ICC311
Estructuras de Datos

Semestre I, 2020
Profesor: Pablo Valenzuela

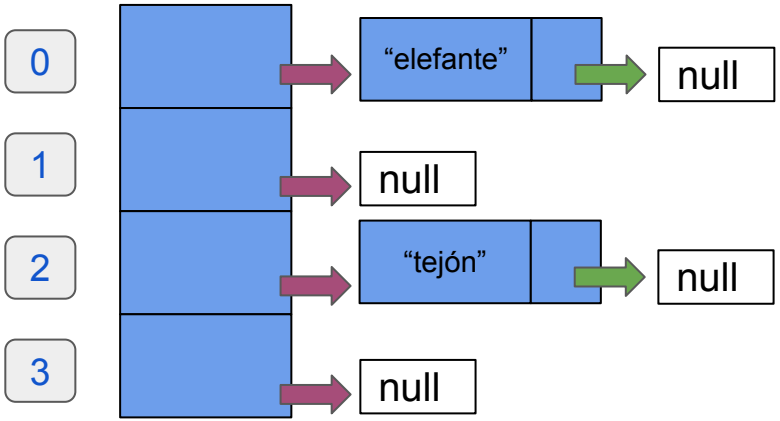
Semana 05 - Parte 02

Tópicos:

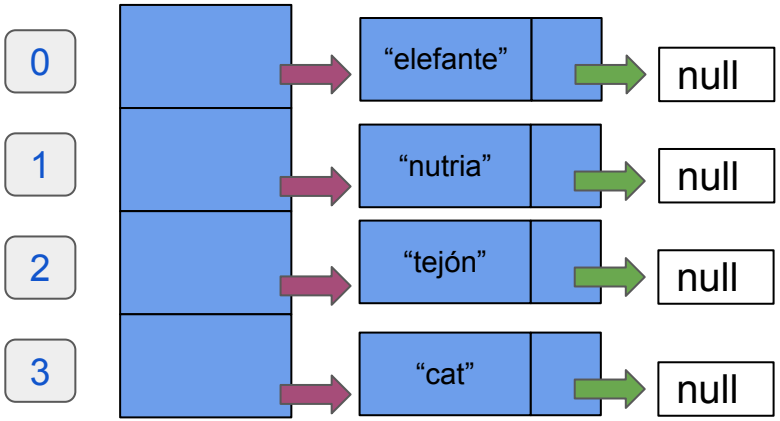
- **Redimensión de Sets**

¿Qué pasa cuando se tiene buckets con demasiados elementos?

