Instituto Tecnológico y de Estudios Superiores de Monterrey

Programación de estructuras de datos y algoritmos fundamentales

Actividad 1.3

Presentan:

Ian Seidman Sorsby A01028650

Gianluca Beltran Binachi A01029098

ASESOR: Leonardo Chang

Campus Santa Fe ***Octubre 2020***

Repositorio Github: https://github.com/Seidmani64/RetoDatabase

Se debe correr reto2.cpp dentro del folder entrega 2

Para realizar esta actividad fue necesario elegir estructuras lineales de cierto tipo para realizar lo pedido. Por esto, para guardar las conexiones entrantes utilizamos pilas, ya que se pedía acceso desde la última a la primera. Para las conexiones salientes fue lo contrario, pidiendo acceso de la primera a la última. Fue por esto que para guardar estas utilizamos colas. Gracias a estas elecciones, no fue necesario recorrer las listas para recibir elementos en el orden que se pedían, simplemente utilizando los métodos de top y front, se podían acceder a los elementos de inmediato.

Más allá de las estructuras que elegimos, también es importante entender qué métodos se usaron para realizar lo que se pidió. Para poder acceder al .csv utilizamos la implementación del ADT original realizada por el profesor, la cual crea objetos llamados Log donde cada Log es un registro del csv y a través del objeto Analytics se almacenan todos los registros. De esta manera, creando solo un objeto Analytics se puede acceder al .csv y buscar ciertas cosas dentro de los registros.

Para guardar todo lo que se pidió de esta actividad en particular, implementamos un nuevo ADT llamado ConexionesComputadora donde se almacena el IP y nombre de la computadora como inicio, y las dos estructuras lineales correspondientes (aunque estas comienzan vacías y se llenan dentro del main). La primera, conexionesEntrantes, una pila; la segunda, conexionesSalientes, una cola. El ADT cuenta también con getters y setters ya que todos los atributos son privados.

La actividad pidió que conexionesEntrantes fueran todos los IPs fuente donde la IP de la computadora fue el destino. Para hacer esto, dentro de Analytics creamos una función get_entry_connections que toma la IP como parámetro y regresa una pila llena de los IPs pedidos. Esto lo hace a través de una búsqueda lineal, por lo tanto tiene una complejidad temporal de O(n). Lo mismo hicimos para conexionesSalientes, pero en lugar de regresar una pila la función get_exit_connections regresa una cola con los IPs destino cuando la IP de la computadora es la fuente. Al igual este método tiene una complejidad temporal de O(n).

Estos métodos se corren dentro del main y se almacenan en ConexionesComputadora a través de un método setConexionesEntrantes y setConexionesSalientes que simplemente toman una cola y una pila respectivamente y igualan los atributos privados a las listas pasadas.

Para contestar las cinco preguntas se utilizaron las siguientes funciones. Para la pregunta uno se utilizó un simple input del usuario entre los numero 1 y 150. Para la segunda pregunta se utilizó getConexionEntrante(), la cual regresa el valor del top de nuestro stack en una complejidad temporal de O(1). Por último para la última pregunta se utilizó la función de la clase Analytics get_exit_connections(), en la que al pasar dos parámetros como referencia podemos saber si algun sitio fue accesado tres veces seguidas y si si que ip tiene el sitio. La función get_exit_connections() tiene una complejidad temporal de O(n) ya que pasa por nuestros logs una sola vez.

Como un ejemplo de las preguntas resueltas, se puede utilizar la computadora de Amy.

Question 1

IP being used: 172.23.5.3

Question 2

172.23.5.140 -> Internal.

Question 3

Number of entry connections: 2

Question 4

Number of exit connections: 309

Extra

This computer had three consecutive connections to this ip: 195.165.80.161