Taller de Hash

Martin *March* Miguel

Algoritmos y Estructruas de Datos II

19 de octubre de 2016

Objetivos del taller

- Repasar tablas de hash y funciones de hash
- Repasar propiedades y características de la función
- Conocer detalles de sus usos prácticos y usos en otros dominios
- Irnos con una implementación hecha de una tabla de hash

Tablas de hash

Tablas de hash como implementación para `dicc(U, S)` con `Definir`, `Definido` y `Obtener` en $\theta(1)$ promedio.

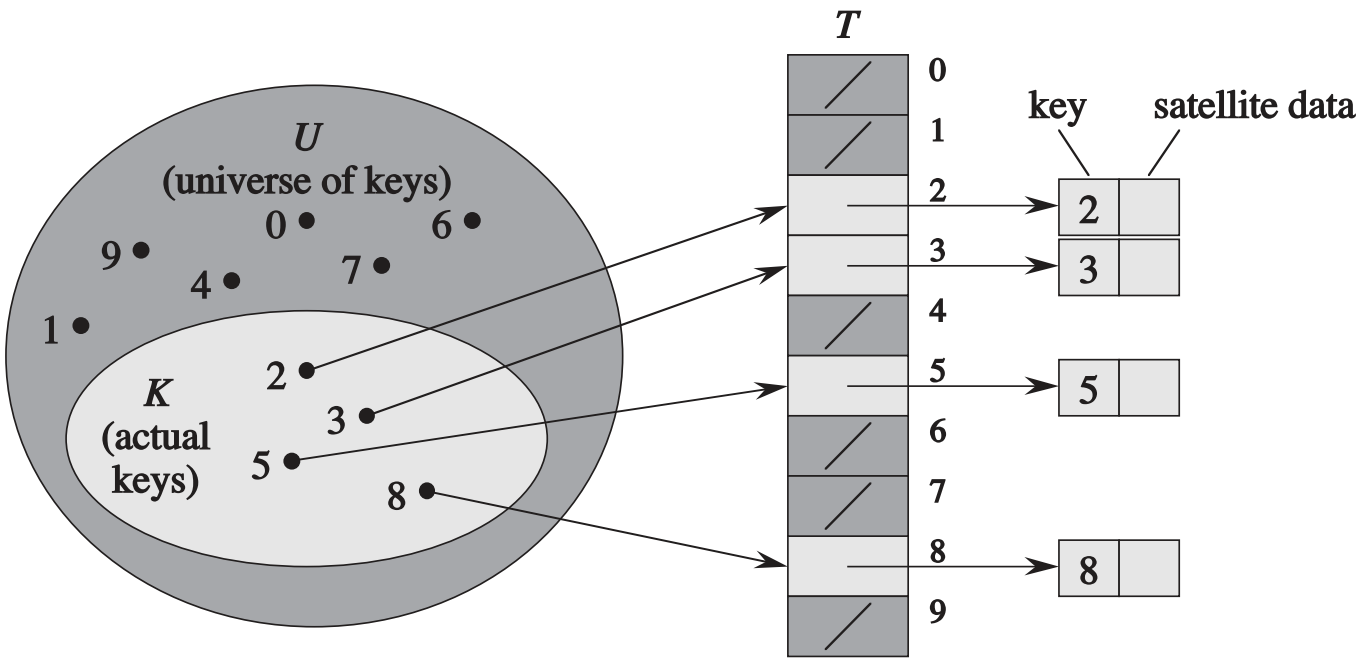


Figura 1: Tabla de Hash. Extraído del Cormen

Definiciones

- U: Universo de claves posibles. $x \in U$
- K: Conjunto de claves realmente utilizadas.
- n: Cantidad de claves utilizadas. $n = |K|$.
- m: Tamaño de la tabla
- h: U $\rightarrow$ [0 .. m-1] Función de hash
- $n_{h(k)} = |T[h(k)]|$: Cantidad de elementos en la lista

Los términos también están extraídos del Cormen

Función de Hash

Dado que voy a usar el resultado de h para definir y obtener, necesito que siempre me de el mismo valor (que sea función).

$$a = b \implies h(a) == h(b)$$

Nos arremangamos

¿Cuál es el costo real de las operaciones de una tabla de hash?

$$
\begin{aligned}
c(T, m, h, k) = & \\
& c(h(k)) && \text{costo de la función de hash para la clave k} \\
& + 1 && \text{costo de acceder a la lista} \\
& + \theta(|T[h(k)]|) && \text{costo de buscar en la lista}
\end{aligned}
$$

