Algorithms - fall 2020

Shortest Paths:
B-F, + intro
to MSSP

Recap

HW due on Monday

↳ over Ch 5 + 6

Otherwise, the usual!

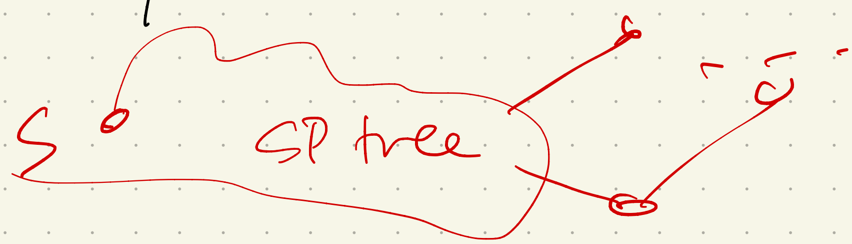
Single Source Shortest Paths (cont)

SSSPs

Key idea:
Store tentative tree distances + parents

initially: s's distance is 0
everyone else $= \infty$

Then iteratively update as you consider edges



We say an edge $\overrightarrow{uv}$ is tense
if $dist(u) + w(u \to v) < dist(v)$:

if tense - relax!
dist(v) to be dist(u) + w(u→v)

Both Dijkstra and B-F relax edges in a loop until done.

Dijkstra:

- in each round, finalize one "tentative" distance

loop repeats $O(V)$

- relax as you hit a new edge (if tense)

Runtime:

$O(E \log V)$

heap

each edge is tense ≤ 1 time

Downside:

need no negative edges

if neg. edges → exponential (edges keep getting tense)

negative cycles → fails

Bellman-Ford:

- Relax edges for a while.
- Stop when every edge has been relaxed at least once.

If any one is still tense: you've relaxed ≥ 2 times!

$\leq V-1$ $\Longrightarrow$ cycle, so halt & fail

Runtime:

$(V-1)(E)+E$

$= O(VE)$

```
BELLMANFORD(s)
  INITSSSP(s)
  repeat V − 1 times
    for every edge u→v
      if u→v is tense
        RELAX(u→v)
  for every edge u→v
    if u→v is tense
      return "Negative cycle!"
```

