Tecnológico de Monterrey

Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Monterrey

Programación de estructuras de datos y algoritmos fundamentales – TC1031

Act 4.3 - Actividad Integral de Grafos (Evidencia Competencia)

Reflexión personal

Grupo 14

Adrián Alejandro Salgado Martínez – A00828843

20 de noviembre de 2020

Estructuras de datos no lineales - Reflexión

Los grafos, dirigidos o no dirigidos, son una estructura de datos no lineales de increíble importancia para modelar y estudiar el mundo real, puesto a que su manera de representar relaciones entre información “de muchos a muchos” es de utilidad para un gran número de situaciones, desde el mapeado de ubicaciones geográficas hasta la planeación de redes informáticas para la interconexión de computadoras. Asimismo, el uso de distintos algoritmos para navegarlos y caracterizarlos, basados en su mayoría por recorridos por profundidad (DFS) o por anchura (BFS), posibilitan la comprensión y análisis sistemático de estas complejas redes y conexiones. De acuerdo a el propósito de la búsqueda o recorrido es el tipo que resulta más conveniente y eficiente, puesto a que en el caso de que se espera encontrar una solución cercana al inicio o que solo se requiera recorrer parcialmente el grafo, un acercamiento por BFS es usualmente más recomendable, en contraste a si se requiere recorrer todo el árbol usualmente DFS es más apropiado al ser más eficiente con la memoria utilizada. No obstante, todo esto es altamente dependiente de la estructura del grafo, la anchura, la profundidad, etc.

Con un contexto similar al último caso mencionado anteriormente, para la situación problema presente se planteó el uso de un grafo dirigido, optando por almacenar los accesos desde una IP hacia otra en una lista de adyacencias. Además de esto, se envolvió este grafo en su propia clase para poder trabajar con una interfaz más abstracta y reusable, basándose para almacenar esta información en el `unordered_map` de la librería estándar de C++. De este modo, las tres importantes secciones en la ejecución de la aplicación fueron las siguientes:

- Creación de vertices del grafo: En esta primera parte, se leyeron los datos de las IPs que se almacenarían como nodos o vertices del grafo, utilizando el `unordered_map::operator[]` e inicializando vectores vacíos para almacenar las adyacencias. Esto significó que esta operación es de complejidad `O(1)` por cada nodo que se inicializa, y `O(n)` para todos los nodos.
- Creación de los arcos del grafo: Posteriormente, se cargó la información de las conexiones por intentos entre IPs, de manera similar a la anterior para el acceso a las adyacencias pero en esta ocasión utilizando `unordered_map::at` para asegurar que los nodos existan en el grafo, y al también ser de complejidad `O(1)` el añadir elementos al vector de adyacencias significa que la complejidad por arco sería constante y para todos los arcos lineal.
- Búsqueda del mayor outdegree: Finalmente, para encontrar todos los nodos que en el contexto de la problemática se titulan como “bot masters”, se utilizó un mapa con los mismos nodos que la lista de adyacencia, pero en lugar del vector de nodos representaría el número de vecinos o “outdegrees”. Con esto, primero se realizó un recorrido para saber cuál era el número mayor de outdegrees en el grafo y siguientemente se realizó otro para mostrar en pantalla todos los coincidentes. A pesar de estas dos vueltas, se dice que la complejidad de esta operación es `O(n)`.

Es de esta manera que gracias al uso de un grafo se pudo encontrar una representación y solución a esta problemática, que no hubiese con la utilización de otras estructuras de datos.