40.5 Bellman-Ford algorithm: Example 1

The example has been adopted from

www.cs.bilkent.edu.tr/~atat/502/SingleSourceSP.**ppt**

Let us consider the same example graph shown now in Figure 40.3 (a). It consists of vertex set V= {z, u, x, v, y}. In accordance with the first step all $d_{s,v}$ and $\pi_{s,v}$ are initialized to be infinity and NIL respectively (Figure 40.3.a). The next step is to identify the shortest path weight. First, it starts with vertex z (the source vertex). Using the step 2 of the algorithm, the shortest path from z to u is identified as the single edge (z,u). Hence $d_{z,u}$ is set to 6 and $\pi_{z,u}$ to NIL. Similarly $d_{z,x}$ is set to 7 and $\pi_{z,x}$ to NIL since these are the only paths from z to u and x respectively (Figure 40.3.b). Next to identify the shortest path between the vertex z and v, the two possible paths (z,u,v) and (z,x,v) are tested. The corresponding values and predecessors of these paths are used to find distances ($d_{z,v}$= 11 (6+5), $\pi_{z,v}$= u) and ($d_{z,v}$ = 4 (7-3), $\pi_{z,v}$= x) for the two paths respectively. Since the second path (z,x,v) has less weight it is chosen as the shortest path between z and v (Figure 40.3.c). Similarly the shortest path between z and y is identified to be (z,u,y) with $d_{z,y}$= 2, $\pi_{z,y}$= u (Figure 40.3.d). As the iteration goes on, a new shortest path from z to u is identified through z and v. Now $d_{z,u}$ is reset to 2, $\pi_{z,u}$= v (Figure 40.3.e). At this stage of the algorithm all the shortest paths are identified. Now the check for negative cycle is performed. There exist a shortest path (z,x,v,u,y) whose $d_{z,y}$ =-2 is less than the already computed $d_{z,y}$ =2 (Figure 40.3.f). This indicates that, there exist a negative cycle (u,v,y) in the graph.

