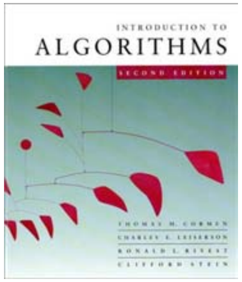
CS 5633 -- Spring 2005



More on Shortest Paths

Carola Wenk

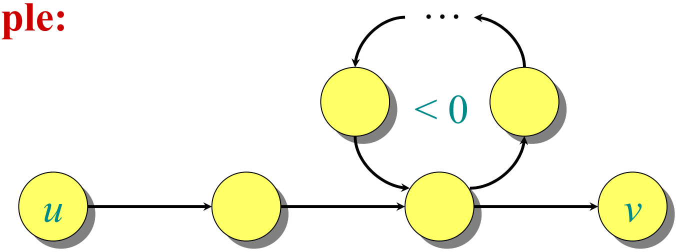
Slides courtesy of Charles Leiserson with small changes by Carola Wenk

---

Negative-weight cycles

Recall: If a graph $G = (V, E)$ contains a negative-weight cycle, then some shortest paths may not exist.

Example:



Bellman-Ford algorithm: Finds all shortest-path weights from a ***source*** $s \in V$ to all $v \in V$ or determines that a negative-weight cycle exists.

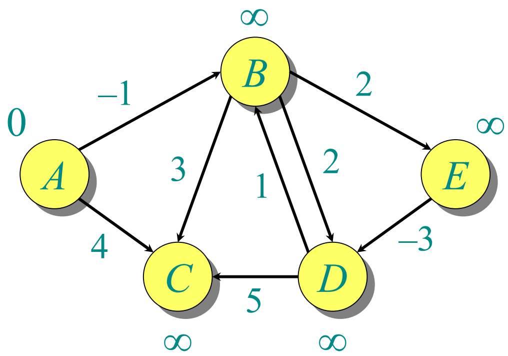
Bellman-Ford algorithm

```
$d[s] \leftarrow 0$                              }
for each $v \in V - \{s\}$                       }  initialization
    do $d[v] \leftarrow \infty$                  }
for $i \leftarrow 1$ to $|V| - 1$ do
    for each edge $(u, v) \in E$ do
        if $d[v] > d[u] + w(u, v)$ then          }
            $d[v] \leftarrow d[u] + w(u, v)$     }  relaxation step
            $\pi[v] \leftarrow u$                }
for each edge $(u, v) \in E$
    do if $d[v] > d[u] + w(u, v)$
        then report that a negative-weight cycle exists
```

At the end, $d[v] = \delta(s, v)$. Time $= O(|V||E|)$.

Example of Bellman-Ford

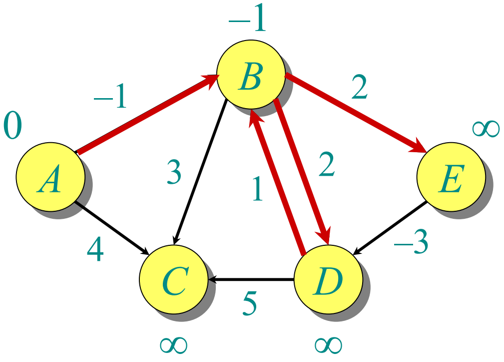
Order of edges: $(B,E), (D,B), (B,D), (A,B), (A,C), (D,C), (B,C), (E,D)$



A	B	C	D	E
0	∞	∞	∞	∞

Example of Bellman-Ford

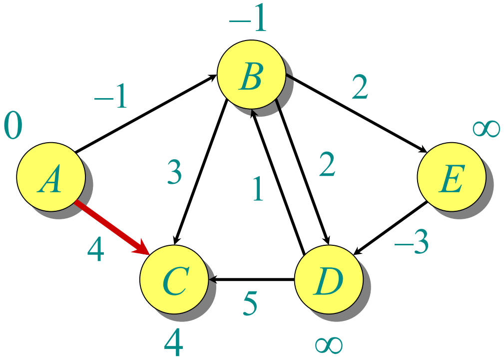
Order of edges: *(B,E), (D,B), (B,D), (A,B), (A,C), (D,C), (B,C), (E,D)*



