GPS 9 : Mon Nov 2

HW 9 : Tue Nov 3

Midterm 2 : Mon Nov 9, 7pm - 9pm

Conflict Midterm 2 : Tue Nov 10

Ford:

InitSSSP(s):

$s.dist \leftarrow 0$

$s.pred \leftarrow$ Null

for all vertices $v \neq s$

$v.dist \leftarrow \infty$

$v.pred \leftarrow$ Null

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

$u \to v$

Relax($u \to v$):

$v.dist \leftarrow u.dist + w(u \to v)$

$v.pred \leftarrow u$

FordSSSP(s):

InitSSSP(s)

while there is at least one tense edge $u \to v$

Relax($u \to v$)

```
Dijkstra(s):
    InitSSSP(s)
    Insert(s, 0)
    while the priority queue is not empty
        u ← ExtractMin()
        for all edges u → v
            if u → v is tense
                Relax(u → v)
                if v is in the priority queue
                    DecreaseKey(v, v.dist)
                else
                    Insert(v, v.dist)
```

With positive weights:

$O(|E| \log |V|)$.

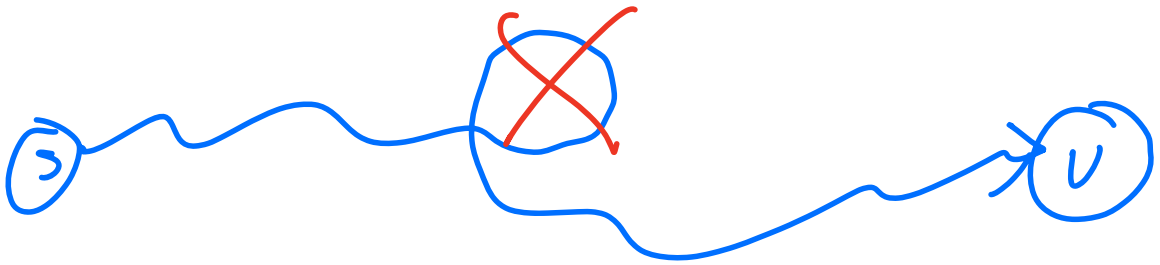
If weights all equal 1:

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

$v.dist \leftarrow u.dist + 1$
$v.pred \leftarrow u$
Push(v)

What about negative edge weights?

Ford SSSP is really looking for shortest walks...



Ok if shortest walks are paths.

No shortest walk if s reaches a negative cycle.

But if no negative cycles, then there are

shortest walks & they are paths.

Lemma: Assuming G has no negative weight cycles, FordSSSP will find distances & shortest paths.

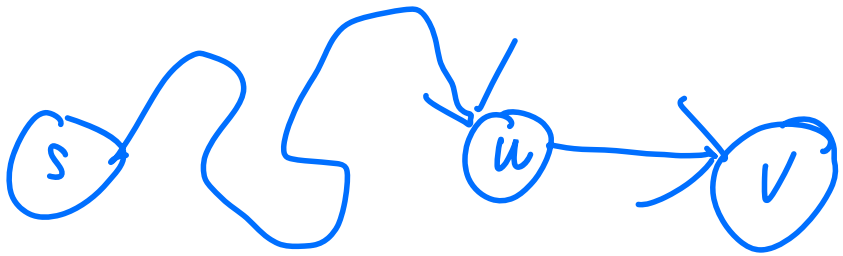
(Our version of) Dijkstra is still correct, but it could be slow with negative edges.

Fast for very few negative edges, though!

Dags $G = (V, E)$

Recursive approach...

distance(v): distance from s to v



$$distance(v) = \begin{cases} 0 & \text{if } v = s \\ \min_{u \to v} \left(distance(u) + w(u \to v) \right) & \text{o.w.} \end{cases}$$

Dependency graph is also a dag (reversal of G)

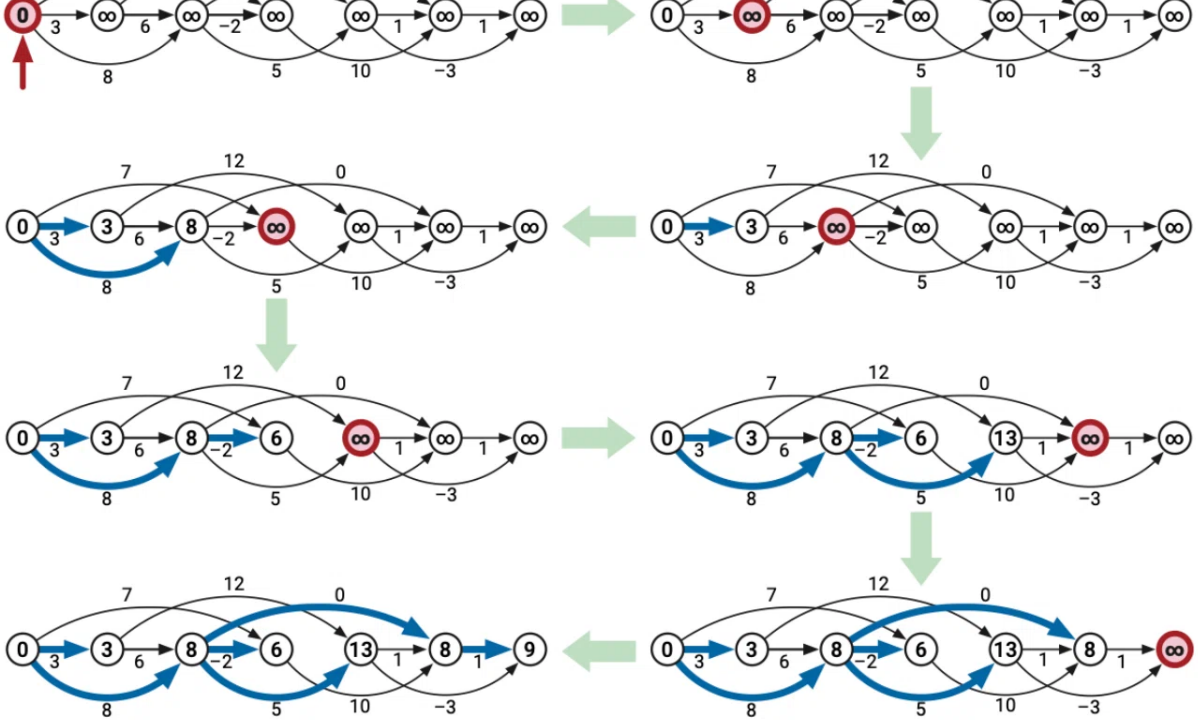
Solve subproblems in topological order!

```
DagSSSPDP(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)
                    v.dist ← u.dist + w(u → v)
```

