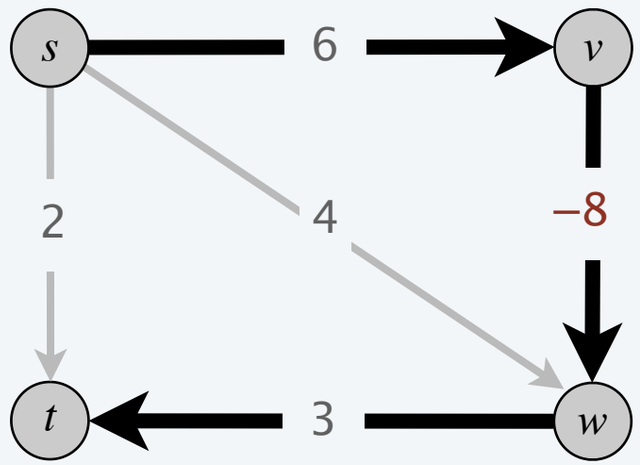
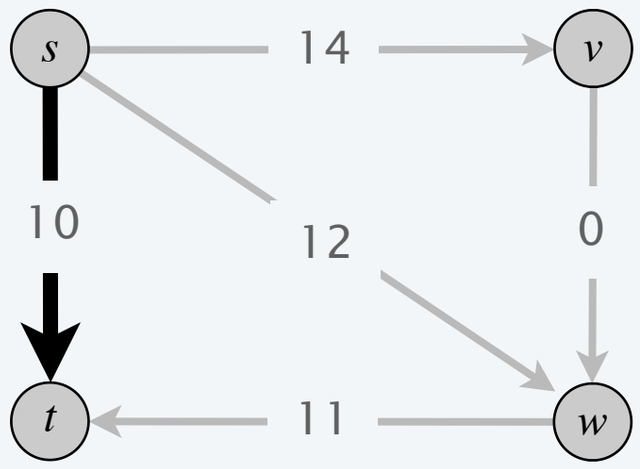
Shortest paths with negative weights: failed attempts

Dijkstra. May not produce shortest paths when edge lengths are negative.

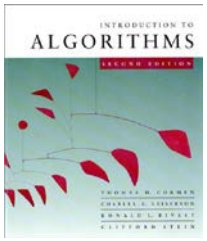


Dijkstra selects the vertices in the order s, t, w, v
But shortest path from s to t is $s \rightarrow v \rightarrow w \rightarrow t$.

Reweighting. Adding a constant to every edge length does not necessarily make Dijkstra's algorithm produce shortest paths.



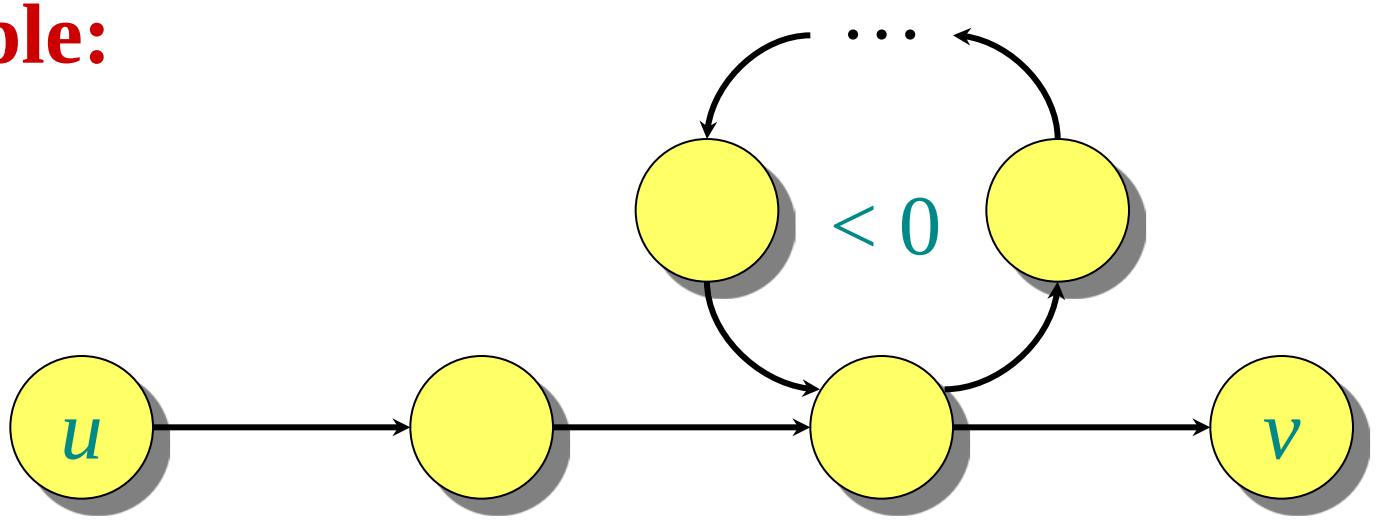
Adding 8 to each edge weight changes the shortest path from $s \rightarrow v \rightarrow w \rightarrow t$ to $s \rightarrow t$.

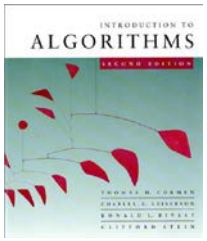


Negative-weight cycles

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

Example:

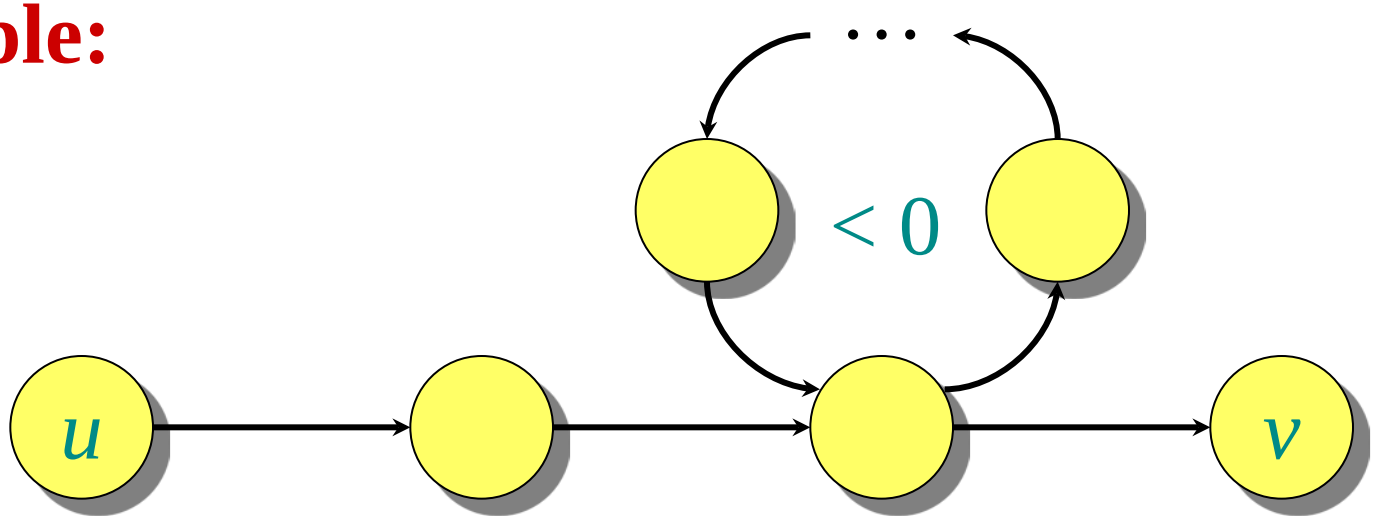




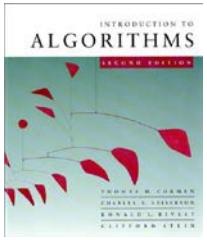
Negative-weight cycles

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 lengths 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 $v \in V - \{s\}$
  do $d[v] \leftarrow \infty$

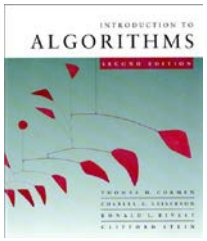
} initialization

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

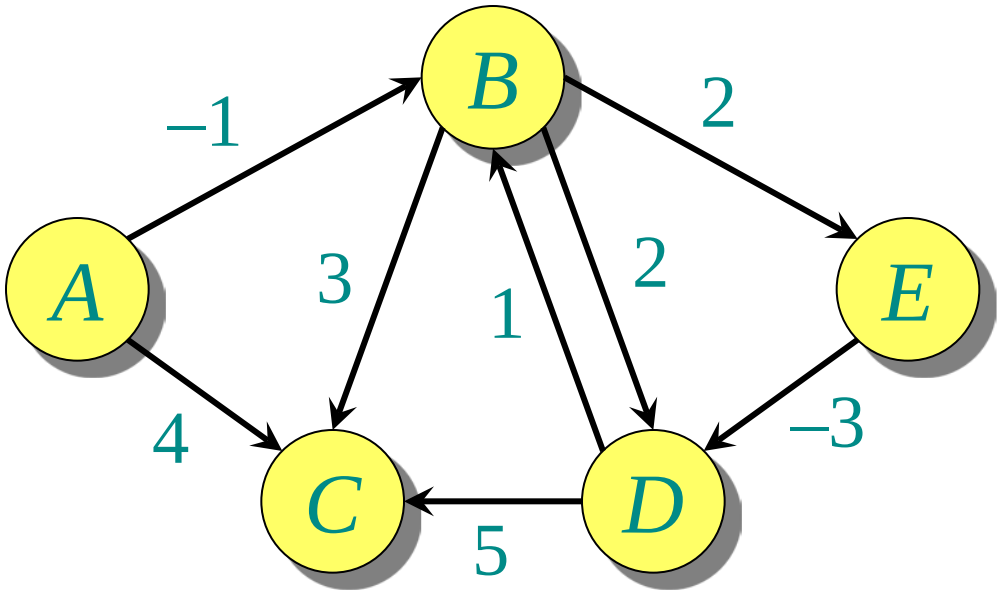
} ***relaxation step***

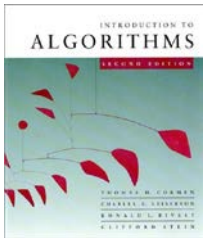
for each edge $(u, v) \in E$
  do if $d[v] > d[u] + w(u, v)$
    then report that a negative-weight cycle exists

At the end, $d[v] = \delta(s, v)$, if no negative-weight cycles.
Time = $O(VE)$.

