Ayudantía 6: Hash

Ziqi Yan : zyan@uc.cl
Fernando Torres : fernando.torres@uc.cl

Contenidos

- Repaso de Hash
- Problemas

¿Qué es Hash?

Tabla de Hash

- **Asociar** un valor con una clave
- **Obtener** el valor asociado a una clave

Tabla de Hash

- **Asociar** un valor con una clave
- **Obtener** el valor asociado a una clave

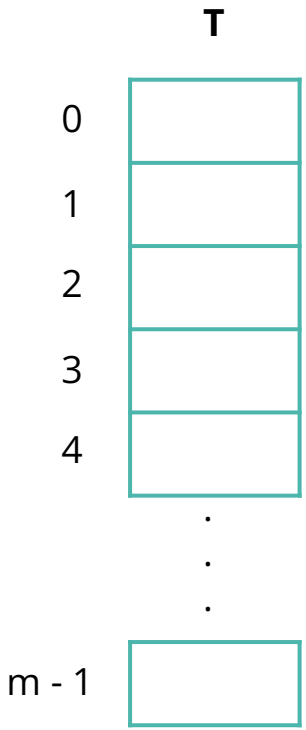


Tabla de Hash

T [k] = ∅

T [k] != ∅

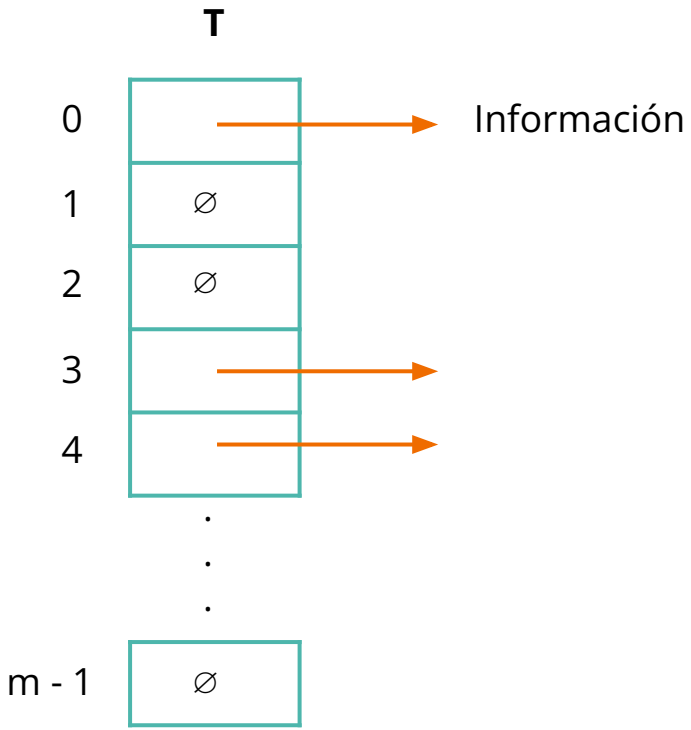


Tabla de Hash

T [k] = ∅ **no está**

T [k] != ∅

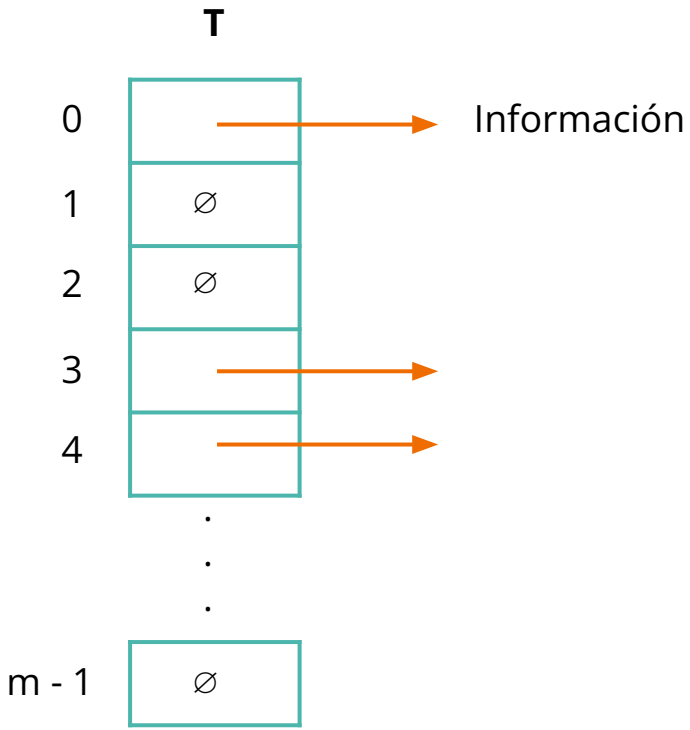
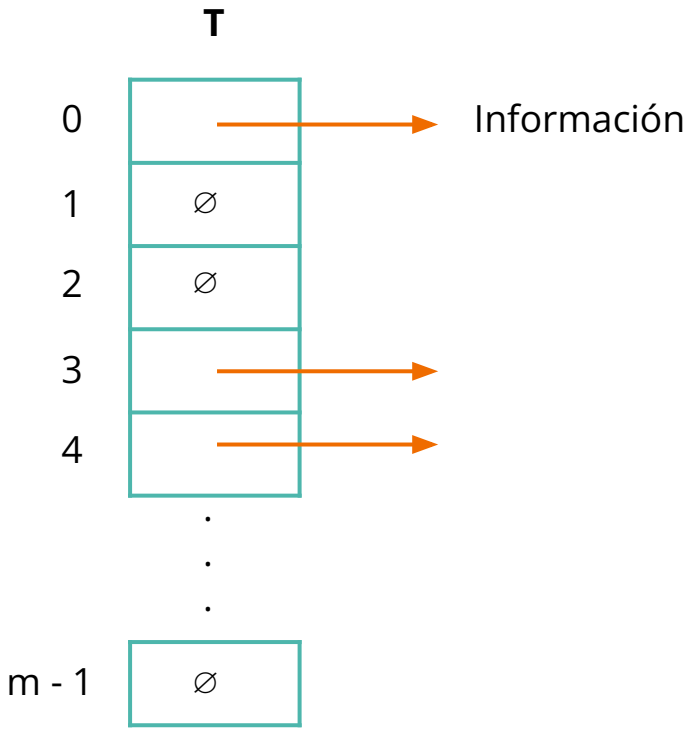


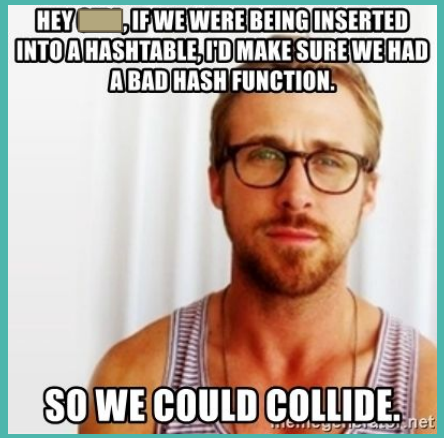
Tabla de Hash

T [k] = ∅ **no está**

T [k] != ∅ **está**

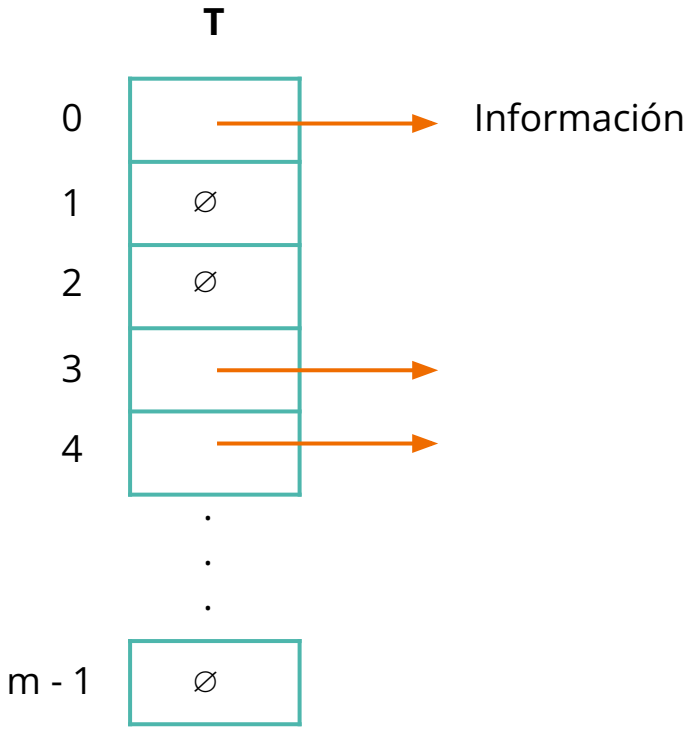


Función de Hash



Función de Hash

- **Calcular** índice a partir de la clave
- **Obtener** el valor asociado a una clave

