**CPSC 6109:** [**Advanced**](https://colstate.view.usg.edu/d2l/lp/ouHome/home.d2l?ou=1218642) **Algorithms**

**Spring 2018**

**Assignment #10**

**Student: Lu Lin**

**Due: 11:59 PM Monday May 7**

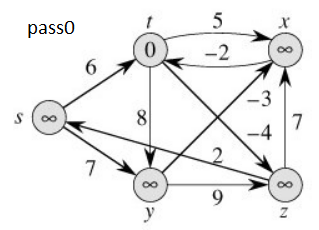
(1) Run the Bellman-Ford algorithm on the directed graph of Figure 24.4 on page 652.

a). using the vertex ***t*** as the source. In each pass, relax edges in the same order as in the figure, and show the **d** and **π** values after each pass.

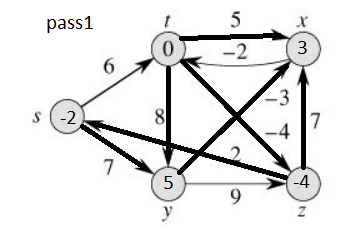
Solution:

In the given problem, each pass relaxes the edges in the same order as in Figure 24.4 in textbook, that is (t,x), (t,y), (t,z), (x,t), (y,x), (y,z), (z,x), (z,s), (s,t), (s,y).

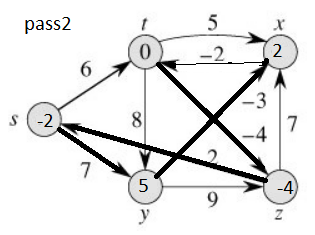
The source is vertex t and the following figure pass0 is the situation before the first pass over the edges:



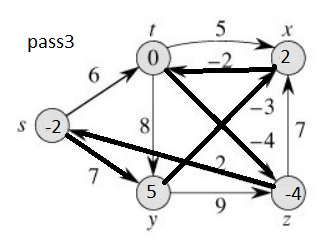
The following figure pass1 is the situation after the first successive pass, the d values are shown in vertices, and shaded edges indicate predecessor values. After the first pass, t.d = 0, x.d= 3, y.d = 5, z.d= -4, s.d= -2; and y.π = s, z.π = t, x.π =z, x.π =y, x.π =t, s.π = z, z.π =t.



The following figure pass2 is the situation after the second successive pass. The d values are shown in vertices, and shaded edges indicate predecessor values. After the pass, t.d = 0, x.d= 2, y.d = 5, z.d= -4, s.d= -2; and y.π = s, z.π = t, s.π = z, x.π =y.



The following figure pass3 is the situation after the third successive pass. The d values are shown in vertices, and shaded edges indicate predecessor values. After the pass, t.d = 0, x.d= 2, y.d = 5, z.d= -4, s.d= -2; and t.π = x, y.π = s, z.π = t, s.π = z, x.π =y.

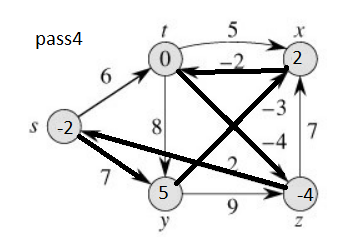


The following figure pass4 is the final pass situation after the fourth successive pass. The d values are shown in vertices, and shaded edges indicate predecessor values. After the pass, the final d and π are as follows:

t.d = 0, x.d= 2, y.d = 5, z.d= -4, s.d= -2;

t.π = x, y.π = s, z.π = t, s.π = z, x.π =y

The Bellman-Ford algorithm returns TRUE in this problem.

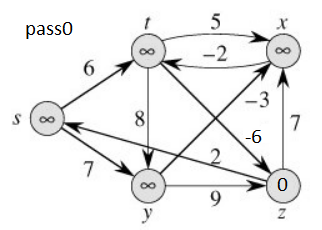


b). change the weight of the edge (***t, z***) to **-6** and run the algorithm again, using the vertex ***z*** as the source.

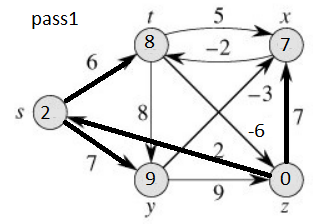
Solution:

In the given problem, each pass relaxes the edges in the same order as in Figure 24.4 in textbook, that is (t,x), (t,y), (t,z), (x,t), (y,x), (y,z), (z,x), (z,s), (s,t), (s,y).

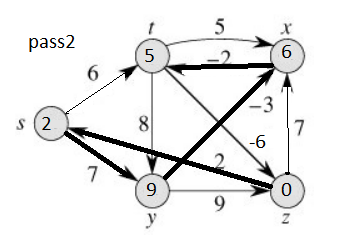
The source is vertex z and the following figure pass0 is the situation before the first pass over the edges (edge (***t, z***) = **-6**):



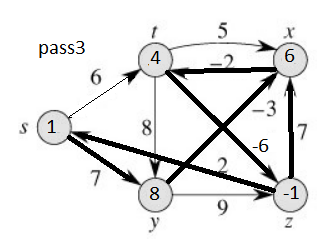
The following figure pass1 is the situation after the first successive pass, the d values are shown in vertices, and shaded edges indicate predecessor values. After the first pass, t.d = 8, x.d= 7, y.d = 9, z.d= 0, s.d= 2; and y.π = s, x.π =z, t.π =s, s.π = z.



The following figure pass2 is the situation after the second successive pass. The d values are shown in vertices, and shaded edges indicate predecessor values. After the pass, t.d = 5, x.d= 6, y.d = 9, z.d= 0, s.d= 2; and y.π = s, s.π = z, x.π = y, t.π = x.



The following figure pass3 is the situation after the third successive pass. The d values are shown in vertices, and shaded edges indicate predecessor values. After the pass, t.d = 4, x.d= 6, y.d = 8, z.d= -1, s.d= 1; and y.π = s, s.π = z, x.π = z, x.π = y, t.π = x, z.π = t.

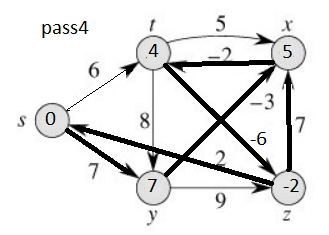


The following figure pass4 is the final pass situation after the fourth successive pass. The d values are shown in vertices, and shaded edges indicate predecessor values. After the pass, the final d and π are as follows:

t.d = 4, x.d= 5, y.d = 7, z.d= -2, s.d= 0;

y.π = s, s.π = z, x.π = z, x.π = y, t.π = x, z.π = t

The Bellman-Ford algorithm returns FALSE in this problem.



(2) Exercises 24.2-2 on page 657. Suppose we change line 3 of DAG-SHORTEST-PATHS to read

3 **for** the first |V| - 1 vertices, taken in topologically sorted order

Show that the procedure would remain correct.

Solution:

When we reach vertex v, the last vertex in the topological sort, it must have out-degree 0. Otherwise there would be an edge pointing from a later vertex to an earlier vertex in the ordering. Thus, the body of the for-loop of line 4 is never entered for this final vertex, so we may as well not consider it.