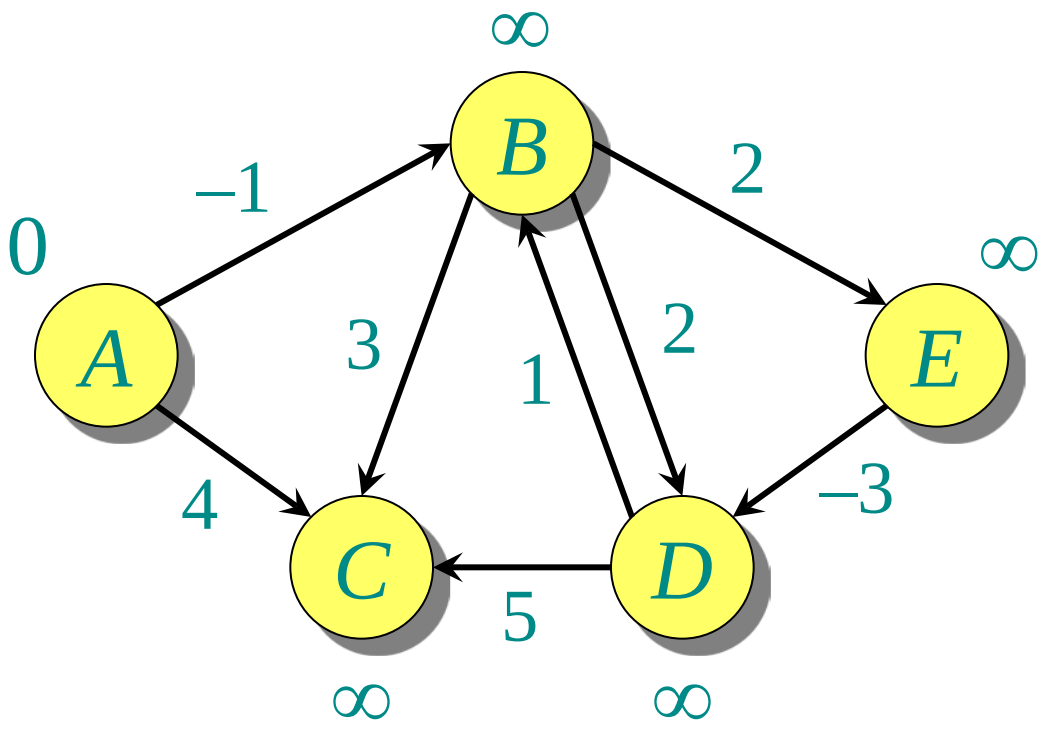


Example of Bellman-Ford

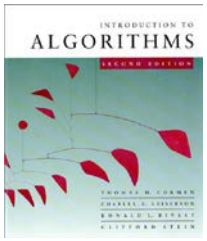




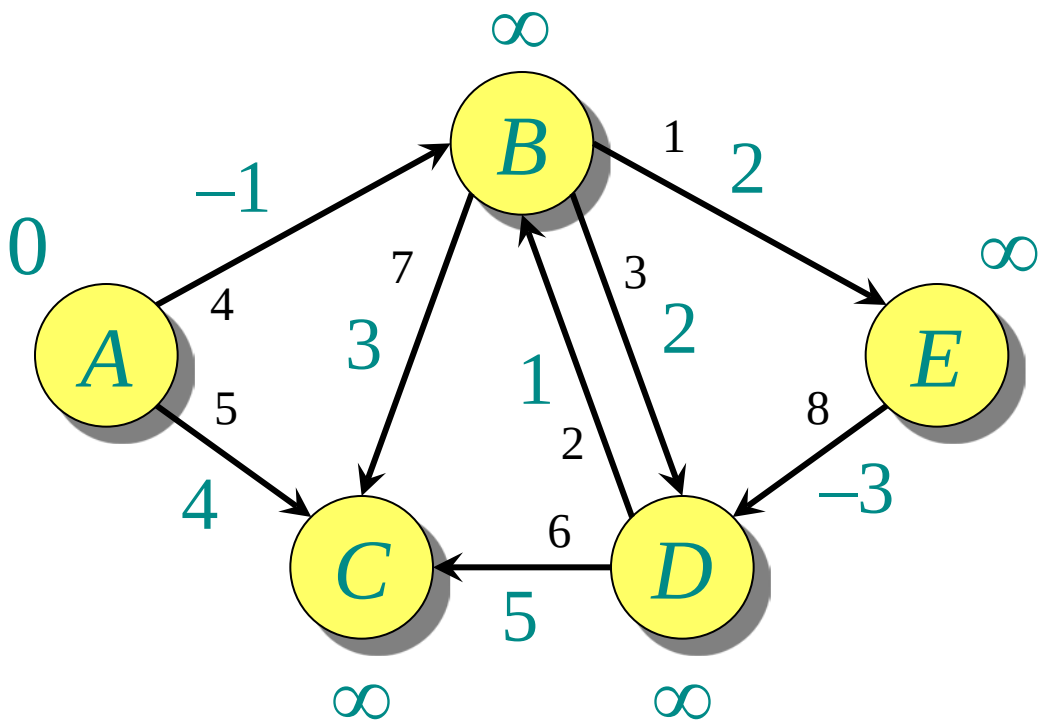
Example of Bellman-Ford



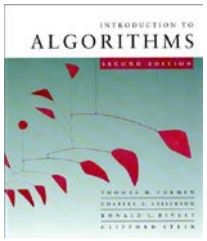
Initialization.



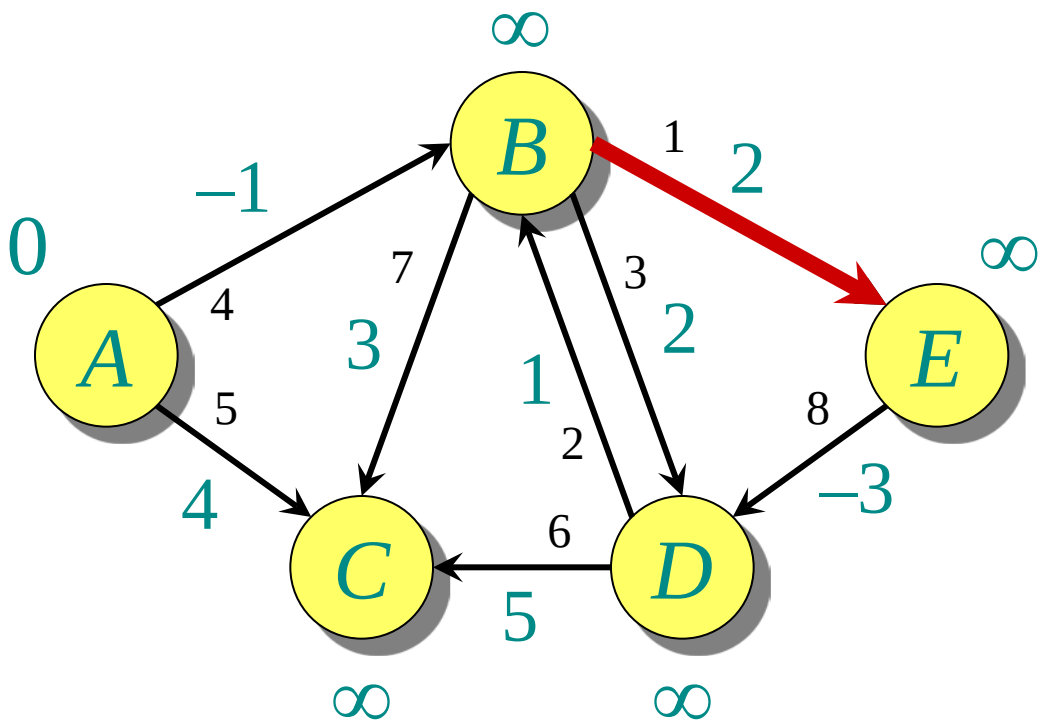
Example of Bellman-Ford

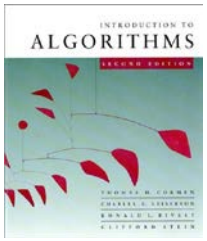


Order of edge relaxation.

