INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

PROGRAMACIÓN DE ESTRUCTURAS DE DATOS Y ALGORITMOS

REFLEXIÓN

TC1031.501: PROGRAMACIÓN DE ESTRUCTURAS DE DATOS Y ALGORITMOS (GPO 501)

Michelle Andrea Arceo Solano.
Tecnológico de Monterrey, Campus AGS
a01625268@itesm.mx

Liga del repositorio: https://github.com/MichelleArceo/TC1031.501/tree/main/Actividad4.3

Las estructuras de datos de red permiten que los datos se encuentren interconectados, esto permite que podamos acceder por medio de distintas conexiones. En este tipo de estructuras, los grafos son la forma más común y su tipo de red es conocida como de muchos a muchos (N:M).

Existen varias formas para representar un grafo pero la forma que se utilizara para el algoritmo será la lista de adyacencia, estas de un lado se encuentra la lista de N nodos y por cada uno de ellos se encuentra una lista de nodos de los cuales tiene adyacencia. [1]

Los grafos en una lista de adyacencia tienen una complejidad total de **O(|Vertex| + |Edge|)** ya que se suma la complejidad temporal de Edge **O(|Edge|)** y Vertex **O(|Vertex|)** pero en el peor de los casos la complejidad es de **$O(n^2)$.** [2]

En la siguiente tabla podemos observar la complejidad de algunas otras operaciones que se pueden implementar en los grafos:

Operaciones	Eficiencia (Peor de los casos)
Recorridos (DFS-BFS) [2]	La complejidad es de *O(|Vertex| + |Edge|)* ya que cuando realiza el recorrido de los vértices (O(Vertex)) explora al mismo tiempo los arcos (O(Edge)) por lo que se suman las dos complejidades de tiempo.
TopologicalSort() [2]	La complejidad es de *O(|Vertex| + |Edge|)* ya que cuando realiza el ordenamiento recorriendo los vértices (O(Vertex)) y los arcos (O(Edge)) por lo que se suman las dos complejidades de tiempo.

Entre las varias ventajas, la lista de adyacencia no requiere con anterioridad la cantidad de nodos y arcos por lo que es excelente para cuando se tienen miles de datos, la única desventaja es que utiliza más espacio de memoria ya que utiliza apuntadores. [1]

Las listas de adyacencia son la estructura de datos utilizados para la mayoría de las aplicaciones de grafos por ello en esta actividad se hace la implementación de grafos por medio de la lista de adyacencia ya que es más rápido que la matriz adyacente para agregar o eliminar vértices y/o arcos, aunque son más difíciles de verificar si un arco se encuentra en el grafo, en espacio son más eficientes y eficaces.

REFERENCIAS

[1] Drozdek, A. (2012). *Data Structures and algorithms in C++*. Boston, United States of America. Cengage Learning.

[2] Allen, M. (2013). Data Structures & Algorithm Analysis in C++. Florida, United States of America. Prentice Hall.