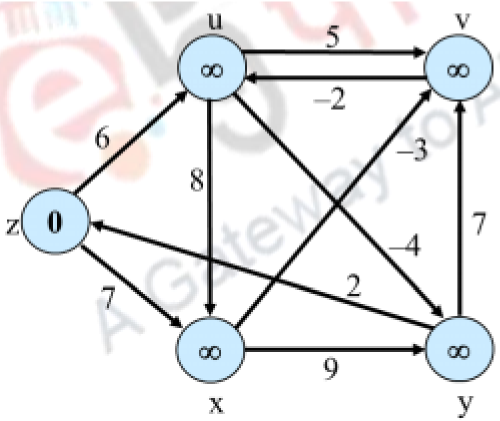

Figure 40.3.a

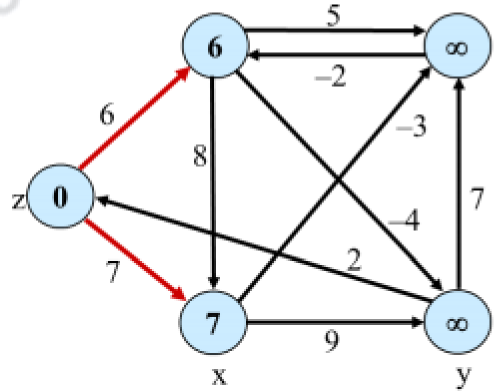

Figure 40.3.b

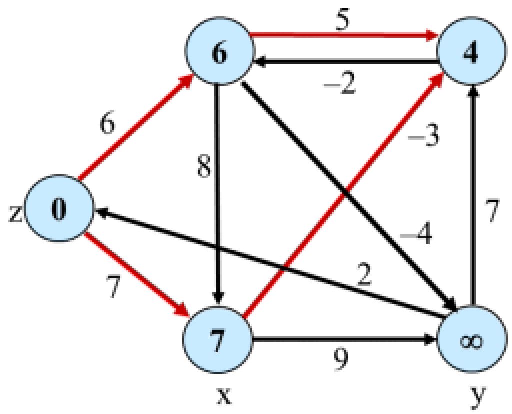

Figure 40.3.c

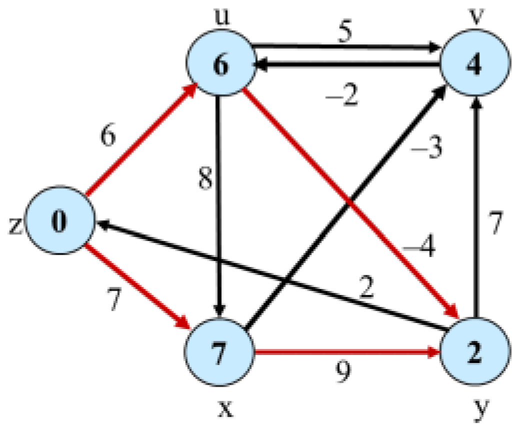

Figure 40.3.d

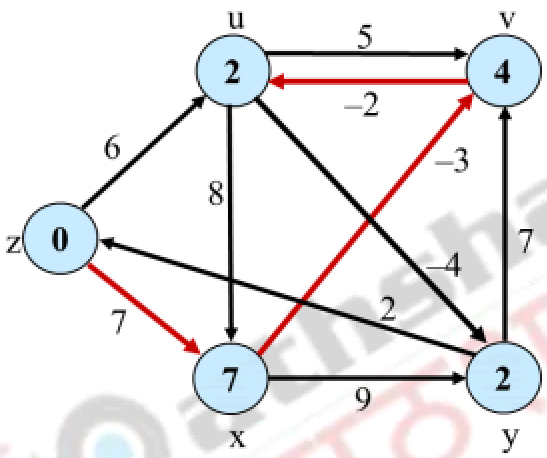

Figure 40.3.e

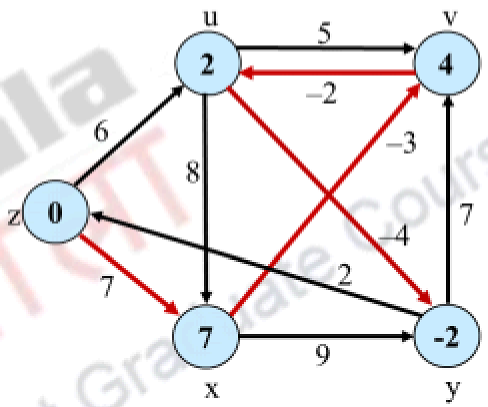

Figure 40.3 f

Figure 40.3 Illustration of Bellman-Ford algorithm

40.6 Bellman-Ford algorithm: Dynamic Programming

For each node v, find the length of the shortest path to any node t that uses at most 1 edge, or write down ∞ if there is no such path. If v = t we get 0; if (v, t) ∈ E then we get len(v, t); else just put down ∞. Now, suppose for all v we have solved for length of the shortest path to t that uses $i - 1$ or fewer edges.

We can use the above to solve for the shortest path that uses i or fewer edges. The shortest path from v to t that uses i or fewer edges will first go to some neighbor x of v, and then take the shortest path from x to t that uses $i-1$ or fewer edges, which we've already solved for. So, we just need to take the minimum over all neighbors x of v. At most $i = n - 1$ edges need to be processed to obtain the answer. The main observation made here are as follows. (i) If there is a negative cycle, then there is no solution. Because adding this cycle again can always produces a less weighted path. (ii) If there is no negative cycle, a shortest path has at most |V|-1 edges. The basic idea behind solving this algorithm using dynamic programming is that for all the paths have at most 0 edge, find all the shortest paths, for all the paths have at most 1 edge, find all the shortest paths and so on and finally for all the paths have at most |V|-1 edge, find all the shortest paths. The algorithm for the above is given below:

Bellman-Ford pseudocode:
initialize d[v][0] = infinity for v != t. d[t][i]=0 for all i.
for i=1 to n-1:
for each v != t:
d[v][i] = min
(v,x)2E
(len(v,x) + d[x][i-1])
For each v, output d[v][n-1].

40.6.1 Bellman-Ford algorithm: Example 2

Based on the above idea, the shortest path from node 1 to other nodes of Figure 40.4 is discovered. The total number of nodes is 3 and hence the process repeats 3 times. First the shortest paths with 0-edge are discovered from vertex 1 to 1, 1 to 2 and 1 to 3. The path weights for 1 to 1, 1 to 2 and 1 to 3 are 0, ∞ and ∞ respectively (Figure 40.4).

