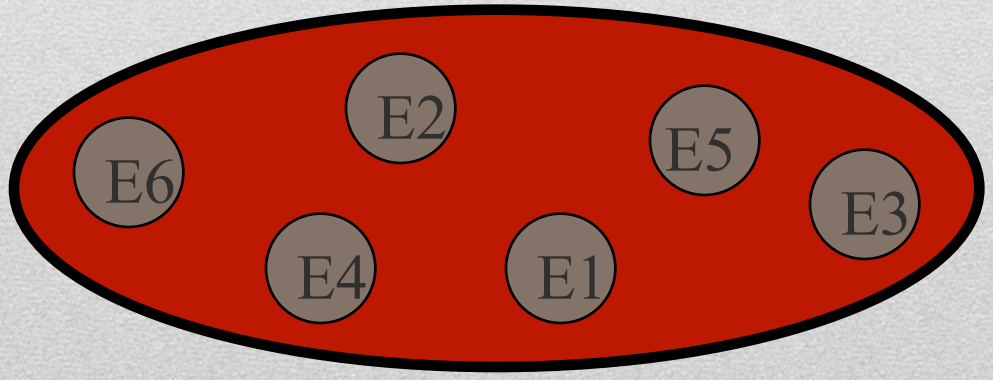
CONJUNTOS

Estructura de Datos que apoya la implementación de la técnica de **Hashing** para la **Búsqueda** de información

Ing. Luis Humberto González

¿Qué es un conjunto?

- ✓ *Desde el punto de vista de las estructuras de datos, un conjunto es un grupo de datos SIN RELACIONES entre sí.*
- ✓ *Al trabajar con un conjunto, lo importante es la* ***pretenencia o no*** *de un elemento en el conjunto.*



¿Qué puede guardar un ADT Conjunto?

- ✓ *Los elementos de un ADT Conjunto NO pueden ser de diferentes tipos.*
- ✓ *El ADT Conjunto NO guarda elementos repetidos.*
- ✓ *Según la aplicación puede guardar:*
  - **Elementos atómicos (datos simples), o**
  - **Elementos estructurados (objetos con atributos, donde uno de estos atributos es una llave).**

Aplicaciones del ADT Conjunto

- *Un conjunto de elementos atómicos tiene aplicaciones relacionadas con las operaciones típicas de conjuntos matemáticos: pertenencia, intersección, **unión**, diferencia, etc. Entre otras cosas es útil para hacer validaciones.*

- *Un conjunto de elementos estructurados puede funcionar como una estructura de **Búsqueda**, gracias a la operación de pertenencia.*

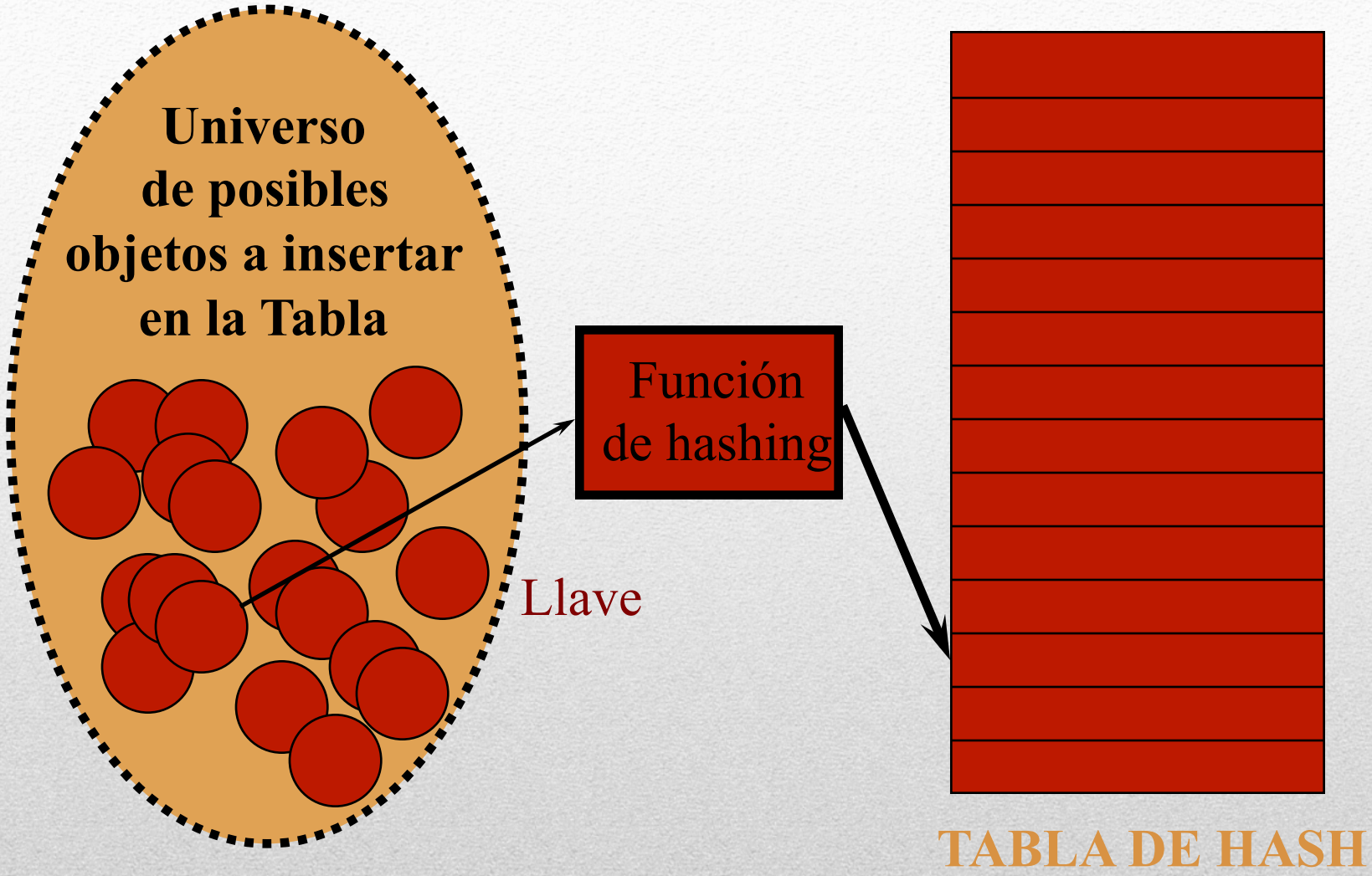
- ✓ *Típicamente, se utiliza una TABLA (arreglo de objetos) para representar al ADT Conjunto de elementos estructurados.*

- ✓ *El orden en que se almacenan los elementos en la Tabla estará determinado por la técnica de **HASHING**.*

Representación física para el ADT Conjunto…

La técnica de HASHING...

