DEPARTAMENTO DE
INGENIERÍA
INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

Análisis de Algoritmos y Estructura de Datos

TDA tabla hash

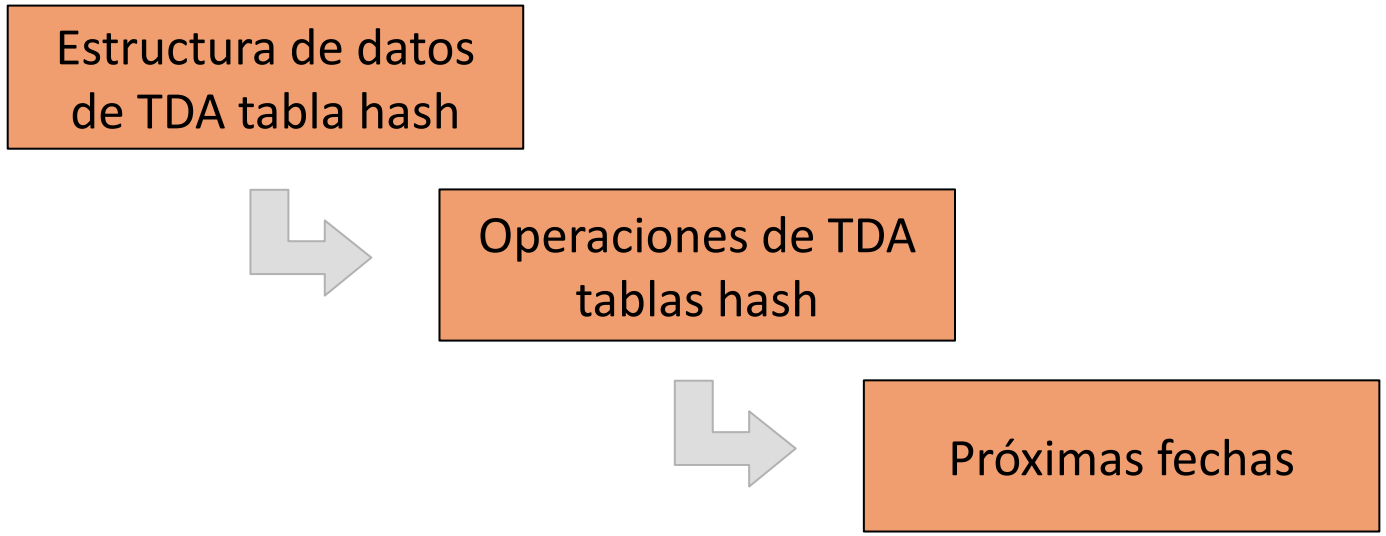
Prof. Violeta Chang C

Semestre 2 – 2023

TDA tabla hash

- **Contenidos**:
  - Estructura de datos TDA tabla hash
  - Operaciones de TDA tabla hash
- **Objetivos**:
  - Comprender estructura de datos de TDA tabla hash
  - Conocer y aplicar operaciones y estrategias de resolución de colisiones en tablas hash

Ruta de la sesión

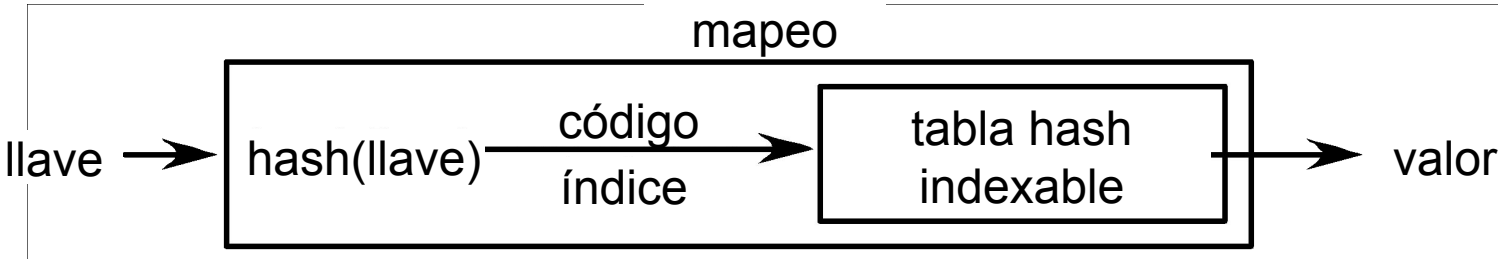


TDA tabla hash

- **Problema**: acceso indexado a una colección de datos es rápida solamente si se conoce el índice de lo que se busca. Sin el índice hay que buscar…
  - O(n) si la colección está desordenada
  - O(log n) si está ordenada
- **Idea**: Que pasaría si se podría usar la llave de búsqueda (lo que se busca) como un índice en la colección que nos lleve directamente al valor asociado?
  - Tiempo de búsqueda O(1)

