Tecnológico de Monterrey

Programación de Estructuras de Datos y Algoritmos Fundamentales

Actividad 1.3 - Actividad Integral de Conceptos Básicos y Algoritmos Fundamentales

Equipo 3

Paolo Antonio Pires Cano A01749355

Gabriel Edid Harari A01782146

Dr. Vicente Cubells

21 de Septiembre del 2023

Para obtener las respuestas a las siguientes preguntas se leyó el archivo CSV indicado. Todos los datos del archivo se guardaron en su atributo correspondiente (estos siendo: Fecha, Hora, IP Origen, Puerto Origen, Nombre Origen, IP Destino, Puerto Destino, Nombre Destino) en un vector de elementos de la clase Registros. Fue a través de ese vector y las diferentes técnicas de programación que se obtuvieron las respuestas.

Nota: Ambos miembros del equipo participamos de forma igual y se trabajaron todos los puntos de manera coordinada y colaborativa.

1. **¿Cuántos registros tiene tu archivo?**

Utilizando la función para vectores `.size()` se pudo calcular el número de registros dentro del vector que almacena todo el archivo CSV. El total de registros es de: 38335.

2. **¿Cuántos récords hay del segundo día registrado? ¿Qué día es este?**

Lo primero que hicimos fue volver a definir el vector de registros pero ordenado por la fecha. Para poder hacer esto tuvimos que sobrecargar el operador para comparar las fechas de manera ascendente. Una vez hecho esto utilizamos el método de ordenamiento por inserción, la razón por la cual escogimos este método es porque al intentar usar quicksort se ocupaba demasiado espacio de memoria y no funcionaba. Pero el método de inserción es más lento, tenía menos complejidad espacial y funcionó mejor. Después de esto creamos una variable de tipo string llamada `fechabusca`, a ésta le asignamos el valor de fecha del primer elemento del vector registros. Después creamos un `for loop` en el que se compara el valor de `fechabusca` con el valor de la fecha en el índice `i` del vector de registros. Una vez que la condición del loop no se cumple, el valor de `fechabusca` cambia al del valor que rompió el loop, ese valor sería un valor diferente y sería el segundo día de registros. Para el conteo creamos una variable llamada `cont` con valor igual a 0 y un `for loop` que compare el valor de `fechabusca` (el segundo día) con el valor de la fecha del registro en el índice de `i`. Si los valores coinciden el valor de `cont` aumenta hasta que los valores no coinciden y el loop se rompa. A través de esta metodología se obtuvo que el segundo día de registros es: 11-8-2020 y ese día tiene 3298 registros.

3. **¿Alguna de las computadoras pertenece a Jeffrey, Betty, Katherine, Scott, Benjamin, Samuel o Raymond? ¿A quiénes?**

Utilizando el algoritmo de búsqueda secuencial se encontró que para Jeffrey, Betty, Katherine, Benjamin, Samuel y Raymond no hay ninguna computadora a su nombre. Sin embargo, para Scott se encontró que hay 1252 computadoras a su nombre. Cabe recalcar que se utilizó la búsqueda secuencial (en todas las búsquedas de la actividad) por el beneficio de que no hay necesidad de ordenar los elementos, además de que su implementación es sencilla. Se modificó la función genérica de búsqueda secuencial para que busque conforme al atributo de Dirección de Origen y regrese y/o acepte otro tipo de dato.

4. **¿Cuál es la dirección de la red interna de la compañía?**

Lo primero que hicimos fue una variable `int` llamada `punto` la cual igualamos a 0 y dos variables de tipo string llamadas `direccionIP` y `registroIP`. Después creamos un `for loop` que pasa por todos los elementos dentro del vector `registros`, checando que el ip de origen sea diferente a `-`, y volviendo a guardar el valor del ip de origen en la variable `registroIP`, esto es posible gracias a que todos tienen la misma red interna (excepto por los que tienen - como valor). Una vez hecho esto, se crea otro `for loop` en donde pasamos por cada elemento dentro del string `registroIP` y le sumamos a nuestra variable de `direccionIP` cada elemento y si es que el elemento es un punto (`.`) aumenta el valor de la variable `punto` . Una vez que esa variable sea igual a 3 ponemos un `break` para que salga del loop y en el `cout` agregamos 000 (representando cualquier otro número). Esto no da que la red interna es: 172.21.8.000