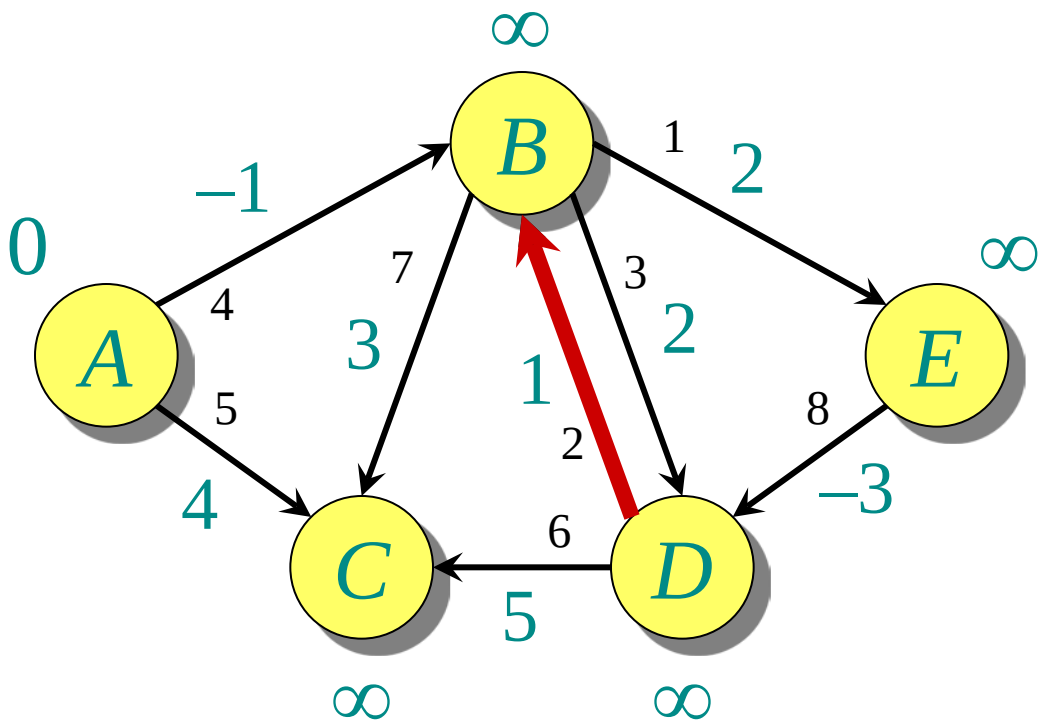


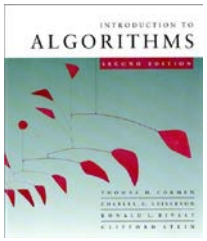
Example of Bellman-Ford



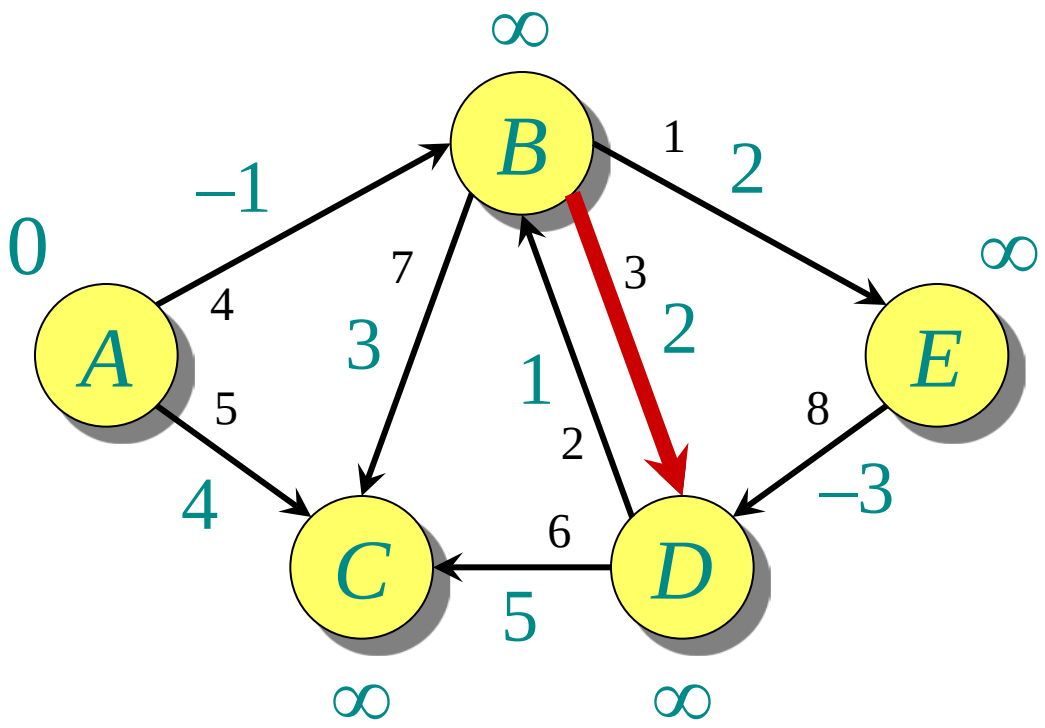


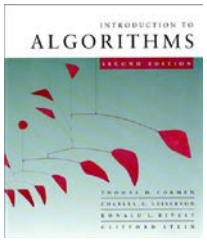
Example of Bellman-Ford



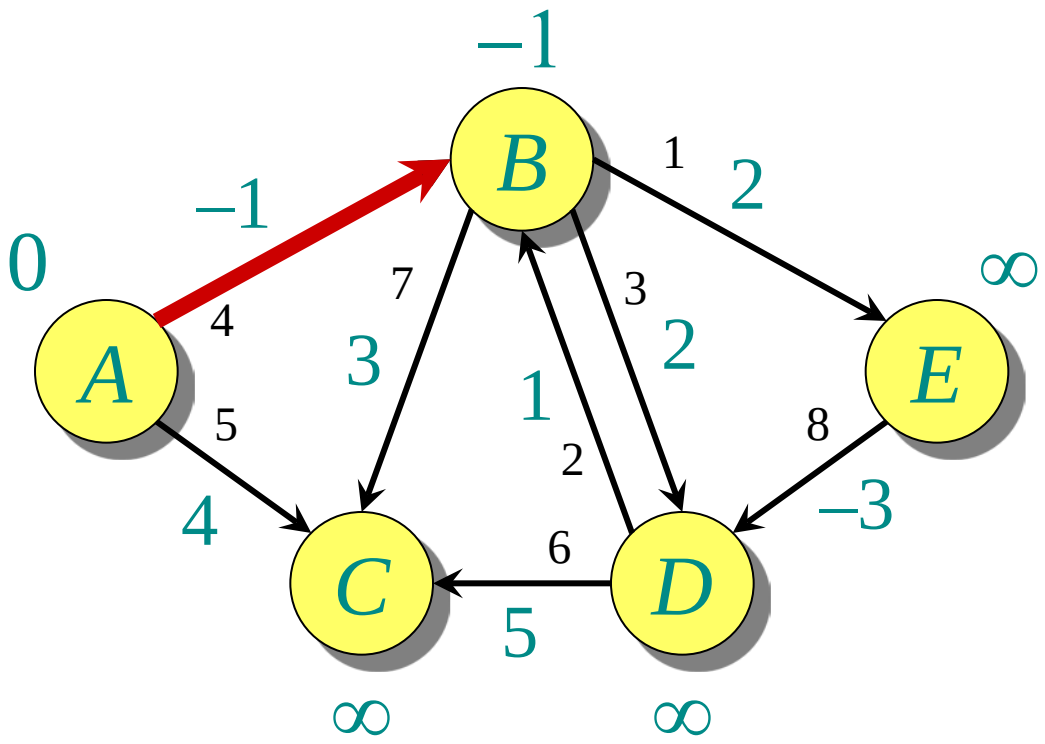


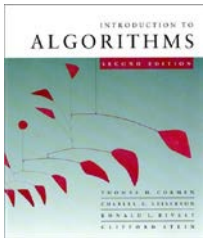
Example of Bellman-Ford



