All-pairs shortest paths:

Given an edge-weighted directed graph $G=(V,E)$.

Want to compute $dist(u,v)$, the distance from u to v $\forall\, u,v \in V$

Today: Assume no negative cycles; there may be negative edges.

```
OBVIOUSAPSP(V, E, w):
    for every vertex s
        dist[s, ·] ← SSSP(V, E, w, s)
```

What kind of graph:

Unweighted: BFS

$V \cdot O(E) = O(VE) = O(V^3)$

A DAG:

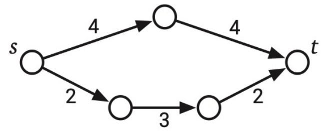
$V \cdot O(E) = O(VE) = O(V^3)$

Non-neg. weights: Dijkstra

$V \cdot O(E \log V) = O(VE \log V)$
$= O(V^3 \log V)$

O.W.: Bellman-Ford

$V \cdot O(VE) = O(V^2 E) = O(V^4)$



A 2 to each edge to change shortest path. :)

So can't just add large numbers to make edges non-negative.

Johnson:

Give each vertex v a price $\pi(v)$.

Define new weights

$$w'(u \to v) = \pi(u) + w(u \to v) - \pi(v)$$

tax

gift

$\forall u, v \in V,$

all u,v-paths $u \rightsquigarrow v$

have same weight change

$$w(v_1 \to v_2 \to v_3) = \pi(v_1) + w(v_1 \to v_2) - \pi(v_2) + \pi(v_2) + w(v_2 \to v_3) - \pi(v_3)$$

$$w'(u \rightsquigarrow v) = \pi(u) + w(u \rightsquigarrow v) - \pi(v)$$

$\Rightarrow$ shortest u,v-path is still shortest

Algorithm:

Compute $dist(s, \cdot)$ using Bellman-Ford.

$$w'(u \to v) := dist(s,u) + w(u \to v) - dist(s,v)$$

no tense edges $\Rightarrow w'(\cdot) \geq 0$

run many Dijkstras

Use a new vertex

s connected everywhere with 0 weight edges as source

Bellman-Ford

```
JohnsonAPSP(V, E, w):
    ⟨⟨Add an artificial source⟩⟩
    add a new vertex s
    for every vertex v
        add a new edge s→v
        w(s→v) ← 0
    ⟨⟨Compute vertex prices⟩⟩
    dist[s,·] ← BellmanFord(V, E, w, s)
    if BellmanFord found a negative cycle
        fail gracefully
    ⟨⟨Reweight the edges⟩⟩
    for every edge u→v ∈ E
        w′(u→v) ← dist[s,u] + w(u→v) − dist[s,v]
    ⟨⟨Compute reweighted shortest path distances⟩⟩
    for every vertex u
        dist′[u,·] ← Dijkstra(V, E, w′, u)
    ⟨⟨Compute original shortest-path distances⟩⟩
    for every vertex u
        for every vertex v
            dist[u,v] ← dist′[u,v] − dist[s,u] + dist[s,v]
```

$O(V)$

$O(VE)$

$O(E)$

$O(VE\log V)$

$O(V^2)$

Total: $O(VE\log V) = O(V^3\log V)$

(Bellman-Ford $|V|$ times was $O(V^4)$)

Good when $|E|$ is small.

Dynamic Programming:

$$dist(u,v) = \begin{cases} 0 & \text{if } u = v \\ \min\limits_{x \to v} \left(dist(u,x) + w(x \to v) \right) & \text{otherwise} \end{cases}$$

$dist(u,v,\ell)$: length of shortest u,v-path with $\leq \ell$ edges.

$$dist(u,v,\ell) = \begin{cases} 0 & \text{if } \ell = 0 \text{ and } u = v \\ \infty & \text{if } \ell = 0 \text{ and } u \neq v \\ \min \left\{ \begin{array}{c} dist(u,v,\ell-1) \\ \min\limits_{x \to v} \left(dist(u,x,\ell-1) + w(x \to v) \right) \end{array} \right\} & \text{otherwise} \end{cases}$$

