Perfect Hashing y Cuckoo Hashing

Guillermo Palma

Universidad Simón Bolívar
Departamento de Computación y T.I.

CI-2612: Algoritmos y Estructuras de Datos II

Plan

Sobre el Perfect Hashing

- Se quiere mejorar el tiempo del peor caso para la búsqueda para que pase de $O(n)$ a $O(1)$
- Una vez que las claves son insertadas en la tabla de hash, esta no puede ser modificada (tabla de hash estática)
- Ej. de aplicaciones:
  - Tabla con las palabras reservadas de un lenguaje de programación
  - Conjunto de archivos en un DVD o CD-ROM
  - Diccionarios ortográficos
- Se quiere que la tabla no use excesivo espacio para obtener un excelente rendimiento del peor caso

Sobre el Perfect Hashing

Definición

Perfect Hashing es la técnica de hashing que tiene un tiempo de $O(1)$ para el peor caso de la operación de búsqueda de claves.

Características del Perfect Hashing

- Se crea la tabla usando dos funciones de hash
- La primera función de hash asigna n claves a una de las m casillas
- Cada uno de los m casillas contiene una segunda tabla de hash S_j usando direccionamiento abierto
- La tabla S_j tiene asociada una función de hash h_j
- Sea n_j el número de elementos en S_j.
- El tamaño de S_j es $m_j = n_j^2$
- Las funciones de hash usadas pertenecen a la clase de funciones de hash universales
- El tiempo del peor caso para la búsqueda es $O(1)$
- El espacio total esperado a ser usado es $O(n)$

Ejemplo de Perfect Hashing

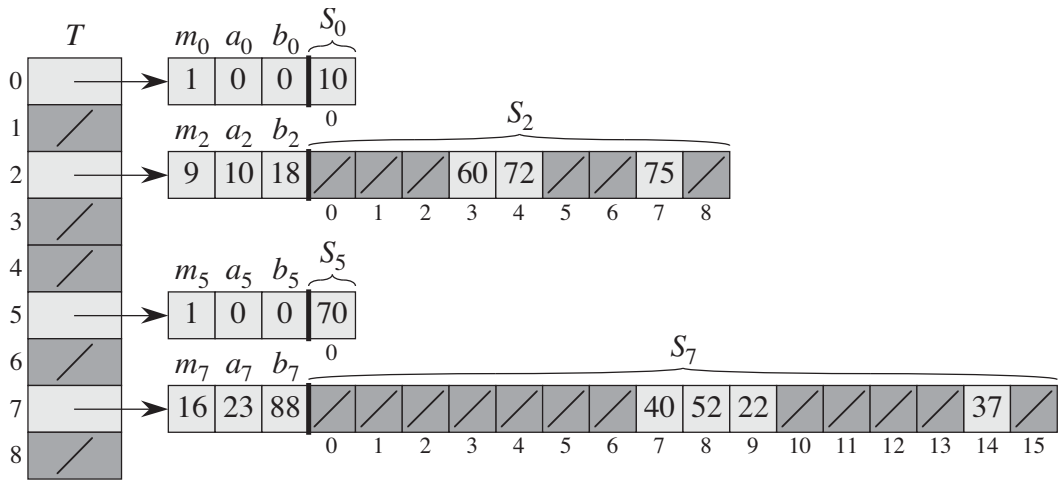

Figura: Perfect hashing para guardar el conjunto de claves $K = \{10, 22, 37, 40, 52, 60, 70, 72, 75\}$. Fuente [1]

