**function** BellmanFord(*list* vertices, *list* edges, *vertex* source)

::distance[],predecessor[]

*// This implementation takes in a graph, represented as*

*// lists of vertices and edges, and fills two arrays*

*// (distance and predecessor) about the shortest path*

*// from the source to each vertex*

*// Step 1: initialize graph*

**for each** vertex v **in** vertices:

distance[v] := **inf** // Initialize the distance to all vertices to infinity

predecessor[v] := **null** // And having a null predecessor

distance[source] := 0 // Except the source vertex, which is zero

*// Step 2: relax edges repeatedly*

**for** i **from** 1 **to** size(vertices)-1:

**for each** edge (u, v) **with** weight w **in** edges:

**if** distance[u] + w < distance[v]:

distance[v] := distance[u] + w

predecessor[v] := u

*// Step 3: check for negative-weight cycles*

**for each** edge (u, v) **with** weight w **in** edges:

**if** distance[u] + w < distance[v]:

**error** "Graph contains a negative-weight cycle"

**return** distance[], predecessor[]