Instituto Politécnico Nacional

Escuela Superior de Cómputo

Estructuras de Datos

Tema 05: Tablas hash

M. en C. Edgardo Adrián Franco Martínez
http://www.eafranco.com
edfrancom@ipn.mx
@edfrancom *edgardoadrianfrancom*

Contenido

- Función de dispersión o hash
  - Ejemplo de uso de un función hash
- Tablas hash o de dispersión
  - Clave y contenido
  - Tablas hash cerradas o de direccionamiento abierto
  - Tablas hash abiertas, de direccionamiento cerrado o encadenamiento separado
- Funciones de dispersión para una tabla hash
  - Método de la división
  - Método de la multiplicación
  - Método de la suma
- Funciones de Hash h(k,i)
  - Ejemplo de hashing H(k,i) cerrado

Función de dispersión o hash

- Una función hash (función picadillo, función resumen o función de digest), es una función H computable mediante un algoritmo,

$$H: U \rightarrow M$$

$$x \rightarrow h(x)$$

- Tiene como entrada un conjunto de elementos, que suelen ser cadenas, y los convierte (mapea) en un rango de salida finito, normalmente cadenas de longitud fija. Es decir, la función actúa como una proyección del conjunto U sobre el conjunto M.

- Normalmente el conjunto M tiene un número elevado de elementos y U es un conjunto de cadenas con un número más o menos pequeño de símbolos. Por esto se dice que estas funciones resumen datos del conjunto dominio.

- La idea básica de un valor hash es que sirva como una representación compacta de la cadena de entrada. Por esta razón decimos que estas funciones resumen datos del conjunto dominio.

$$H: U \rightarrow M$$
$$x \rightarrow h(x)$$

Ejemplo de uso de un función hash

El código ASCII asigna un número a cada letra o signo de puntuación

65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**

97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**	**k**	**l**	**m**	**n**	**o**

33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
!	**"**	**#**	**$**	**%**	**&**	**'**	**(**	**)**	*****	**+**	**,**	**-**	**.**	**/**

ASCII, es un estándar internacional.

Podemos substituir cada letra de un texto por su código ASCII

E	n		u	n		r	i	n	c	ó	n		d	e	
69	110	32	117	110	32	114	105	110	99	243	110	32	100	101	32

I	a		M	a	n	c	h	a		d	e		c	u	y
108	97	32	77	97	110	99	104	97	32	100	101	32	99	117	121

o		n	o	m	b	r	e		n	o		q	u	i	e
111	32	110	111	109	98	114	101	32	110	111	32	113	117	105	101

Podemos utilizar los códigos ASCII de un texto para hacer cualquier cálculo

E	n		u	n		r	i	n	c	ó	n		d	e	
69	110	32	117	110	32	114	105	110	99	243	110	32	100	101	
	-1312			224			990			-15840			-6868		-22806

	I	a		M	a	n	c	h	a		d	e		c	
32	108	97	32	77	97	110	99	104	97	32	100	101	32	99	
	-7372			-4365			1144			6500			6831		2738

u	y	o		n	o	m	b	r	e		n	o		q	
117	121	111	32	110	111	109	98	114	101	32	110	111	32	113	
	-444			-8658			1254			7590			8927		8669

-11399

Aquí, cada tres caracteres, con sus códigos ASCII, se opera

(1º-2º)*3º

La suma de los resultados es una **función HASH** que identifica perfectamente el texto.

Cualquier modificación en el texto provoca un cambio en el valor de la función HASH

E	n		u	n		r	i	n	c	o	n		d	e	
69	110	32	117	110	32	114	105	110	99	111	110	32	100	101	
		-1312			224			990			-1320			-6868	-8286

	l	a		M	a	n	c	h	a		d	e		c	
32	108	97	32	77	97	110	99	104	97	32	100	101	32	99	
		-7372			-4365			1144			6500			6831	2738

u	y	o		n	o	m	b	r	e		n	o		q	
117	121	111	32	110	111	109	98	114	101	32	110	111	32	113	
		-444			-8658			1254			7590			8927	8669
															3121

Por ejemplo, al substituir “**rincón**” por “**rincon**” sin acento, el valor HASH ha pasado de -11.399 a 3.121

Ejemplo de uso

Ana envía un mensaje a Benito.
Al final del mensaje le añade el valor HASH del texto según una función en la que se han puesto previamente de acuerdo.

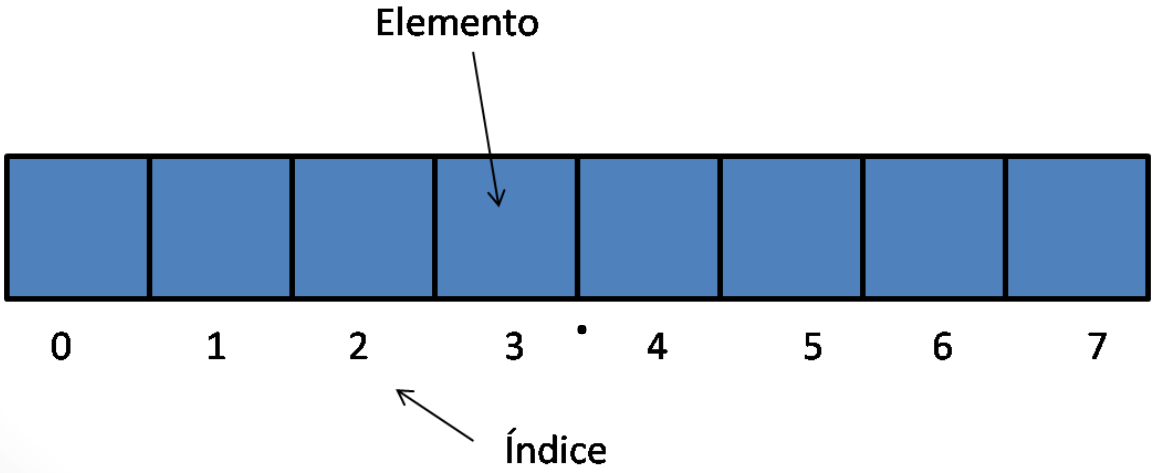
Benito recibe el mensaje y calcula el valor HASH.
Si coincide con el que ha dicho Ana puede estar seguro de que el mensaje no ha sido modificado.

