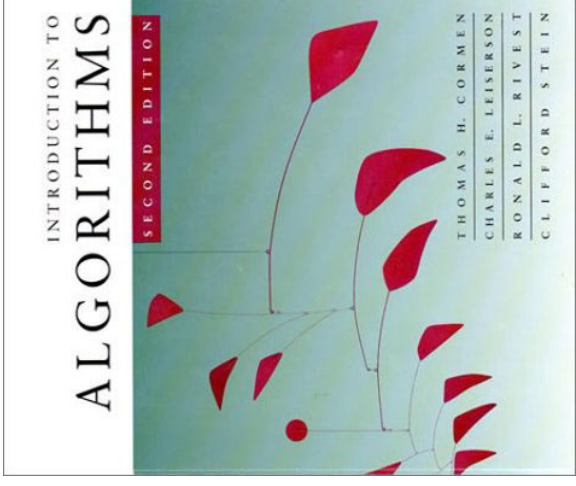


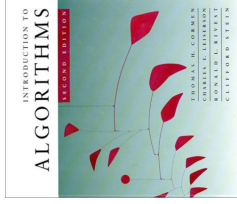
CS 445



Shortest Paths in Graphs III

Alon Efrat

Slides courtesy of Erik Demaine with
small changes by Carola Wenk



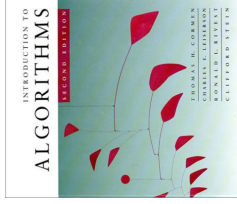
Shortest paths

Single-source shortest paths

- Nonnegative edge weights
 - Dijkstra's algorithm: $O(E \lg V)$
- General
 - Bellman-Ford: $O(VE)$
- DAG
 - One pass of Bellman-Ford: $O(V + E)$

All-pairs shortest paths

- Nonnegative edge weights
 - Dijkstra's algorithm $|V|$ times: $O(VE \lg V)$
- General
 - Bellman-Ford form each vertex $O(V^2E \lg V)$



All-pairs shortest paths

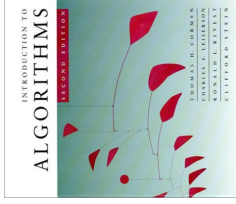
Input: Digraph $G = (V, E)$, where $|V| = n$, with edge-weight function $w : E \rightarrow \mathbb{R}$.

Output: $n \times n$ matrix of shortest-path lengths $\delta(i, j)$ for all $i, j \in V$.

Homework – what happened if we actually need the shortest path itself.
IDEA #1:

- Run Bellman-Ford once from each vertex.
- Time = $O(V^2E)$.
- Dense graph $\Rightarrow O(V^4)$ time.

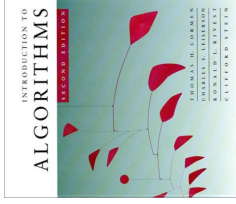
Good first try!



Johnson's algorithm

1. Compute a weight function \hat{w} from w such that $\hat{w}(u, v) \geq 0$ for all $(u, v) \in E$.
 - Run Dijkstra's algorithm from each vertex using \hat{w} .
 - Time = $O(VE \lg V)$.
 - Reweight each shortest-path length $\hat{w}(p)$ to produce the shortest-path lengths $w(p)$ of the original graph.
 - Time = $O(V^2)$.

How do we compute \hat{w} ???



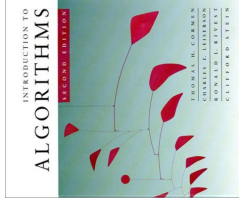
Graph reweighting

Given a label $h(v)$ for each $v \in V$, **reweight** each edge $(u, v) \in E$ by $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$.

Theorem. all paths between the same two vertices are reweighted by the same amount.

Proof. Let $p = v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_k$ be a path in the graph.

$$\begin{aligned} \text{Then, we have } \hat{w}(p) &= \sum_{i=1}^{k-1} \hat{w}(v_i, v_{i+1}) \\ &= \sum_{i=1}^{k-1} (w(v_i, v_{i+1}) + h(v_i) - h(v_{i+1})) \\ &= \sum_{i=1}^{k-1} w(v_i, v_{i+1}) + h(v_1) - h(v_k) \\ &= w(p) + h(v_1) - h(v_k). \quad \square \end{aligned}$$



Reweighting - cont

Theorem If p' is a shortest path $v_1 \rightarrow v_k$ after the reweighting then it is also the shortest path before the reweighting.

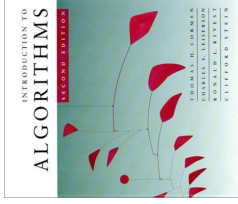
Proof. Assume $p = v_1 \rightarrow \dots \rightarrow v_k$ is the shortest path in the original graph, and $p' = v_1 \rightarrow \dots \rightarrow v_k$ be a shortest path in the reweighted graph. Hence $\hat{w}(p') \leq \hat{w}(p)$.

But then

$$\hat{w}(p') = w(p') + h(v_1) - h(v_k) \leq \hat{w}(p) = w(p) + h(v_1) - h(v_k) \quad \text{or} \\ w(p') \leq w(p).$$

One the other hand, p is a shortest path, so $w(p) \leq w(p')$.

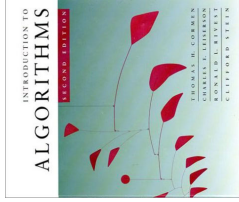
Yielding $w(p) = w(p')$. □



Find vertex labeling h

1. Let s be an additional vertex (not in V).
 - New edges (s, v) for all $v \in V$, all with weigh 0.
 - \Rightarrow New graph $G' = (V \cup \{s\}, E')$ where
$$E' = E \cup \{(s, v) : v \in V\}$$
2. G' has no negative-weight cycles if and only if G has no negative-weight cycles
3. Define $h(v) = \delta(s, v)$
 - $\Rightarrow h(v) \leq h(u) + w(u, v)$ by triangle inequality
 - Set $\hat{w}(u, v) = w(u, v) + h(u) - h(v) \geq 0$

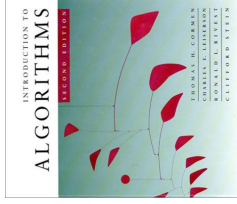
Compute h and \hat{w} using Bellman-Ford on G'



Johnson's algorithm

1. Find a vertex labeling h such that $\hat{w}(u, v) \geq 0$ for all $(u, v) \in E$ by using Bellman-Ford, or determine that a negative-weight cycle exists.
 - Time = $O(VE)$.
2. Run Dijkstra's algorithm from each vertex using \hat{w} .
 - Time = $O(VE \lg V)$.
3. Reweight each shortest-path length $\hat{w}(p)$ to produce the shortest-path lengths $w(p)$ of the original graph.
 - Time = $O(V^2)$.

Total time = $O(VE \lg V)$.



Shortest paths

Single-source shortest paths

- Nonnegative edge weights
 - Dijkstra's algorithm: $O(E + V \lg V)$
 - General: Bellman-Ford: $O(VE)$
 - DAG: One pass of Bellman-Ford: $O(V + E)$
- } adj. list

All-pairs shortest paths

- Nonnegative edge weights
 - Dijkstra's algorithm $|V|$ times: $O(VE \lg V)$
 - General
 - Bellman-Ford $|V|$ times: $O(V^2E)$
 - Matrix multiplication: $O(V^3 \log V)$
 - Floyd-Warshall: $O(V^3)$
 - Johnson's algorithm: $O(VE \lg V)$
- } adj. list
- } adj. matrix
- } adj. list