- *Dispersa **aleatoriamente** los elementos que se insertan en la Tabla.*
- *Se realiza una función matemática (FUNCION DE HASHING) a la llave del elemento para localizar la posición dentro de la Tabla que le corresponde.*
- *Lo ideal es que esa posición SIEMPRE esté disponible para él.*



La técnica de HASHING...

HASHING en la realidad…

- ✓ *La técnica de Hashing se preocupa por:*
  1. Diseñar una buena función de hashing para distribuir a los elementos lo más aleatoriamente posible en la Tabla.
     - ✳ **Dado que NO es posible asegurar que una función de hashing arrojará un valor distinto para cada llave, entonces, también se preocupa por:**
  2. Aplicar una estrategia eficiente para controlar las "**COLISIONES**" dentro de la Tabla.

Las propiedades que debe poseer una función de Hashing para ser considerada "buena" son:

- ✓ Ejecución Rápida (conversiones sencillas).
- ✓ Distribución Uniforme en todas las posiciones de la Tabla.

Funciones de Hashing

- ✓ *La función de Hashing depende directamente de las características de las llaves a considerar y de la Tabla sobre la que se trabajará (Tamaño).*
- ✓ *A la fecha, no existe ninguna metodología que genere "automáticamente" la MEJOR función de Hashing para un conjunto de llaves, sin embargo, existen métodos que pueden ayudar a su diseño.*

¿Cómo diseñar una Función de Hashing?

Metodologías útiles para diseñar una Función de Hashing

Algunos de los métodos más comúnmente empledos en el diseño de funciones son:

- ✓ Selección de Dígitos.
- ✓ Residuales (División).
- ✓ Cuadrática (Multiplicación).
- ✓ Folding (Plegamiento).

- ✓ *En general, los métodos para diseñar funciones de Hashing presuponen que la llave es un elemento numérico y en base a eso trabajan.*

- ✓ *Si la llave no fuera numérica, se realiza una conversión previa basada en códigos ASCII y los métodos se aplican sobre el resultado.*

Consideraciones especiales de las Metodologías

Selección de Dígitos

- *Esta técnica consiste en seleccionar algunos de los dígitos que conforman la llave y con ellos generar el índice para la Tabla.*

Ej: $H(d_1 d_2 d_3 d_4 d_5 d_6 d_7 d_8) = d_2 d_4 d_6$

donde $0 <= d_i <= 9$

- *Esto serviría para accesar una Tabla de 1000 elementos (000-999)*

- ✓ *Es una de los métodos más utilizados.*
- ✓ *Consiste en utilizar como índice, el residuo de dividir la llave entre el tamaño de la Tabla.*

Ej: $H(d_1d_2d_3d_4d_5d_6d_7d_8) = d_1d_2d_3d_4d_5d_6d_7d_8 \bmod N$

donde $0 <= H(\text{llave}) <= N-1$

N debe ser primo

Residuales

- *Consiste en elevar al cuadrado el valor de la llave y del resultado tomar los dígitos centrales (ya que son los más involucrados en el proceso).*

$$Ej:\ H(d_1d_2d_3) = digitos_centrales\ (d_1d_2d_3)^2$$
$$= r_3r_4$$

donde $0 <= r_3r_4 <= 99$

Cuadrática

- ✓ *Es una técnica muy utilizada.*
- ✓ *Consiste en agrupar algunos de los dígitos de la llave y aplicarles alguna(s) operación matemática para generar un índice.*

Ej: $H(d_1 d_2 d_3 d_4 d_5 d_6 d_7 d_8) = d_1+d_2+d_3+d_4+d_5+d_6+d_7+d_8$

donde $0 <= d_i <= 9$ y $0 <= H(\text{llave}) <= 54$

Folding

Suponer que se desea trabajar con objetos que tienen llave de86 dígitos (ej: matrícula del Tec) .

$$\text{llave} = d_1 d_2 d_3 d_4 d_5 d_6 d_7 d_8$$

Si la tabla fuera de 100 elementos (0-99), ¿sería "bueno" seleccionar a los primeros 2 dígitos como índice para la Tabla?

NO (Habría demasiadas repeticiones).

Ejemplos...

Si la tabla fuera de 100 elementos (0-99), ¿sería "bueno" sumar todos los dígitos de la llave y usar el resultado como índice para la Tabla?

NO (el mayor índice sería 72)

Ejemplos…

Si la tabla fuera de 100 elementos (0-99), y se utilizara la función $H(d_1d_2d_3d_4d_5d_6d_7d_8) = d_5d_6$

Esta función genera únicamente valores entre 00 y 99, teóricamente se distribuyen aleatoriamente. Por lo tanto es una "buena" función... pero,

¿Qué pasaría si se quisieran insertar las llaves 11234567, 22469067 y 34987467?

Según la función a todas esas llaves les corresponde la posición 67... Las tres llaves colisionan.

Ejemplos...

Las estrategias más comunes se dividen en dos grupos, dependiendo de cómo y dónde acomodan al elemento colisionado. Estos grupos son:

- ✓ Métodos de Dirección Abierta (también conocidos como métodos de Hashing Cerrado o de Rehashing).
- ✓ Métodos de Encadenamiento (también conocidos como métodos de Hashing Abierto).

Estrategias para el manejo de Colisiones

- ✓ *Su principal característica es que el elemento colisionado es "reacomodado" dentro del espacio de la Tabla.*
- ✓ *Todos los métodos de esta categoría consideran que la Tabla es un espacio CIRCULAR y su funcionamiento se basa en el índice original generado por la función de Hashing (dirección base)*

Métodos de Dirección Abierta.

Métodos de Dirección Abierta.

