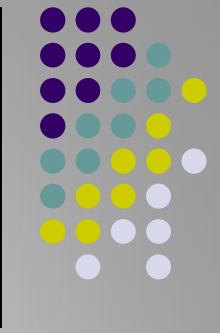
Tablas Hash

(Definición)



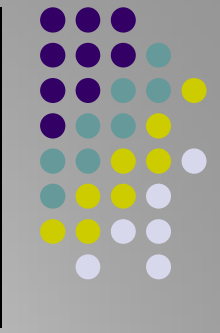
Una tabla «hash» es un conjunto finito de posiciones direccionables a través de una función que se denomina función «hash».

Dada una clave «x» determinada, la función «hash» hace que esta clave sea asignada a una dirección valida de la tabla.

FH(X) → Dirección valida de la Tabla

Tablas Hash

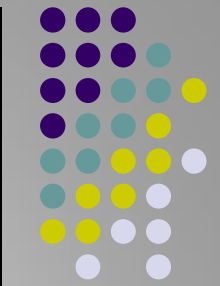
(Características)



- Se pueden decir que:
  - La tabla hash debe cubrir el universo posible de valores que pueda determinar la función «hash». Esto es un «hash estático»
  - La función «hash» debe ser lo mas simple y efectiva posible.
  - La función «hash» debe desperdiciar el mínimo posible de posiciones de la tabla.
  - Cada posición de la tabla puede ser una estructura de datos o un dato simple dependiendo el problema.

Tablas Hash

(Características de búsqueda)



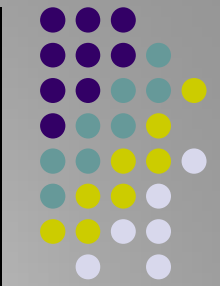
Es un proceso por el cual **un ítem** de datos es **acomodado** en una estructura basándose en una **transformación de su clave**, tal acción, posibilita que de forma eficiente **O(1)** puedan buscarse y recuperarse ítems de datos utilizando su clave.

"La idea básica es utilizar la clave para determinar la dirección del elemento, pero para no desperdiciar tanto espacio, hay que realizar una transformación mediante una función hash del conjunto K de claves sobre el conjunto L de direcciones de memoria"

h (K) ----> L

Tablas Hash

(Claves Sinónimas)



Es posible que dos claves diferentes K1 y K2 den la misma dirección. Entonces se dice que:

K1 y K2 son claves sinónimas

Entonces se produce el fenómeno de la colisión, y se debe de usar algún método para resolverlo.

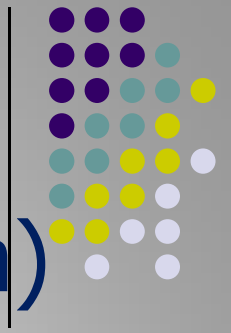
H(K1) -> Ln

H(K2) -> Ln

«Ambas claves diferentes K1 <> K2 pasadas por la función hash retornar la misma posición de la tabla»

Tablas Hash

(Categorización de Funciones Hash)



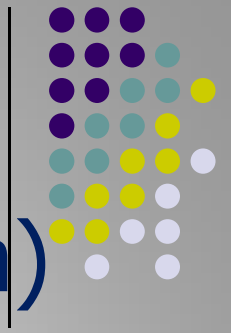
Perfectas Cada clave se convierte en un único entero (no hay claves sinónimas) (función biyectiva).

Imperfectas Muchas claves se convierten en un único entero (hay claves sinónimas).

Mínimas Perfectas Además de ser perfecta la función, el juego de n claves es convertido en un juego de enteros {0, 1, 2, ...n-1}. (no hay claves sinónimas y además no existen elementos libres en el espacio de direccionamiento)

Tablas Hash

(Categorización de Funciones Hash)



Existen tres componentes involucrados en la técnica de hashing:

* La tabla hash
* La función de hash o dispersión
* Colisiones

Espacio de almacenamiento en memoria principal o en almacenamiento secundario es la «tabla hash»

La relación entre la **CLAVE DE BUSQUEDA** y la posición en la tabla es lo que se llama ***función de dispersión***.