A	*B*	*C*	*D*	*E*
0	∞	∞	∞	∞
0	–1	∞	∞	∞

Example of Bellman-Ford

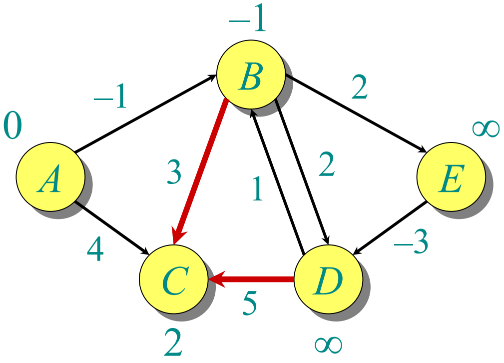
Order of edges: *(B,E), (D,B), (B,D), (A,B), (A,C), (D,C), (B,C), (E,D)*



A	*B*	*C*	*D*	*E*
0	∞	∞	∞	∞
0	–1	∞	∞	∞
0	–1	4	∞	∞

Example of Bellman-Ford

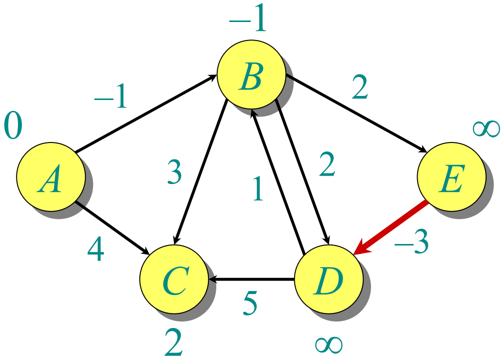
Order of edges: *(B,E), (D,B), (B,D), (A,B), (A,C), (D,C), (B,C), (E,D)*



A	*B*	*C*	*D*	*E*
0	∞	∞	∞	∞
0	–1	∞	∞	∞
0	–1	4	∞	∞
0	–1	2	∞	∞

Example of Bellman-Ford

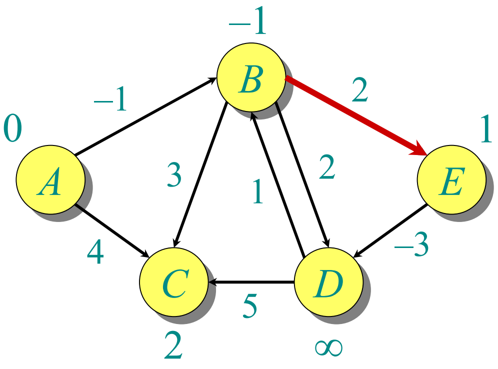
Order of edges: *(B,E), (D,B), (B,D), (A,B), (A,C), (D,C), (B,C), (E,D)*



A	*B*	*C*	*D*	*E*
0	∞	∞	∞	∞
0	–1	∞	∞	∞
0	–1	4	∞	∞
0	–1	2	∞	∞

Example of Bellman-Ford

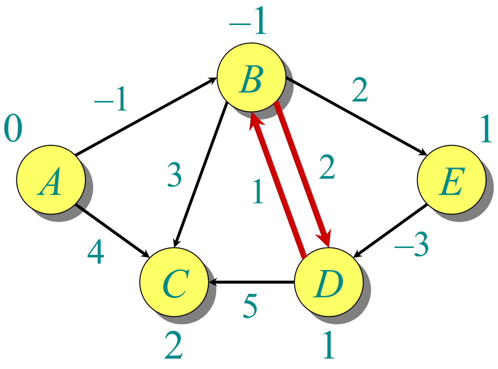
Order of edges: *(B,E), (D,B), (B,D), (A,B), (A,C), (D,C), (B,C), (E,D)*



A	*B*	*C*	*D*	*E*
0	∞	∞	∞	∞
0	–1	∞	∞	∞
0	–1	4	∞	∞
0	–1	2	∞	∞
0	–1	2	∞	1

Example of Bellman-Ford

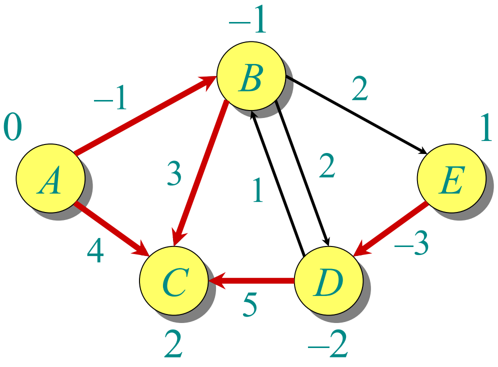
Order of edges: *(B,E), (D,B), (B,D), (A,B), (A,C), (D,C), (B,C), (E,D)*