Los principales métodos de la categoría de Dirección Abierta son:

- ✓ Prueba Lineal
- ✓ Prueba Cuadrática
- ✓ Prueba Reasignación Aleatoria
- ✓ Prueba Doble Hashing

- *Una vez que se genera una colisión, esta prueba busca en forma secuencial una posición disponible , a partir de la dirección base del elemento.*

  Nueva_posición = dir_base + J

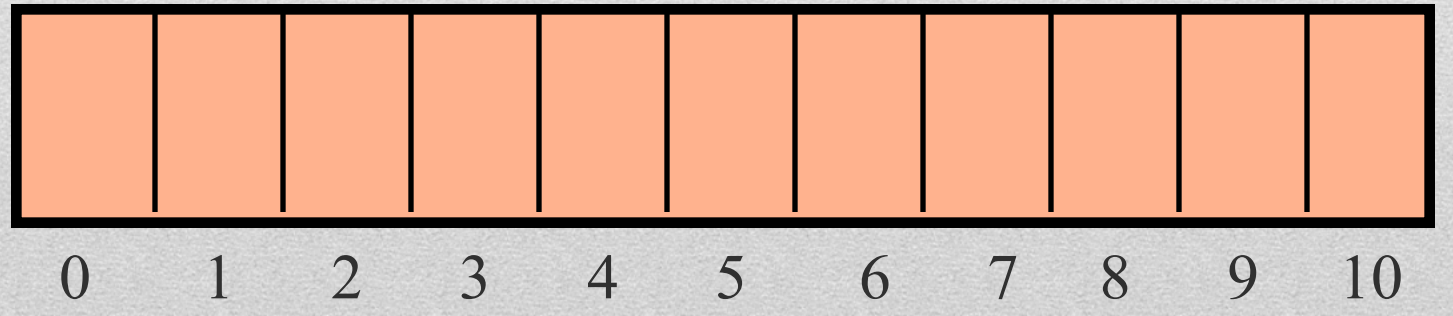
  donde J es la cantidad de colisiones que ha provocado esa llave.

- *Es la técnica más sencilla de implementar, pero es muy ineficiente*

Prueba Lineal

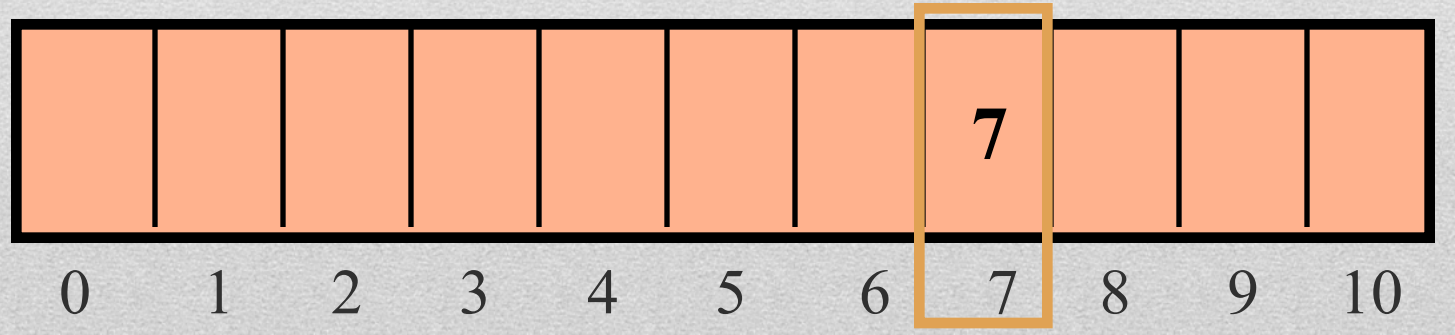
Prueba Lineal... ejemplo

- Suponer : H(llave) = llave mod 11
- Insertar la secuencia: 7, 17, 6, 33, 28, 52



Prueba Lineal... ejemplo

- Suponer : H(llave) = llave mod 11
- Insertar la secuencia: 7, 17, 6, 33, 28, 52



Prueba Lineal... ejemplo

- Suponer : H(llave) = llave mod 11
- Insertar la secuencia: 7, 17, 6, 33, 28, 52

0	1	2	3	4	5	6	7	8	9	10
						17	**7**			

Prueba Lineal... ejemplo

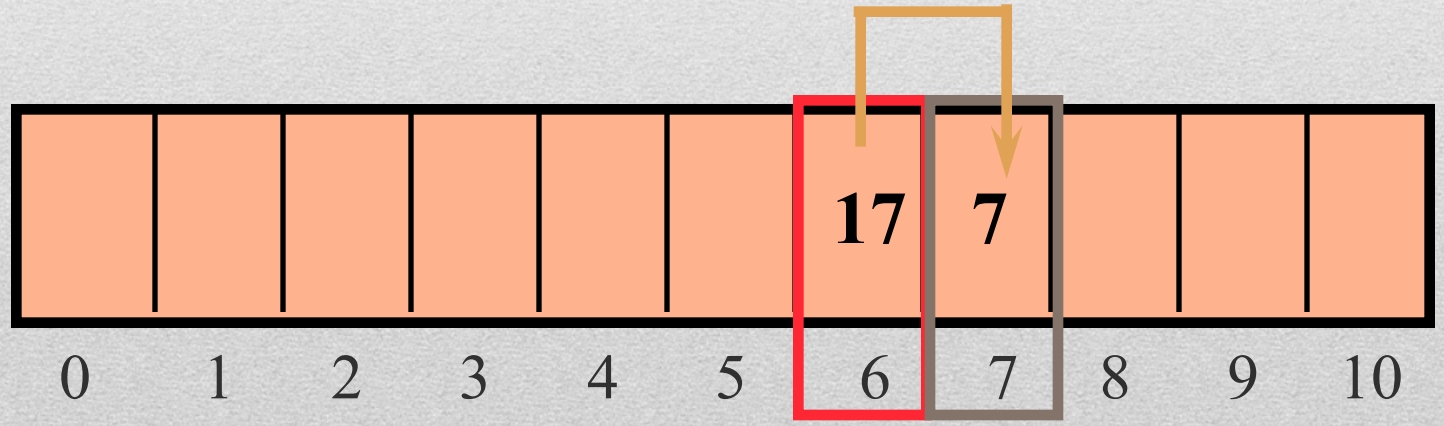
- Suponer : H(llave) = llave mod 11
- Insertar la secuencia: 7, 17, 6, 33, 28, 52

						17	**7**			
0	1	2	3	4	5	6	7	8	9	10

COLISION !!!

