

# Tablas de hash basadas en direccionamiento abierto

Guillermo Palma

Universidad Simón Bolívar  
Departamento de Computación y T.I.

CI-2612: Algoritmos y Estructuras de Datos II



(USB)

Tablas de hash direccionamiento abierto

CI-2612 enero-marzo 2020

1 / 19

## Plan

- 1 Direccionamiento abierto
- 2 Funciones de hash para el direccionamiento abierto



(USB)

Tablas de hash direccionamiento abierto

CI-2612 enero-marzo 2020

2 / 19

## Características de las tablas de hash basadas en direccionamiento abierto

- La idea es guardar todas las claves (o el par clave, valor) en la misma tabla
- Se tiene que el tamaño de la tabla es mayor que el numero de elementos a agregar, esto es  $m > n$
- No hay necesidad de usar listas enlazadas
- La inserción consiste en recorrer la tabla en busca de una casilla disponible
- Para la búsqueda se recorre la tabla de la misma manera que la inserción para encontrar un elemento
- Borrar un elemento es complicado, porque hay que hacer una marca especial en la casilla en donde se encontraba el elemento eliminado
- El tiempo de la búsqueda de la inserción depende de la longitud de la secuencia de elementos probados hasta alcanzar una casilla libre (probe sequence)



## Ejemplo de direccionamiento abierto

0	
1	79
2	
3	
4	69
5	98
6	
7	72
8	
9	14
10	
11	50
12	

**Figura:** Ejemplo de la inserción del elemento 14, en una tabla que contiene a los elementos 79, 69, 98, 72 y 50. En la inserción la probe sequence es  $\{1, 5, 9\}$ . Fuente [1]



## Función de hash del direccionamiento abierto

- La función de hash tiene dos argumentos, primero la clave  $k$  del objeto a insertar y segundo el número de prueba (probe number)  $i$

$$h(k, i), \quad \text{donde } i = 0, 1, \dots, m - 1$$

- Secuencia de prueba (Probe sequence)

$$\langle h(k, 0), h(k, 1), \dots, h(k, m - 1) \rangle$$

- La secuencia de prueba deber ser una permutación de la secuencia de las  $m$  casillas, es decir, permutación de  $\{0, 1, \dots, m - 1\}$
- Existen  $m!$  secuencias de prueba
- Una buena función de hash debe ser capaz de producir todas las  $m!$  posibles secuencias de prueba



## Inserción en el direccionamiento abierto

---

### Procedimiento HASH-INSERT( $T, k$ )

---

#### inicio

```

     $i \leftarrow 0$ ;
    repetir
         $j \leftarrow h(k, i)$  ;
        si  $T[j] = NIL$  entonces
             $T[j] \leftarrow k$  ;
            devolver  $j$ ;
        en otro caso
             $i \leftarrow i + 1$  ;
    hasta que  $i = m$ ;
    imprimir (Desbordamiento de la tabla de hash) ;

```

---



# Búsqueda en el direccionamiento abierto

---

## Procedimiento HASH-SEARCH( $T, k$ )

---

**inicio**

```

 $i \leftarrow 0$ ;
repetir
     $j \leftarrow h(k, i)$ ;
    si  $T[j] = k$  entonces
        devolver  $j$ ;
     $i \leftarrow i + 1$ 
hasta que  $T[j] = NIL \vee i = m$ ;
devolver  $NIL$ ;
  
```

---



## Principales funciones de hash en el direccionamiento abierto

- Linear probing
- Quadratic probing
- Double hashing

Ninguna de estas funciones es capaz de generar más de  $m^2$  secuencias de prueba