Hashing Uniforme Simple

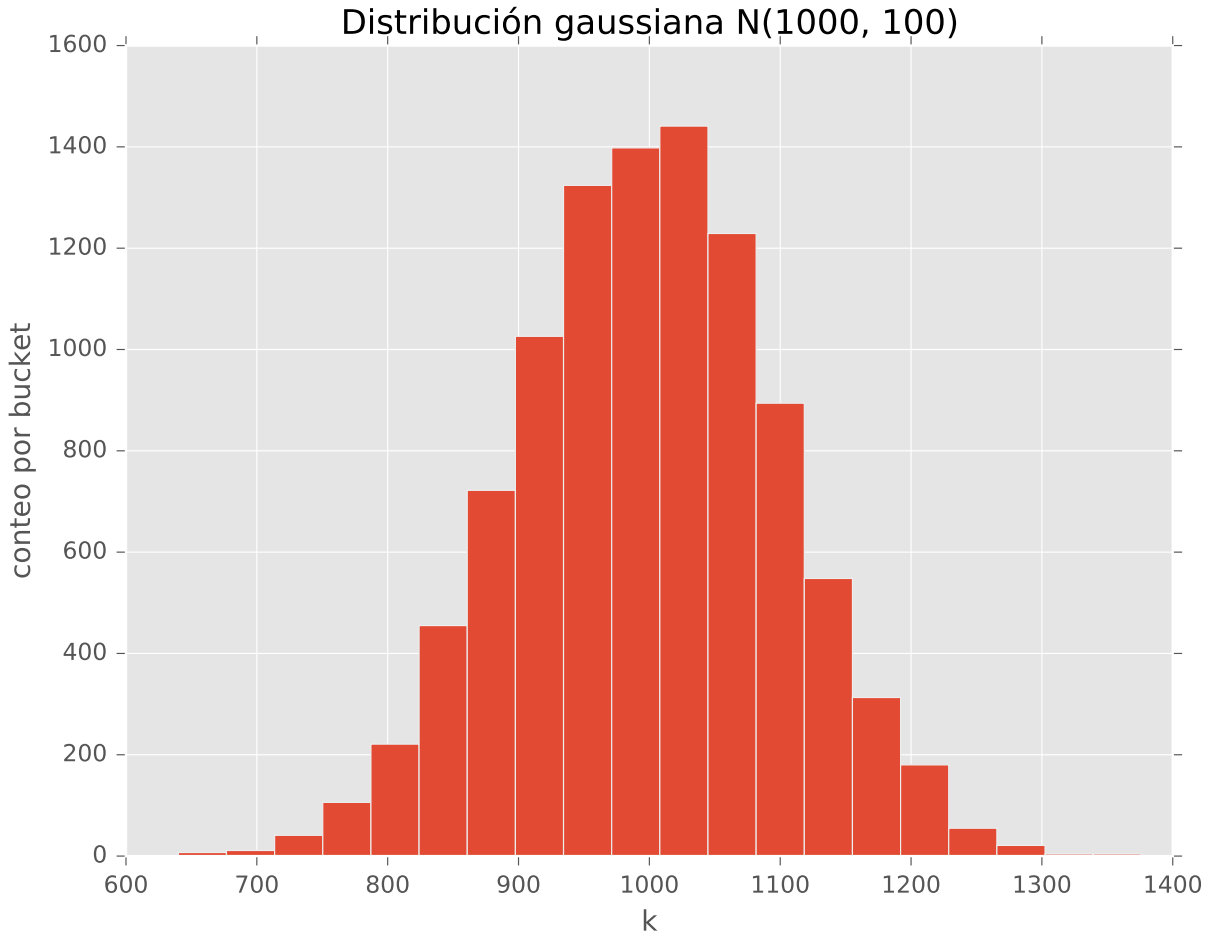
Si ...

