



Tablas de hash



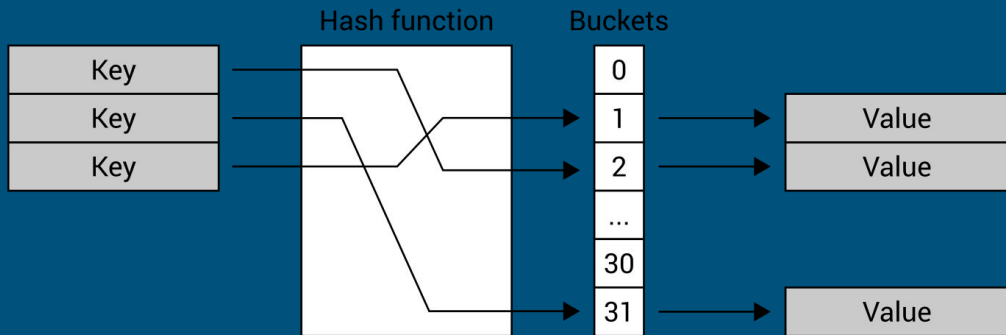
“Donde el $O(1)$ está a la mano”



Introducción

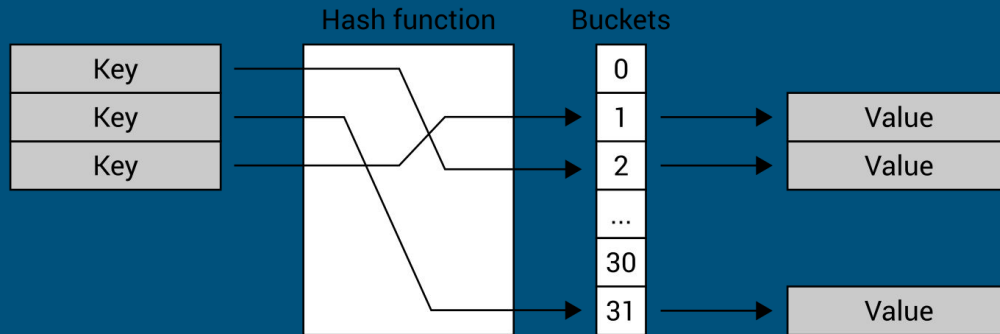
Una aproximación

Estructura de datos que relaciona dos elementos (claves y valores) y permite realizar operaciones (insertar, buscar y eliminar) en $O(1)$ promedio.



Elementos/componentes

- Un array donde se almacenan los datos clave-valor (K,V).
- A cada celda de dicho array se lo conoce como “buckets”.
- Una función de hash $f:K \rightarrow \text{Int}$ que determina la posición de los elementos (que bucket tiene asignada).



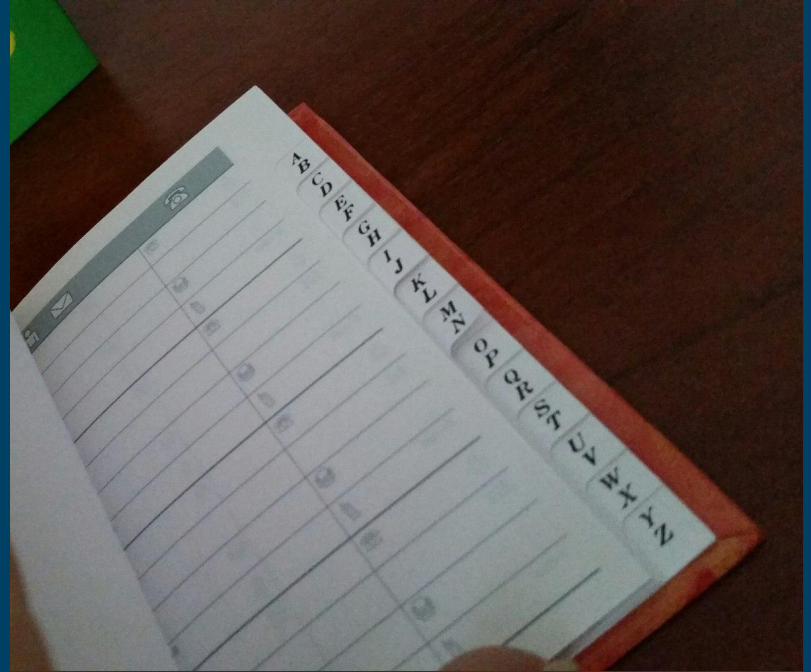
Ejemplos

- DNS (String->String): dominios -> IPs
- Agenda (String->Int): nombres -> #teléfono
- Dictado (Int->List<Alumnos>): #dictado -> listado de estudiantes
- Tabla de puntos (String->Int): nombre equipo -> puntos

