**Instituto Tecnológico y de Estudios Superiores de Monterrey.**

**PROGRAMACIÓN DE ESTRUCTURA DE DATOS Y ALGORITMOS FUNDAMENTALES.**

****

**Actividad 5.2: Actividad Integral sobre el uso de códigos de hash.**

## Profesor: *Luis Humberto González Guerra.*

**Alumna*: A00831510*** Andrea Marisol Salcedo Vázquez

*Grupo 1*

*28/11/2021*

**Reflexión actividad 5.2**

Andrea Marisol Salcedo Vázquez

Durante esta actividad hicimos uso de unordered\_map el cuál es un tipo de hashing por lo que no guarda los elementos en un orden, es decir, el primero elemento que insertas no quiere decir que estará en la primera posición, así como el último elemento que insertas no necesariamente se encuentra en la última posición, si no que es como los algoritmos que vimos en clase los cuales usaban módulos o divisiones para decidir en qué posición se guardará el elemento.

Para realizar el algoritmo primero guardamos los elementos de descripciones de puestos que consistían en una string y un valor entero el cuál indicaba la cantidad que vale esa string, estos los guardamos en el unordered\_map en donde la palabra era la llave y el entero el valor que se guarda en dicha posición actuando como un diccionario, se busca la llave pero se obtiene el valor que está guardado dentro.

Una vez que guardamos los valores recibimos los párrafos en los cuales mediante un while pudimos ir evaluando palabra por palabra cuando el usuario la insertaba, si el string era una llave la cual contenía un número se le sumaba al total, en caso contrario de que ese string no existiera en la llave entonces se le suma un 0. Cuando el string insertado es un punto entonces se detecta que hay un cambio de párrafo y se despliega el total, para después volverlo 0 para sumar las palabras del siguiente párrafo y le resta uno a la cantidad de párrafos para que de esta forma cuando este sea 0 el while se detenga y no sea infinito.

Como logramos observar esta actividad fue sencilla gracias a las técnicas de hashing que vimos con anterioridad ya que si hubiéramos usado alguna otra técnica como vectores y algoritmos de ordenamiento y búsqueda vistos anteriormente hubiéramos obtenido complejidades más grandes ya que la mayoría de ellos son O(log(n)) u O(n) mientras que la complejidad para encontrar algo mediante este método es O(1) es decir que es constante y en un algoritmo esa es la mejor complejidad que podemos obtener.

De igual forma, si hubiéramos usado otras estructuras que vimos con anterior como las LinkedList o Grafos no obtendríamos complejidades constantes ya que para llegar a un elemento de alguno de los anteriores tenemos que recorrer los que están antes de estos, mientras que en hashing vas directo al elemento que buscas sin necesidad de pasar por otros previamente para encontrarlo y si este no se encuentra en la posición que esperabas no tienes que recorrer otras posiciones como lo vimos en los algoritmos que implementamos de hashing en los cuales podría estar ocupada la posición esperada por otro elemento y entonces debíamos pasar a la siguiente posición y obteníamos un algoritmo lineal por lo que no era lo más optimo usarlos.

Finalmente, podemos observar que dependiendo de la situación es el algoritmo que podemos implementar y que es más eficiente, así como que un método puede ser implementado de diferentes maneras y aunque no todas son igual de eficientes algunas son más útiles en ciertos casos.