Ejemplo de Perfect Hashing

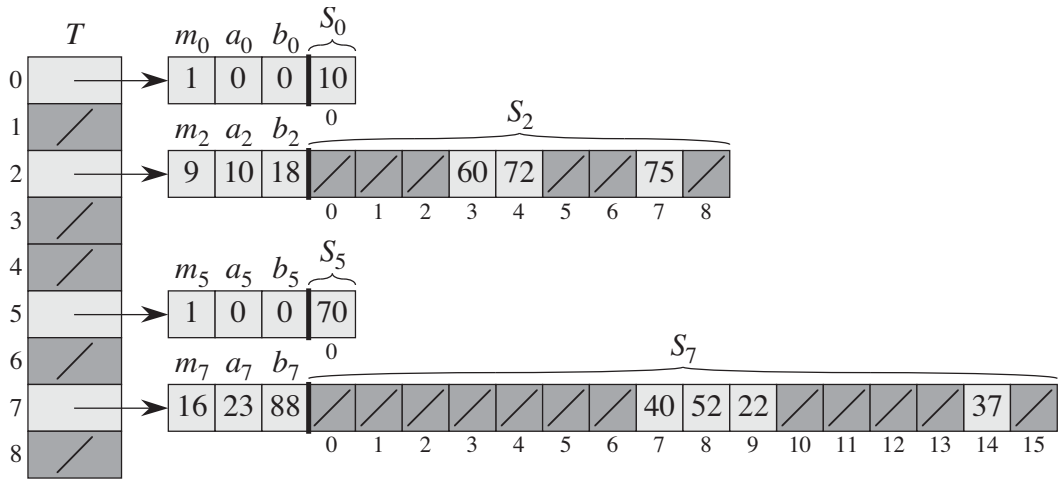

Figura: Perfect hashing para guardar el conjunto de claves $K = \{10, 22, 37, 40, 52, 60, 70, 72, 75\}$. Fuente [1]

- $h_1(k) = ((ak + b) \mod p) \mod m$. Si $a = 2$, $b = 42$ y $p = 101$, $h_1(75) = 7$

Ejemplo de Perfect Hashing

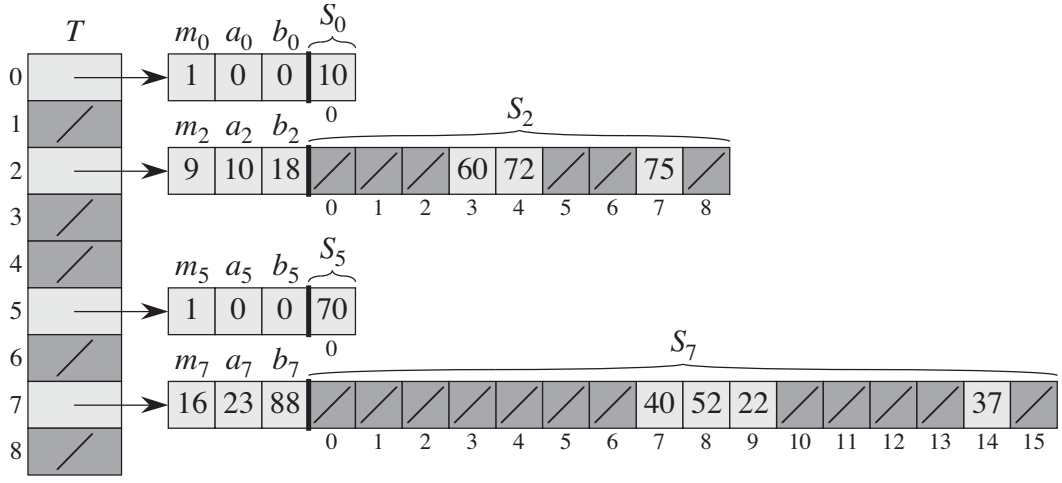

Figura: Perfect hashing para guardar el conjunto de claves $K = \{10, 22, 37, 40, 52, 60, 70, 72, 75\}$. Fuente [1]

- $h_1(k) = ((ak + b) \mod p) \mod m$. Si $a = 2$, $b = 42$ y $p = 101$, $h_1(75) = 2$
- $h_j(k) = ((a_j k + b_j) \mod p) \mod m_k$. Se tiene que $h_2(75) = 7$