1st time: $d_{\leq 1}(v)$

2nd: $d_{\leq 2}(v)$

⋮

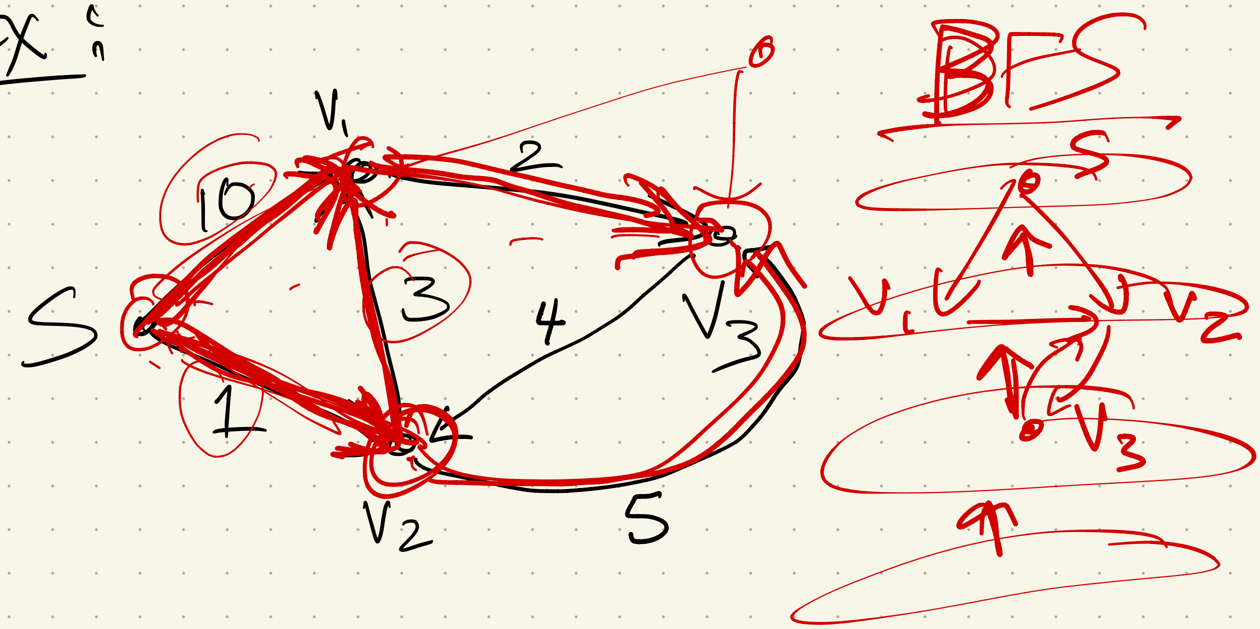
$V-1$st: $d_{\leq V-1}(v)$

How to prove correctness?

Notation:

Let $\boxed{dist_{\leq i}(v)}$:= length of the shortest s-to-v path using at most i edges.

Ex:



S:	v_1:	v_2:	v_3:
$d_{\leq 0}(s)=0$	$d_{\leq 0}(v_1)=\infty$	$d_{\leq 0}(v_2)=\infty$	$d_{\leq 0}(v_3)=\infty$
$d_{\leq 1}(s)=1$	$d_{\leq 1}(v_1)=10$	$d_{\leq 1}(v_2)=1$	$d_{\leq 1}(v_3)=\infty$
⋮	$d_{\leq 2}(v_1)=4$	$d_{\leq 2}(v_2)=1$	$d_{\leq 2}(v_3)=6$
all 0	$d_{\leq 3}(v_1)=4$	$d_{\leq 3}(v_2)=1$	$d_{\leq 3}(v_3)=6$

If no negative cycles: $d_{\leq V-1}$(vertex) is correct distance

Claim: $\forall v \, \& \, i$, after i iterations of B-F,

$$\mathrm{dist}(v) \leq \mathrm{dist}_{\leq i}(v)$$

current "guess"

Why?

Induction on i:

BC: $i = 0$

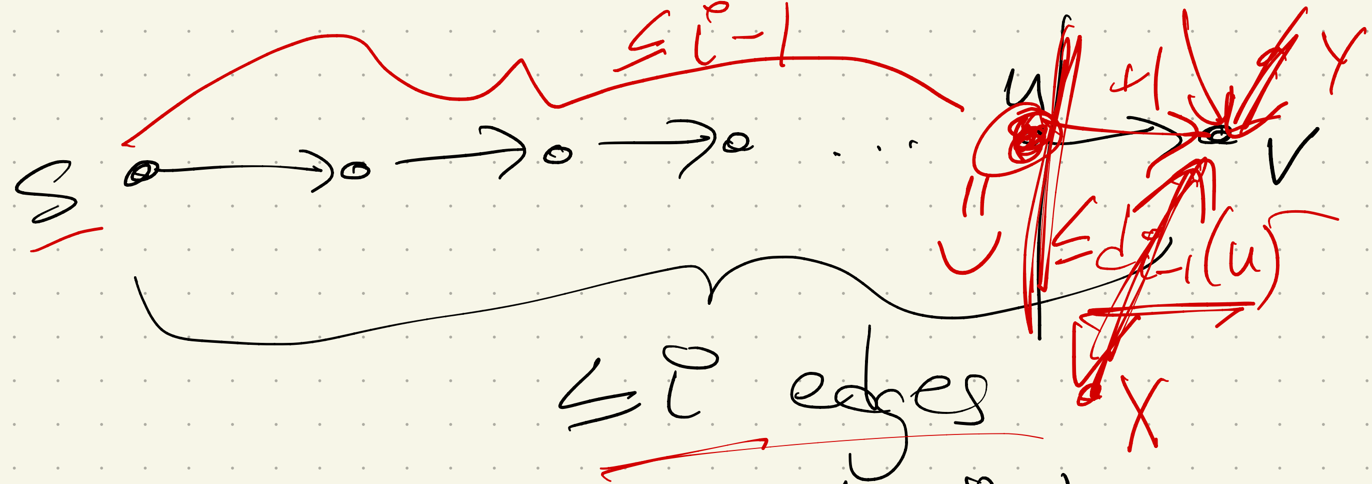
s has dist 0

all others ∞ ✓

IH: After $i-1$ iterations, all ~~tentative guesses~~ are dist(v) $\leq d_{\leq i-1}(v)$.

