Review:
- Shortest paths in weighted graph from single source
  - Triangle inequality property leads to concept of "relaxing an edge"
    \[ d(u,v) \leq d(u,w) + w(w,v) \]
  - Leads to generic shortest paths algorithm:
    - Initialize data structures \( P, d \)
    - Repeatedly select and relax edges until all satisfy \( d(v) = \delta(u,v) + w(u,v) \)
- Two problems
  - How to select edges efficiently?
  - What to do about negative-weight cycles?
- For case of DAG or no-negative edges, can use topological sort or Dijkstra's alg., respectively
- Dijkstra's algorithm
  - greedy edge selection based on shortest estimated distance to \( s \)
  - only relax each edge once
  - \( O(E \log V) \) for binary heap, all vertices reachable

**Today:**

- Bellman-Ford algorithm - neg.-wt. cycles allowed
  - systematic, repeated loop through edges
  - relax each edge \( |V|-1 \) times
  - \( O(E \cdot V) \)
- Difference constraint problem (recitation)
  - maps to a shortest paths problem

---

**Bellman-Ford Algorithm**

for all \( v \in V \)

\[
d[v] \leftarrow \infty \\
\pi[v] \leftarrow \text{null}
\]

\[
d[s] \leftarrow 0
\]

for \( i = 1 \) to \( |V|-1 \)

    for each edge \( (u,v) \in E \)

        loop through all edges

        if \( d[v] > d[u] + w(u,v) \)

            \[ d[v] \leftarrow d[u] + w(u,v) \]

        \( |V|-1 \) times,

        \( \pi[v] \leftarrow u \)

    relax each \( u \) return

for each edge \( (u,v) \in E \)

if \( d[v] > d[u] + w(u,v) \)

report "negative-weight cycle exists"

Check for negative-weight cycles
Running time of Bellman-Ford Alg.

$O(V)$ initialization
$O(V \cdot E)$ main loop of relaxations
$O(E)$ negative wt. cycle detection

$O(V \cdot E)$

$\Rightarrow O(V^3)$ for dense graphs $|E| = O(V^2)$

$\Rightarrow O(V^2)$ for sparse graphs $|E| = \Omega(V)$

Correctness of Bellman-Ford Alg.

Lemma: $\delta(s, v) \leq d[v]$ for all $v$ at all times

Proof: True initially. When $d[v]$ is updated by relaxation, it corresponds to some path and its weight, which is not less than weight of shortest path.

Lemma: Let $s \rightarrow v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_k$ be a shortest path from $s$ to $v_k$. Then, after $k$ iterations of main loop $d[v_k] = \delta(s, v_k)$

Proof: Induction on $k$. Base case ($k=1$) true after relaxation of $(s, v_1)$. Could also use $k=0$ base case. If true after $k-1$ iterations that $\delta(s, v_{k-1}) = d[v_{k-1}]$, then after $k$ iterations $(v_{k-1}, v_k)$ will be relaxed and $d[v_k] = \delta(s, v_k)$. 
Lemma: If \( s \rightarrow v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_k \) is a shortest path, then \( k \leq |V|-1 \).

Proof: A shortest path has no cycles, so it can use at most \(|V|-1\) edges to connect \(|V|\) vertices. The index \( k \) corresponds to the number of edges, which is at most \(|V|-1\).

Theorem: If no path from \( s \) to \( v \) contains a negative-weight cycle, then (there exists a shortest path \( s \rightarrow v \) or \( d[v] = \infty \) and) \( d[v] = \delta(s,v) \) at the end of Bellman-Ford algorithm.

Proof: Follows from above lemmas. Each shortest path has at most \(|V|-1\) edges, so after \(|V|-1\) iterations of Bellman-Ford main loop, each vertex \( v \) of each shortest path will obey \( d[v] = \delta(s,v) \). Non-reachable vertices will have \( d[v] = \infty \) from initialization.

Note: Question? How does Bellman-Ford avoid the potentially exponential problem identified in L13.5 (page 5 of lecture 13)?
Treatment of negative-weight cycles in Bellman-Ford

Theorem: If \( \exists \) a neg.-wt. cycle reachable from \( s \), then at the end of the main loop:
\[
(\exists u)(\exists v) \text{ s.t. } d[v] > d[u] + w(u,v).
\]

Proof: This condition just states that further relaxations are possible. We saw in L13.6 that a negative-weight cycle reachable from \( s \) leads to an infinite set of relaxations, so after a finite number of relaxations from the main loop, more will be possible.

Thus, Bellman-Ford correctly reports existence of negative-weight cycles reachable from \( s \).

To actually identify all vertices with \( \delta(s,v) = -\infty \):
- run Bellman-Ford again (without re-initializing)
- any vertex \( v \) whose value \( d[v] \) decreases is one whose \( \delta(s,v) = -\infty \).
Recitation 15 Preview: Difference Constraints Problem

Difference constraints problem

Given:
\[ x_1 - x_2 \leq 0 \]
\[ x_3 - x_1 \leq 5 \]
\[ x_2 - x_3 \leq -2 \]

Find: Valid values of \( x_1, x_2, \text{ and } x_3 \)

Map to constraint graph

- \( v_1, v_2, \text{ and } v_3 \) represent \( x_1, x_2, \text{ and } x_3 \), respectively
- Black edges and their weights reflect difference constraints

By Triangle Inequality:
\[ \delta(v_0, v_j) \leq \delta(v_0, v_i) + \omega(v_i, v_j) \]
Assign:
\[ x_j \leq x_i + \omega(v_i, v_j) \]
Then:
\[ x_j - x_i \leq \omega(v_i, v_j) \]
This is the form of the original difference constraints

A solution here is:
\[ (x_1, x_2, x_3) = (-2, -2, 0) \]

Note that for any constant \( d \), can add \( d \) to each \( x_i \) and obtain another valid solution.

\((d-2, d-2, d) \rightarrow (-1, -1, 1), (0, 0, 2), (1, 1, 3), \ldots\)

There are other classes of solutions as well.