TDA tabla HASH

- **Estructura de datos:**
  - Una **tabla hash** es un colección indexable, donde cada posición de la tabla se conoce como **bucket** (o celda) y puede tener una o más entradas
  - Una **función hash** convierte una llave de entrada en un índice que es usado para acceder a un bucket en la tabla hash y debe ser *determinística*, *eficiente* y *uniforme*
- **Operaciones:**
  - **buscar(T,dato)**
  - **insertar(T,dato)**
  - **eliminar(T,dato)**



TDA tabla HASH

- **Lo bueno**: si cada llave mapea a una única posición en la tabla hash, la mayoría de las operaciones (inserción, eliminación y búsqueda) será de tiempo constante O(1)
- **Lo malo**: para garantizar O(1), la tabla debe tener tantos buckets como posibles llaves hayan
  - Como el rango de las llaves puede ser muy largo (por ejemplo, los 8 dígitos del RUT o los 9 dígitos de los números de celular), esto no es práctico
- **Idea**: trabajar con una aproximación a lo ideal
  - Permitir que una tabla hash tenga *m* buckets (*m*<<rango de llaves)
  - Permitir que la función hash mapee más de una llave al mismo bucket → **colisión**
  - Se necesitan estrategias para lidiar eficientemente con las colisiones

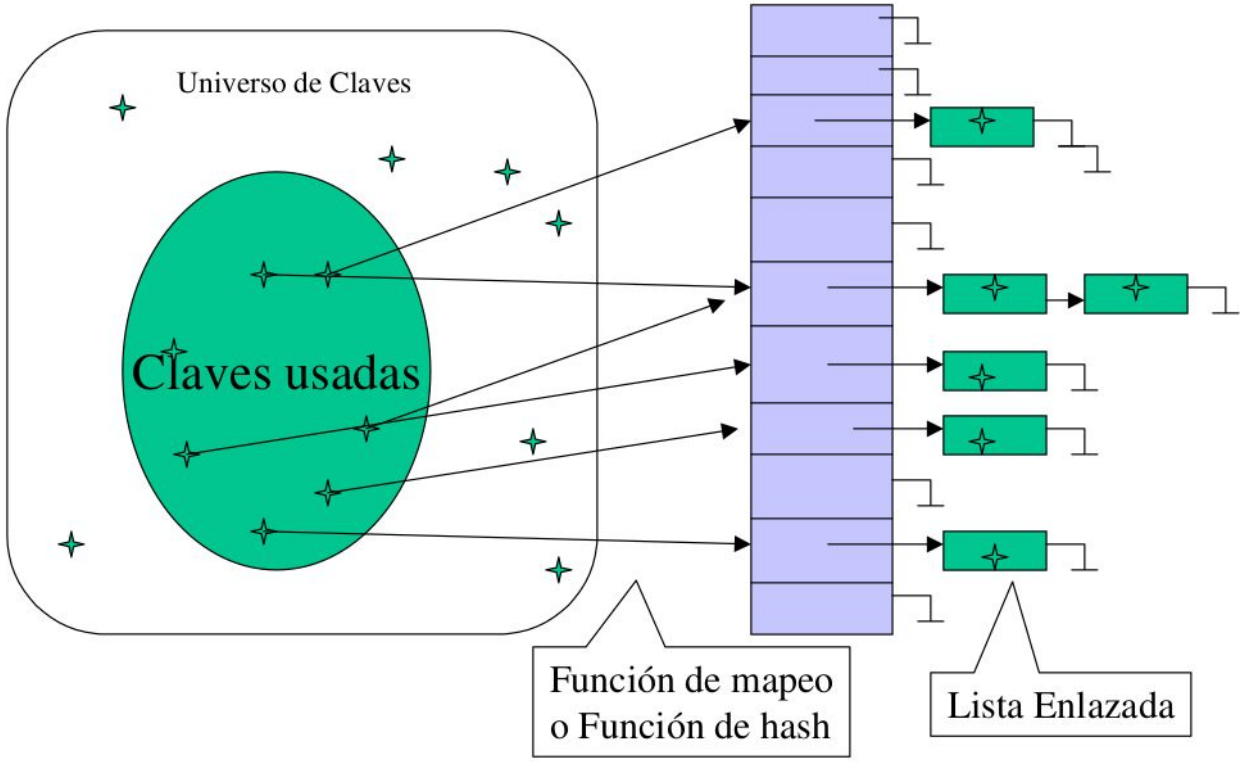
Resolución de colisiones

- **Hashing abierto - Encadenamiento**
  - Cada entrada debe ser almacenada en la posición (bucket) a la cual fue mapeada originalmente
  - La estrategia consiste en permitir que la capacidad de cada bucket sea mayor que 1. Por lo tanto, cada bucket es una colección (ojalá pequeña) por sí mismo
  - Ejemplo de estrategia: **encadenamiento**