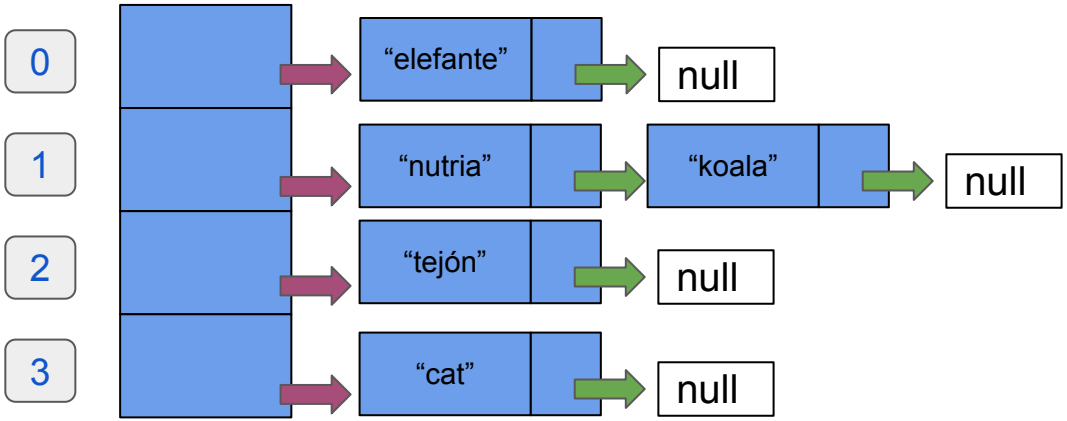
Demasiados elementos en Buckets



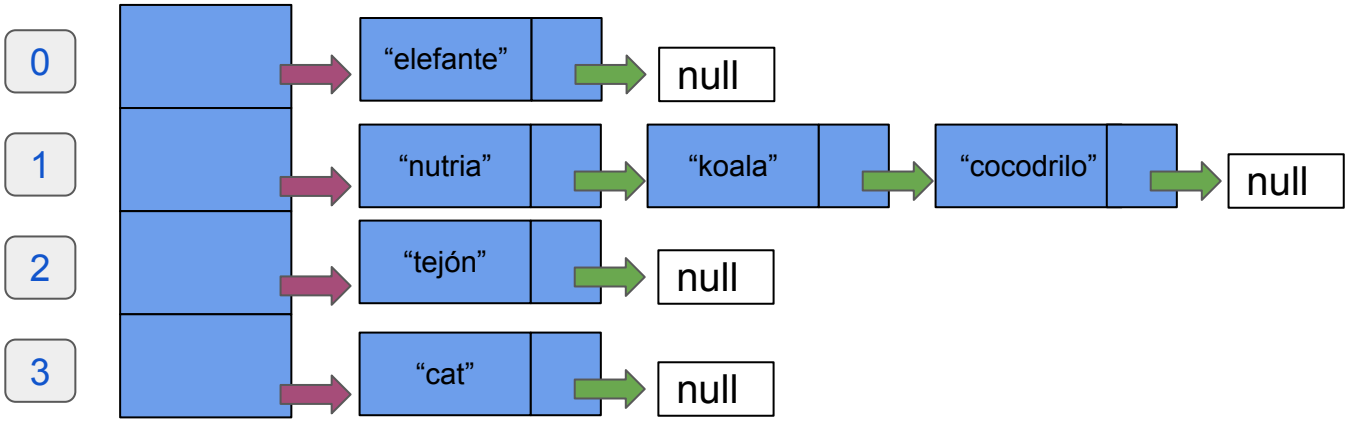
Demasiados elementos en Buckets



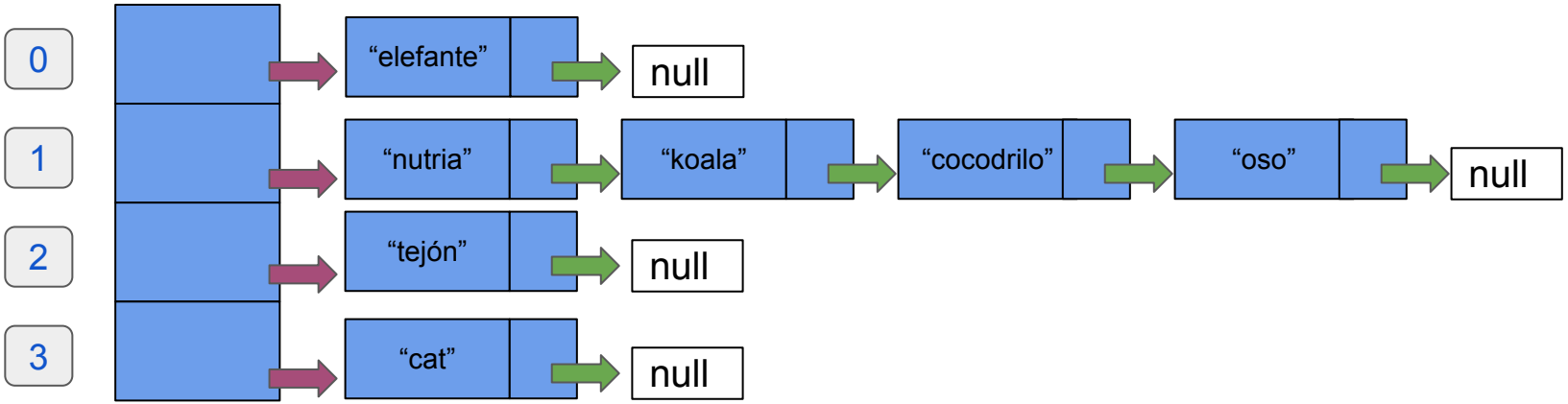
Demasiados elementos en Buckets



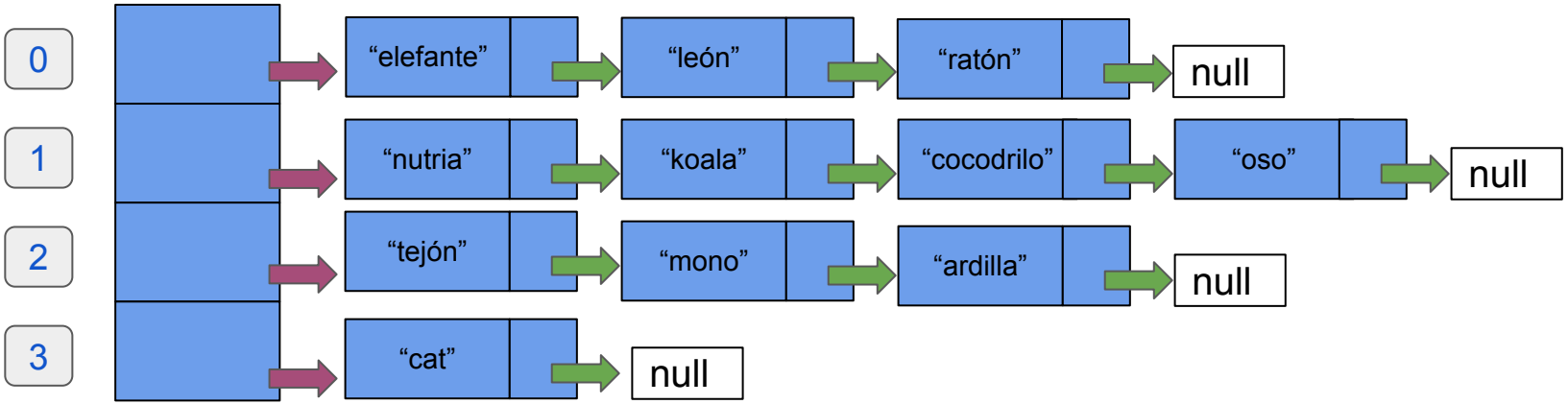
Demasiados elementos en Buckets



Demasiados elementos en Buckets



Demasiados elementos en Buckets



Solución:

el hashset debe automáticamente

redimensionar el nº de buquets

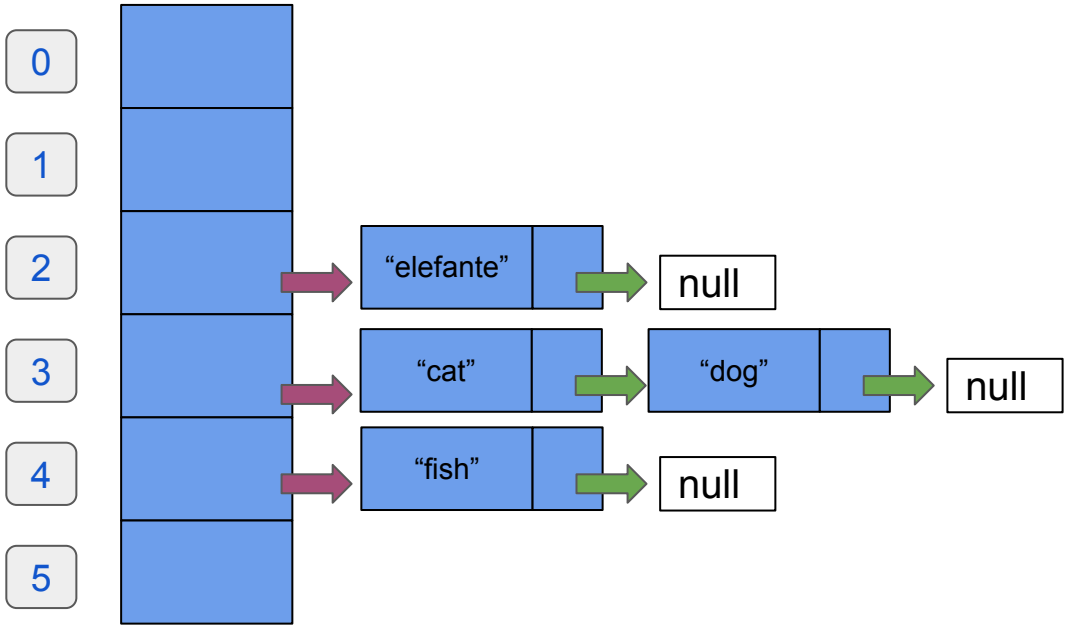
Redimensionado automático

Operaciones hasta ahora:
hashSet.add(“elefante”);
hashSet.add(“dof”);
hashSet.add(“cat”);
hashSet.add(“fish”);

factorIndicador = 0.75

buckets.length = 6

tamanoActual = 4



Redimensionado automático

Operaciones hasta ahora:
hashSet.add(“elefante”);
hashSet.add(“dof”);
hashSet.add(“cat”);
hashSet.add(“fish”);

factorIndicador = 0.75

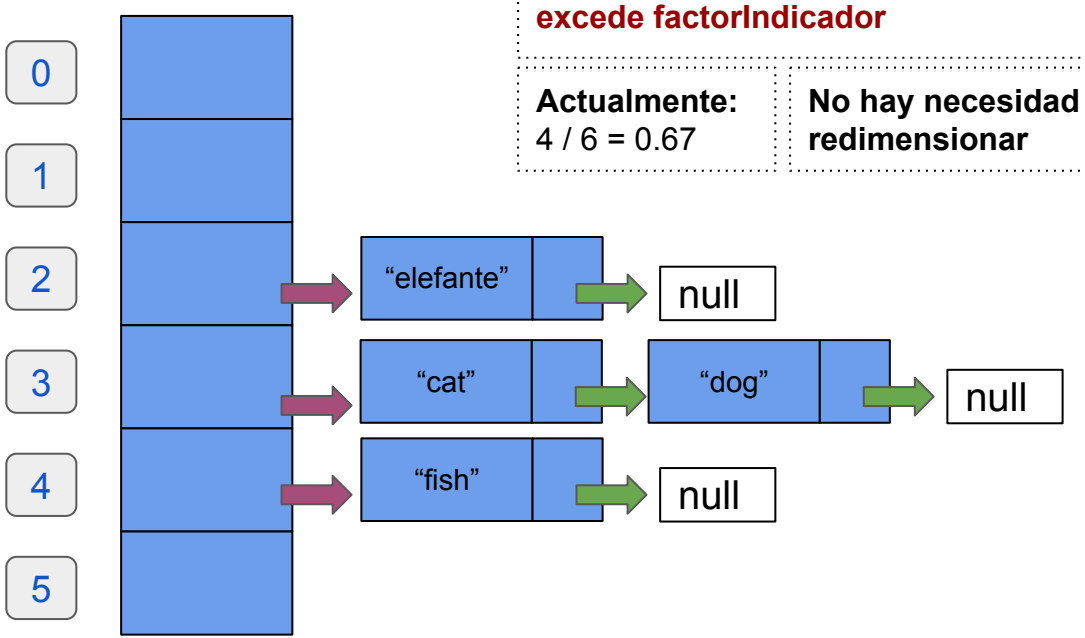
buckets.length = 6

tamanoActual = 4

Redimensionar el nº promedio de elementos por bucket (tamanoActual/buckets.length) excede factorIndicador

Actualmente:
4 / 6 = 0.67

No hay necesidad de redimensionar



Redimensionado automático

Operaciones hasta ahora:
hashSet.add(“elefante”);
hashSet.add(“dof”);
hashSet.add(“cat”);
hashSet.add(“fish”);

