28/03/2022

All pairs shortest path.

Goal: To find Shortest path between every pair of vertices.

An immediate algo: $G = (V, E)$, $|V| = n$, $|E| = m$

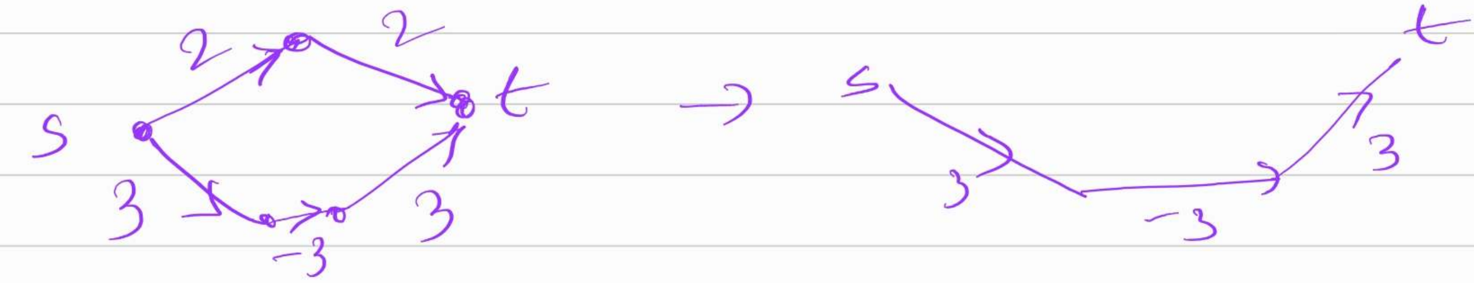
→ Run single source shortest path algo. from every vertex

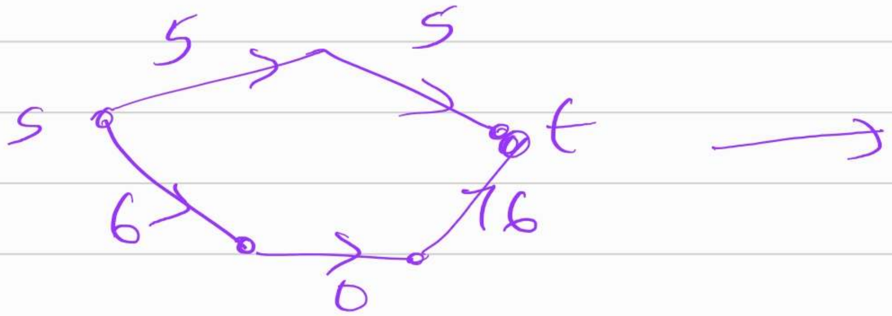
if non negative edge weights. then run Dijkstra n times.

total runtime. $O(n \cdot m \log n)$
$= O(n^3 \log n)$

if negative edge weights then run Bellman-Ford n times.
total runtime. $O(n \cdot m \cdot n) = O(n^4)$

Jhonson's Algorithm:





Shortest path changes after translation.

$$w : E \to \mathbb{R}$$

$$h : V \to \mathbb{R}.$$

$$\hat{w} : E \to \mathbb{R}$$

$$\hat{w}(u \to v) := h(u) + w(u \to v) - h(v)$$

$$\hat{w}(u \to v \to w \to t)$$

$$= h(u) + w(u \to v) - \cancel{h(v)}$$

$$\cancel{h(v)} + w(v \to w) - \cancel{h(w)}$$

$$\cancel{h(w)} + w(w \to t) - h(t)$$

$$= \underline{h(u)} + w(\text{path}) - \underline{h(t)}$$

So this implies that every path between $u \rightsquigarrow t$ is translated by the same amount $h(u) - h(t)$

Q. Is the shortest path preserved between any two vertices by this translation? YES!

<u>How to find $h : V \to \mathbb{R}$?</u>

i.e. an h that makes every edge weight non-negative under $\hat{w}$.

s.t. $\exists$ a path from s to every vertex in the graph.

Define $h(v)$:= length of the shortest path from s to v.

