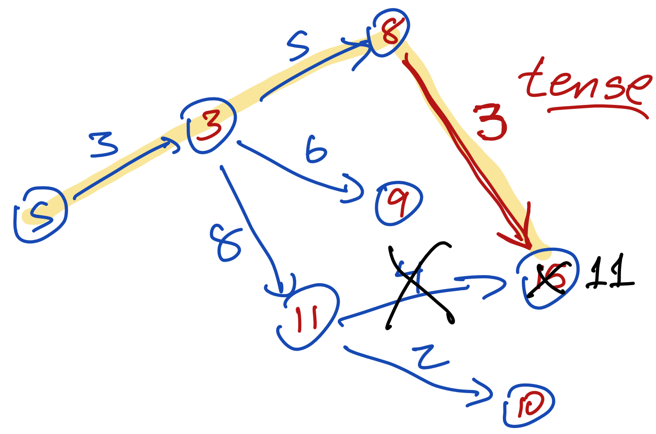
Ford's generic SP algorithm

Every node v stores

$dist(v)$ — estimate of sh. path distance from s to v

$pred(v)$ — parent of v on shortest(?) path from s to v

```
InitSSSP(s):
    dist(s) ← 0
    pred(s) ← Null
    for all vertices v ≠ s
        dist(v) ← ∞
        pred(v) ← Null
```



$u \to v$ is tense if

$$dist(u) + \ell(u \to v) < dist(v)$$

```
Relax(u→v):
    dist(v) ← dist(u) + w(u→v)
    pred(v) ← u
```

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