

ESTRUCTURAS DE DATOS

DICCIONARIOS

# **Análisis de coste en tablas *hash***

Manuel Montenegro Montes  
Departamento de Sistemas Informáticos y Computación  
Facultad de Informática – Universidad Complutense de Madrid

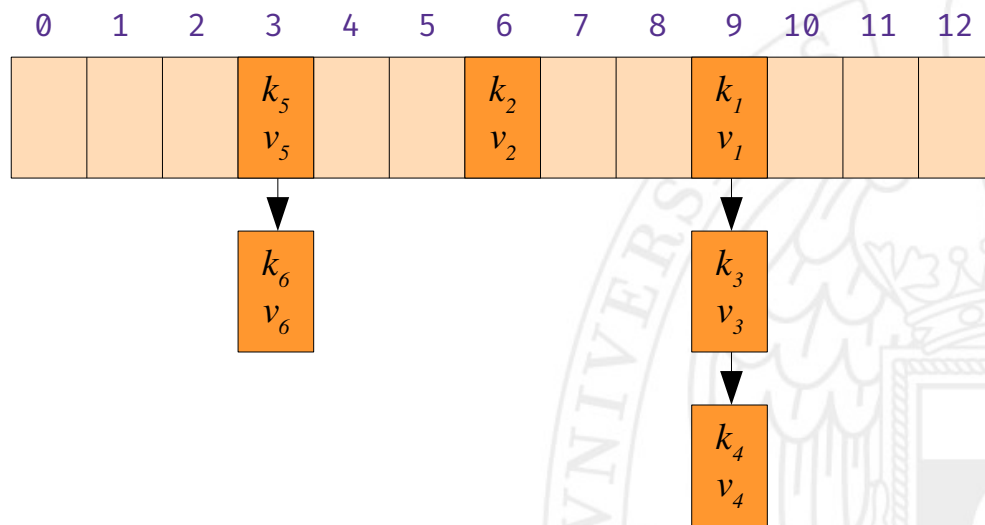
# Tablas *hash* abiertas

En este vídeo vamos a hablar del coste de cada una de las operaciones. Tanto para las tablas hash abiertas como para las tablas hash cerradas.

Tenemos que tener en cuenta el factor de carga de una tabla hash.

# Recordatorio

- Una tabla *hash* abierta asocia cada cajón con una lista de entradas.
- En caso de colisión entre claves, las entradas acaban en la misma lista.