5. **¿Alguna computadora se llama server.reto.com?**

Utilizando el algoritmo de búsqueda secuencial se encontró que no hay ningún registro bajo el nombre server.reto.com. Se modificó la función genérica de búsqueda secuencial para que busque conforme al atributo de Dirección de Origen y regrese y/o acepte otro tipo de dato.

6. **¿Qué servicio de correo electrónico utilizan (algunas ideas: Gmail, Hotmail, Outlook, Protonmail)?**

Para poder resolver este problema lo primero que hicimos fue crear un vector de tipo `string` llamado `correos` en donde metimos diferentes correos electrónicos posibles para ver qué tipos de correos electrónicos utilizan. Después se creó un `for loop` que itera por el vector con los correos electrónicos para introducirlos en el algoritmo de búsqueda. Se creó una variable de tipo `int` llamada `correoEncontrado` la cual lleva el conteo de cuántas veces se repite cada correo que encuentra loop. Se modificó la función genérica de búsqueda secuencial para que busque conforme al atributo de Nombre de Destino y regrese y/o acepte otro tipo de dato. Se obtuvieron los siguientes resultados:

Se encontraron 0 registros con los correos: hotmail.com, outlook.com, protonmail.com

Se encontraron 89 registros con el correo google.com

Se encontraron 13620 registros con el correo freemailserver.com

Se encontraron 109 registros con el correo mail.yahoo.com

7. **Considerando solamente los puertos destino ¿Qué puertos abajo del 1000 se están usando? Lista los puertos e investiga qué aplicación/servicio lo utiliza generalmente.**

Utilizando el algoritmo de búsqueda secuencial se obtuvo que los puertos utilizados abajo del 1000 son los siguientes: 135, 443, 465, 53, 67, 80, 965 y 993. Para evitar datos repetidos al obtener los puertos, los datos de la búsqueda se guardaron en un tipo de dato `<set>` el cual no permite que el mismo dato sea guardado dos veces. Se buscó conforme al atributo de Puertos de Destino.

Puerto 135: epmap.

Puerto 444: https.

Puerto 465: urd, smtps y igmpv3lite.

Puerto 53: domain.

Puerto 67: bootps.

Puerto 80: http y www-http.

Puerto 965: ipcserver.

Puerto 993: imaps y ipcserver,

Se modificó la función genérica de búsqueda secuencial para que busque conforme al atributo de Puerto Destino y regrese y/o acepte otro tipo de dato.

Reflexión Individual Gabriel Edid:

Los algoritmos de búsqueda y ordenamiento son una parte esencial de la programación. Con ellos se facilita enormemente la comprensión de conjuntos de datos, así como la eficiencia de un programa. Evidentemente existen distintas variantes de estos algoritmos y es importante considerar la eficiencia de cada uno al ser implementados a un código. Para este caso los algoritmos utilizados fueron: búsqueda secuencial y ordenamiento por inserción. Se eligió la búsqueda secuencial ya que tiene el beneficio de que no hay necesidad de ordenar los elementos, además de que su implementación es sencilla. Si se hubiera usado la búsqueda binaria habríamos de haber ordenado los elementos y al utilizar la búsqueda secuencial el programa se puede "ahorrar" esa operación y ser más eficiente. Por otro lado, se utilizó el ordenamiento por inserción ya que el método quicksort (que es más eficiente) no funcionaba ya que ocupaba mucho espacio en la memoria. A pesar de que el ordenamiento por inserción es más lento, fue el algoritmo de ordenamiento que mejor funcionó. También fue importante considerar la complejidad de cada uno de los algoritmos, lo que se encontró fue: la búsqueda secuencial tiene una complejidad en el peor caso de O(n) y el ordenamiento por inserción tiene una complejidad en el peor caso de $O(n^2)$. Estos factores son importantes a considerar para un reto de esta naturaleza ya que un ataque de una BotNet puede ser espontáneo y repentino. Es por esto que es muy importante que el programa pueda actuar de forma rápida y eficiente sin perder tiempo para que los daños no sean mayores. Entender cómo encontrar y detener una BotNet es de increíble importancia ya que es un tipo de ciberataque que puede ser imperceptible y saber implementar métodos de seguridad es cada vez más y más relevante. Es preocupante cuántas personas y empresas no tienen sus redes, dispositivos y datos protegidos, por lo que concientizarnos acerca del tema es muy valioso.