IS: Now consider $d_{\leq i}(v)$:

built from a path $\longrightarrow$



We know in round $i-1$,

$d_{\le i-1}(u)$ was correct for all u.

Consider $u \to v$ in next round:

It was tense:

relaxed, $\to d(v) \le d_{\le i-1}(u) + w(u \to v)$

or not:

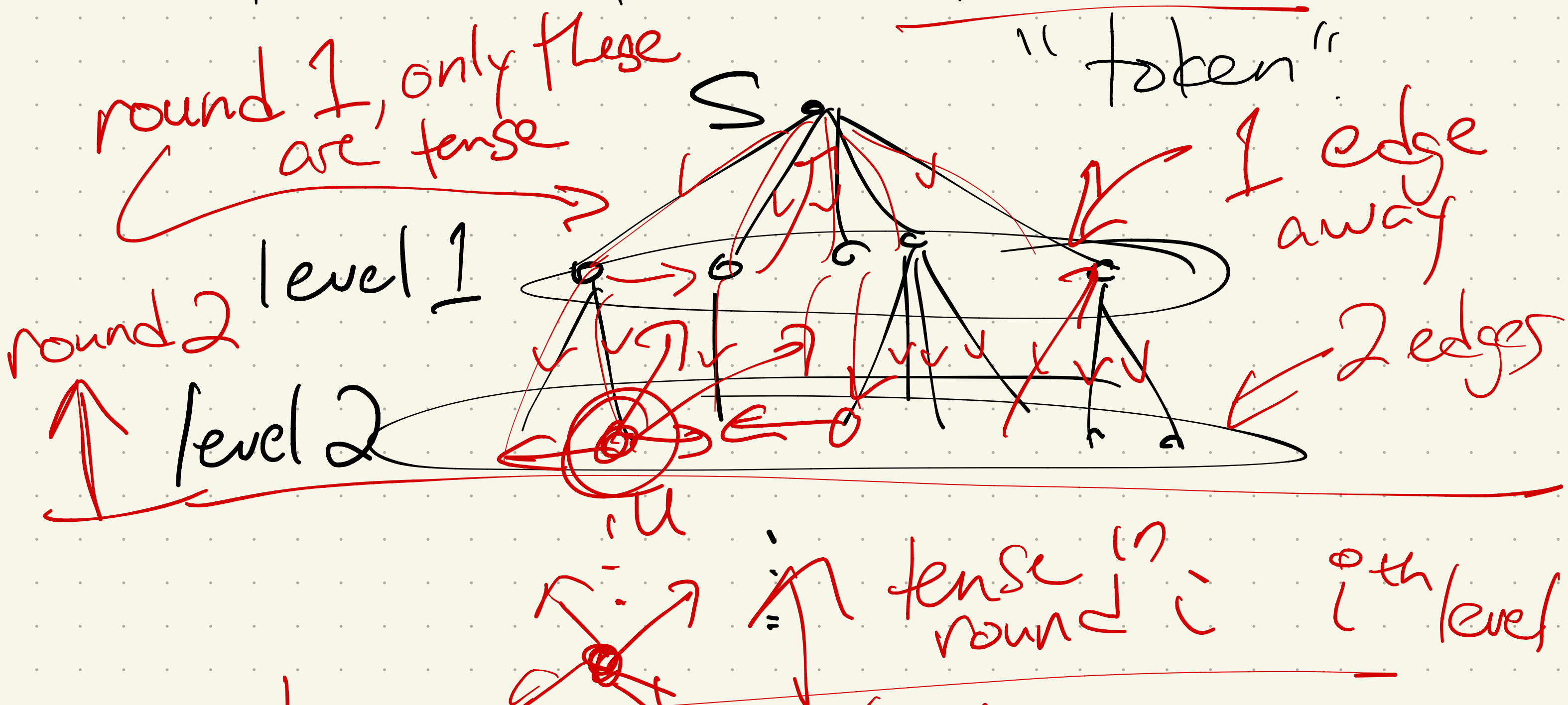
$d(v)$ is unchanged

Since all $d_{\le i-1}(u)$ were correct, one of them will give $d_{\le i}(v)$

The rest: an (in practice) speed-up

BF looks at every edge. Do we need to?

Think of a BFS tree + "token"



Combines B-F w/ BFS

```
MOORE(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 u→v is tense
                  RELAX(u→v)
                  if v is not already in the queue
                      PUSH(v)
```

only considering level i nbrs

queue: ~~s~~ ~~✠~~ ~~s's nbrs~~ ~~✠~~ dist 2 nbrs ✠

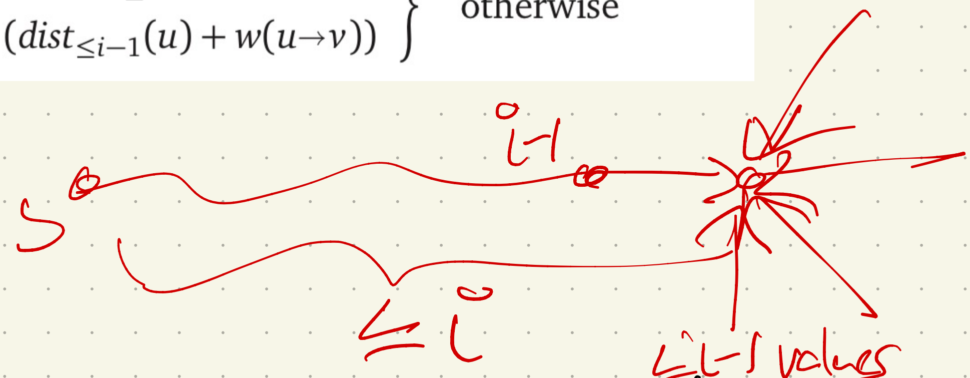
Still worst case $V \cdot E$

Final version: Bellman's!

Full circle → recursively!