Ejemplo de Perfect Hashing

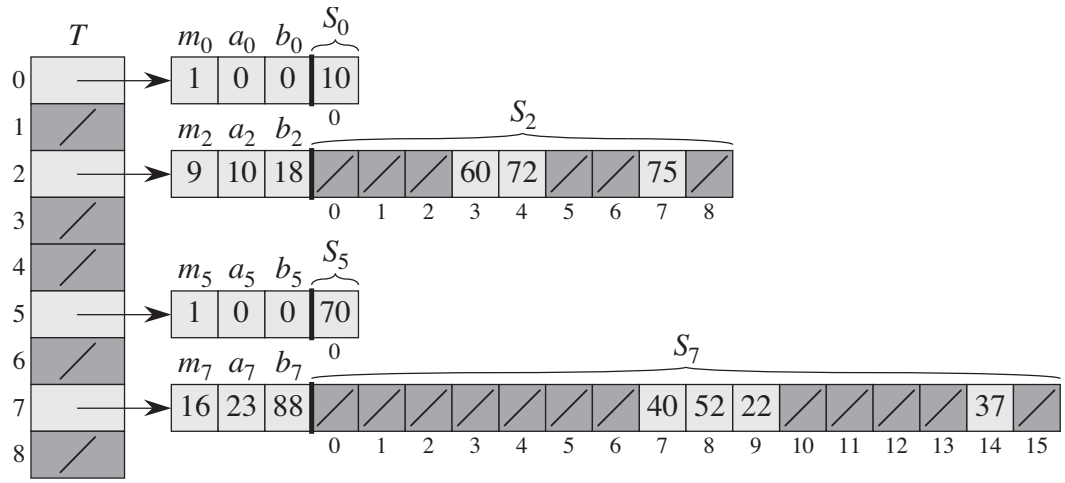

Figura: Perfect hashing para guardar el conjunto de claves $K = \{10, 22, 37, 40, 52, 60, 70, 72, 75\}$. Fuente [1]

- $h_1(k) = ((ak + b) \mod p) \mod m$. Si $a = 2$, $b = 42$ y $p = 101$, $h_1(75) = 2$
- $h_j(k) = ((a_j k + b_j) \mod p) \mod m_k$. Se tiene que $h_2(75) = 7$
- $m_j = n_j^2$

Teoremas del Perfect Hashing

Teorema

Si guardamos n claves en una tabla de hash de tamaño n^2 usando una función de hash universal, escogida aleatoriamente, entonces la probabilidad de una colisión es menor que $\frac{1}{2}$

Teorema

Si se almacenan n claves en una tabla de hash de tamaño $n = m$ usando una función de hash universal, escogida aleatoriamente, entonces

$$E[\sum_{j=0}^{n-1} n_j^2] < 2n$$

donde n_j es el número de claves almacenadas en la casilla j. Es decir, el espacio esperado a usar en la tabla de hash es menor que $2n$

Teoremas del Perfect Hashing

Corolario

Si se almacenan n claves en una tabla de hash de tamaño $n = m$ usando una función de hash universal, escogida aleatoriamente, y se configura el tamaño de las segundas tablas de hash S_j a $m_j = n_j^2$, entonces:

- El cantidad de almacenamiento requerido por todas las tablas secundarias es menor que $2n$
- La probabilidad de que el almacenamiento total usado por todas las tablas secundarias, exceda $4n$, es menor que $\frac{1}{2}$

Sobre el Cuckoo Hashing

Idea Básica

Se quiere tener una tabla de hash dinámica, que ene le peor caso de la búsqueda tenga un tiempo de $O(1)$, que el tiempo amortizado de la inserción sea $O(1)$ y el tiempo de eliminar un elemento sea en el peor caso sea $O(1)$. También se quiere que la implementación de las operaciones sea sencilla y el rendimiento en la práctica sea competitivo.

