Bellman Ford Algorithm

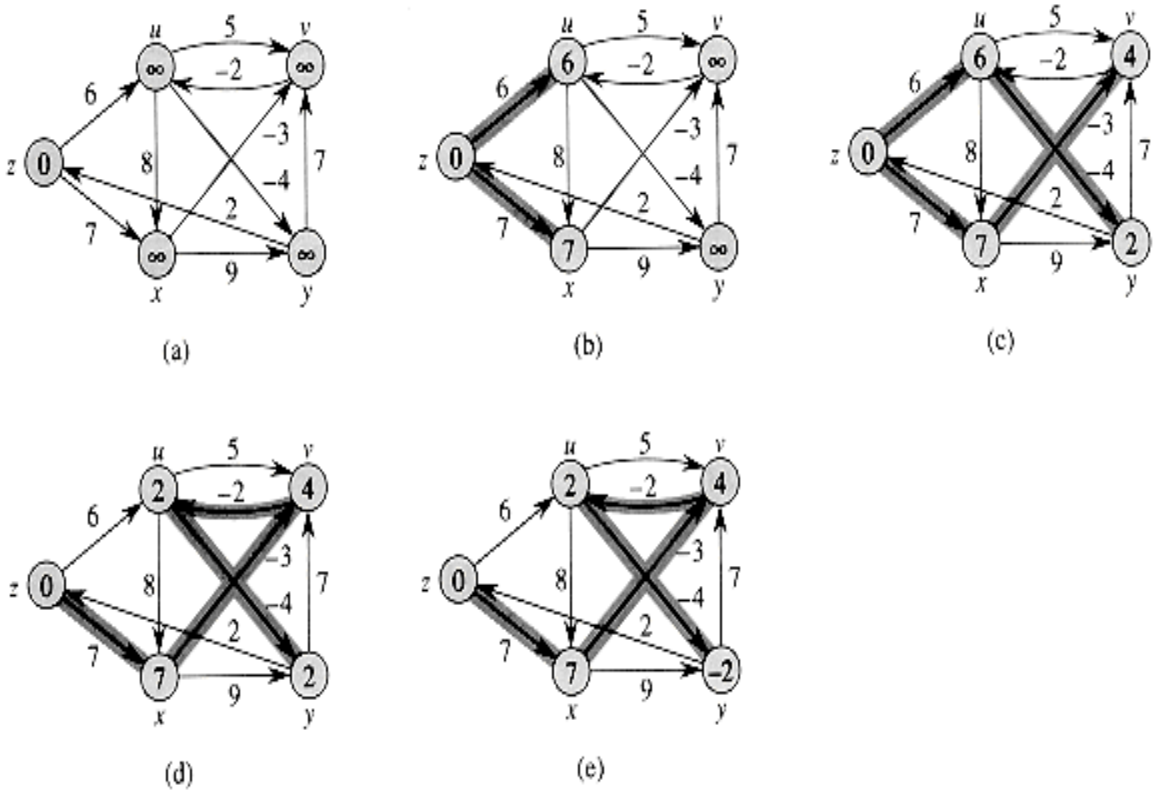
Bellman-Ford algorithm solves the single-source shortest-path problem in the general case in which edges of a given digraph can have negative weight as long as G contains no negative cycles.

This algorithm, like Dijkstra's algorithm uses the notion of edge relaxation but does not use with greedy method. Again, it uses d[u] as an upper bound on the distance d[u, v] from u to v.

The algorithm progressively decreases an estimate d[v] on the weight of the shortest path from the source vertex s to each vertex v in V until it achieve the actual shortest-path. The algorithm returns Boolean TRUE if the given digraph contains no negative cycles that are reachable from source vertex s otherwise it returns Boolean FALSE.

```
  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)
for each edge (u, v) ∈ E[G]
do if d[v] > d[u] + w(u, v)
then return FALSE
return TRUE
}
```

Below figure shows the execution of the Bellman-Ford algorithm on a graph with 5 vertices. After initializing the d and π values of all vertices in line 1, the algorithm makes |V| - 1 passes over the edges of the graph. Each pass is one iteration of the for loop of lines 2-4 and consists of relaxing each edge of the graph once. Figures (b)-(e) show the state of the algorithm after each of the four passes over the edges. After making |V|- 1 passes, lines 5-8 check for a negative-weight cycle and return the appropriate boolean value.



Execution of the Bellman-Ford algorithm: The source is vertex s. The d values are shown within the vertices, and shaded edges indicate predecessor values: if edge (u, v) is shaded, then π[v] = u. In this particular example, each pass relaxes the edges in the order (t, x), (t, y), (t, z), (x, t), (y, x), (y, z), (z, x), (z, s), (s, t), (s, y). (a) The situation just before the first pass over the edges. (b)-(e) The situation after each successive pass over the edges. The d and π values in part (e) are the final values. The Bellman-Ford algorithm returns TRUE in this example.

Analysis

• The initialization in line 1 takes (v) time

• For loop of lines 2-4 takes O(E) time and For-loop of line 5-7 takes O(E) time.

Thus, the Bellman-Ford algorithm runs in O(E) time.

Source:

http://www.learnalgorithms.in/#