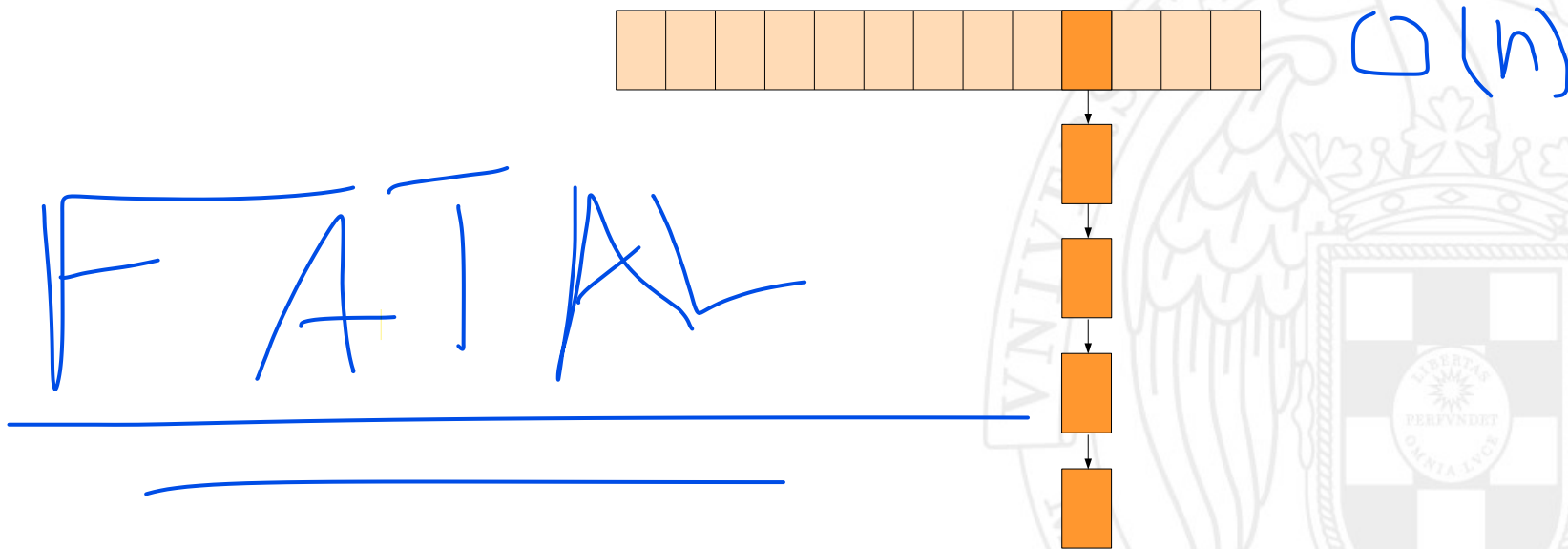
# Factor de carga

- El **factor de carga**  $\alpha$  de una tabla *hash* es el cociente entre el número de entradas en la tabla y el número de cajones.
- Sean:  
 $n$  - número de entradas en la tabla  
 $m$  - número de cajones  
$$\alpha = \frac{n}{m}$$
- Expresaremos el **coste** de los algoritmos en función del factor de carga.

Va a aparecer en este vídeo y en los vídeos siguientes.

# Dispersión uniforme

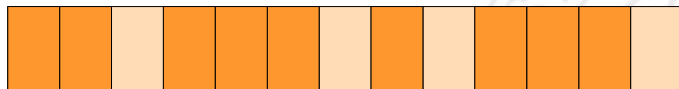
- La eficiencia de una tabla *hash* viene determinada por las propiedades de la función *hash* utilizada.
- Una función *hash* nefasta enviaría todas las claves al mismo cajón.



# Dispersión uniforme

**Suposición de dispersión uniforme:** ESTO SERÍA UNA BUENA FUNCIÓN

La función *hash* distribuye uniformemente todas las claves a lo largo de los cajones de la tabla.



Por ejemplo. Queremos meter en la tabla hash una clave  $k$ . Una buena función hash haría que la  $k$  tuviera la misma probabilidad de caer en cualquiera de los cajones.

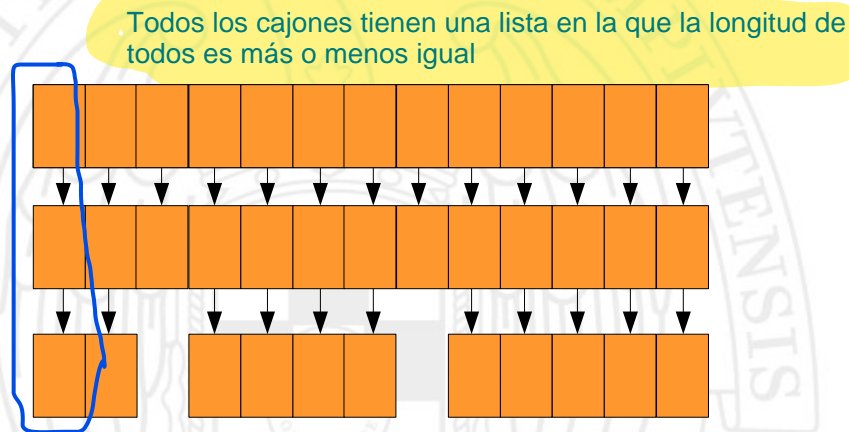
# Longitud media de las listas

- Supongamos que tenemos  $n$  elementos y  $m$  cajones, y que la propiedad de distribución uniforme se cumple.
- Sea  $N_i$  la longitud de la lista del cajón  $i$ -ésimo.
- ¿Cuál es el valor promedio de  $N_i$ ?

$$\underline{E[N_i]} = \frac{n}{m} = \alpha$$

FACTOR DE CARGA.

