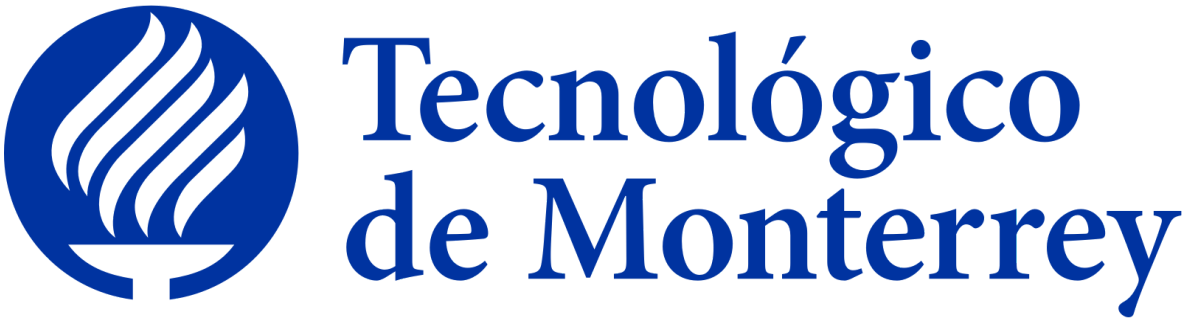
Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Monterrey



Programación de estructuras de datos y algoritmos fundamentales
Grupo 850

Después de clase | Tarea individual:Act-Integradora-4 Grafos

Docente: Dr. Eduardo Arturo Rodríguez Tello

Víctor Huerta Loretz A01365532
Jesús Alonso Galaz Reyes A00832930

Reflexión Vìctor Manuel Huerta Lorentz A01365532

En una situación como la que se acaba de resolver en la cual se analizan muchos registros, los grafos cuentan con mucha importancia y eficiencia a la hora de analizar los mismos. ya que pueden dar una representación visual de la relaciones de los mismos. para así tener un entendimiento màs claro sobre los registros y poder encontrar algùn patrón que nos pueda indicar algo màs.

En este caso utilizamos algoritmos como Dijkstra ya que son esenciales a la hora de trabajar con grafos. El ejemplo de Dijkstra es que nos ayuda a encontrar el camino más corto de un nodo a cualquier otro, utilizando estructuras como heap que permiten realizar una búsqueda eficiente. teniendo en comparación una búsqueda en grafos tendría una complejidad de O(V^2) mientras que Dijkstra tiene //O(V + E + V log V). Brindándonos una búsqueda más eficiente.

En general, los algoritmos de grafos pueden ser computacionalmente costosos debido a la necesidad de atravesar y analizar todos las aristas y nodos. Sin embargo, debido a su capacidad para modelar relaciones complejas, los gráficos brindan una forma eficiente de resolver problemas específicos, como ubicar el maestro de arranque o la ruta más corta a través de una red de conexiones.

Reflexión Jesús Galaz A00832930:

La aplicación presentada es una buena representación de cómo se pueden utilizar los grafos para resolver problemas en el mundo real, especialmente en el campo de la ciberseguridad y el análisis de redes. En este caso, se utilizó un grafo para modelar una red de conexiones entre distintas direcciones IP.

El primer requisito de la actividad era leer y almacenar datos de un archivo en una lista de adyacencias. En este caso, se eligió una lista de adyacencias porque es una forma eficiente de representar un grafo, especialmente cuando el grafo es disperso (la mayoría de los nodos no están conectados entre sí). En términos de complejidad de tiempo, la construcción de la lista de adyacencias tendría una complejidad O(n), donde n es el número de líneas en el archivo.

Después, se solicita calcular el grado de salida de cada nodo y almacenarlo en un archivo. El grado de salida de un nodo en un grafo dirigido es simplemente el número de aristas que salen del nodo. En este caso, se utiliza un mapa para almacenar los grados de los nodos, lo que resulta en una complejidad de tiempo de O(n) para calcular y almacenar los grados de todos los nodos.

La actividad también pide encontrar las 5 direcciones IP con mayor grado de salida. Para esto, se usa una implementación de un heap máximo, que es una estructura de datos eficiente para mantener un conjunto de elementos en un orden determinado. La complejidad temporal de construir este heap y extraer los cinco nodos superiores es O(n log n).

El problema del camino más corto se resuelve utilizando el algoritmo de Dijkstra, que es un algoritmo de búsqueda de camino más corto para grafos con pesos no negativos. La complejidad de tiempo del algoritmo de Dijkstra es O((V + E) log V) en este caso, donde V es el número de nodos y E es el número de aristas.

Finalmente, se pide determinar la dirección IP que requiere menos esfuerzo para que el boot master la ataque. Este problema también se resuelve con el algoritmo de Dijkstra, pero en este caso se busca el nodo con la menor distancia al boot master.

El uso de grafos en problemas como este es importante por varias razones. Primero, permite una representación visual y conceptual de los datos que puede facilitar la comprensión del problema. Segundo, los grafos permiten utilizar una serie de algoritmos y técnicas bien establecidas para resolver problemas comunes, como la búsqueda del camino más corto. Tercero, los grafos son muy flexibles y pueden ser utilizados para representar una amplia gama de situaciones y problemas.

Profe, le adjuntamos también un .zip con una versión del código que cumple hasta el requisito 4, no obstante, se la compartimos porque en ella sí pudimos dejar solo funciones en el main.cpp, no obstante, con las versión del código que sí cumple con todos los puntos, no logramos solo dejar declaraciones de los métodos en el main. Espero pueda tomar esto en cuenta cuando esté calificando.

Saludos!

Bibliografia:

- Cassingena Navone, E. (2022, octubre 24). Algoritmo de la ruta más corta de Dijkstra - Introducción gráfica y detallada. freeCodeCamp1
- Algoritmo de Dijkstra. (s.f.). En Wikipedia. Recuperado el 31 de mayo de 2023 2
- Rutas más cortas de fuente única: algoritmo de Dijkstra. (s.f.). Techie Delight3

- GeeksforGeeks. (s.f.). Heap Sort. Recuperado de https://www.geeksforgeeks.org/heap-sort/

- Sciencing. (s.f.). The Advantages of Heap Sort. Recuperado de https://sciencing.com/the-advantages-of-heap-sort-12749895.html
- GeeksforGeeks. (s.f.). Dijkstra's shortest path algorithm | Greedy Algo-7. Recuperado el 31 de mayo de 2023, de https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/
- Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to algorithms (3rd ed.). The MIT Press.
- Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. Numerische Mathematik, 1, 269-271.
- Sedgewick, R., & Wayne, K. (2011). Algorithms (4th ed.). Addison-Wesley Professional.