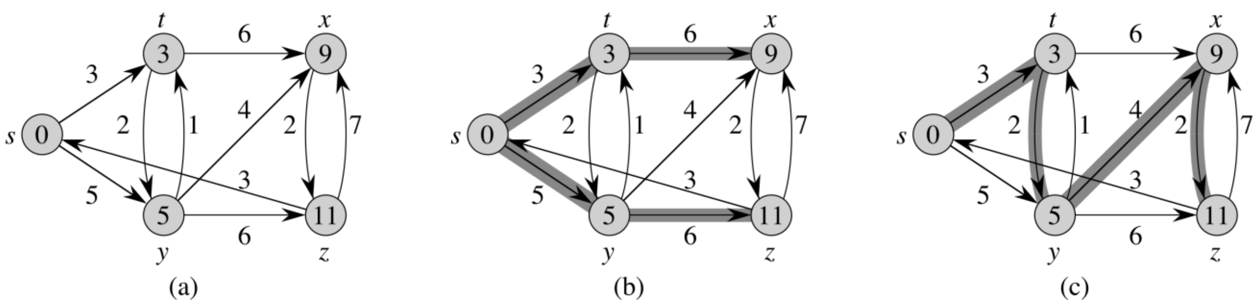
Chapter 24

Single-Source Shortest Paths

Let G be a weighted directed graph where the weight attached to each edge is the cost to traverse the edge.

A path from vertex u to vertex v is a sequence of one or more edges, $\{(v_1, v_2), (v_2, v_3), \ldots, (v_{r-1}, v_r)\}$, in G.E where $v_1 = u$ and $v_r = v$. The cost of the path is the sum of the weights of the edges in the sequence.

The shortest path weight from u to v is the minimum cost of all paths from u to v. If there is no path from u to v then the shortest path weight is ∞.



Shortest paths are not necessarily unique, and neither are shortest- path trees. The above figure shows a weighted, directed graph and two shortest-path trees with the same root.

Four different shortest path problems can be posed on the weighted directed graph, G.

1. From a given source vertex, s, find the shortest path weights to all other vertices in G.V.
2. To a given destination vertex, t, find the shortest path weights from all other vertices in G.V.
3. Given two vertices, u and v, find the shortest path from u to v.
4. For every pair of vertices, u and v, find the shortest path weight from u to v.

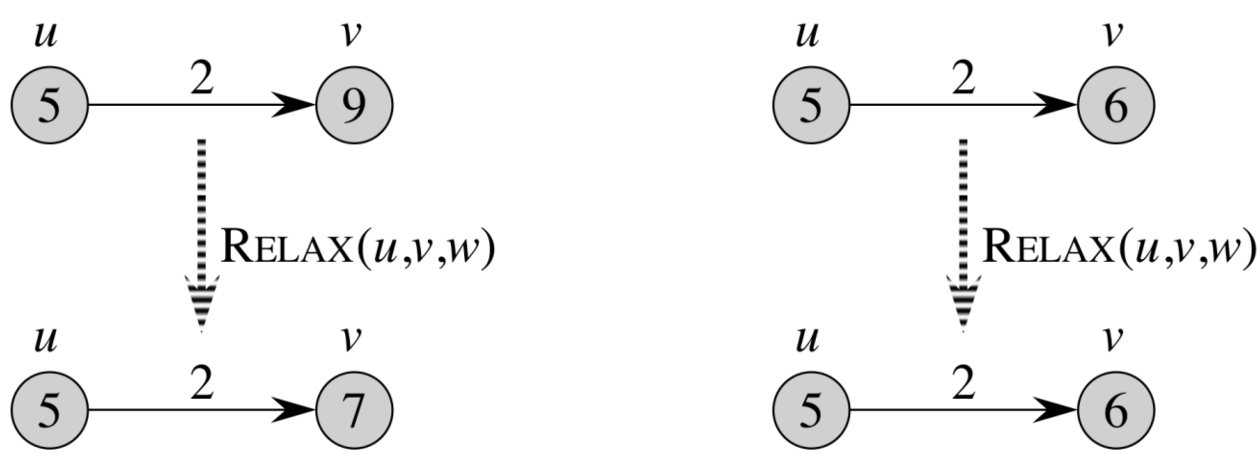
Negative weights: Suppose G has one or more edges with negative weights. Such an edge actually reduces the cost of any path using it. Suppose there's a cycle using edges whose total cost is negative. A path can run around the cycle many, many times and get any negative cost desired. If the cycle can be reached from the source vertex, s, and if the destination vertex, d, can be reached from the cycle then the shortest-path weight from s to d is $-\infty$.

