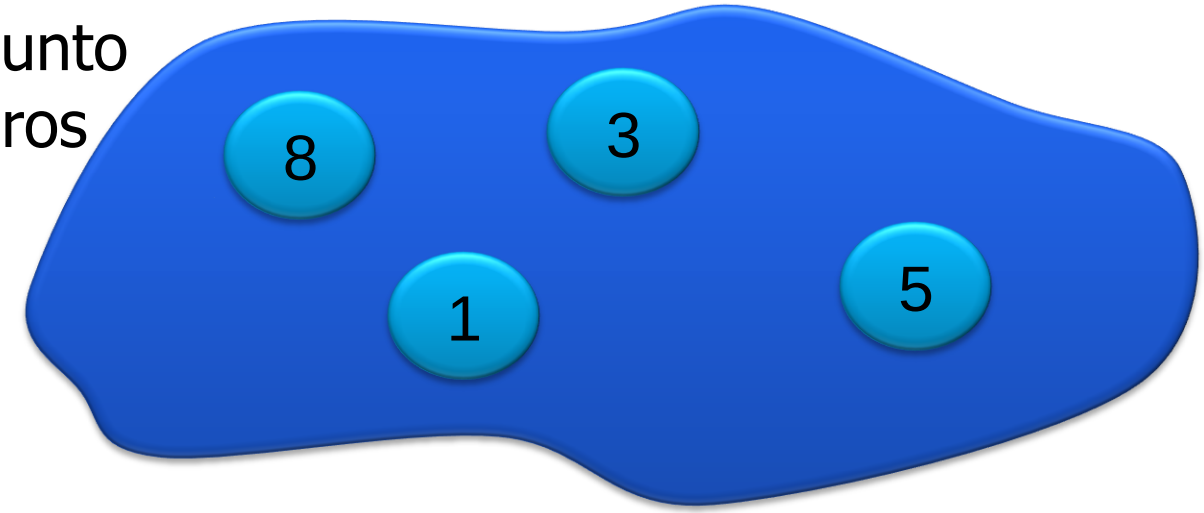
TAD

- ✓TAD Conjunto
- ✓Análisis del TAD Conjunto
- ✓Posibles implemantaciones

TAD Conjunto
Definición

- ✓ **Conjunto:** Colección no ordenada de elementos (o miembros) distintos.
- ✓ **Elemento:** Cualquier cosa, puede ser un elemento primitivo o, a su vez, un conjunto.

C: Conjunto de enteros



TAD Conjunto

- Los elementos son del mismo tipo: **ConjuntoU** (conjuntos de enteros, de caracteres, de cadenas ...)

- ¿En qué se diferencia el TAD Conjunto del TAD Lista?

TAD Conjunto

- ✓ Puede existir una relación de orden en el conjunto.
- ✓ **Relación "<" de orden** en un conjunto C:
  - **Propiedad transitiva**: para todo *a*, *b*, *c*, si (*a*<*b*) y (*b*<*c*) entonces (*a*<*c*).
  - **Orden total**: para todo *a*, *b*, sólo una de las afirmaciones (*a*<*b*), (*b*<*a*) o (*a*=*b*) es cierta.

- ✓ … **<u>colección no ordenada</u>**… → Se refiere al orden de inserción de los elementos.

TAD Conjunto. Repaso de notación de conjuntos

- ✓ **Definición**:

Por extensión	Mediante proposiciones
A= {a, b, c, ..., z}	C= {x | proposición de x}
B= {1, 4, 7} = {4, 7, 1}	D= {x | x es primo y menor que 90}

- ✓ **Pertenencia**: $x \in A$
- • **No pertenencia**: $x \notin A$
- ✓ **Conjunto vacío**: $\emptyset$
- • **Conjunto universal**: U
- ✓ **Inclusión**: $A \subseteq B$
- • **Intersección:** $A \cap B$
- ✓ **Unión**: $A \cup B$
- • **Diferencia**: $A - B$

TAD Conjunto. Operaciones mas comunes

C: Conjunto de todos los ConjuntoU

$a, b, c \in C$; $x \in U$

- Vacío : $\rightarrow C$ — $a := \emptyset$
- Unión : $C \times C \rightarrow C$ — $c := a \cup b$
- Intersección : $C \times C \rightarrow C$ — $c := a \cap b$
- Diferencia : $C \times C \rightarrow C$ — $c := a - b$
- Combina : $C \times C \rightarrow C$ — $c := a \cup b$, con $a \cap b = \emptyset$
- Miembro : $U \times C \rightarrow B$ — $x \in a$

Conjuntos. Operaciones más comunes

- Inserta : U x C $\rightarrow$ C $a := a \cup \{x\}$
- Suprime : U x C $\rightarrow$ C $a := a - \{x\}$
- Min : C $\rightarrow$ U $\min_{\forall x \in a}(x)$
- Max : C $\rightarrow$ U $\max_{\forall x \in a}(x)$
- Igual : C x C $\rightarrow$ B $a == b$