Nuevo elemento:
hashSet.add(“marmota”);

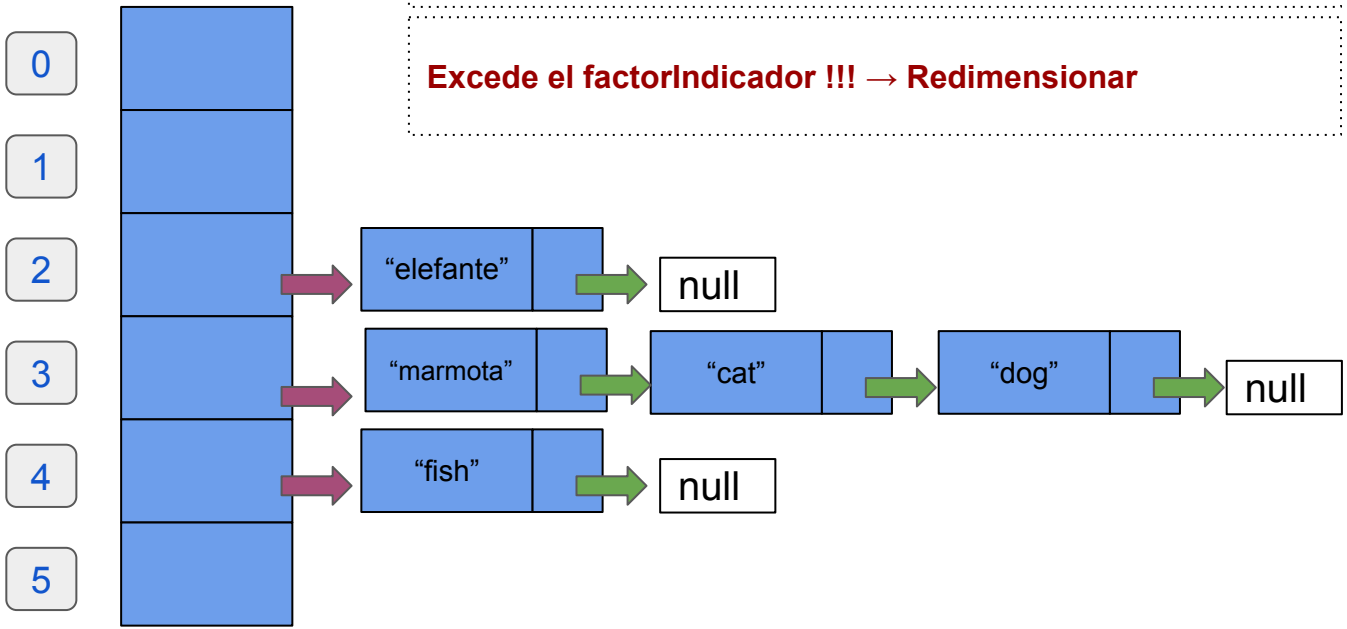
factorIndicador = 0.75

buckets.length = 6

tamanoActual = 4

Nuevo promedio = tamanoActual / buckets.length = 0.83

Excede el factorIndicador !!! → Redimensionar



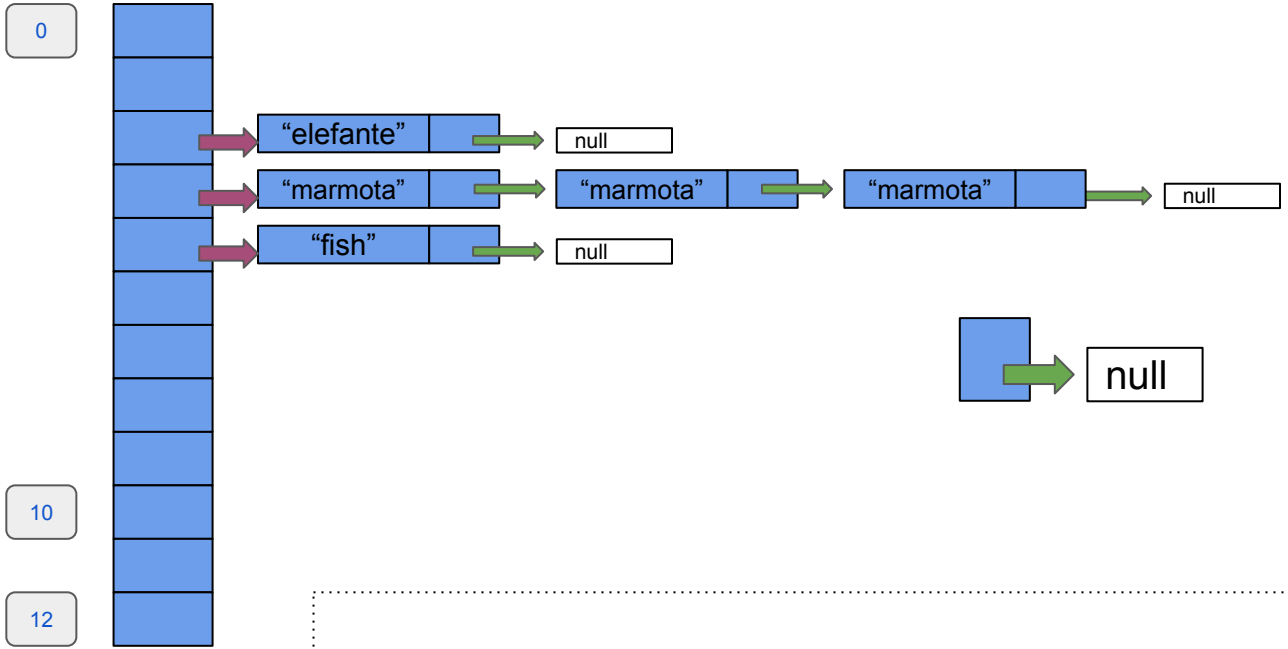
1.- Doble nº de Buckets