A	*B*	*C*	*D*	*E*
0	∞	∞	∞	∞
0	–1	∞	∞	∞
0	–1	4	∞	∞
0	–1	2	∞	∞
0	–1	2	∞	1
0	–1	2	1	1

Example of Bellman-Ford

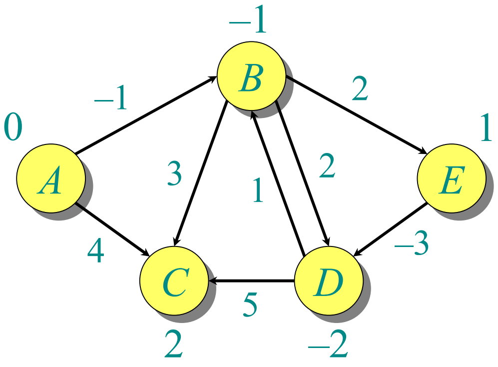
Order of edges: *(B,E), (D,B), (B,D), (A,B), (A,C), (D,C), (B,C), (E,D)*



A	*B*	*C*	*D*	*E*
0	∞	∞	∞	∞
0	–1	∞	∞	∞
0	–1	4	∞	∞
0	–1	2	∞	∞
0	–1	2	∞	1
0	–1	2	1	1
0	–1	2	–2	1

Example of Bellman-Ford

Order of edges: *(B,E), (D,B), (B,D), (A,B), (A,C), (D,C), (B,C), (E,D)*



A	*B*	*C*	*D*	*E*
0	∞	∞	∞	∞
0	–1	∞	∞	∞
0	–1	4	∞	∞
0	–1	2	∞	∞
0	–1	2	∞	1
0	–1	2	1	1
0	–1	2	–2	1

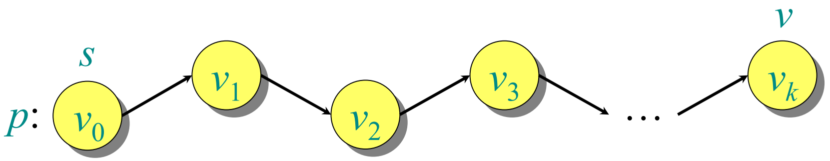
Note: Values decrease monotonically.

… and 2 more iterations

Correctness

Theorem. If $G = (V, E)$ contains no negative-weight cycles, then after the Bellman-Ford algorithm executes, $d[v] = \delta(s, v)$ for all $v \in V$.

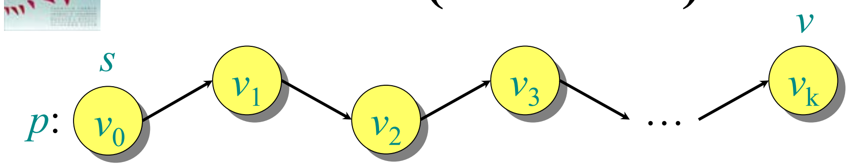
Proof. Let $v \in V$ be any vertex, and consider a shortest path p from s to v with the minimum number of edges.



Since p is a shortest path, we have

$$\delta(s, v_i) = \delta(s, v_{i-1}) + w(v_{i-1}, v_i) .$$

Correctness (continued)



Initially, $d[v_0] = 0 = \delta(s, v_0)$, and $d[s]$ is unchanged by subsequent relaxations.

- After 1 pass through E, we have $d[v_1] = \delta(s, v_1)$.
- After 2 passes through E, we have $d[v_2] = \delta(s, v_2)$.

  …
- After k passes through E, we have $d[v_k] = \delta(s, v_k)$.

