Non-negative weights

Dijkstra

$u \to v$ is tense if $dist(u) + w(u \to v)$
$< dist(v)$

Two observations:

1) $u \to v$ can become tense only when $dist(u)$ decreases

2) Relaxing $u \to v$ sets $dist(v) \geq dist(u)$, so relaxing

$u \to v$ with lowest dist(u) shouldn't result in a chain of relaxations that eventually changes dist(u).

- keep a priority queue of tail) vertices u. Only add v to priority queue when dist(v) drops.

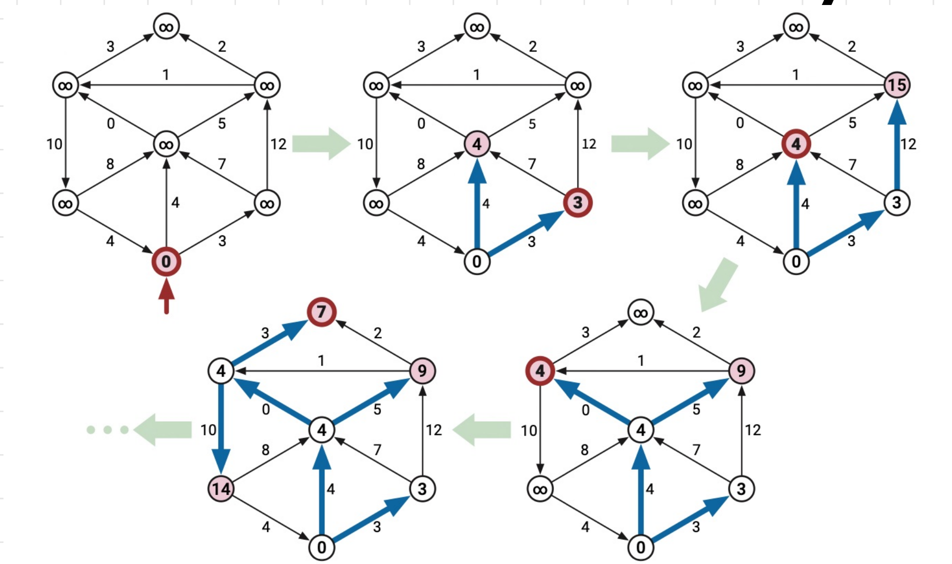
```
DIJKSTRA(s):
    INITSSSP(s)
    INSERT(s, 0)
    while the priority queue is not empty
        u ← EXTRACTMIN()
        for all edges u→v
            if u→v is tense
                RELAX(u→v)
                if v is in the priority queue
                    DECREASEKEY(v, dist(v))
                else
                    INSERT(v, dist(v))
```

insert s into queue with priority 0.

Is Ford SSSP, so it will find shortest paths, even with negative weights! (no proof today)

- may be slow with negative weights

With no negative weights...



Let u_i be the ith vertex returned by ExtractMin.

Let d_i be $dist(u_i)$ at moment ExtractMin returns u_i.

(u_i <u>may</u> $= u_j$ when $i \neq j$)

Lemma: For all $i < j$, we have $d_i \le d_j$.

Proof: Fix some i. Will show $d_{i+1} \ge d_i$.

Suppose we relax $u_i \to u_{i+1}$ during ith iteration.

Then
$$\begin{aligned} d_{i+1} &= \operatorname{dist}(u_{i+1}) \\ &= \operatorname{dist}(u_i) + w(u_i \to u_{i+1}) \\ &= d_i + \underbrace{w(u_i \to u_{i+1})}_{\text{at least } 0} \\ &\ge d_i \end{aligned}$$

Otherwise, u_{i+1} was already in queue.

ExtractMin chose u_i, so

$$d_i \le d_{i+1}.$$

Lemma: Every vertex v is extracted at most once.

Proof: Otherwise $v = u_i = u_j$ for some $i < j$.

To put v back in queue after iteration i, we must decrease dist(v).

So $d_j < d_i$. $\perp$ contradiction!

Lemma: When Dijkstra ends, for all vertices v, $dist(v)$ is the distance from s to v.

Proof: By induction on min # edges on a shortest path to v.