Prueba Lineal... ejemplo

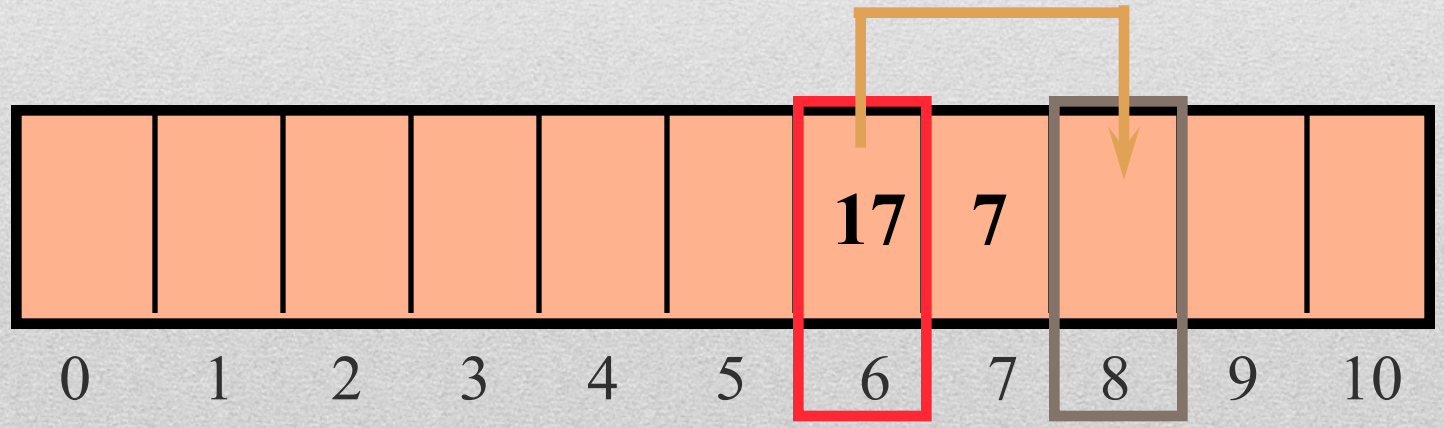
- Suponer : H(llave) = llave mod 11
- Insertar la secuencia: 7, 17, 6, 33, 28, 52



COLISION !!!

Prueba Lineal... ejemplo

- Suponer : H(llave) = llave mod 11
- Insertar la secuencia: 7, 17, 6, 33, 28, 52



DISPONIBLE !!!

Prueba Lineal... ejemplo

- Suponer : H(llave) = llave mod 11
- Insertar la secuencia: 7, 17, 6, 33, 28, 52

						17	**7**	**6**		
0	1	2	3	4	5	6	7	8	9	10

Prueba Lineal... ejemplo

- Suponer : H(llave) = llave mod 11
- Insertar la secuencia: 7, 17, 6, 33, 28, 52

0	1	2	3	4	5	6	7	8	9	10
33						**17**	**7**	**6**		

Prueba Lineal... ejemplo

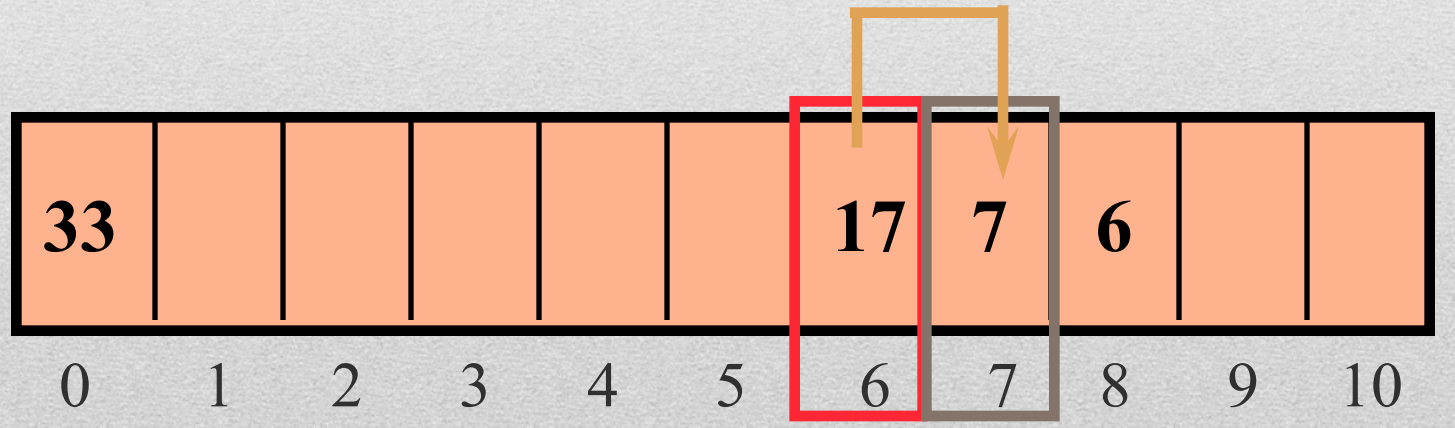
- Suponer : H(llave) = llave mod 11
- Insertar la secuencia: 7, 17, 6, 33, 28, 52

0	1	2	3	4	5	6	7	8	9	10
33						**17**	**7**	**6**		

COLISION !!!

Prueba Lineal... ejemplo

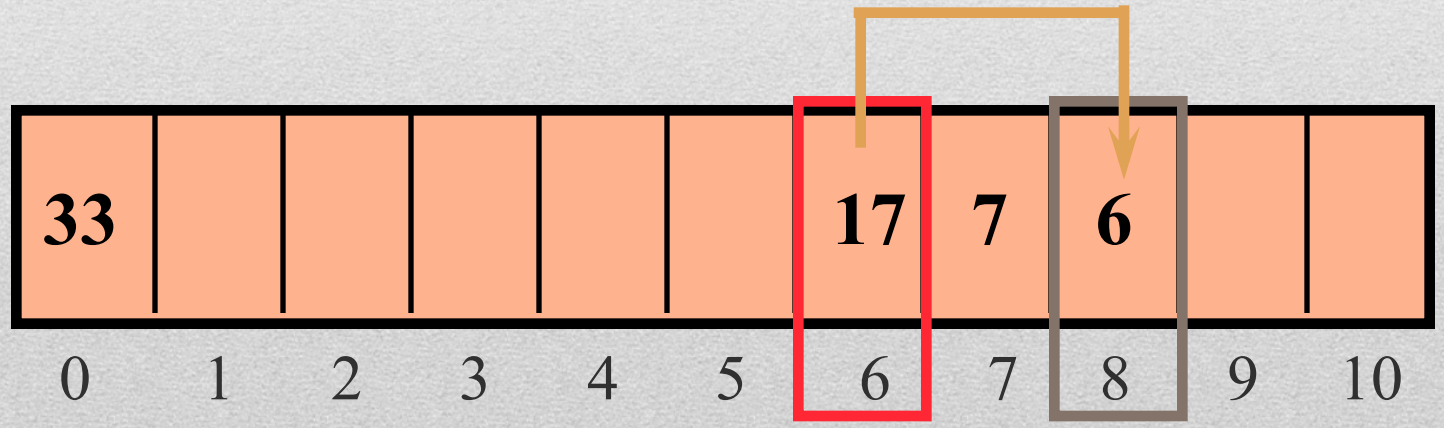
- Suponer : H(llave) = llave mod 11
- Insertar la secuencia: 7, 17, 6, 33, 28, 52



COLISION !!!

