Notes, 4(b)

ECE 606

Single-source shortest paths

In a ***shortest-paths problem***, we are given a weighted, directed graph $G = (V, E)$, with weight function $w : E \to \mathbf{R}$ mapping edges to real-valued weights. The ***weight*** of path $p = \langle v_0, v_1, \ldots, v_k \rangle$ is the sum of the weights of its constituent edges:

$$w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i) .$$

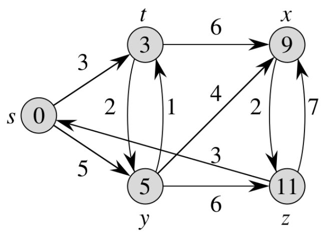
We define the ***shortest-path weight*** from u to v by

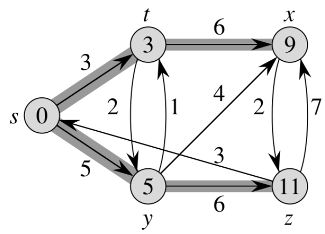
$$\delta(u, v) = \begin{cases} \min\{w(p) : u \overset{p}{\rightsquigarrow} v\} & \text{if there is a path from } u \text{ to } v \text{ ,} \\ \infty & \text{otherwise .} \end{cases}$$

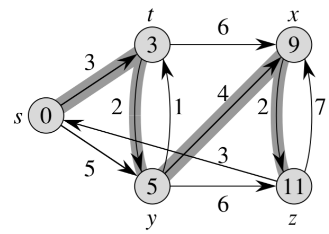
A ***shortest path*** from vertex u to vertex v is then defined as any path p with weight $w(p) = \delta(u, v)$.

The single-source shortest distances and paths problem:

- Inputs:
  1. Weighted directed or undirected graph, $G = \langle V, E, w \rangle$, and,
  2. A source-vertex, $s \in V$.
- Output:
  - Shortest-path weights from s to every $u \in V$.
    - Auxiliary output: a shortest paths tree rooted at s.







"Optimal substructure" of shortest paths:

Claim 1. *A subpath of a shortest path is a shortest path. That is, if* $u \rightsquigarrow x \rightsquigarrow y \rightsquigarrow v$ *is a shortest path from* u *to* v*, then the subpath* $x \rightsquigarrow y$ *is a shortest path from* x *to* y*.*

Another example of "optimal substructure": a sub-array of a sorted array is itself sorted.

Another property:

Claim 2. *If a graph has no negative edge-weight cycles that are reachable from the source-vertex* s*, then for all* $u \in V$ *that are rechable from* s*, there is a shortest path* $s \rightsquigarrow u$ *that is simple.*

All our single-source shortest paths algorithms maintain and finally output two things:

- For every $u \in V$, $d[u]$, a shortest-distance estimate.
  - We initialize each $d[u]$ to ∞, and $d[s]$ to 0.
  - We expect that when the algorithm halts, for every $u \in V$, $d[u] = \delta(s, u)$.
- For every $u \in V$, $\pi[u]$, the parent vertex in a shortest-paths tree.
  - When the algorithm halts, $\pi[u] =$ NIL if and only if: either (i) $u = s$, or, (ii) u is not reachable from s.

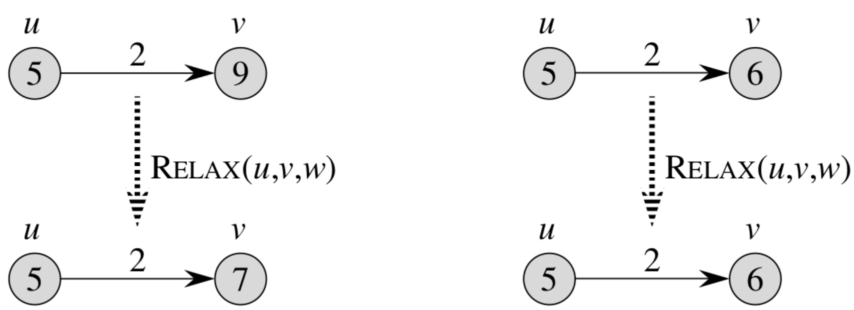
Two useful subroutines:

INITIALIZE-SINGLE-SOURCE(G, s)

for each vertex $v \in V[G]$
  do $d[v] \leftarrow \infty$
    $\pi[v] \leftarrow$ NIL
$d[s] \leftarrow 0$

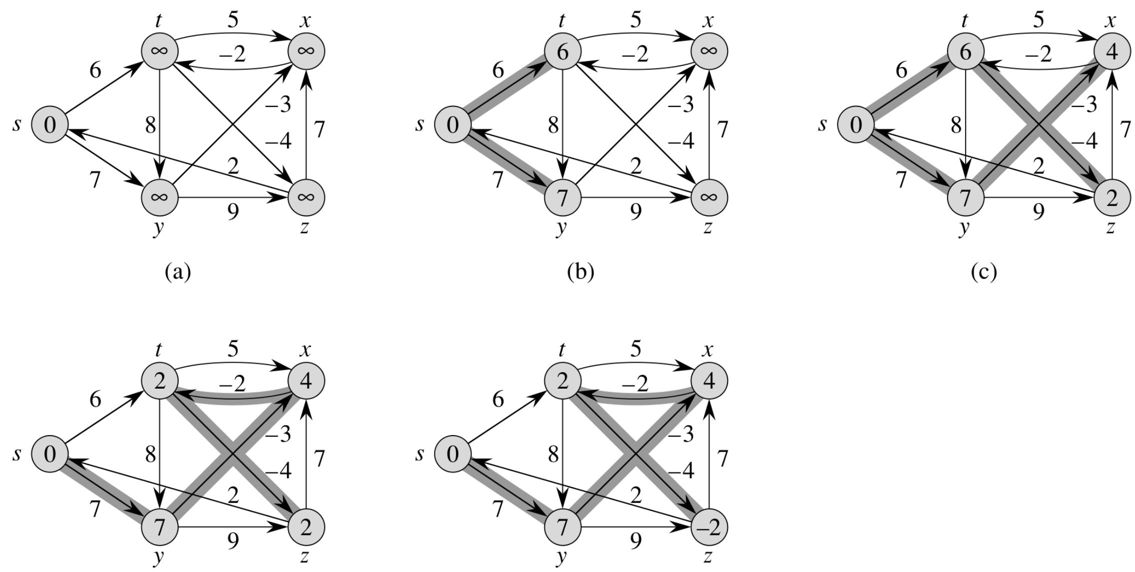
RELAX(u, v, w)