Tablas Hash

(Ejemplo)

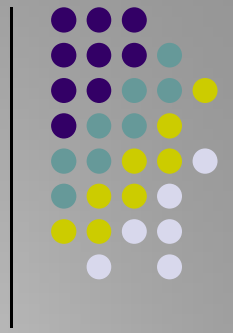
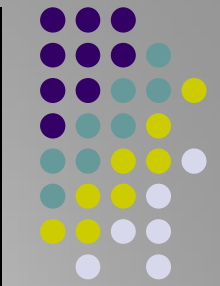


Tabla Hash

0	Volvo
1	
2	Ferrari
3	
4	
5	BMW
6	
7	
8	
9	Jaguar

Tablas Hash

(Función Hash por modulo)



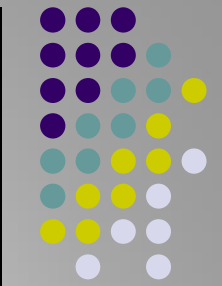
Es la correspondencia entre la clave y un índice de la tabla; y en general, cuando las claves son números enteros la ***función de dispersión*** toma la forma:

h(clave) = clave MODULO máxima_Q_claves

El tamaño de la tabla debe ser un ***número primo*** cercano a la cantidad de claves a mapear. De esta forma, y si las claves son números aleatorios, esta función suele distribuir las claves uniformemente.

Tablas Hash

(F.Hash por modulo - Ejemplo)



Ejemplo, suponga que se posee un conjunto de legajos de alumnos de alrededor de 1000 elementos (claves). La elección de M en este caso será 997 (se mapearán sobre una tabla de índices entre 0 y 996), que es el numero primo más próximo. Se aplica la función hash a los alumnos cuyo número es: «245643, 245981, 257135»

y se obtienen las direcciones (posición de la tabla)

h (245643) = 245643 mod 997 = 381

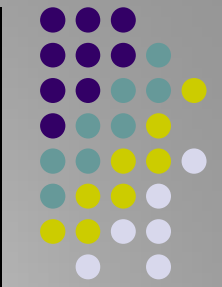
h (245981) = 245981 mod 997 = 719

h (257135) = 257135 mod 997 = 906

Cuando las claves son **cadenas de caracteres** lo usual es sumar los valores ASCII de los caracteres de la cadena o armar una secuencia.

Tablas Hash

(Función Hash por Plegamiento)



La técnica consiste en partir la clave K en varias partes K1, K2, K3,..., Kn, y la combinación de las partes de un modo conveniente da como resultado la dirección a almacenar la clave en la tabla.

h (x) = k1 + K2 + K3 + ... + Kr

Aplicando la función hash de plegamiento a los alumnos cuyos números son: «245643, 245981, 257135» se obtienen estas direcciones:

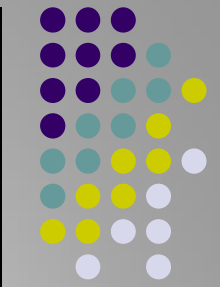
h (245643) = 245 + 643 = 888

h (245981) = 245 + 981 = 1226 = 226 (se ignora el acarreo 1)

h (257135) = 257 + 135 = 392

Tablas Hash

(F. Hash por Mitad del Cuadrado)



Función hash que consiste en calcular el cuadrado de la clave K y la dirección del registro viene representada por los dígitos de K al cuadrado que ocupan cierta posición.

Ejemplo, claves a mapear: 245643, 245981, 257135

(245643)^2 --> 60340**483**449 tomando los dígitos cuarto, quinto y sexto por la derecha

h (245643) = **483**

(245981)^2 --> 60506**652**361

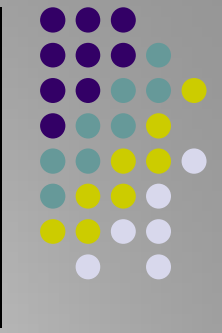
h (245981) = **652**

(257135)^2 --> 68118**408**225

h (257135) = **408**

Tablas Hash

(Colisiones)



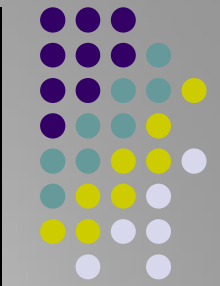
Ocurren cuando dos elementos distintos se dispersan en el mismo valor, es decir que la función de dispersión devuelve un mismo valor para dos claves distintas (H(k1)=X y H(k2)=X).

