CAMPUS QUERÉTARO

TC1031. PROGRAMACIÓN DE ESTRUCTURAS DE DATOS Y ALGORITMOS FUNDAMENTALES

(GRUPO 826)

"Actividad Integradora 4. Grafos"

Evidencia presentada por la estudiante:

A01706095 - Naomi Estefanía Nieto Vega

Profesor:

Dr. Eduardo Arturo Rodríguez Tello

Fecha de entrega:

Miércoles 26 de mayo de 2021

Actividad Integradora 4: Grafos

Naomi Estefanía Nieto Vega

***Resumen*—En este documento se plantea la solución para la cuarta actividad integradora, que da solución a la problemática presentada que es conocer las IPs que tienen mayor cantidad de adyacencias utilizando la estructura de datos grafos.**

***Index Terms*—data structures, graph, adjacency, heap.**

I. INTRODUCCIÓN

Hoy en día los algoritmos de ordenamiento y las estructuras de datos son muy importantes en el área computacional y en muchas más áreas en los que son utilizados ya que de acuerdo con la literatura un algoritmo es una serie o secuencia ordenada de pasos que dan solución a un determinado problema. No obstante, pueden existir distintos algoritmos que den solución a un mismo problema y es aquí donde entra la importancia de implementar el más eficiente en cuanto a tiempo y recursos ya que estos deben ser utilizados correctamente y de esta forma la computadora podrá darnos resultados muy buenos.

Asimismo, al conocer distintos tipos de estructuras de datos y su implementación podemos hacer uso de ellos para mejorar el manejo que tenemos sobre los algoritmos de acuerdo con las necesidades que tenga el problema, y así proveer una solución adecuada, eficiente y rápida para el problema presentado.

Para el desarrollo de esta cuarta situación problema, principalmente nos enfrentamos al problema del manejo de una bitácora con gran volumen de datos, aunado a esto debemos realizar búsquedas y conteos dentro de la bitácora haciendo uso de los grafos para conocer las IPs que tienen mayor cantidad de adyacencias, esto quiere decir, la cantidad de conexiones que tienen los nodos de una IP, coloquialmente podríamos llamarlo como los "vecinos" que tiene dicha IP.

II. DESARROLLO

Las estructuras de datos son formas de organización y representación de datos o información, en el caso de la programación, son formas de organización que nos permiten utilizar estos datos de una forma efectiva. Existen dos tipos de estructuras de datos:

- Estructuras de datos lineales
- Estructuras de datos no lineales

Su principal diferencia es que la estructura de datos lineal tiene a sus elementos de datos dispuestos de manera secuencial y cada uno está conectado con su elemento anterior y siguiente. Por otra parte, las estructuras de datos no lineales no tienen una secuencia establecida para conectar a sus elementos, por lo que un elemento puede tomar varias rutas para conectarse con otros elementos. Este tipo de estructuras son un poco más complejas de implementar pero son más eficientes en cuanto a uso de memoria en la computadora. [2]

Los grafos son una estructura de datos conformada por un conjunto finito de elementos o nodos (vértices) y un conjunto de aristas que los conectan. Se considera como una arista a un par (a,b) que nos dice que el vértice a está conectado con el vértice b. [3] Existen dos tipos de grafos:

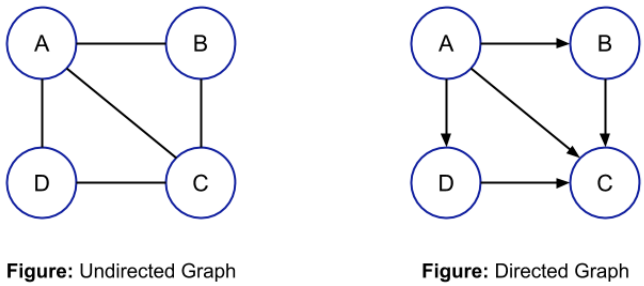

Figure: Undirected Graph **Figure:** Directed Graph

Figura 1. Tipos de grafos. Obtenido de: https://afteracademy.com/blog/introduction-to-graph-in-programming

- Grafo no dirigido

En este tipo de grafo los nodos se conectan por bordes bidireccionales, esto quiere decir que se puede ir de un nodo a otro y viceversa. Por ejemplo, podemos atravesar del nodo A al nodo B y del nodo B al nodo A.

- Grafo dirigido

Por otra parte, en los grafos dirigidos los nodos se conectan por bordes que son dirigidos, es decir, van en una sola dirección. Por ejemplo, si hay una conexión entre el nodo A y B pero la flecha apunta hacia B, significa que ese nodo sólo se puede atravesar de A hacia B y no en la dirección opuesta.

