

CS 430 Lecture 27 Activities

Shortest Path Problem

How to find the shortest route between two points on a map.

Input

- Directed graph $G = (V, E)$
- Weight function $w : E \rightarrow \mathbf{R}$

Weight of path

$$\begin{aligned} p &= \langle v_0, v_1, \dots, v_k \rangle \\ &= \sum_{i=1}^k w(v_{i-1}, v_i) \\ &= \text{sum of edge weights on path } p. \end{aligned}$$

Shortest-path weight u to v

$$\delta(u, v) = \begin{cases} \min\{w(p) : u \rightsquigarrow^p v\} & \text{if there exists a path } u \rightsquigarrow v, \\ \infty & \text{otherwise.} \end{cases}$$

Shortest path u to v is any path p such that $w(p) = \delta(u, v)$.

Variants

- Single-source: Find shortest paths from a given source vertex $s \in V$ to every vertex $v \in V$.
- Single-destination: Find shortest paths to a given destination vertex.
- Single-pair: Find shortest path from u to v . No way known that's better in worst case than solving single-source.
- All-pairs: Find shortest path from u to v for all $u, v \in V$. We'll see algorithms for all-pairs in the next chapter.

Negative-weight edges – OK, as long as no negative-weight cycles are reachable from the source.

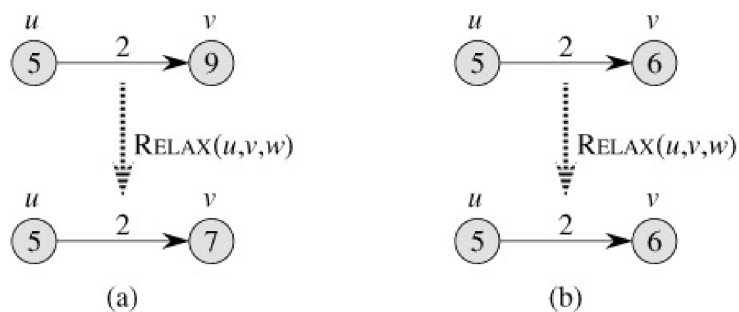
- If we have a negative-weight cycle, just keep going around it, and get $w(s, v) = -\infty$ for all v on the cycle.
- But OK if the negative-weight cycle is not reachable from the source.
- Some algorithms work only if there are no negative-weight edges in the graph.

Algorithm 27.2 Relaxing an Edge

```

1: function RELAX( $u, v, w$ )
2:   if  $d[v] > d[u] + w(u, v)$  then
3:      $d[v] \leftarrow d[u] + w(u, v)$ 
4:      $\pi[v] \leftarrow u$ 
5:   end if
6: end function

```



1. What would the brute force approach be to solve the shortest path problem, and what is its run time?

1 Path of 1 edge
 $|V| - 2$ Paths of 2 edges
 $(|V| - 2)(|V| - 3)$ Paths of 3 edges
 \vdots
 $|V|!$ Paths of V edges

2. Prove optimal substructure for the shortest path problem. Since shortest paths contain shortest subpaths (optimal solution to subproblem must in in optimal answer to problem.) $G\{V, E\}$, weight function on edge given source v , find shortest path to u

Output of single-source shortest-path algorithm For each vertex $v \in V$:

- $d[v] = \delta(s, v)$, Initially, $d[v] = \infty$; reduces as algorithms progress. But always maintain $d[v] \leftarrow \delta(s, v)$. Call $d[v]$ a shortest-path estimate.
- $\pi[v]$ = predecessor of v on a shortest path from s . If no predecessor, $\pi[v] = NIL$, π induces a tree—shortest-path tree.

Initialization – All the shortest-paths algorithms start with INIT-SINGLE-SOURCE.

Algorithm 27.1 Single Source Initialization

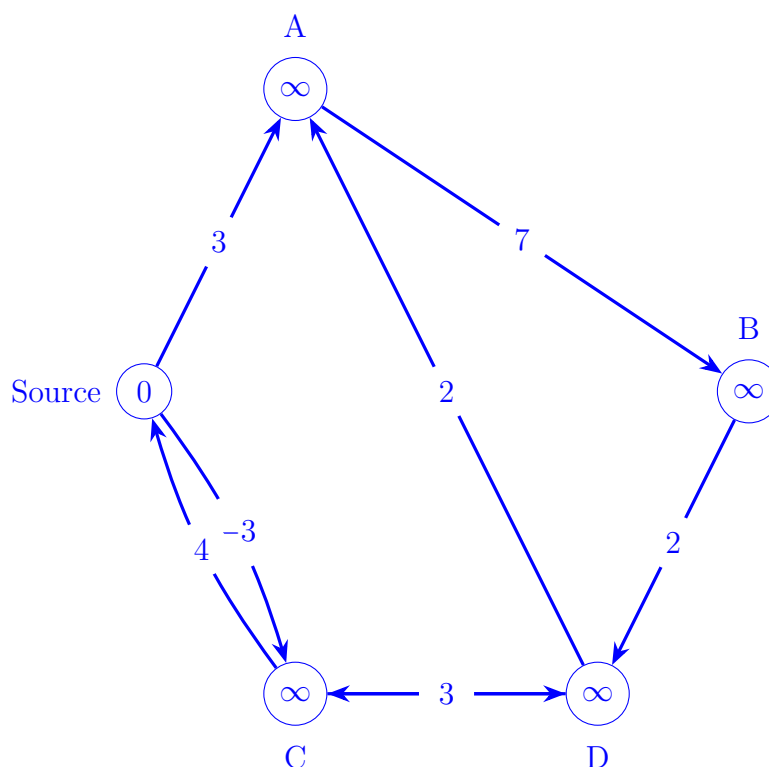
```

1: function INIT-SINGLE-SOURCE( $V, s$ )
2:   for all  $v \in V$  do
3:      $d[v] \leftarrow \infty$ 
4:      $\pi[v] \leftarrow NIL$ 
5:   end for
6:    $d[s] \leftarrow 0$ 
7: end function

