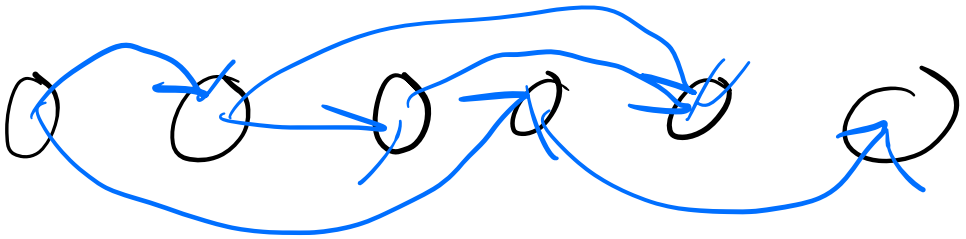
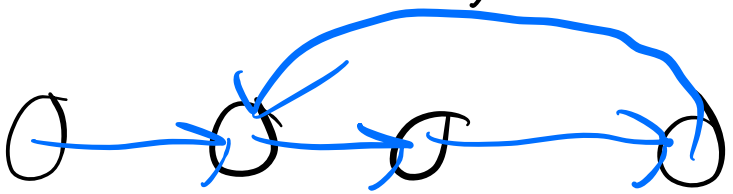
Given directed $G = (V, E)$.

A topological ordering: a total ordering on V where for all $u \to v \in E$, $u \prec v$.



No ordering is G has cycles!



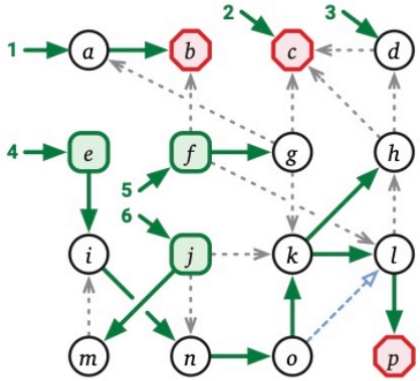
But if G is a dag...

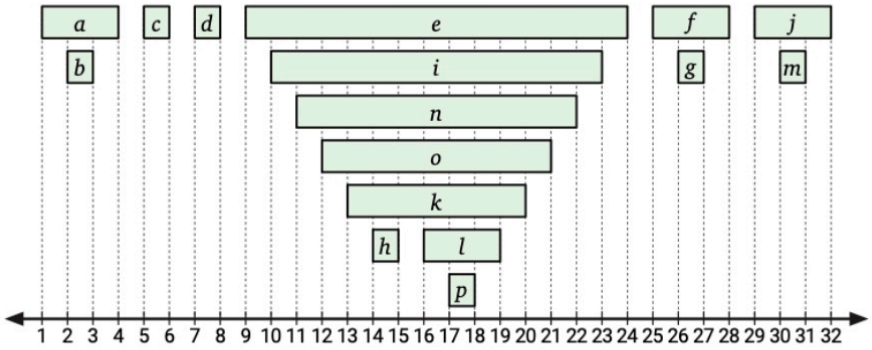
$\Rightarrow$ G has no back edges

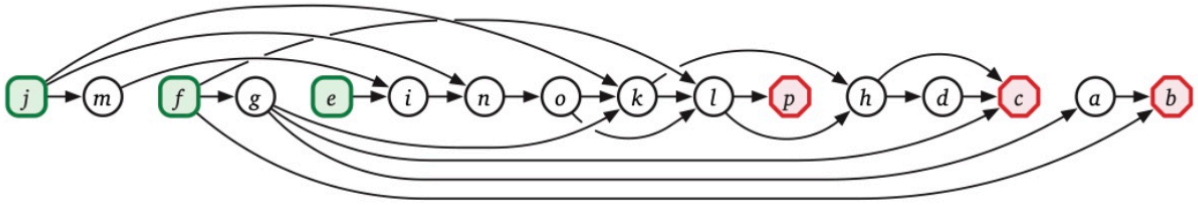
$\Rightarrow$ for $u \to v \in E$, $u.post > v.post$.