$$dist_{\leq i}(v) = \begin{cases} 0 & \text{if } i = 0 \text{ and } v = s \\ \infty & \text{if } i = 0 \text{ and } v \neq s \\ \min \left\{ \begin{array}{c} dist_{\leq i-1}(v) \\ \min\limits_{u \to v} \left(dist_{\leq i-1}(u) + w(u \to v) \right) \end{array} \right\} & \text{otherwise} \end{cases}$$

Why??



Using i again as # of edges in the path!

Showing recursion → equivalent

Since all paths are ≤ V-1,

$dist_{V-1}(v)$ is $dist(v)$

(assuming no negative cycles)

runtime: same $O(VE)$

dynamic programming!

Nicer: Cleaned up DP ✓

BELLMANFORDDP(s)

$dist[0, s] \leftarrow 0$
for every vertex $v \neq s$
  $dist[0, v] \leftarrow \infty$
for $i \leftarrow 1$ to $V - 1$
  for every vertex v
    $dist[i, v] \leftarrow dist[i-1, v]$
    for every edge $u \to v$
      if $dist[i, v] > dist[i-1, u] + w(u \to v)$
        $dist[i, v] \leftarrow dist[i-1, u] + w(u \to v)$

space $O(V^2)$

relax tense edges

$\forall$ $dist_{0,v}$

$v_1 - - - - v_n$

s

V^2 space

Later observation:

Really don't need the i.