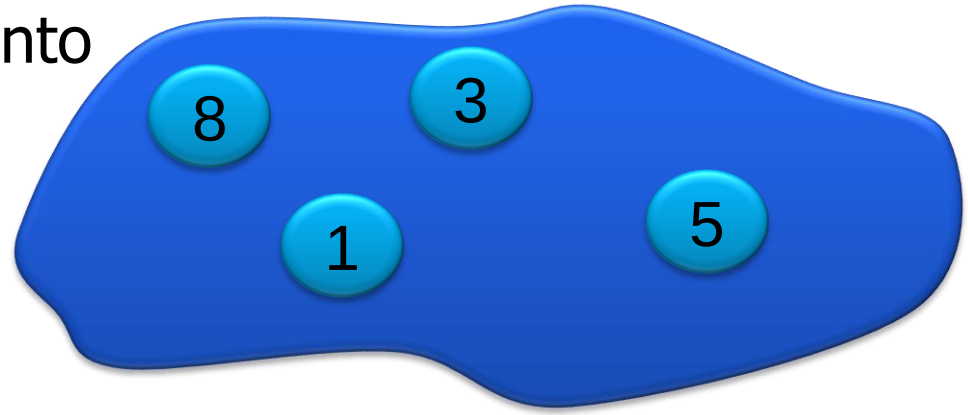
- … **elementos distintos**… → Si insertamos un elemento que ya pertenece, obtenemos el mismo conjunto.

Implementaciones básicas

- **Problema:** ¿Cómo representar el tipo conjunto, de forma que las operaciones se ejecuten rápidamente, con un uso razonable de memoria?
- **Respuesta:**
- Dos tipos de **implementaciones básicas**:
  - Mediante arrays de booleanos.
  - Mediante listas de elementos.
- La mejor implementación depende de cada aplicación concreta:
  - Operaciones más frecuentes en esa aplicación.
  - Tamaño y variabilidad de los conjuntos usados.
  - Etc.

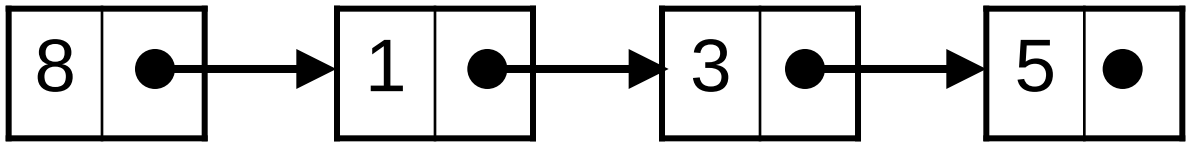
Implementaciones básicas

C: Conjunto



1	2	3	4	5	6	7	8	9	10
1	0	1	0	1	0	0	1	0	0

Array de booleanos



Lista de elementos

d

Implementaciones básicas

Mediante array de booleanos

- **Idea:** Cada elemento del conjunto universal se representa con 1 bit. Para cada conjunto concreto A, el bit asociado a un elemento vale:

  1 - Si el elemento pertenece al conjunto A

  0 - Si el elemento no pertenece a A

- **Definición**:

  tipo

  ConjuntoU = **array** [1..RangoU] **de** booleano

  Donde RangoU es el tamaño del conj. universal.

Mediante array de booleanos

- **Ejemplo**: RangoU = {a, b, …, g}

  C= ConjuntoU

  A = {a, c, d, e, g}

  B = {c, e, f, g}

a	b	c	d	e	f	g
1	0	1	1	1	0	1

A: Conjunto[a..g]

a	b	c	d	e	f	g
0	0	1	0	1	1	1

B: Conjunto[a..g]

- Unión, intersección, diferencia: se transforman en las operaciones booleanas adecuadas.

Mediante array de booleanos

```
Procedure Union (A, B: ConjuntoU; var C: ConjuntoU);
Var i :integer;
begin
    for I:=1 to RangoU do
            C[i]:= A[i] OR B[i]
end;
```

Mediante arrays de booleanos

```
Procedure Suprime (x: Integer; var C: ConjuntoU)
    C[x]:= 0
```

- ¿Cómo serían: Igual, Min, Max, ...?

