

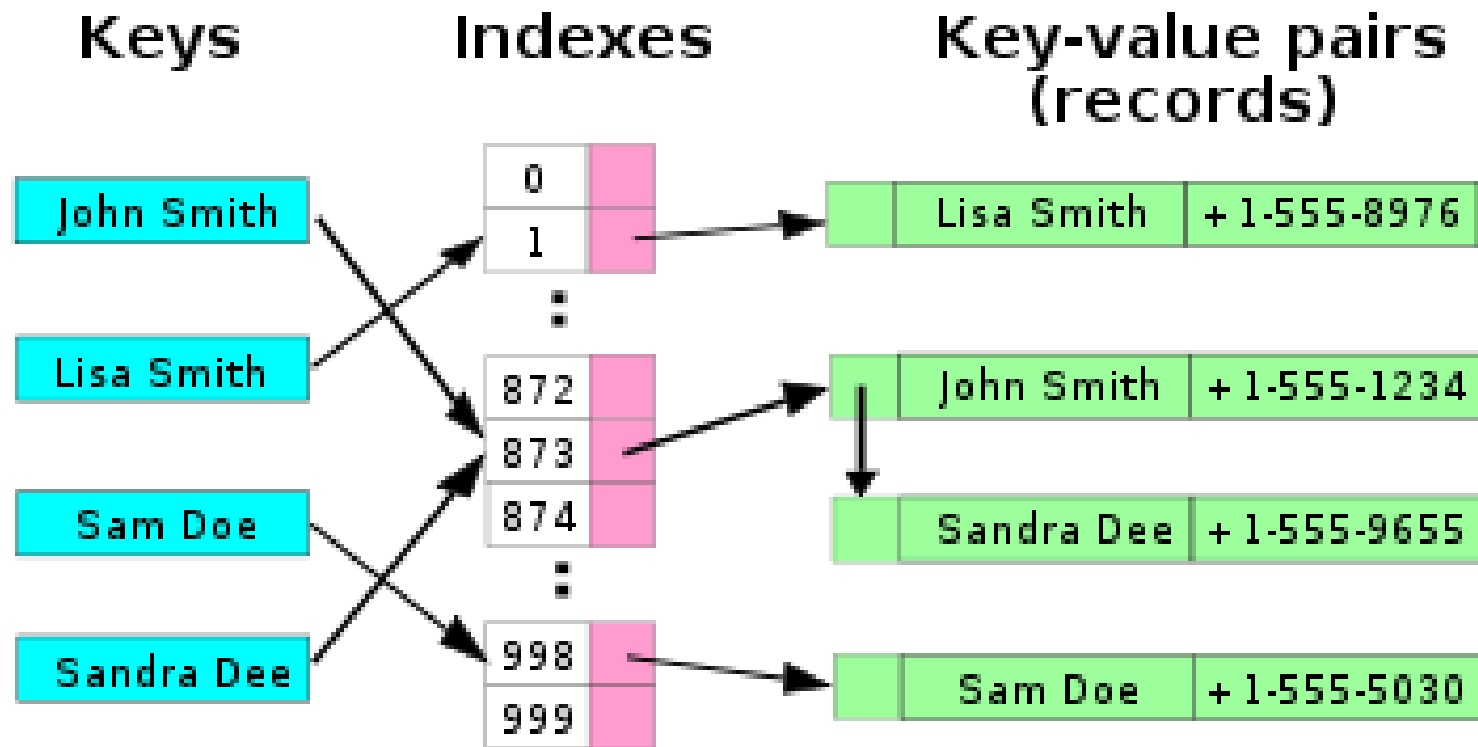
# Hashing

Estructura de Datos

# Tablas Hash

- ▶ Estructuras de datos que se *asocia llaves o claves* a cada elemento.
- ▶ Operaciones principales son de inserción y búsqueda, almacenando información (teléfonos, direcciones, etc.) a partir de una clave generada (id, fecha, nombre, cuenta, etc)
- ▶ Limitadas en tamaño ya que están basadas en arreglos.
- ▶ Su tamaño puede cambiar pero es recomendable evitar el cambio de tamaño.
- ▶ Su tamaño es recomendable que sea un número primo.