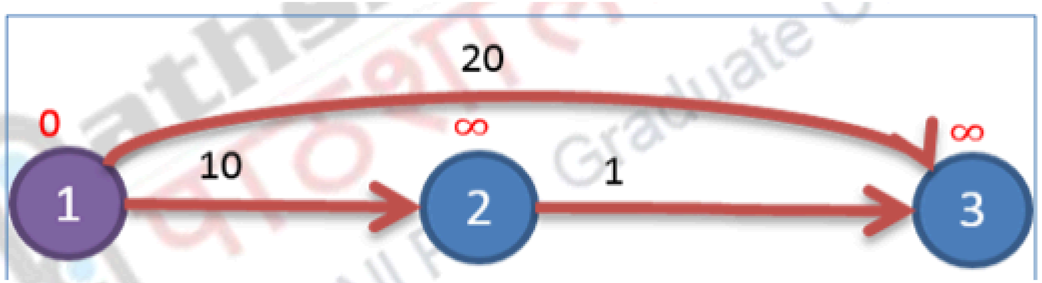

Figure 40.4 Shortest paths with 0-edge

Next, the shortest paths with 1-edge are discovered from vertex 1 to 1, 1 to 2 and 1 to 3. The path weights for 1 to 1, 1 to 2 and 1 to 3 are 0, 10 and 20 respectively (Figure 40.5).

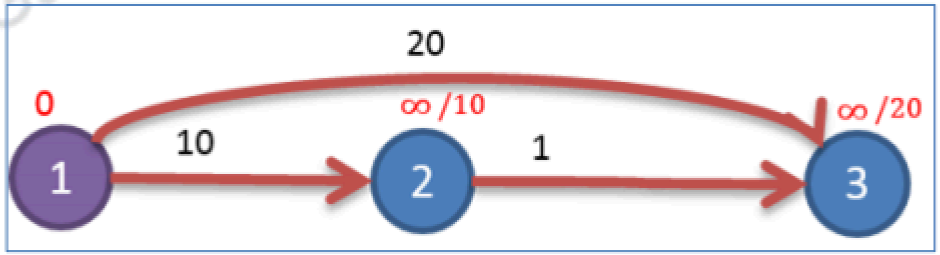

Figure 40.5 Shortest paths with 1-edge

Finally, the shortest paths with 2-edges are discovered from vertex 1 to 1, 1 to 2 and 1 to 3. The path weights for 1 to 1, 1 to 2 and 1 to 3 are 0, 10 and 11 respectively. The path from 1 to 3 changes from 20 to 10 since the 2-edge path is shorter than the 1-edge path (Figure 40.6).

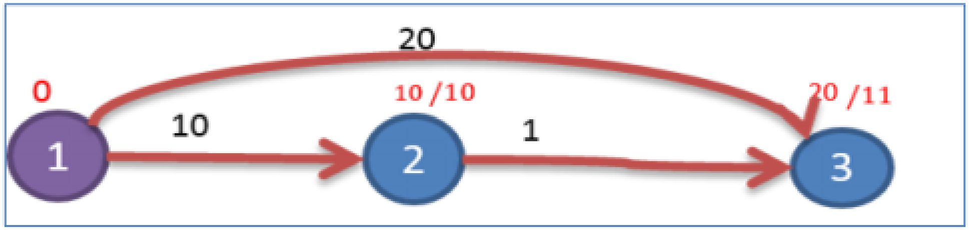

Figure 40.6 Shortest paths with 2-edges

40.7 Bellman-Ford algorithm- Walkthrough

Let us consider one more example for the graph given in Figure 40.7 (a). The same procedure mentioned above is repeated to find the shortest path from vertex 1.