Ejemplo: agenda

String -> Int

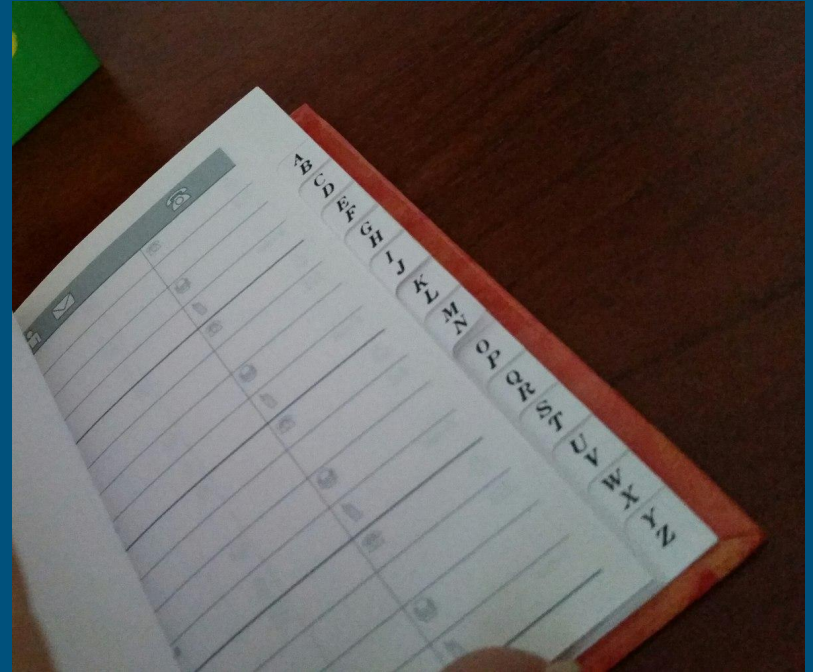
Nombre -> #telefono



Tablas de hash desde antes de saber lo que es

Cuando utilizamos una agenda estamos utilizando una tabla de hash implícitamente. Puede reconocer los siguientes elementos? :