$dist(u,v) = dist(u,v,|V|-1)$

($\leq |V|-1$ edges in any path)

Shimbel ['43]

```
SHIMBELAPSP(V, E, w):
  for all vertices u
    for all vertices v
      if u = v
        dist[u, v, 0] ← 0
      else
        dist[u, v, 0] ← ∞
  for ℓ ← 1 to V − 1
    for all vertices u
      for all vertices v ≠ u
        dist[u, v, ℓ] ← dist[u, v, ℓ − 1]
        for all edges x→v
          if dist[u, v, ℓ] > dist[u, x, ℓ − 1] + w(x→v)
            dist[u, v, ℓ] ← dist[u, x, ℓ − 1] + w(x→v)
```

Touches each ℓ, u, edge triple at most once.

Time: $O(V^2E) = O(V^4)$

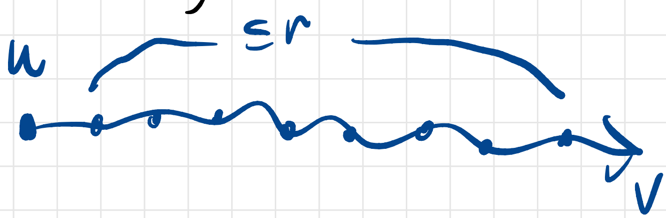
Guess middle vertex & use $\log(V)$ values of ℓ for $O(V^3 \log V)$ time.

Different Third Variable:

Number the vertices arbitrarily from 1 to $|V|$.

$\pi(u,v,r) :=$ shortest path from u to v where all <u>intermediate</u> vertices have number $\leq r$.

$\text{dist}(u,v,r) = w(\pi(u,v,r))$



$\text{dist}(u,v) = \text{dist}(u,v,|V|)$

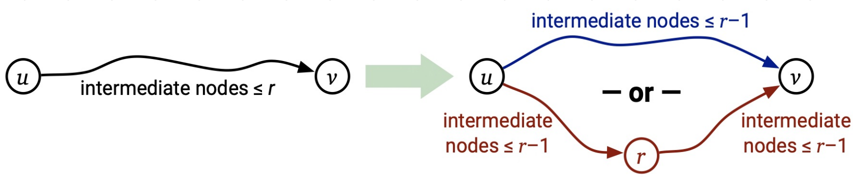
$\pi(u,v,0) = u \to v$ if $u \to v \in E$

$\emptyset$ o.w.

(let's just say $u \to v$ always exists by setting $w(u\to v) := \infty$ if not

$$dist(u,v,0) = w(u \to v)$$

$\pi(u,v,r)$ for $r \geq 1$:



$$dist(u,v,r) = \begin{cases} w(u\to v) & \text{if } r = 0 \\ \min \left\{ \begin{array}{c} dist(u,v,r-1) \\ dist(u,r,r-1) + dist(r,v,r-1) \end{array} \right\} & \text{otherwise} \end{cases}$$

Time: $O(V^3)$

```
KleeneAPSP(V, E, w):
    for all vertices u
        for all vertices v
            dist[u, v, 0] ← w(u→v)

    for r ← 1 to V
        for all vertices u
            for all vertices v
                if dist[u, v, r − 1] < dist[u, r, r − 1] + dist[r, v, r − 1]
                    dist[u, v, r] ← dist[u, v, r − 1]
                else
                    dist[u, v, r] ← dist[u, r, r − 1] + dist[r, v, r − 1]
```

```
FloydWarshall(V, E, w):
    for all vertices u
        for all vertices v
            dist[u, v] ← w(u→v)

    for all vertices r
        for all vertices u
            for all vertices v
                if dist[u, v] > dist[u, r] + dist[r, v]
                    dist[u, v] ← dist[u, r] + dist[r, v]
```

$O(V^3)$ time

$O(V^2)$ space