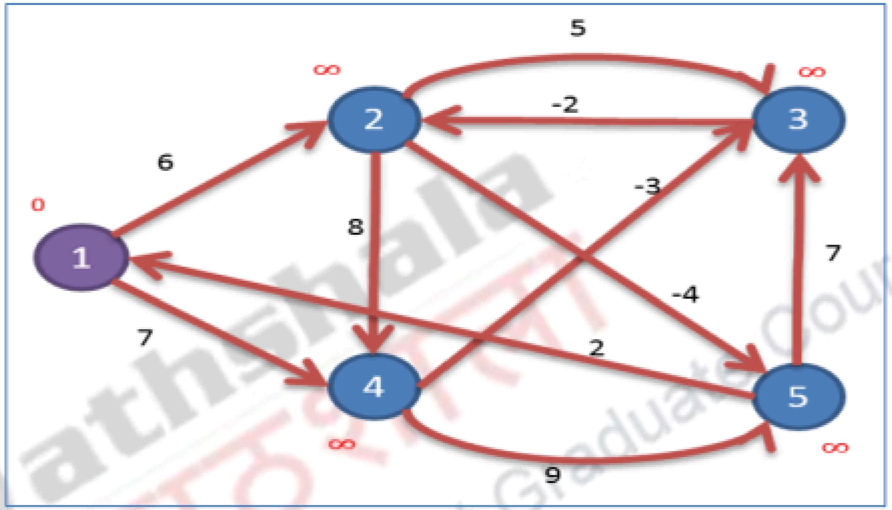

Figure 40.7 (a) Shortest paths with 0-edge

Now we find the shortest path from 1 with 1 edge which is 1-2 and 1-4. Hence the d value of 2 changes from ∞ to 6 and that of 4 changes from ∞ to 7 (Figure 40.7 (b)).

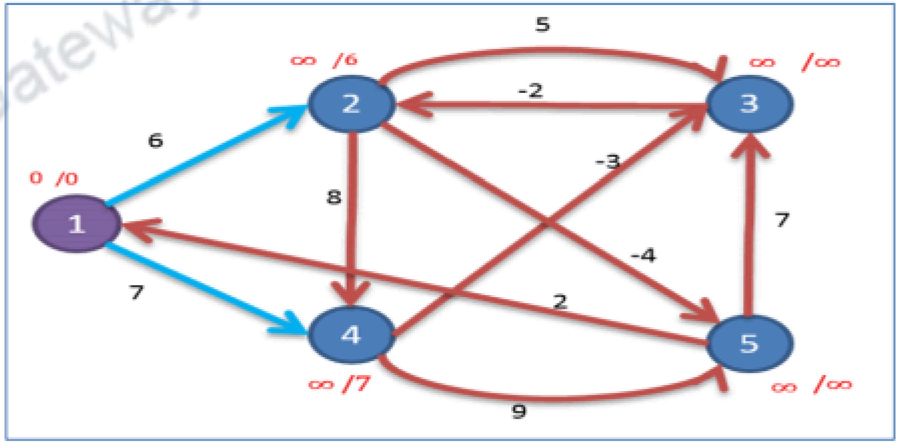

Figure 40.7 (b) Shortest paths with 1-edge

Now we find the shortest path from 1 with 1 edge which is 1-2 and 1-4. Hence the d value of 2 changes from ¥ to 6 and that of 4 changes from ¥ to 7 (Figure 40.7 (b)).

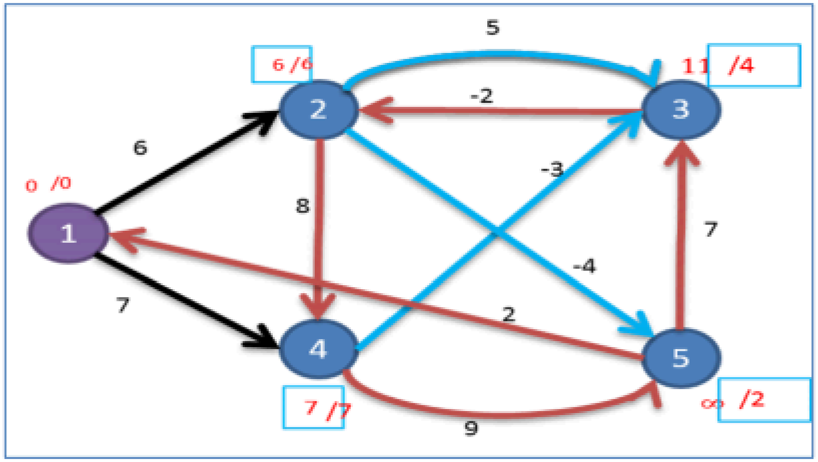

Figure 40.7 (c) Shortest paths with 2-edges

Now we find the shortest path from 1 with 3 edges such as 1-4-3-2(d =2) Hence the d value of 2 is 6/2. (Figure 40.7 (d)).

