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

**for** (DirectedEdge e : G.edges()) {

**if** (e.weight() < 0)

**throw** **new** IllegalArgumentException("edge " + e + " has negative weight");

}

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();

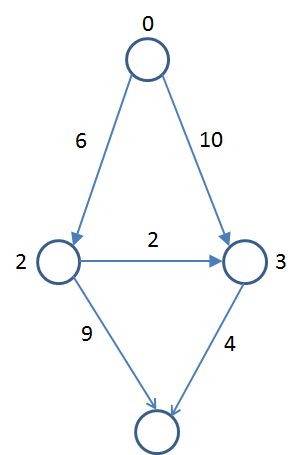
**int** w = e.to();

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

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

edgeTo[w] = e;

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

 }

}