Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Monterrey

Programación de estructura de datos y algoritmos fundamentales

TC1031, Grupo 601

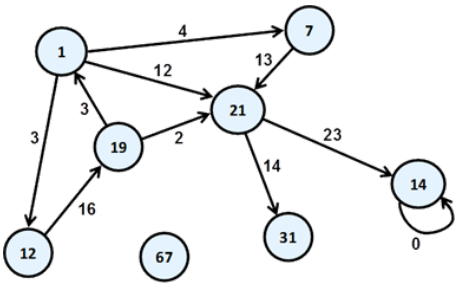
Nombre del profesor: Dr. Eduardo Arturo Rodríguez Tello

Actividad 4.3 – Reflexión personal

Samuel Acosta Ugarte | A00833547

13 de junio 2022

Grafos



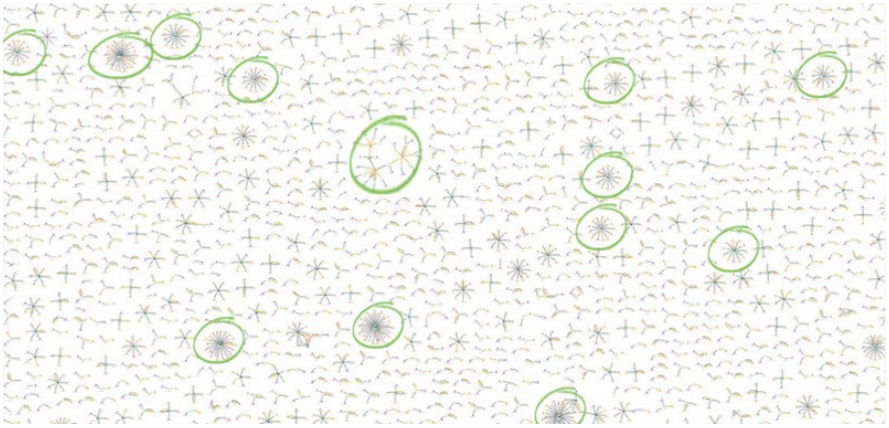
Los grafos son una estructura de datos que pueden ser usadas para representar relaciones no lineales complejas entre objetos (Isaac Computer Science, 2022). Además, ayudan a transmitir información que pude ser muy complicada de describir en un texto o un espacio (). De manera básica, los nodos contienen nodos, también llamados vértices, y aristas que conectan elementos entre ellos. En el caso de ser un grafo dirigido, los grafos tienen una conexión única, es decir que no necesariamente porque dos nodos estén conectados, tengan que estar relacionados de ida y regreso. Estos grafos dirigidos pueden ir más allá y tener una ponderación, una medida que puede indicar una distancia entre nodos, un peso o algo que describa mejor su relación.

Dos términos importantes:

- **Nodos vecinos** – Los nodos que están directamente conectados con el nodo de estudio.
- **Grado de salida o entrada** – Numero de aristas que salen al nodo de estudio y numero de aristas que entran al nodo de estudio.

Aplicaciones – Análisis de ataque

Equipos de ciberseguridad tienen en sus manos muchísima información. Cantidades tan grandes hacen la tarea de analizar ataques algo muy difícil, sin embargo, usando tecnologías de grafos, hace fácil el almacenar datos que no estar estructurados, aun cuando el volumen de datos incremente (Linkurious, 2014). Además, hace fácil ver anomalías en redes.



Realización de actividad 4.3 y análisis computacional de métodos.

En esta actividad se implementaron grafos dirigidos ponderados (representados por lista de adyacencia) que contienen pares de enteros de índice e IP. Existen diferentes contenedores usados como el mapa y el vector para facilitar encontrar cierto tipo de información acerca de cualquier ip encontrado en el grafo. En la clase grafo están declaradas, además, atributos que detallan más la relación entre un ip y otro, como el numero de aristas que tiene, el grado del nodo (ip).

Esta información fue almacenada y procesada a través de los siguientes métodos:

- **loadGraphList (string)** **Complejidad: O ((V Log V) + (E Log V))**

    - Este es el que lee la bitácora completa y almacena todo en un grafo representado por lista de adyacencia. Por cada línea que describe nodos (hay V líneas, nodos), está insertándolo en un mapa que tiene como complejidad log n, en este caso log V por insertar hasta número de nodos. Juntos se multiplican generando **V Log V** para la primera parte.

    - Cuando llegamos a las líneas donde se encuentran las incidencias, o el numero de aristas, por cada arista E se busca un valor donde recorre el numero de nodos V, creando una complejidad para esta parte **E Log V**.

    - Juntos crean la complejidad final **O ((V Log V) + (E Log V))**

- **printIpGrado ()** **Complejidad: O (V log V)**

    - En este método se imprimen todos los elementos del vector ips (igual al número de nodos V) que contiene objetos tipo ip. Por cada elemento V, también se inserta un elemento a el max heap de pares, con complejidad log n, o como estamos hablando de numero de nodos log V. Esto junto crea la complejidad **V Log V**.

- **printTop5()** **Complejidad: O (Log V)**

    - Este método se encarga de imprimir las cinco IPs con mayor grado de salida mediante un for de 5 elementos. Sin embargo, por cada elemento impreso, se realiza un pop de nuestro max heap de pares, para ver la siguiente IP con mayor grado de salida. Esto tiene una complejidad de **Log V**, donde V es el número de nodos, que también representa el número de nodos del heap.

- **printDistance ()** **Complejidad: O (E log V)**

    - Este método implementa el algoritmo de Dijkstra por medio de un priority queue, un min heap. Esto lo hace muy eficiente ya que su complejidad es E Log V. Aun cuando es modificado para imprimir mas datos, se utiliza el mismo for que ya estaba implementado, solo que con operaciones constantes O (1) de más.

Conclusión

En conclusión, esta actividad me reto, pero aprendí mucho sobre la aplicación de los grafos y como mediante de los grafos podemos analizar grandes cantidades de datos bajo nuestros propios criterios para tener las soluciones a preguntas que nosotros mismos planteamos. La versatilidad de analizar grandes cantidades de datos que se relacionan mediante grafos dirigidos ponderados es bastante enriquecedora y además muy eficiente. A lo largo de mi carrera estaré revisando muy bien cada vez que se presente un problema de esta naturaleza.

Referencias bibliográficas

Slutsky D. J. (2014). The effective use of graphs. Journal of wrist surgery, 3(2), 67–68. https://doi.org/10.1055/s-0034-1375704

Disney, A. (2017). Graph visualization use cases: cyber security. 2022, de Cambridge Intelligence Sitio web: https://cambridge-intelligence.com/use-cases-graph-visualization-cyber-security/

Linkurious. (2014). Cyber security : how to use graphs to do an attack analysis. 2022, de Linkurious Sitio web: https://linkurious.com/blog/cyber-security-use-graphs-attack-analysis/