Redimensionado automático: 1 doblar nº buckets

factorIndicador = 0.75

buckets.length = 12

tamanoActual = 5



Nuevo promedio = tamanoActual / buckets.length = 0.42

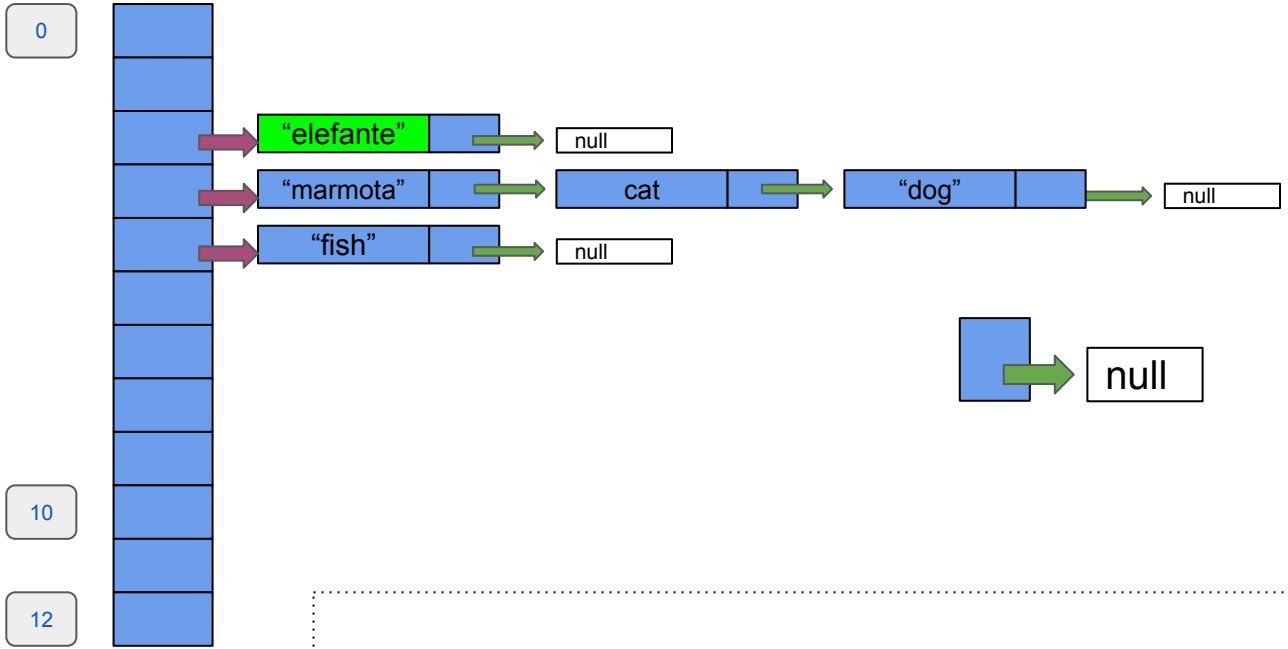
2.- Reinsertar valores existentes

Redimensionado automático: 2 reinsertar

factorIndicador = 0.75

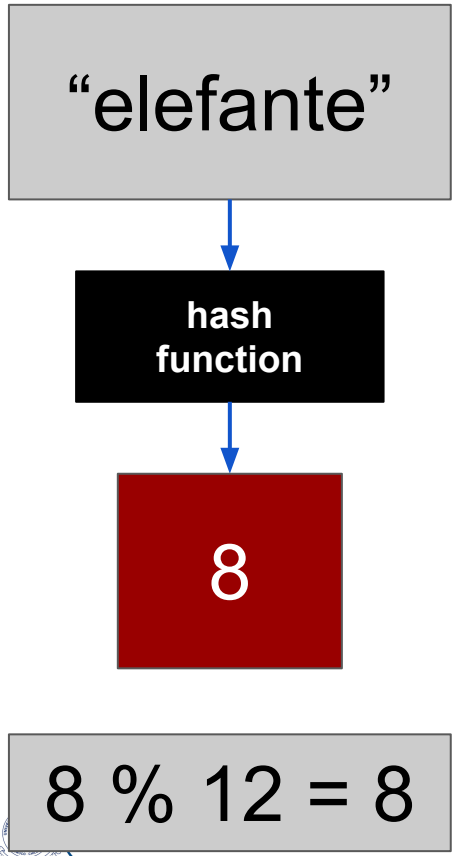
