Problem solving by reduction

• You are given a directed graph $G = (V, E)$ with non-negative weights $w : E \rightarrow \mathbb{R}_{\geq 0}$, along with nodes $s, t \in V$, and a positive integer $k$

• Edges are colored red and green

• A path is called admissible if it contains at most $k$ red edges

• Determine if there is an admissible path from $s$ to $t$, and if so, find one that is shortest (relative to $w$)
All pairs shortest paths

One approach:

- Run a single-source shortest path algorithm from each vertex
  - Bellman-Ford (no negative cycles): $O(|V|^2(|V| + |E|))$, or $O(|V|^4)$
  - Dijkstra (no negative edges): $O(|V|(|V| \log |V| + |E|))$, or $O(|V|^3)$

Floyd-Warshall Algorithm:

- no negative cycles
- running time $O(|V|^3)$
• Number the vertices 1..n
• For a path \( \rho = (v_0, v_1, \ldots, v_{k-1}, v_k) \), we say that \( v_1, \ldots, v_{k-1} \) are intermediate vertices
• For \( k = 0 \ldots n \), \( \delta^{(k)}(i, j) := \) length of the shortest path from \( i \) to \( j \) whose intermediate vertices belong to \( \{1, \ldots, k\} \)

\[
\delta^{(0)}(i, j) = \begin{cases} 
0 & \text{if } i = j; \\
w(i, j) & \text{if } i \neq j \text{ and } (i, j) \in E \\
\infty & \text{otherwise}
\end{cases}
\]

• For \( k > 0 \)

\[
\delta^{(k)}(i, j) = \min \left( \delta^{(k-1)}(i, j), \delta^{(k-1)}(i, k) + \delta^{(k-1)}(k, j) \right)
\]
Straightforward implementation:

• Use a 3D array $D[i, j, k]$

$$D[i, j, 0] \leftarrow \delta^{(0)}(i, j) \text{ for } i, j = 1 \ldots n$$

for $k \leftarrow 1$ to $n$ do

  for $i \leftarrow 1$ to $n$ do

    for $j \leftarrow 1$ to $n$ do

      $d' \leftarrow D[i, k, k - 1] + D[k, j, k - 1]$  

      if $d' < D[i, j, k - 1]$  

        then  $D[i, j, k] \leftarrow d'$  

        else  $D[i, j, k] \leftarrow D[i, j, k - 1]$  

• Running time: $O(n^3)$

• Space: $O(n^3)$
Improving the space requirement:

- Since $D[\cdot, \cdot, k]$ depends only on $D[\cdot, \cdot, k-1]$, we can obviously get by with just two 2D arrays.
- In fact, we can get by with just a single array, with updates “in place”.

Justification:

\[
\delta^{(k)}(i, i) = 0 \text{ for all } i, k \quad \text{(no negative cycles)}
\]

\[
\delta^{(k)}(i, k) = \min(\delta^{(k-1)}(i, k), \delta^{(k-1)}(i, k) + \delta^{(k-1)}(k, k)) = \delta^{(k-1)}(i, k)
\]

\[
\delta^{(k)}(k, j) = \delta^{(k-1)}(k, j)
\]
Improved implementation:

- Use a 2D array $D[i, j]$

$$D[i, j] \leftarrow \delta^{(0)}(i, j) \text{ for } i, j = 1 \ldots n$$

for $k \leftarrow 1$ to $n$ do

  for $i \leftarrow 1$ to $n$ do

    for $j \leftarrow 1$ to $n$ do

      $$d' \leftarrow D[i, k] + D[k, j]$$

      if $d' < D[i, j]$

        then $D[i, j] \leftarrow d'$
Adding path recovery:

- Two arrays: $D[i, j], N[i, j]$
  
  $D[i, j] \leftarrow \delta^{(0)}(i, j)$ for $i, j = 1 \ldots n$

  $N[i, j] \leftarrow j$ for $i, j = 1 \ldots n$

  for $k \leftarrow 1$ to $n$ do
    for $i \leftarrow 1$ to $n$ do
      for $j \leftarrow 1$ to $n$ do
        $d' \leftarrow D[i, k] + D[k, j]$
        if $d' < D[i, j]$
          then $D[i, j] \leftarrow d'$

          $N[i, j] \leftarrow N[i, k]$

Printing a shortest path from $u$ to $v$:

$x \leftarrow u$, print $x$
while $x \neq v$ do: $x \leftarrow N[x, v]$, print $x$
Johnson’s Algorithm

Motivation:

- For sparse graphs with no negative edges, running Dijkstra from each node is faster than Floyd-Warshall
- But what if we have a sparse graph with negative edges, and no negative cycles?

Idea:

- Convert $G$ to a graph $G'$ such that
  - $G'$ has no negative edges
  - Shortest paths in $G'$ are shortest paths in $G$
- How? Re-weighting
• Let \( G = (V, E) \) be a directed graph with weights \( w : E \to \mathbb{R} \), and no negative cycles.
• Let \( h : V \to \mathbb{R} \), and define

\[
\hat{w}(u, v) = w(u, v) + h(u) - h(v)
\]

• For a path \( p \) in \( G \), define \( \hat{w}(p) \) to be the weight of the path, using \( \hat{w} \).
• Observation: if \( p \) is a path from \( u \) to \( v \), then

\[
\hat{w}(p) = w(p) + h(u) - h(v) \quad \text{(telescoping sum)}
\]
• No negative cycles relative to \( \hat{w} \).
• Define \( \hat{\delta}(u, v) \) to be the length of the shortest path from \( u \) to \( v \), relative to \( \hat{w} \).
• Therefore, any shortest path relative to \( \hat{w} \) is also a shortest path relative to \( w \), and

\[
\hat{\delta}(u, v) = \delta(u, v) + h(u) - h(v)
\]
Johnson’s Algorithm:

- Construct a new weighted graph $G'$ by adding a node $s$ to $G$ and 0-weight edges connecting $s$ to every other vertex in $G$
- Run Bellman-Ford starting from $s$, to obtain $\delta(s, v)$ for all nodes $v$
- Define $h(v) := \delta(s, v)$, and re-weight as above
- By Triangle Inequality, for every edge $(u, v)$:
  
  \[ h(v) = \delta(s, v) \leq \delta(s, u) + w(u, v) = h(u) + w(u, v) \]
  and so
  
  \[ \hat{w}(u, v) = w(u, v) + h(u) - h(v) \geq 0 \]
- Run Dijkstra from every node, using $\hat{w}$
Running time:

- Bellman-Ford: $O(|V|(|V| + |E|))$
- Dijkstra: $O(|V|(|V| \log |V| + |E|))$
- Dijkstra dominates