Características del Cuckoo Hashing

- Se mantienen dos tablas, cada una tiene m elementos
- Se escogen dos funciones de hash universales h_1 y h_2
- Cada clave a insertar debe estar en la casilla $h_1(x)$ o en la casilla $h_2(x)$
- El peor caso para la búsqueda es $O(1)$, porque solo se busca en dos casillas, una en cada tabla
- El peor caso para la eliminación es $O(1)$, porque solo se busca en dos casillas, una en cada tabla

Búsqueda en Cuckoo Hashing

Función CUCKOO-SEARCH(T_1, T_2, x)

inicio
  devolver $T_1[h_1(x)] = x \vee T_2[h_2(x)] = x$

Eliminación en Cuckoo Hashing

Procedimiento CUCKOO-ELIMINACION(T_1, T_2, x)

inicio

- **si** $T_1[h_1(x)] = x$ **entonces**
  - $T_1[h_1(x)] \leftarrow NIL$;
  - **devolver**
- **si** $T_2[h_2(x)] = x$ **entonces**
  - $T_2[h_2(x)] \leftarrow NIL$;
  - **devolver**

Inserción en Cuckoo Hashing

Procedimiento CUCKOO-INSERT(T_1, T_2, x)

inicio

- **si** `CUCKOO-SEARCH(`T_1, T_2, x`)` **entonces**
  - **devolver**
- **para** i **a** *MaxLoop* **hacer**
  - `SWAP(`x, $T_1[h_1(x)]$`)` ;
  - **si** $x = NIL$ **entonces**
    - **devolver**
  - `SWAP(`x, $T_2[h_2(x)]$`)` ;
  - **si** $x = NIL$ **entonces**
    - **devolver**
- `REHASH()` ;
- `CUCKOO-INSERT(`T_1, T_2, x`)` ;

Ejemplo de inserción en Cuckoo Hashing

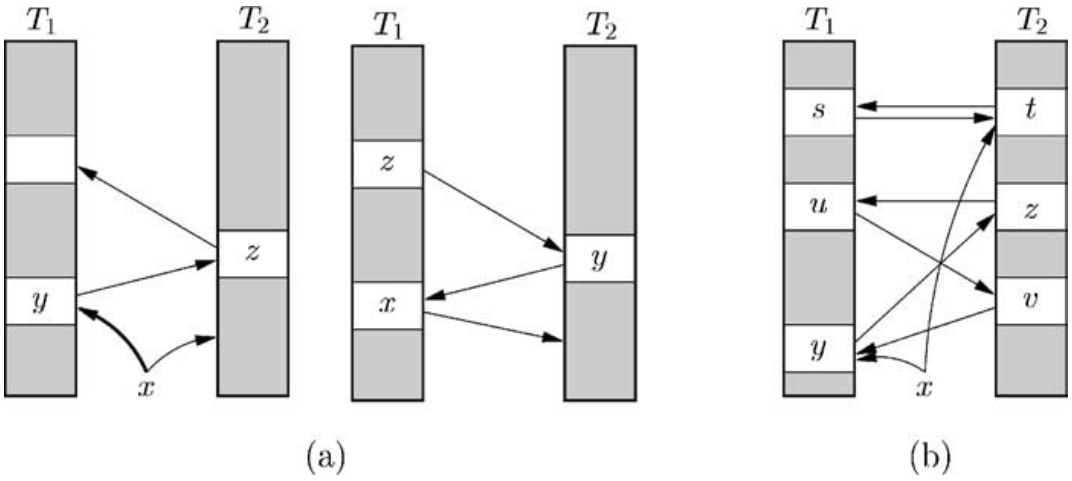

Figura: (a) Se inserta la clave x de forma exitosa. (b) La clave x no puede ser insertada. Fuente [2]

