Instituto Tecnológico y de Estudios Superiores de Monterrey

TC1031.701 Programación de Estructura de Datos y Algoritmos Fundamentales

Actividad: Reflexión 4.3

Yahir Rivera Huerta	A00572029
Saul Castañeda	A01541099
Luisa Castaños	A01366643

Profesor: Eduardo Arturo Rodríguez Tello

Reflexión Individual y Investigación

Fecha de Entrega: 31 de Marzo del 2022

Investigación

La teoría de Grafos para la ciencia y la computación es vital, ya que con ella a lo largo de la historia se ha hecho uso en areas como el análisis de las redes sociales, la sociometría, en ciencias sociales, en antropología, etc…

El principal uso de la teoría de grafos se nos muestra en problemas sobre sintesis de circuitos, contadores o sistemas de apertura y se usa en diferentes areas como el dibujo computacional o en ciertas áreas de la ingeniería. También encontramos el uso de grafos en trayectorias de autobuses, que por medio de el algoritmo de floyd podemos encontrar la ruta más optima para que se llegue a un destino de punto A a punto B.

Existen dos tipos de grafos, los cuales son ponderados y no ponderados, la unica diferencia de estos es que hay casos donde a cada arista de un grafo se le tienen que adjuntar valores, ya sean distancias o pesos, por lo tanto cuando hacemos eso como resultado tenemos un grafo ponderado, en adición a esto en estos grafos cada arista tiene que tener una dirección sin retorno.

Complejidad Computacional

Algoritmo	Complejidad
Lectura de datos	O(n^2)
Lista adyacencias	O(n+a)
Ordenar datos	O(n logn)
Búsqueda de información	O(|V|+|E|)
Despliegue de datos (grado de grafos)	O(1)

Reflexión

Yahir

Para la realización de esta entrega se nos pidió que usaramos las ip’s y los grafos para la correcta realización de esta, siendo que se tenía que relacionar una ip de entrada con una ip de salida, por lo que obviamente una de las mejores formas de hacerlo era con los grafos y teníamos que encontrar al bot master, el cual por logica podemos deducir que es quien tiene cierta cantidad de accesos a otras ip’s de una manera exagerada y no acorde a las demás ip’s.

Además creo que personalmente para mi lo que más representa un reto es siempre el saber que tipo de variable vamos a crear para hacer nuestra lista, ya que uno como programador tiene que interpretar cada uno de los puntos que nos está pidiendo la actividad y en base a eso saber que tipo de objeto o que procesos tenemos que implementar dentro de nuestras clases para generar un optimo funcionamiento de nuestro programa.

También reflexionar sobre el uso adecuado de los grafos y el como en problemas de esta naturaleza nos ayudan a encontrar de una manera optima nuestro resultado ya que por medio de las complejidades hemos estado viendo que algoritmos de ordenamiento y busqueda son los mejores para cada tipo de situación y también remarcar la implementación de el algoritmo de Dijkstra para realizar la busqueda de el camino más corto y el como esa implementación nos facilitó de una manera muy rapida esa busqueda.

Sin lugar a dudas creo que el tema de grafos es uno de los temas que mas me ha gustado por todo lo que implica en el desarrollo de el algoritmo y también la increible cantidad de metodos que se le pueden complementar para gestionar las busquedas de manera facil y rapida.

Saúl

Dado que buscamos establecer una relación entre nuestros nodos (IP 's) y un nodo en concreto (boot master), es de mucha utilidad trabajar con una estructura de datos de tipo grafo, más específicamente un grafo ponderado. Pues nos permite establecer dicha relación, mediante un conjunto de arcos, a los cuales se les asigna un peso. Y dado que buscamos encontrar el bot master, es correcto asumir que la IP, la cual tenga la mayor cantidad de arcos, es quien es la que está detrás del

ataque, pues es quien más outputs está generando. De igual forma, el hecho de conocer este bot master, y del tener asignado un peso (para este caso representa la distancia) a cada uno de nuestros arcos, nos permite calcular cuáles IP son más o menos vulnerables, dependiendo de la distancia a la que se encuentren del bot master. Esto fue logrado gracias al algoritmo de Dijkstra con una complejidad de O(|E| log |V|), lo cual para grandes cantidades de datos viene muy bien pues es una complejidad no muy pesada.

De igual forma este código representa una forma relativamente eficaz de solucionar la problemática puesto que, en el ordenamiento con priority queue del total de los arcos de cada nodo, cuenta con una complejidad O(n logn), lo que se asume como aceptable. Pero mejora este ordenamiento cuando usamos Dijkstra para ordenar los nodos y la distancia al botmaster, con una complejidad de O(|E| log |V|), lo cual se asume que es más que aceptable. Y también tenemos la función de nuestra búsqueda de información, que nos proporcionan complejidad O(|E| + |V|), que se asume que ya representa una complejidad buena. (Estas asunciones sobre la categorización de las complejidad se baso del cheat sheet de “Big-O Cheat Sheet” link: https://www.bigocheatsheet.com/)

Por último, pero no por eso menos importantes, tenemos las funciones que denomino como “complementarias” que si bien cumplen con una función primordial en el código, no representan el objetivo de nuestro reto. Las cuales son la lectura de nuestros datos con una complejidad no tan eficiente como O(n^2), pero esto es debido a que trabajamos con 2 estructuras de datos para la demás implementación del código (vector de listas de adyacencia y vector de objetos de tipo ipAddress). Así pues, tenemos la creación de las listas de adyacencia con una complejidad de O(n+a), lo cual nos indica una complejidad sumamente eficiente y por último, el despliegue de los datos con la complejidad más eficiente de todas con O(1).

Luisa

Gracias a la teoría de grados se puede resolver diversos problemas como por ejemplo el de la situación problema ya que un grafo es una estructura matemática que permite modelar problemas de la vida cotidiana, mediante, una representación

gráfica formada por nodos vértices que muestran a los actores y artistas que sirven para representar los lazos y relaciones entre los actores.

Así mismo la complejidad computacional tiene como finalidad la creación de mecanismo y herramientas capaces de describir y analizar la complejidad de un problema, su algoritmo y de identificar la eficiencia del algoritmo con independencia de la potencia de la máquina, como lo podemos ver reflejados en la evidencia que entregamos ya que trabajar con grafos nos permiten una mejor relación entre el IP y los boot master.

Y por último, como podemos ver en la tabla de complejidades son los casos que podemos llamar ideales ya que teniendo en cuenta los tiempos que lleva ejecutarse y la cantidad de espacio en memoria que requiera.

Bibliografía:

Big-O Algorithm Complexity Cheat Sheet (Know Thy Complexities!) @ericdrowell. (2022). Retrieved 11 June 2022, from https://www.bigocheatsheet.com/

Teoría de grafos - Wikipedia, la enciclopedia libre. (2022). Retrieved 11 June 2022, from https://es.wikipedia.org/wiki/Teor%C3%ADa_de_grafos#:~:text=Aplicaciones,-La%20teor%C3%ADa%20de&text=Gracias%20a%20la%20teor%C3%ADa%20de,o%20en%20%C3%A1reas%20de%20Ingenier%C3%ADa.