Tablas hash o de dispersión

- Muchas aplicaciones requieren una estructura de datos que soporte las operaciones de un diccionario: ***Insert***, ***Search***, ***Delete***.

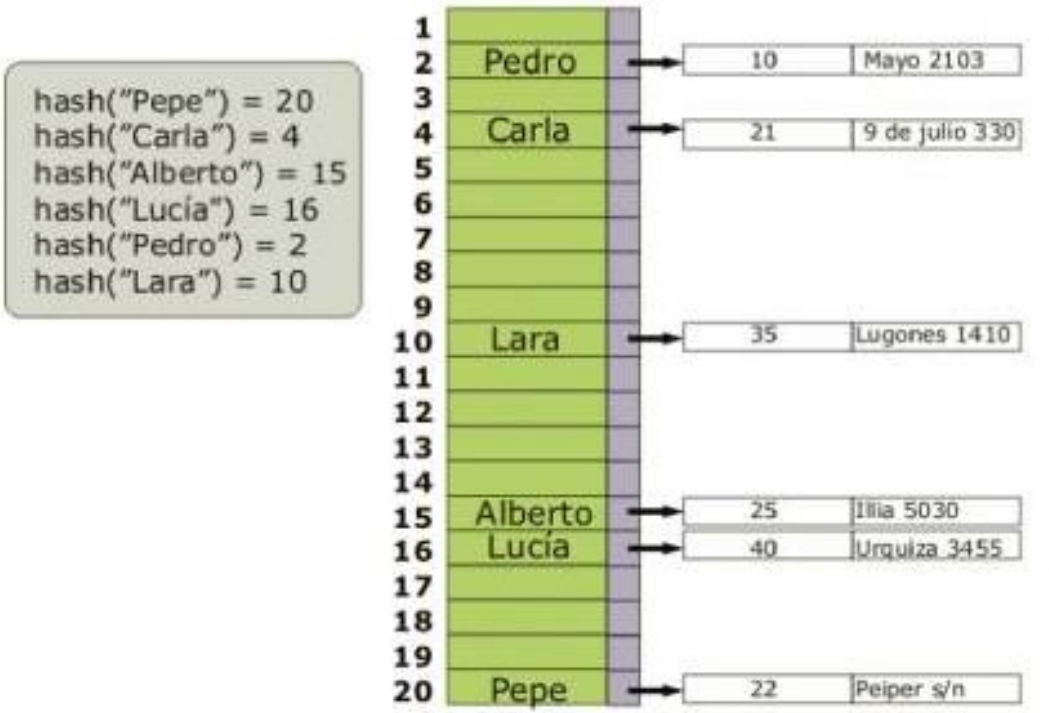
- Es posible hacer uso de una lista enlazada con un tiempo **O(n);** sin embargo, este tiempo se puede reducir notablemente a orden **O(1)** en la mayoría de los casos usando una **tabla hash o de dispersión**

- La idea surge de **los arreglos** que nos permiten acceso a sus elementos en orden **O(1).**



- **Una tabla hash o mapa** hash **es una estructura de datos** que asocia llaves o claves con valores. La operación principal que soporta de manera eficiente es la búsqueda: permite el acceso a los elementos almacenados a partir de una clave de este.

- Funciona transformando la clave con una función hash en un hash, un número que la tabla hash utiliza para localizar el valor deseado.



- Las tablas hash se suelen implementar sobre vectores de una dimensión, aunque se pueden hacer implementaciones multi-dimensionales basadas en varias claves. Como en el caso de los arrays.

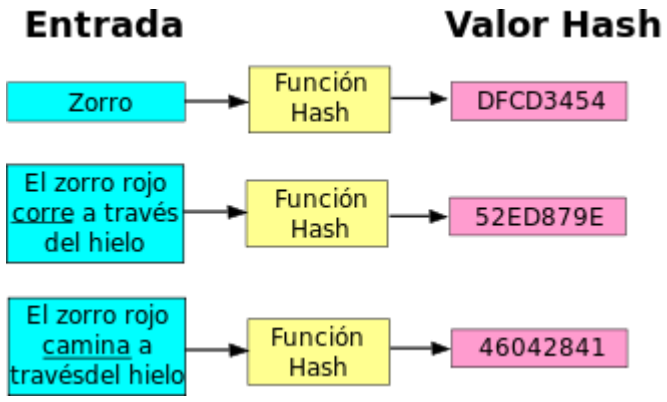
- Las tablas hash proveen tiempo constante de **búsqueda promedio O(1),** sin importar el número de elementos en la tabla. Sin embargo, en casos particularmente malos el tiempo de búsqueda puede llegar a **O(n),** es decir, en función del número de elementos.

- Comparada con otras estructuras de arrays asociadas, las tablas hash son más útiles cuando se almacenan **grandes cantidades de información**.

- Si se desea tener una estructura dinámica similar **los arreglos** que permita el acceso a sus elementos en orden **O(1),** se tienen básicamente dos opciones:

1. Una opción sería usar un arreglo tan grande como el rango de posibles claves. La desventaja es el espacio de memoria requerido en tal estrategia.