Since G contains no negative-weight cycles, p is simple. Longest simple path has $\leq |V| - 1$ edges. □

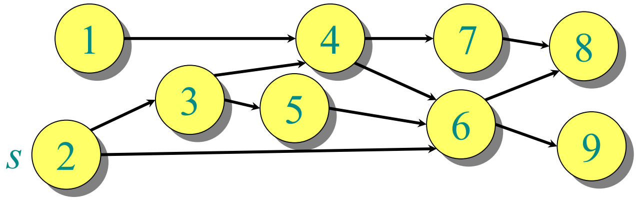
Detection of negative-weight cycles

Corollary. If a value $d[v]$ fails to converge after $|V| - 1$ passes, there exists a negative-weight cycle in G reachable from s. □

DAG shortest paths

If the graph is a ***directed acyclic graph*** (***DAG***), we first ***topologically sort*** the vertices.

- Determine $f : V \rightarrow \{1, 2, \ldots, |V|\}$ such that $(u, v) \in E \Rightarrow f(u) < f(v)$.
- $O(|V| + |E|)$ time using depth-first search.



Walk through the vertices $u \in V$ in this order, relaxing the edges in $Adj[u]$, thereby obtaining the shortest paths from s in a total of $O(|V| + |E|)$ time.

Shortest paths

Single-source shortest paths

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

All-pairs shortest paths

- Nonnegative edge weights
  - Dijkstra's algorithm $|V|$ times: $O(|V||E| \log |V|)$
- General
  - Bellman-Ford $|V|$ times: $O(|V|^2|E|)$
  - Floyd-Warshall: $O(|V|^3)$

All-pairs shortest paths

Input: Digraph $G = (V, E)$, where $|V| = n$, with edge-weight function $w : E \to \mathbb{R}$.
Output: $n \times n$ matrix of shortest-path lengths $\delta(i, j)$ for all $i, j \in V$.

Algorithm #1:

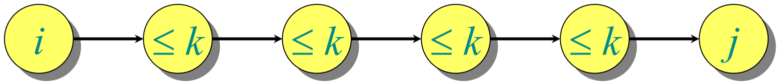
- Run Bellman-Ford once from each vertex.
- Time = $O(|V|^2|E|)$.
- But: Dense graph ⇒ $O(|V|^4)$ time.

Floyd-Warshall algorithm

Dynamic programming algorithm.

Assume $V=\{1, 2, \ldots, n\}$.

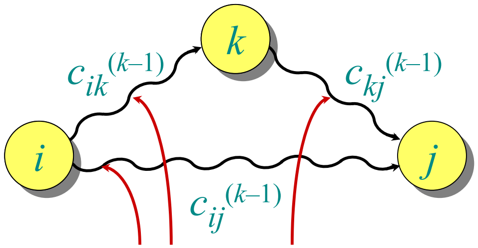
Define $c_{ij}^{(k)}$ = weight of a shortest path from i to j with intermediate vertices belonging to the set $\{1, 2, \ldots, k\}$.



Thus, $\delta(i, j) = c_{ij}^{(n)}$. Also, $c_{ij}^{(0)} = a_{ij}$.

Floyd-Warshall recurrence

$$c_{ij}^{(k)} = \min \{c_{ij}^{(k-1)}, c_{ik}^{(k-1)} + c_{kj}^{(k-1)}\}$$



intermediate vertices in $\{1, 2, \ldots, k\}$

Pseudocode for Floyd-Warshall

```
for $k \leftarrow 1$ to $n$ do
  for $i \leftarrow 1$ to $n$ do
    for $j \leftarrow 1$ to $n$ do
      if $c_{ij}^{(k-1)} > c_{ik}^{(k-1)} + c_{kj}^{(k-1)}$ then
          $c_{ij}^{(k)} \leftarrow c_{ik}^{(k-1)} + c_{kj}^{(k-1)}$
      else
          $c_{ij}^{(k)} \leftarrow c_{ij}^{(k-1)}$
```

relaxation

- Runs in $\Theta(n^3)$ time and space
- Simple to code.
- Efficient in practice.

Shortest paths

Single-source shortest paths

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

adj. lis

All-pairs shortest paths

- Nonnegative edge weights
  - Dijkstra's algorithm $|V|$ times: $O(|V||E| \log |V|)$

adj. list

- General
  - Bellman-Ford $|V|$ times: $O(|V|^2|E|)$
  - Floyd-Warshall: $O(|V|^3)$

adj. list

adj. matrix