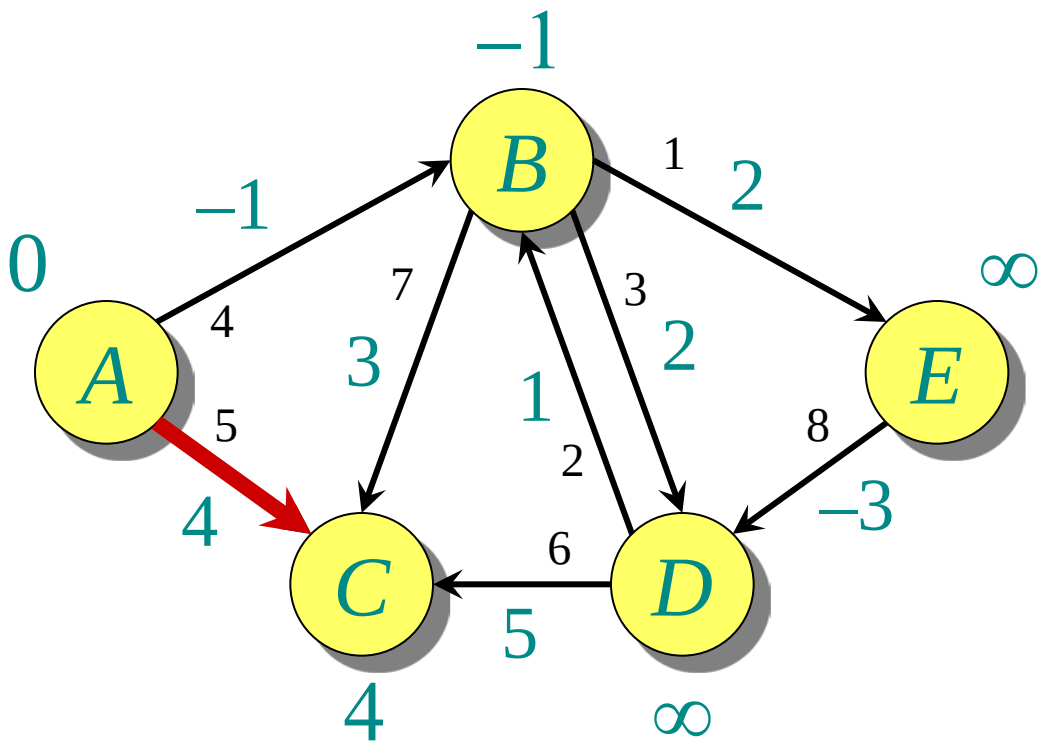


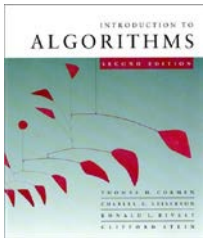
Example of Bellman-Ford



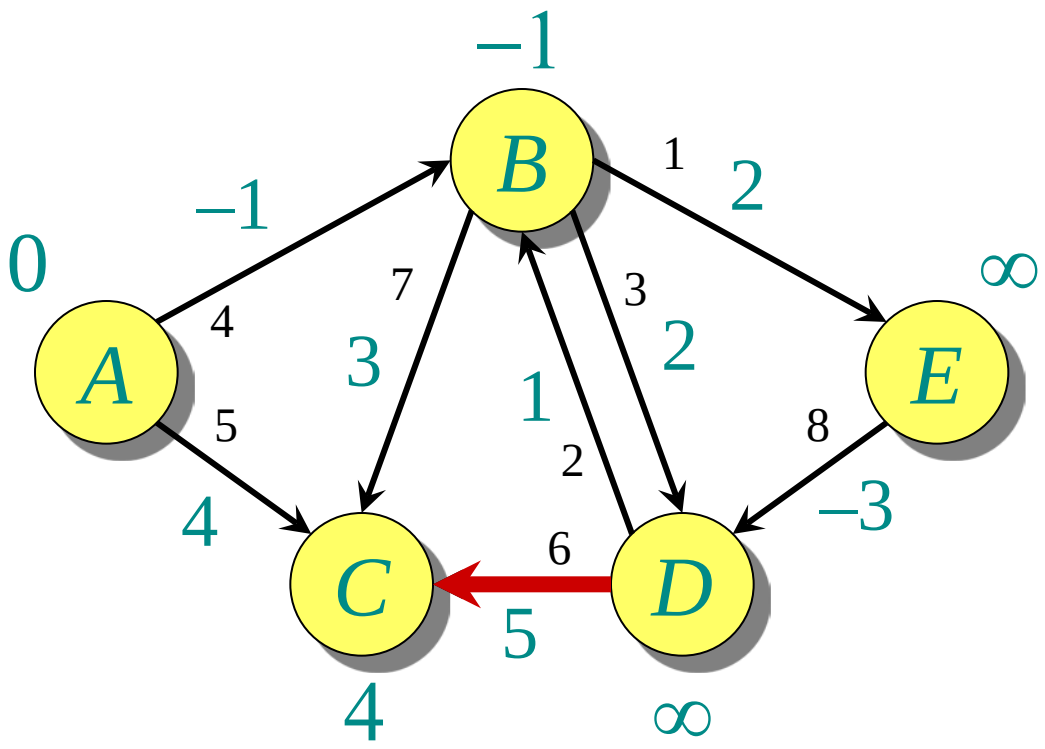


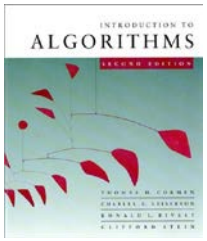
Example of Bellman-Ford



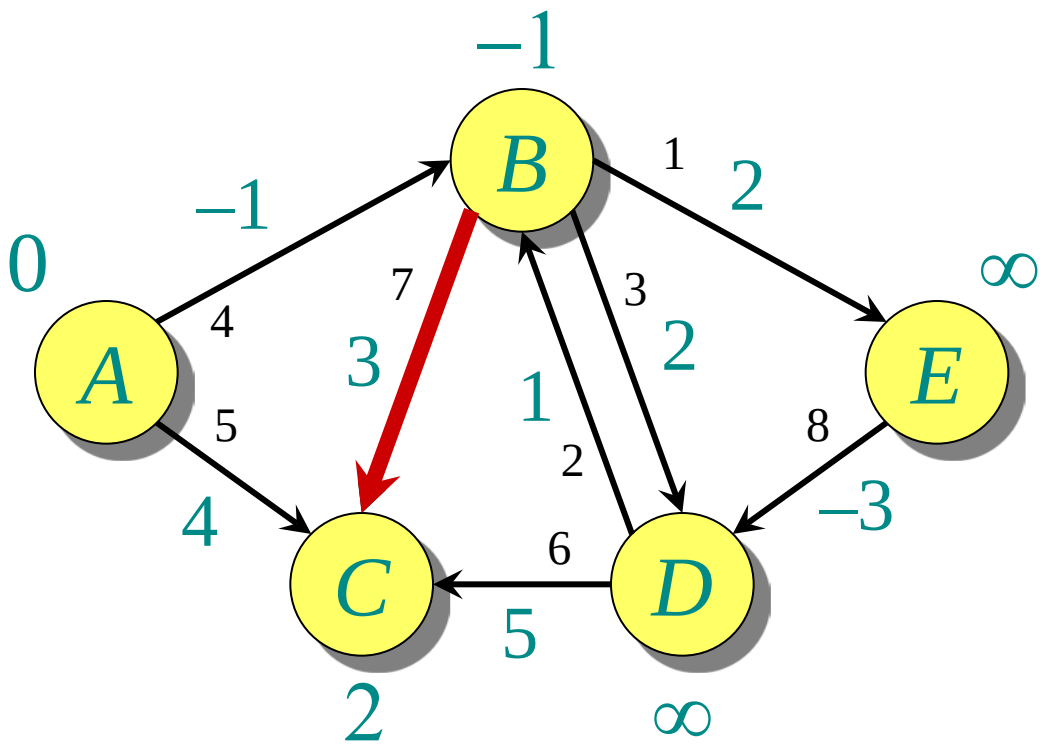


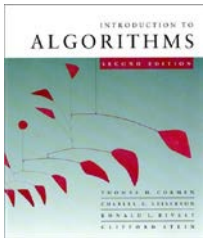
Example of Bellman-Ford