2. Otra opción es usar un arreglo menor, al cual podemos mapear las claves en uso. Esta **función de mapeo** es la **función hash**. La tabla así organizada es la tabla hash.



Clave y contenido

- El arreglo se organiza con elementos formados por dos miembros: **clave** y **contenido**.

Clave	Contenido
253	Elemento 1
124	Elemento 2
***	***
021	Elemento n

- La **clave** constituye el medio de acceso al **contenido**.

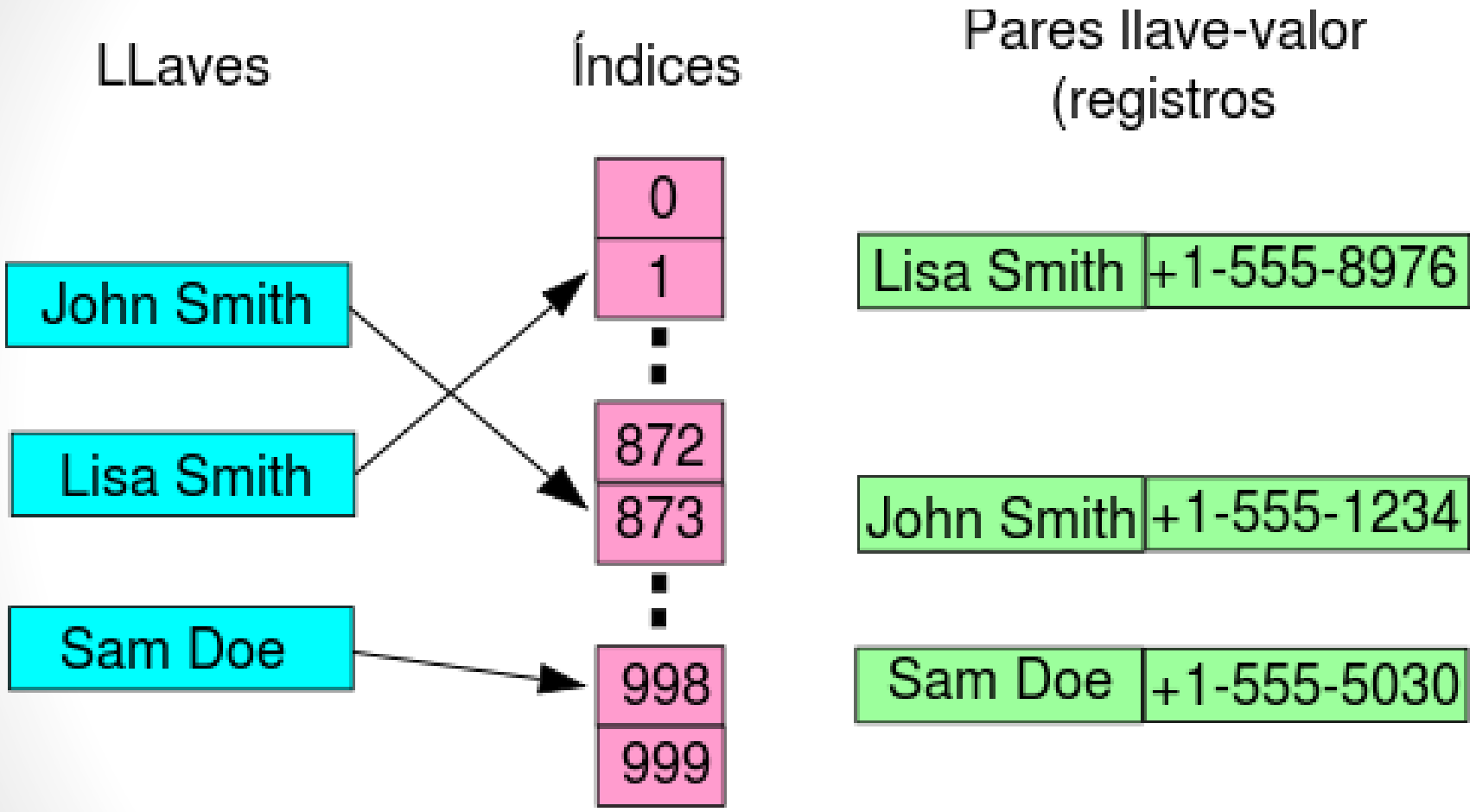
- Aplicando a la **clave** una función de acceso **f**, previamente definida, obtenemos un número entero **i**, que nos da la posición del elemento correspondiente del arreglo: **i = f(clave)**.

- Conociendo la posición, tenemos acceso al contenido. El contenido puede ser la información, ó un apuntador a la dirección si la cantidad de datos es muy grande. Este acceso se conoce como **acceso directo**.

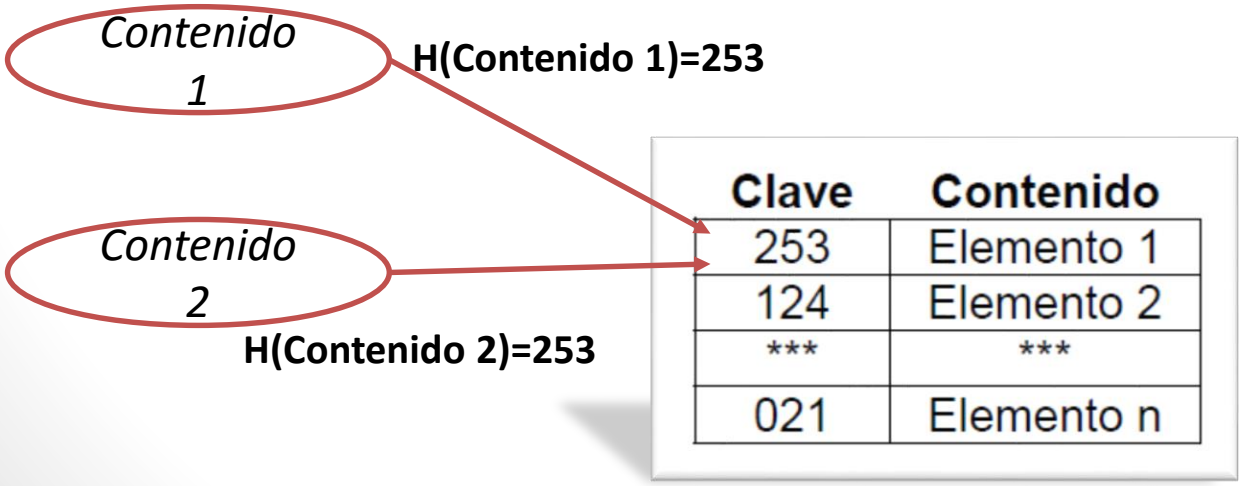
- Sin embargo, el cálculo de la **clave** es un factor crítico en este tipo de búsquedas, debido a que **varios contenidos pueden producir valores idénticos para la clave**, este proceso se conoce como **colisión**.

