Stephen Woodbury 5/22/17 CMPS 102
I have read and agree to the collaboration policy. Stephen Woodbury.
Collaborators: none

Assignment 3_4 : Graphs :

a) Increasing Edge Weights in Path Algorithm Design:

Assumptions:

1) Graph is connected.
2)

Algorithm Components:

nodesLeadingToNode: Array of arrays of nodes leading to a spec. node.
ShortestPathLength: array of size n, for every index, init. to inf., gives the known shortest Path Length from source to vertex.
NodesLeadToByNode: array of arrays of nodes lead to by specific node.
nodesToExplore: Queue of nodes to explore, init to all, ensure source on front

Algorithm:

```
1 SetUp nodesLeadingToNode, init shortestPathLength[source] = 0
2
3 findShortestPath(source)
4
5 findShortestPath(j)
6   for(all members m of nodesLeadToByNode[j])
7     if(shortestPathLength[m] > w(j, m) + shortestPathLength[j])
8       shortestPathLength[m] = w(j, m) + shortestPathLength[j]
9     nodesToExplore.dequeue()
10    if(m exists in nodesToExplore)
11      findShortestPath(m)
12return shortestPathLength
```

Algorithm Description

Setup necessary data structures. Start finding the shortest path from the source. From source, analyze all nodes source leads to. Determine in weight of edge from source to nodes + min weight path's weight of source to source is less than the current min weight path's weight of source to node. If it is, replace the min path weight of node with the min weight path weight of source from source + w(source, node). Call *findShortestPath* on said node.

Proof of Correctness : Termination

Claim: Our algorithm terminates
Pf: every time *findShortestPath* is called, a new node is explored and dequeued from nodes left to explore meaning a node already explored won't be explored again. There are only n nodes to explore, therefore, *findShortestPath* will end.

Proof of Correctness : Shortest Path Found

claim: Our algorithm yields the shortest path from source to every vertex
Pf: every possible path possibility is explored, and if the path currently being explored has a weight that is less than the current weight of the smallest path of an index, that weight is replaced.

RunTime Analysis: ***O(v^2)*** Couldn't find wanted run time.
Space Complexity: ***O(e)***

b) Same as a.