Just update those "tentative" distances, & trust it'll halt.

dist:

BELLMANFORDFINAL(s)

$dist[s] \leftarrow 0$
for every vertex $v \neq s$
  $dist[v] \leftarrow \infty$
for $i \leftarrow 1$ to $V - 1$
  for every edge $u \to v$
    if $dist[v] > dist[u] + w(u \to v)$
      $dist[v] \leftarrow dist[u] + w(u \to v)$

space $O(V)$

overwrites a single array

Same runtime as prior BF
(just a bit more confusing!)

Next reading (tomorrow)

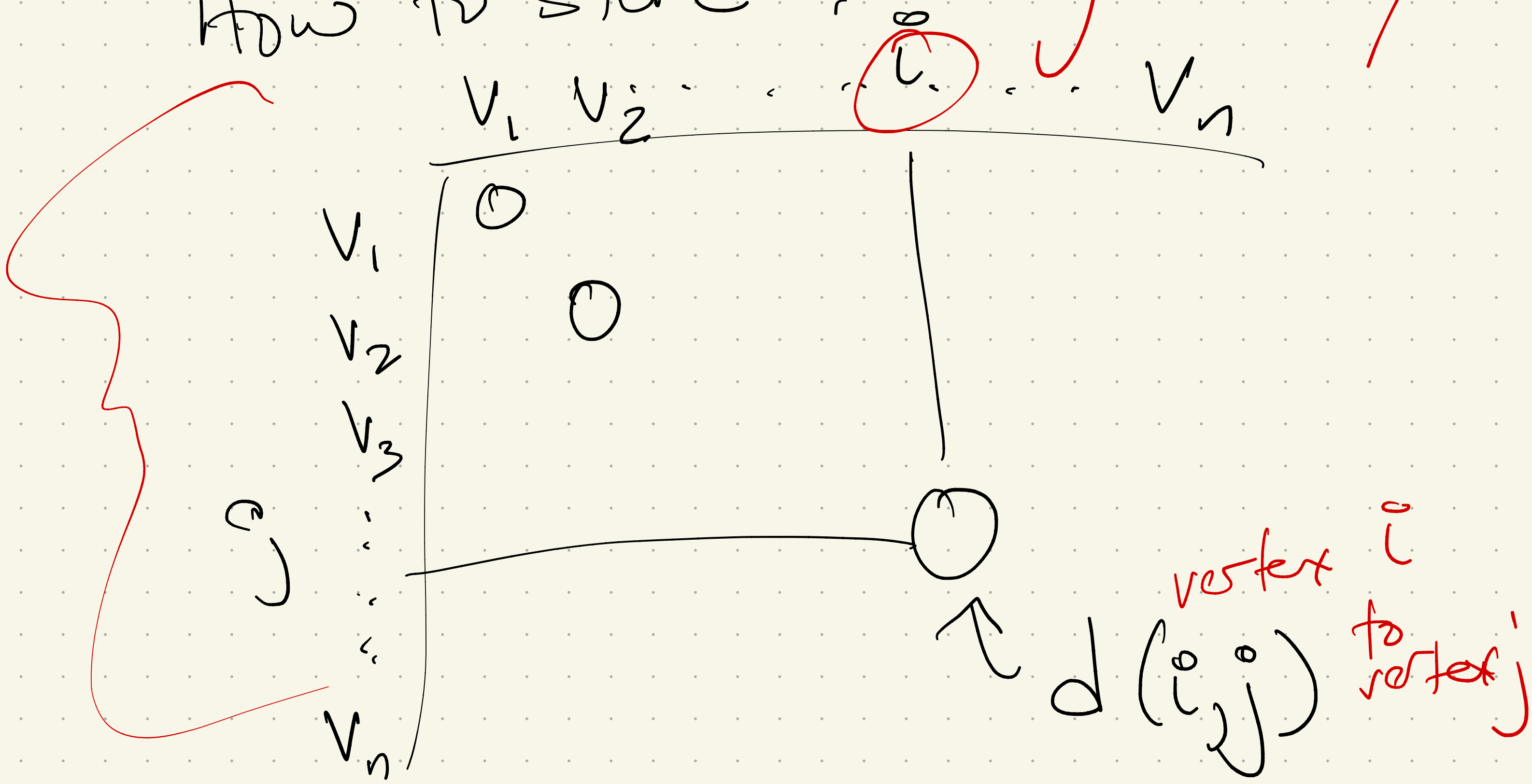
SSSPs are nice, but:

What if we are doing lots of shortest path computations?