if $d[v] > d[u] + w(u, v)$
  then $d[v] \leftarrow d[u] + w(u, v)$
    $\pi[v] \leftarrow u$



Bellman-Ford(G, w, s)

```
Initialize-Single-Source(G, s)
for i ← 1 to |V[G]| − 1
     do for each edge (u, v) ∈ E[G]
          do Relax(u, v, w)
```



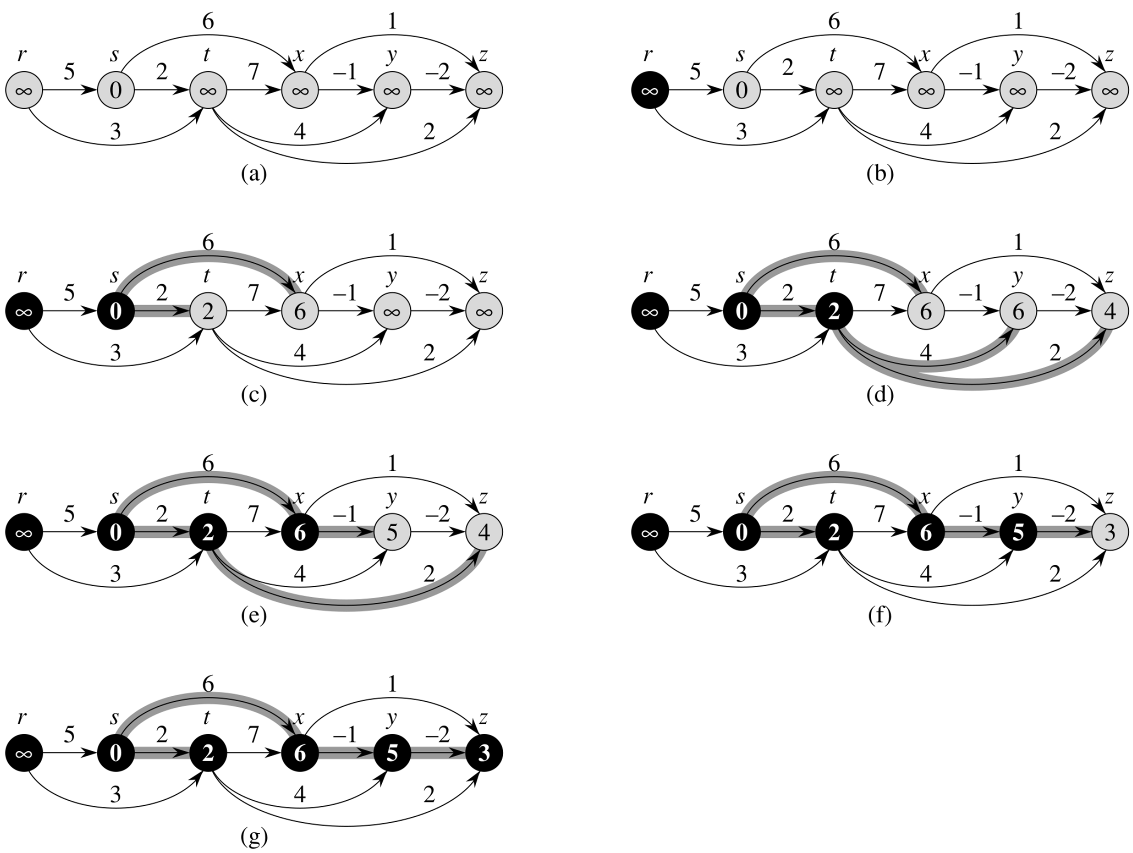
(a) (b) (c) (d) (e)

Order of relaxation happens to be, always:
$\langle t,x\rangle, \langle t,y\rangle, \langle t,z\rangle, \langle x,t\rangle, \langle y,x\rangle, \langle y,z\rangle, \langle z,x\rangle, \langle z,s\rangle, \langle s,t\rangle, \langle s,y\rangle$.

Correctness from: (i) every shortest path has $\leq |V| - 1$ edges, (ii) it suffices that we Relax every edge in a shortest path to every vertex in order once, and, (iii) redundant calls to Relax do no harm.

Time-efficiency: $\Theta(|V||E|)$.

```
DAG-SHORTEST-PATHS(G, w, s)
topologically sort the vertices of G
INITIALIZE-SINGLE-SOURCE(G, s)
for each vertex u, taken in topologically sorted order
    do for each vertex v ∈ Adj[u]
           do RELAX(u, v, w)
```



Time-efficiency: $\Theta(|V| + |E|)$.