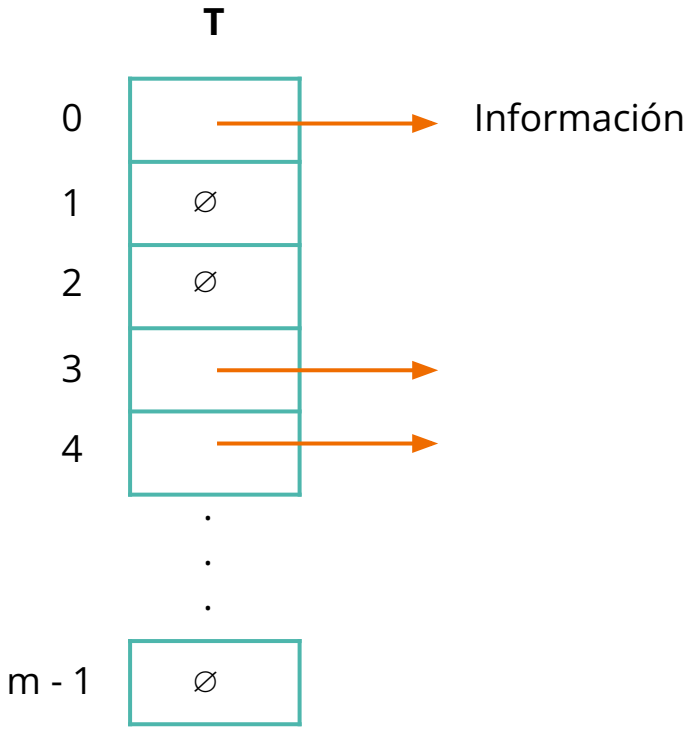


Función de Hash

- **Calcular** índice a partir de la clave
- **Obtener** el valor asociado a una clave

Definimos una función **h**:

h : **D** → {0, 1, ..., m-1}

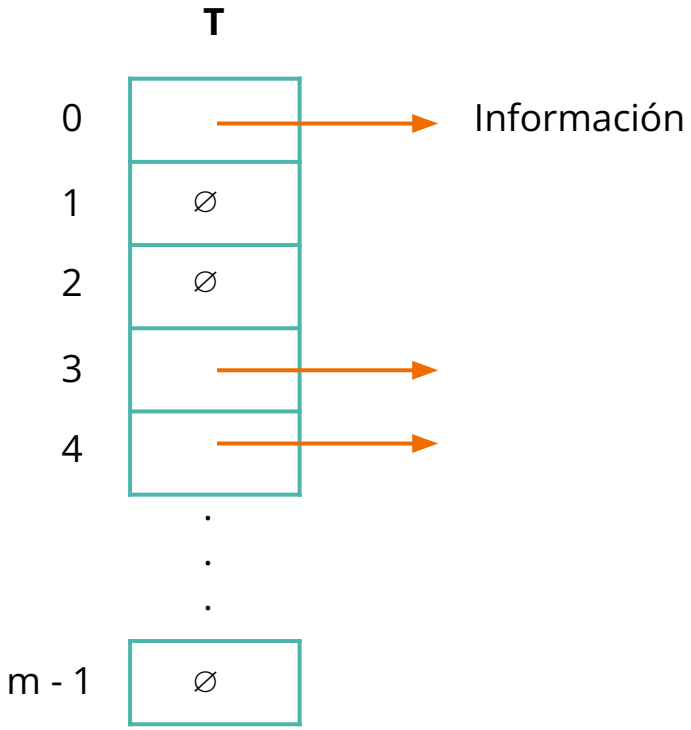


Función de Hash

- **Calcular** índice a partir de la clave
- **Obtener** el valor asociado a una clave

Definimos una función **h**:

h : **D** → {0, 1, ..., m-1}

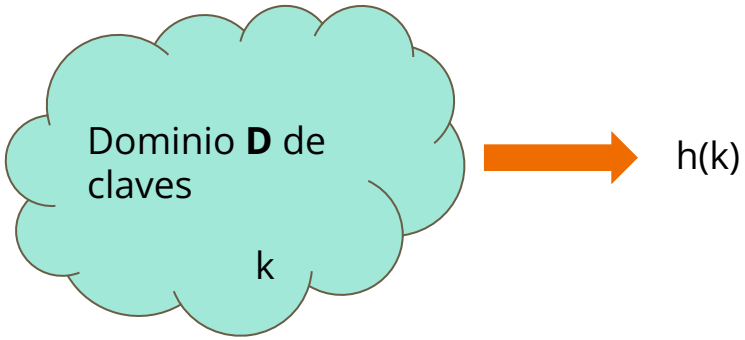


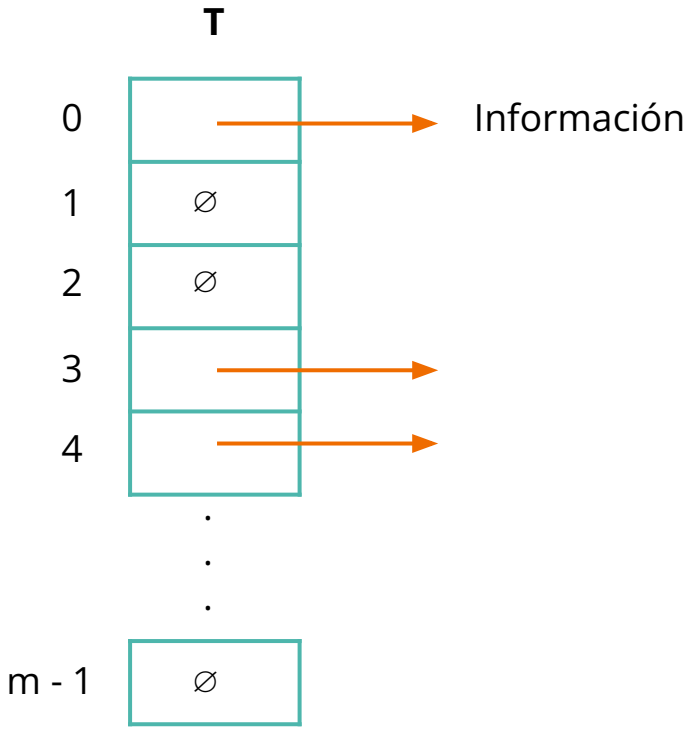
Función de Hash

- **Calcular** índice a partir de la clave
- **Obtener** el valor asociado a una clave

Definimos una función **h**:

h : **D** → {0, 1, ..., m-1}



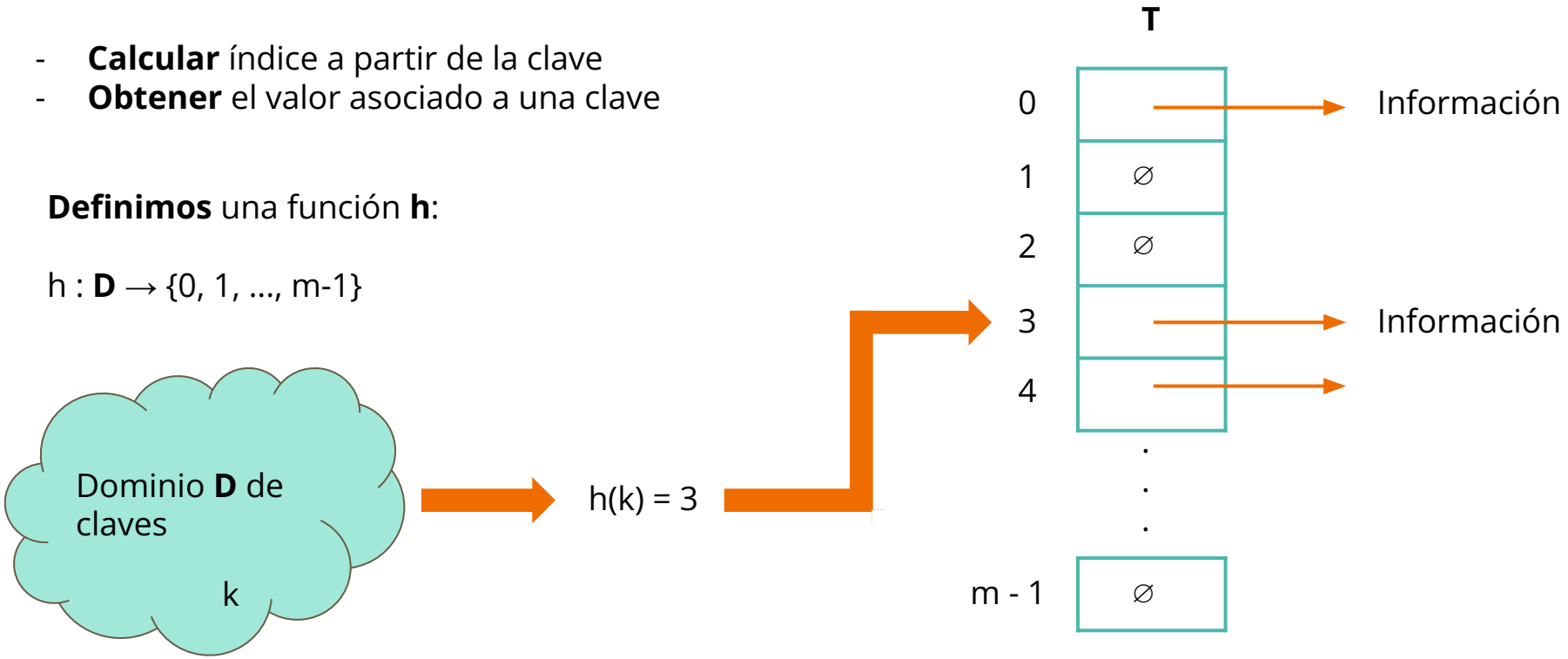


Función de Hash

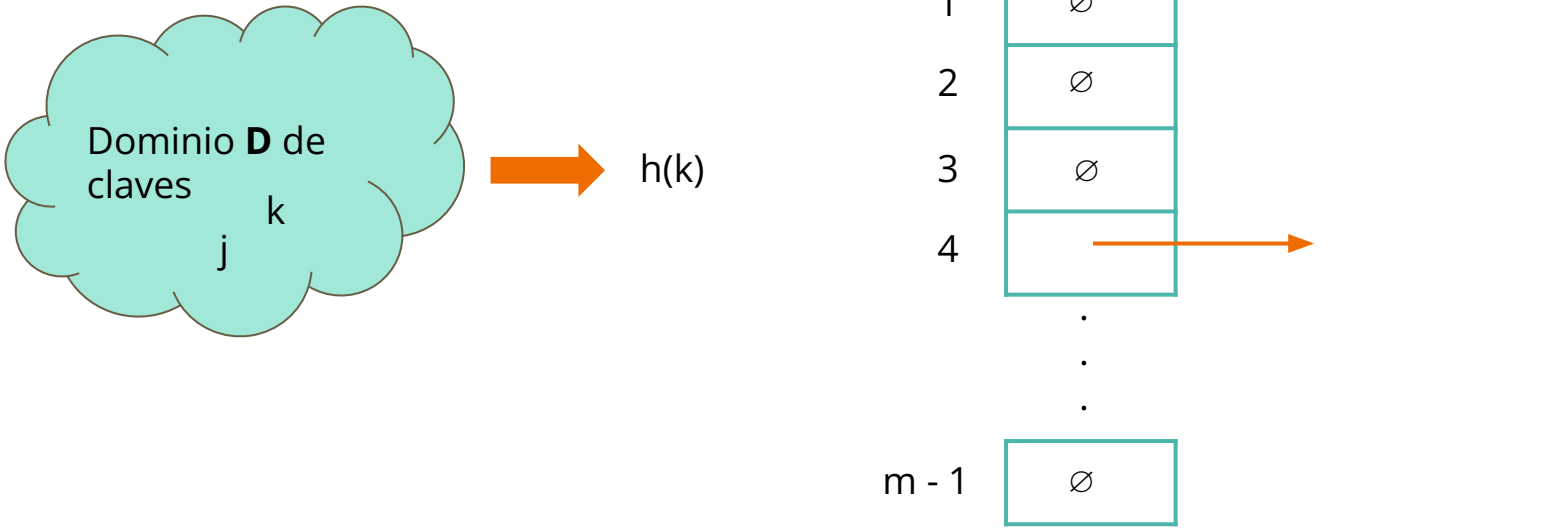
- **Calcular** índice a partir de la clave
- **Obtener** el valor asociado a una clave

Definimos una función **h**:

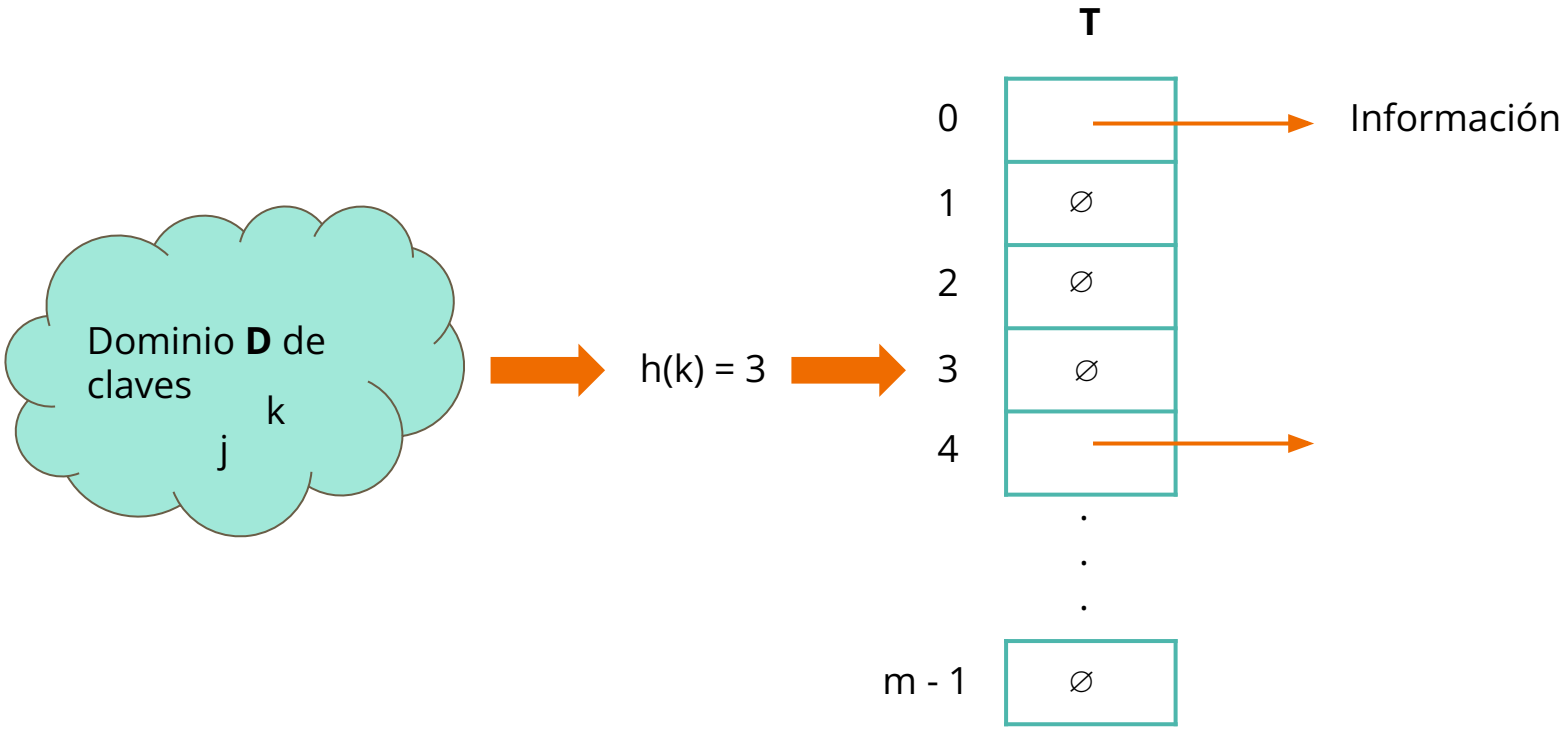
h : **D** → {0, 1, ..., m-1}



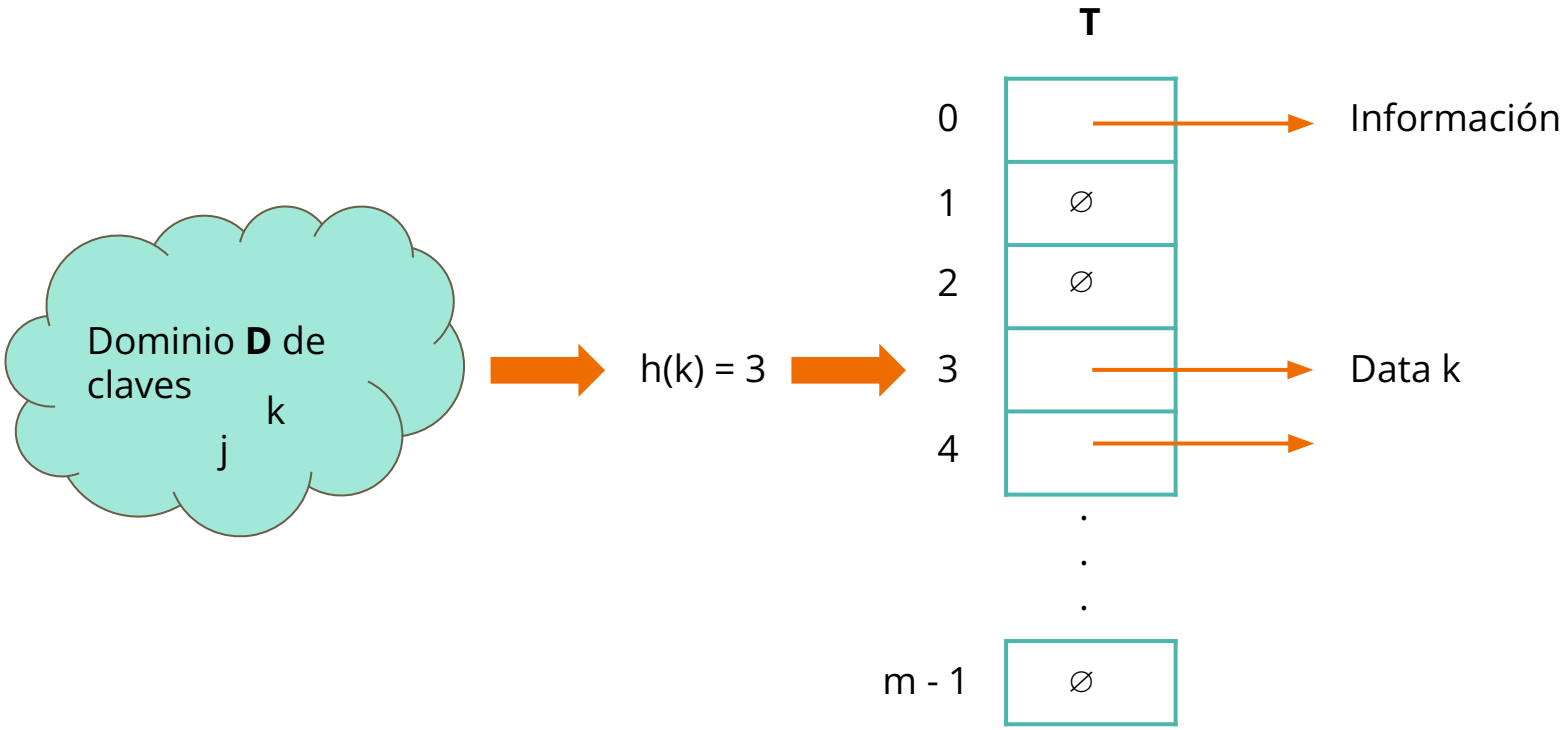
¿Colisiones?