buckets.length = 12

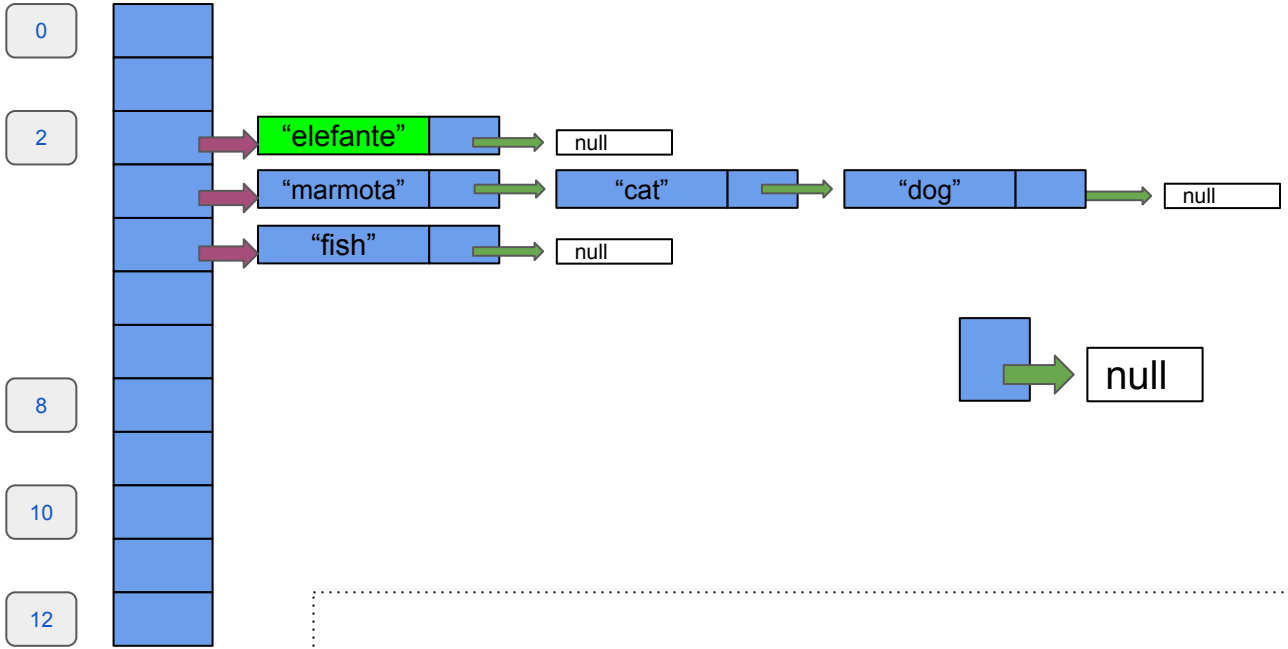
tamanoActual = 5



Nuevo promedio = tamanoActual / buckets.length = 0.42

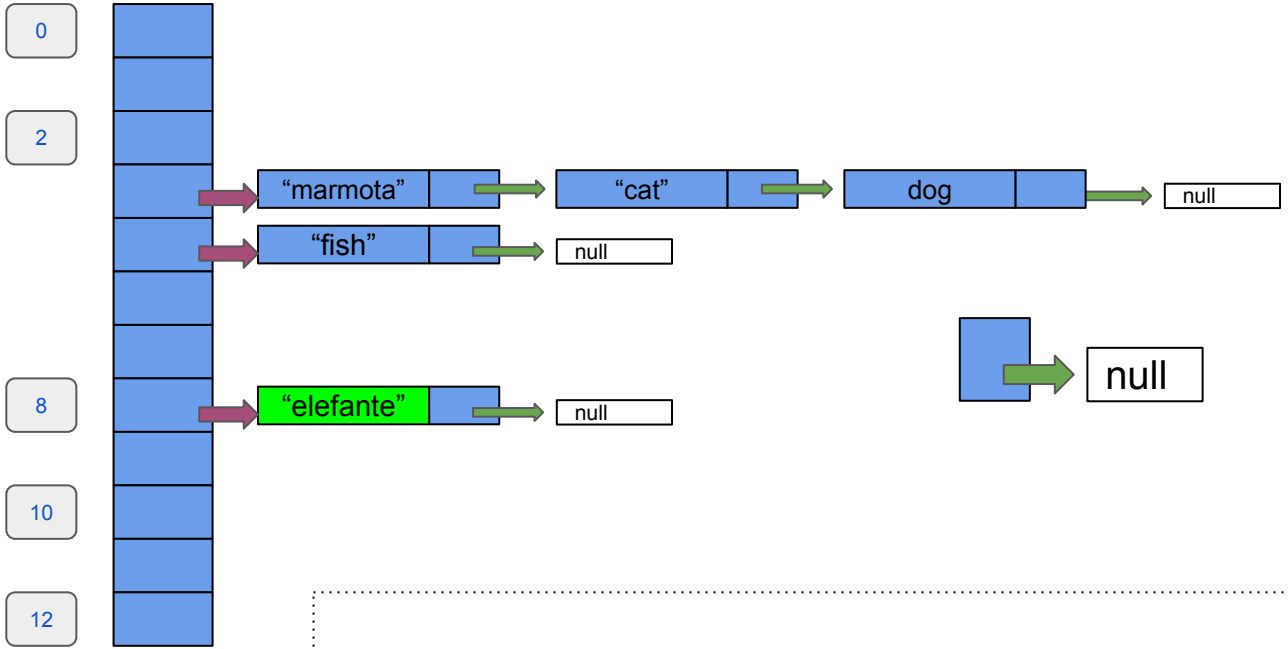
Redimensionado automático: 2 reinsertar





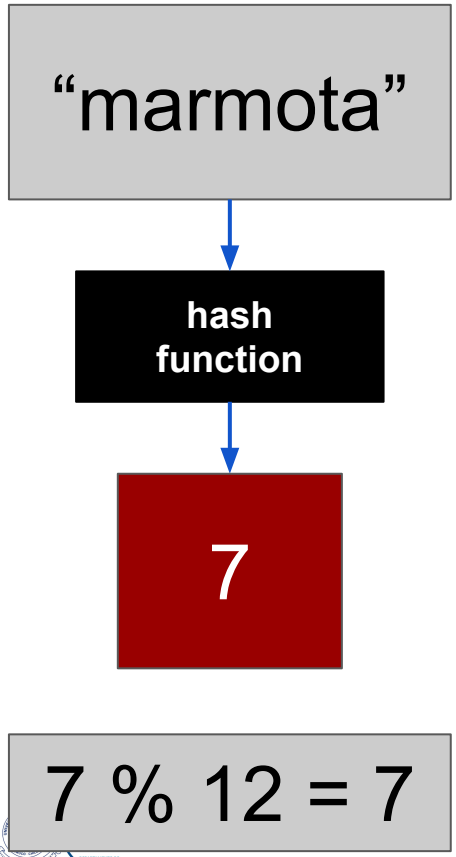
Nuevo promedio = tamanoActual / buckets.length = 0.42