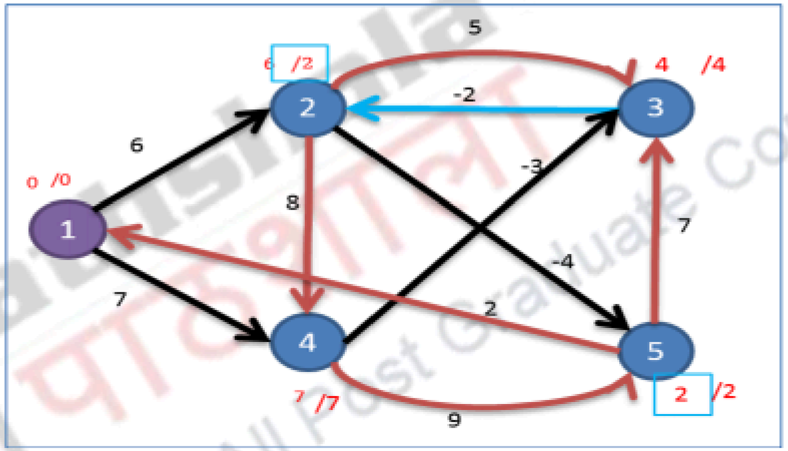

Figure 40.7(d) Shortest paths with 3-edges

Now we find the shortest path from 1 with 2 edges which is 1-2-3 (d =11) and 1-4-3 (d=4). Hence the d value of 3 is 11/4. The 2 edge path to 5 is 1-2-5 and so d value of 5 is ¥/2 (Figure 40.7 (c)).

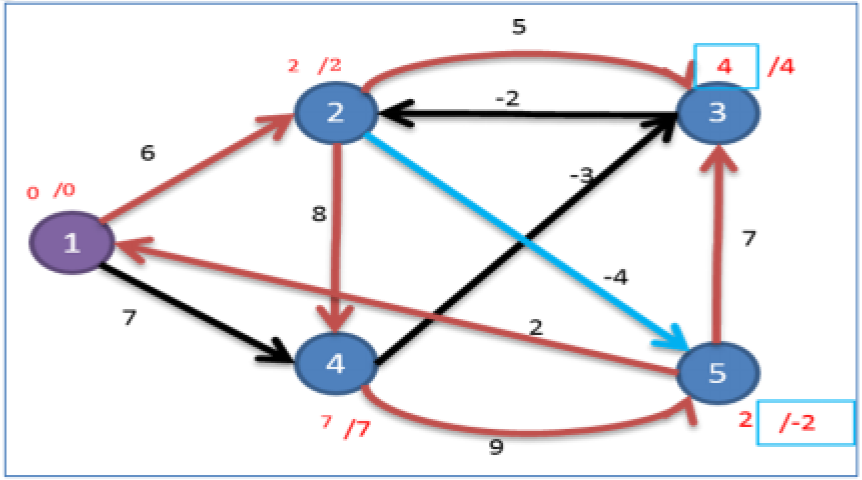

Figure 40.7 (e) Shortest paths with 4-edges

The shortest paths with negative cycle is shown in Figure 40.7 (f). The shortest path from 1 to 5 is 1-4-3-2-5 and its weight is -2.

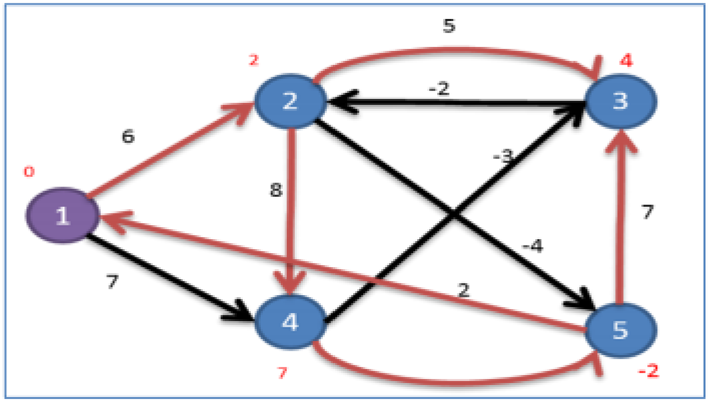

Figure 40.7 (f) Shortest paths with negative cycle