$3 = N_0$

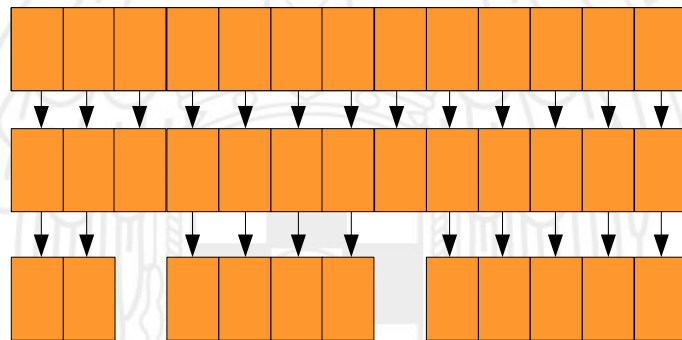


# Búsqueda de una clave (fallo)

- Supongamos que realizamos una búsqueda de una clave que no se encuentra en la tabla.
- El coste debe recorrer una de las listas en su totalidad.
- Por tanto, el coste medio es proporcional a  $\alpha$ .  $\rightarrow$  Proporcional al número de elementos de ese cajón
- Similarmente para la inserción y borrado.

$$O(1 + \alpha)$$

Ese 1 viene del cálculo de la función HASH inicial

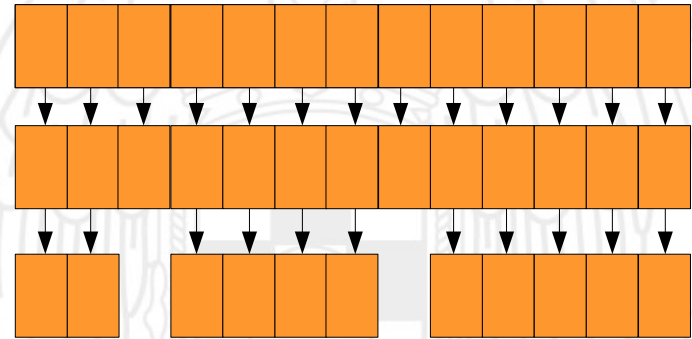




# Búsqueda de una clave (éxito)

- Supongamos que realizamos una búsqueda de una clave que sí se encuentra en la tabla.
- El número medio de elementos recorridos es  $1 + \frac{\alpha}{2} - \frac{\alpha}{2n}$
- Similarmente para la inserción y borrado.

$$1 + \frac{\alpha}{2} - \frac{\alpha}{2n} \in O(1 + \alpha)$$



# Conclusión

- El coste de todas las operaciones está acotado por  $\alpha$ .  $O(1+\alpha)$
- Si conseguimos mantener  $\alpha$  acotado, el coste de las operaciones será constante.**

Si conseguimos mantener alfa por debajo de una cota/determinado nivel

- ¿Cómo conseguimos mantener  $\alpha$  acotado?

$$\alpha = \frac{n}{m}$$

La  $n$  aumenta cuantos más elementos metamos en la tabla.

Haciendo que el número de cajones aumente proporcionalmente con el número de entradas → Tabla dinámicamente redimensionable.

Si conseguimos que, ha medida que aumenta la  $n$ , que aumente también la  $m$  con eso vamos a conseguir que la alfa se mantenga acotada.

# Tablas *hash* cerradas

PASAMOS A ANALIZAR EL COSTE DE LAS TABLAS HASH CERRADAS.

# Recordatorio

- Una tabla *hash* cerrada contiene, en cada cajón, una única entrada.
- En caso de colisión entre claves, las entradas acaban en cajones distintos según la estrategia de redistribución utilizada.

0	1	2	3	4	5	6	7	8	9	10	11	12
			$k_5$ $v_5$	$k_6$ $v_6$		$k_2$ $v_2$			$k_1$ $v_1$	$k_3$ $v_3$	$k_4$ $v_4$	

# Factor de carga

- Sean:

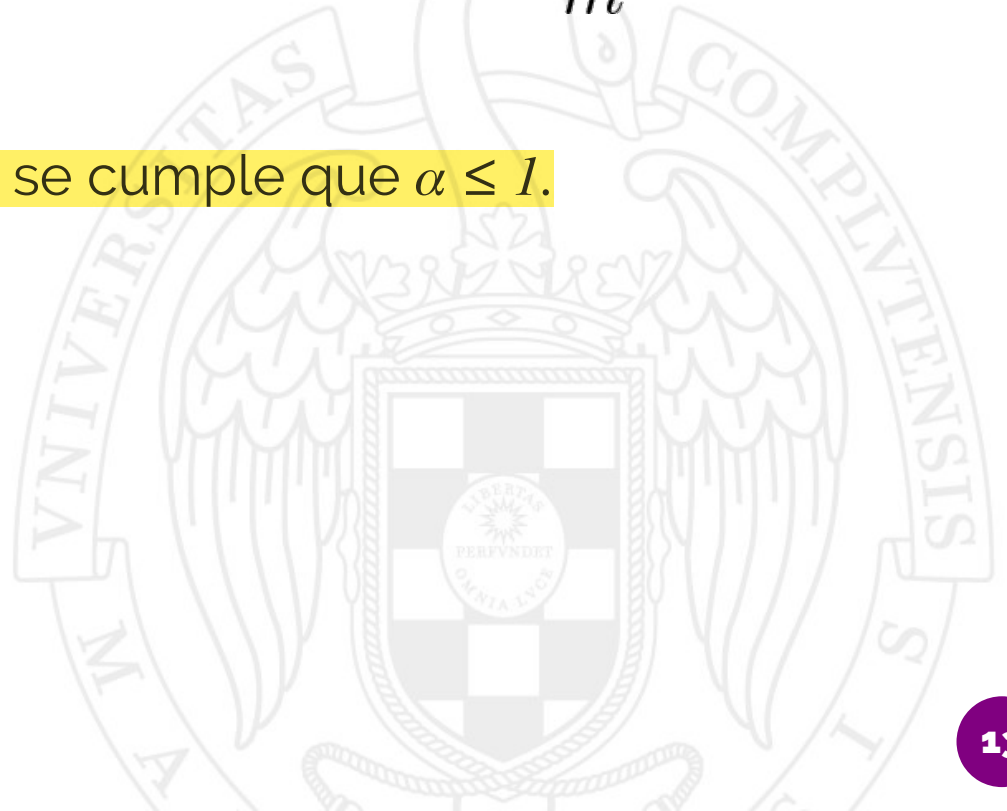
Una única entrada por cajón.

$n$  - número de entradas en la tabla

$m$  - número de cajones

$$\alpha = \frac{n}{m} \quad n \leq m$$

- En una *tabla hash* cerrada, siempre se cumple que  $\alpha \leq 1$ .

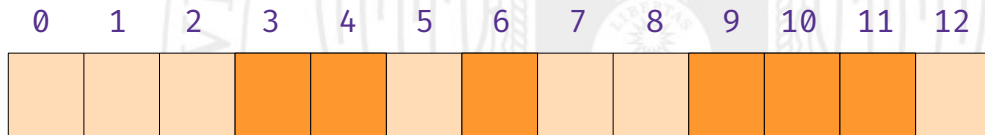


# Búsqueda de una clave (fallo)

- Sea  $N$  una variable aleatoria que denota el número de intentos infructuosos hasta que llegamos a un cajón vacío.

$N \geq 1$

$$P\{N \geq 1\} = \frac{n}{m}$$

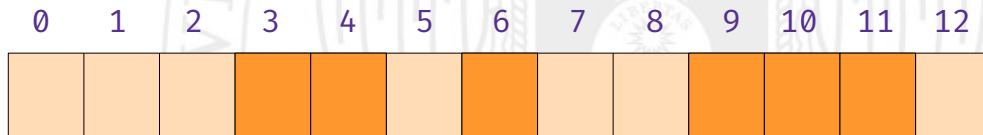


# Búsqueda de una clave (fallo)

- Sea  $N$  una variable aleatoria que denota el número de intentos infructuosos hasta que llegamos a un cajón vacío.

$$P\{N \geq 2\} = \frac{n}{m} * \frac{n-1}{m-1}$$

segunda posición también ocupada

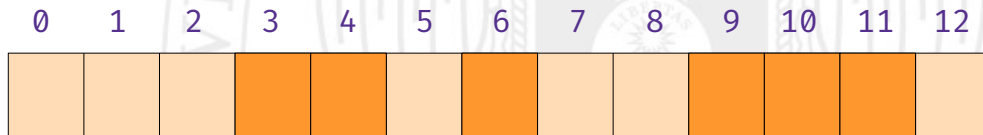


# Búsqueda de una clave (fallo)

- Sea  $N$  una variable aleatoria que denota el número de intentos infructuosos hasta que llegamos a un cajón vacío.

$$P\{N \geq 3\} = \frac{n}{m} * \frac{n-1}{m-1} * \frac{n-2}{m-2}$$

Probabilidad de que el primer intento sea fallido, por la de que el segundo sea fallido, por la de que el tercero sea fallido.