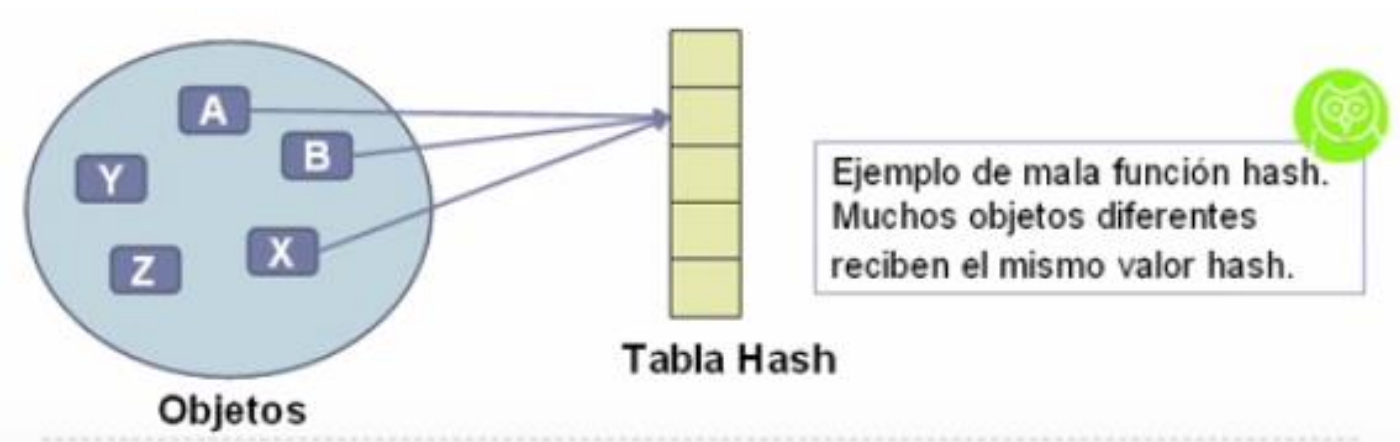
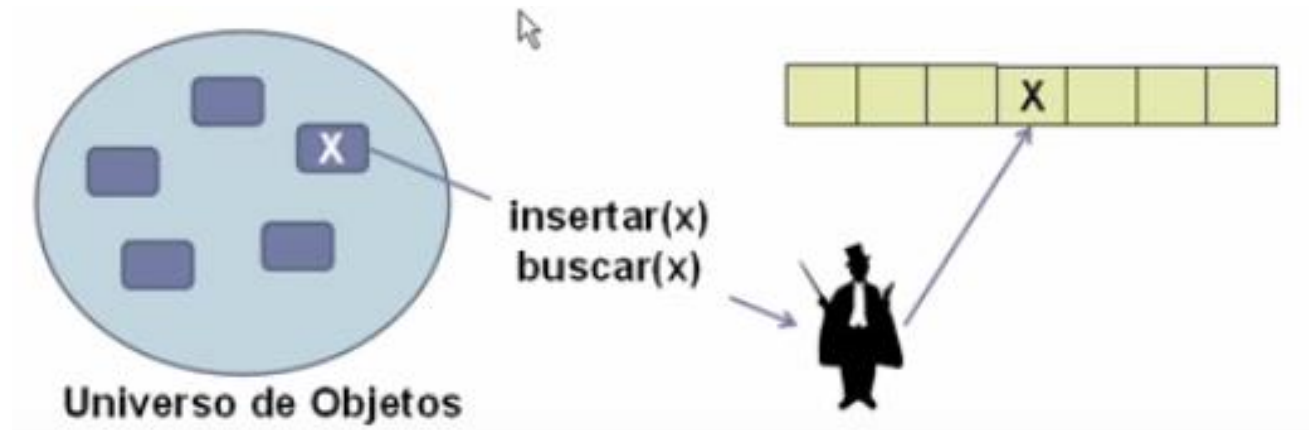
# Tablas Hash