Tratamiento de colisiones por

- **dispersión abierta** y
- **dispersión cerrada**

Tablas Hash

(Colisiones – Dispersión Abierta)



La *dispersión abierta* (o *encadenamiento separado)* consiste en asociar a la tabla una lista de claves que colisionan.

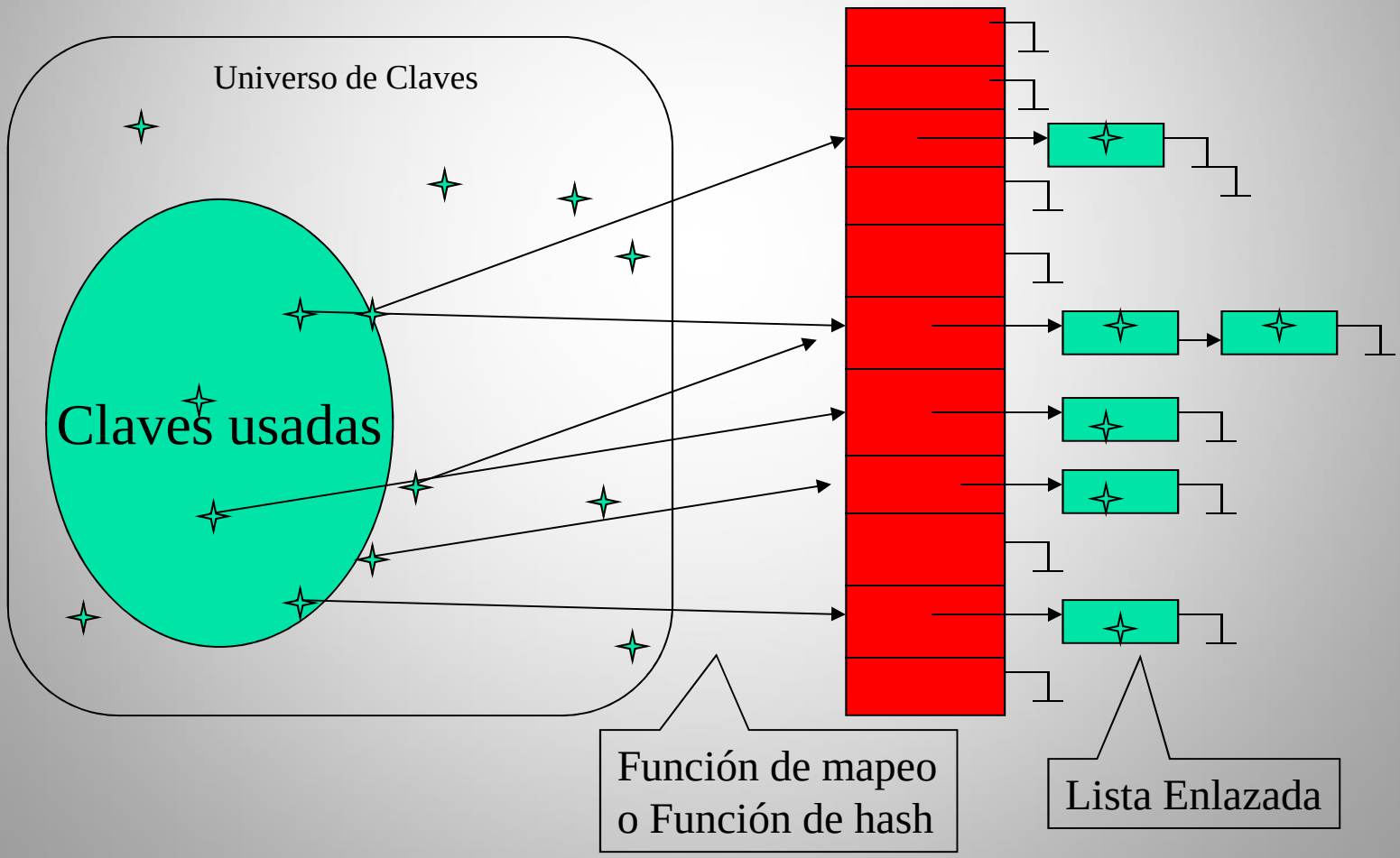
Suponiendo una tabla de tamaño 11 y con la función de dispersión:

$$h(k) = \left[\sum_{i=1}^{3} ASCII(Ki) \right] \bmod 11$$

Tablas Hash

(Colisiones – Dispersión Abierta)

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



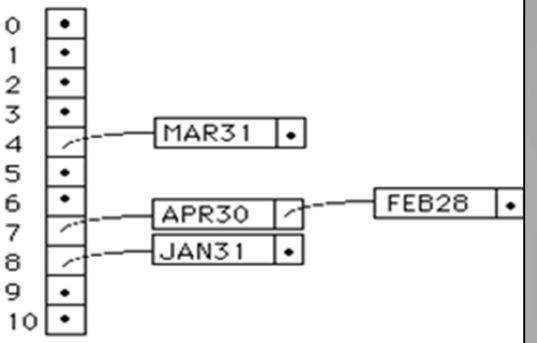
Tablas Hash
(Colisiones – Dispersión Abierta)

Suponiendo una tabla de tamaño 11 y con la función de dispersión:

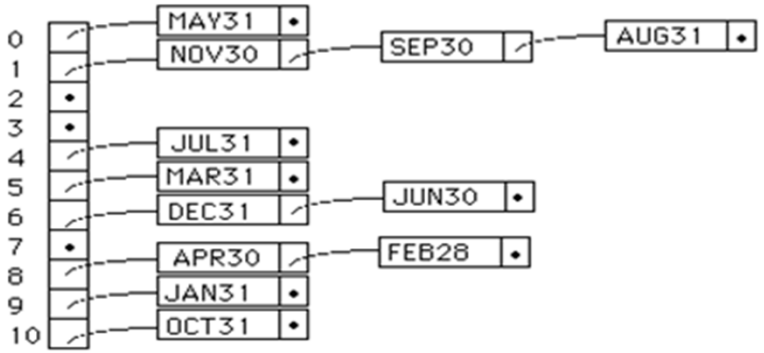
$$h(k) = \left[\sum_{i=1}^{3} ASCII(k_i) \right] \bmod 11$$

K	DAYS	a(K)	h(K)
JAN	31	217	8
FEB	28	205	7
MAR	31	224	4
APR	30	227	7
MAY	31	231	0
JUN	30	237	6
JUL	31	235	4
AUG	31	221	1
SEP	30	232	1
OCT	31	230	10
NOV	30	243	1
DEC	31	204	6

(a) Calendar Table

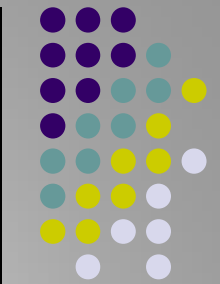


(b) Hash table. After inserting the first four records



(c) Hash table. After inserting all the records

Tablas Hash
(Disp. Abierta – Zona Overflow)

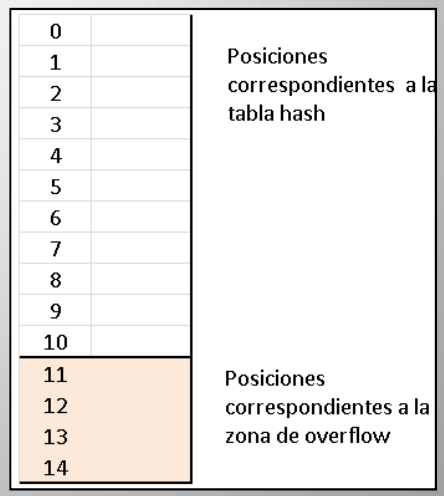


La «zona de overflow» es una lugar de la tabla destinada a guardar las claves que colisionan, pero esta parte de la tabla no es alcanzada por la función «hash».

