Luis Cano Irigoyen

ReflexAct5.2

Al trabajar en estructura de datos puede llegar a ser muy importante el uso de Hash Tables, las cuales con el uso de la función de hashing permiten almacenar valores ocupando una cantidad mínima de espacio a comparación de otros valores de diversas longitudes. Esto lo logra convirtiendo llaves a hashs, almacenando los valores en la tabla ya sea utilizando un método cuadrático, lineal o de cadena. En el caso de esta situación problema, yo utilice el método de Quadratic Probing para realizar la Hash Table.

En esta situación problema nos encontramos con más de 100mil líneas de datos, contando con 13370 IPs diferentes y 91910 interacciones entre IPs; esta es una cantidad masiva de datos, por lo cual sería necesario encontrar las estructuras de datos y los algoritmos de búsqueda más eficientes para poder trabajar de manera fluida con los datos. El trabajar, por ejemplo, con búsquedas secuenciales en situaciones de este estilo, volvería el tratar los datos una tarea casi imposible, tardando tiempos muy extendidos en correr el código, pero al utilizar estructuras como la Hash Table se soluciona este problema. En el programa, los datos de la bitácora se almacenaron en un Grafo, pero en el constructor de este Grafo se adaptó para que funcionara con un hash table creada a partir de los vértices (IPs). Esto permitió más tarde que al buscar un IP en especifico para encontrar sus adyacencias, solo tuviera que realizar un findData de la función de hashing para encontrar el índice de ese IP, y poder directamente imprimir todos los valores en la fila de ese índice en la Lista de Adyacencias, sin tener que buscar más.

Esta solución lleva un proceso muy eficiente ya que las Hash Tables permiten acceder a elementos en un tiempo promedio de O(1). Es por eso que especialmente en situaciones de este estilo en donde se trabaja con cantidades masivas de datos, y un $O(n^2)$ tardaría más tiempo de lo que uno puede esperar, el uso de Hash Tables llega a ser muy importante.