Prueba Lineal... ejemplo

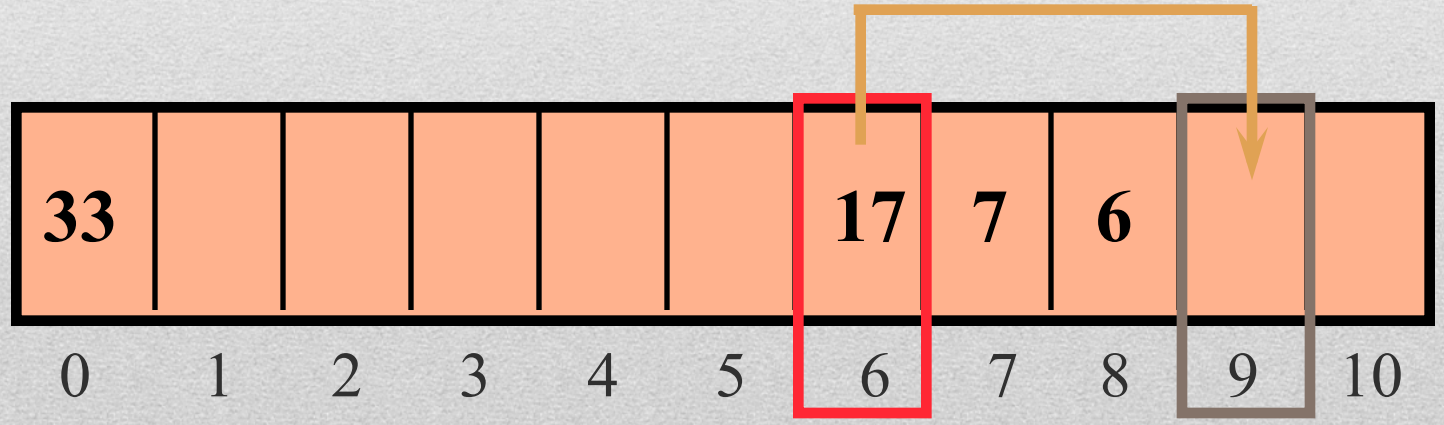
- Suponer : H(llave) = llave mod 11
- Insertar la secuencia: 7, 17, 6, 33, 28, 52



COLISION !!!

Prueba Lineal... ejemplo

- Suponer : H(llave) = llave mod 11
- Insertar la secuencia: 7, 17, 6, 33, 28, 52



DISPONIBLE !!!

Prueba Lineal... ejemplo

- Suponer : H(llave) = llave mod 11
- Insertar la secuencia: 7, 17, 6, 33, 28, 52

33						**17**	**7**	**6**	**28**	
0	1	2	3	4	5	6	7	8	9	10

Prueba Lineal... ejemplo

- Suponer : H(llave) = llave mod 11
- Insertar la secuencia: 7, 17, 6, 33, 28, 52

0	1	2	3	4	5	6	7	8	9	10
33						**17**	**7**	**6**	**28**	

COLISION !!!

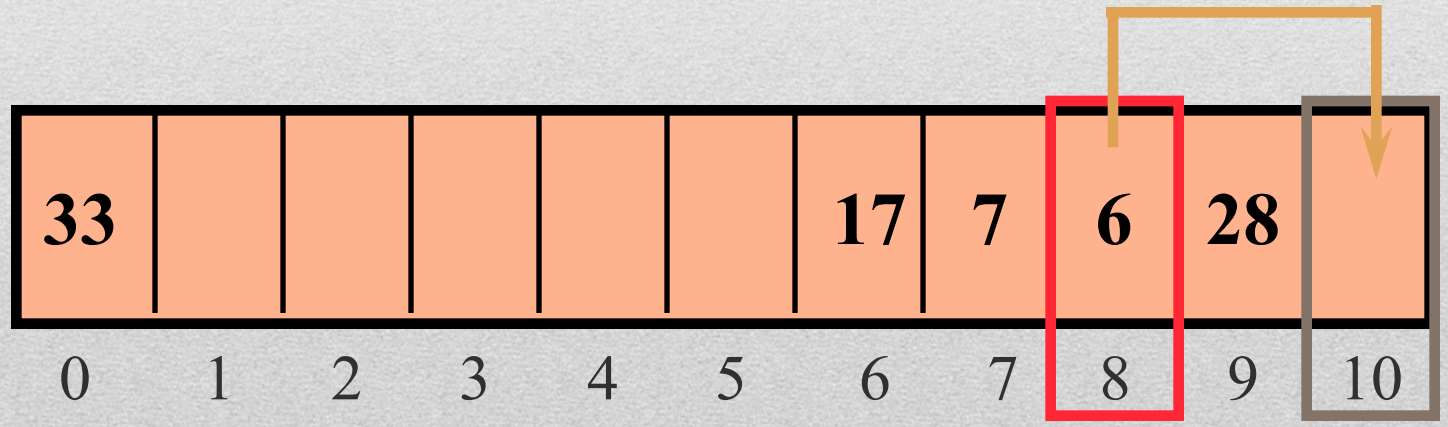
Prueba Lineal... ejemplo

- Suponer : H(llave) = llave mod 11
- Insertar la secuencia: 7, 17, 6, 33, 28, 52

0	1	2	3	4	5	6	7	8	9	10
33						**17**	**7**	**6**	**28**	

COLISION !!!

- Suponer : H(llave) = llave mod 11
- Insertar la secuencia: 7, 17, 6, 33, 28, 52



DISPONIBLE!!!

Prueba Lineal... ejemplo

- Suponer : H(llave) = llave mod 11
- Insertar la secuencia: 7, 17, 6, 33, 28, 52

0	1	2	3	4	5	6	7	8	9	10
33						**17**	**7**	**6**	**28**	**52**

Prueba Lineal... ejemplo

- Suponer : H(llave) = llave mod 11
- Insertar la secuencia: 7, 17, 6, 33, 28, 52

¿Qué pasaría si ahora se quisiera insertar el 76?

