Tecnológico de Monterrey

Act 2.3 - Actividad integral de Grafos

Reflexión personal

TC1031

Programación de estructuras de datos y algoritmos fundamentales

Profesor: Dr Eduardo A. Rodríguez Tello

Rogelio Guzman Cruzado A01636244

Introducción

Vivimos en un punto en nuestra historia en donde la gran mayoría de nosotros nos hemos tenido que adaptar a una idea completamente nueva que implica nuestra propia responsabilidad en la forma en la que actuamos en la red, Esta siendo nuestra identidad digital y nuestra seguridad y privacidad cibernética. El hacking y los ataques cibernéticos coordinados se han convertido a lo largo de las últimas décadas en una preocupación latente de muchas empresas y de cualquier usuario promedio de la red, creando de frente la pregunta "¿Qué tan protegido estoy yo de estos ataques?

En nuestra situación problema actual, se nos muestra el ejemplo de una botnet, este siendo una red de máquinas infectadas que funcionan como zombies para actuar como un enjambre que podrían servir para crear todo tipo de crímenes cibernéticos, usando distintos tipos de estructuras de datos podremos analizar arduamente el funcionamiento y la naturaleza de esta red de máquinas, y podremos saber con certeza si tiene un propósito malicioso.

Importancia de los grafos y las estructuras de datos

Aquí es donde, se nos pondrá en duda cuál será la forma más eficiente y acertada en la que podemos representar la información que se nos da, esta siendo las ips de las máquinas en la red, y las conexiones que tienen entre sí. Debido a que una mala elección de estructura en nuestro proyecto podría imposibilitar el analizar correctamente el funcionamiento de esta botnet, y terminaríamos trabajando sobre información y datos irrelevantes que evitarán que podamos completar nuestros objetivos.

recuperado de:
https://softraysolutions.medium.com/why-you-should-know-graph-data-structures-and-algorithms-b5d88846b513

En la ciencia computacional, los grafos son utilizados como una manera de representar el flujo de la computación, por lo que en una situación problema como la nuestra, su implementación puede perfectamente representar una red de máquinas, ya que intuitivamente, podríamos representar cada IP como un nodo, y cada conexión entre dispositivos como una arista, fácilmente representando como una gran grafo toda la

botnet. Por eso mismo, la implementación de otras estructuras de datos en la solución tendrían un rol de soporte para facilitar nuestro procesamiento de la información. Por esto mismo, los grafos son utilizados en varios de los sistemas más importantes y utilizados en el campo de la ciencia computacional. Por ejemplo, el sistema de amigos de Facebook funciona como un grafo no dirigido, ya que cada usuario es representado como un nodo, y sus interacciones y amigos representan las aristas que los unen con todas las demás personas. La World Wide Web funciona como un grafo masivo de millones de páginas que considera cada página como un vértice, y sus conexiones con cualquier otra página como una arista. Este uso de grafos de la World Wide Web dio la idea base a uno de los algoritmos más importantes en el planeta, este siendo el Google Page Rank.

recuperado de:
https://www.geeksforgeeks.org/applications-of-graph-data-structure/

Complejidad de los métodos.

Clase	Método	Complejidad
Graph.cpp	void Load Graph List	O(n)
Graph.cpp	void printGraph	O(n)
Graph.cpp	void maxDegrees	O(n)
Graph.cpp	void bootMaster	O(n)
Graph.cpp	void shortestPath	O(ELogV)
Graph.cpp	void menosVulnerable	O(1)
ipAddress.cpp	getX	O(1)
ipAddress.cpp	constructor Ip	O(n)
Graph.cpp	std::priorityqueue	O(nlogn)
ipAddress.cpp	Sobrecargas de operadores	O(1)

Como podemos observar, la mayoría de los métodos utilizados en mi implementación tienen una complejidad de O(n) o O(1), haciendo el procesado de nuestra información bastante eficiente. El algoritmo de Dijkstra usualmente trabaja con una complejidad de O(v^2), pero gracias a la implementación de un max heap en el algoritmo, se logra disminuir a un O(ELogV). Estas priority queues trabajan por sí solas en una complejidad de O(nlogn) haciéndolas una herramienta esencial para mantener la eficiencia en tu proyecto, ordenando automáticamente sets de datos debido a la naturaleza de los maxheaps, podemos ahorrarnos el tener un algoritmo de ordenamiento que usualmente son difíciles de implementar, o trabajan en complejidades temporales muy altas. A lo largo de este curso he podido apreciar la elegancia que hay en poder llegar a la solución más eficiente bajo un umbral masivo de soluciones que se le pueden dar a un problema, debido a que esta preferencia por lo eficaz, nuestra tecnología tecnología ha avanzado de una manera radical y que a día de hoy, si comparamos la velocidad promedio de un dispositivo actual, con uno de hace veinte años, exista una brecha de eficiencia altísima.

info Dijkstra recuperada de:
https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/