  - Existen varias técnicas que permiten reducir el número de colisiones que se presentan en cada conjunto de datos. La manera en que cada una de esas técnicas resuelva las colisiones afecta directamente la eficiencia de la búsqueda.

Clave	Contenido
253	Elemento 1
124	Elemento 2
***	***
021	Elemento n


LLaves
Índices
Pares llave-valor
(registros
John Smith
Lisa Smith
Sam Doe
0
1
872
873
998
999
Lisa Smith
+1-555-8976
John Smith
+1-555-1234
Sam Doe
+1-555-5030

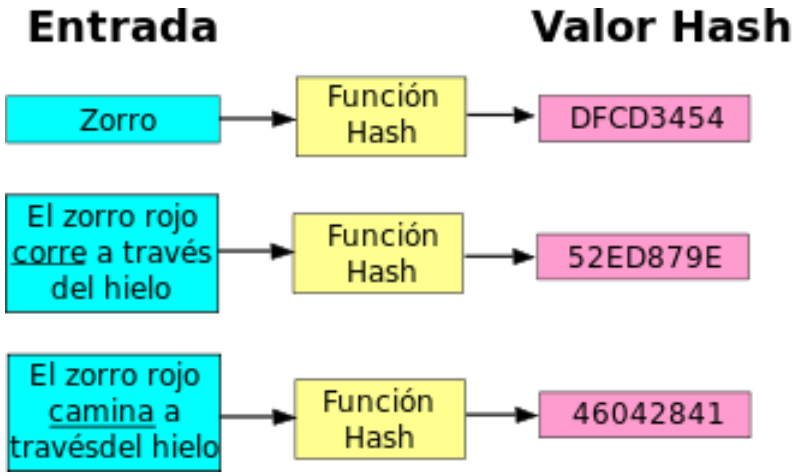
- Los **algoritmos sobre las tablas hash** son métodos de búsqueda, que proporcionan una **longitud de búsqueda** pequeña y una flexibilidad superior a la de los otros métodos, como puede ser el método de búsqueda binaria, el cuál requiere que los elementos del arreglo se encuentren ordenados. La propiedad más importante de una buena función de hash, es que pueda ser calculada muy rápidamente, y que al mismo tiempo se minimicen las colisiones.



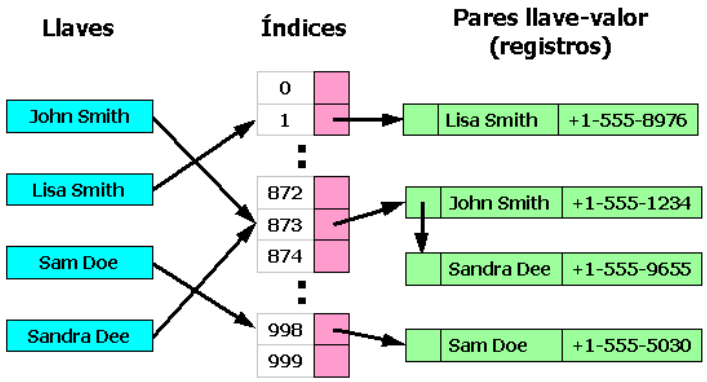
- Por **longitud de búsqueda** se entiende el número de accesos que es necesario efectuar sobre un arreglo para encontrar el elemento deseado.

- Las **tablas hash**, permiten como operaciones básicas: búsqueda, inserción y supresión.

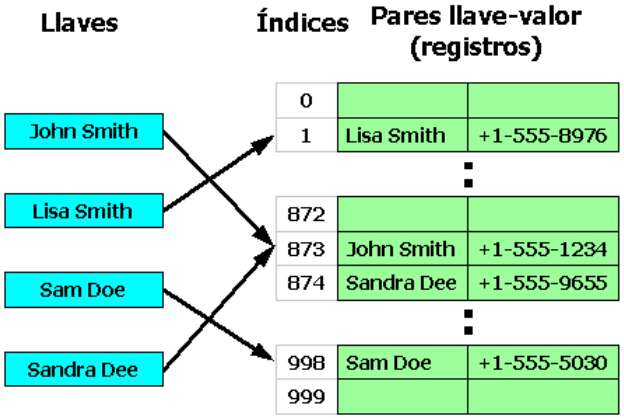
- Un **arreglo hash** es un arreglo producto de la aplicación de una **función de *hasheo***.



- Para dos claves que conduzcan al mismo mapeo (colisión), es necesario buscar formas para resolver esta situación.
  - Una forma, conocida como **hashing abierto, de direccionamiento cerrado o encadenamiento separado**, crea una lista asociada a cada entrada del arreglo.



