Escuela de Ingeniería Civil Informática
Facultad de Ingeniería

Estructuras de datos

Capítulo IV: Hashing

Fabián Riquelme Csori
fabian.riquelme@uv.cl

2017-II

Index

Arreglo asociativo

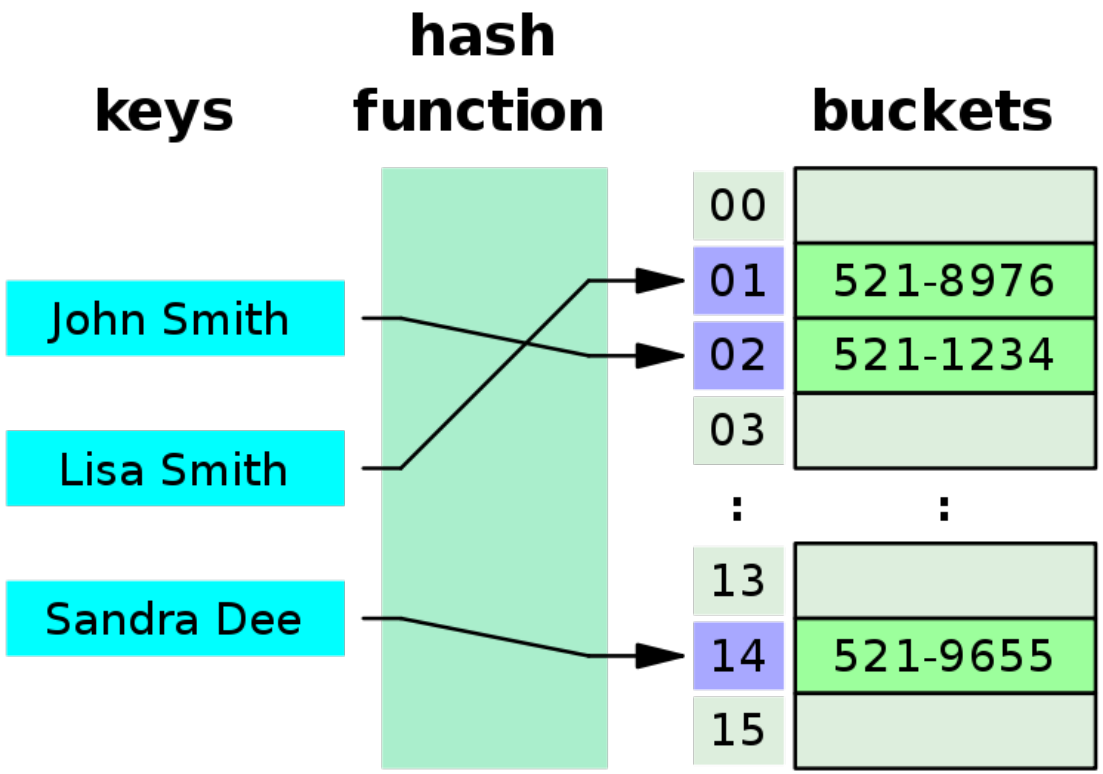
associative array	Collection of (key, value) pairs, such that each possible key appears at most once in the collection.
operations	
add(key,value)	Add a new pair to the collection.
remove(key)	Remove the pair from the collection associated to the key.
reassign(key,value)	Replace the value of an existing pair associated to the key.
lookup(key)	Find the value of the pair associated with a given key.

- Los arreglos asociativos también se conocen como vectores asociativos, mapas, tablas de símbolos o diccionarios.
- Las estructuras de datos más comunes que se implementan para manipularlos son las **tablas hash** y los árboles de búsqueda.

Arreglo asociativo: implementación

		Hash table	Self-balancing binary search tree
add	average	$O(1)$	$O(\log n)$
	worst case	$O(n)$	$O(\log n)$
remove	average	$O(1)$	$O(\log n)$
	worst case	$O(n)$	$O(\log n)$
lookup	average	$O(1)$	$O(\log n)$
	worst case	$O(n)$	$O(\log n)$

Tablas Hash



[Jorge Stolfi, Wikipedia]