Reflexión Individual Paolo Pires:

Cuando se tiene una cantidad pequeña de datos, puedes hacer todo a mano, contar a mano, buscar a mano, incluso ordenar a mano. ¿Pero qué se hace cuando tienes 3298 registros? En momentos así es cuando es importante buscar soluciones rápidas y eficientes, especialmente para empresas y personas que trabajan con tantos datos, es imposible poder realizar tantas cosas a mano. Aquí es donde entra la programación y las varias ventajas que se tienen cuando se trabaja con un conjunto de varios datos. En este caso debemos de utilizar algoritmos de ordenamiento y búsqueda, ya que estamos tratando de resolver un problema con una cantidad "enorme" de registros. Según freeCodeCamp los algoritmos de ordenamiento son muy importantes ya que reducen la complejidad de un problema. Gracias a esto, tienen aplicaciones directas en varias cosas como algoritmos de estructura de datos, algoritmos de búsqueda y más. En este caso ordenamos todos los registros de forma ascendente en base a su fecha (para poder realizar esto tuvimos que hacer sobrecarga de operadores). En el caso de los algoritmos de búsqueda, recuerda como en el pasado la gente tenía que buscar por ejemplo un archivo en específico dentro de miles dentro de una papelera, si no se tenía algún tipo de organización se tardaban demasiado tiempo en encontrar ese archivo. Y aún así cuando estaba organizado era complicado encontrar ese archivo. Ahora imaginemos que tenemos millones de datos y necesitamos encontrar un dato en específico, sería muy difícil de encontrar a mano. Aquí es dónde entran para ayudarnos a encontrar ese dato o datos específicos los diferentes algoritmos de búsqueda que podemos utilizar. Para los dos ordenamiento y búsqueda, debemos de escoger el algoritmo que más nos convenga tomando en consideración la complejidad temporal y espacial en el caso específico con el que estamos trabajando. En el caso de ordenamiento, decidimos utilizar inserción ya que aunque Quicksort y Mergesort son bastante eficientes, son recursivos y consumen demasiada memoria. Primero se intentó con Quicksort pero no sirvió debido al espacio que ocupaba, entonces decidimos utilizar inserción y nos funcionó bastante bien aunque tuviera una complejidad de O(n^2) en el peor caso. En el caso de búsqueda, decidimos utilizar la búsqueda secuencial ya que no es necesario tener los elementos ordenados y la implementación es fácil, tiene una complejidad de O(n) en el peor caso.

Podemos ver como la programación nos facilita y optimiza varios procesos que no podríamos hacer manualmente y probablemente nos costarían bastante trabajo de otra manera. Los algoritmos de búsqueda y ordenamiento son importantes y útiles para varias aplicaciones en la vida real.

Bibliografía:

- adminsub.net. (2016, March 30). *Descubridor de puertos TCP UDP online - adminsub.net*. Adminsub.net. https://es.adminsub.net/tcp-udp-port-finder
- De La Rivera, C. (2012, June 19). *Búsqueda y Orden*. Algoritmos; Algoritmos. https://algcdlr.wordpress.com/quienes-somos/contacto/

- freeCodeCamp. (2020, 18 de Enero). *Sorting Algorithms Explained.* freeCodeCamp. https://www.freecodecamp.org/news/sorting-algorithms-explained/#:~:text=Since%20sorting%20can%20often%20reduce,structure%20algorithms%2C%20and%20many%20more.