Ejemplo de inserción en Cuckoo Hashing

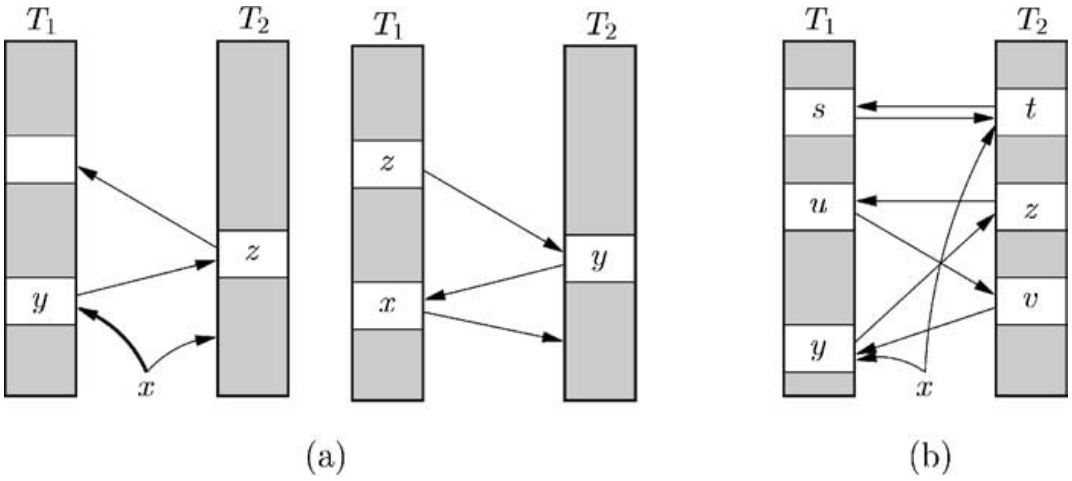

Figura: (a) Se inserta la clave x de forma exitosa. (b) La clave x no puede ser insertada Fuente [2]

Ejemplo de inserción en Cuckoo Hashing

	20	50	53	75	100	67	105	3	36	39
h_1(key)	9	6	9	9	1	1	6	3	3	6
h_2(key)	1	4	4	6	9	6	9	0	3	3

Figura: Claves y sus valores de hash

Ejemplo de inserción en Cuckoo Hashing

table[1]	-	-	-	-	-	-	-	-	-	20	-
table[2]	-	-	-	-	-	-	-	-	-	-	-

Figura: Se inserta la clave 20

table[1]	-	-	-	-	-	-	50	-	-	20	-
table[2]	-	-	-	-	-	-	-	-	-	-	-

Figura: Se inserta la clave 50

Ejemplo de inserción en Cuckoo Hashing

.