Resolución de colisiones

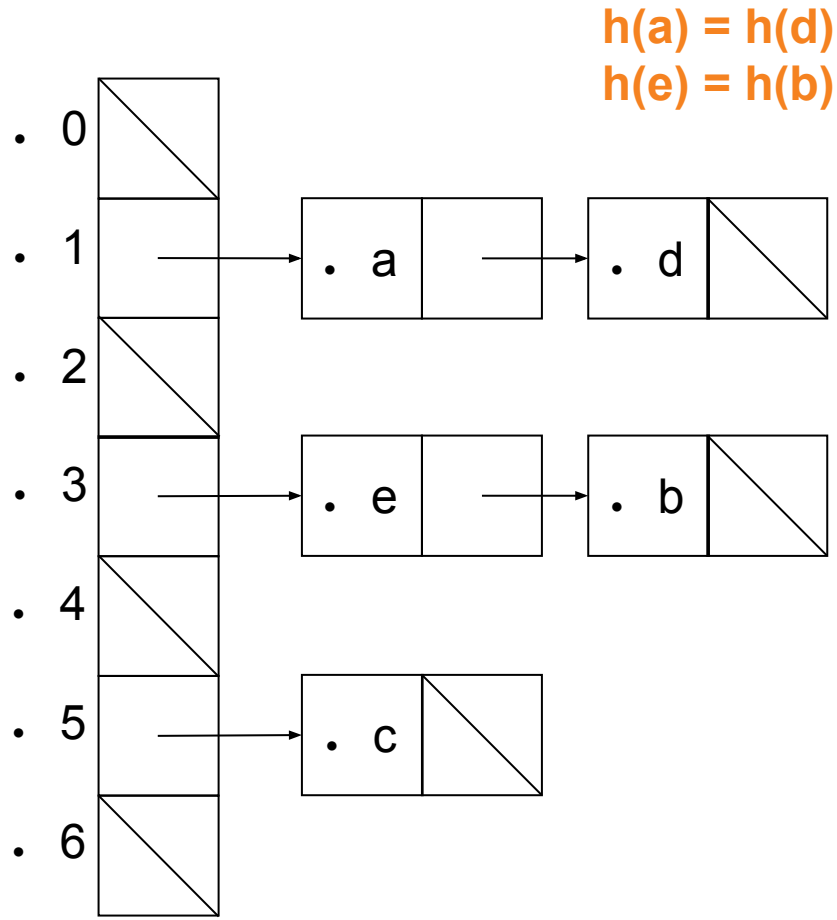
- **Hashing abierto - Encadenamiento**

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



Resolución de colisiones

- **Hashing abierto con encadenamiento**
  - Cada bucket apunta a una cadena de entradas
  - Generalmente se ocupa una lista enlazada simple (desordenada) para cada cadena
  - El desempeño decae con el largo de las cadenas