0	1	2	3	4	5	6	7	8	9	10
33						**17**	**7**	**6**	**28**	**52**

Prueba Lineal... ejemplo

- Suponer : H(llave) = llave mod 11
- Insertar la secuencia: 7, 17, 6, 33, 28, 52

¿Qué pasaría si ahora se quisiera insertar el 76?

0	1	2	3	4	5	6	7	8	9	10
33						**17**	**7**	**6**	**28**	**52**

COLISION !!!

Prueba Lineal... ejemplo

- Suponer : H(llave) = llave mod 11
- Insertar la secuencia: 7, 17, 6, 33, 28, 52

¿Qué pasaría si ahora se quisiera insertar el 76?

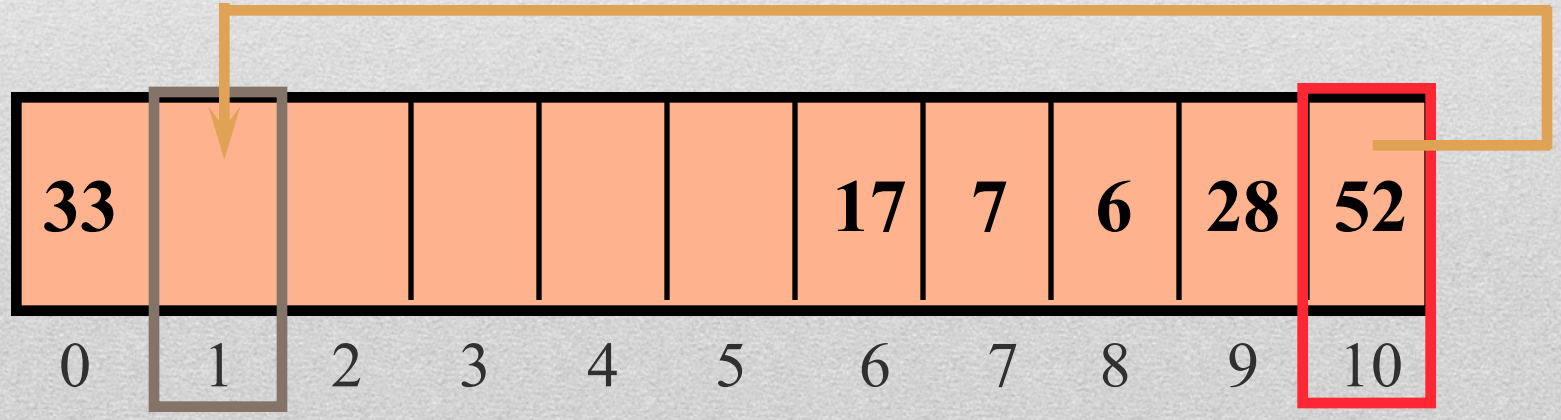
0	1	2	3	4	5	6	7	8	9	10
33						**17**	**7**	**6**	**28**	**52**

COLISION !!!

Prueba Lineal... ejemplo

- Suponer : H(llave) = llave mod 11
- Insertar la secuencia: 7, 17, 6, 33, 28, 52

¿Qué pasaría si ahora se quisiera insertar el 76?



DISPONIBLE !!!

Prueba Lineal... ejemplo

- Suponer : H(llave) = llave mod 11
- Insertar la secuencia: 7, 17, 6, 33, 28, 52

¿Qué pasaría si ahora se quisiera insertar el 76?

0	1	2	3	4	5	6	7	8	9	10
33	**76**					**17**	**7**	**6**	**28**	**52**

Prueba Lineal

- ✓ *El proceso de búsqueda usando esta prueba :*

  Calcula la dirección base del elemento buscado.

  Repite:

  Compara el contenido de la casilla con el elemento.

  Si es el elemento buscado : *EXITO!!* y termina.

  Si no.. se mueve a la siguiente posición

  Hasta que: se encuentre el elemento, o llegue a una casilla vacía, o llegue a la dirección base.

  Si se llegó a una posición vacía o la la dirección base, *FRACASO!!* el elemento no se encuentra.

- ✓ *Los procesos de Inserción y Eliminación llevan implícito al proceso de búsqueda.*

Prueba Lineal... ejemplo

- Suponer que ahora se da de baja al elemento 6.

33	**76**					**17**	**7**	**6**	**28**	**52**
0	1	2	3	4	5	6	7	8	9	10

Prueba Lineal... ejemplo

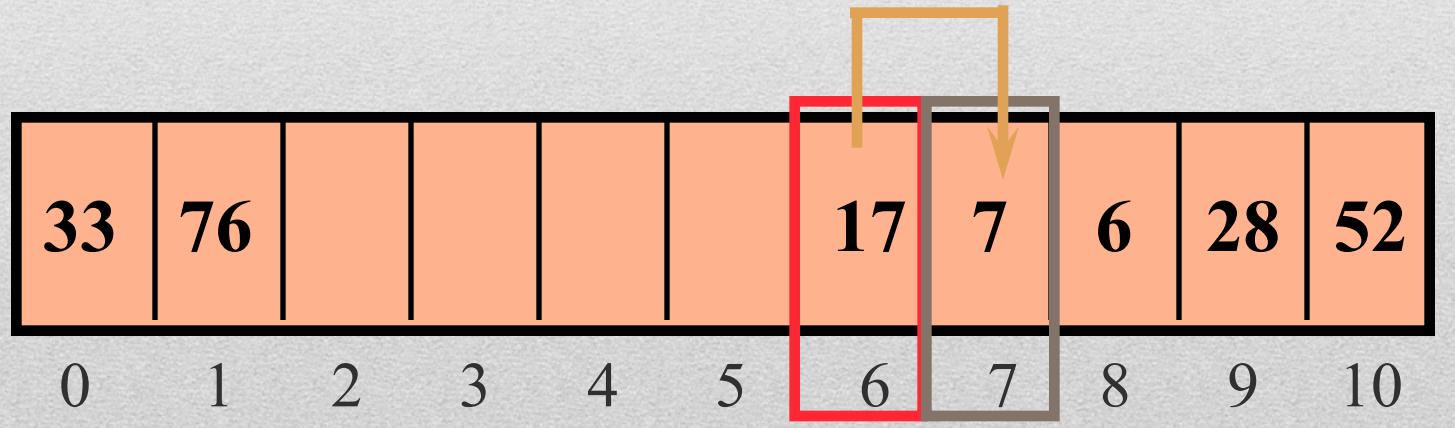
- Suponer que ahora se da de baja al elemento 6.