  - Otra forma, conocida como **hashing cerrado o direccionamiento abierto**, almacena las claves en las mismas entradas del arreglo o tabla hash.



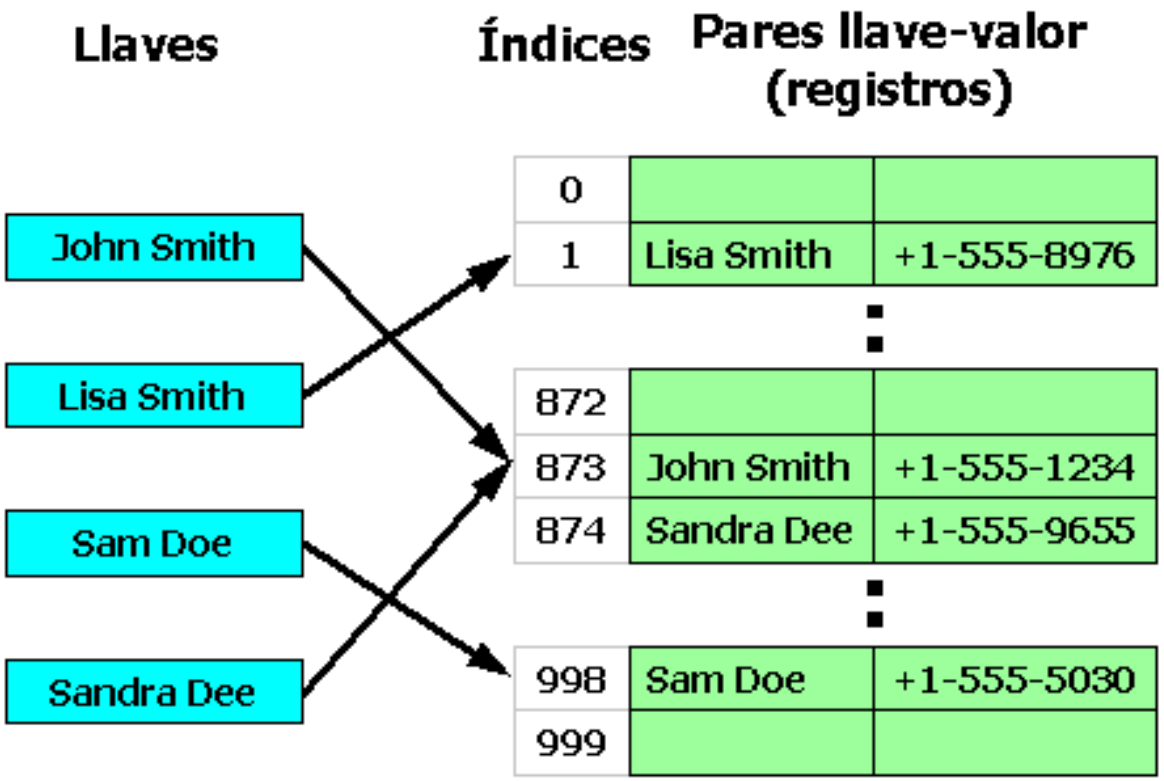
Tablas hash cerradas o de direccionamiento abierto

- La **dispersión cerrada** o de **direccionamiento abierto** es un tipo de tablas hash que almacena los registros (Contenidos) directamente en la tabla hash.

- Una tabla Hash cerrada se utiliza cuando el número maximo de elementos que se va almacenar es conocido de antemano y se puede crear desde el inicio una tabla del tamaño especifico a utilizar.

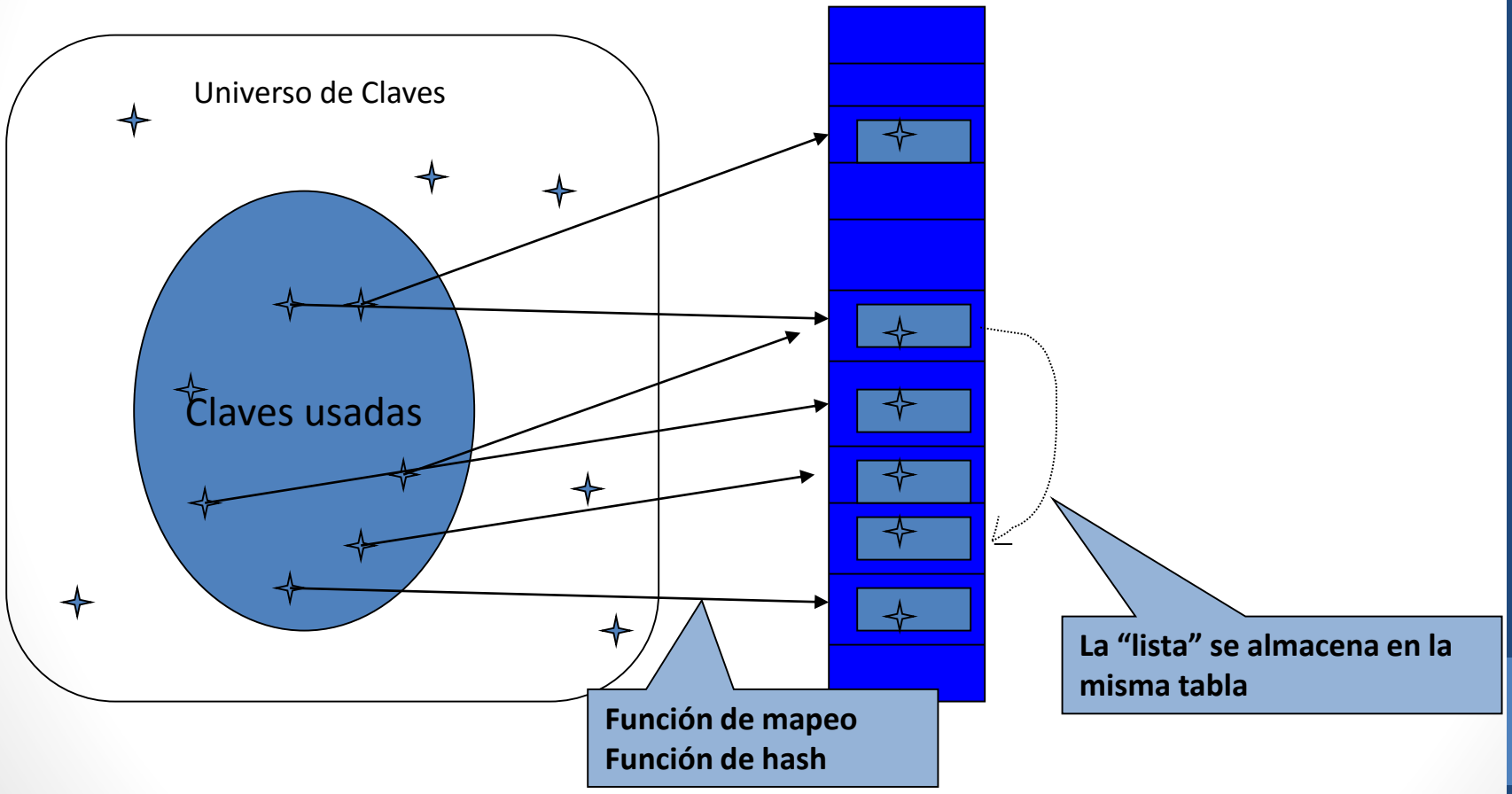
- Las colisiones se resuelven mediante un sondeo de la tabla, en el que se buscan diferentes localidades (secuencia de sondeo) hasta que el registro es encontrado o se llega a una casilla vacía, indicando que no existe esa llave en la tabla.