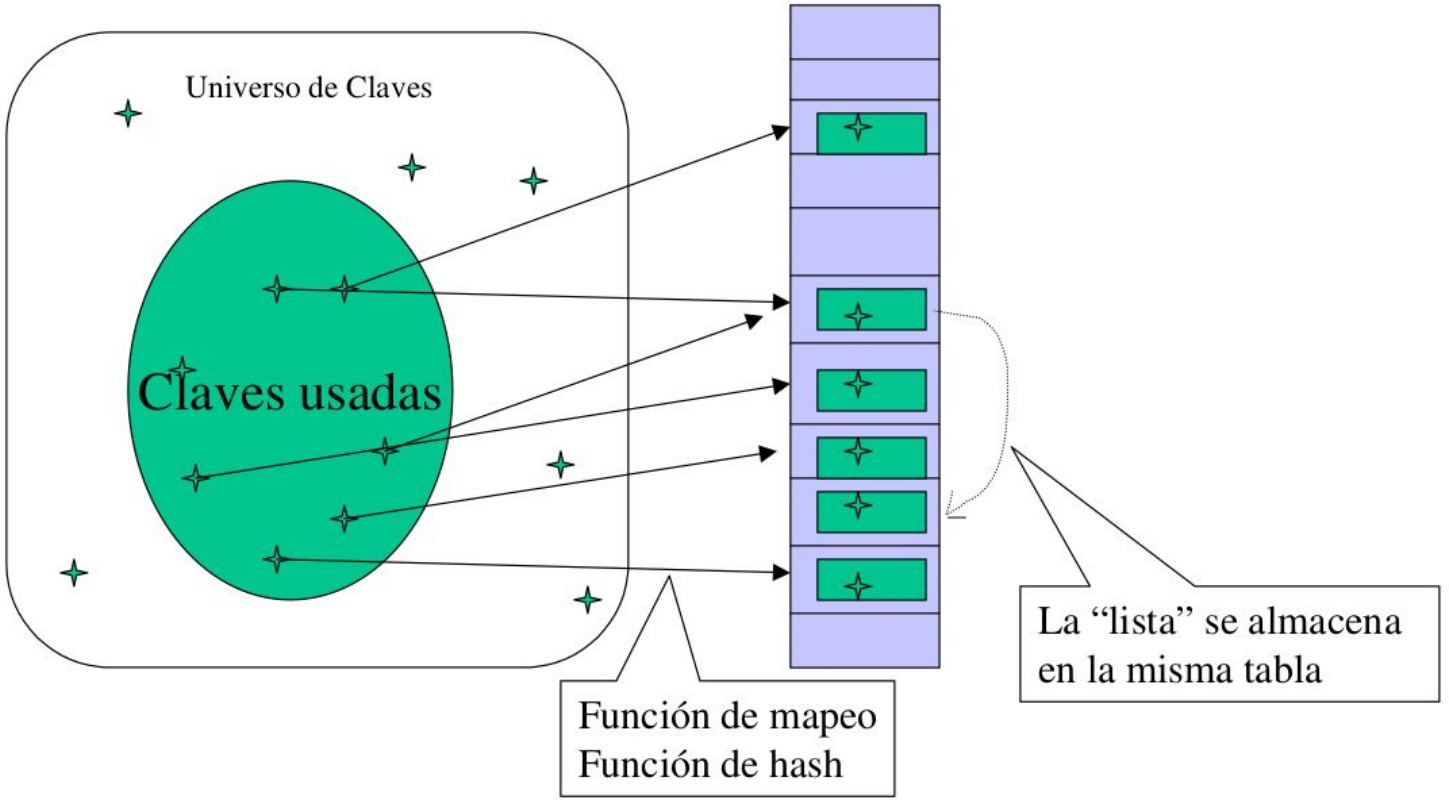


Resolución de colisiones

- **Hashing cerrado**
  - Cada bucket contiene una única entrada
  - Cuando ocurre una colisión, se debe encontrar un bucket en otra posición en la tabla
  - Por lo tanto, se permite que la entrada sea almacenada en un bucket distinto al que fue mapeado originalmente (su índice hash primario)
  - Ejemplo de estrategia: **prueba lineal**

Resolución de colisiones

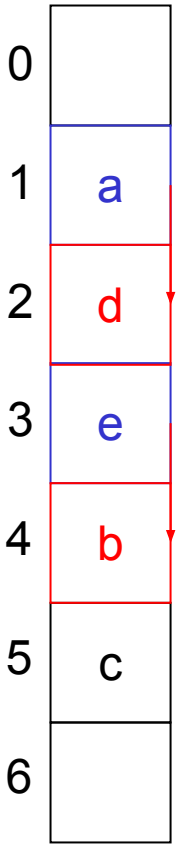
- **Hashing cerrado**
  - Desde un "gran" Universo sólo un número reducido de claves serán consideradas.



Resolución de colisiones

- **Hashing cerrado con prueba lineal**
  - **Idea:** ocurre una colisión en posición *i* y se debe encontrar un nuevo bucket para la entrada
  - Tratar con la posición (i+1), si está ocupada tratar con (i+2) y así sucesivamente
  - **OBS:** Wrapping de la tabla
  - El desempeño decae con la dificultad de encontrar el bucket disponible

h(a) = h(d)
h(e) = h(b)



Inserción de elemento

- **Ejemplo**: Considerando la función
  - *hash*(x)= *ascii*(x) *mod* 11

Insertar los siguientes elementos en una tabla hash usando **prueba lineal** como estrategia de resolución de colisiones

B O E P V L X N K M

Inserción de elemento

*hash(*key*)= ascii(*key*) mod* 11

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

0
2
3
3
9
10
0
1
9
0

M = 11

	KEY	VAL
0	B	0
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		

Inserción de elemento

insertar(B,0)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

	KEY	VAL
0	B	0
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		

Inserción de elemento

insertar(O,1)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

	KEY	VAL
0	B	0
1		
2	O	1
3		
4		
5		
6		
7		
8		
9		
10		

Inserción de elemento

insertar(E,2)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

	KEY	VAL
0	B	0
1		
2	O	1
3	E	2
4		
5		
6		
7		
8		
9		
10		

Inserción de elemento

insertar(P,3)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

Primer bucket vacío?