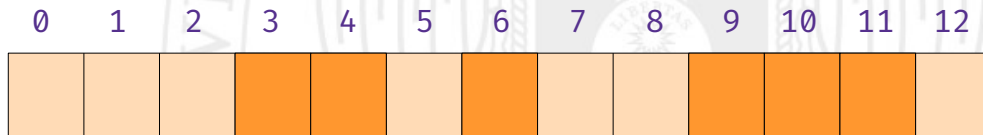
# Búsqueda de una clave (fallo)

- Sea  $N$  una variable aleatoria que denota el número de intentos infructuosos hasta que llegamos a un cajón vacío.

$$P\{N \geq i\} = \frac{n}{m} * \frac{n-1}{m-1} * \frac{n-2}{m-2} * \dots * \frac{n-i+1}{m-i+1}$$

Suponemos que el  $-1$ ,  $-2$   $-i+1$  es despreciable y no lo tenemos en cuenta. Tendríamos:

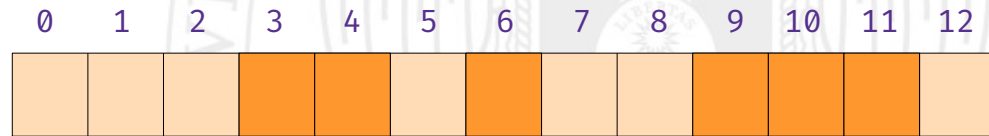
$$\left(\frac{n}{m}\right)^i \approx \alpha^i$$



# Búsqueda de una clave (fallo)

- Sea  $N$  una variable aleatoria que denota el número de intentos infructuosos hasta que llegamos a un cajón vacío.

$$P\{N \geq i\} \leq \alpha^i$$



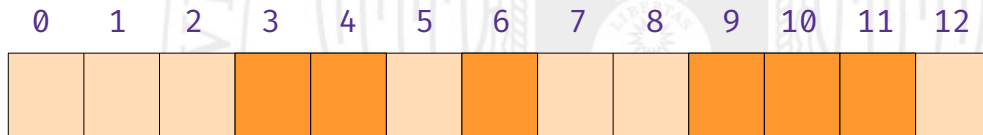
# Búsqueda de una clave (fallo)

- Promedio de cajones visitados para buscar una clave:

$$1 + E[N] = 1 + \sum_{i=1}^{\infty} P\{N \geq i\} \leq 1 + \sum_{i=1}^{\infty} \alpha^i = \sum_{i=0}^{\infty} \alpha^i = \frac{1}{1 - \alpha}$$

Búsqueda del cajón vacío

Valor esperado de N



# Búsqueda de una clave (fallo)

- Promedio de cajones visitados para buscar una clave:

$$1 + E[N] = 1 + \sum_{i=1}^{\infty} P\{N \geq i\} \leq 1 + \sum_{i=1}^{\infty} \alpha^i = \sum_{i=0}^{\infty} \alpha^i = \frac{1}{1 - \alpha}$$

Si tabla está llena al 90%

Sustituimos el 0.9 aquí

- Si  $\alpha = 0.9$ , entonces se visitan 10 cajones en el caso medio.
- Si  $\alpha = 0.8$ , entonces se visitan 5 cajones en el caso medio.
- Si  $\alpha = 0.5$ , entonces se visitan 2 cajones en el caso medio.

# Búsqueda de una clave (éxito)

- Promedio de cajones visitados para buscar una clave:

$$\frac{1}{\alpha} * \ln \frac{1}{1 - \alpha}$$

- Si  $\alpha = 0.9$ , entonces se visitan 2.56 cajones en el caso medio.
- Si  $\alpha = 0.8$ , entonces se visitan 2 cajones en el caso medio.
- Si  $\alpha = 0.5$ , entonces se visitan 1.38 cajones en el caso medio.

# Conclusión

- Si conseguimos mantener  $\alpha$  inferior a 1, el coste de las operaciones será constante.
- Con  $\alpha \leq 0.8$  se obtienen constantes razonables.
- ¿Cómo conseguimos mantener  $\alpha$  constante?
  - Haciendo que el número de cajones aumente proporcionalmente con el número de entradas → *Tabla dinámicamente redimensionable*.

# Bibliografía

- T. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein  
*Introduction to Algorithms (3ª edición)*  
The MIT Press (2009)  
Capítulo 11
- R. Peña  
*Diseño de Programas. Formalismo y Abstracción (3ª edición)*  
Pearson Educación (2005)  
Sección 8.1.3