# Tablas Hash

- ▶ Las claves asignadas a elementos en una tabla hash se realizan utilizando funciones hash.
- ▶ Una función hash ayuda a calcular el índice óptimo en el cual un elemento debería ubicarse.
- ▶ El índice debe ser menor que el tamaño de la tabla y menor o igual a cero.
- ▶ No debe haber datos repetidos en una tabla hash.

# Tablas Hash



# Tablas Hash

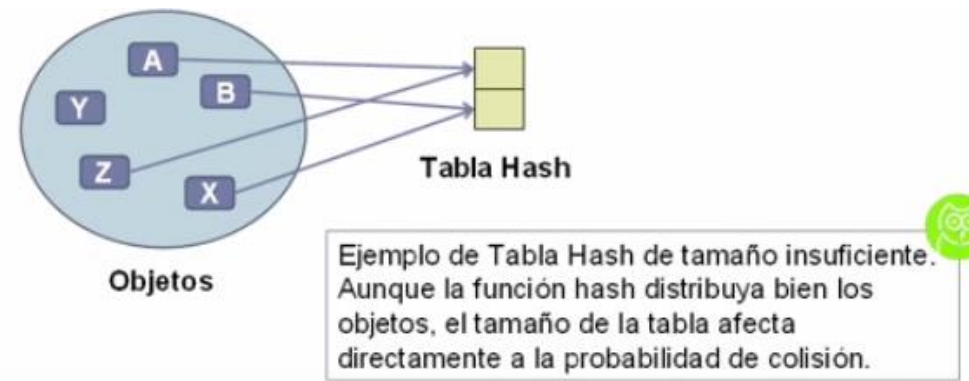
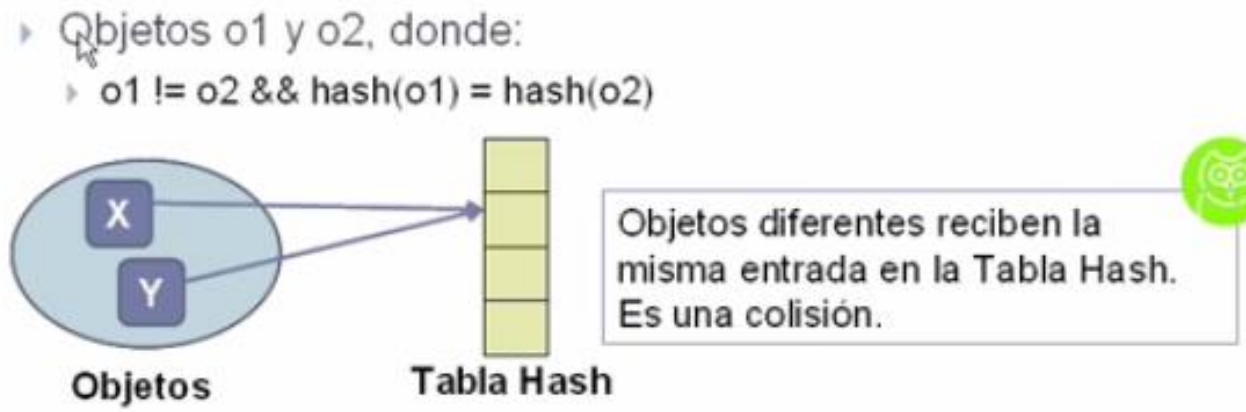
- ▶ Para usar una tabla hash se necesita:
  - ▶ Una estructura de acceso directo (array)
  - ▶ Una estructura de datos con una clave
  - ▶ Una función resumen (hash) cuyo dominio sea el espacio de claves y su imagen (o rango) los números naturales