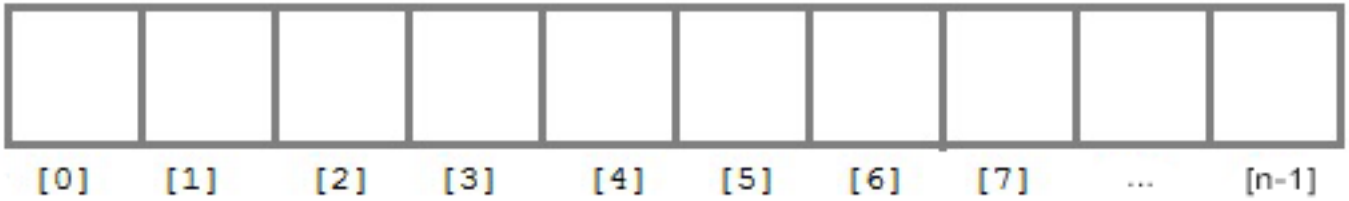
- Si la función de hash es inyectiva se habla de hashing perfecto.
- Usualmente la tabla hash debe lidiar con colisiones.

Tablas Hash: Aplicaciones

- El hashing se usa para almacenar muchos datos sobre los que se realizarán operaciones de búsqueda e inserción.

  Ej: diccionarios, indexación de BDs, conjuntos, cachés, criptografía, etc.
- Los datos se almacenan en posiciones pseudo-aleatorias, no están ordenados (a diferencia de los árboles), y recorrerlos por orden es lento (e inservible).
- Es una estructura de datos tan usada que varios lenguajes la incluyen como tipos básicos o librerías estándar.
- `http://www.sha1-online.com/`

Ejemplo



- Considere una función de hash tal que value=table[key % n].
  - Si $n = 10$, ¿qué pasa con 143,674 y 645,394?
- Alternativa: suma de dígitos
  - $143{,}674 = 1 + 4 + 3 + 6 + 7 + 4 = 25$
  - $645{,}394 = 6 + 4 + 5 + 3 + 9 + 4 = 31$
  - ...¿problemas?
- Hashing doble:
  - $25 = 2 + 5 = 7$
  - $31 = 3 + 1 = 4$
- El problema de colisiones se mantiene... ¿soluciones?

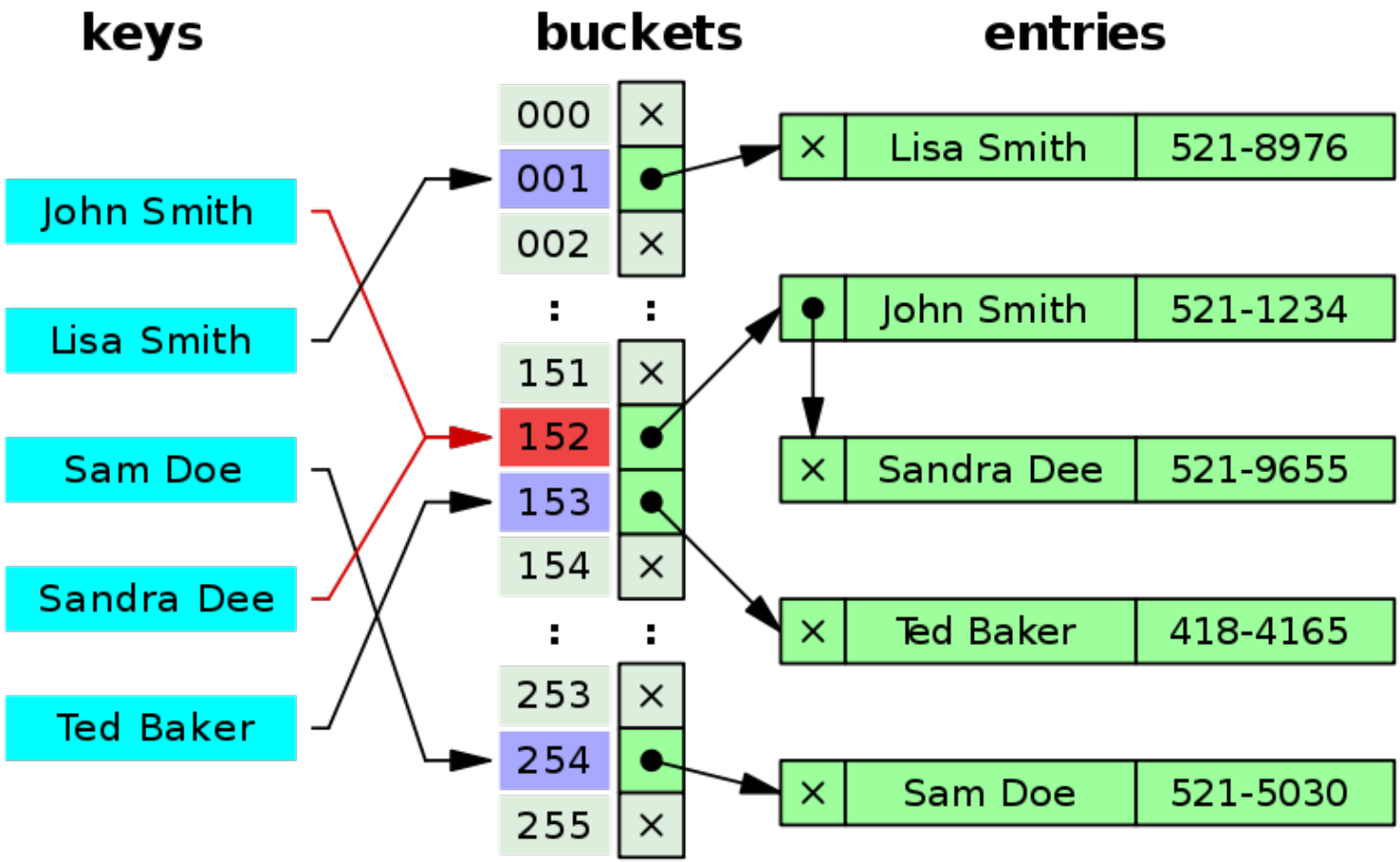
Corrección de colisiones: hashing cerrado