Multiple source shortest paths (MSSP)

Goal: precompute these, & store them!

How to store? big array



Then, lookup time: $O(1)$

→ how fast can I calculate the array?

Obvious answer

Well, we just designed two or three SSSP algorithms — use them!

MSSP(G):

for each $v \in G$:

  run SSSP(v)

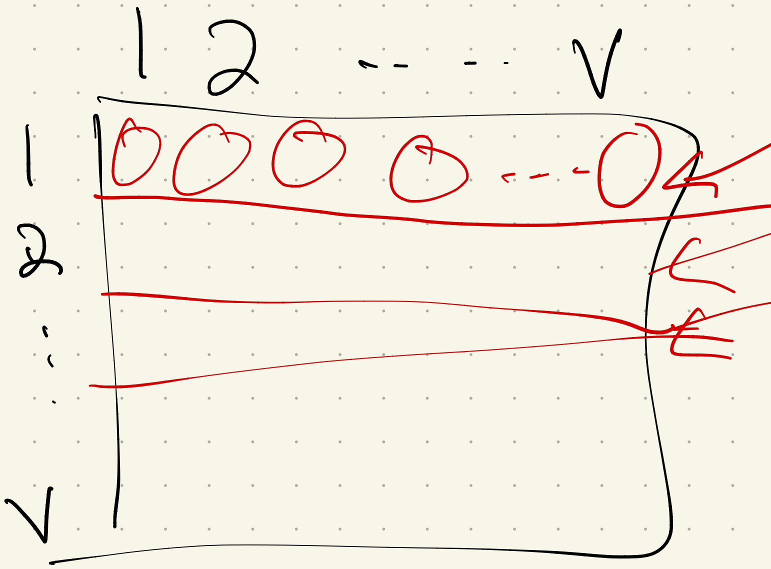
  store tree distances in dist[s, ·]

$O(V)$ times

?

$d[v, \cdot]$

"everyone"



runtime?

Runtime: $O(V \times (\text{SSSP alg}))$

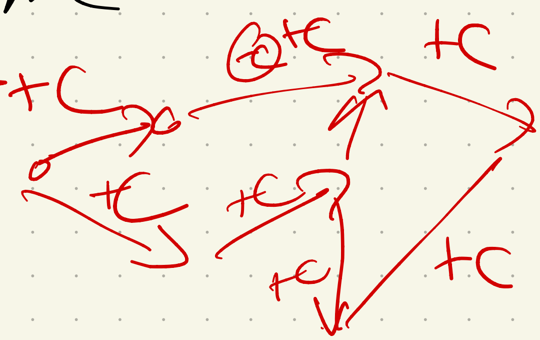
- if ~~undirected~~ weighted or a DAG: BFS, top order
  SSSP was $O(V+E)$ BFS time
  $\Rightarrow O(V(V+E)) = O(\cancel{V^2} + V \cdot E)$
  $= O(VE) = O(V^3)$
- no negative edge weights:
  Dijkstra was $O(E \log V)$
  $\Rightarrow O(VE \log V)$:
  E is $O(V^2)$
  $\Rightarrow \cancel{O(V^3 \log V)}$
- Bellman-Ford was $O(VE)$
  $\Rightarrow O(V^2 E)$
  $= O(V^4)$

Now: Can we do better?
Spoiler — YES!

Side question! (it's not...)

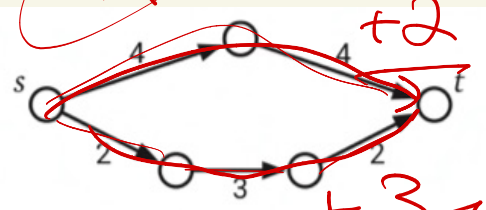
Since negative edges are bad, why can't we just re-weight?