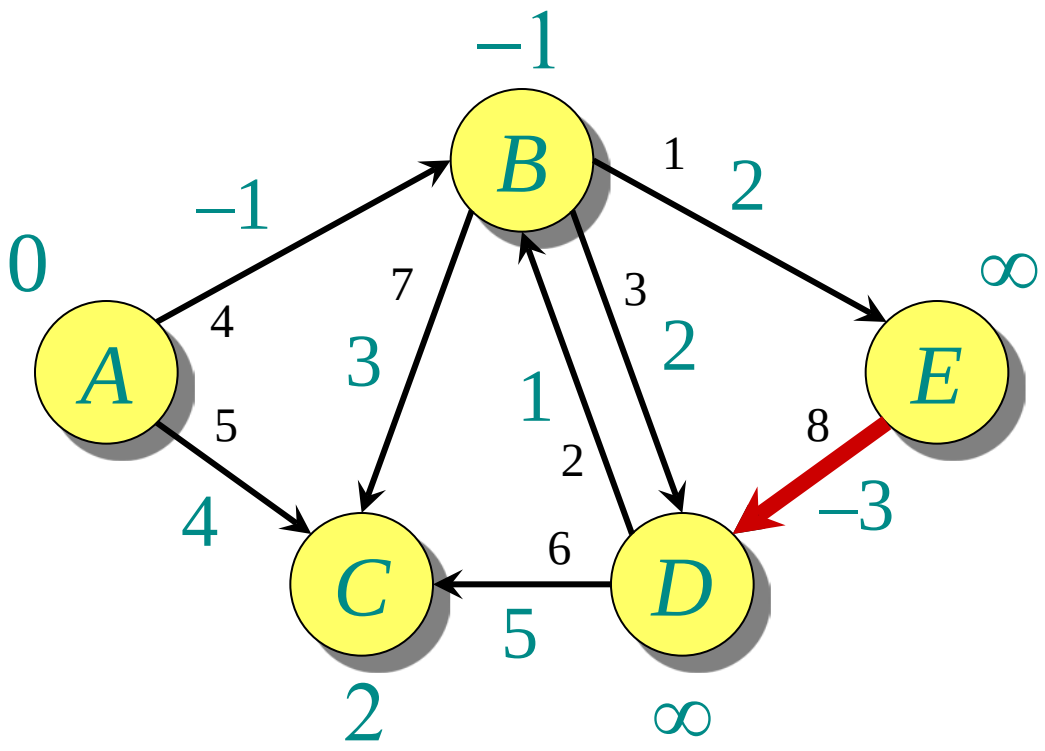


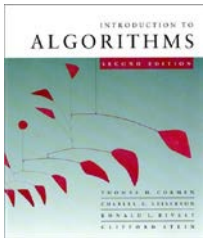
Example of Bellman-Ford



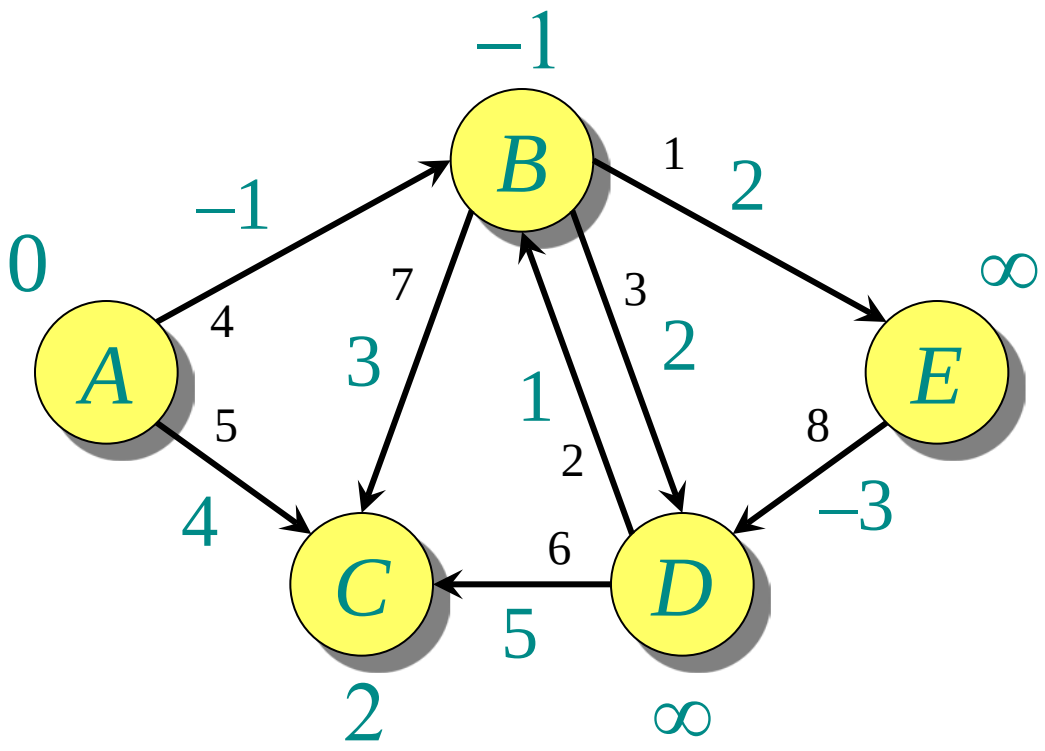


Example of Bellman-Ford

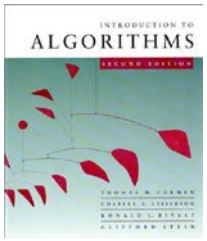




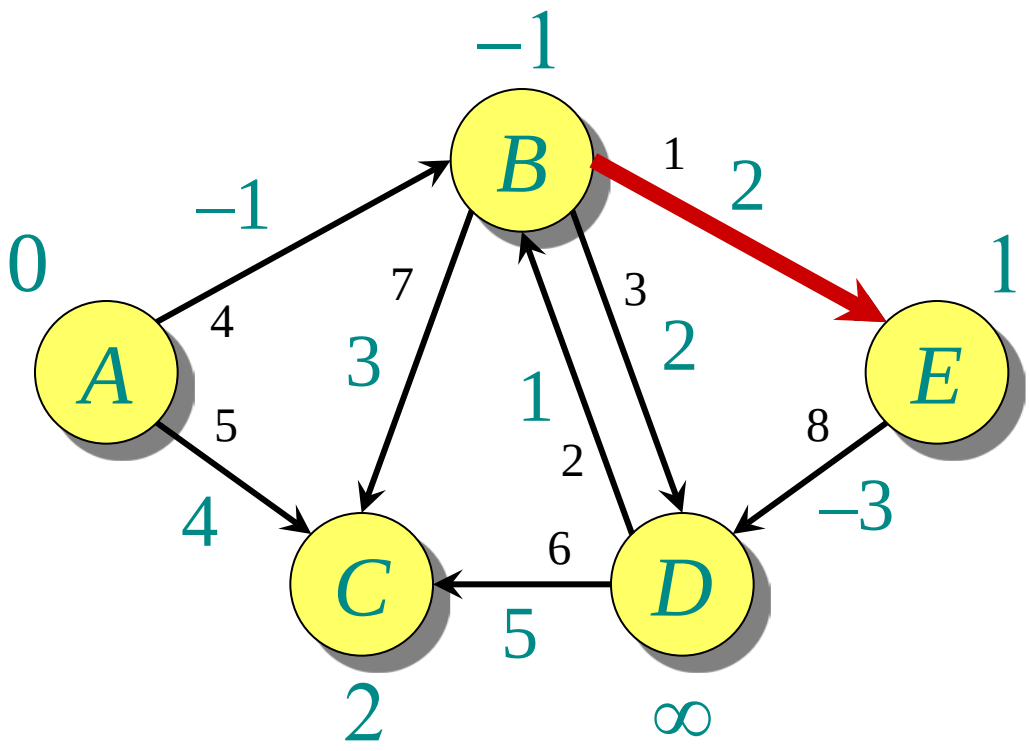
Example of Bellman-Ford

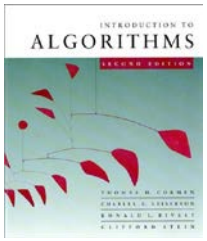


End of pass 1.