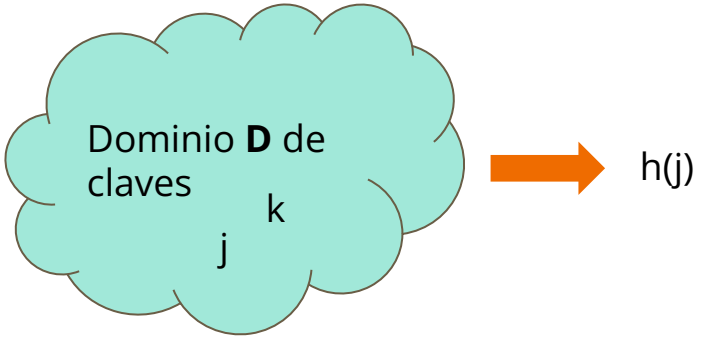
Insertemos la clave k

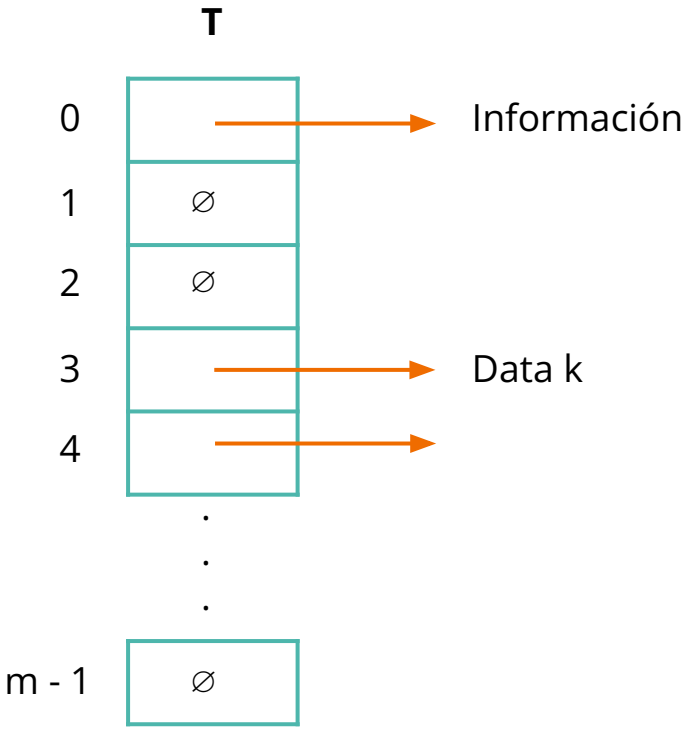


Insertamos información

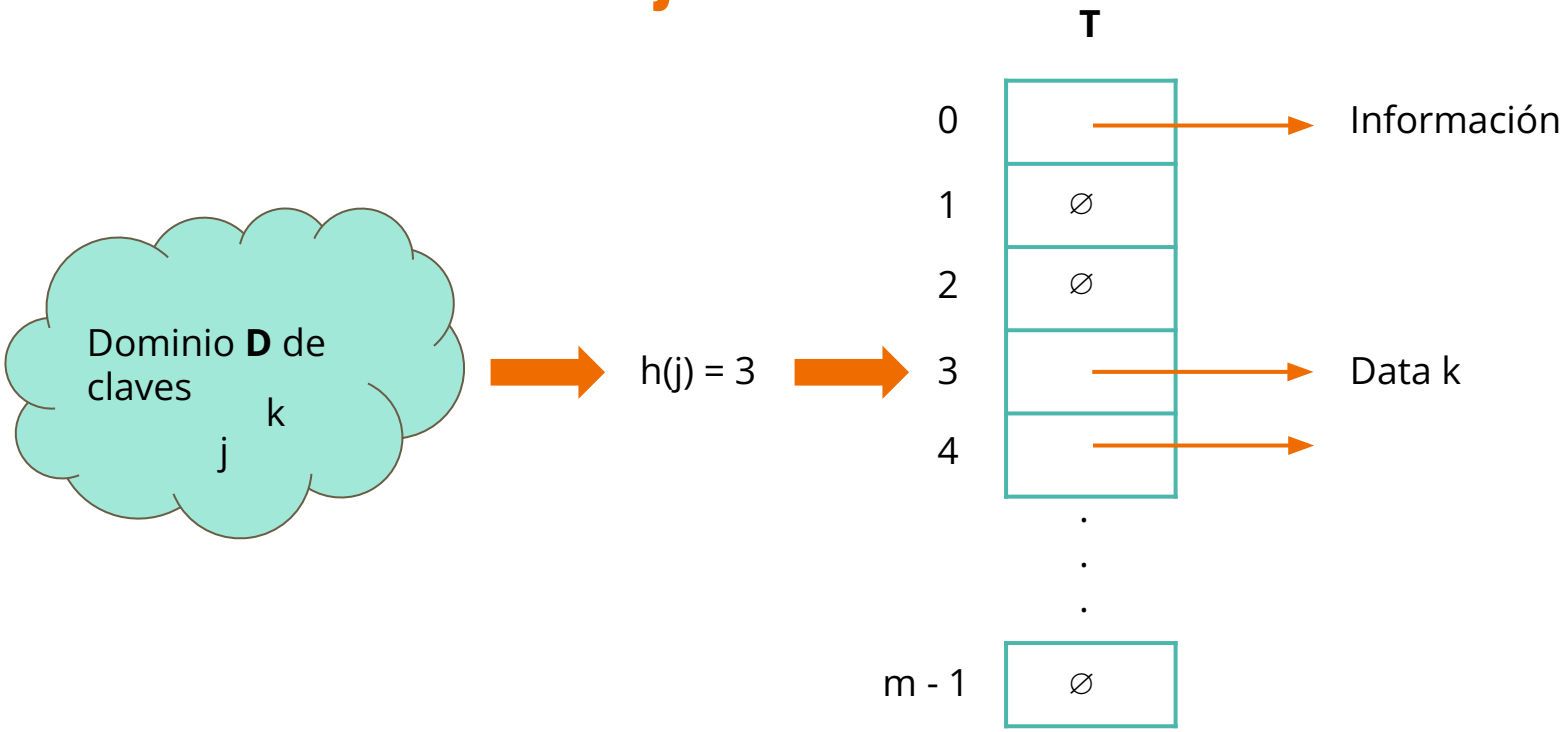


Insertemos la clave j

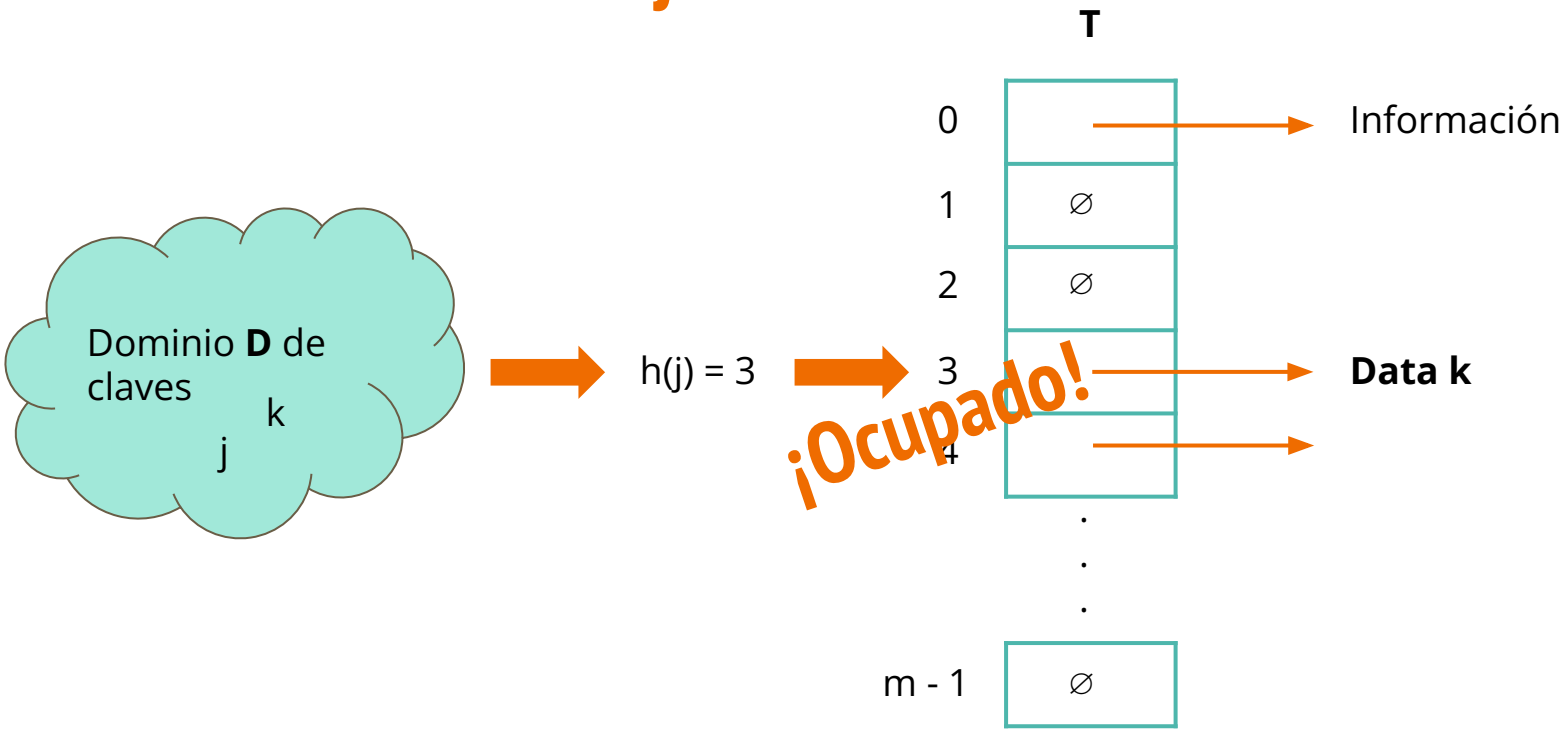




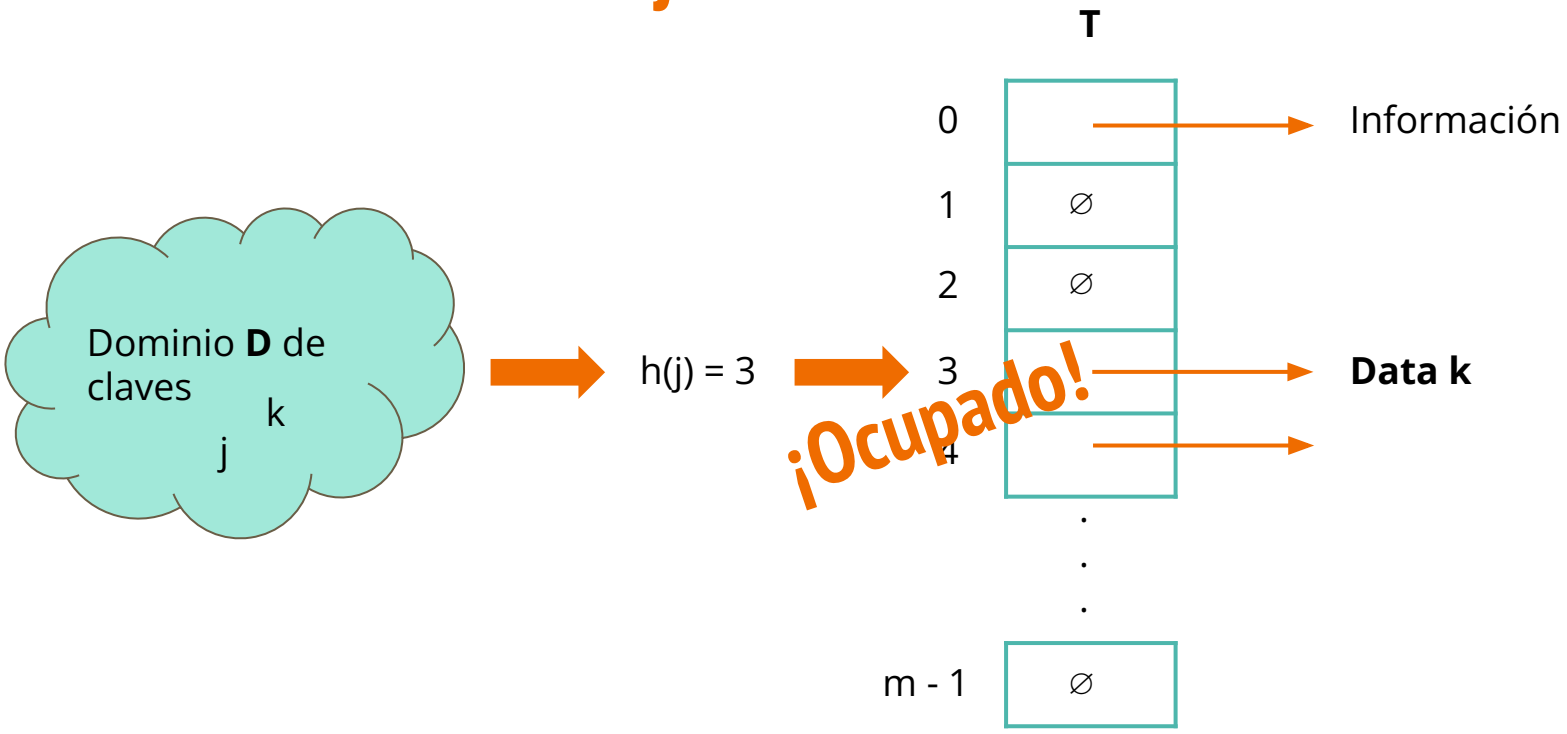
Insertemos la clave j



Insertemos la clave j

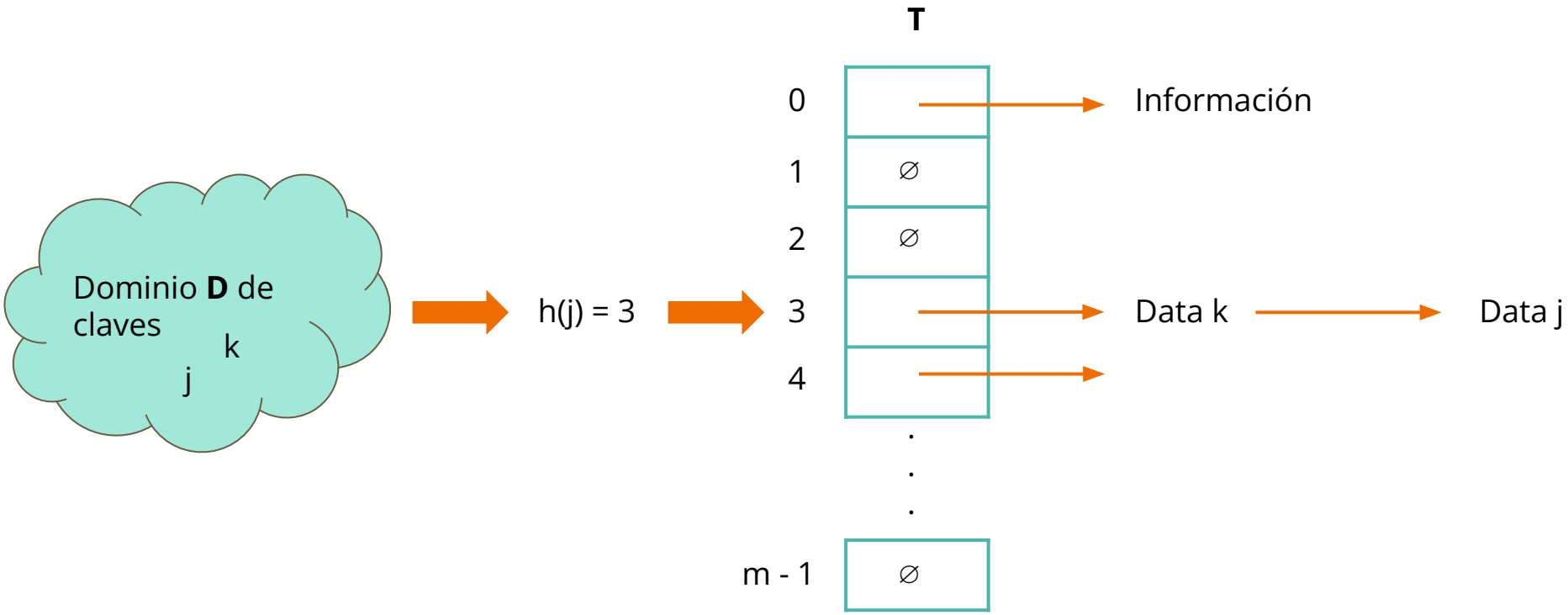


Insertemos la clave j



Solución: Encadenamiento