$\Rightarrow$ reverse post order is topological!

(G has a topological ordering iff G is a dag.)





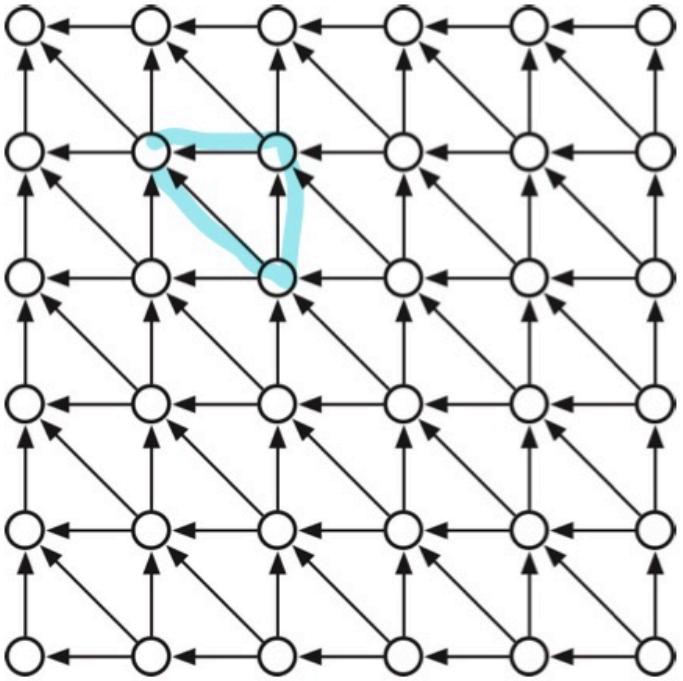



```
TopologicalSort(G):
    clock ← V
    for all vertices v in postorder
        S[clock] ← v
        clock ← clock − 1
    return S[1..V]
```

O(V + E) time

Dependency graph: vertices are subproblems

edge $u \to v$ if u makes a call to v



Edit distance

Must be a dag!

```
MEMOIZE(x):
    if value[x] is undefined
        initialize value[x]

        for all subproblems y of x
            MEMOIZE(y)
            update value[x] based on value[y]
        finalize value[x]
```

```
DFS(v):
    if v is unmarked
        mark v
        PREVISIT(x)
        for all edges v→w
            DFS(w)

        POSTVISIT(x)
```

```
DynamicProgramming(G):
    for all subproblems x in postorder
        initialize value[x]
        for all subproblems y of x
            update value[x] based on value[y]
        finalize value[x]
```

Given a directed $G = (V, E)$ with lengths $\ell : E \to \mathbb{R}$ & two vertices $s, t \in V$.

Goal: Want length of <u>longest</u> path from s to t. (Assume G is a dag).

$LLP(v)$: length of longest path from v to t ($-\infty$ if no such path exists)

$$LLP(v) = \begin{cases} 0 & \text{if } v = t \\ \max_{v \to w} \{\ell(v \to w) + LLP(w)\} & \text{otherwise} \end{cases}$$

✓ $-\infty$ if no $v \to w$ & $v \neq t$

(only works if G is a dag!)

Dependency graph for day G is G.

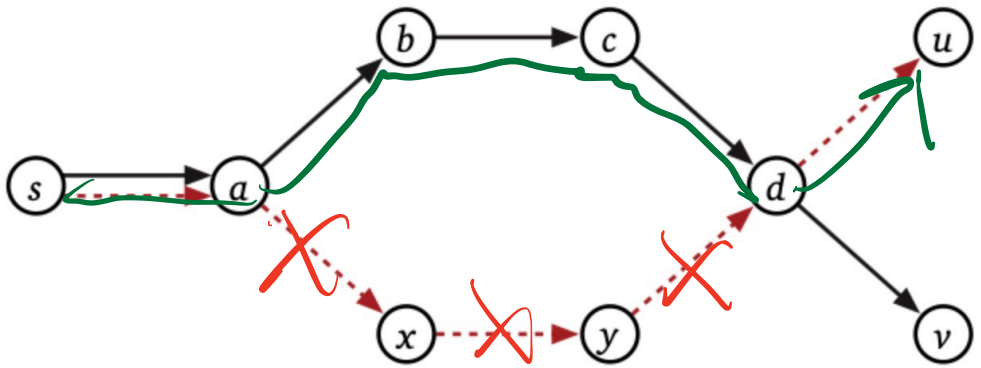
```
LONGESTPATH(s, t):
  for each node v in postorder
      if v = t
          v.LLP ← 0
      else
          v.LLP ← −∞
          for each edge v→w
              v.LLP ← max{v.LLP, ℓ(v→w) + w.LLP}
  return s.LLP
```