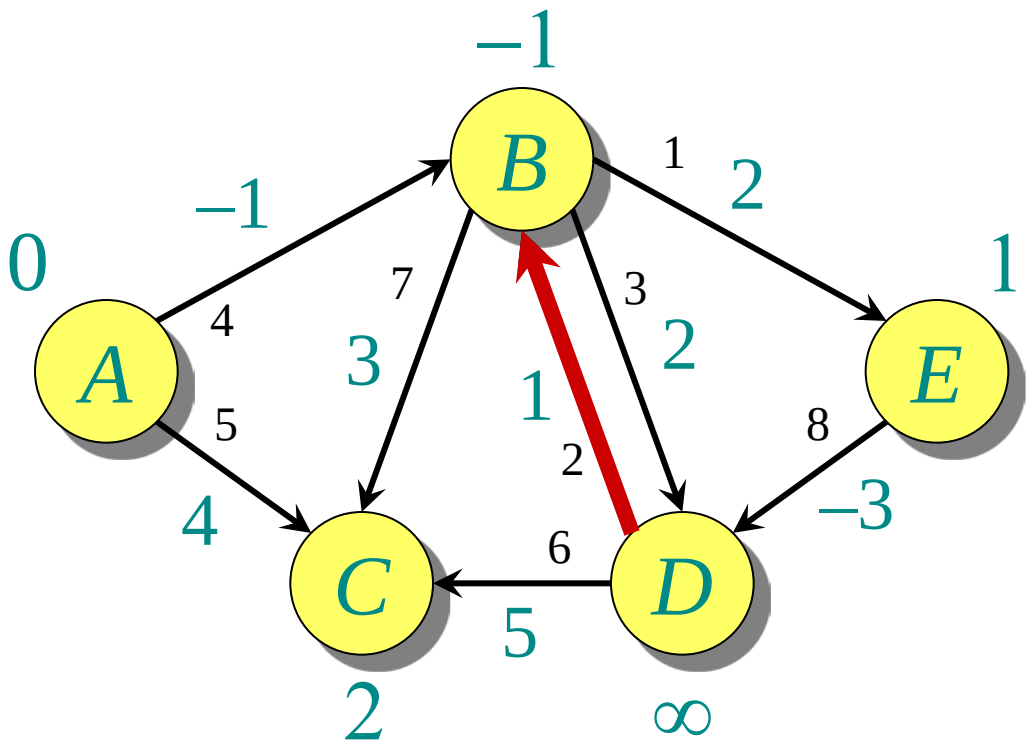


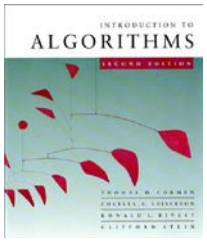
Example of Bellman-Ford



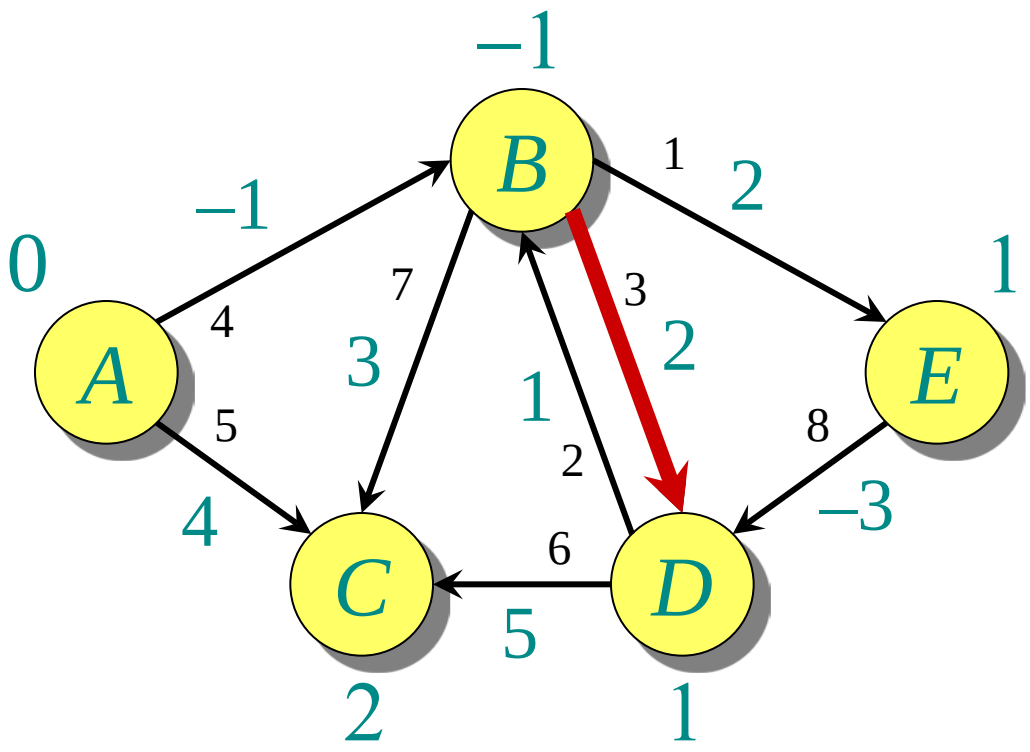


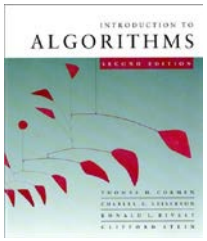
Example of Bellman-Ford



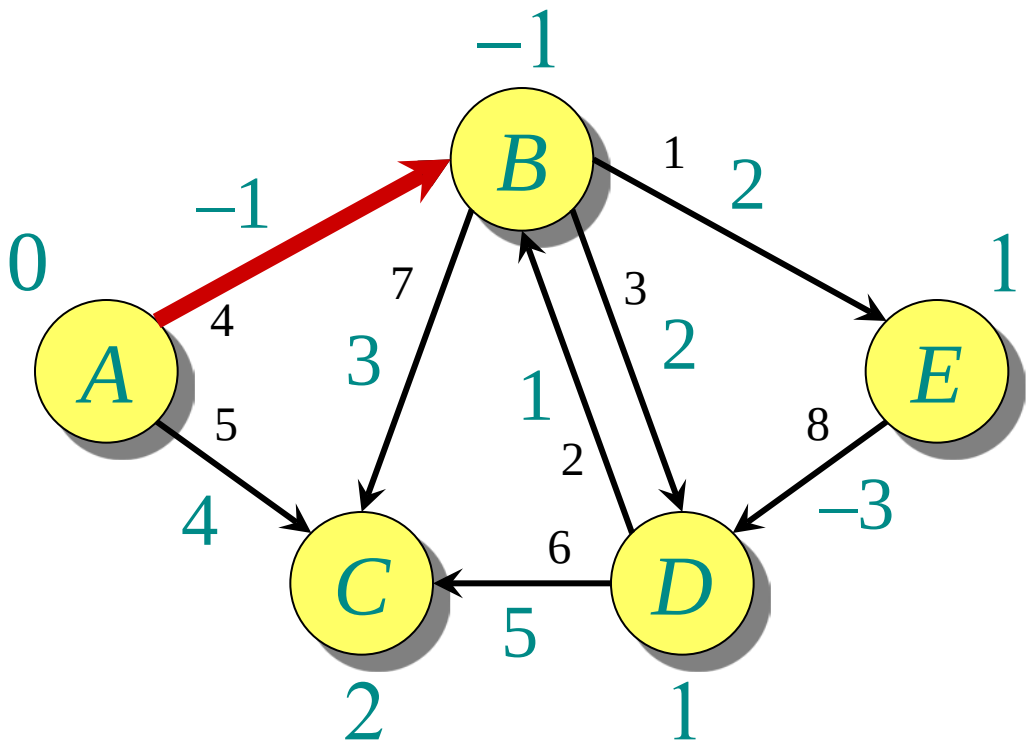


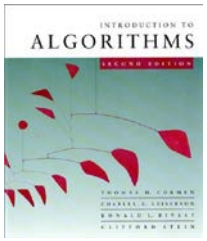
Example of Bellman-Ford



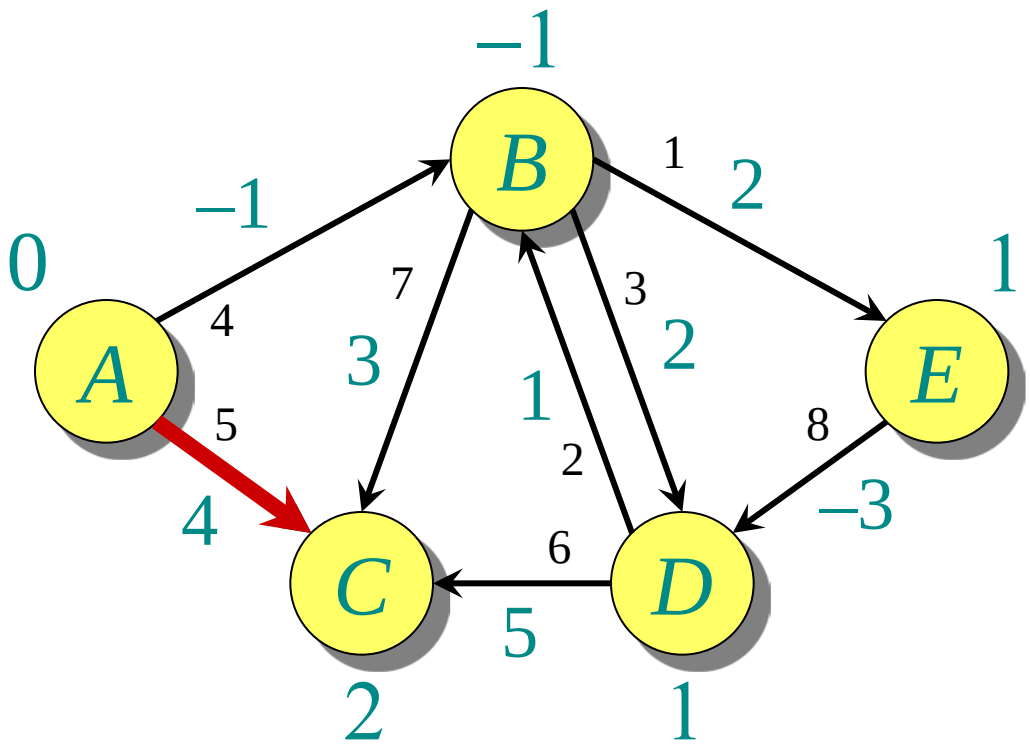


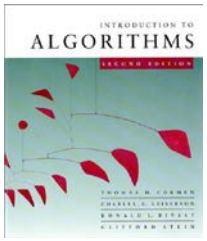
Example of Bellman-Ford