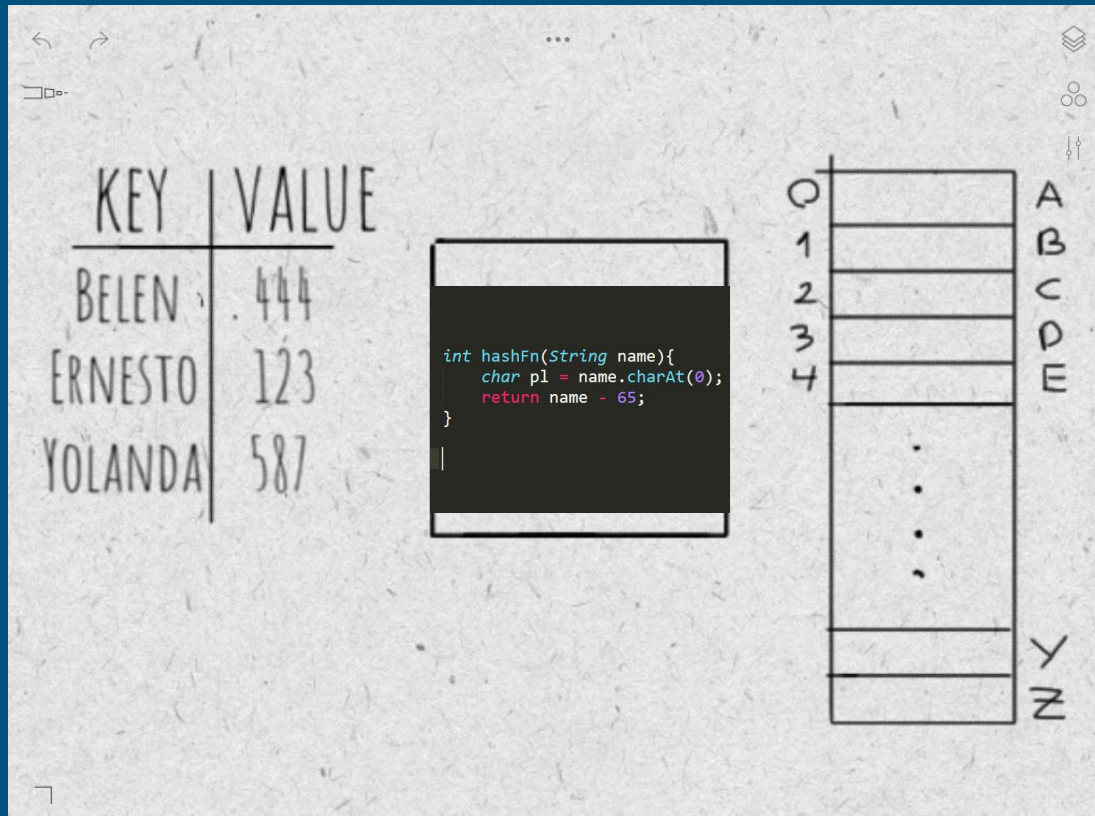
- Clave
- Valor
- Función de hash



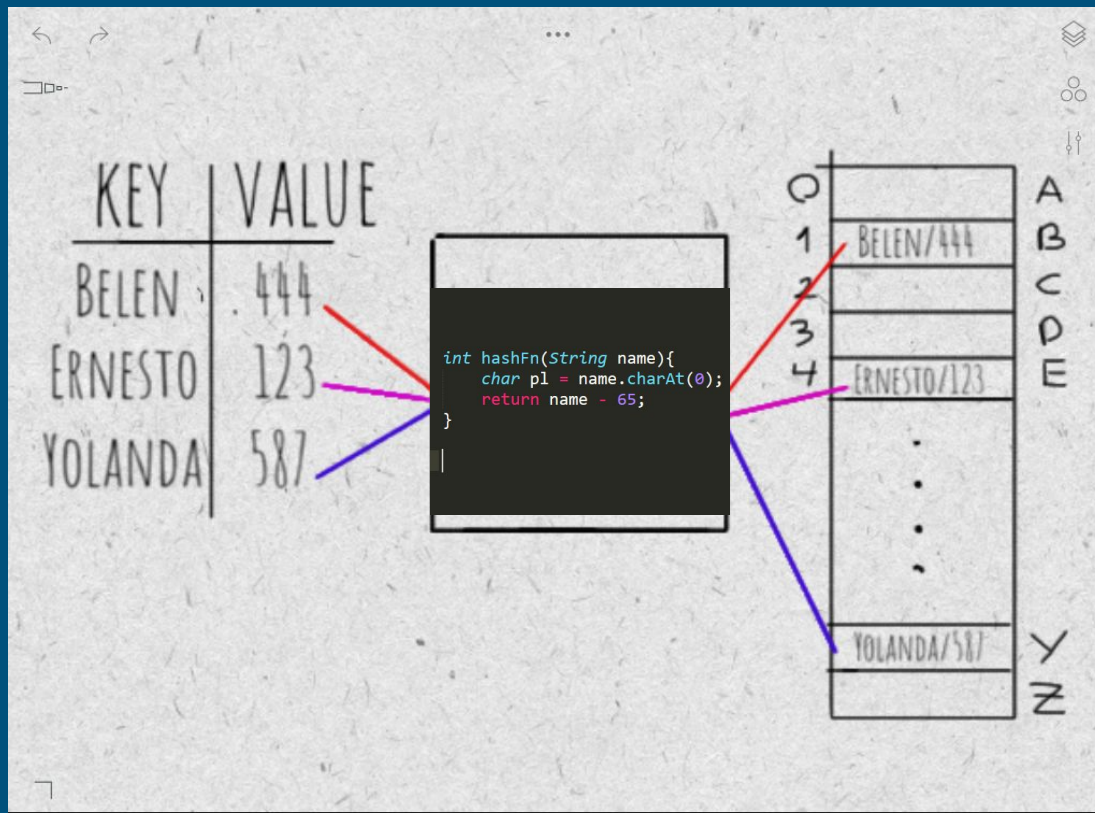
KEY	VALUE
BELEN	444
ERNESTO	123
YOLANDA	587