Time: $O(V+E)$

Given directed $G=(V,E)$ with edge weights $w: E \to \mathbb{R}^+$ and a single vertex $s \in V$.

Want shortest paths from s.

Single source shortest path (SSSP) problem.



Shortest paths form a tree rooted at s.

Maintain for each vertex v

$v.dist$: length of some (s,v)-path (or ∞)

$v.pred$: predecessor of the path of length $v.dist$ (or Null if $v.dist = \infty$ or $v = s$).

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

Edge $u \to v$ is tense

if $u.dist + w(u \to v) < v.dist$

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

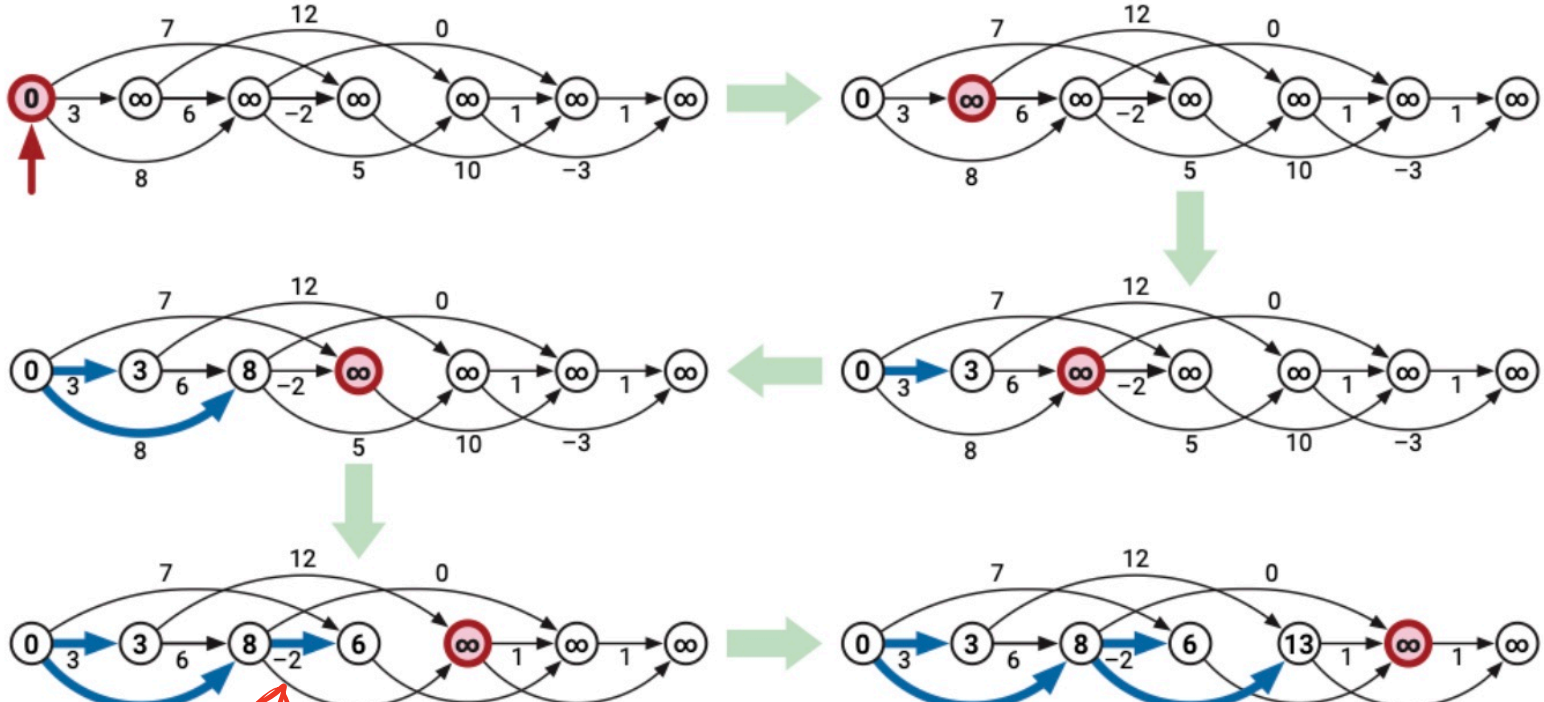
```
FordSSSP(s):
    InitSSSP(s)
    while there is at least one tense edge
        Relax any tense edge
```