INITILAIZE-SINGLE-SOURCE(G, s)

```
for each vertex v ∈ G.V
    v.d = ∞
    v.π = NIL
s.d = 0
```

RELAX(u, v, w)

```
if v.d > u.d + w(u, v)
    v.d = u.d + w(u, v)
    v.π = u
```



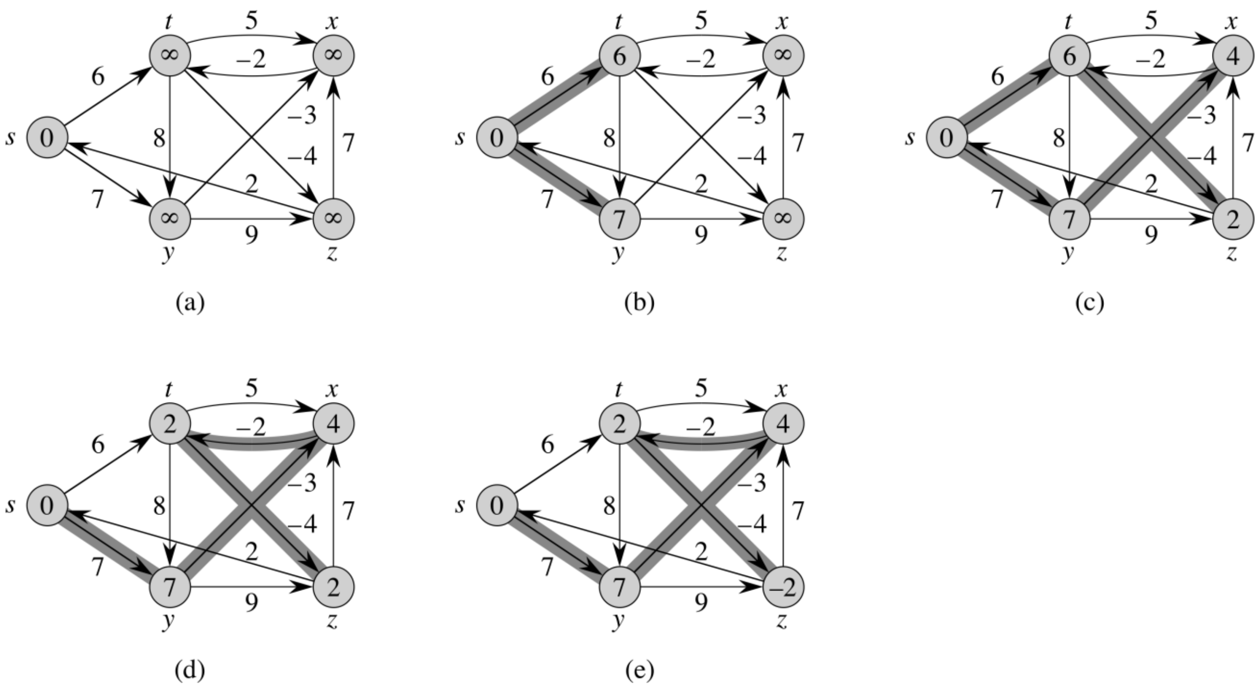
(a) (b)

The Bellman-Ford algorithm

- Allows negative-weight edges
- Computes $v.d$ and $v.\pi$ for all $v \in V$
- Returns TRUE if no negative-weight cycles reachable from s, FALSE otherwise

BELLMAN-FORD(G, w, s)

1 INITIALIZE-SINGLE-SOURCE(G, s)
2 **for** $i = 1$ **to** $|G.V| - 1$
3   **for** each edge $(u, v) \in G.E$
4   RELAX(u, v, w)
5 **for** each edge $(u, v) \in G.E$
6   **if** $v.d > u.d + w(u, v)$
7   **return** FALSE
8 **return** TRUE



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

The nested for loops relax all edges |V| - 1 times. The Bellman-Ford algorithm runs in time $\Theta(|V||E|)$, since the initialization in line 1 takes $\Theta(|V|)$ time, each of the |V| -1 passes over the edges in lines 2-4 takes $\Theta(|E|)$ time, and the for loop of lines 5-7 takes $\Theta(|E|)$ time.