# Método de inserción

- ▶ Para almacenar un elemento en la tabla hash se convierte su clave a un número. Esto se consigue aplicando la función hash a la clave del elemento.
- ▶ El resultado de la función hash se mapea al espacio de direcciones del vector que se emplea como soporte, lo cual se consigue con la función módulo. Con este paso se obtiene un índice válido para la tabla.
- ▶ El elemento se almacena en la posición de la tabla en el paso anterior.
  - ▶ Si en la posición de la tabla ya existe el elemento, se produce una COLISIÓN.
  - ▶ Esto se soluciona asociando una lista a cada posición de la tabla, aplicando otra función o buscando el siguiente elemento libre.

# Colisiones

- ▶ Una colisión se produce cuando objetos diferentes dan lugar al mismo índice hash.





# Métodos para resolver las colisiones

## Resolución mediante exploración

- ▶ Tratar de buscar otra posición libre en la tabla para albergar la inserción del elemento.
- ▶ Pueden ser:
  - ▶ **Exploración lineal:** visita la siguiente casilla. Si está libre realiza la inserción. Sino, visita la siguiente casilla.
  - ▶ **Exploración cuadrática:** visita la casilla  $i^2$  posiciones más allá de la que causó el  $i$ -ésima colisión. Si está libre, inserta. Sino, repite el proceso hasta encontrar una libre.
- ▶ El proceso de búsqueda sigue la misma secuencia de casillas.

