

# Diccionarios 1

## *Diccionario implementado mediante arreglos*

Cátedra 25

Otoño 2005

Un diccionario es una estructura de datos básica, que permite administrar información. Siendo tan común, existen distintas implementaciones que satisfacen requerimientos particulares. La implementación más básica y sencilla es mediante un arreglo, pero también es la más limitada. En la clase de hoy se exploran dos implementaciones y se comparan.

Por lo general la información se guarda de forma estructurada, para poder agilizar no solo su búsqueda, sino también su mantención.

Un Diccionario es un conjunto de llaves con valores asociados. Por ejemplo, palabras en castellano (llaves) con su traducción al inglés (los valores).

Un diccionario se puede representar mediante una clase cuyos métodos sean

**void put(String llave, Object valor)** agrega una llave con su valor al diccionario. Por ejemplo, llave puede ser "raton" y valor "mouse".

**Object get(String llave)** devuelve el valor asociado a una llave. Si la llave no existe en el diccionario, devuelve null.

**void remove(String llave)** elimina una llave (y su valor) del diccionario

Aparte de tener una aplicación directa en la implementación de diccionarios, esta clase permite trabajar con colecciones de elementos. Por ejemplo, en un diccionario se puede guardar la información de todos los chilenos, donde la llave es el RUT y el valor un objeto con el nombre, dirección y teléfono.

## 1 Ejemplo de Uso

```
Diccionario d=new Diccionario();
d.put("Perro","Dog");
d.put("Gato","Cat");
d.put("Canario","Canary");
d.put("Elefante","Big mouse");
String s=(String)d.get("Gato");
con.print(s); // escribe gato
d.remove("Elefante");
```

## 2 Implementación mediante un arreglo

Una representación interna sencilla es:

```
class Diccionario {
    String llave[];
    Object valor[];
    int total;
```

El constructor crea los arreglos y los marca como "vacíos":

```
Diccionario() {
    llave=new String[1000];
    valor=new String[1000];
    total=0;
}
```

El método put agrega un elemento al final del arreglo. Por generalidad, el valor es de la superclase Object:

```
void put(String l, Object v) {
    llave[total]=l;
    valor[total]=v;
    total++;
}
```

El método get busca en todo el arreglo la llave, hasta encontrarla:

```
Object get(String l) {
    for (int i=0; i<total;i++)
        if (llave[i].equals(l))
            return valor[i];
    return null;
}
```

El método remove busca el elemento a borrar. Cuando lo encuentra, lo reemplaza por el último y decrementa el total de elementos.

```

void remove(String l) {
    for (int i=0; i<total;i++)
        if (llave[i].equals(l)) {
            llave[i]=llave[total-1];
            valor[i]=valor[total-1];
            total--;
            return;
        }
}

```

## 2.1 Ventajas

Esta implementación tiene las siguientes ventajas:

- Código sencillo
- La inserción toma un número constante de operaciones. Esto se denota como  $O(1)$

## 2.2 Desventajas

La búsqueda toma un tiempo proporcional a la cantidad de elementos, es decir  $O(n)$  operaciones.

- La eliminación de llaves toma  $O(n)$  también
- La cantidad de datos que se pueden guardar está acotada
- Si se guardan menos de 1000 datos, se desperdicia memoria

# 3 Implementación mediante un arreglo ordenado

Una segunda versión consiste en mantener todos los elementos del arreglo ordenados en todo momento.

La representación interna y el constructor permanecen inmodificados.

El método put busca la posición donde se debe agregar el elemento, mueve todos los mayores a él (para hacerle espacio) y lo inserta. En promedio, esto toma  $n/2$  operaciones.

```

void put(String l, Object v) {
    for (int i=0; i<total && llave[i].compareTo(l)<0;i++);
    for (int j=total;j>i;j--) {
        llave[j]=llave[j-1];
        valor[j]=valor[j-1];
    }
    llave[i]=l;
    valor[i]=v;
    total++;
}

```

El método get, al estar todos los elementos ordenados, se vuelve muy eficiente. La búsqueda se realiza descartando mitades, como cuando uno busca en la guía telefónica. Si el diccionario contiene  $n$  elementos, se realizan  $\log(n)$  operaciones en promedio.

```

Object get(String s) {
    int ini=0;
    int fin=total-1;
    while (ini<fin) {
        int m=(ini+fin)/2;
        if (s.compareTo(llave[m])==0)
            return valor[m];
        else if (s.compareTo(llave[m])<0)
            m=ini+1;
        else
            m=fin-1;
    }
    return null
}

```

Eliminar es exactamente lo contrario de agregar. Se busca el elemento y se corren todos los mayores para eliminarlo.

```

void remove(String s) {
    for (int i=0; i<llave.length;i++)
        if (llave[i].equals(l)) {
            for (int j=i;j<total;j++) {
                llave[j]=llave[j+1];
                valor[j]=valor[j+1];
            }
            total--;
            return;
        }
}

```

## AL MARGEN

### Pon tus conocimientos a prueba

- Cuál de las dos implementaciones es la mejor
- Qué modificaciones hay que realizar para el que el diccionario pase a ser un diccionario de antónimos?