Part 3: BGP Analysis (30 *points*)

Your goal is to analyze BGP Routing Tables.

Some background information:

Up until this point, we have looked at top traffic flows at the level of IP addresses and ports, but a network operator might also be interested in exploring which other networks (i.e., autonomous systems) are responsible for sending or receiving traffic to the network. From our previous lectures on peering and Internet business relationships, it should be clear why an operator might care about knowing which ASes are sending traffic its way! This information may also be useful for exploring various kinds of network attacks (e.g., sources of denial of service attacks), which we will explore in the next assignment.

The RouteViews project allows network operators to obtain real-time information about the global routing system from the perspectives of several different autonomous systems around the Internet. RouteViews servers act as software BGP routers, obtaining their BGP routing information via BGP sessions, just like any other router would learn BGP routes. The main difference between the RouteViews servers and other BGP-speaking routers is that the RouteViews servers do not forward any real Internet traffic.

RouteViews periodically logs BGP routing tables (sometimes called a Routing Information Base, or a "RIB") in a binary format called MRT. We collected data from one such server and used the *bgpdump* tool to parse the data into a more parsable output format. The entries in the BGP RIB table look like the ones shown below:

```
TIME: 03/07/16 02:00:00

TYPE: TABLE_DUMP_V2/IPV4_UNICAST

PREFIX: 0.0.0.0/0

SEQUENCE: 0

FROM: 185.44.116.1 AS47872

ORIGINATED: 03/06/16 20:27:05

ORIGIN: IGP

ASPATH: 47872 3356

NEXT_HOP: 185.44.116.1
```

COMMUNITY: 3356:2 3356:514 3356:2087 47872:1 47872:3356

TIME: 03/07/16 02:00:00

TYPE: TABLE_DUMP_V2/IPV4_UNICAST

PREFIX: 0.0.0.0/0

SEQUENCE: 0

FROM: 80.241.176.31 AS20771

ORIGINATED: 03/04/16 10:21:21

ORIGIN: IGP

ASPATH: 20771 1299

NEXT_HOP: 80.241.176.31

BGP RIBs might have multiple entries for an IP prefix.

For this assignment, we considered only a single entry for an IP prefix. We translated this data into the bgp_rib.csv file. Each contains the following fields:

TIME, ORIGIN, FROM, SEQUENCE, ASPATH, PREFIX, NEXT_HOP

The following code imports bgp_rib.csv file into a list of dicts.

```
import csv

with open('bgp_rib.csv', 'r') as bgp_file:
    bgp_reader = csv.DictReader(bgp_file,delimiter=";")
    bgp_data = list(bgp_reader)

print "Number of BGP RIBs: {}".format(len(bgp_data))
print
print "Sample BGP RIB: {}".format(bgp_data[0])
```

What is the longest route（by number of unique ASes）in the BGP RouteViews data？

To answer this question, you will need to sort bgp_data by the number of unique AS numbers in the "ASPATH" field. Complete the code below to find the longest ASPATH and assign it to the longest_aspath variable.

Questions

Q1 (5 points). If you search online for "AS number lookup" you will find several AS search services. Look up the ASes in this longest route to find their countries of origin. List these countries in order.

A1.

ASes: 12859 (Netherlands), 2914 (United States), 9498 (India), 55644 (India).

Q2 (5 points). Why might such long BGP routes be a concern for network operators? Give at least 2 reasons.

A2. Long BGP routes may be a concern because they increase latency due to more networks being traversed. They can also result in increased BGP convergence time, which delays updates and reduces stability.

Q3 (10 points). Construct the BGP table for all prefixes "owned" by AS 36992 (i.e., where the destination is in AS 36992). How many prefixes are there in this table? What is the range of prefix lengths in this table?

A3. There are 287 prefixes in this table, ranging in length from 13 - 48. The table bgp_36992.csv has been saved in the folder BGP_Analysis for this project.

Q4 (10 points). Now turn this single-destination-AS BGP table from Q3 into a routing table (see Table 14 here) as stored in the border router of AS12859. Use shortest path matching to break ties, if any. You may find it useful to implement this library: py-radix

```
Forwarding Table Example(in CSV):

Destination Prefix,Next Hop\n

a.b.c.d/24, w.x.y.z\n
```

```
a.b.c.d/32,a.b.c.d\n
```

A4.

TODO: Save the forwarding table into a csv file named forwarding-table.csv

The file forwarding-table.csv has been saved in the BGP_Analysis folder of this project.