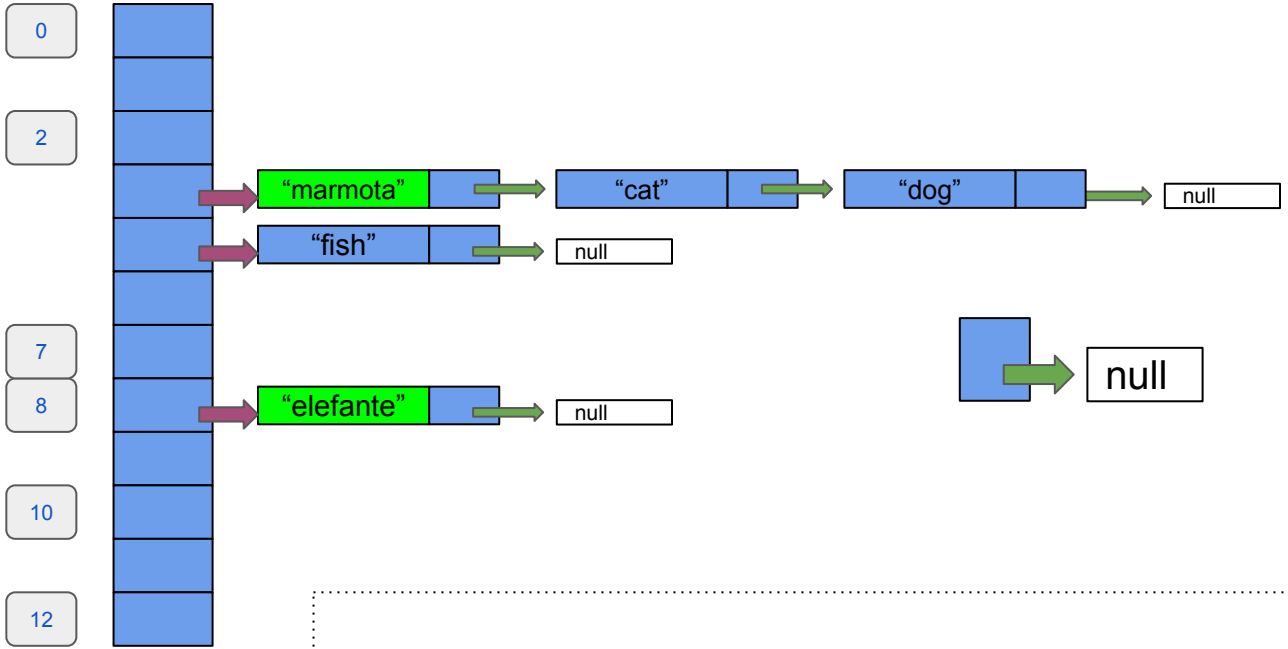
Redimensionado automático: 2 reinsertar



Nuevo promedio = tamanoActual / buckets.length = 0.42

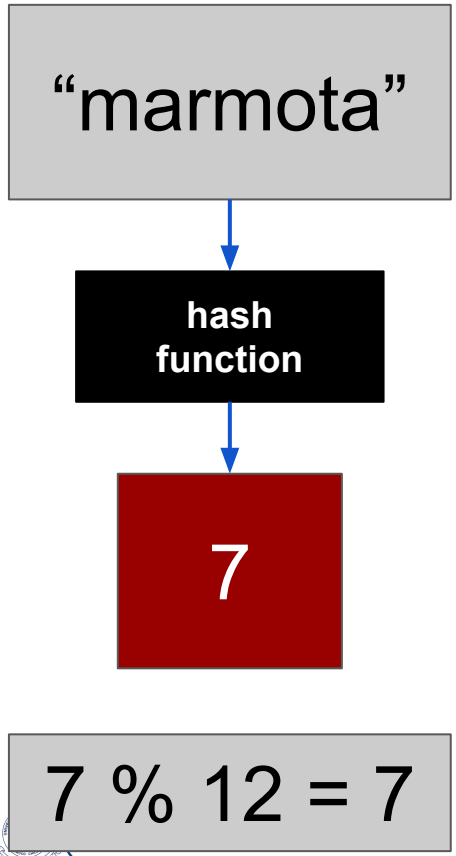
Redimensionado automático: 2 reinsertar

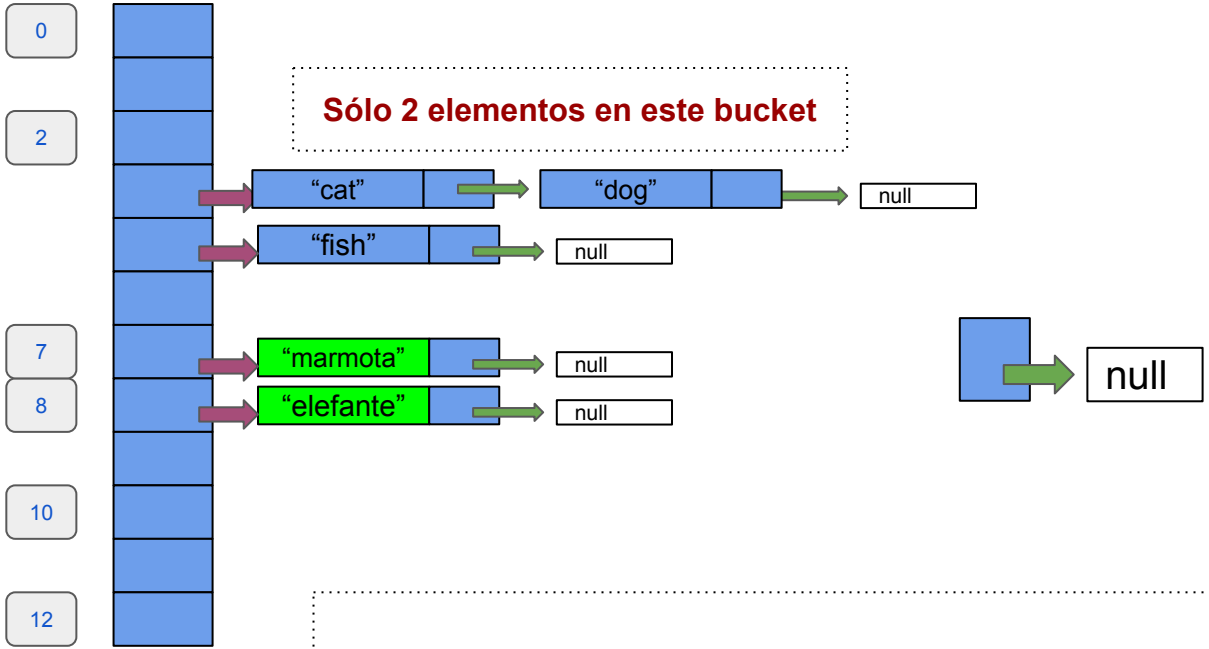




Nuevo promedio = tamanoActual / buckets.length = 0.42

Redimensionado automático: 2 reinsertar





Nuevo promedio = tamanoActual / buckets.length = 0.42

Redimensionado automático

```java
public class HashSetResizing {
    private LinkedList<String>[] buckets;
    private int tamanoActual = 0;
    //factor que determina el tamaño que debe tener el arreglo
    private double factorIndicador;

    public HashSetResizing(int tamano, double factorIndicador) {
        buckets = new LinkedList[tamano];
        for (int i = 0; i < tamano; i++) {
            buckets[i] = new LinkedList<String>();
        }
        this.factorIndicador = factorIndicador;
    }

    private int hashCode(String valor) {
        return valor.length();
    }
```

```java
    public boolean add(String valor) {
        if (!contiene(valor)) {
            int index = hashCode(valor) % buckets.length;
            LinkedList<String> bucket = buckets[index];
            bucket.addFirst(valor);
            tamanoActual++;