- En un sistema de dispersión cerrada, si ocurre una colisión, se buscan celdas alternativas hasta encontrar una vacía; por lo que se necesita una tabla más grande para poder cargar todos los datos.
  - i.e, todos los elementos se almacenan en su propia tabla hash o dispersa. Las colisiones se resuelven calculando una secuencia de lugares vacíos o huecos de dispersión.



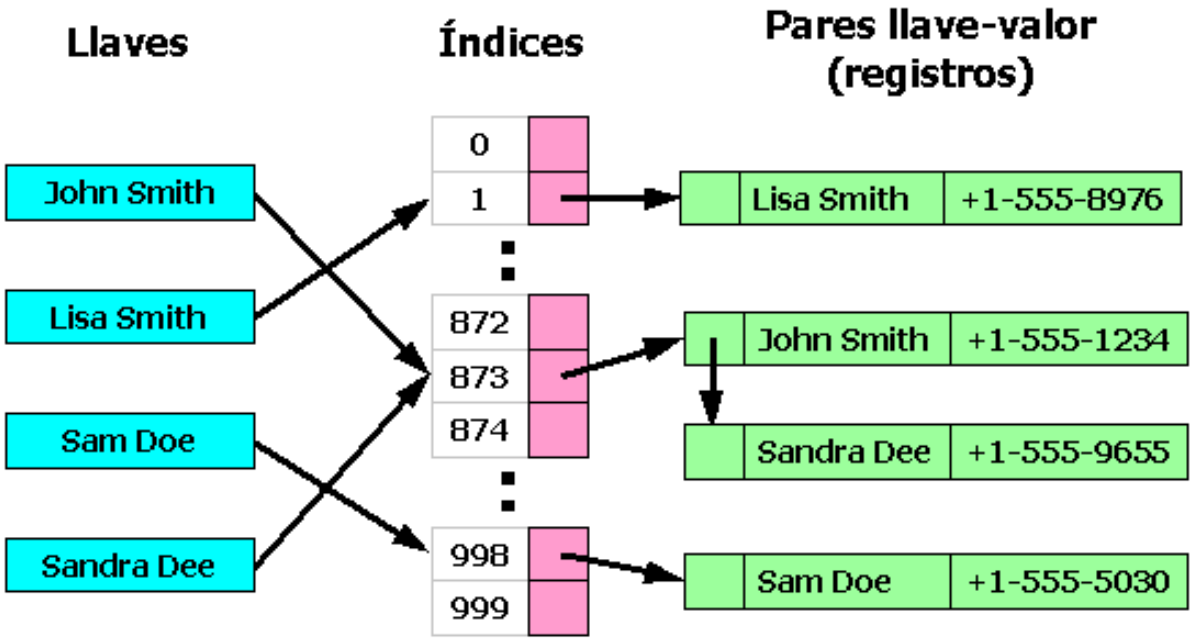
Visión gráfica (hashing cerrado)

- Desde un “gran” Universo sólo un número reducido de claves serán consideradas.