$$P(h(k)|K) \sim U(0, m-1)$$

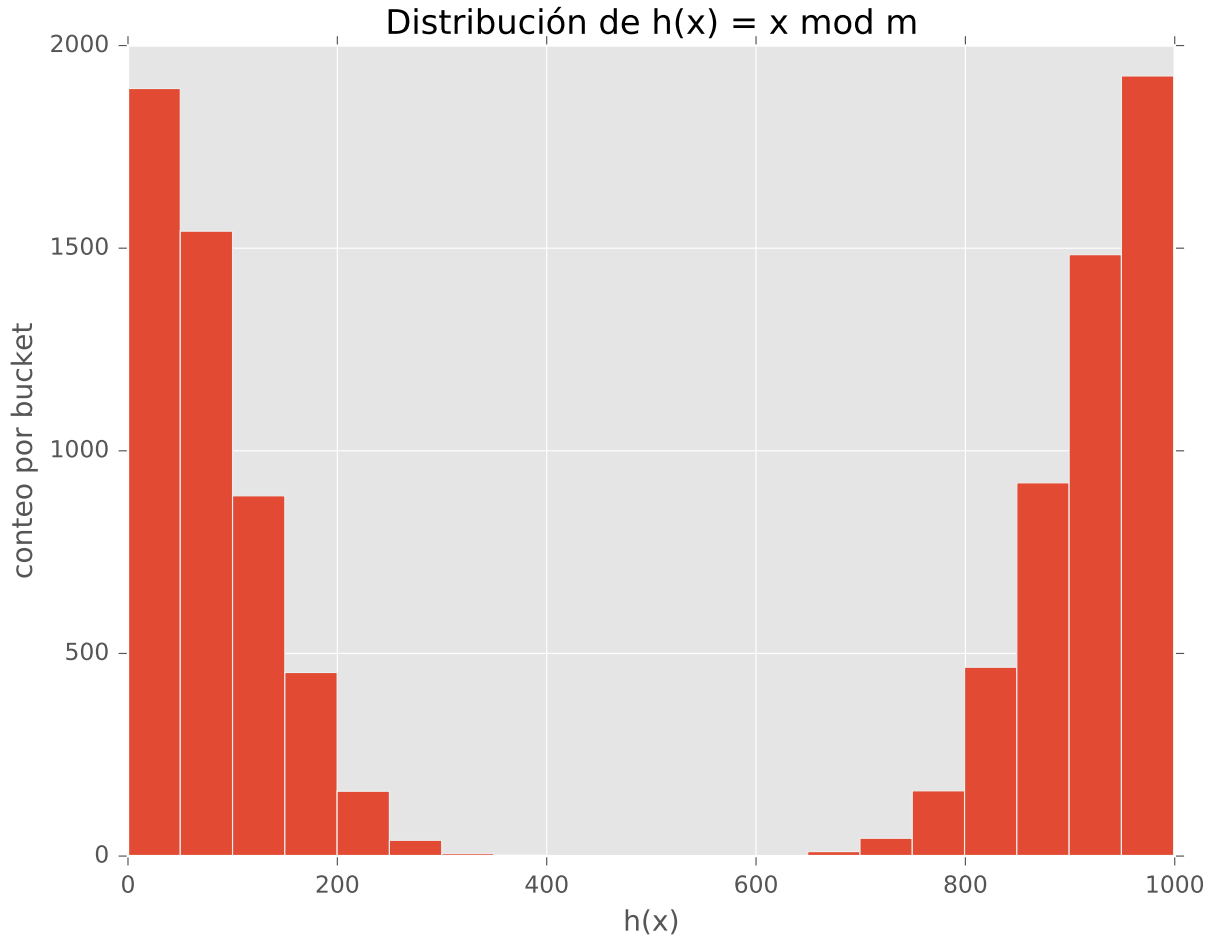
Entonces el largo promedio de las listas ($E[T[h(k)]]$) es