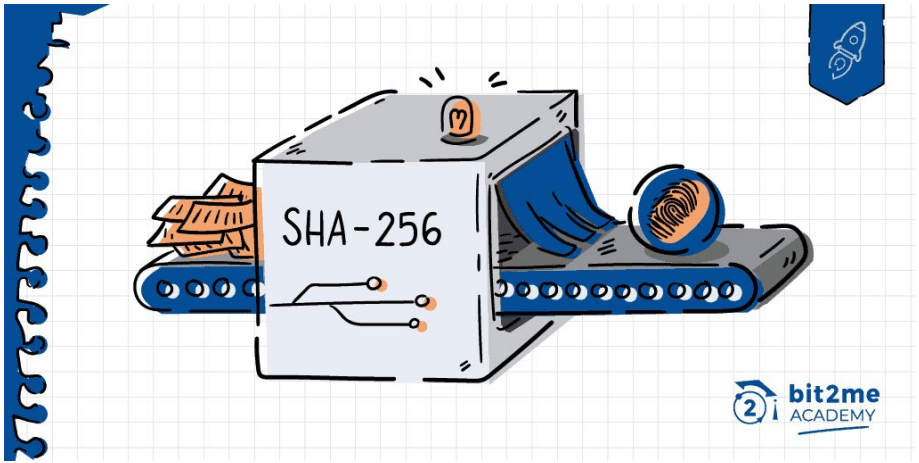
Encadenamiento



¿Por qué Hash?

Propiedades de Hashing

- Almacenar grandes cantidades de información (imágenes, encriptación, integridad de archivos)



Propiedades de Hashing

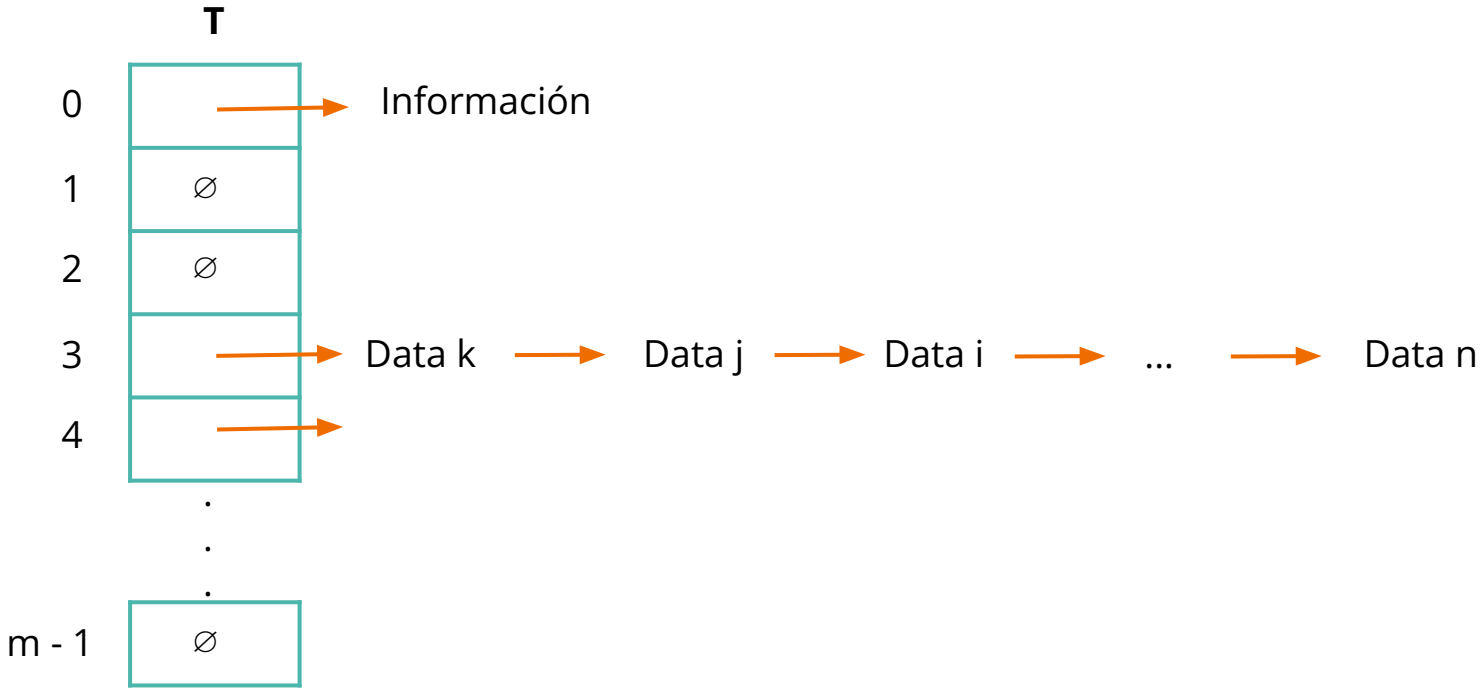
- Almacenar grandes cantidades de información (imágenes, encriptación, integridad de archivos)
- **Buscar** el dato con clave *k* en **O(1)** promedio