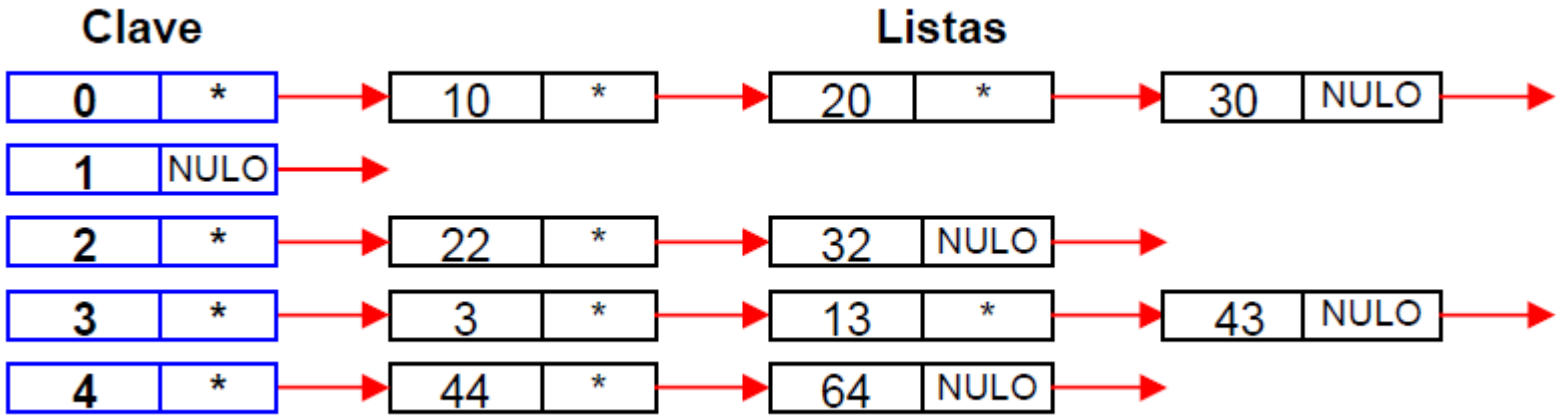


Tablas hash abiertas, de direccionamiento cerrado o encadenamiento separado

- Una de las estrategias más simples de resolución de colisiones es la **dispersión abierta o encadenamiento separado**, consiste en tener una **lista** de todos los elementos que se dispersan en el mismo valor (**colisionan**). Así, para buscar un valor, se encuentra su valor de dispersión y luego se recorre la lista. Para insertar, se hace lo mismo, y se inserta en la lista adecuada.

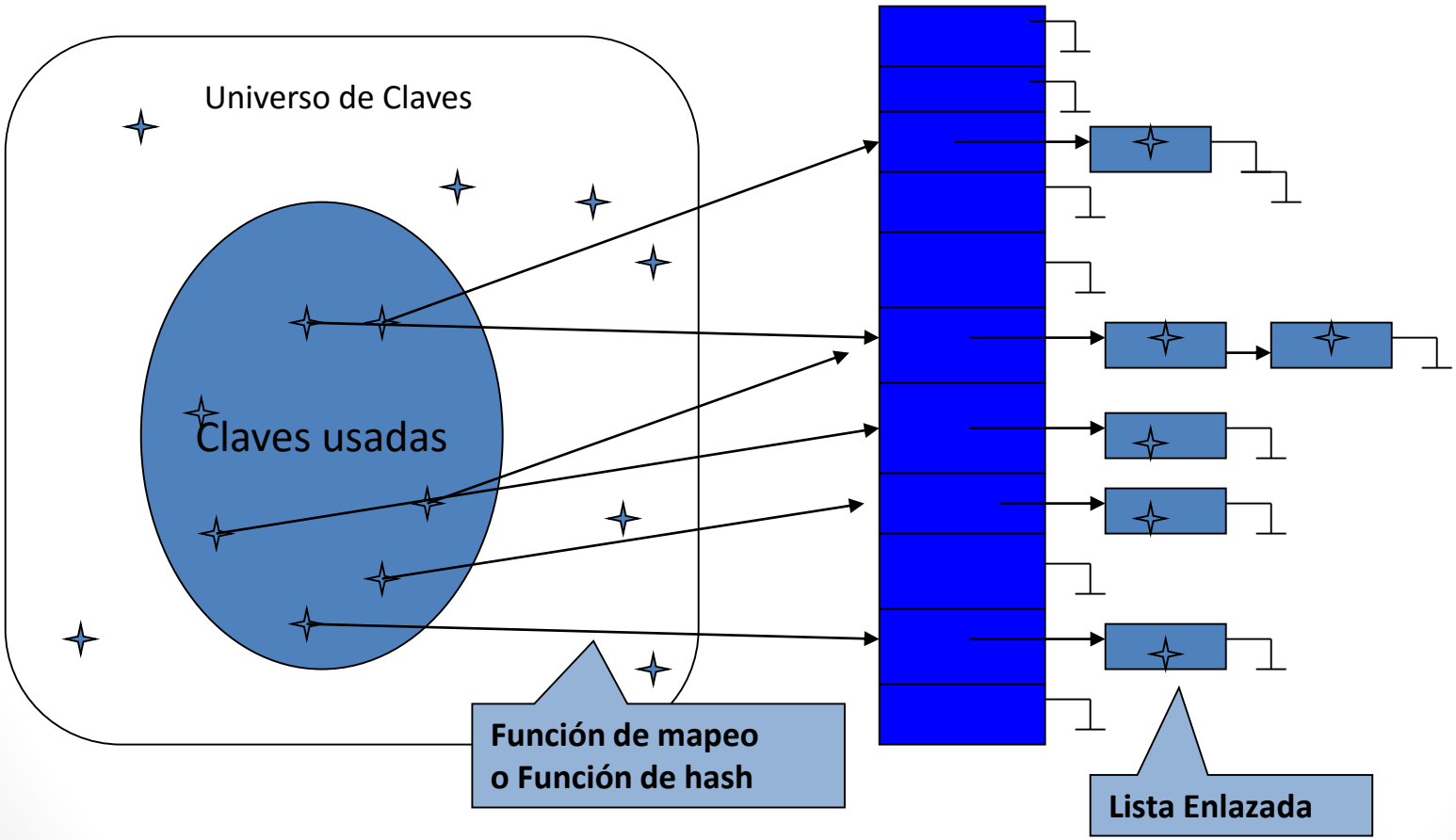


- En al **figura**, se observa un **tabla hash**, en donde se tienen **5** valores para la tabla, en donde se ha empleado **clave % tamaño,** *(i.e. clave % 5)* para asignar los valores a las listas.