Idea: increase all edge weights by some amount



Doesn't work!

+1 to each edge



8 → 10

7 → 10 X

Figure 9.1. Increasing all the edge weights by 2 changes the shortest path from s to t.

Why?

change in path weight is not same — based on # of edges also.

Another idea (that works):

more complex re-weighting

Suppose each v has a price attached, $\pi(v)$.

(Still have edge weights.)

change   vertex weights

$$\begin{tikzcd} \pi(u) & & \pi(v) \\ u \arrow[rr, "w(u\to v)"] & & v \end{tikzcd}$$

weight $w'(u\to v)$

Define a new function w':

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

This w' will be our new weight function!

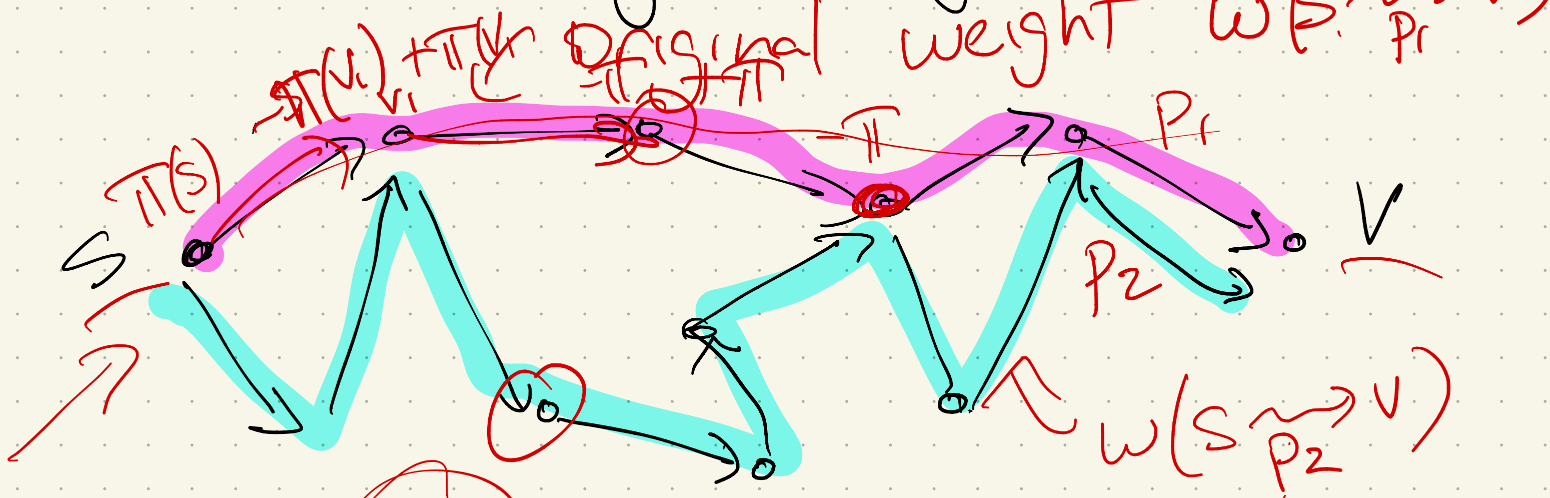
Why? preserve shortest paths, & be positive

↳ use Dijkstra! faster.

Claim: under w', shortest paths ~~are~~ the same relative order as under w.

Why?

Consider 2 $s \rightsquigarrow v$ paths:
(in original weights)



Under w':

purple path $= \pi(s) + w(\text{using purple}) - \pi(v)$

blue path $= \pi(s) + w(\text{blue}) - \pi(v)$

still same order!

Johnson's algorithm for MSSP:

use this new kind of weight function!

- Run Bellman-Ford once
  - ↳ no negative cycles or give up
- Reweight (if it works)
- Now, all paths are positive, so can use faster algorithm for other SSSPs!

More Friday!