table[1]	-	-	-	-	-	-	50	-	-	53	-
table[2]	-	20	-	-	-	-	-	-	-	-	-

Figura: Se inserta la clave 53

table[1]	-	-	-	-	-	-	50	-	-	75	-
table[2]	-	20	-	-	53	-	-	-	-	-	-

Figura: Se inserta la clave 75

Ejemplo de inserción en Cuckoo Hashing

table[1]	-	100	-	-	-	-	50	-	-	75	-
table[2]	-	20	-	-	53	-	-	-	-	-	-

Figura: Se inserta la clave 100

table[1]	-	67	-	-	-	-	50	-	-	75	-
table[2]	-	20	-	-	53	-	-	-	-	100	-

Figura: Se inserta la clave 67

Ejemplo de inserción en Cuckoo Hashing

table[1]	-	67	-	-	-	-	105	-	-	53	-
table[2]	-	20	-	-	50	-	75	-	-	100	-

Figura: Se inserta la clave 105

Experimentos con Cuckoo Hashing I

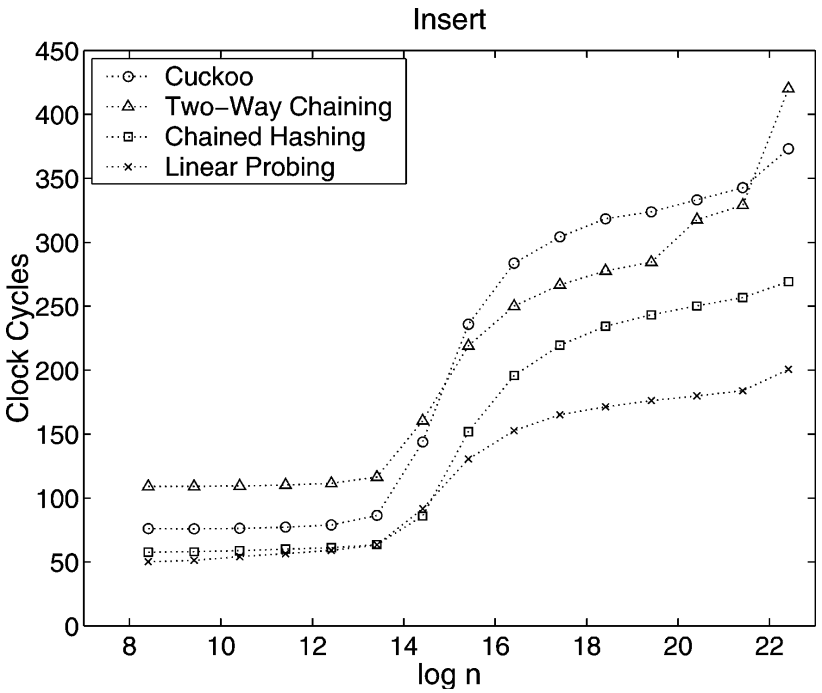

Figura: Tiempo de inserción de elementos en las tablas. Fuente [2]

Experimentos con Cuckoo Hashing II

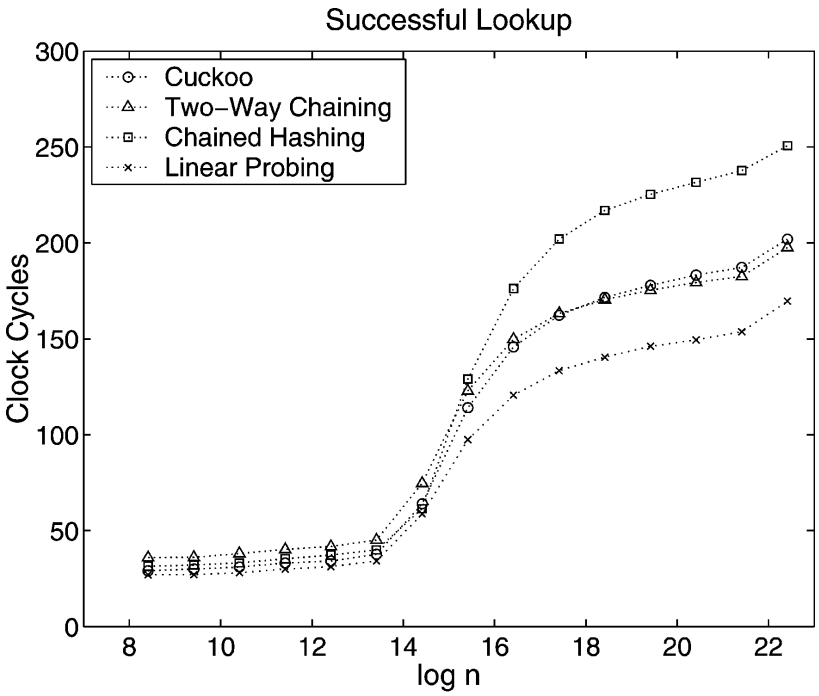

Figura: Tiempo de búsquedas de elementos que se encuentran en las tablas. Fuente [2]