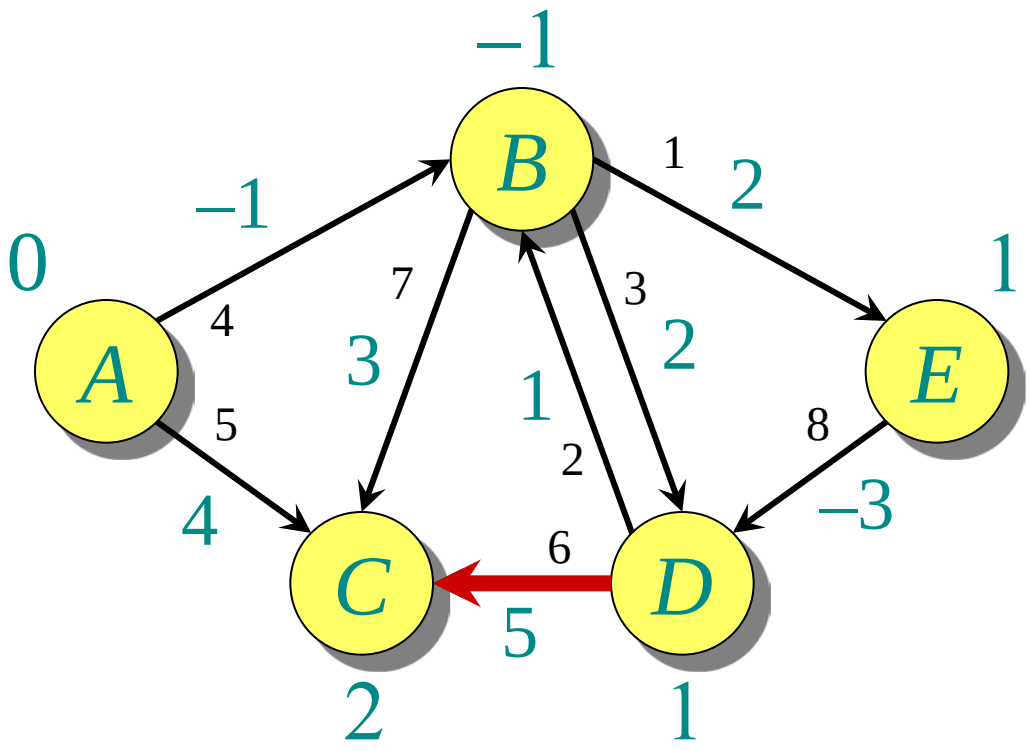


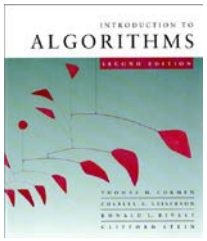
Example of Bellman-Ford



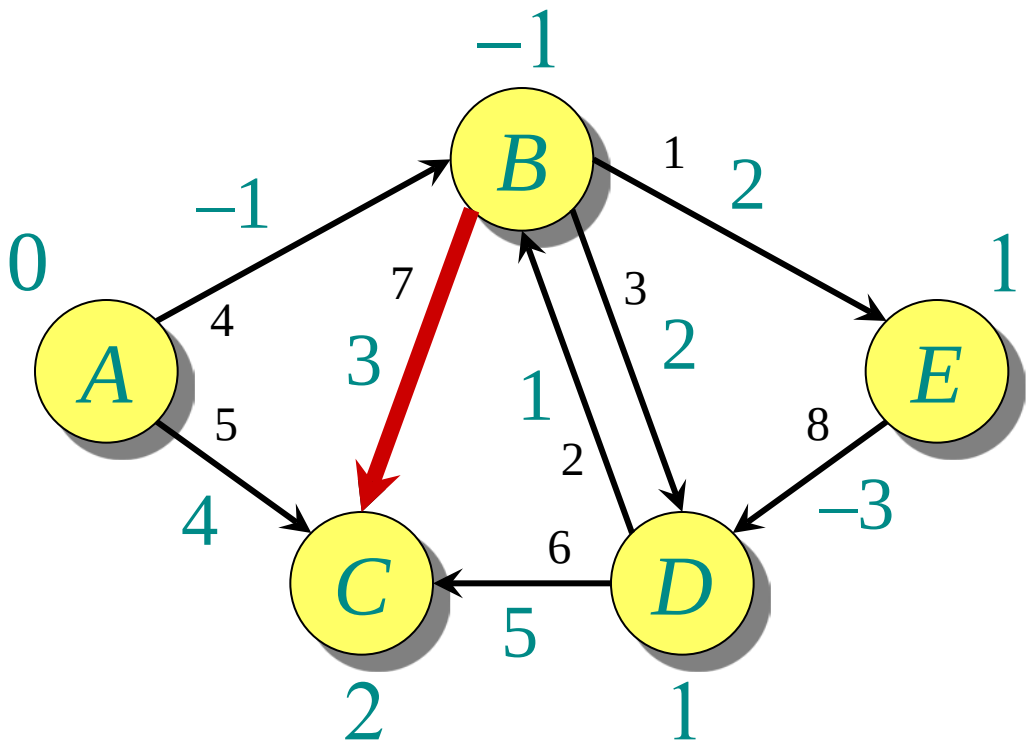


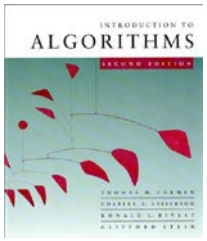
Example of Bellman-Ford



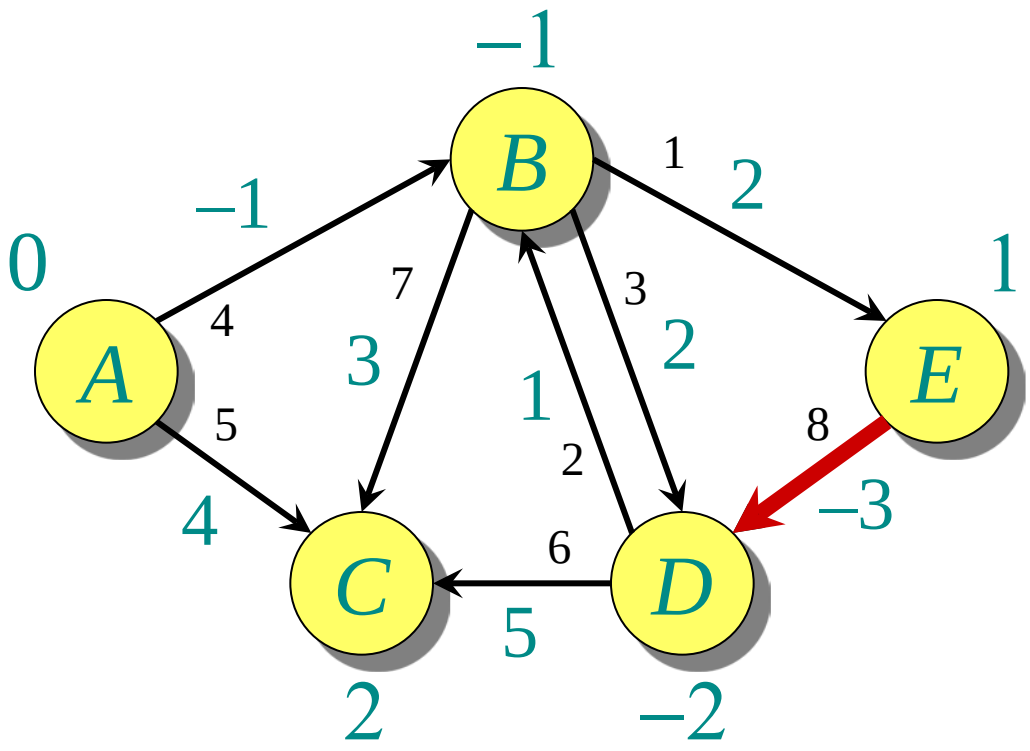


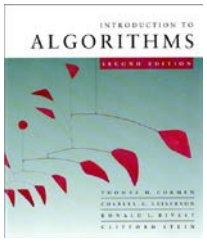
Example of Bellman-Ford



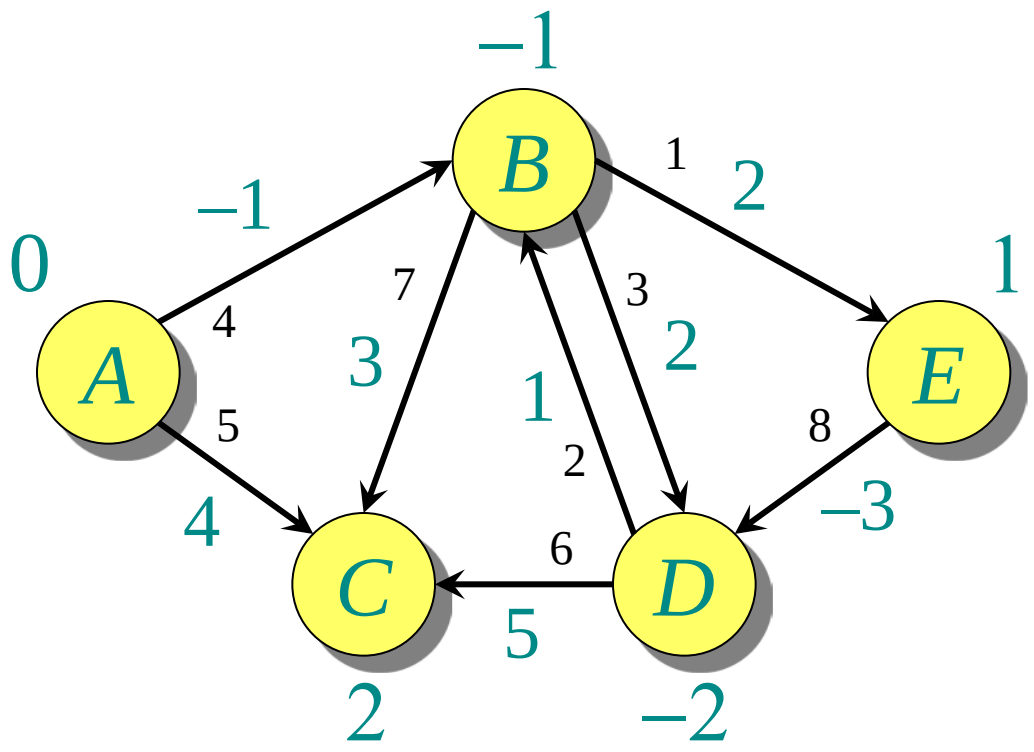


Example of Bellman-Ford





Example of Bellman-Ford



End of pass 2 (and 3 and 4).

