Tecnológico de Monterrey

Actividad 2 - Implementación de una tabla Hash

Leon Daniel Vilchis Arceo A01641413

Álvaro González Martínez A01646343

Gregorio Alejandro Orozco Torres A01641967

Daniel Hernández Gutiérrez A01640706

Lorenzo Orrante Román A01641580

12 de septiembre del 2025

Programación de estructuras de datos y algoritmos fundamentales

Grupo 604

Explicación tabla hash:

Una tabla hash es una estructura de datos que sirve para guardar información en manera de clave y valor de manera muy rápida, se podría decir que funciona como un conjunto de cajones numerados y que cada clave se convierte en un número mediante una función hash, y ese número indica en qué cajón debe ir el dato. En cuanto al rendimiento, insertar, buscar y eliminar suelen hacerse en tiempo constante (O(1)) porque normalmente basta con ir directo al cajón que corresponde, sin embargo, en el peor de los casos, cuando la tabla está muy llena y hay muchas colisiones, puede tocar revisar varias posiciones y la operación se vuelve más lenta, llegando a O(n)

Justificación del método de colisiones:

En las tablas hash puede pasar que dos claves distintas caigan en la misma posición de la Aquí se eligió "quadratic probing" como forma de resolverlas. Esto funciona de manera de que primero, intentamos poner la llave en su posición “natural” (la que da la función hash), pero si vemos que ya está ocupada, en vez de movernos al siguiente lugar inmediato, buscamos otro, pero no lo vamos buscando de uno en uno, sino que vamos probando posiciones más lejos cada vez, por ejemplo primero avanzamos 1 lugar, luego avanzamos 4 lugares, luego 9 lugares, luego 16 lugares, siempre empezando desde el lugar original y dando vueltas con el “% capacidad” para no salirnos de la tabla. Esto sirve para que los datos no se amontonen todos seguidos, sino que queden más repartidos.

Complejidad de insert, get y remove:

Insert: En el mejor de los casos este método tiene una complejidad de O(1), esto es cuando no hay ninguna colisión o cuando actualiza una clave que ya existe que encuentra inmediatamente. En el peor de los casos podemos decir que tiene una complejidad de O(n), ya que puede haber ocasiones en donde existan muchas colisiones y se tenga que recorrer toda la tabla.

Get: Tiene una complejidad de O(1) cuando la clave está en la posición hash original. Si existen muchas colisiones, se deberá de recorrer varias posiciones haciendo una complejidad de O(n).

Remove: Es un método con complejidad O(1) cuando hay un factor de carga razonable, por ejemplo cuando la clave se encuentra en su posición inicial. Si se debe de recorrer muchas direcciones por colisiones, la complejidad se convierte en O(n).

- **Link del video**:
  https://youtu.be/4waTavk23FQ