Let L_w denote distance from s to w.

Let $P = s \rightarrow \dots \rightarrow u \rightarrow v$ be shortest path to v with min # edges.

If P has no edges, then $v = s$, $dist(s) = 0$ ✓

Otherwise, by induction we set $dist(u)$ to L_u, add u to queue, & later Extract it.

Maybe $dist(v) \leq dist(u) + w(u \to v)$ already.

If not.. we will Relax $u \to v$

Either way,

$$dist(v) \leq dist(u) + w(u \to v)$$
$$= L_u + w(u \to v)$$
$$= L_v$$

But it can't go lower than L_v, so
$dist(v) = L_v$. ✓

Analysis: Each priority queue operation take $O(\log V)$ time.

Each vertex Extracted + Inserted $\le$ once.

Each edge relaxed $\le$ once.

$$O((V+E)\log V) = O(E\log V)$$

Time

assuming graph is connected

May be fast even with a few negative edges.

CLRS version always fast but neg edges may break it!

If all weights are 1.

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

$O(V+E)$ Time.

All-pairs shortest paths

Compute dist(u,v), the distance from u to v for all vertices u & v.

We'll assume no negative cycles today.

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

If using Bellman-Ford, takes

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

Dynamic Programming

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

Cannot be used if there are directed cycles!

Makes an infinite loop!

Need a parameter that actually decreases...

Limit which vertices can appear in path

Arbitrarily number vertices from 1 to $|V|$.

$\pi(u,v,r)$:= shortest path from u to v where each intermediate (internal, not u or v) is numbered at most r.

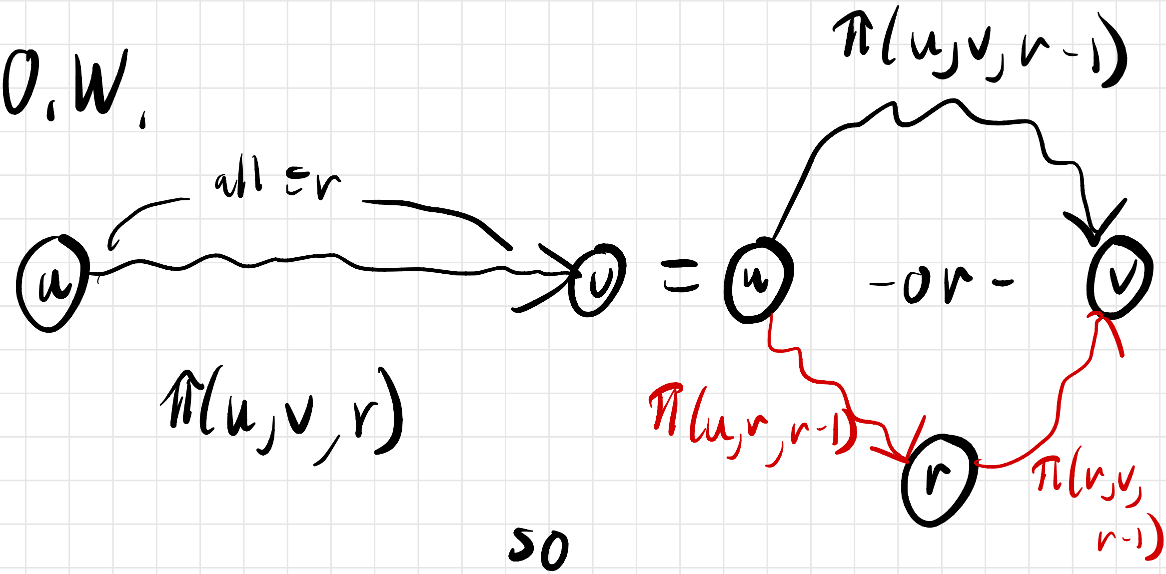
$dist(u,v,r)$:= length of $\pi(u,v,r)$

$\pi(u,v,|V|)$ is the true u–v shortest path

If $r = 0$.

$\pi(u, v, r)$ is $u \to v$

O.W.



so

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

$\Theta(V^3)$ subproblems in constant time each $\Rightarrow O(V^3)$ time

```
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]
```