Jorge Del Barco Garza
A01284234

19 de Noviembre 2021

Reflexión de Actividad 4.3: Actividad Integral de Grafos

En esta actividad, utilizamos grafos dirigidos para representar una red de computadoras interconectadas por medio del protocolo IP/TCP. Este grupo de dispositivos parece que está sufriendo un intento de ataque de tipo “brute force” desde algún dispositivo ya infectado dentro de la red. Esto se puede justificar por los mensajes de error en los logs de la red que muestran intentos fallidos de obtener acceso a otros dispositivos. El objetivo de esta actividad era encontrar al “bootmaster”, o a las posibles computadoras que intenten conectarse a la mayor cantidad de dispositivos posibles e identificarlas con una IP.

El uso de los grafos para representar redes es muy útil y es una estructura de datos en lo personal interesante para lo que es la ciberseguridad y las redes que son temas que me interesan. El hecho de ser nosotros capaces de atravesar una red de una manera ordenada y conocer los detalles de cada nodo resulta favorable. Aprendí que se pueden realizar ciertos recorridos como el Breadth-First Search o el Depth-First Search y que ambos tienen aplicaciones distintas, como podría se encontrar el camino más corto entre un nodo u otro (BFS) o para detectar si en una red existe una conexión entre ciertos nodos (DFS). Ambos ejemplos son importantes y aplicables a lo que se vio aquí que fue asegurar la integridad de una red de computadoras. Por ejemplo, si de una computadora personal de un empleado se puede llegar a un servidor operativo, ¿cuál es la ruta más corta? O ¿Existe una conexión entre ambos dispositivos a través de la red? Esto es algo que nos debemos preguntar cuando se evalúa la seguridad de una red.

Para esta actividad se eligió la representación de un grafo en el programa por medio de una lista de adyacencia. Esta se programó explícitamente con un vector de vectores de nodos (<vector<vector<Node>>>), contemplando que se iba a requerir de un acceso directo a los nodos adyacentes de cada vértice en la red, y que las operaciones de lectura y de inserción a un vector son de $O(1)$ si se realiza un *append()* o *push_back()* a la estructura.

Se optó por la lista sobre una matriz de adyacencia con el fin de optimizar el uso de memoria del programa y disminuir el nivel de complejidad computacional en espacio. Otra vía que se tomó para optimizar el programa, fue utilizar el algoritmo binary search de complejidad $O(logn)$ para acomodar las IPs en su respectivo lugar de la matríz de adyacencia y hacer un *append()* para conectar el nodo de origen con su nodo de destino y así para todas las IPs y sus nodos conectados.