

ESTRUCTURAS DE DATOS

TIPOS ABSTRACTOS DE DATOS ARBORESCENTES

El TAD Conjunto

Manuel Montenegro Montes
Departamento de Sistemas Informáticos y Computación
Facultad de Informática – Universidad Complutense de Madrid

Conjuntos

Al principio del curso vimos el TAD Conjunto pero solo para letras. Ahora vamos a ver que la implementación puede servir para otro tipo de datos distintos.

- Un **conjunto** es una colección de elementos del mismo tipo.
- ¿Cuál es la diferencia entre un conjunto y una lista?

Lista

- El orden de los elementos es relevante:

No están ordenados de la misma manera. Son por tanto listas distintas.

$$[1, 4, 5] \neq [4, 5, 1]$$

- Pueden contener elementos duplicados:

El hecho de que el 4 esté repetido es relevante.

$$[1, 4, 4, 5] \neq [1, 4, 5]$$

Aquí importan cuantos elementos de cada tipo (por así decirlo) y en qué orden están.

Conjunto

- No existe el concepto de orden entre elementos:

Mismo conjunto aunque tengan distinto orden.


$$\{1, 4, 5\} = \{4, 5, 1\}$$

- La existencia de duplicados es irrelevante:

$$\{1, 4, 5\} \cup \{4\} = \{1, 4, 5\}$$

Aquí nos da igual el número de cuatros o el número de cincos, nos da igual el orden que tengan... dos conjuntos son iguales si tienen los mismos elementos, nos da igual el contador de cada elemento y su orden

Modelo de conjuntos

- Varias formas de implementar un conjunto.
- Cuando el conjunto está implementado y tan solo tenemos que utilizarlo, pensamos en términos del **modelo**.  ya vimos la diferencia entre el modelo y la representación
- Cada instancia del TAD Conjunto representa un **conjunto finito**.

$$\{x_1, x_2, x_3, \dots, x_n\} \rightarrow n \neq \infty$$

siempre conjuntos finitos,

Operaciones en el TAD conjunto

- Constructoras:
 - Crear un conjunto vacío: **create_empty**
- Mutadoras:
 - Añadir un elemento al conjunto: **insert**
 - Eliminar un elemento del conjunto: **erase**
- Observadoras:
 - Averiguar si un elemento está en el conjunto: **contains**
 - Saber si el conjunto está vacío: **empty**
 - Saber el tamaño del conjunto: **size**

Operaciones constructoras y mutadoras

$\{ \text{true} \}$

create_empty() $\rightarrow (S: \text{Set})$

$\{ S = \emptyset \}$

vamos, que la raíz apunte a null

Para cualquier entrada devuelve el conjunto vacío

CONSTRUCTORA

$\{ \text{true} \}$

erase($x: \text{Elem}, S: \text{Set}$)

$\{ S = \text{old}(S) - \{x\} \}$

Conjunto antiguo menos el elemento x.

$\{ \text{true} \}$

insert($x: \text{Elem}, S: \text{Set}$)

$\{ S = \text{old}(S) \cup \{x\} \}$

Conjunto antiguo u old y además el nuevo elemento x.

OPERACIONES MUTADORAS.

Operaciones observadoras

$\{ \text{true} \}$

contains($x: \text{Elem}, S: \text{Set}$) $\rightarrow (b: \text{bool})$

$\{ b \Leftrightarrow x \in S \}$

$\{ \text{true} \}$

empty($S: \text{Set}$) $\rightarrow (b: \text{bool})$

$\{ b \Leftrightarrow S = \emptyset \}$

$\{ \text{true} \}$

size($S: \text{Set}$) $\rightarrow (n: \text{int})$

$\{ n = |S| \}$

Esto representa el cardinal del conjunto.

Interfaz en C++

```
class set {  
public:  
    set();  
    set(const set &other);  
    ~set();  
  
    void insert(const T &elem);  
    void erase(const T &elem);  
  
    bool contains(const T &elem) const;  
    int size() const;  
    bool empty() const;  
  
private:  
    // ???  
};
```

clase conjunto

OPERACIONES CONSTRUCTORAS Y DESTRUCTORAS.

OPERACIONES MUTADORAS.

OPERACIONES OBSERVADORAS.

Depende de la implementación.

Dos implementaciones

- Mediante **listas**.
- Mediante **árboles binarios de búsqueda**.

Clase especial de árboles binarios que vamos a ver a lo largo de la semana.

El TAD Conjunto es un tipo YA existente en C++ que podemos usar utilizando CREO `#include <set>`