También se suele usar para la «zona de overflow» una tabla aparte (separada) a la de la función «hash».

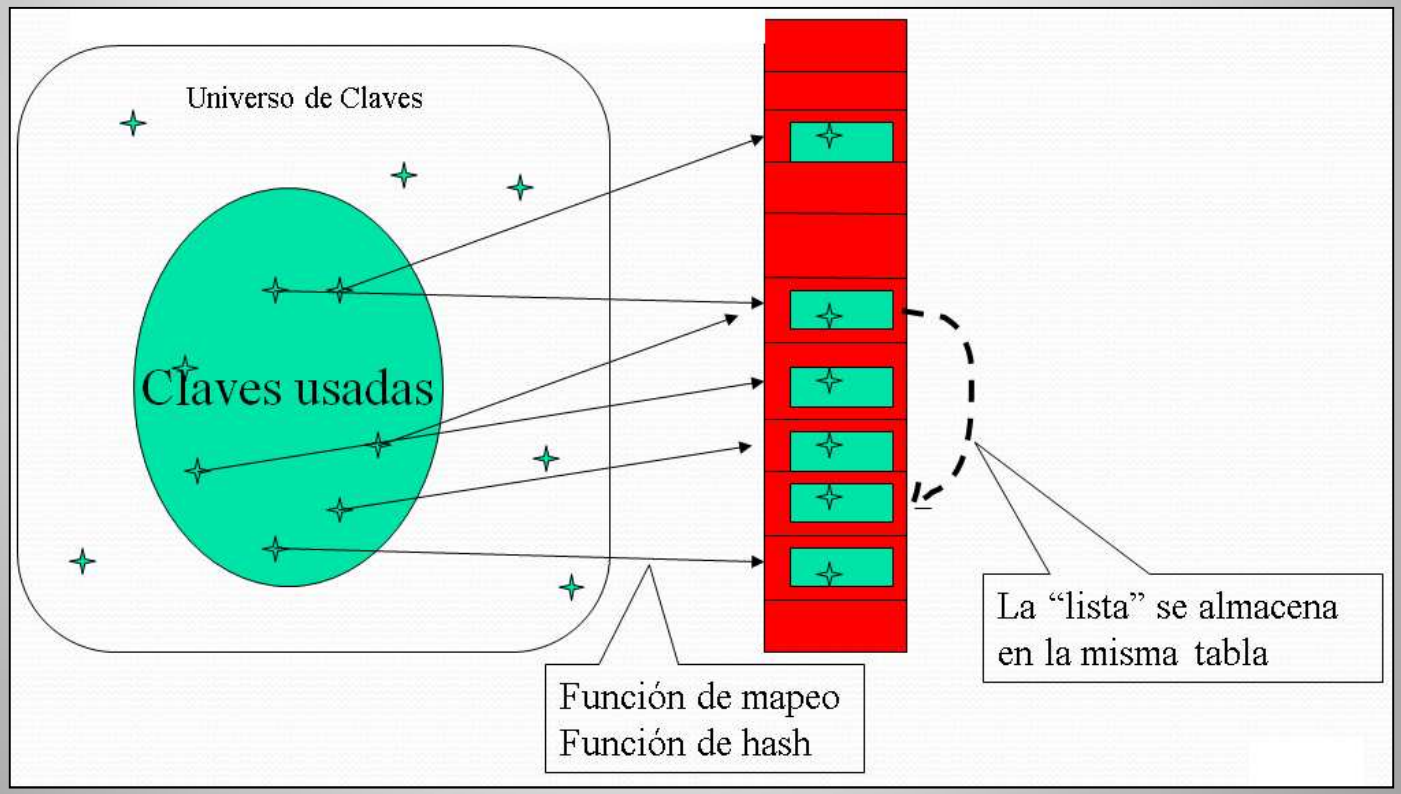
Si la función hash es «FH = clave modulo 11» se define una tabla con 15 posiciones donde las posiciones desde la «0» a la «10» son alcanzadas por la función «hash» y las posiciones de la 11 a la 14 quedan para la «zona de overflow»

En la «zona de overflow» las claves se pueden ubicar secuencialmente (inspección lineal) o usar algún otro método de ubicación de clave como puede ser una inspección cuadrática.



