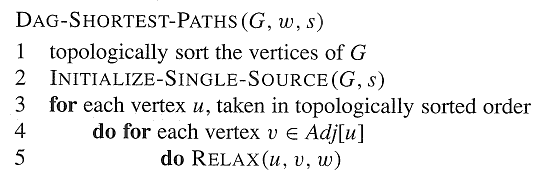
**CS430 Lecture 25 Activities**

Opening Questions

1. We saw the Bellman-Ford algorithm found the shortest path from a source to all other vertices by “brute force” relaxing every edge in the graph in a fixed order |v|-1 times. Why did it need to do this |v|-1 times? And with this in mind could we improve on the Bellman-Ford for certain graphs?

DAG Shortest Path Algorithm

By relaxing the edges of a weighted DAG (directed acyclic graph) *G* = (*V*, *E*) in topological sort order of its vertices, we can compute shortest paths from a single source. Shortest paths are always well defined in a DAG, since even if there are negative-weight edges, no negative-weight cycles can exist.



1. Here is the topological sort on a DAG. Find the shortest path from s to every other vertex.



2. What is the runtime for DAG Shortest Path?

3. Discuss why DAG Shortest Path is correct.

4. If we restrict the graph to having no negative edges, given a source s, what is the shortest path from s to one of its adjacent vertexes?

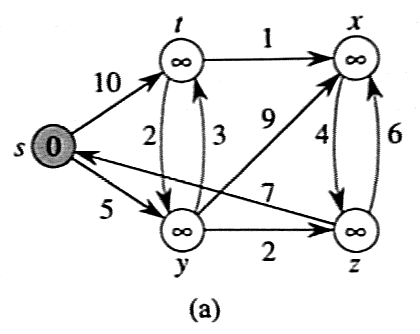
Dijkstra’s Shortest Path Algorithm

* No negative-weight *edges*.
* Essentially a weighted version of breadth-first search.
  + Instead of a FIFO queue, uses a priority queue.
  + Keys are shortest-path weight estimates (*d*[*v*]).
* Have two sets of vertices:
  + *S* = vertices whose final shortest-path weights are determined,
  + *Q* = priority queue = *V-S*.
* Dijkstra’s algorithm can be viewed as greedy, since it always chooses the “lightest” (“closest”) vertex in V-S to add to S.

DIJKSTRA*(V, E, w, s)*

INIT-SINGLE-SOURCE*(V, s)  
 S* ← empty set  
 *Q* ← *V //* i.e., insert all vertices into *Q by “d” values* **while** *Q not empty  
 u* ← EXTRACT-MIN*(Q)  
 S* ← *S* ***∪*** {*u*}   
 **for** each vertex *v ∈*  *Adj*[*u*]  
 RELAX*(u,v,w)* // possibly updates a short path estimate “d” value and moves the vertex forward in the queue

5. Here is a graph with no negative edges. Find the shortest path from s to every other vertex using Dijkstra’s algorithm.



6. What is the runtime for Dijkstra’s algorithm?

7. Prove the greedy choice in Dijkstra’s algorithm (pick the vertex with the smallest shortest path estimate, not including the vertices we are done with) leads to an optimal solution.

Dijkstra's Algorithms <http://oopweb.com/Algorithms/Documents/AnimatedAlgorithms/VolumeFrames.html>