Time: $O(|V| + |E|)$

```
DagSSSP(s):
    InitSSSP(s)
    for all vertices v in topological order
        for all edges u → v
            if u → v is tense
                Relax(u → v)
```





IS doing dynamic programming on a dag with dependencies on outgoing edges..

Ex: Given $G = (V, E)$, weights $w: E \to \mathbb{R}$, & $t \in V$. Find longest path lengths to t.

longest(u): longest path length from u to t

$$longest(u) = \begin{cases} 0 & \text{if } u = t \\ \max_{u \to v} \left(w(u \to v) + longest(v) \right) & \text{o.w.} \end{cases}$$

Solve in postorder.

```
DagLongestPaths(t):
    for all vertices u in postorder
        if u = t
            u.longest ← 0
        else
            u.longest ← ∞
            for all edges u → v
                if u.longest < w(u → v) + v.longest
                    u.longest ← w(u → v) + v.longest
```

$O(|V| + |E|)$

All dynamic programming is solving subproblems in postorder of the dependency graph.

Now, G has cycles, but none have negative weight.

Today: Via reduction to DagSSSP.

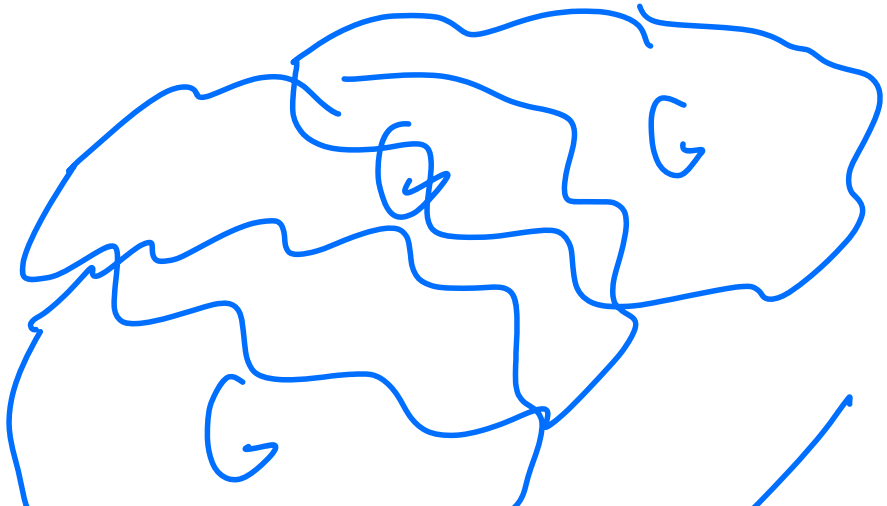
Build a dag $G' = (V', E')$ with weights $w' : E' \to \mathbb{R}$.

G' will help us compute shortest paths will different numbers of edges.

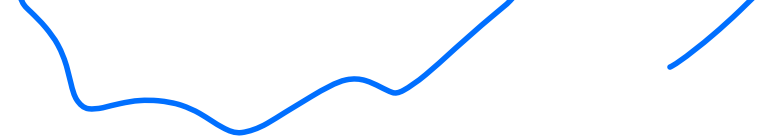
$V' := \{(v, i) \mid v \in V,\ i \in \{0, \ldots, |V|-1\}\}$

$E' := \{(u, i-1) \to (v, i) \mid u \to v \in E,\ i \in \{1, \ldots, |V|-1\}\}$

$w'((u, i-1) \to (v, i)) := w(u \to v)$ for all $u \to v \in E$, $i \in \{1, \ldots, |V|-1\}$



$|V|$ copies of G



G' is a dag, because second component of vertices always increases.

Min length of any (s,v)-walk in G with exactly i edges is distance from $(s,0)$ to (v,i).

Distance from s to v in G is

min over all $i \in \{0, \ldots, |V|-1\}$ of distance from $(s,0)$ to (v,i) in G'.

So run DagSSSP((s,0)) & find those mins for all $v \in V$.

$|V'| = |V|^2$

$|E'| = (|V|-1)|E| = O(|E||V|)$

So $O(|V|^2 + |E||V|) = O(|E||V|)$ time

(assuming simple & connected)

If s can reach a negative weight cycle, an edge will be tense.

Run this algo & check if any edge is

tense to detect negative cycles in $O(|E||V|)$ time.

Can call this "Bellman-Ford"