ECS 152A - Project 1 - Report

Han Nguyen (917278789) & Blake McMurray (999162729)

1/27/2020

1 Instructions to Run the ip2as Program

The source code is **ip2as.py** and the output file is **output_student.txt** (to avoid confusions with professor's output.txt). In Terminal/Command Prompt, locate to the directory that includes the source code and the output file, and run the following command: **python ip2as.py ip2as DB_091803.txt IPlist.txt**. The DB file and IP file can be replaced upon favor of the tester, just make sure that the files are also in the same directory with the source code **ip2as.py**.

2 Programming Report & Logic

2.1 Partnership and Communication

Our partnership is formed via Discord and we maintained communication via this channel as well. It takes us total 12 hours to finish the whole Project #1, six hours of which are for the ***ip2as*** program (Part B). We start the project early and work on a daily basis. Our contribution is equal: each team member works on half of each lab and the programming assignment.

2.2 Preliminary Work

In the programming assignment, we agree to use Python programming language for our best coding convenience.

In this preliminary step, we examine the text files provided by the professor: the **DB_091803.txt**, the **IPlist.txt** and the **output.txt**. We come to understand that the goal of the programming assignment is to read through each IP address in the **IPlist.txt**, and for each IP address, we will look through the addresses, prefixes and AS numbers in the **DB_091803.txt** to find the longest prefix match for the IP address, and generate the result to an output file under the form:

[DB address with longest prefix match]/[DB prefix] [DB AS number] [IP address]

Also, we notice that there are total five invalid IP addresses in the **DB_091803.txt**. As mentioned on Piazza, we have to find a way to ignore these addresses in our analysis.

From this understanding, we start to discuss the coding logic to execute such requirements.

2.3 Coding Logic

First of all, we set up a system format check. If the user types the command in an incorrect format (not in **python ip2as.py ip2as** [**DB file**] [**IP file**]), then there will be a pop-up message that tells the user to re-enter the command. If the command line is entered correctly, the **main()** function will be called to run the program.

Inside the **main()** function, we use **open()** to open the DB file and the IP file, retrieved from **sys.argv**[**2**] and **sys.argv**[**3**]. First, we use a **while** loop to read each line of the DB file, and store the address, prefix and AS number into an array called **DB_info** by using **.split()**. Knowing that the DB file contains invalid IP addresses, we write a helper function called **validate_ip()** to check if the address is valid. If yes, then we will store such address and its according prefix and AS number in arrays called **DB_ip, DB_prefix and DB_as** by using **.append()**. The loop reads such information until end of file.

The same logic applies to store IP addresses in IP file into an array called **IP_list**.

Next, we have two **for** loops, one nested in the other, to find the longest prefix match for the IP addresses from the DB addresses. For each IP address in **IP_list**, we loop through all of the addresses stored in **DB_ip**. We use **os.path.commonprefix** to find the common prefix between the IP address and the DB address. If the common prefix length is larger than the current longest prefix match, we will keep updating under the variable **current_longest_match** and also update the index of the longest-match DB address under the variable **current_best_index**. In the special case where the new found common prefix has the same length with the current longest prefix match, we use the functions **ip_network** and **ip_address** under the Python library **ipaddress** to determine if the IP address is in range with the DB address and prefix, and if yes, choose the address with the larger prefix (closer to 32). At the end of the loop through every DB address, we have determined the value of the **current_best_index**, and will store it in the array called **longest_prefix_match_index**. We repeat the same process for all IP addresses in the IP file.

After yielding the list of all the indices of the longest-prefix-match DB addresses, we open an output file in ”write” mode. Then, we create a **for** loop to loop through every index in the array **longest_prefix_match_index**, and get the DB address, prefix and AS number at that index from **DB_ip, DB_prefix and DB_as** to write to the output file, under the format mentioned above.

2.4 Results

Our generated **output_student.txt** totally matches with the professor’s **output.txt**.

2.5 Algorithm Analysis

Our coding logic runs in $O(nm)$, where n is the number of IP addresses in the IP file and m is the number of addresses in the DB file. Our source code takes 7-8 seconds to generate the output file from the Terminal (Macbook Pro Retina Early 2015, i5 processor, 8GB RAM).

2.6 Limitations

Our coding logic relies on the Python library **ipaddress** in order to check if the IP address is in range, instead of developing our own logic to determine whether or not the IP address is in range based on the CIDR logic.

3 References

- https://stackoverflow.com/questions/3462784/check-if-a-string-matches-an-ip-address-pattern-in-python (function to check if an IP address is valid).

- https://stackoverflow.com/questions/39358869/check-if-an-ip-is-within-a-range-of-cidr-in-python (function to check if an IP address is in CIDR range)