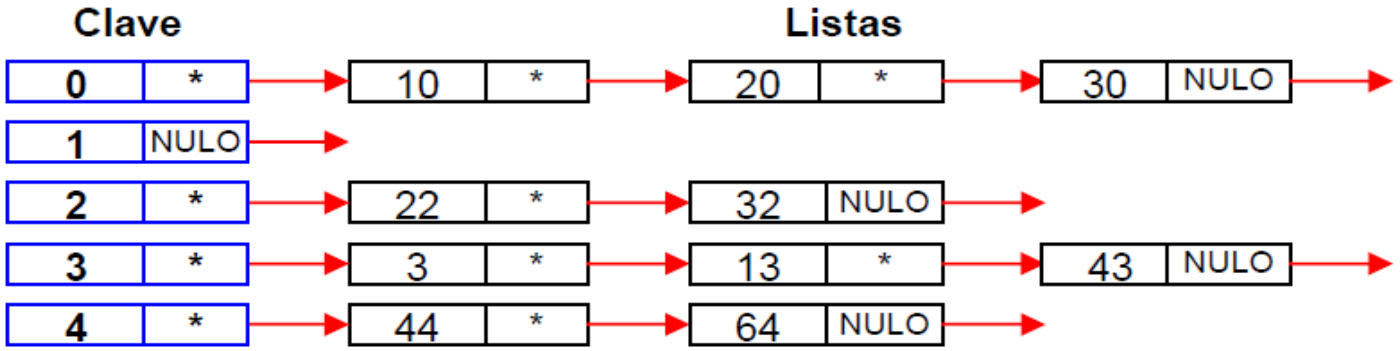
Visión gráfica (hashing abierto)

- Desde un “gran” Universo sólo un número reducido de claves serán consideradas.



- Si las listas se utilizan de forma no ordenada, el tiempo para realizar la inserción es **O(1)**.

- Los tiempos de ejecución en el peor caso son **O(n)**, con **n** como el número de elementos que se dispersan en la misma lista, sin embargo, si se supone una dispersión uniforme, los tiempos se reducen a **O(n/m)**, para las operaciones buscar y borrar, donde **m** es el tamaño de la tabla.



Funciones de dispersión para una tabla hash

- **Método de la división**
  - Son aquellas funciones de dispersión o hash que se generan calculando una **división**, p.g. **k mod m**.
  - Si los contenidos de entrada son enteros, se acepta como una buena estrategia la función **clave = contenido % tamaño**, es decir, el **modulo** (%) entre el **contenido** y el **tamaño** de la **tabla hash**. Sin embargo se puede presentar una mala elección del tamaño, por ejemplo si todos los contenidos acaban con cero y se elige un **tamaño de 10**, entonces la dispersión estándar es una mala elección.
    - Se ha encontrado que una buena **elección para el tamaño**, es regularmente un **número primo**, para un conjunto de contenidos aleatorios, **las llaves generadas con un número primo** son muy uniformes.

Método de la multiplicación

- Se generan valores para la dispersión en dos pasos.
  - Primero se calcula la parte decimal del producto de **k** y cierta constante real **A**, donde **0 < A < 1**.
  - Este resultado es entonces multiplicado por **m** antes de aplicarle la función de truncado, para obtener el valor de la dispersión.
    - **clave = m (contenido * A – contenido * A)**, Donde **(contenido * A – contenido * A)**, obtiene la parte decimal del número real. Como la parte decimal es mayor que cero, los valores de la dispersión son enteros positivos en el rango entre **0**, **1**, **2**, .., **m -1**.
- Una elección para elegir **A**, con la cual se obtiene una buena dispersión es la **razón Áurea**: **A = Φ = 1.61803399**

- **Método de la suma**
  - Si los contenidos son cadenas de caracteres, se pueden sumar los valores de los caracteres **ASCII**, y declarar un **índice** para cada uno de esos valores, de la forma, **clave = tamaño_suma_cadena**. Este método proporciona un valor que puede después utilizar el método de la división o multiplicación para ajustarse al tamaño de la tabal.
    - Aunque está función es fácil de implementar, las claves no siempre se e de claves.

H	o	l	a	M	u	n	d	o	!
@	s	e	c	p	r	e	@	s	e

72	111	108	97	77	117	110	100	111	33
64	115	101	99	112	114	101	64	115	101
136	226	209	196	189	231	211	164	226	134

Funciones de Hash H(k,i)

- Para la resolución y mejora de las colisiones en tablas hash existen al menos dos formas para encontrar una reasignación de los elementos pudiendo usar una: prueba lineal o el doble hashing.
- **Prueba lineal**
  - La función es: H(k,i) = (H'(k) +i) mod m, donde i es un índice que indica el número de colisiones.
  - Una desventaja de este método es la tendencia a crear largas secuencias de entradas ocupadas, incrementando el tiempo de inserción y búsqueda.
- **Doble hashing**
  - La función es: H(k,i) = (h1(k) + i*h2(k)) mod m, donde i es un índice que indica el número de colisiones.
  - Por ejemplo:
  - h1 = k mod m
  - h2 = 1 + (k mod (m-1))

Ejemplo de hashing H(k,i) cerrado

- **Sea $h(k,i) = (h_1(k) + i*h_2(k)) \bmod 13$**; con
  $h_1 = k \bmod 13$
  $h_2 = 1 + (k \bmod 11)$

h(79,0) = 1
h(72,0) = 7
h(98,0) = 7
h(98,1) = (7+11) mod 13 =5
h(14,0) = 1
h(14,1) = (1+4) mod 13 = 5
h(14,2) = (1+2*4) mod 13 = 9