# Assignment 3 - Simple Routing Calculation

In the lectures, you learned about Distance Vector routing protocols, one of the two classes or routing protocols. DV protocols, such as RIP, use a fully distributed algorithm that finds shortest paths by solving the Bellman-Ford equation at each node. In this assignment, you will develop a distributed Bellman-Ford algorithm and use it to calculate routing paths in a network. This assignment is different from the previous assignments in that we're not running using the Mininet environment. Rather, we will be simulating a small network running a Bellman-Ford algorithm.

You should review some materials on Bellman-Ford. Some resources follow:

- Wikipedia
- YouTube

Before getting started, you will need to update your CS6250 repository using the following commands from the 2015-Summer-OMS6250 directory:

```
git commit -a -m "Saving work"
git pull --rebase
```

In the assignment-3 directory, you can see quite a few files. They're described below:

- Node.py - This is where all your work will be. It defines how a topology is run, along with what Nodes are can can do.
- run_topo.py - This will actually run the topology file. It's mostly a wrapper for Node.py.
- helpers.py - This contains logging functions, very similar to Assignment 2's.
- topoX.py - These are valid topology files.
- badtopo.py - This is an invalid topology file.

There are a few TODOs in Node.py that you will have to complete. First, you will need to decide on your data structure that you're using to keep track of path weights (i.e., your distance vector). Second, you will need to implement the Bellman-Ford algorithm. (Assume all links have a weight of 1.) Third, you will need to write a logging function.

For this assignment, you will be able to create as many topologies as you wish and share them on Piazza. We encourage that you do share new topologies, and, unlike Assignment 2, you can share your log outputs. If there's a problem with a created topology, you will get an error back when you try to run it. We've included three good and one bad topology so that you can see what will happen with an invalid topology.

To run, it's a matter of using the `run_topo.py` script:

```
python run_topo.py topo1 topo1.log
```

This will run topo1 and log to topo1.log. Note that you should *not* include the .py from the topology (so topo1.py should only be topo1 as in this example).

## Rubric

We will be testing based on various topologies with different complications: with loops, without loops, different numbers of nodes, etc. Being able to handle a variety of non-looping topologies and create a correct distance vector will result in 60% credit. Handling looping topologies will get up to 100% credit. Zero credit will be given if logging is incorrect. (Incorrectly formatted logs will not match

properly and fail the autograder, and we will not be manually inspecting the logs due to the number of students in the class.)

This page was last edited on 2015/06/05 23:15:51.