	KEY	VAL
0	B	0
1		
2	O	1
3	E	2
4	P	3
5		
6		
7		
8		
9		
10		

Inserción de elemento

insertar(V,4)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

	KEY	VAL
0	B	0
1		
2	O	1
3	E	2
4	P	3
5		
6		
7		
8		
9	V	4
10		

Inserción de elemento

insertar(L,5)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

	KEY	VAL
0	B	0
1		
2	O	1
3	E	2
4	P	3
5		
6		
7		
8		
9	V	4
10	L	5

Inserción de elemento

insertar(X,6)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

Primer bucket vacío?

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5		
6		
7		
8		
9	V	4
10	L	5

Inserción de elemento

insertar(N,7)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

Primer bucket vacío?

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6		
7		
8		
9	V	4
10	L	5

Inserción de elemento

insertar(K,8)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

Primer bucket vacío?

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	K	8
7		
8		
9	V	4
10	L	5

Inserción de elemento

insertar(M,9)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

Primer bucket vacío?

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	K	8
7	M	9
8		
9	V	4
10	L	5

Inserción de elemento

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	K	8
7	M	9
8		
9	V	4
10	L	5

Búsqueda de elemento

- **Ejemplo**: Considerando la función

$$hash(x) = ascii(x) \bmod 11$$

Y la tabla hash creada con esa función con los siguientes elementos insertados en ella usando **prueba lineal** como estrategia de resolución de colisiones:

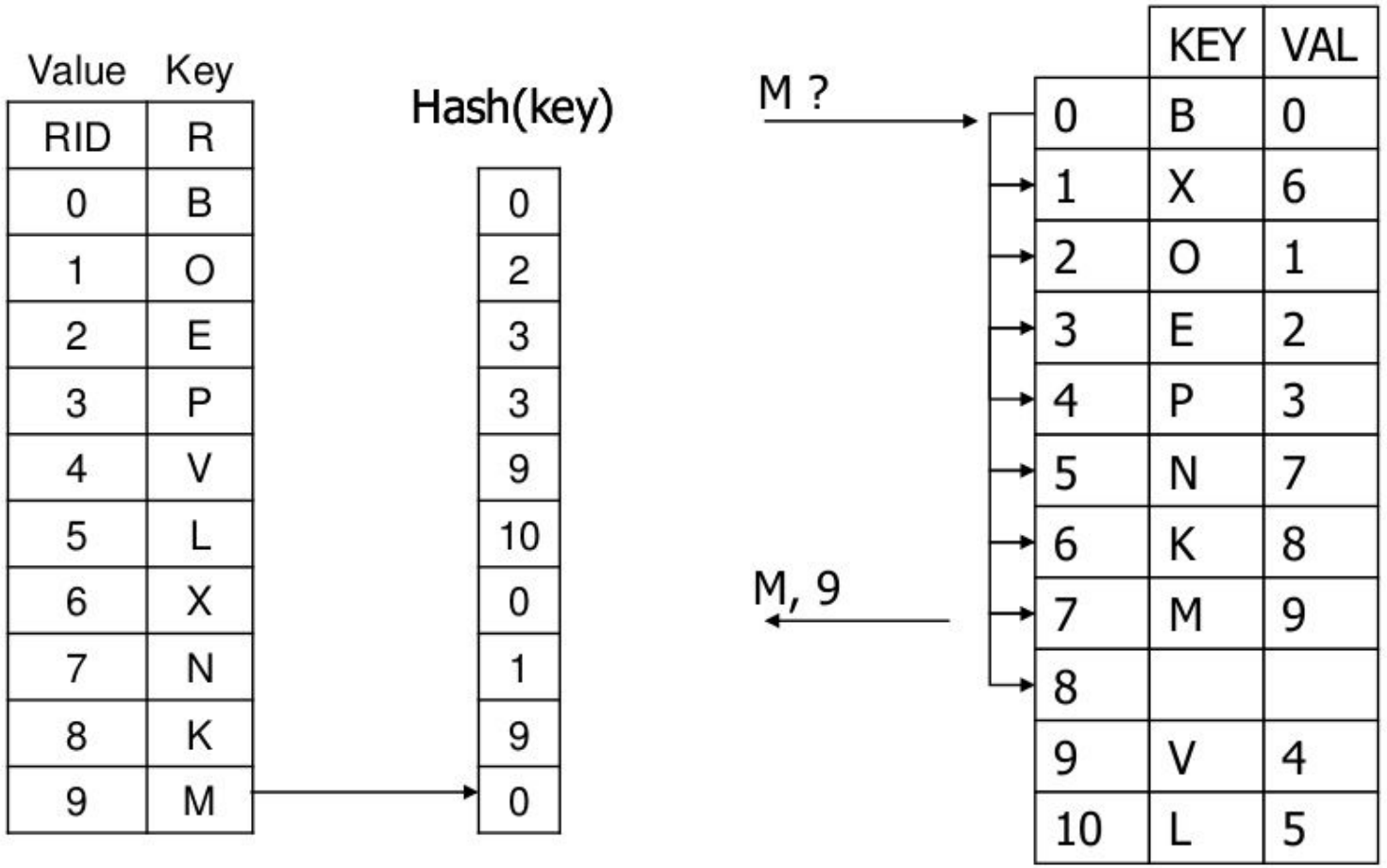
B	O	E	P	V	L	X	N	K	M

Buscar los siguientes elementos:

M	K	Y

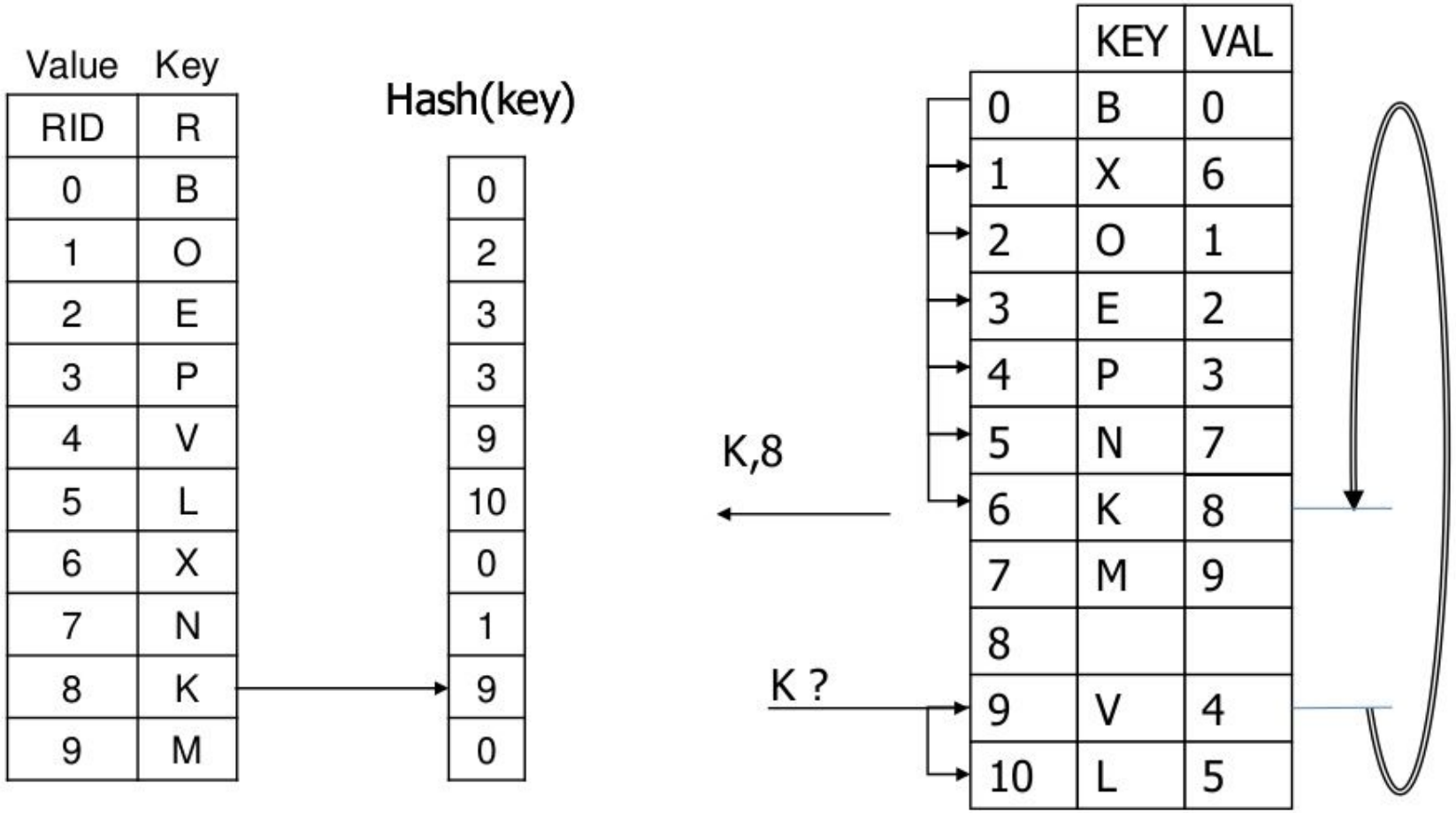
Búsqueda de elemento

buscar(M)



Búsqueda de elemento

buscar(K)



Búsqueda de elemento

ascii(Y) mod 11 = 89 mod 11 = 1

buscar(Y)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

Y ?

