**public** DijkstraSP(EdgeWeightedDigraph G, **int** s) { Frequency Cost

distTo = **new** **double**[G.V()]; \_\_\_ \_\_\_

edgeTo = **new** DirectedEdge[G.V()]; \_\_\_ \_\_\_

**for** (**int** v = 0; v < G.V(); v++) \_\_\_ \_\_\_

distTo[v] = Double.***POSITIVE\_INFINITY***; \_\_\_ \_\_\_

distTo[s] = 0.0; \_\_\_ \_\_\_

// relax vertices in order of distance from s

pq = **new** IndexMinPQ<Double>(G.V()); \_\_\_ \_\_\_

pq.insert(s, distTo[s]); \_\_\_ \_\_\_

**for** (**int** v = 0; v < G.V(); v++) { \_\_\_ \_\_\_

**if** (v != s) { \_\_\_ \_\_\_

pq.insert(v, Double.***POSITIVE\_INFINITY***); \_\_\_ \_\_\_

}

}

**while** (!pq.isEmpty()) { \_\_\_ \_\_\_

**int** v = pq.delMin(); \_\_\_ \_\_\_

**for** (DirectedEdge e : G.adj(v)) \_\_\_ \_\_\_

relax(e); \_\_\_ \_\_\_

}

}

// relax edge e and update pq if changed

**void** relax(DirectedEdge e) {

**int** v = e.from(); \_\_\_ \_\_\_

**int** w = e.to(); \_\_\_ \_\_\_

**if** (distTo[w] > distTo[v] + e.weight()) { \_\_\_ \_\_\_

distTo[w] = distTo[v] + e.weight(); \_\_\_ \_\_\_

edgeTo[w] = e; \_\_\_ \_\_\_

pq.decreaseKey(w, distTo[w]); \_\_\_ \_\_\_

}

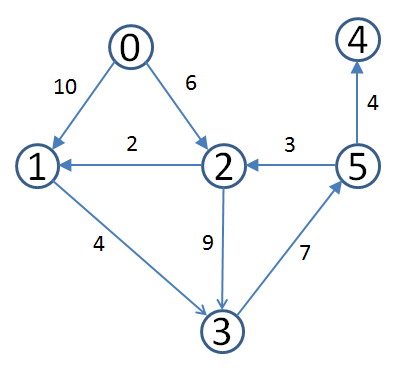
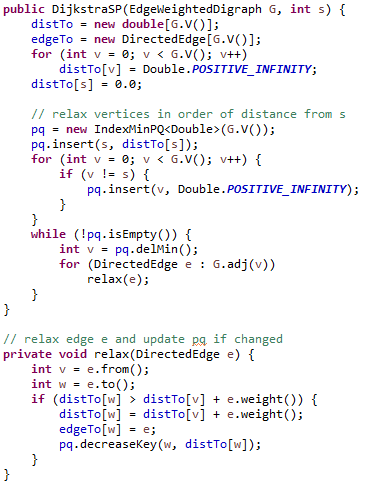
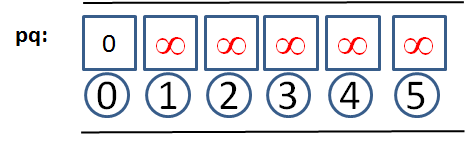
}

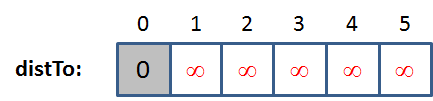
set dist[v] to INFINITY for all v in G \_\_\_ \_\_\_  
set pred[v] to -1 for all v in G \_\_\_ \_\_\_  
set visited[v] to false for all v in G \_\_\_ \_\_\_

**while** (some unvisited vertex v has dist[v] < INFINITY do) \_\_\_ \_\_\_  
 determine u whose dist[u] is smallest of unvisited vertices \_\_\_ \_\_\_

**if** (dist[u] = INFINITY) return \_\_\_ \_\_\_  
 visited[u] = true \_\_\_ \_\_\_  
 **foreach** neighbor v of u do: \_\_\_ \_\_\_  
 w = weight of edge (u,v) \_\_\_ \_\_\_  
 newLen = dist[u] + w \_\_\_ \_\_\_  
 **if** (newLen < dist[v]) \_\_\_ \_\_\_  
 dist[v] = newLen \_\_\_ \_\_\_  
 pred[v] = u \_\_\_ \_\_\_

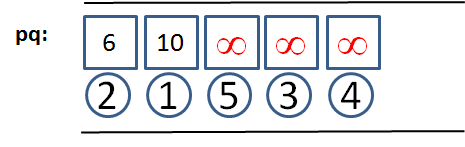
}

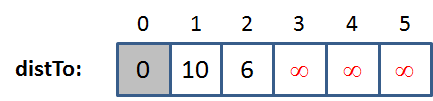
 Starting Priority Queue. Yes it is a  
 Heap but you should represent as  
 a Min priority queue

 Distances are precomputed also

Operates by removing **min vertex**  
 from the priority queue (0 to start).

There are two opportunities to relax  
 an edge: (0,1) and (0,2). Once done,   
 state is as follows:





You should be able to continue this  
 execution until there are no more  
 edges to relax.