All-pairs Shortest Paths

Given directed graph $G=(V,E)$ with edge weights $w: E \to \mathbb{R}$.

Goal: For all $u, v \in V$, compute $dist(u,v)$: the distance from u to v.

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

$O(V^3)$ time if we can use BFS, DAG alg, or Dijkstra (with non-neg weights)

Bellman Ford $|V|$ times

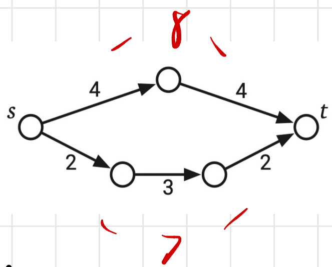
$$V \cdot O(VE) = O(V^2 E)$$
$$= O(V^4)$$

Can we get $O(V^3)$ with some neg. weight edges?

We'll assume no negative weight cycles.

Reweighting:

What if we add a large value to all weights so they aren't negative?



If we add two to all edges...

Top gets $8+4=12$

Bottom $7+6=13$

The new shortest path is wrong!

Let's assign a <u>price</u> to each $\pi : V \to \mathbb{R}$

Set $w'(u \to v) :=$

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

tax to leave u & gift to enter v.

Consider any walk

$$u \rightsquigarrow v := (u = v_1 \to v_2 \to \dots \to v_k = v)$$

$$w'(u \rightsquigarrow v) = \pi(v_1) + w(v_1 \to v_2) - \pi(v_2)$$
$$+ \pi(v_2) + w(v_2 \to v_3) - \pi(v_3)$$
$$+ \dots$$

$$+ \pi(v_{k-1}) + w(v_{k-1} \to v_k) - \pi(v_k)$$

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

Shortest u,v-path under w is shortest under w'.

We want non-neg. weights.

$$0 \le w'(u \to v) = \pi(u) - w(u \to v) + \pi(v)$$

$$\Leftrightarrow$$

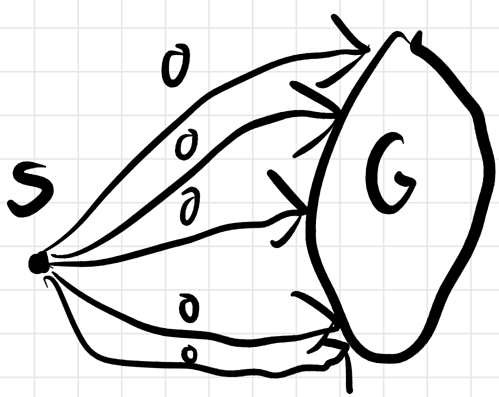
$$\pi(v) \le \pi(u) - w(u \to v)$$

relaxed edge wrt
distances π

So use distances from
some vertex as π.

Johnson's algorithm

Start with a new vertex



<u>JohnsonAPSP(V, E, w) :</u>

⟨⟨*Add an artificial source*⟩⟩
add a new vertex s
for every vertex v
    add a new edge $s \to v$
    $w(s \to v) \leftarrow 0$ $\quad O(E)$

⟨⟨*Compute vertex prices*⟩⟩
$dist[s, \cdot] \leftarrow$ BellmanFord(V, E, w, s)
if BellmanFord found a negative cycle
    fail gracefully $\quad O(VE)$

⟨⟨*Reweight the edges*⟩⟩
for every edge $u \to v \in E$
    $w'(u \to v) \leftarrow dist[s, u] + w(u \to v) - dist[s, v]$ $\quad O(E)$

⟨⟨*Compute reweighted shortest path distances*⟩⟩
for every vertex u
    $dist'[u, \cdot] \leftarrow$ Dijkstra(V, E, w', u) $\quad O(V^2 \log V + VE)$

⟨⟨*Compute original shortest-path distances*⟩⟩
for every vertex u
    for every vertex v
        $dist[u, v] \leftarrow dist'[u, v] - dist[s, u] + dist[s, v]$ $\quad O(V^2)$

Time: $O(V^2 \log V + VE)$

$= O(V^3)$ (Fib. heaps)

$O(V^3 \log V)$ (binary heaps)

Dynamic Programming

"obvious" recurrence

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

Can't use for recursion if there is a cycle!

We need a parameter to get smaller!

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

$$dist(u,v,\ell) = \begin{cases} 0 & \text{if } \ell = 0 \,\&\, u = v \\ \infty & \text{if } \ell = 0 \,\&\, u \neq v \\ \min \begin{cases} dist(u, v, \ell-1), \\ \min_{x \to v} \left(dist(u, x, \ell-1) + w(x \to v) \right) \end{cases} & \text{o.w.} \end{cases}$$