Vacío

Fin

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	K	8
7	M	9
8		
9	V	4
10	L	5

Devolver (-1)

Eliminación de elemento

- **Ejemplo**: Considerando la función

$$hash(x) = ascii(x) \bmod 11$$

Y la tabla hash creada con esa función con los siguientes elementos insertados en ella usando **prueba lineal** como estrategia de resolución de colisiones:

B	O	E	P	V	L	X	N	K	M

Eliminar los siguientes elementos:

M	N	L

Eliminación de elemento

eliminar(M)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

K(0) != M

Buscar M

En blanco

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	K	8
7	M	9
8		
9	V	4
10	L	5

Eliminación de elemento

eliminar(M)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

K(0) != M

Buscar M

En blanco

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	K	8
7		
8		
9	V	4
10	L	5

Eliminación de elemento

eliminar(N)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

K(1) != N

Buscar N

En blanco y rehash

Fin del cluster

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	K	8
7	M	9
8		
9	V	4
10	L	5

Eliminación de elemento

eliminar(N)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

K(1) != N

Buscar N

En blanco y rehash

Fin del cluster

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5		
6	K	8
7	M	9
8		
9	V	4
10	L	5

EOf cluster

Eliminación de elemento

eliminar(N)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

K(1) != N

Buscar N

En blanco y rehash

Fin del cluster

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5		
6	K	8
7	M	9
8		
9	V	4
10	L	5

EOf cluster

enBlanco Key(6), Val(6)
insertar(K,8)

Eliminación de elemento

eliminar(N)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

K(1) != N

Buscar N

En blanco y rehash

Fin del cluster

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5		
6		
7	M	9
8		
9	V	4
10	L	5

EOf cluster

→ enBlanco Key(6), Val(6)
insertar(K,8)

Eliminación de elemento

eliminar(N)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

K(1) != N

Buscar N

En blanco y rehash

Fin del cluster

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5		
6		
7	M	9
8		
9	V	4
10	L	5

enBlanco Key(6), Val(6)
insertar(K,8)

Eliminación de elemento

eliminar(N)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

K(1) != N

Buscar N

En blanco y rehash

Fin del cluster

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	K	8
6		
7	M	9
8		
9	V	4
10	L	5

enBlanco Key(6), Val(6)
insertar(K,8)

Eliminación de elemento

eliminar(N)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

K(1) != N

Buscar N

En blanco y rehash

Fin del cluster

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	K	8
6		
7		
8		
9	V	4
10	L	5

→ enBlanco Key(7), Val(7)
insertar(M,9)

Eliminación de elemento

eliminar(N)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

K(1) != N

Buscar N

En blanco y rehash

Fin del cluster

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	K	8
6		
7		
8		
9	V	4
10	L	5

Primer bucket vacío?

enBlanco Key(7), Val(7)
insertar(M,9)

Eliminación de elemento

eliminar(N)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

K(1) != N

Buscar N

En blanco y rehash

Fin del cluster

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	K	8
6	M	9
7		
8		
9	V	4
10	L	5

Primer bucket vacío?

enBlanco Key(7), Val(7)
insertar(M,9)

Eliminación de elemento

eliminar(L)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	K	8
7	M	9
8		
9	V	4
10	L	5

En blanco y rehash
Fin del cluster

K(10) = L ⟶ poner en blanco

Eliminación de elemento

eliminar(L)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	K	8
7	M	9
8		
9	V	4
10		

En blanco y rehash
Fin del cluster

K(10) = L ⟶ poner en blanco

Eliminación de elemento

eliminar(L)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	K	8
7	M	9
8		
9	V	4
10		

EOf cluster

En blanco y rehash
Fin del cluster

K(10) = L ⟶ poner en blanco

Eliminación de elemento

eliminar(L)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	K	8
7	M	9
8		
9	V	4
10		

EOf cluster

En blanco y rehash
Fin del cluster

enBlanco Key(0), Val(0)
insertar(B,0)