33	76					17	7	6	28	52
0	1	2	3	4	5	6	7	8	9	10

NO ES EL ELEMENTO BUSCADO!!

Prueba Lineal... ejemplo

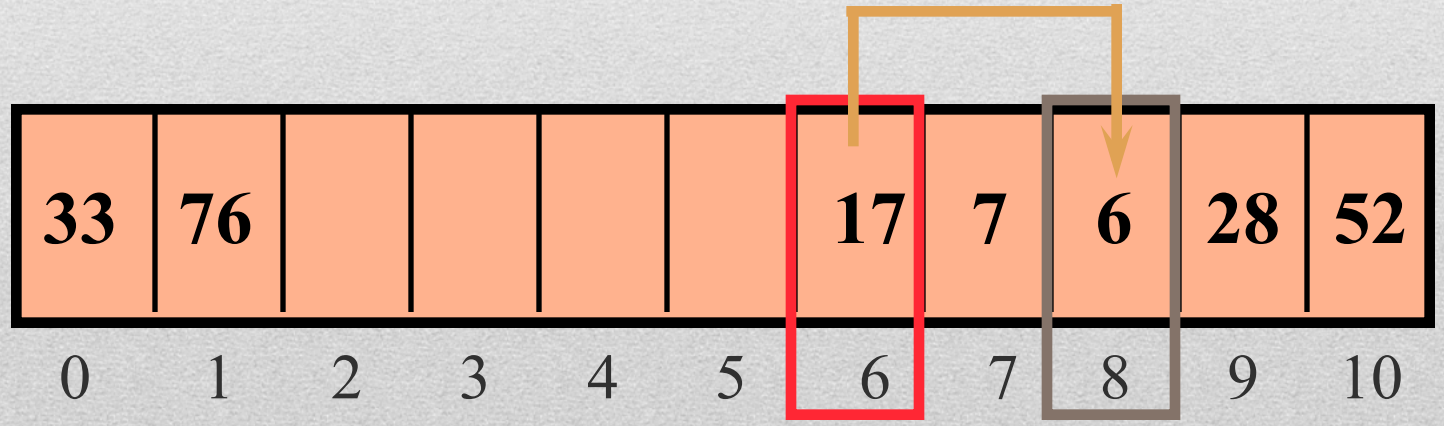
- Suponer que ahora se da de baja al elemento 6.



TAMPOCO ESTE!!

Prueba Lineal... ejemplo

- Suponer que ahora se da de baja al elemento 6.



ESTE SI ES!!!

Prueba Lineal... ejemplo

- Suponer que ahora se da de baja al elemento 6.
- ¿Qué pasaría ahora si se quisiera borrar al 52?

33	**76**					**17**	**7**		**28**	**52**
0	1	2	3	4	5	6	7	8	9	10

Prueba Lineal… ejemplo

- Suponer que ahora se da de baja al elemento 6.
- ¿Qué pasaría ahora si se quisiera borrar al 52?

33	**76**					**17**	**7**		**28**	**52**
0	1	2	3	4	5	6	7	8	9	10

Prueba Lineal… ejemplo

PROBLEMAS !!!!

Prueba Lineal

- ¿Cómo resolver el problema anterior?
- Las pruebas correspondientes a los Métodos de Dirección Abierta requieren que cada casilla de la Tabla de almacenamiento contenga lo siguiente:
  - Espacio para almacenar al elemento.
  - Una bandera de estado (status) que indique si la casilla está vacía, está ocupada o está borrada (porque ya alguna vez tuvo un dato).

Prueba Lineal… ejemplo

¿Por qué es importante la bandera de status?

- ✓ Sirve para controlar los procesos de búsqueda, inserción y eliminación.
  - La búsqueda, en el peor caso compara hasta llegar a una casilla VACIA.
  - La inserción, agrega al nuevo elemento al encontrar una casilla VACIA o BORRADA.
  - La eliminación, no se detiene hasta encontrar al elemento o llegar a una casilla VACIA (no está).

Prueba Lineal... ejemplo

- ✓ Recordando el problema anterior...
- ✓ ¿Qué pasaría ahora si se quisiera borrar al 52?
- ✓ Ahora manejando status...

0	1	2	3	4	5	6	7	8	9	10
33	**76**					**17**	**7**		**28**	**52**

PROBLEMAS !!!!

Prueba Lineal... ejemplo

- ✓ Recordando el problema anterior...
- ✓ ¿Qué pasaría ahora si se quisiera borrar al 52?
- ✓ Ahora manejando status...

o	o	v	v	v	v	o	o	b	o	o
33	**76**					**17**	**7**		**28**	**52**
0	1	2	3	4	5	6	7	8	9	10

o = ocupado, v= vacío, b= borrado

Prueba Lineal... ejemplo

- ¿Qué pasaría ahora si se quisiera buscar al 76?
- ¿Qué pasa si se desea insertar el 39?

o	o	v	v	v	v	o	o	b	o	b
33	**76**					**17**	**7**		**28**	
0	1	2	3	4	5	6	7	8	9	10

o = ocupado, v= vacío, b= borrado

Prueba Lineal... ejemplo

- ¿Qué pasaría ahora si se quisiera buscar al 76?
- ¿Qué pasa si se desea insertar el 39?

o	o	v	v	v	v	o	o	o	o	b
33	**76**					**17**	**7**	**39**	**28**	
0	1	2	3	4	5	6	7	8	9	10

o = ocupado, v= vacío, b= borrado

- ✓ *Su principal característica es que el elemento colisionado es "reacomodado" en un espacio externo a la Tabla.*
- ✓ *Todos los métodos de esta categoría "encadenan" a todas las llaves que generaron la misma dirección base.*

Métodos de Encadenamiento.

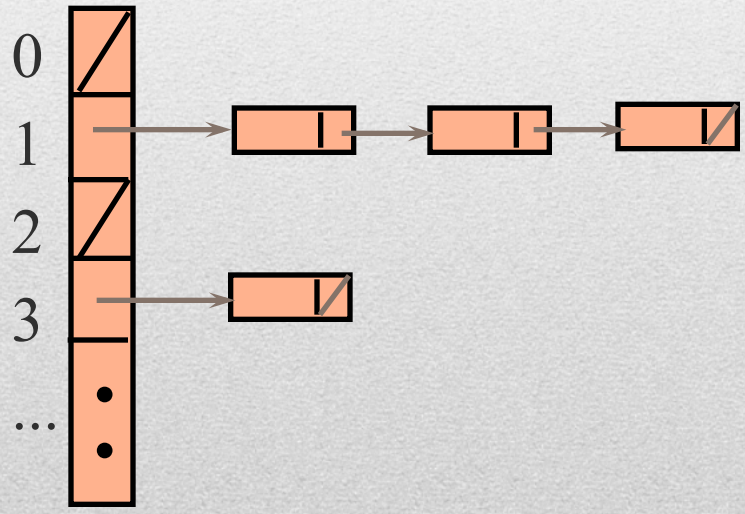
Los principales métodos de la categoría de Encadenamiento son:

- ✓ Método de Encadenamiento Externo.
- ✓ Método de Encadenamiento de Colisiones.