0		A
1	444	B
2	123	C
3	587	D
4	...	E
		Y
		Z

Ejemplo de inserción



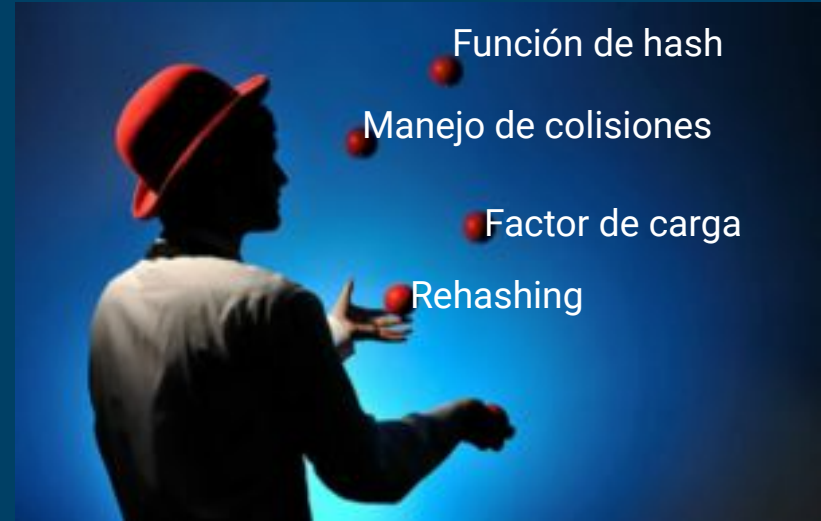
Ejemplo de inserción



Ejemplo de inserción

Puntos clave

Garantizar el $O(1)$ depende de muchos factores



Función de hash

Una elección clave

Qué buscamos de una “buena” función?

1. **Evitar colisiones**, es decir, que sea una función dispersa
2. Que sirva a medida que vaya creciendo la cantidad de datos
3. De $O(1)$

Funciones hash, ejemplos de distribución

<https://cseweb.ucsd.edu/~kube/cls/100/Lectures/lec16/lec16-13.html>

<https://cseweb.ucsd.edu/~kube/cls/100/Lectures/lec16/lec16-15.html>

Colisión: cuando con dos entradas diferentes se determina usar el mismo bucket. Ej: “Belén/123” y “Bernardo/443”

Manejo de colisiones

Algo inevitable

Cómo manejamos las colisiones?

- Hash abierto (a continuación)
- Hash cerrado
 - Lineal
 - Cuadrática
 - Doble hash



Factor de carga

A tener en el radar

Factor de carga - definición

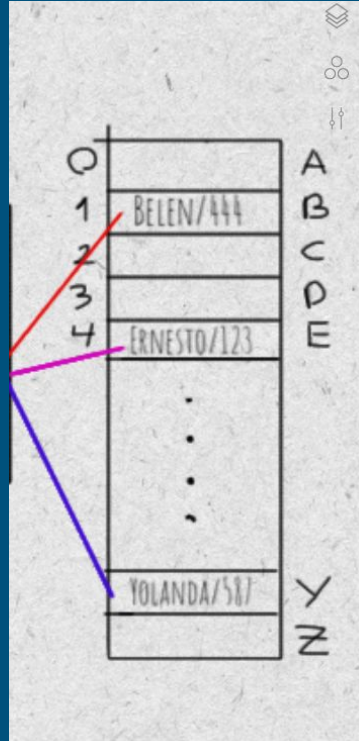
El factor de carga es un número (≥ 0) que indica que tan “lleno” está nuestra estructura.

$$\lambda = N / B$$

Donde N es la cantidad de elementos y B la cantidad de buckets

Factor de carga - ejemplo

$$\lambda = N / B$$



A hand-drawn table with 10 rows and 2 columns. The left column contains labels Q, 1, 2, 3, 4, followed by three dots, and then YOLANDA/587. The right column contains labels A, B, C, D, E, followed by three dots, and then Y and Z. A red line connects the label '1' to the cell containing 'BELEN/444'. A pink line connects the label '4' to the cell containing 'ERNESTO/123'. A purple line connects the label 'YOLANDA/587' to the cell containing 'YOLANDA/587'. There are also three dots in the cell between 'ERNESTO/123' and 'YOLANDA/587'.

Q		A
1	BELEN/444	B
2		C
3		D
4	ERNESTO/123	E
.	.	.
YOLANDA/587	YOLANDA/587	Y
		Z

人 - 人

Factor de carga

- Un factor de carga “alto” hace propenso las colisiones
 - Un factor de carga “bajo” quiere decir que reservamos espacio que no se usa.
 - Buenos valores? 0.5 - 0.7
 - Qué hacer cuando crece? -> rehashing (crecer la cantidad de buckets)
-
- En cuanto a los valores posibles, hay alguna diferencia entre el hash abierto y el hash cerrado? <-

Rehashing

Rehashing

A medida que la tabla de hash crece, nuestro λ también lo hace, lo cual aumenta la posibilidad de colisiones.