A dynamic programming view of Bellman-Ford

Another view

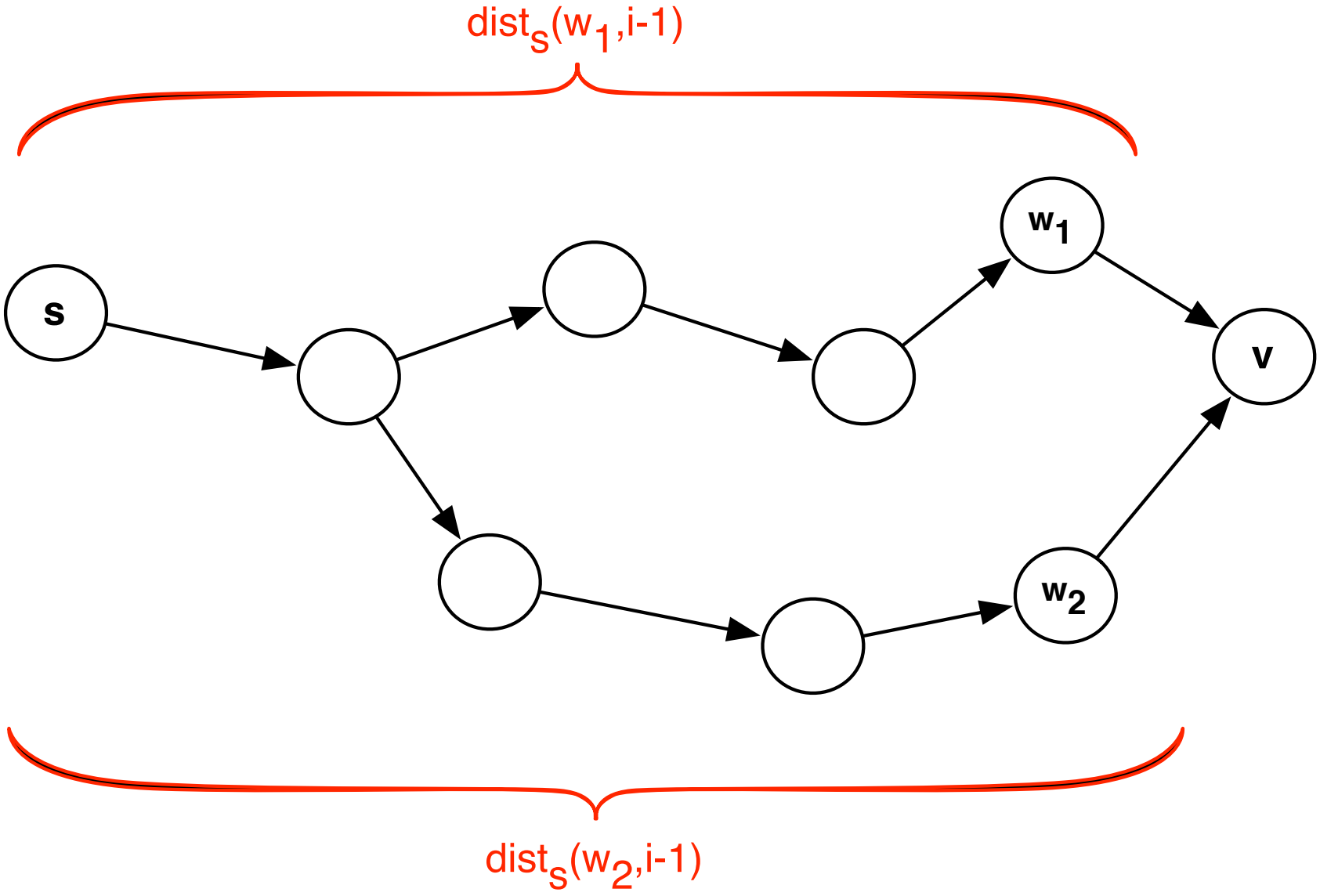
Definition. Let $dist_s(v, i)$ be minimum cost of a path from s to v that uses at most i edges.

Can recursively define $dist_s(v, i)$:

1. If best $s - v$ path uses at most $i - 1$ edges, then $dist_s(v, i) = dist_s(v, i - 1)$.

2. If best $s - v$ uses i edges, and the last edge is (w, v), then $dist_s(v, i) = d(w, v) + dist_s(w, i - 1)$.

Subproblems, picture



Recurrence

Let $N(w)$ be the neighbors of w.

$dist_s(v, i)$ = cost of best path from s to v using at most i edges.

Recurrence:

$$dist_s(v, i) = \min \begin{cases} dist_s(v, i-1) \\ \min_{w \in N(v)} dist_s(w, i-1) + d(w, v) \end{cases}$$

Base case: $dist_s(v, 1) = d(s, v)$ or ∞ if (s, v) does not exist

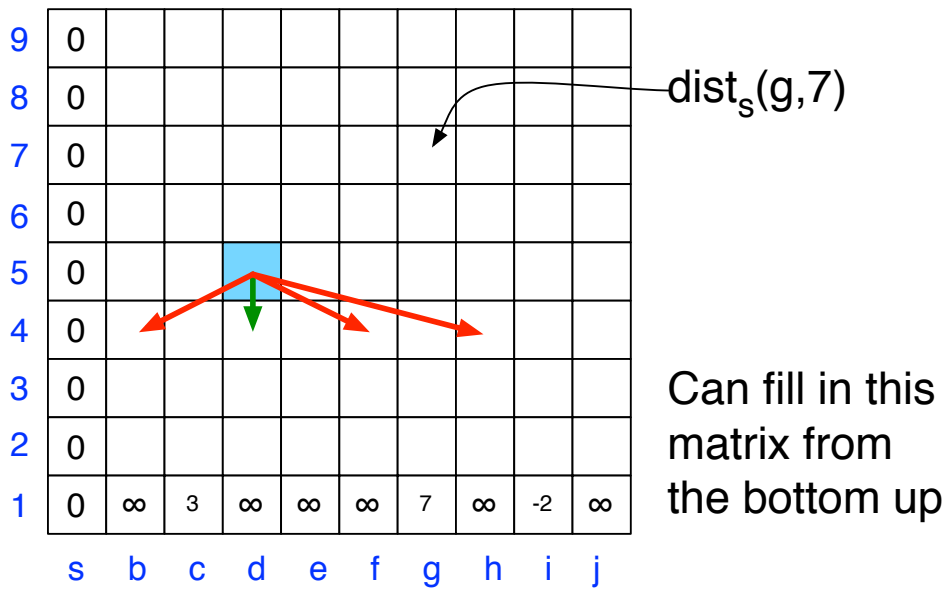
Goal: Compute $dist_s(t, n-1)$.

Important Facts About This Recurrence

Recurrence:

$$dist_s(v,i) = \min \begin{cases} dist_s(v, i-1) \\ \min_{w \in N(v)} dist_s(w, i-1) + d(w,v) \end{cases}$$

- $dist_s(v,x)$ depends only on $dist_s(w,y)$ for which y is smaller than x.
- There are only $|V| \times (|V|-1)$ possible arguments for $dist_s(\cdot,\cdot)$.



Code

```
BellmanFord(G=(V,E), s, t):
 Initialize dist_s[x, 1] to d(s,x) for all x
 For i = 1,...,|V|-1:
   For v in V:
     // find the best w on which to apply the Ford rule
     best_w = None
     for w in N(v):  // N(v) are neighbors of v
        best_w = min(best_w, dist_s[w, i-1] + d[w,v])

     dist_s[v,i] = min(best_w, dist_s[v, i-1])
   EndFor
 EndFor
 Return dist_s[t, n-1]
```

Running Time

Simple Analysis:

- $O(n^2)$ subproblems
- $O(n)$ time to compute each entry in the table (have to search over all possible neighbors w).
- Therefore, runs in $O(n^3)$ time.

A better analysis:

- Let n_v be the number of edges entering v.
- Filling in each entry actually only takes $O(n_v)$ time.
- Total time $= O\left(n \sum_{v \in V} n_v\right) = O(nm)$.

Single-source shortest paths with negative weights

year	worst case	discovered by
1955	$O(n^4)$	Shimbel
1956	$O(m\, n^2\, W)$	Ford
1958	$O(m\, n)$	Bellman, Moore
1983	$O(n^{3/4}\, m \log W)$	Gabow
1989	$O(m\, n^{1/2} \log(nW))$	Gabow–Tarjan
1993	$O(m\, n^{1/2} \log W)$	Goldberg
2005	$O(n^{2.38}\, W)$	Sankowsi, Yuster–Zwick
2016	$\tilde{O}(n^{10/7} \log W)$	Cohen–Mądry–Sankowski–Vladu
20xx		

single-source shortest paths with weights between –W and W