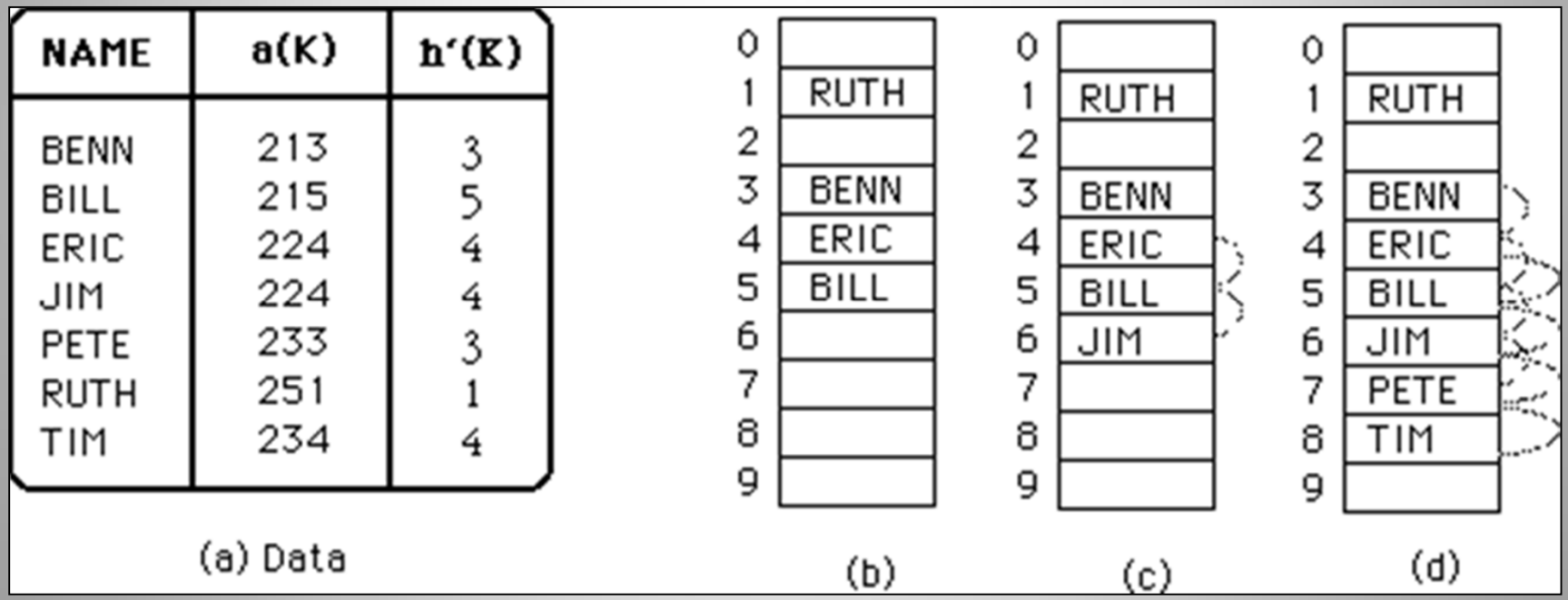
Tablas Hash
(Colisiones – Dispersión Cerrada)

Soluciona las colisiones buscando celdas alternativas hasta encontrar una vacía (dentro de la misma tabla).



Tablas Hash
(Disp.Cerrada – Exploración Lineal)

En caso de colisión se recorren las celdas en secuencia buscando una vacía; es decir, si hay una colisión se prueba en la celda siguiente y así sucesivamente hasta encontrar una vacía.