Dijkstra's algorithm

Dijkstra's algorithm solves shortest-path problem 1 for directed weighted graph with non-negative edge weights. It is a greedy algorithm and similar to MST-Prim. Starting at the source vertex, s, it grows a tree, T, that eventually spans all vertices reachable from s. If v is a vertex in G.V then *v.d* holds the shortest-path weight from s to v. During the algorithm there are three classes of vertices: T-vertices that are in T; A-vertices not in T but adjacent to T-vertices; and X-vertices that are not adjacent to T-vertices.
Vertices are added to T in distance order; first s, then the vertex closest to s, then the next closest, etc. If u is a T-vertex then $u.\pi$ holds the tree-parent of u and *u.d* holds the shortest path weight from s to u.

If an A-vertex, v, is adjacent to one T-vertex, u, then $v.\pi = u$ and $.d = u.d + w(u,v)$. If an A-vertex is adjacent to multiple T-vertices, u_1, u_2, …, u_r, then

$v.d = \min(u_i\,.d + w(u_i, v))$ for $1 \le i \le r$

to select the T-vertex, u_k, that gives the shortest s-to-v path and $v.\pi = u_k$.

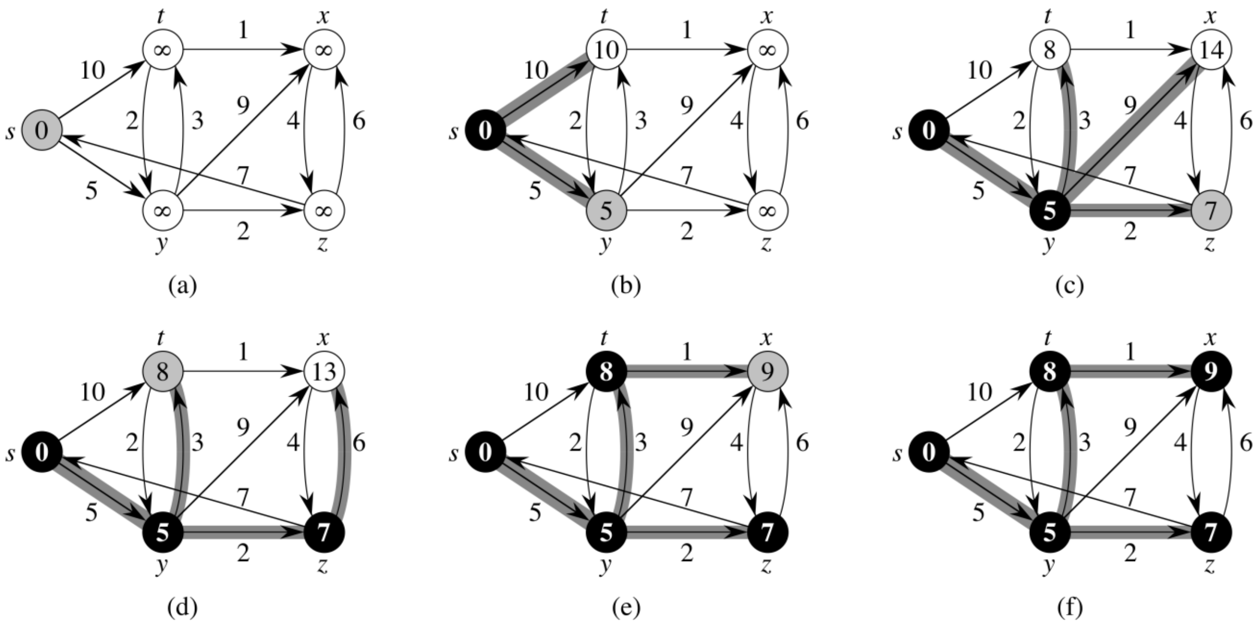
If *v* is an X-vertex then $v.\pi = NIL$ and $v.d = \infty$.

When a vertex, u, becomes a T-vertex, Adj[u] is scanned for A-vertices and X-vertices. If A-vertex, v, is adjacent to u then *v.d* is compared to *u.d* + *w*(*u, v*) to see if *v.d* and *v.π* need to be updated. If X-vertex, v, is adjacent to u then v is changed to an A-vertex, *v.π* = *u*, and *v.d* = *u.d* + *w*(*u, v*).

A-vertices and X-vertices are stored in a priority queue, Q, ordered by their d-variables with maximum priority given to the vertex with least distance, d. As with MST-PRIM , EXTRACT-MIN(Q) extracts the vertex with least distance.

DIJKSTRA(G, w, s)

```
INITIALIZE-SINGLE-SOURCE(G, s)
S = ∅
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    S = S U {u}
    for each vertex v ∈ G.Adj[u]
        RELAX(u, v, w)
```



Like MST-PRIM, DIJKSTRA runs in O(|E| lg |V|) time.