Mediante array de booleanos

Ventajas

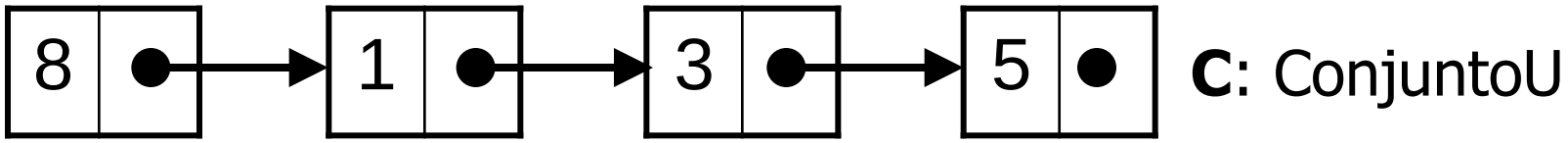
- Operaciones muy sencillas de implementar.
- No hace falta usar memoria dinámica.
- El tamaño usado es **proporcional al tamaño del conjunto universal**, independientemente de los elementos que contenga el conjunto.
- ¿Ventaja o inconveniente?

Mediante listas de elementos

- **Idea**: Guardar en una lista los elementos que pertenecen al conjunto.
- **Definición**:

  tipo

```
conjuntoU = ^nodoconj
nodoconj = record
        dato: integer;
        sig; conjuntoU;
 end;
```



C = { 5,3,1 8}

Mediante listas de elementos

Ventajas:

- Utiliza espacio proporcional al tamaño del conjunto representado (no al conjunto universal).
- El conjunto universal puede ser muy grande, o incluso infinito.

Inconvenientes:

- Las operaciones son menos eficientes si el conjunto universal es reducido.
- Gasta más memoria y tiempo si los conjuntos están muy llenos.
- Más complejo de implementar.

Mediante listas de elementos

```
Function Miembro (x: integer; C: ConjuntoU): booleano
var  actual: nodoconj;
 Begin
     actual: = C;
    while actual <> nil and actual.dato <>x)  do
          actual:= actual^.sig;
    miembro:= (actual  <> nil);
End;
```

Mediante listas de elementos

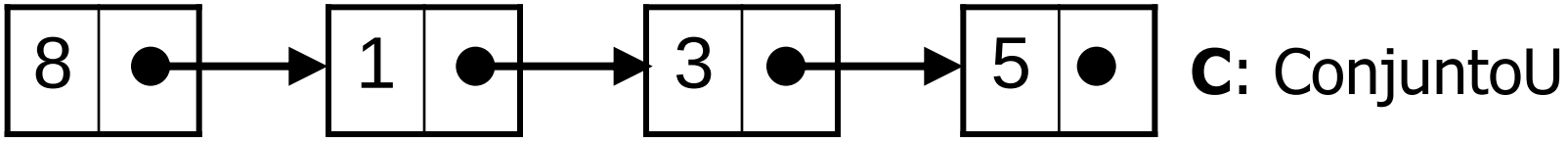
- ¿Cuánto tiempo tardan las operaciones anteriores? Suponemos una lista de tamaño **n** y otra **m** (o ambas de tamaño **n**).

- ¿Cómo sería Unión, Diferencia, Inserta, Suprime, etc.?

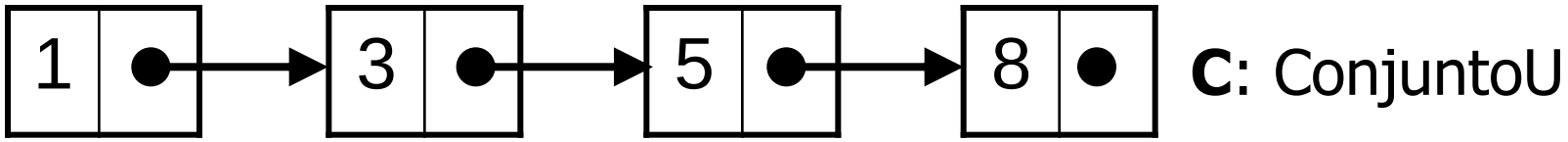
- **Inconveniente:** Unión, Intersección y Diferencia recorren la lista B muchas veces (una por cada elemento de A).
- ¿Se puede mejorar usando listas ordenadas?

Mediante listas de elementos

- Listas no ordenadas.



- Listas ordenadas.



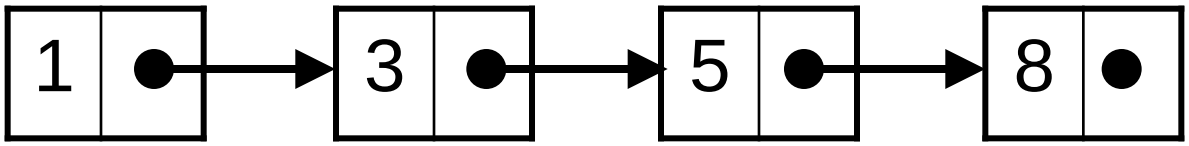
MEJORAS AL USAR LISTAS ORDENADAS:

- **Miembro, Inserta, Suprime**: se debe Parar si encontramos un elemento mayor que el buscado.
- **Unión**, **Intersección**, **Diferencia**: Recorrido simultáneo (y único) de ambas listas.