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

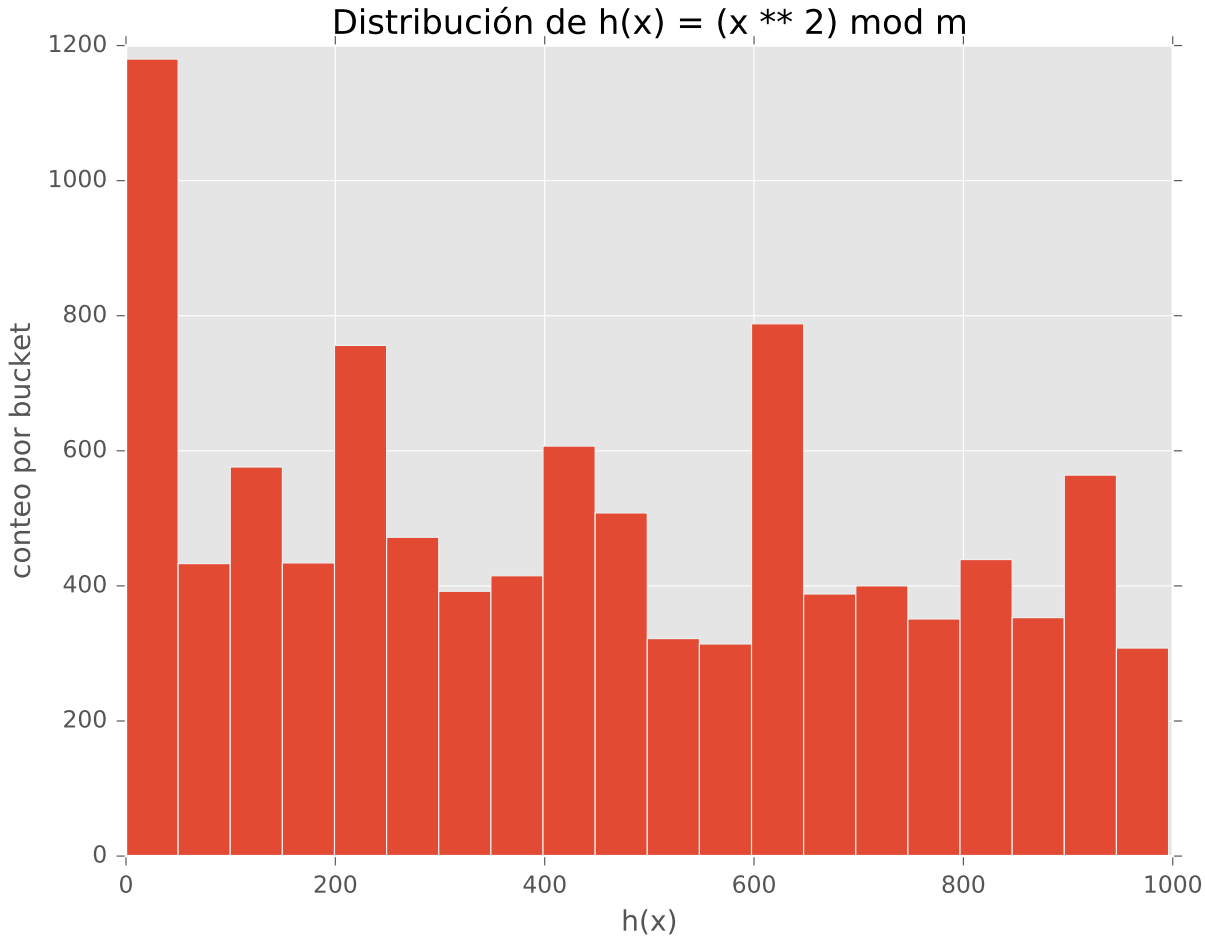
Ejemplo. Claves gaussianas



$h(k) = k \bmod m$


Distribución de h(x) = x mod m
conteo por bucket
h(x)
0
500
1000
1500
2000
0
200
400
600
800
1000

h(k) = (k ** 2) mod m (Mid-square method)



Welcome, to the real world.

Las tablas de hash se usan como estructura subyacente para muchos diccionarios de los lenguajes de programación. En python, la estructura asociativa por excelencia `dict` tiene una tabla de hash interna. En java, todo tipo de contenedores están implementados sobre tabla de hash (HashSet, HashMap).

¿Cómo hago para hashear un objeto de un tipo cualquiera? Pensemos en las clases que nosotros armamos en C++. Una opción es usar su dirección de memoria (si es que esta no va a cambiar). Así tenemos un número con el que trabajar como estábamos acostumbrados.

Equivalente, igual y lo mismo

¿Puedo hashear cualquier objeto? ¿Cuando dos instancias son iguales? ¿Cuando es la misma?

	Igual (obs)	lo mismo	h(k)
C++	operator==	comparar punteros	Hash
Java	equals*	==	hashCode*
Python	__eq__*	is	__hash__*

Cuadro 1: Implementaciones de equivalencias según lenguajes

Si sobreescribo la igualdad, tengo que definir `h` congruente. Si no tengo una igualdad bien definida, no debería definir una función de hash.**

---

*Tiene implementación por defecto en el lenguaje

**`https://docs.python.org/2/reference/datamodel.html#object.__hash__`

Otros usos

- Calcular intersección de conjuntos
- Heurística de búsqueda de substrings[1]
- Fingerprinting[2]
  - CDDB: `https://en.wikipedia.org/wiki/CDDB`

[1] `https://en.wikipedia.org/wiki/Rabin-Karp_algorithm`
[2] `https://en.wikipedia.org/wiki/Fingerprint_(computing)`

Otros usos

Función de hash criptográfica:

- Dado d = h(k), difícil averiguar k
- Difícil encontrar a != b / h(a) == h(b)
- Si a y b son levemente distintas, h(a) es muy distinto de h(b)

Usos:

- Se usa para no transmitir contraseñas en texto plano.
- Se usa para verificar la integridad de transmisiones.

Ejemplos:

- *SHA-1*: `https://en.wikipedia.org/wiki/SHA-1`