**SSSP Algorithm for a DAG using Topological Sort**

The following algorithm computes the source shortest paths from a given source to all the other nodes for a given directed acyclic graph. The algorithm assumes that a topological sort algorithm already exists.

**SSSP\_DAG(G,s)**

1. Initialize the distance array d with source distance as zero and others as infinite.
2. Initialize the parent array p with parent of all the vertices initially set to NULL.
3. Apply the topological sort algorithm to graph G, to obtain a topological ordering of the vertices.
4. **For** **each** vertex u in V[G] taken in topological order

**Do**

**For each** vertex v adjacent to u

**Do If** d[u] is not infinity

**Then** RELAX(u,v,w)

The operation RELAX above can be defined as below:

**RELAX(u,v,w)**

{

**If** ( d[v] > d[u] + w(u,v) )

**Then**

d[v] = d[u] + w(u,v)

p[v] = u

}

The final shortest paths distances from the source to other vertices can be found in the array d.

The shortest path tree can be traced back using the parent array p.

**Running Time Analysis**

Step 1 and 2 take O(V) time. Step 3 takes O(V+E) time.

Since we do not have to find the minimum in the distance array (after each step as we do in Dijkstra) we do not need to implement the distance array as a heap which makes the RELAX operation run in O(1) time.

Considering this, the step 4 in the above algorithm will run in O(V+E) time. **Thus we get an overall time of O(V+E)**.

**Proof for correctness of the above algorithm**

**Claim:** Right after the j-th iteration of the outer for loop in step 4, the vertex j+1 as occurring in the topological order, the distance d[j+1] = Δ ( s,uj+1) where Δ is the weight of the shortest path from s to uj+1.

Lets prove the correctness of the above claim through induction.

**Base:** All the vertices v before the source vertex in the topological order will have d[v] as infinity and thus have no path from source to them which is indeed true as the graph is acyclic.

As and when the source is encountered in the topological order the claim holds true as d[s] is zero.

**Hypothesis:** Lets assume that the claim is true for all vertices um such that m < |V| - 1 and thus um+1 will have the correct shortest path distance from source s at the start of the next iteration of the outer for loop.

**Proof**

In the (m+1)thiteration of the loop RELAX(um+1, v,w) is called for every v ∈ Adj[um+1]. Also the vertices v’s will lie ahead of um+1 in the topological ordering of the graph.

Now if um+1 is not reachable from s, then v is also not reachable through um+1.

If um+1 is not reachable for s then v also is not reachable from s through um+1 and the value of v is not changed.

If um+1 is reachable from s, then v is also reachable from s. If d[v] > d[um+1]+ w(um+1,v), then d[v] is updated with the shorter value.

Now however if v = um+2, then d[v] = Δ (s,v) because there is no other node in {um+3, um+4, … ,un } from which um+2 can be reached since graph will never have any back edges.

Hence our claim is true for (m+1)th iteration and is thus true generally.