Instituto Tecnológico y de Estudios Superiores de Monterrey

Arturo Sanchez Rodriguez - A01275427

Programación de estructura de datos y algoritmos fundamentales (Gpo 850)

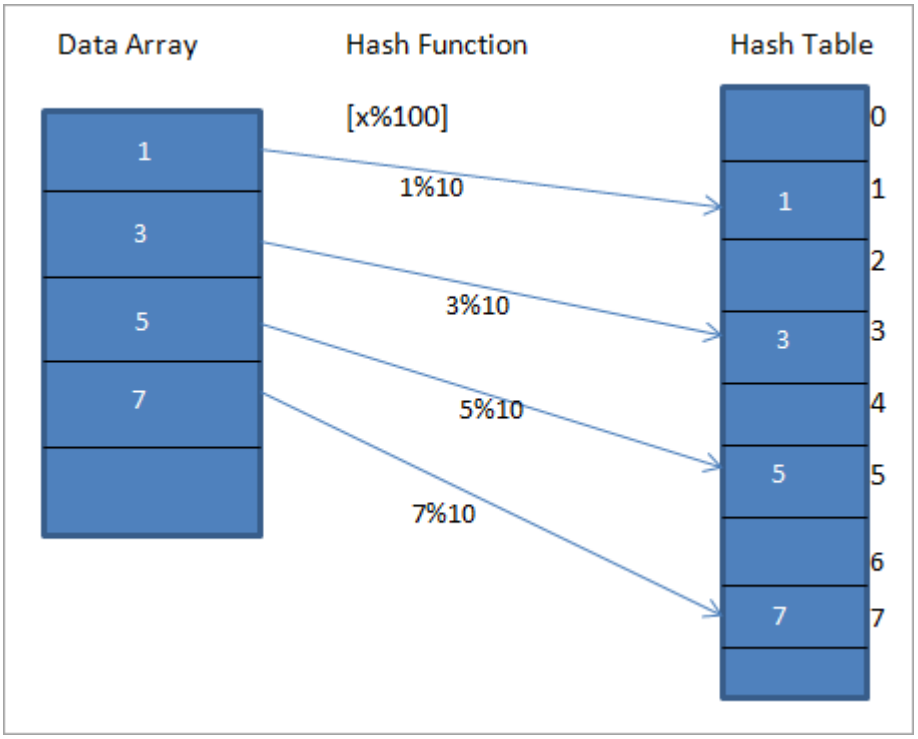
ReflexActIntegradora5

10 de Junio del 2023

Introducción:

Las tablas hash son estructuras de datos ampliamente utilizadas en ciencias de la computación y programación debido a su eficiencia y versatilidad. En esta investigación, se explora la importancia y eficiencia del uso de las tablas hash en situaciones problema, así como la complejidad computacional asociada con su implementación. Además, examinaremos cómo la complejidad se ve afectada cuando el número de colisiones en la tabla hash aumenta.

Las tablas hash son estructuras que permiten almacenar y recuperar información de manera rápida y eficiente. Se basan en una función de dispersión (hash function) que mapea los elementos a posiciones específicas en la tabla. Esta función asigna una clave única a cada elemento, lo que facilita la búsqueda y recuperación posterior de datos.



Una de las principales ventajas de las tablas hash es su eficiencia en términos de tiempo de acceso. En promedio, la búsqueda, inserción y eliminación de elementos en una tabla hash tiene una complejidad de O(1), lo que significa que el tiempo requerido para estas operaciones no depende del tamaño de la tabla. Esto contrasta con otras estructuras de datos, como las listas enlazadas o los árboles, que pueden tener una complejidad de O(n) en el peor de los casos.

Sin embargo, la eficiencia de una tabla hash depende en gran medida de la calidad de la función de dispersión y de la forma en que se manejan las colisiones. Las colisiones ocurren cuando dos elementos diferentes son asignados a la misma posición de la tabla hash por la función de dispersión. En tales casos, es necesario resolver estas colisiones utilizando técnicas como el encadenamiento (chaining) o la resolución por exploración (probing).

Cuando el número de colisiones en una tabla hash aumenta, la eficiencia del acceso a los elementos puede disminuir. Esto se debe a que las colisiones requieren tiempo adicional para resolverlas y puede llevar a un aumento en la longitud de las cadenas o al agrupamiento de elementos en ciertas posiciones de la tabla. Si el número de colisiones es alto, la complejidad de las operaciones de búsqueda, inserción y eliminación puede aumentar, acercándose a una complejidad lineal O(n).

Es importante destacar que existen técnicas para reducir el número de colisiones, como el uso de funciones de dispersión más sofisticadas y el rehashing periódico para redistribuir los elementos en la tabla. Estas técnicas ayudan a mantener una eficiencia aceptable incluso en presencia de colisiones, pero su implementación y configuración adecuada son fundamentales para obtener buenos resultados.

Conclusión:

Las tablas hash son estructuras de datos importantes y eficientes que ofrecen tiempos de acceso rápidos para la búsqueda, inserción y eliminación de elementos. Su complejidad computacional promedio de O(1) las hace muy atractivas en situaciones donde la velocidad y la eficiencia son prioritarias.

Sin embargo, la eficiencia de una tabla hash puede verse afectada negativamente si el número de colisiones aumenta significativamente. Las colisiones pueden ralentizar las operaciones y acercar la complejidad a una escala lineal, especialmente si no se gestionan adecuadamente.

Es esencial elegir una función de dispersión adecuada y aplicar técnicas de manejo de colisiones para minimizar su impacto. A pesar de las colisiones, las tablas hash siguen siendo una opción eficiente en muchas aplicaciones y ofrecen una mejora significativa en comparación con otras estructuras de datos más tradicionales. La comprensión de la complejidad asociada con las tablas hash y cómo se ve afectada por las colisiones es fundamental para aprovechar al máximo su potencial y utilizarlas de manera efectiva en situaciones problemáticas.

Referencias -

1- Tabla hash en C ++: programas para implementar tablas hash y mapas hash. (s. f.). *Otro*. https://spa.myservername.com/hash-table-c-programs-implement-hash-table

2- *Cómo implementar una tabla hash de muestra en C/C++*. (s. f.). https://es.linux-console.net/?p=6742#gsc.tab=0