Asimismo los dos tipos de representaciones de grafos más comunes son:

- Lista de adyacencias

La lista de adyacencias se crea utilizando una matriz de listas en la que el tamaño de la matriz es igual al número de nodos o vértices, además un único índice representa la lista de nodos adyacentes al i-ésimo nodo. Se pueden usar para representar grafos ponderados y los pesos de sus aristas como listas de pares (a,b).

Las ventajas que tienen las listas de adyacencias es que ahorran mucho espacio ya que tienen complejidades de O(|V| + |E|) y en el peor caso puede existir un número C(V,2) de aristas que consuma un espacio de O(V^2). Además agregar vértices es más sencillo. Por otra parte las desventajas podrían ser que las consultas como checar si hay una conexión desde un vértice A hasta el vértice B no son muy eficientes y manejan complejidades de O(V).

- Matriz de adyacencias

La matriz de adyacencia es una matriz 2D de tamaño VxV en donde V es el número de vértices del grafo, entre sus características sabemos que para el caso de los grafos no

dirigidos esta matriz siempre es simétrica y también se utiliza para representar grafos ponderados. [4]

Entre las ventajas que tiene la matriz de adyacencias es que su representación es más fácil de implementar, y que sus funciones para eliminar una conexión o realizar una consulta del vértice A hasta el B se pueden realizar en O(1), por otro lado su desventaja podría ser que aún así el grafo consume más espacio (aunque tenga menos aristas), y que agregar un vértice tiene una complejidad de O(V^2). [4] Sus principales funciones tienen las siguientes complejidades como se puede observar en la Figura 2. Además de todo esto los grafos también pueden tener peso o no tenerlo.

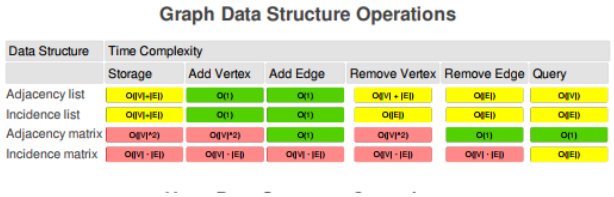

Figura 2. Complejidades de las principales funciones de grafos. Obtenida de: https://stackoverflow.com

Cabe señalar que esta estructura de datos es más compleja de utilizar pero se utiliza mucho para resolver problemas de la vida real, por ejemplo: en las conexiones entre perfiles en las redes sociales, las redes telefónicas, los mapas, entre otros.

La importancia de los grafos radica en que son formas de visualizar relaciones entre datos y su principal propósito es representar información que es muy grande en volumen de datos o muy compleja para describir en texto o menos espacio. Esto nos facilita a nosotros como personas la visualización de esta información pero además podemos añadir funciones que con base en ciertas variables nos den información valiosa, como lo es el caso de Google Maps, que calculan la mejor ruta con base en la distancia (kilómetros) entre dos vértices. Similar es el caso de Facebook, en el que los usuarios son vértices nodos y las sugerencias de amigos se basan en las conexiones que hay entre nodos. Esto permite hacer más eficientes los procesos de relación de información y sacar el máximo provecho de esta.

III. CONCLUSIÓN

En conclusión, gracias a la actividad pude comprender mejor el funcionamiento, implementación y representaciones de grafos, además de sus aplicaciones y la importancia que tienen en aplicaciones de nuestro día a día como lo son los mapas y sobretodo en la situación problema presentada, en la que gracias a los grafos se pueden realizar las búsquedas de IPs con mayor cantidad de incidencias de una forma eficiente y rápida.

REFERENCIAS

[1] Big-O Algorithm Complexity Cheat Sheet (Know Thy Complexities!) @ericdrowell. (2020). Big O Algorithm Complexities. https://www.bigocheatsheet.com/

[2] Difference between Linear and Non-linear Data Structures. (2020). Tutorialspoint. https://www.tutorialspoint.com/difference-between-linear-and-non-linear-data-structures

[3] Edpresso Team. (2021, 23 abril). What is a graph (data structure)? Educative: Interactive Courses for Software Developers. https://www.educative.io/edpresso/what-is-a-graph-data-structure

[4] GeeksforGeeks. (2021, 8 marzo). Graph and its representations. https://www.geeksforgeeks.org/graph-and-its-representations/

[5] Introduction to Graph in Programming. (2021). AfterAcademy. https://afteracademy.com/blog/introduction-to-graph-in-programming