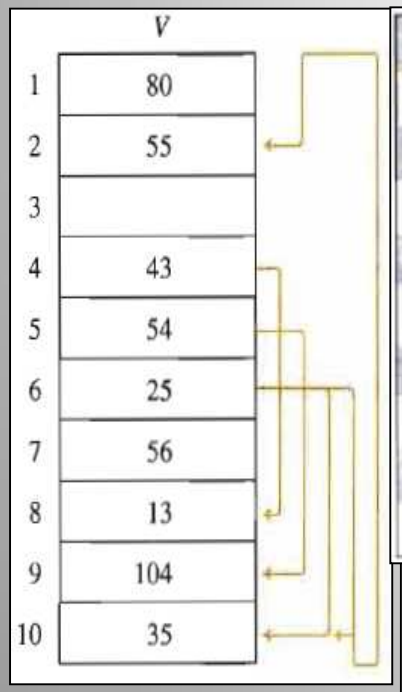
NAME	a(K)	h'(K)
BENN	213	3
BILL	215	5
ERIC	224	4
JIM	224	4
PETE	233	3
RUTH	251	1
TIM	234	4

(a) Data

	(b)	(c)	(d)
0			
1	RUTH	RUTH	RUTH
2			
3	BENN	BENN	BENN
4	ERIC	ERIC	ERIC
5	BILL	BILL	BILL
6		JIM	JIM
7			PETE
8			TIM
9			

Tablas Hash
(Disp.Cerrada – Exp. Cuadrática)

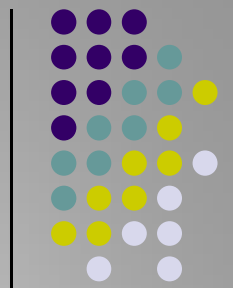
En caso de colisión se recorren las celdas usando los cuadrados a partir de la dirección que retorno la función hash. Si la función «hash» retorna la dirección «D» entonces se probara con la dirección D+1, luego D+4, D+9, D+16,…,D+i^2 y así sucesivamente.



K	H(K)
25	6
43	4
56	7
35	6
54	5
13	4
80	1
104	5
55	6

Al aplicar la función *hash* a la clave 35, se obtiene una dirección (*D*) igual a 6; sin embargo, en esa dirección no se encuentra el elemento buscado. Se calcula posteriormente *DX*, como la suma *D* + (*I* * *I*), obteniéndose de esta forma la dirección 7. El algoritmo de búsqueda concluye cuando se encuentra el valor deseado en la décima posición del arreglo.

Tablas Hash
(Disp.Cerrada – Exploración Doble)



Este método utiliza una segunda función hash para resolver una colisión. Suponga que al registro de clave K le corresponde la dirección h(x) = P, pero resulta que esta dirección esta ocupada por otro registro. Entonces con la segunda función h'(x) = P', la nueva dirección es P + P'; si también hubiera colisión, la siguiente seria P + 2P'; y así sucesivamente. Ejemplo:

$$h_1(k) = \left[\sum_{i=1}^{3} ASCII(K_i) \right] \bmod 13$$

$$h_2(k) = \left[\sum_{i=1}^{3} ASCII(K_i) \right] + 3j \bmod 13$$

K	DAYS	a(K)	h1(K)	h2(K) j	h2(K)
JAN	31	217	9		
FEB	28	205	10		
MAR	31	224	3		
APR	30	227	6		
MAY	31	231	10	5	12
JUN	30	237	3	4	2
JUL	31	235	1		
AUG	31	221	0		
SEP	30	232	11		
OCT	31	230	9	3	5
NOV	30	243	9	4	8
DEC	31	204	9	7	4

(a)

0	AUG31	
1	JUL31	
2	JUN30	← h1(JUN) = 3
3	MAR31	
4	DEC31	← h1(DEC) = 9
5	OCT31	← h1(OCT) = 9
6	APR30	
7		
8	NOV30	← h1(NOV) = 9
9	JAN31	
10	FEB28	
11	SEP30	
12	MAY31	← h1(MAY) = 10

(b)