Non-negative weights

Dijkstra

\( a \to v \) is tense if \( \text{dist}(u) + w(u \to v) < \text{dist}(v) \)

Two observations:

1) \( u \to v \) can become tense only when \( \text{dist}(u) \) decreases

2) Relaxing \( u \to v \) sets \( \text{dist}(v) \geq \text{dist}(u) \), so relaxing
choose vertex $u$ with lowest $\text{dist}(u)$ should not result in a chain of relaxations that eventually changes $\text{dist}(u)$.

- keep a priority queue of tail vertices $u$. Only add $v$ to priority queue when $\text{dist}(v)$ drops.
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 $i$th vertex returned by $\text{Extract Min}$. Let $d_{i\hat{\omega}}$ be $\text{dist}(u_i)$ at moment $\hat{\omega}$. $\text{Extract Min}$ returns $u_i$. ($u_i$ may $= u_j$ when $i \neq j$)
Lemma: For all \( \hat{\omega} \neq j \), we have 
\[ d_{\hat{\omega}} \leq d_j. \]

Proof: Fix some \( \hat{\omega} \). Will show \( d_{\hat{\omega}+1} \geq d_{\hat{\omega}} \).

Suppose we relax \( u_{\hat{\omega}} \rightarrow u_{\hat{\omega}+1} \) during with iteration.

Then 
\[
d_{\hat{\omega}+1} = \text{dist}(u_{\hat{\omega}+1}) \\
\quad = \text{dist}(u_{\hat{\omega}}) + w(u_{\hat{\omega}} \rightarrow u_{\hat{\omega}+1}) \\
\quad = d_{\hat{\omega}} + w(u_{\hat{\omega}} \rightarrow u_{\hat{\omega}+1}) \\
\geq d_{\hat{\omega}} \] at least 0
Otherwise, \( u_{\hat{w}+1} \) was already in queue.

\[ \text{Extract Min chose } u_{\hat{w}}, \text{ so } d_{\hat{w}} = d_{\hat{w}+1}. \]

Lemma: Every vertex \( v \) is extracted at most once.

Proof: Otherwise \( v = u_{\hat{w}} = u_{\hat{w}+1} \) for some \( \hat{w} < j \).

To put \( v \) back in queue after iteration \( \hat{w} \), we must decrease \( \text{dist}(v) \). So \( d_{\hat{w}} < d_{\hat{w}+1} \). \( \square \)
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 $P = s \rightarrow \ldots \rightarrow u \rightarrow v$ be shortest path to $v$ with min # edges.
If $P$ has no edges, then $v = s$, $\text{dist}(s) = 0$.

Otherwise, by induction we set $\text{dist}(u)$ to $L_u$, add $u$ to queue, and later Extract it.

Maybe $\text{dist}(v) \leq \text{dist}(u) + w(u \rightarrow v)$ already.

If not, we will Relax $u \rightarrow v$

Either way, $\text{dist}(v) = \text{dist}(u) + w(u \rightarrow v)$

$= L_u + w(u \rightarrow v)$

$= L_v$
But it can't go lower than $L_v$, so $\text{dist}(v) = L_v$.

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

Each vertex Extracted or Inserted = once.
Each edge relaxed = once.

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

assuming time

graph is connected
May go 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
    - $u \leftarrow \text{Pull}()$
    - for all edges $u \rightarrow v$
      - if $\text{dist}(v) > \text{dist}(u) + 1$
        - $\langle\text{if } u \rightarrow v \text{ is tense}\rangle$
        - $\text{dist}(v) \leftarrow \text{dist}(u) + 1$
        - $\langle\text{relax } u \rightarrow v\rangle$
      - $\text{pred}(v) \leftarrow u$
    - **Push($v$)**

$O(V + E)$ Time.
All-pairs shortest paths
Compute $\text{dist}(u, v)$, the distance from $u$ to $v$ for all vertices $u \neq v$.

We'll assume no negative cycles today.

**ObviousAPSP(V, E, w):**
for every vertex $s$
\[
\text{dist}[s, \cdot] \leftarrow \text{SSSP}(V, E, w, s)
\]

If using Bellman-Ford, takes
\[
V \cdot O(VE) = O(V^2 E) = O(V^4)
\]
Dynamic Programming

\[ \text{dist}(u, v) = \begin{cases} 
0 & \text{if } u = v \\
\min_{x \rightarrow v} (\text{dist}(u, x) + w(x \rightarrow v)) & \text{otherwise}
\end{cases} \]

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 IV.

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

\( \text{dist}(u, v, r) \): length of \( \pi (u, v, r) \)

\( \pi (u, v, IV) \) is the true \( u-v \) shortest path.
If \( r = 0 \),
\[
\pi(u_j, v, r) \text{ is } u \rightarrow v
\]

O.w.,

\[
\pi(u_j, v, r) = \pi(u_j, v, r-1)
\]

so

\[
dist(u_j, v, r) = \begin{cases} 
  w(u \rightarrow v) & \text{if } r = 0 \\
  \min \left\{ \begin{array}{c}
  \{ \text{dist}(u_j, v, r-1) \\
  \text{dist}(u_j, v, r-1) + \\
  \text{dist}(r, v, r-1) \} \end{array} \right\} & \text{o.w.}
\end{cases}
\]

\( \Theta(V^3) \) subproblems in constant time each \( \Rightarrow O(V^3) \) time
**KleineAPSP**($V, E, w)$:

for all vertices $u$

    for all vertices $v$

        $\text{dist}[u, v, 0] \leftarrow w(u \rightarrow v)$

for $r \leftarrow 1$ to $V$

    for all vertices $u$

        for all vertices $v$

            if $\text{dist}[u, v, r - 1] < \text{dist}[u, r, r - 1] + \text{dist}[r, v, r - 1]$

                $\text{dist}[u, v, r] \leftarrow \text{dist}[u, v, r - 1]$

            else

                $\text{dist}[u, v, r] \leftarrow \text{dist}[u, r, r - 1] + \text{dist}[r, v, r - 1]$

---

**FloydWarshall**($V, E, w)$:

for all vertices $u$

    for all vertices $v$

        $\text{dist}[u, v] \leftarrow w(u \rightarrow v)$

for all vertices $r$

    for all vertices $u$

        for all vertices $v$

            if $\text{dist}[u, v] > \text{dist}[u, r] + \text{dist}[r, v]$

                $\text{dist}[u, v] \leftarrow \text{dist}[u, r] + \text{dist}[r, v]$