## Linear probing

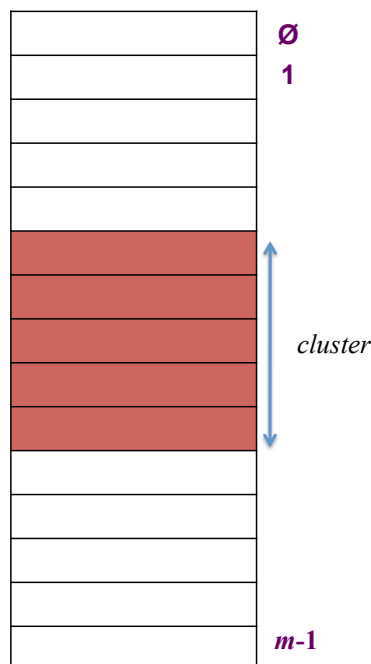
- Consiste en que cuando hay una colisión, se prueba la próxima posición en la tabla
- Viene dada por la función:

$$h(k, i) = (h_1(k) + i) \text{ mód } m, \quad \text{donde } i = 0, 1, \dots, m-1$$

- La secuencia de prueba es  $\{h_1(k), h_1(k) + 1, h_1(k) + 2, \dots\}$
- Solo puede generar  $m$  secuencias de prueba



## Ejemplo de *clustering* en la inserción con linear probing



## Eliminación con Linear probing

- Si se marca la casilla a eliminar como vacía, no es posible obtener las claves después que la casilla fue ocupada
- Se marca la casilla a eliminar con un valor centinela como **Borrado**
- Las casillas marcadas como **Borrado** pueden ser usadas para insertar un nuevo elemento
- Con esta estrategia es posible hacer la búsqueda de todas las claves



## Búsqueda con Linear probing

- Casos posibles:
  - ▶ La casilla está ocupada con un elemento con distinta clave a la buscada
  - ▶ La casilla está ocupada con un elemento con igual clave a la buscada
  - ▶ La casilla está vacía
  - ▶ La casilla está marcada como **Borrado**
- La búsqueda se termina cuando se encuentra una casilla vacía, cuando se encuentra una casilla con igual clave o cuando se hace toda la secuencia de prueba



## Quadratic probing

- Se usa una función de hash auxiliar  $h_1$  y dos constantes  $c_1$  y  $c_2$
- Viene dada por la función:

$$h(k, i) = (h_1(k) + c_1 i + c_2 i^2) \text{ mód } m, \quad \text{donde } i = 0, 1, \dots, m-1$$

- Solo puede generar  $m$  secuencias de prueba



## Double hashing

- Se usa una primera función de hash  $h_1$  para determinar la primera casilla
- Se usa una segunda función de hash  $h_2$  para determinar el incremento de la secuencia de prueba
- Viene dada por la función:


$$h(k, i) = (h_1(k) + h_2(k)i) \text{ mód } m, \quad \text{donde } i = 0, 1, \dots, m-1$$

- Primer elemento de la secuencia de prueba es  $h_1(k)$
- Tiene la ventaja que evita el agrupamiento (*clustering*) de elementos en una parte de la tabla
- Hace que la operación de eliminar un elemento de la tabla sea más difícil
- Es la función más usada de las tres
- Puede generar un máximo de  $m^2$  secuencias de prueba



## Análisis del caso en que la clave no se encuentre en la tabla

- El factor de carga (load factor), llamado  $a$ , es el número de claves en la tabla dividido entre el número de casillas de la tabla
- La probabilidad de que la casilla este ocupada  $a$
- La probabilidad de que la casilla este vacía  $1 - a$
- Probabilidad de finalizar las pruebas de búsqueda (Probe) en dos pasos  $a(1 - a)$
- Probabilidad de finalizar las pruebas de búsqueda (Probe) en  $k$  pasos  $a^{k-1}(1 - a)$
- El número de intentos esperados a realizar en una búsqueda de una clave que no está en la tabla es:

$$E(\#intentos) = \sum_{k=1}^{\infty} ka^{k-1}(1 - a) = \sum_{k=1}^{\infty} ka^{k-1}(1 - a) = \frac{1}{1 - a}$$


## Análisis del caso en que la clave se encuentre en la tabla

El número de intentos esperados a realizar en una búsqueda de una clave que sí está en la tabla es:

$$E(\#intentos) = \frac{1}{a} \ln\left(\frac{1}{1 - a}\right)$$





# Referencias



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