Experimentos con Cuckoo Hashing III

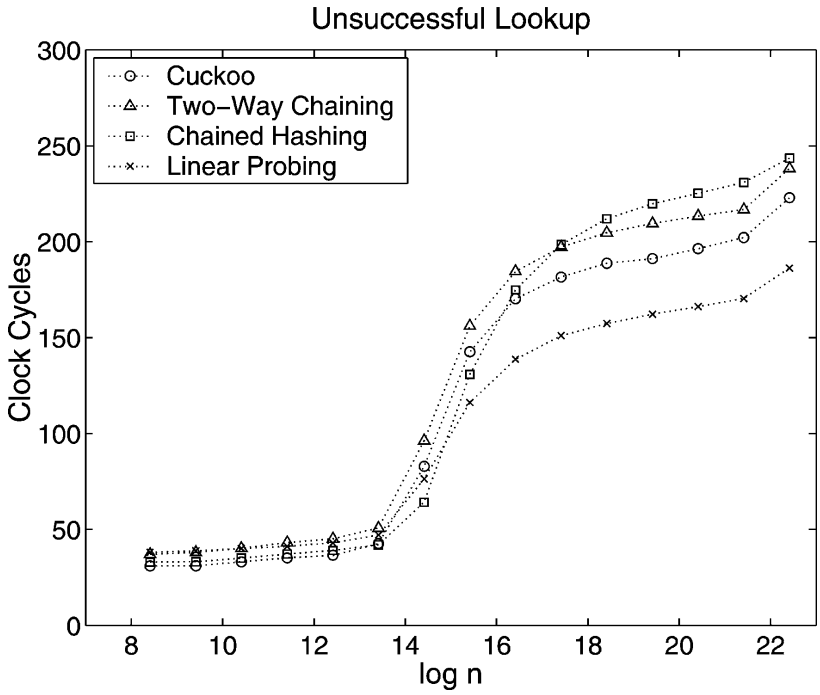

Figura: Tiempo de búsquedas de elementos que no se encuentran en las tablas. Fuente [2]

Experimentos con Cuckoo Hashing IV

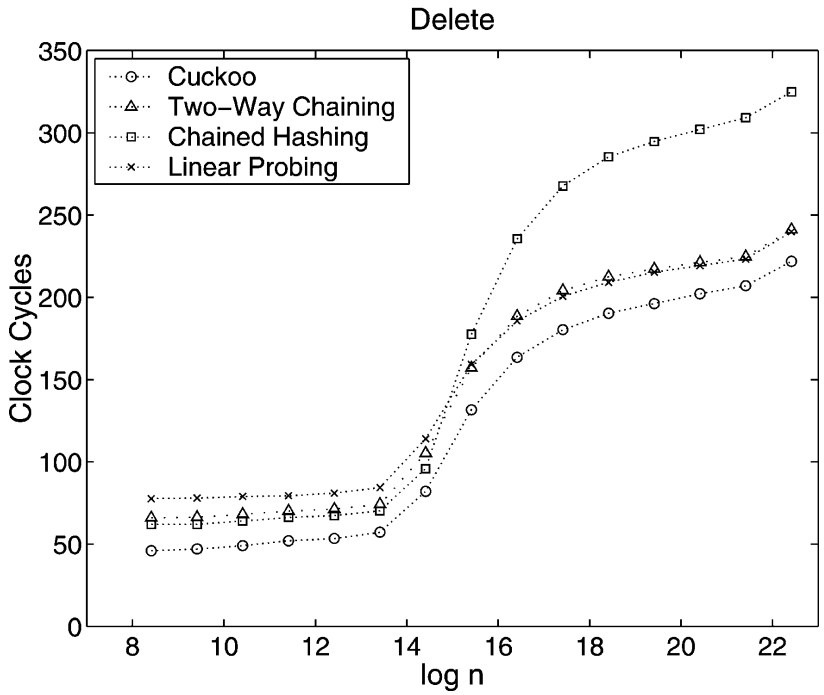

Figura: Tiempo de eliminación de elementos en las tablas. Fuente [2]

Análisis general del Cuckoo Hashing

- El tiempo del peor caso para la búsqueda y eliminación es $O(1)$
- Las inserciones tienen un tiempo amortizado de $O(1)$
- Las contantes ocultas son pequeñas, es práctico construir este tipo de tablas.

Referencias


T. Cormen, C. Leirserson, R. Rivest, and C. Stein.
Introduction to Algorithms.
McGraw Hill, 3ra edition, 2009.


Rasmus Pagh and Flemming Friche Rodler.
Cuckoo hashing.
Journal of Algorithms, 51(2):122–144, 2004.