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

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)
```

Time: Each edge used once per ℓ & u.

$$\underbrace{O(V)}_{\ell} \cdot O(V) \cdot O(E)$$

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

Basically $|V|$ runs of Bellman-Ford!

Idea 1:

Try to guess middle vertex x of path & use

$$\text{dist}(u, x, \ell/2) + \text{dist}(x, u, \ell/2)$$

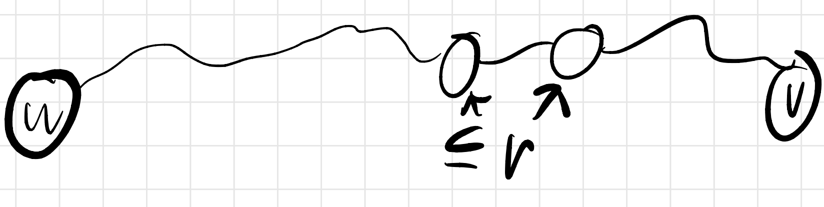
Use $O(\log V)$ values of ℓ.

Total time $O(V^3 \log V)$

Idea 2: Limit which vertices can appear in path instead of # edges.

Arbitrarily number vertices 1 through $|V|$.

$\pi(u, v, r)$: the shortest path from u to v using <u>intermediate</u> vertices from $\{1, 2, \ldots, r\}$.



$dist(u, v, r)$: length of $\pi(u, v, r)$.

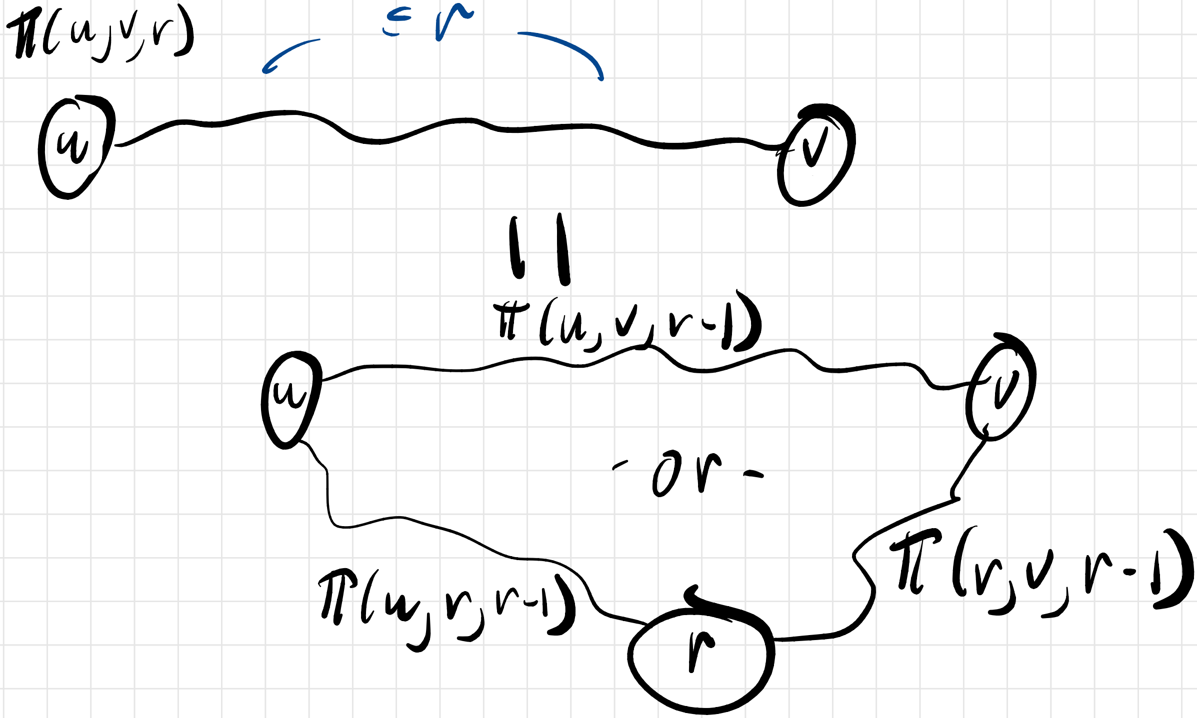
$\pi(u, v, |V|)$ is true shortest path from u to v.

=> $dist(u, v) = dist(u, v, |V|)$

$\pi(u, v, 0)$ is either $u \to v$ or is undefined

->

$$dist(u, v, 0) = \begin{cases} w(u \to v) & \text{if } u \to v \in E \\ \infty & \text{o.w.} \end{cases}$$



$$
dist(u,v,r) = \begin{cases} w(u \to v) & \text{if } r = 0,\ u \to v \in E \\ \infty & \text{if } r = 0,\ u \to v \notin E \\ \min \begin{cases} dist(u,v,r-1), \\ dist(u,r,r-1) + dist(r,v,r-1) \end{cases} & \end{cases}
$$

$|V| \cdot |V| \cdot (|V|+1) = O(V^3)$ subproblems

$O(1)$ per

$O(V^3)$ time total

$O(V^3)$ space

```
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]
```

($w(u \to v) = \infty$ if $u \to v \notin E$)

Cleanup: No need to rember all dist(u, v, r). Just best dist(u, v, ·) found so far.

Don't need to know numbers.
Just loop through all vertices
r.

FloydWarshall(V, E, w):
for all vertices u
  for all vertices v
    $dist[u,v] \leftarrow w(u \rightarrow v)$

for all vertices r
  for all vertices u
    for all vertices v
      if $dist[u,v] > dist[u,r] + dist[r,v]$
        $dist[u,v] \leftarrow dist[u,r] + dist[r,v]$

$O(V^3)$ time

$O(V^2)$ space