Mediante lista ORDENADA de elementos

```
function Miembro (x: integer; C: Conjunto): booleano
    var  actual: nodoconj
begin
     actual := C; Primero(C)
     while (actual <> nil and actual.dato <x) do
          actual:= actual^.sig;
    Miembro: = (actual.dato = x);
End;
```

;

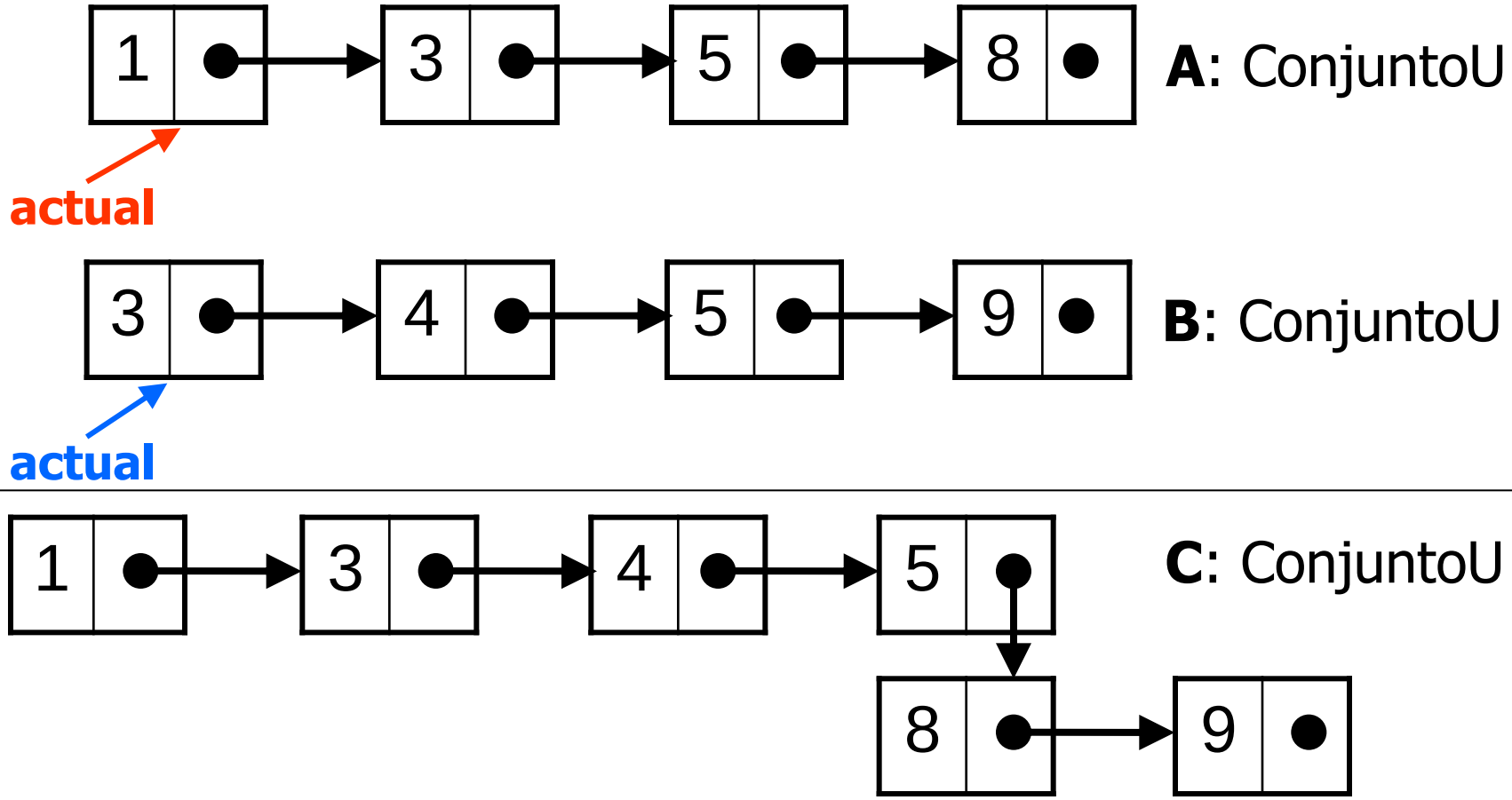


C: ConjuntoU

- ¿Cuál es el tiempo de ejecución ahora?

Mediante listas de elementos

- **Unión:** Idea parecida al procedimiento de mezcla, en la ordenación por mezcla.



Conclusiones

Arrays de booleanos:

- muy rápida para las operaciones de inserción y consulta.
- Inviable si el tamaño del conjunto universal es muy grande.

Listas de elementos:

- uso razonable de memoria, proporcional al tamaño usado.
- Muy ineficiente para la inserción y consulta de un elemento.