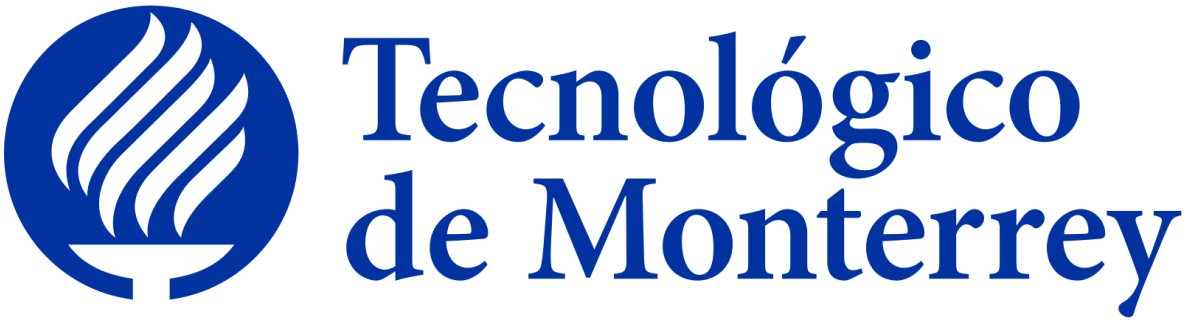
Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Monterrey



Programación de estructuras de datos y algoritmos fundamentales
Grupo 850

Después de clase | Tarea individual: Act-Integradora-5 - Uso de Hash Tables

Docente: Dr. Eduardo Arturo Rodríguez Tello

Víctor Huerta Loretz A01365532
Jesús Alonso Galaz Reyes A00832930

Reflexión Vìctor Manuel Huerta Loretz A01365532

Las tablas Hash son fundamentales en situaciones en las cuales se realiza una búsqueda y acceso a datos de manera eficiente. esto permite que esta recuperaciòn de datos cuente con una complejidad computacional de o(1), ya que esta se realiza con una funciòn hash que asigna una clave a la ubiaciòn de la tabla, permitiendo el acceso a cada elemento sin necesidad de recorrer todos los elementos a diferencia de otros métodos como árboles binarios o linked lists.

Para este tipo de problemas usar hash tables es ideal para poder buscar y recuperar los datos como las ips usadas en este problema permitiendo que la bitácora sea organizada de manera eficiente, para asì encontrar grados de salida y permitir bùsquedas eficientes para asì determinar el boost master y caminos màs cortos.

En conclusión el uso de hash tables son ideales para resolver este tiempo de problemas en los cuales se usan bastantes datos que necesitan ser rastreados, ya que permiten que la búsqueda y recuperación de los mismos sea eficiente. permitiendo algoritmos màs eficientes para interacciòn con los datos

Reflexión Jesús Alonso Galaz Reyes A00832930:

Las tablas hash son estructuras de datos extremadamente importantes y eficientes para resolver problemas donde necesitamos acceso rápido a los datos asociados con una determinada clave, en este caso, una dirección IP. La razón por la cual las tablas hash son tan eficientes es que ofrecen tiempos de búsqueda, inserción y eliminación promedio de O(1). Esto significa que podemos acceder, insertar o eliminar datos en un tiempo constante, sin importar el tamaño de la tabla hash.

Sin embargo, la eficiencia de las tablas hash depende de una buena función hash que distribuya las claves de manera uniforme en la tabla para minimizar las colisiones. Una colisión ocurre cuando dos o más claves tienen el mismo valor hash y por lo tanto deben ubicarse en la misma posición de la tabla. Cuando ocurre una colisión, se deben usar métodos adicionales para resolverla, como el direccionamiento abierto con prueba cuadrática que se usa en esta implementación. En este caso, si ocurre una colisión, la función intentará encontrar otra posición en la tabla para almacenar la información.

La eficiencia de una tabla hash se ve afectada por el número de colisiones. Un mayor número de colisiones podría resultar en un tiempo de búsqueda peor que el promedio de O(1), potencialmente llegando a O(N) en el peor de los casos si todas las claves colisionaran en la misma posición de la tabla. Sin embargo, este es un caso extremadamente raro y la

tabla hash sigue siendo una de las estructuras de datos más eficientes en la mayoría de los casos.

Al aumentar el tamaño de la tabla hash de 50,000 a 100,000, vemos que el número total de colisiones disminuye significativamente de 2428 a 1062. Esto tiene sentido porque con un tamaño de tabla mayor, hay más espacio para distribuir las claves y, por lo tanto, es menos probable que ocurran colisiones. Sin embargo, tener una tabla hash muy grande también tiene sus desventajas, ya que puede resultar en un uso de memoria ineficiente si la tabla está mayormente vacía.

En resumen, las tablas hash son una herramienta muy eficiente y poderosa para este tipo de problemas. Sin embargo, su eficiencia puede verse afectada por el número de colisiones, que a su vez depende del tamaño de la tabla y de la calidad de la función hash. A pesar de esto, las tablas hash generalmente ofrecen un excelente rendimiento en la mayoría de las situaciones.

Bibliografía

GeeksforGeeks. (2022). Hashing in Data Structure. Recuperado de https://www.geeksforgeeks.org/hashing-data-structure/

Koffman, E., & Wolfgang, P. (2016). Objects, Abstraction, Data Structures and Design: Using C++. Recuperado de https://www.wiley.com/en-us/Objects%2C+Abstraction%2C+Data+Structures+and+Design%3A+Using+C%2B%2B-p-9780471467557

Programiz. (2023). Hash Table in C/C++. Recuperado de https://www.programiz.com/dsa/hash-table