Instituto Tecnológico y de Estudios Superiores de Monterrey

TC1031.701 Programación de Estructura de Datos y Algoritmos Fundamentales

Act 3.4 - Actividad Integral de BST

Yahir Rivera Huerta	A00572029
Saul Castañeda Carrillo	A01541099
Luisa Castaños	A01366643

Profesor: Eduardo Arturo Rodríguez Tello

Complejidad computacional y reflexiones finales

Fecha de Entrega: 24 de abril del 2022

Complejidad computacional

Algoritmo	Complejidad
Ordenar Registros	O(1)
Contabilizar la cantidad de accesos de IP	O(n)

Refelxiones Individuales

Yahir:

Dentro de la actividad 3.4 utilizamos varias funcionalidades de c++, una de ellas son los templates, los cuales nos permiten utilizar clases enteras sin especificar el tipo de dato con el cual se utilizará, gracias a esto podemos implementar de una manera sencilla la actividad de los priority heap y gracias a esto podemos hacer un BST (binary search tree) con una combinación de datos entre enteros y strings.

Además de que aprendí que para poder usar muchas funciones simples con objetos se tiene que usar la sobrecarga de operadores casi siempre y aprendí que cada operador tiene su propia forma de programar para hacer una sobrecarga ya que no todos funcionan de la misma manera y depende de para que lo quieras utilizar es el como se tiene que programar. Me gustaría recalcar que gracias a estas evidencias he podido darme cuenta que realmente no se mucho sobre el lenguaje c++, ya que en cada paso que hemos implementado hemos estado usando nuevas funciones.

Ahora bien si hablamos de nuestro resultado que es un BST ordenado de mayor a menor, de inmediato nos podemos dar cuenta de que ip se está repitiendo de manera constante y si hacemos una búsqueda por fecha sobre esa ip podríamos saber si es una red infectada por el número de accesos que ha tenido en cierto periodo de tiempo, ya que lo normal sería ver a lo mucho unas cuantas veces el acceso al día o incluso a la semana, pero si nos encontramos con una ip que acceso 50 o 40 veces al día, podemos intuir que probablemente es un bot tratando de infectar la red o en este caso el BST.

Luisa:

Los árboles binarios nos permiten hacer búsquedas, inserciones y eliminaciones de elementos con un rendimiento impresionante, puesto que su función de complejidad tiene un tiempo promedio de O(log2 N) donde N es el número de elementos y en este caso tenemos muchos IP (dirección única que identifica a un dispositivo en internet), gracias al HeapSort fuimos más eficientes en nuestro código y al contar los accesos a los diferentes IP como si fuera un ABB que nos va a permitir manejar un mayor número de datos, y es así como podemos ver si una red está infectada o no ya que al analizar todos los los accesos si la cantidad de solicitudes sobrepasa los límites de capacidad de cualquiera de los componentes de la IP ahi podemos ver que hay un error y esto sucede un periodo de tiempo corto y que normalmente no tiene tantos accesos como indico en ese momento.

Solo para complementar esta evidencia fue complicado, todavía me falta entender muchos conceptos y practicar otros tantos, pero puedo decir que no pude pedir mejor equipo para realizarla ya que con su esfuerzo pudimos realizarla lo mejor que pudimos.

Saul:

Para la solución de este reto, fue de suma importancia el saber jugar con los diferentes tipos de estructuras de datos, pues utilizamos un HeapSort con un vector con la finalidad de acomodar nuestros datos con base a su IP. Pudiendo así trabajar con una base de datos bastante extensa, que de no haber sido por la implementación que utilizamos en este HeapSort, probablemente hubiese utilizado mayor capacidad computacional para su ejecución.

Ya que pudimos realizar este ordenamiento, procedimos a hacer uso de una estructura BST, pues nos ordena nuestra base de datos de forma eficiente ya que cuenta con una complejidad de O(log2 N), colocando en la cúspide del mismo, el IP el cual tuvo mayor cantidad de accesos, y siendo sus hermanos (Left y Right) los siguientes IP's con mayor cantidad de accesos y así sucesivamente.

Aspecto que nos ayudó una vez que creamos la lista de las 5 IP 's con mayor cantidad de accesos. Este ejercicio con el BST se hace con la finalidad de conocer que IP's habían provocado la mayor cantidad de accesos, y de tener un control de esto, y nos permite el preguntarnos si dicho acceso desde tal IP, es un cliente

recurrente del servidor, o si se trata de un bot, o incluso de un ataque DDOS, tomando en cuenta: el IP, la fecha en que accedió y el número de accesos.

En conclusión, la implementación de distintas estructuras de datos, nos provee de un código tanto más limpio, como más eficiente, y al hablar de bases de datos extensas, es algo que se ha de tomar en cuenta.

Liga al replit :)

https://replit.com/join/cfqkaxwgax-saulcastaneda

Referencias

C++ Program for Heap Sort - GeeksforGeeks. (2022). Retrieved 20 May 2022, from https://www.geeksforgeeks.org/cpp-program-for-heap-sort/

Dividir string c++. (split) [duplicada] (2022). Retrieved 20 May 2022, from https://es.stackoverflow.com/questions/64062/dividir-string-c-split

¿Qué son los ataques DDoS?. Retrieved 20 May 2022, from https://latam.kaspersky.com/resource-center/threats/ddos-attacks

Overloading the << Operator for Your Own Classes. (2022). Retrieved 21 May 2022, from https://docs.microsoft.com/en-us/cpp/standard-library/overloading-the-output-operator-for-your-own-classes?view=msvc-170