$:= d(s,v)$

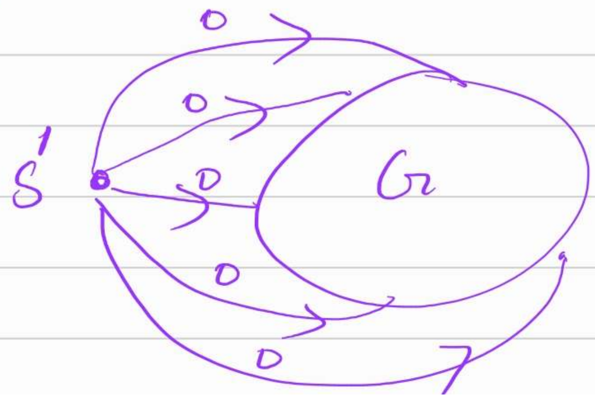
$$\hat{\omega}(u \to v) := \underline{h(u)} + \omega(u,v) - \underline{h(v)}$$

$$= d(s,u) + \omega(u,v) - d(s,v)$$

≥ 0 ?

So $\hat{\omega}(e) \geq 0$ for all edges e in the graph.

if no such s exist, then



add a new vertex s' to G.

add edge (s',v) with weight 0.

In particular no incoming edges to s'.

now, define $h(v) :=$ length of the shortest path from s' to v.

Jhonson's Algorithm.

- Add s' —— $O(n)$
- Do Bellman-Ford to get the function h. — $O(nm)$
- Reweight the edges according to $\hat{\omega}$. using h. — $O(m)$
- Run Dijkstra n times. — $O(n \cdot m \log n)$
- Let $d'(u,v)$ be the length of the shortest path from u to v.

then define $d(u,v) = d'(u,v) - h(u) + h(v)$.

$O(n^2)$.

what is the runtime?

$$O(nm\log n) = O(n^3 \log n)$$

— × — × —

Dynamic Programming based Algorithm for APSP

- $dist[u, v, l]$:= length of a shortest path from u to v using at most l edges.

$$dist[u, v, l] := \begin{cases} 0 & \text{if } l = 0 \text{ and } u = v \\ \infty & \text{if } l = 0 \text{ and } v \neq u \\ \min \begin{cases} dist[u, v, l-1], \\ \min\limits_{x \to v} \left\{ dist[u, x, l-1] + w(x \to v) \right\} \end{cases} \end{cases}$$

DP based APSP :

- Initailize $dist[u, v, l]$
- ~~for~~ $l = 1$ to $n-1$
- For every vertex u

  For every vertex $v \neq u$

  implement the recursion.

$dist[u, v, n-1] :=$ length of the shortest path from u to v.

Runtime $= O(n^2 \cdot m)$

$= O(n^4)$

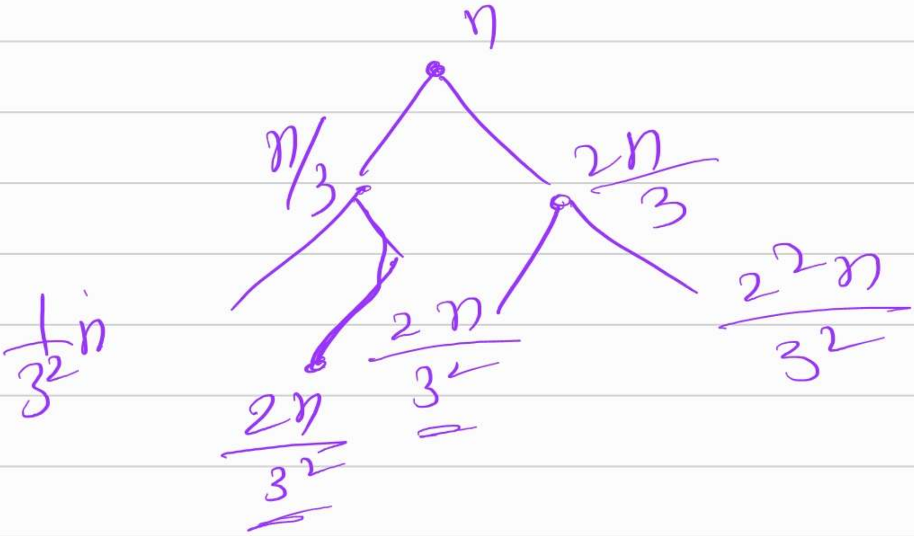
$dist[u, v, 1] = w(u \to v)$

$$dist[u, v, \ell] = \min_{x \notin \{u, v\}} \left\{ dist[u, x, \tfrac{\ell}{2}] + dist[x, v, \tfrac{\ell}{2}] \right\}$$

$dist[u, v, \ell] = dist[u, v, n-1]$

for all $\ell \geq n-1$.

$O(n^3 \cdot \log(n-1))$.



$l=1$ $\quad$ $l=1$