Tener en cuenta...

- Uniformidad (encadenamiento)

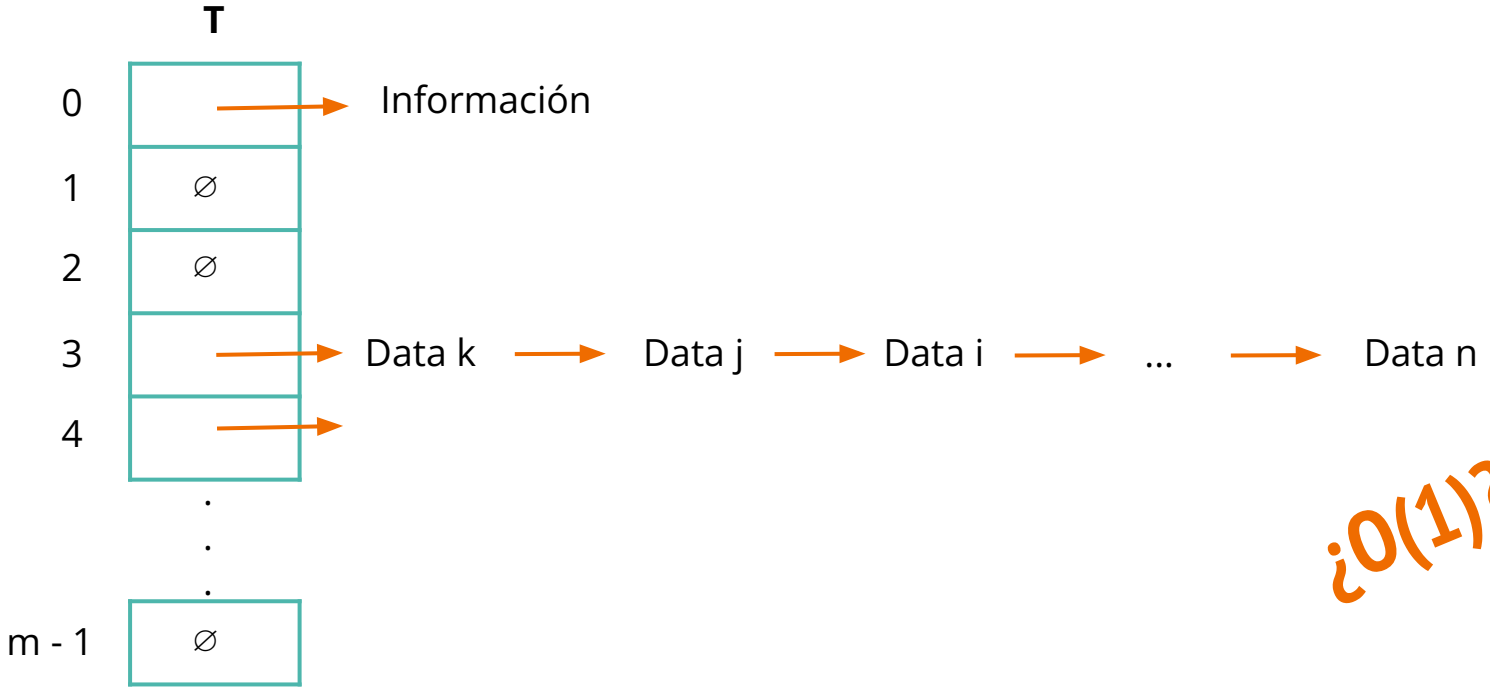
Tener en cuenta...

- Uniformidad (encadenamiento)



Tener en cuenta...

- **Uniformidad** (encadenamiento)



¿O(1)?

Tener en cuenta...

- Uniformidad (encadenamiento)

Tener en cuenta...

- Uniformidad (encadenamiento)
- Tamaño

Tener en cuenta...

- Uniformidad (encadenamiento)
- Tamaño

¡*Trade off* tamaño-eficiencia!

Trade off tamaño-eficiencia

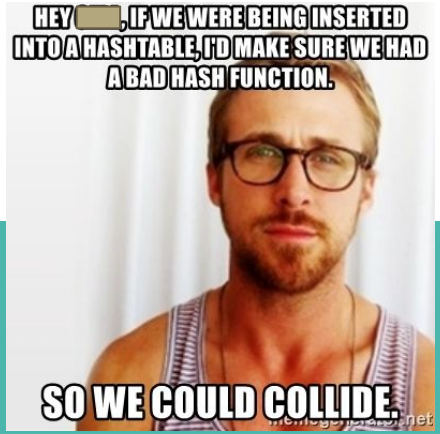
- Uniformidad (encadenamiento)
- Tamaño de la tabla

Trade off tamaño-eficiencia

- Uniformidad (encadenamiento)
- Tamaño de la tabla
- Colisiones

Trade off tamaño-eficiencia

- Uniformidad (encadenamiento)
- Tamaño de la tabla
- Colisiones
- Rapidez de cálculo de la función


HEY [redacted], IF WE WERE BEING INSERTED INTO A HASHTABLE, I'D MAKE SURE WE HAD A BAD HASH FUNCTION.
SO WE COULD COLLIDE.


HAVING NO COLLISIONS IN YOUR HASH TABLE
makeameme.org

Problemas

P1 Encadenamiento

Considera una **tabla de hash** en que las colisiones se resuelven mediante encadenamiento: ¿Con qué estructuras podrías implementar el encadenamiento?

P1 Encadenamiento

Considera una **tabla de hash** en que las colisiones se resuelven mediante encadenamiento: ¿Con qué estructuras podrías implementar encadenamiento?

Indica tres estructuras de datos fundamentalmente diferentes para implementar este "encadenamiento". Explica de manera breve y precisa las ventajas y desventajas principales de cada una.

P1 Encadenamiento

Considera una **tabla de hash** en que las colisiones se resuelven mediante encadenamiento: ¿Con qué estructuras podrías implementar encadenamiento?

Indica tres estructuras de datos fundamentalmente diferentes para implementar este "encadenamiento". Explica de manera breve y precisa las ventajas y desventajas principales de cada una.

P1 Encadenamiento

Considera una **tabla de hash** en que las colisiones se resuelven mediante encadenamiento: ¿Con qué estructuras podrías implementar encadenamiento?

Indica tres estructuras de datos fundamentalmente diferentes para implementar este "encadenamiento". Explica de manera breve y precisa las ventajas y desventajas principales de cada una.

- **Lista ligada**

P1 Encadenamiento

Considera una **tabla de hash** en que las colisiones se resuelven mediante encadenamiento: ¿Con qué estructuras podrías implementar encadenamiento?

Indica tres estructuras de datos fundamentalmente diferentes para implementar este "encadenamiento". Explica de manera breve y precisa las ventajas y desventajas principales de cada una.

- **Lista ligada**: Tomando *n* como el largo de la lista ligada podemos insertar al inicio en *O(1)*, búsqueda en *O(n)* y eliminación en *O(1)* si es doblemente ligada.

  Ventajas: Fácil de implementar, inserción y eliminación eficiente.