Days: dist(v): distance from s to v

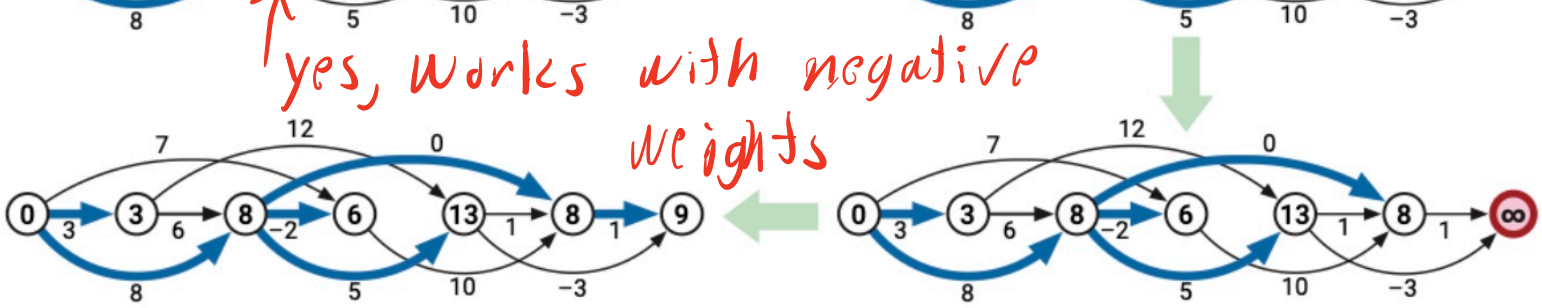
$$dist(v) = \begin{cases} 0 & \text{if } v = s \\ \min_{u \to v} (dist(u) + w(u \to v)) & \text{otherwise} \end{cases}$$

```
DAGSSSP(s):
  for all vertices v in topological order
    if v = s
      v.dist ← 0
    else
      v.dist ← ∞
      for all edges u→v
        if v.dist > u.dist + w(u→v)          ⟨⟨if u→v is tense⟩⟩
          v.dist ← u.dist + w(u→v)           ⟨⟨relax u→v⟩⟩
```

```
DAGSSSP(s):
  INITSSSP(s)
  for all vertices v in topological order
    for all edges u→v
      if u→v is tense
        RELAX(u→v)
```