Una técnica es el re-hashing, que consiste en aumentar la cantidad de buckets.

El re-hashing es una operación MUY costosa y debe ser debidamente planeada.

El nuevo tamaño tiene al menos el doble de buckets.

Tiene $O(N)$ promedio.

<https://www.youtube.com/watch?v=7PuZOsxe-mo>

Otras consideraciones



Órdenes

Data Structure	Time Complexity								Space Complexity
	Average				Worst				Worst
	Access	Search	Insertion	Deletion	Access	Search	Insertion	Deletion	
Hash Table	N/A	$\theta(1)$	$\theta(1)$	$\theta(1)$	N/A	$O(n)$	$O(n)$	$O(n)$	$O(n)$

Uso

Las tablas de hash pueden ser muy útiles en operaciones como: buscar, insertar y eliminar elementos.

Son pobres para el uso de operaciones donde los elementos están relacionados entre sí, por ejemplo listar de forma ordenada, o buscar el mínimo.

Operaciones

- `void insertar(K,V)`
- `void eliminar(K)`
- `bool existe(K)`
- `V recuperar(K)`
- `bool esVacia()`

TAD Tabla \leftrightarrow Tablas de hash

TAD Tabla



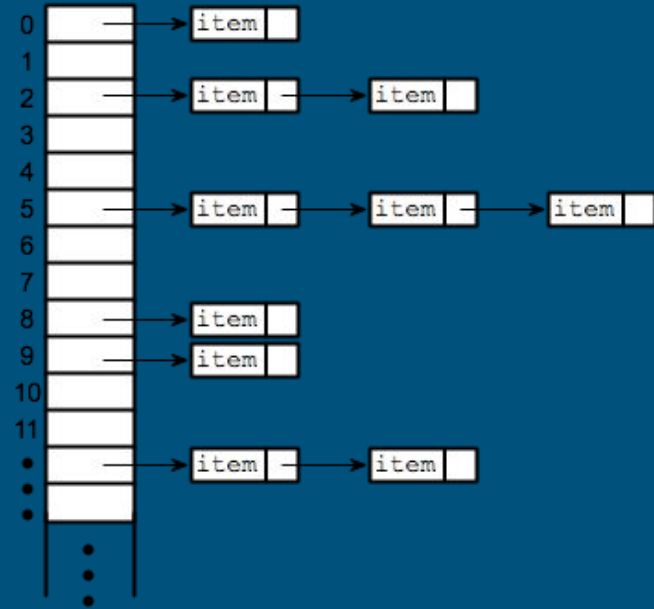
Tablas de
hash

Hash abierto

Hash abierto - Intro

La forma más popular y simple de manejar las colisiones es el hash abierto.

Los datos no se almacenan directamente en las buckets, sino que cada bucket apunta a una estructura, por lo general una lista.

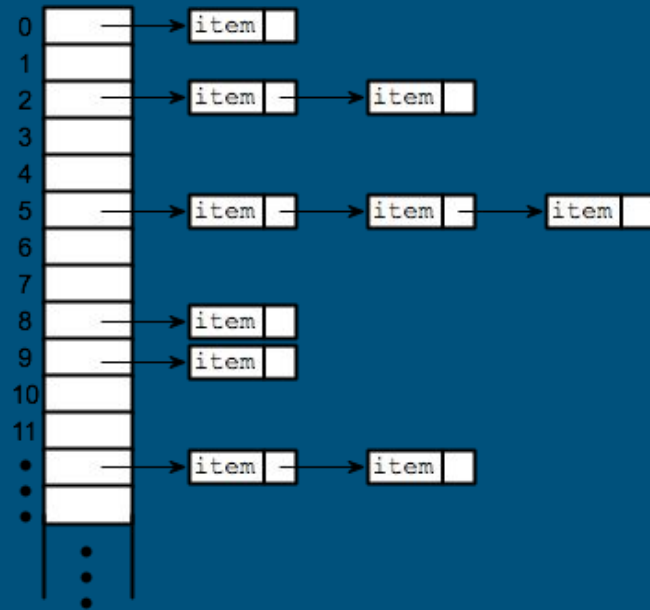


Hash abierto - en acción

<https://www.cs.usfca.edu/~galles/visualization/OpenHash.html>

ver las funcionalidades de:

- Insertar
- Buscar
- Eliminar



Tiempo de codificar

Hash cerrado

Hash cerrado - Intro

A diferencia del hash abierto, el hash cerrado almacena los elementos directamente en los buckets.