- Linear probing: Si la posición está ocupada, se ocupa la siguiente vacía.
- Ejemplo:

key	hash	index	lin.prob
1	$1 \% 20 = 1$	1	1
2	$2 \% 20 = 2$	2	2
42	$42 \% 20 = 2$	2	3
4	$4 \% 20 = 4$	4	4
12	$12 \% 20 = 12$	12	12
14	$14 \% 20 = 14$	14	14
17	$17 \% 20 = 17$	17	17
13	$13 \% 20 = 13$	13	13
37	$37 \% 20 = 17$	17	18

- ¿Problemas? Arruina la eficiencia: $O(1) \rightarrow O(n)$
- Variaciones: quadratic overflow, etc.

Corrección de colisiones: hashing abierto



Uso de listas enlazadas

[Jorge Stolfi, Wikipedia]

Buenas prácticas

Función de hash

- Debe ser lo más simple posible: $O(1)$
- Para un gran número de claves, debe disponer los valores lo más distantes posible.

Tabla de hash

- Si es muy pequeña, aumentará el número de colisiones.
  - o bien el uso de listas enlazadas aumentará la complejidad.
- Si es muy grande, aumentará innecesariamente el uso de memoria.

Buenas alternativas son las llamadas funciones hash criptográficas.

Ejercicios (1)

- Busque códigos que implementen una tabla hash
  - C, C++, Python, Java, otros...
- Observe los códigos e identifique:
  - ¿qué función de hash implementan?
  - ¿cómo manejan las colisiones?

Ejercicios (2)

Dadas las siguientes claves:

$$12, 44, 13, 88, 23, 94, 11, 39, 20, 16, 5$$

y la función de hash $h(i) = (2i + 5) \text{mod } 11$.

- Dibuje una tabla hash de 11 posiciones de memoria que represente el hashing con un manejo de colisiones usando listas enlazadas.
- Repita el ejercicio anterior, pero usando linear probing.
- Repita el ejercicio anterior, pero usando quadratic probing.
- Implemente cada una de las soluciones.

Ejercicios (3)

Busque y responda:

- ¿Qué se entiende por *rehashing* y cuando se aplica?
- ¿Qué es el *load factor*?
- Como formas de manejo de colisiones hemos visto *linear probing*, *quadratic probing* y uso de listas enlazadas. ¿En qué consiste el *double hashing*?

Bibliografía recomendada

- Weiss, M., Estructura de datos y algoritmos, Addison-Wesley, 1995.
- Aho, Hopcroft y Ullman, Estructuras de datos y algoritmos, Addison-Wesley, 1988.

Recursos

- Wikimedia Commons.