**LONGEST PREFIX MATCH**

In this project, we implemented a software application that takes a text file containing IPV4 prefixes and their corresponding next hop addresses as input, build routing tables by loading them into specified data structures and giving various statistics about the routing table and the operations being performed on it.

The method we used was as follows:

* We construct our routing tables by loading prefixes from the input file that are longer than one bit and shorter than 31 into the specified data structures(Binary and Multi-bit tries).
* Record the time for the trie operations to construct the routing table using the **RDTSC** instruction.
* Record the storage requirements for the routing table.
* Perform the longest prefix match
* Calculate and record the minimum, maximum, mean and median times for performing operations of looking up the longest prefix match, updating and deleting prefixes from the routing tables.

We first read the input file line by line through the fgets() function and divide the input string into tokens. We then convert the prefixes into binary and the CIDR lengths into integer and pass them into the function that creates the tree.

Our first data structure to implement the routing table, the Binary Trie is a simple form of Binary search tree. We declared our structure of the trie containing two pointer elements to point towards right and left directions and a string variable to assign the next hop address for a specified IP address in the routing table.

We do not store the ip address in our trie and depend on the structure of the trie to traverse and point to the prefix in order to assign the next hop address.

The same logic is followed for the multibit trie. However we are comparing two bits as compared to 1. We check if the ip address being read in the form of binary is 00,01,10, or 11.

We wrote functions for creating, searching for longest prefix, updating and deleting for both binary and multi bit tries. We also wrote common functions and called them in both the data structures to judge the performance of the functions as required by the specifications of the project.

Deliverables:

Our program outputs the following statistics:

* Total number of prefixes in the routing table
* Number of prefixes outside of the specified lengths
* Total time to construct the routing table. This time should include only the actual trie operations for inserting the prefixes into the table. It should not include file reading and parsing time.
* Memory required to hold the routing table
* Minimum, maximum, mean, and median time to perform a longest prefix match. These times should include only the trie lookup itself. It should not include time to read the input file, parse it, or print the results.

We were not able to do the do following:

* Minimum, maximum, mean, and median time to update an entry in the routing table. These times should include only the trie lookup itself.