Eliminación de elemento

eliminar(L)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	K	8
7	M	9
8		
9	V	4
10		

EOf cluster

En blanco y rehash
Fin del cluster

enBlanco Key(1), Val(1)
insertar(X,6)

Eliminación de elemento

eliminar(L)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	K	8
7	M	9
8		
9	V	4
10		

EOf cluster

En blanco y rehash
Fin del cluster

enBlanco Key(2), Val(2)
insertar(O,1)

Eliminación de elemento

eliminar(L)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	K	8
7	M	9
8		
9	V	4
10		

EOf cluster

En blanco y rehash
Fin del cluster

enBlanco Key(3), Val(3)
insertar(E,2)

Eliminación de elemento

eliminar(L)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	K	8
7	M	9
8		
9	V	4
10		

EOf cluster

En blanco y rehash
Fin del cluster

enBlanco Key(4), Val(4)
insertar(P,3)

Eliminación de elemento

eliminar(L)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	K	8
7	M	9
8		
9	V	4
10		

EOf cluster

En blanco y rehash
Fin del cluster

enBlanco Key(5), Val(5)
insertar(N,7)

Eliminación de elemento

eliminar(L)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	K	8
7	M	9
8		
9	V	4
10		

EOf cluster

En blanco y rehash
Fin del cluster

enBlanco Key(6), Val(6)
insertar(K,8)

Eliminación de elemento

eliminar(L)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6		
7	M	9
8		
9	V	4
10		

EOf cluster

En blanco y rehash
Fin del cluster

enBlanco Key(6), Val(6)
insertar(K,8)

Eliminación de elemento

eliminar(L)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6		
7	M	9
8		
9	V	4
10	K	8

EOf cluster

En blanco y rehash
Fin del cluster

enBlanco Key(6), Val(6)
insertar(K,8)

Eliminación de elemento

eliminar(L)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6		
7	M	9
8		
9	V	4
10	K	8

EOf cluster

En blanco y rehash
Fin del cluster

enBlanco Key(7), Val(7)
insertar(M,9)

Eliminación de elemento

eliminar(L)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	M	9
7		
8		
9	V	4
10	K	8

EOf cluster

En blanco y rehash
Fin del cluster

enBlanco Key(7), Val(7)
insertar(M,9)

Eliminación de elemento

eliminar(L)

Value	Key
RID	R
0	B
1	O
2	E
3	P
4	V
5	L
6	X
7	N
8	K
9	M

Hash(key)

0
2
3
3
9
10
0
1
9
0

	KEY	VAL
0	B	0
1	X	6
2	O	1
3	E	2
4	P	3
5	N	7
6	M	9
7		
8		
9	V	4
10	K	8

EOf cluster

Desempeño de tablas hash

- El **factor de carga** de una tabla hash se define como

L = número de buckets ocupados/número de buckets

- El **número de comparaciones** requeridas para encontrar un elemento en una tabla hash usando prueba lineal como estrategia de resolución de colisiones y un factor de carga L es

$$c = \frac{1}{2}\left(1 + \frac{1}{1-L}\right)$$

- El **número de comparaciones** requeridas para encontrar un elemento en una tabla hash usando encadenamiento como estrategia de resolución de colisiones y un factor de carga L es

$$c = 1 + \frac{L}{2}$$

Desempeño de tablas hash

L	Number of Probes with Linear Probing	Number of Probes with Chaining
0.0	1.00	1.00
0.25	1.17	1.13
0.5	1.50	1.25
0.75	2.50	1.38
0.85	3.83	1.43
0.9	5.50	1.45
0.95	10.50	1.48

Requerimiento de espacio

- El desempeño de búsqueda en una tabla hash es preferible al de búsqueda binaria en un arreglo ordenado (o en un ABB) si L es menor que 0.75
- OBS: a menor factor de carga, más buckets desocupados en una tabla hash
- Un ABB requiere al menos tres referencias por nodo (dato, subárbol izquierdo y subárbol derecho), por lo tanto, se requiere mayor espacio de almacenamiento para un ABB que para una tabla hash con un factor de carga de 0.75
- Para una tabla hash con factor de carga L, sea n el tamaño de la tabla (número de buckets). Entonces:
  - Hashing cerrado requiere espacio n
  - Hashing abierto requiere espacio n+2L (L es el número promedio de nodos en cada lista)

Próximas fechas...

- Resumen de la semana:
  - TDA tabla HASH

    ~~cátedra~~ – refuerzo – ~~laboratorio~~

- Próxima semana:
  - **PEP3**
  - **Entrega de Tarea 3**
  - **Pruebas recuperativas**

U4 - S13