# Ejemplo

- ▶ Insertar {20, 33, 21, 10, 12, 14, 56, 100}

Índice	Valor
0	
1	
2	
3	
4	
5	
6	
7	

Calcula el valor MÓDULO número de índice

Número del índice = 7

# Ejemplo

- Insertar {20, 33, 21, 10, 12, 14, 56, 100}

Índice	Valor
0	
1	
2	
3	
4	
5	
6	20
7	

Insertar el 20

$$20 \bmod 7 = 6$$

# Ejemplo

- Insertar {20, 33, 21, 10, 12, 14, 56, 100}

Índice	Valor
0	
1	
2	
3	
4	
5	33
6	20
7	

Insertar el 33

$$33 \bmod 7 = 5$$

# Ejemplo

- ▶ Insertar {20, 33, 21, 10, 12, 14, 56, 100}

Índice	Valor
0	21
1	
2	
3	
4	
5	33
6	20
7	

Insertar el 21

$$21 \bmod 7 = 0$$

# Ejemplo

- ▶ Insertar {20, 33, 21, 10, 12, 14, 56, 100}

Índice	Valor
0	21
1	
2	
3	10
4	
5	33
6	20
7	

Insertar el 10

$$10 \bmod 7 = 3$$

# Ejemplo

- ▶ Insertar {20, 33, 21, 10, 12, 14, 56, 100}

Índice	Valor
0	21
1	
2	
3	10
4	
5	33
6	20
7	12

Insertar el 12

$$12 \bmod 7 = 5$$

Se produce una COLISIÓN ya que en ese índice ya existe un elemento

Se busca hacia adelante una posición vacía  
En este caso en el índice 7

# Ejemplo

- Insertar {20, 33, 21, 10, 12, 14, 56, 100}

Índice	Valor
0	21
1	14
2	
3	10
4	
5	33
6	20
7	12

Insertar el 14

$$14 \bmod 7 = 0$$

Se produce una COLISIÓN ya que en ese índice ya existe un elemento

Se busca hacia adelante una posición vacía  
En este caso en el índice 1



# Ejemplo

- Insertar {20, 33, 21, 10, 12, 14, 56, 100}

Índice	Valor
0	21
1	14
2	56
3	10
4	
5	33
6	20
7	12

Insertar el 56

$$56 \bmod 7 = 0$$

Se produce una COLISIÓN ya que en ese índice ya existe un elemento

Se busca hacia adelante una posición vacía  
En este caso en el índice 2

# Ejemplo

- ▶ Insertar {20, 33, 21, 10, 12, 14, 56, 100}

Índice	Valor
0	21
1	14
2	56
3	10
4	100
5	33
6	20
7	12

Insertar el 100

$$100 \bmod 7 = 2$$

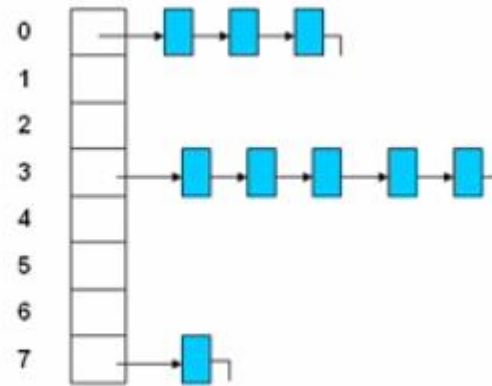
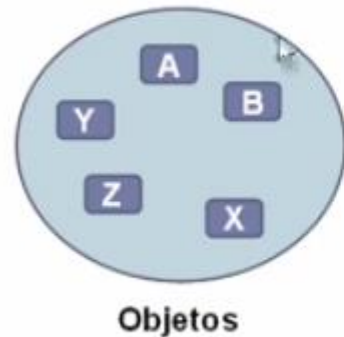
Se produce una COLISIÓN ya que en ese índice ya existe un elemento

Se busca hacia adelante una posición vacía  
En este caso en el índice 4

# Métodos para resolver las colisiones

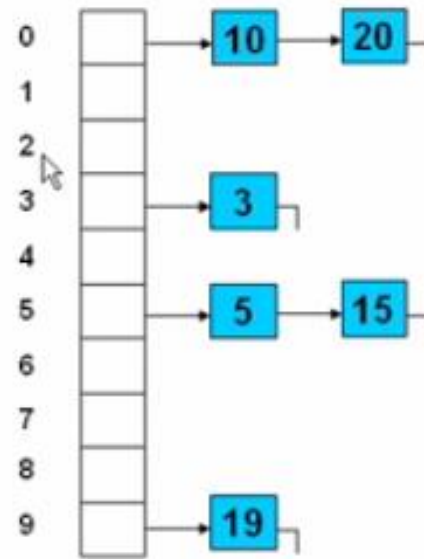
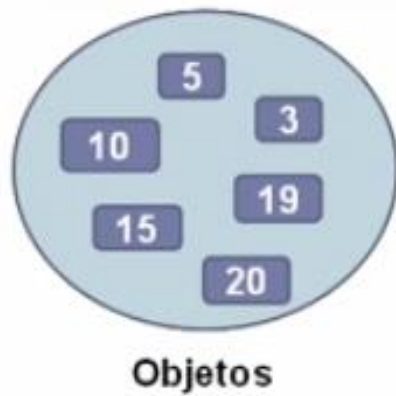
## Resolución mediante encadenamiento enlazado

- ▶ En cada posición de la tabla se mantiene una lista enlazada en la que se va insertando los elementos cuyo valor hash les asigna la misma posición.



# Ejemplo

- ▶ Insertar los elementos {5, 10, 3, 15, 20, 19}
- ▶ Tabla hash de tamaño 10
  - ▶  $\text{Hash}(x) = x \% 10$




# Método de búsqueda

- ▶ Para recuperar los datos, es necesario únicamente conocer la clave del elemento, a la cual se le aplica la función hash.
- ▶ El valor obtenido se mapea al espacio de direcciones de la tabla.
- ▶ Si el elemento existente en la posición indicada en el paso anterior tiene la misma que la empleada en la búsqueda, entonces es el deseado.
- ▶ Si la clave es distinta, se busca el elemento según la técnica empleada para resolver el problema de las colisiones al almacenar el elemento.

# Ejemplo

► Buscar {21}



Índice	Valor
0	21
1	14
2	56
3	10
4	100
5	33
6	20
7	12

$$21 \bmod 7 = 0$$

Elemento encontrado

**GRACIAS**