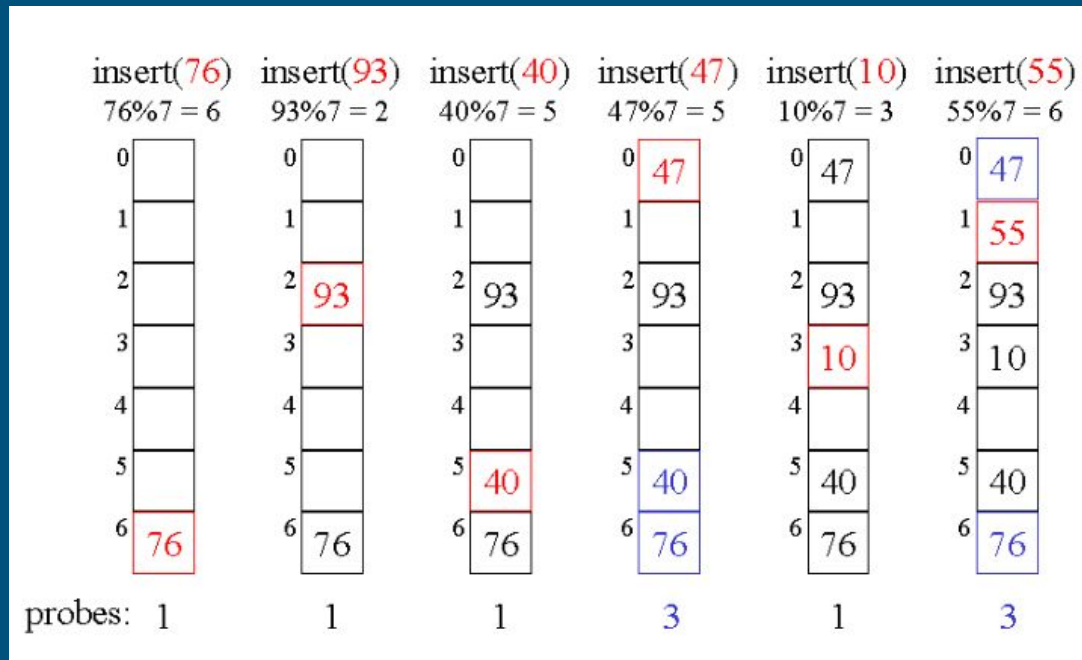
Las tres formas de resolución de colisiones son:

1. Lineal
2. Cuadrático https://en.wikipedia.org/wiki/Quadratic_probing
3. Doble hash https://en.wikipedia.org/wiki/Double_hashing

<https://www.cs.usfca.edu/~galles/visualization/ClosedHash.html>

Hash cerrado - Lineal

Básicamente resuelve el próximo bucket a visitar de forma lineal (sumando 1 por ejemplo)



Hash cerrado - cuadrático

$$H + 1^2, H + 2^2, H + 3^2, H + 4^2, \dots, H + k^2$$

Hash cerrado - doble hash

$$h(i, k) = (h_1(k) + i \cdot h_2(k))$$

*Importante: la cant. de buckets tiene que ser un número primo para que no haya ciclos

Hash cerrado - en acción

<https://www.cs.usfca.edu/~galles/visualization/ClosedHash.html>

ver las funcionalidades de:

- Insertar
- Buscar
- Eliminar

Tiempo de codificar

Breve puesta a punto

<https://forms.office.com/Pages/ResponsePage.aspx?id=zSCX18DYDE2kBC3NAI8B4zHaHUx7m7VBrn630x8c1m5URTVZVTNKT1lSNDNOR1VZMUoxRjIwTFJRUY4u>

Links de interés

- <https://www.geeksforgeeks.org/hashing-set-1-introduction/>
- <https://www.geeksforgeeks.org/hashing-set-2-separate-chaining/>
- <https://www.geeksforgeeks.org/hashing-set-3-open-addressing/>
- <https://www.geeksforgeeks.org/double-hashing/>
- <https://stackoverflow.com/questions/9127207/hash-table-why-deletion-is-difficult-in-open-addressing-scheme>
- <https://www.geeksforgeeks.org/load-factor-and-rehashing/>
- <https://www.youtube.com/watch?v=7PuZOsx-mo>