Métodos de Encadenamiento.

Cuando una llave tiene como dirección base una posición (bucket) ocupada, este método acomoda dicha llave dentro del mismo bucket. Típicamente éstos son representados como listas encadenadas.

Gráficamente:



Encadenamiento Externo.

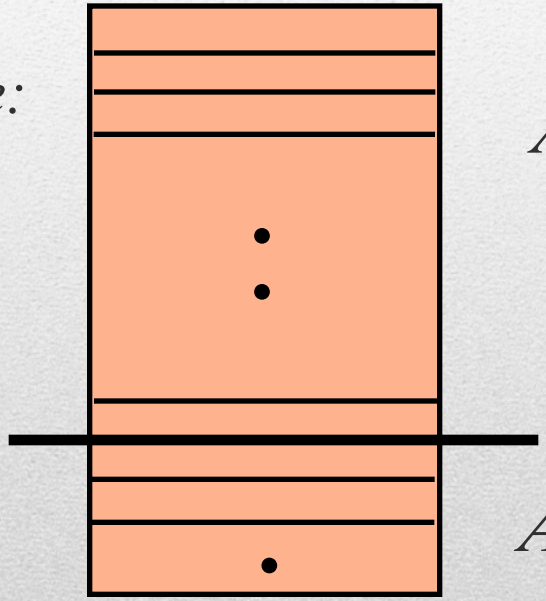
- ✓ *Los buckets pueden ser representados a través de listas encadenadas, árboles de búsqueda, tablas, etc...*

- ✓ *En esta representación, las casillas (buckets) de la Tabla de Hashing ya no requieren tener una bandera de status, ya que el proceso de búsqueda sólo analizará los elementos que se encuentran en el mismo bucket.*

Encadenamiento Externo.

Este método es muy similar al anterior, sólo que aquí los elementos colisionados en un bucket son acomodados en un espacio contiguo a la Tabla de Hash, denominado Area de Overlay.

Gráficamente:



Area Normal (o de Hash)

Area de Overlay

Encadenamiento de Colisiones.

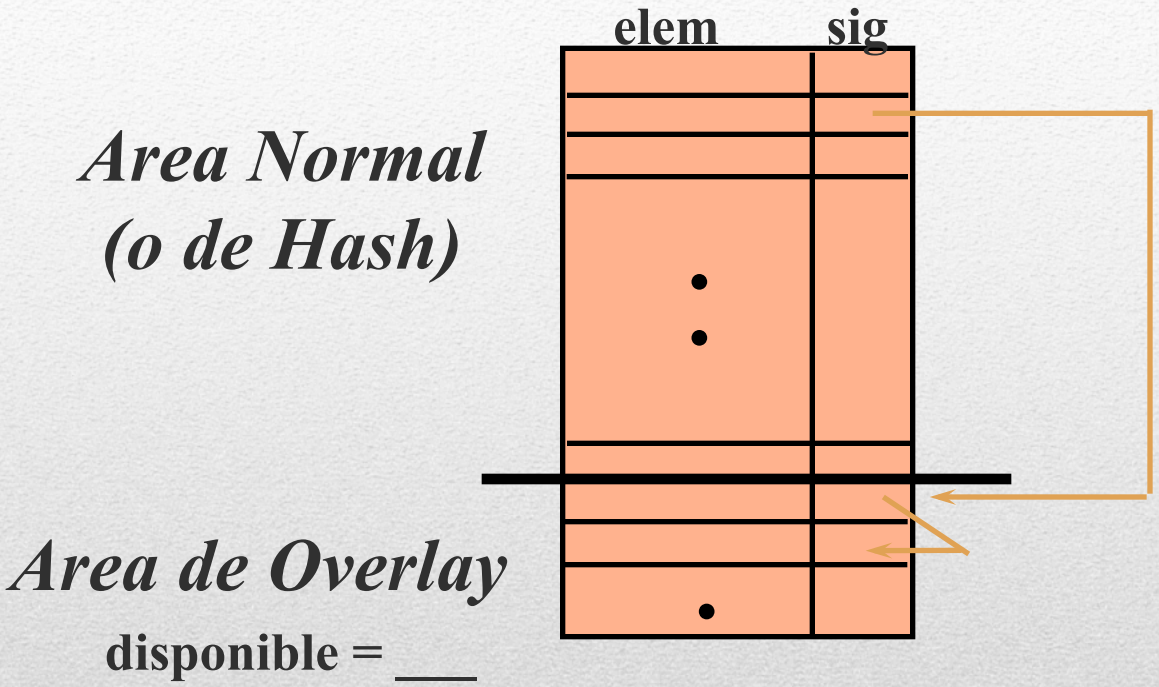
- ✓ *El área Normal es la Tabla de Hashing tradicional.*
- ✓ *El área de Overlay es un fragmento adicional a la Tabla.*
- ✓ *Por razones de eficiencia el área de Overlay debe tener entre un 14 y un 21 % del tamaño del área Normal.*

Encadenamiento de Colisiones.

- ✓ *En este método, cada casilla (buckets) debe tener, además del espacio para un elemento, un apuntador que almacene la posición del siguiente elemento (en el área de Overlay) que pertenece al mismo bucket.*
- ✓ *El área de Overlay debe tener un apuntador hacia el siguiente espacio disponible.*
- ✓ *Esta área puede ser administrada como fila, pila, o en forma aleatoria (no recomendable).*

Encadenamiento de Colisiones.

Gráficamente:



Encadenamiento de Colisiones.

- ✓ *El proceso de búsqueda e inserción es igual que en el método de encadenamiento, con el inconveniente de que el espacio de overlay es limitado.*

- ✓ *El proceso de eliminación puede llegar a complicarse, sobre todo cuando se pretende eliminar al elemento que se encuentra en el inicio del bucket (dentro del área normal).*

Encadenamiento de Colisiones.