```

Relaxing an edge (u, v) - Can we improve the shortest-path estimate (best seen so far) from the source s to v be going through u and taking edge (u, v) ?

The algorithms differ in the order and how many times they relax each edge.



Shortest Path Algorithm - Bellman-Ford

The most straightforward of the “relax an edge” algorithms.¹ Relaxes the edges in a fixed order (any fixed order) $|V| - 1$ times.² Not a greedy algorithm.

- Allows negative-weight edges.
 - Computes $d[v]$ and $\pi[v]$ for all $v \in V$.
 - Returns TRUE if no negative-weight cycles are reachable from s , FALSE otherwise.
3. Execute BELLMAN-FORD on the above graph from source s for this edge order $(t, x), (t, y), (t, z), (x, t), (y, x), (y, z), (z, x), (z, s), (s, t), (s, y)$. Update the $d[v]$ and $\pi[v]$ values for each iteration.

¹Works on any graph with no negative cycles, and the algorithm finds the negative cycles.

²Most # of edges on short path.

Algorithm 27.3 Bellman-Ford Shortest Path Algorithm
 $O(|V||E|) = O(|V|^3)$

```

1: function BELLMAN-FORD( $V, E, w, s$ )
2:   INIT-SINGLE-SOURCE( $V, s$ )
3:   for  $i \leftarrow 1$  to  $|V| - 1$  do
4:     for all edge  $(u, v) \in E$  do
5:       RELAX( $u, v, w$ )
6:     end for
7:   end for
8:   for all edge  $(u, v) \in E$  do
9:      $\triangleright$  All edges, in any order, same order each time
10:    if  $d[v] > d[u] + w(u, v)$  then            $\triangleright$  Must be a
        negative cycle
11:      return FALSE
12:    end if
13:  end for
14:  return TRUE
15: end function

```

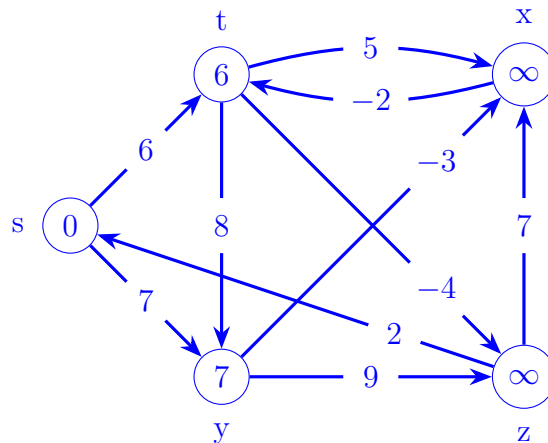
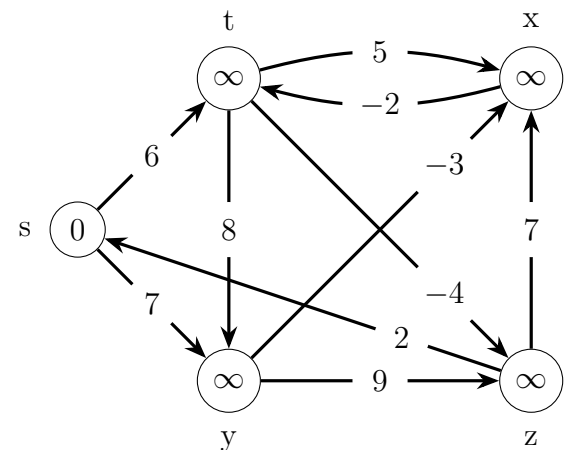


Figure 27.1: Example of the Execution of Bellman-Ford Step 1.

4. What is the runtime of Bellman-Ford? $O(|V||E|) = O(|V|^3)$

5. Prove Bellman-Ford is correct.

Values you get on each pass how quickly it converges depends on order of relaxation. But guaranteed to converged after $|V| - 1$ passes, assume no negative-weight cycles.