            double promedio = tamanoActual / (double) buckets.length;
            if (promedio > factorIndicador) {
                reinsertarTodo();
            }
        }
        return false;
    }
```

```java
    private void reinsertarTodo() {
        LinkedList<String> oldBucket[] = buckets;
        buckets = new LinkedList[buckets.length * 2];

        for (LinkedList<String> bucket : buckets) {
            for (String elemento : bucket) {
                int index = hashCode(elemento) % buckets.length;
                LinkedList<String> nuevoBucket = buckets[index];
                nuevoBucket.addFirst(elemento);
            }
        }
    }

    public boolean contiene(String valor) {
        int index = hashCode(valor) % buckets.length;
        LinkedList<String> bucket = buckets[index];
        return bucket.contains(valor);
    }
}
```

Recapitulación

Recapitulación: HashSet

- En HashSet, los elementos son almacenados en arreglos de Listas Enlazadas, o “buckets”
- Si los buckets comienzan a llenarse, se pierde la habilidad para encontrar elementos rápidamente
- El HashSet es capaz de auto redimensionarse, tal que elementos de un bucket se pueden mover a nuevos espacios, permitiendo encontrar elementos rápidamente

ICC311
Estructuras de Datos

Semestre I, 2020
Profesor: Pablo Valenzuela