  Desventajas: Búsqueda ineficiente.

P1 Encadenamiento

Considera una **tabla de hash** en que las colisiones se resuelven mediante encadenamiento: ¿Con qué estructuras podrías implementar encadenamiento?

Indica tres estructuras de datos fundamentalmente diferentes para implementar este "encadenamiento". Explica de manera breve y precisa las ventajas y desventajas principales de cada una.

- **Lista ligada**: **V**: Implementación, Inserción, eliminación. **DV**: Búsqueda.

P1 Encadenamiento

Considera una **tabla de hash** en que las colisiones se resuelven mediante encadenamiento: ¿Con qué estructuras podrías implementar encadenamiento?

Indica tres estructuras de datos fundamentalmente diferentes para implementar este "encadenamiento". Explica de manera breve y precisa las ventajas y desventajas principales de cada una.

- **Lista ligada**: **V**: Implementación, Inserción, eliminación. **DV**: Búsqueda.
- **Árbol Binario de Búsqueda**

P1 Encadenamiento

Considera una **tabla de hash** en que las colisiones se resuelven mediante encadenamiento: ¿Con qué estructuras podrías implementar encadenamiento?

Indica tres estructuras de datos fundamentalmente diferentes para implementar este "encadenamiento". Explica de manera breve y precisa las ventajas y desventajas principales de cada una.

- **Lista ligada**: **V**: Implementación, Inserción, eliminación. **DV**: Búsqueda.
- **Árbol Binario de Búsqueda:** Tomando *n* como la cantidad de objetos del árbol, podemos insertar, eliminar y buscar en *O(log n)* promedio, o peor caso con un árbol balanceado.

  Ventajas: Complejidad no varía, evitamos sorpresas.

  Desventajas: Implementación no tan simple.

P1 Encadenamiento

Considera una **tabla de hash** en que las colisiones se resuelven mediante encadenamiento: ¿Con qué estructuras podrías implementar encadenamiento?

Indica tres estructuras de datos fundamentalmente diferentes para implementar este "encadenamiento". Explica de manera breve y precisa las ventajas y desventajas principales de cada una.

- **Lista ligada**: **V**: Implementación, Inserción, eliminación. **DV**: Búsqueda.
- **Árbol Binario de Búsqueda: V**: Complejidad. **DV**: Implementación.

P1 Encadenamiento

Considera una **tabla de hash** en que las colisiones se resuelven mediante encadenamiento: ¿Con qué estructuras podrías implementar encadenamiento?

Indica tres estructuras de datos fundamentalmente diferentes para implementar este "encadenamiento". Explica de manera breve y precisa las ventajas y desventajas principales de cada una.

- **Lista ligada**: **V**: Implementación, Inserción, eliminación. **DV**: Búsqueda.
- **Árbol Binario de Búsqueda: V**: Complejidad. **DV**: Implementación.
- **Tabla de Hash**

P1 Encadenamiento

Considera una **tabla de hash** en que las colisiones se resuelven mediante encadenamiento: ¿Con qué estructuras podrías implementar encadenamiento?

Indica tres estructuras de datos fundamentalmente diferentes para implementar este "encadenamiento". Explica de manera breve y precisa las ventajas y desventajas principales de cada una.

- **Lista ligada**: **V**: Implementación, Inserción, eliminación. **DV**: Búsqueda.
- **Árbol Binario de Búsqueda: V**: Complejidad. **DV**: Implementación.
- **Tabla de Hash**: ???

P1 Encadenamiento

Considera una **tabla de hash** en que las colisiones se resuelven mediante encadenamiento: ¿Con qué estructuras podrías implementar encadenamiento?

Indica tres estructuras de datos fundamentalmente diferentes para implementar este "encadenamiento". Explica de manera breve y precisa las ventajas y desventajas principales de cada una.

- **Lista ligada**: **V**: Implementación, Inserción, eliminación. **DV**: Búsqueda.
- **Árbol Binario de Búsqueda: V**: Complejidad. **DV**: Implementación.
- **Tabla de Hash**: Hash por cada celda, podemos buscar e insertar en *O(1).*

  Ventajas: Complejidad constante.

  Desventajas: Implementación.

P1 Encadenamiento

Considera una **tabla de hash** en que las colisiones se resuelven mediante encadenamiento: ¿Con qué estructuras podrías implementar encadenamiento?

Indica tres estructuras de datos fundamentalmente diferentes para implementar este "encadenamiento". Explica de manera breve y precisa las ventajas y desventajas principales de cada una.

- **Lista ligada**: **V**: Implementación, Inserción, eliminación. **DV**: Búsqueda.
- **Árbol Binario de Búsqueda: V**: Complejidad. **DV**: Implementación.
- **Tabla de Hash**: **V**: Buscar e insertar. **DV**: Implementación.

P2 Los hashes te van a salvar para entrevistas de código!

Dado un array de int llamado nums y un int target, retorna el índice de 2 números que al sumarlos da como resultado target.

```
Input: nums = [2,7,11,15], target = 9
Output: [0,1]
Output: Because nums[0] + nums[1] == 9, we return [0, 1].
```

P2 Los hashes te van a salvar para entrevistas de código!

Dado un arry de int llamado nums y un int target, retorna el índice de 2 números que al sumarlos da como resultado target.

```
Input: nums = [2,7,11,15], target = 9
Output: [0,1]
Output: Because nums[0] + nums[1] == 9, we return [0, 1].
```

Solucion facil:

```
class Solution(object):
    def twoSum(self, nums, target):
        sol = []
        for i in range(len(nums)):
            for j in range(len(nums)):
                if nums[i] + nums[j] == target:
                    sol.append(i)
                    sol.append(j)
                    return sol
```

Cuál es su complejidad?
¿Alguna mejora?

P2

Dado un arry de int llamado nums y un int target, retorna el índice de 2 números que al sumarlos da como resultado target.

```
Input: nums = [2,7,11,15], target = 9
Output: [0,1]
Output: Because nums[0] + nums[1] == 9, we return [0, 1].
```

Solucion con hash:

```python
class Solution(object):
    def twoSum(self, nums, target):
        dic = dict()
        for index, number in enumerate(nums):
            if target - number not in dic:
                dic[number] = index
            else:
                return [dic[target-number] , index]
```

Saving the day with hashing.
lineal run time O(n)

P3

Dado 2 strings s y t, determinan si son anagramas. Los strings son anagramas cuando con las letras de un stirng se puede formar el otro string

```
Input: s = "anagram", t = "nagaram"
Output: true
```

P3

Dado 2 strings s y t, determinan si son anagramas.

```
Input: s = "anagram", t = "nagaram"
Output: true
```

Solucion con hash:

```python
class Solution(object):
    def isAnagram(self, s, t):
        dic = dict()
        for i in range(26):
            dic[i] = 0
        for i in range(len(s)):
            idx = ord(s[i]) - ord("a")
            dic[idx] += 1
        for j in range(len(t)):
            idx = ord(t[j]) - ord("a")
            dic[idx] -= 1
        for i in range(26):
            if dic[i] != 0:
                return False
        return True
```

Saving the day with hashing.
lineal run time O(n)

P4

Dado 2 strings s y t de mismo largo, determinan si son isomórficos. 2 strings son isomórficos cuando los caracteres de s pueden ser reemplazados para tener t. Ojo que cada carácter solo puede ser reemplazado por un único carácter y pueden ser reemplazado por el mismo.

Example 1:

```
Input: s = "egg", t = "add"
Output: true
```

Example 2:

```
Input: s = "foo", t = "bar"
Output: false
```

P4

Dado 2 strings s y t de mismo largo, determinan si son isomórficos. 2 strings son isomórficos cuando los caracteres de s pueden ser reemplazados para tener t. Ojo que cada carácter solo puede ser reemplazado por un único carácter y pueden ser reemplazado por el mismo.

```
class Solution(object):
    def isIsomorphic(self, s, t):
        dic_s= dict()
        dic_t= dict()
        i = 0
        j = 0
        while i < len(s):
            if s[i] not in dic_s.keys():
                dic_s[s[i]] = t[j]
            if t[j] not in dic_t.keys():
                dic_t[t[j]] = s[i]
            if dic_t[t[j]] != s[i] or dic_s[s[i]] != t[j]:
                return False
            i += 1
            j += 1
        return True
```