yes, works with negative weights



$O(V+E)$

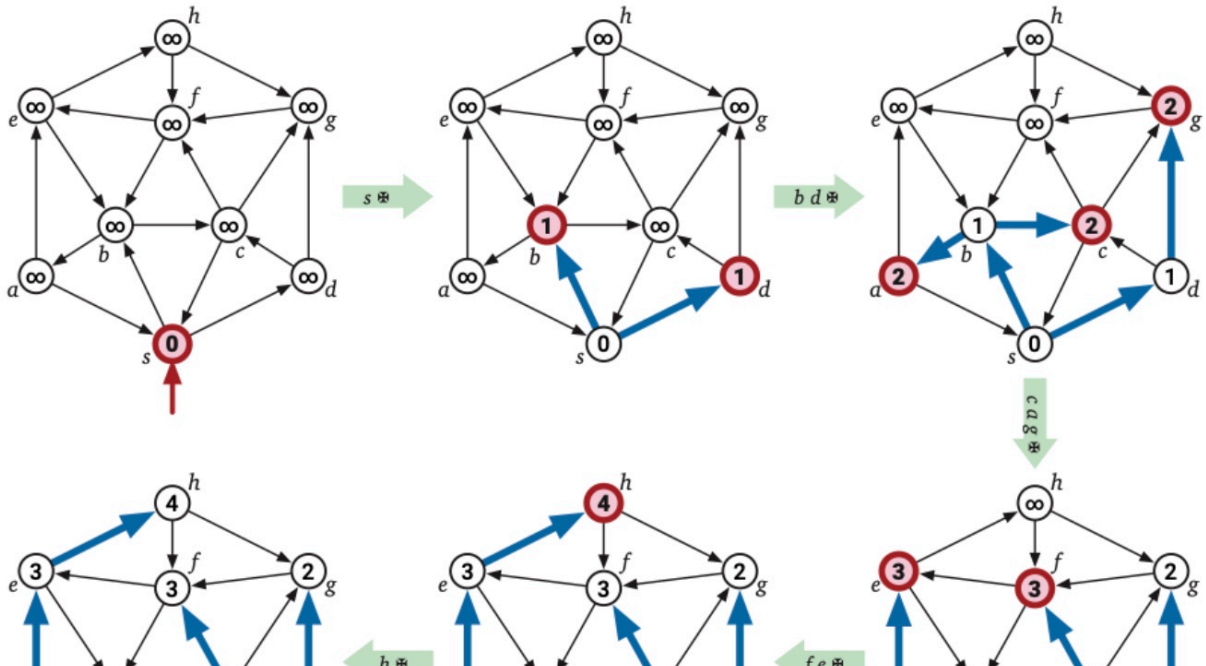
Unweighted graphs (may not be a dag)

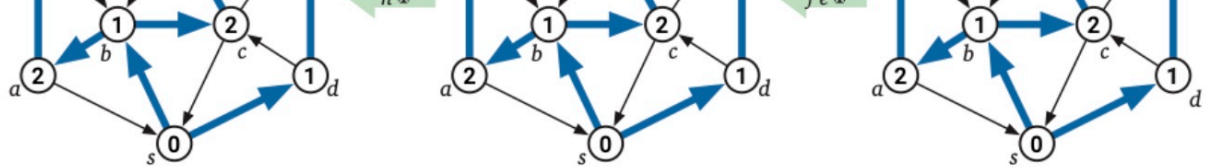
For proof only

```
BFS(s):
  InitSSSP(s)
  Push(s)
  while the queue is not empty
      u ← Pull( )
      for all edges u→v
          if v.dist > v.dist + 1          ⟨⟨if u→v is tense⟩⟩
              v.dist ← u.dist + 1         ⟨⟨relax u→v⟩⟩
              v.pred ← u
              Push(v)
```

```
BFSWithToken(s):
  InitSSSP(s)
  Push(s)
  Push(✻)                                 ⟨⟨start the first phase⟩⟩
  while the queue contains at least one vertex
      u ← Pull( )
      if u = ✻
          Push(✻)                         ⟨⟨start the next phase⟩⟩
      else
          for all edges u→v
              if v.dist > u.dist + 1      ⟨⟨if u→v is tense⟩⟩
                  v.dist ← u.dist + 1     ⟨⟨relax u→v⟩⟩
                  v.pred ← u
                  Push(v)
```

Phase i begins when we push ✻ for the ith time.





Claim: At the end of the ith phase,
either $v.dist = \infty$ or $v.dist \leq i$.
Also v is in the queue iff $v.dist = i$.

=> Every vertex in queue at most once.
=> $O(V+E)$ time

=> We also set v to correct distance i during phase i.

Summary: days: DagSSSP in $O(V+E)$
unweighted: BFS in $O(V+E)$