**Hash**

**1. Introducción**

En el ámbito de la informática y la programación, los algoritmos de búsqueda juegan un papel fundamental en la organización y recuperación eficiente de datos. Uno de los métodos más importantes utilizados en estos algoritmos es el Hashing, una técnica que permite asociar claves con valores de forma rápida y eficiente. Este manual tiene como objetivo proporcionar una comprensión clara y concisa del concepto de Hashing y su aplicación en los algoritmos de búsqueda.

**2. ¿Qué es el Hashing?**

El Hashing es una técnica que se utiliza para mapear datos de tamaño variable a datos de tamaño fijo. Esto se logra mediante una función denominada función hash, que toma una entrada (o 'clave') y la transforma en una dirección única en una tabla de dispersión (o 'hash table'). Esta dirección se utiliza entonces para acceder al valor asociado con esa clave de forma rápida.

**Imagen 1** Ciclo de vida del TDA

**3. Componentes del Hashing**

Función Hash: Es el corazón del proceso de Hashing. Esta función toma una clave como entrada y devuelve una dirección única en la tabla de dispersión. Una buena función hash debe ser rápida de calcular y producir distribuciones uniformes de direcciones.

Tabla de Dispersión (Hash Table): Es una estructura de datos que almacena los pares clave-valor. Está compuesta por un conjunto de 'buckets' o 'slots' donde se almacenan estos pares. La dirección devuelta por la función hash determina en qué 'bucket' se almacenará el par clave-valor.

Colisión: Ocurre cuando dos claves diferentes generan la misma dirección hash. Esto puede ocurrir debido a limitaciones en la función hash o a la naturaleza del conjunto de claves. Las colisiones deben ser manejadas adecuadamente para garantizar la integridad de la tabla de dispersión.

**4. Operaciones de Hashing**

Inserción (Insert): Consiste en añadir un nuevo par clave-valor a la tabla de dispersión. Para ello, se calcula la dirección hash de la clave y se inserta el par en el 'bucket' correspondiente. En caso de colisión, se utiliza una estrategia de resolución de colisiones.

Búsqueda (Search): Implica recuperar el valor asociado a una clave dada. Se calcula la dirección hash de la clave y se busca en el 'bucket' correspondiente. Si se encuentran colisiones, se resuelven de acuerdo a la estrategia establecida.

Eliminación (Delete): Permite eliminar un par clave-valor de la tabla de dispersión. Se busca la clave en la tabla, se elimina el par si se encuentra y se ajusta la estructura de la tabla si es necesario.

**5. Aplicaciones del Hashing en Algoritmos de Búsqueda**

Búsqueda Rápida: El hashing permite una búsqueda rápida y eficiente de elementos en grandes conjuntos de datos. La complejidad temporal de las operaciones de búsqueda, inserción y eliminación es O(1) en promedio, lo que significa que el tiempo de ejecución no depende del tamaño de los datos.

Gestión de Colisiones: A través de diversas técnicas de resolución de colisiones, como el encadenamiento o la exploración lineal, el hashing permite gestionar eficazmente las colisiones y mantener un rendimiento óptimo incluso en presencia de claves duplicadas.

Estructuras de Datos Eficientes: Muchas estructuras de datos populares, como los conjuntos y los mapas, se implementan utilizando tablas de dispersión basadas en hashing. Estas estructuras ofrecen un equilibrio entre la rapidez de acceso y la flexibilidad en la manipulación de datos.

**6. Conclusiones**

El Hashing es una técnica poderosa y versátil utilizada en una amplia gama de aplicaciones informáticas, especialmente en algoritmos de búsqueda. Proporciona una manera eficiente de asociar claves con valores y permite realizar operaciones de búsqueda, inserción y eliminación en tiempo constante en promedio. Comprender los fundamentos del hashing y su aplicación en los algoritmos de búsqueda es esencial para cualquier desarrollador de software que busque optimizar el rendimiento y la eficiencia de sus aplicaciones.

**7. Referencias**

Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. "Introduction to Algorithms." 4th Edition, The MIT Press, 2020.

Sahni, Sartaj, and Sanguthevar Rajasekaran. "Data Structures, Algorithms, and Applications in Java." 2nd Edition, Wiley, 2018.

Lafore, Robert. "Data Structures and Algorithms in Java." 2nd Edition, Sams Publishing, 2019.

Drozdek, Adam. "Data Structures and Algorithms in C++." 5th Edition, Cengage Learning, 2019.