Ana Paula Katsuda, A01025303
Andreína Sanánez, A01024927
Programación de estructuras de datos y archivos fundamentales
Prof. Jorge Rodríguez Ruíz
21 de Septiembre del 2021

Reporte Reto 2

1. Introducción

Recopilar información sobre las interacciones de los usuarios con distintos servicios, es de crucial importancia para identificar si existen accesos maliciosos en redes y proteger datos relevantes. Los accesos maliciosos tienen la posibilidad de afectar enormemente a empresas e individuos; puede costar la privacidad de éstos, e incluso puede tener impactos monetarios significativos.

Dada la importancia de dar seguimiento de dichas interacciones, el objetivo del presente escrito es continuar con el análisis de las interacciones de los usuarios pertenecientes a la empresa “reto.com” con servicios externos a ella, especialmente acerca de las conexiones entrantes y salientes registradas para un individuo en particular. Esto se lleva a cabo mediante el uso de estructuras lineales, las cuales son de especial utilidad al momento de organizar y acceder a la información de la mejor manera, así como a través de la implementación de una clase ConexionesComputadora.

2. ADT

Nombre: ConexionesComputadora
Datos:
- IP (String)
- Nombre (String)
- Conexiones entrantes (Stack de Records)
- Conexiones salientes (Queue de Records)

Lista de operaciones: N/A

3. Justificación de los Algoritmos

Referente a la solución de la segunda sección del reto, es relevante mencionar que se utilizaron las estructuras lineales implementadas por la librería estándar de C++, Stack y Queue, para almacenar las conexiones entrantes y salientes respectivamente.

Comenzando con el Stack, esta estructura lineal se eligió en cierta parte, ya que al funcionar mediante el principio (last in - first out) done el último elemento agregado es el primero que sale, se facilitaba en gran medida el acceso a la última conexión entrante así como su lectura desde la última hasta la primera registrada.

De forma similar, el Queue es una estructura que se apega considerablemente bien a los requisitos necesarios para almacenar las conexiones salientes, ya que al funcionar de forma (first in - first out) hizo posible el orden de estas conexiones desde la primera hasta la última registrada.

Ahora bien, tanto el Stack como el Queue se seleccionaron principalmente debido a su complejidad temporal de $O(1)$ para la inserción de nuevos elementos “*push()*”; determinación del tamaño de la estructura; y para el acceso a la última conexión entrante y primera conexión saliente. La eficiencia de estas estructuras se aprecia especialmente al

compararla con la de otras estructuras como los vectores y las listas ligadas simples que para insertar tienen una complejidad temporal de $O(n)$. (Falta Complejidad Espacial)

4. Resolución de las Preguntas

¿Qué dirección ip estás usando?

Al utilizar la función que concatena il IP interno 172.24.238. con el número deseado (en este caso 38), obtenemos:

```
Dirección de IP que se está utilizando:
172.24.238.38
```

¿Cuál fue la ip de la última conexión que recibió esta computadora? ¿Es interna o externa?

Dado que las conexiones recibidas por la computadora son aquellas en las que el IP del equipo es el IP destino, se utiliza el “stack” de ConexionesEntrantes en la clase. Por el funcionamiento del stack, se sabe que el último elemento (top) es la última conexión recibida. Con lo anterior, se obtiene el IP de dicha conexión:

```
IP de la última conexión que recibió la computadora:
172.24.238.38
```

Es notorio que esta conexión es interna ya que se tiene el IP de la compañía 172.24.238.X.

¿Cuántas conexiones entrantes tiene esta computadora?

Utilizando la función .size() para el stock, se obtiene:

```
Número de conexiones entrantes de la computadora:
2
```

¿Cuántas conexiones salientes tiene esta computadora?

Usando la función .size() para el queue, se obtiene:

```
Número de conexiones salientes de la computadora:
2262
```

Extra: ¿Tiene esta computadora 3 conexiones seguidas a un mismo sitio web?

Para obtener las conexiones seguidas a un mismo sitio web (es decir, las conexiones salientes), se iteró en el queue para lograr comparar las direcciones de IP destino. Es importante mencionar que se requirió el uso de pop() para ir al siguiente

elemento de la cola, por lo que se utilizó una variable temporal para hacer la comparación mencionada.

```
La computadora tiene 3 conexiones seguidas con el IP address:  59.127.69.10
```

5. Aportaciones

En cuanto a las aportaciones del equipo, es importante mencionar que el análisis de la situación problema, así como su resolución fue hecha en conjunto por las dos integrantes del equipo. Si bien ciertas partes fueron realizadas de manera individual, se tuvo una sesión de revisión para verificar el correcto funcionamiento de la incorporación de cada una de las aportaciones. A continuación, se mencionan las divisiones de tareas realizadas por cada integrante:

En conjunto:

- Elaboración escrita del ATD
- Elaboración del reporte
- Implementación de la clase ConexionesComputadora

Andreína Sanánez / A01024927:

- Preguntas 3, 4 y 5.
- Comentarios explicativos del código.

Ana Paula Katsuda / A01025303:

- Preguntas 1 y 2
- Función que concatena string de IP