

# Métodos Avanzados de Programación Científica y Computación

Jesús Cigales Canga

## Tema 5. Métodos algorítmicos para resolución de problemas

# ¿Cómo estudiar este tema?

## Material de Estudio

[Tema 5](#)

## Material complementario

[Algorithms](#)

[Uso de herísticas](#)

[Ejercicios resueltos de programación dinámica](#)

## Magistrales

### Magistrales



### Material complementario

[Test Tema 5.pdf](#)

# Algoritmos



<https://es.wikipedia.org/wiki/Al-Juarismi>

## Algoritmo

Conjunto de reglas que se aplican para realizar algún cálculo y obtener un resultado.

## Clasificación:

- Por la forma
- Por la función

# Organización



# Algoritmos de ordenación de listas.

## Método de la Burbuja

Lista						Hay intercambio
35	36	17	73	8	0	No
35	36	17	73	8	0	Si
35	17	36	73	8	0	No
35	17	36	73	8	0	si
35	17	36	8	73	0	Si
35	17	36	8	0	73	El 73 queda ordenado

Bucle externo  $n$  veces  
Bucle interno  $n - i$  veces

$$O(n^2)$$

```
for i in range(1, n):
    for j in range(n-i):
        numComparaciones += 1
        if lista[j] > lista[j+1]:
            'intercambio de valores'
            lista[j], lista[j+1] = lista[j+1], lista[j]
```

# Algoritmos de ordenación de listas.

## Ordenación por selección

Lista						Comparaciones/intercambio
35	36	17	73	8	0	5/Si
0	36	17	73	8	35	4/Si
0	8	17	73	36	35	3/No
0	8	17	73	36	35	2/Si
0	8	17	35	36	73	1/Si
0	8	17	35	36	73	Fin

Bucle externo  $n - 1$  veces  
Bucle interno  $n - i$  veces

$$O(n^2)$$

```
for i in range(n - 1):
    minimo = i
    for j in range(i + 1, n):
        numComparaciones += 1
        if lista[j] < lista[minimo]:
            minimo = j
    lista[i], lista[minimo] = lista[minimo], lista[i]
```

# Divide y vencerás.

## Ordenación por mezcla

Lista de números						
35	36	17	73	8	0	
La lista se divide en dos						
35	36	17		73	8	0
Se hace llamada recursiva para ordenar cada una de las partes						
17	35	36		0	8	73
Se mezclan						
0	8	17	35	36	73	

```

izquierda = lista[:medio]
derecha = lista[medio:]
izquierda = mergeSort(izquierda)
derecha = mergeSort(derecha)
return merge(izquierda, derecha)

```

$$t(n) = 2t\left(\frac{n}{2}\right) + g(n), g(n) \in \Theta(n).$$

$$O(n \log n)$$

$$t(n) \in \begin{cases} \Theta(n^k) & \text{si } l < b^k \\ \Theta(n^k \log n) & \text{si } l = b^k, \text{ siendo } t(n) = lt\left(\frac{n}{b}\right) + g(n), \\ \Theta(n^{\log_b l}) & \text{si } l > b^k \end{cases}$$

$k = 1$   
 $2 = 2^G$

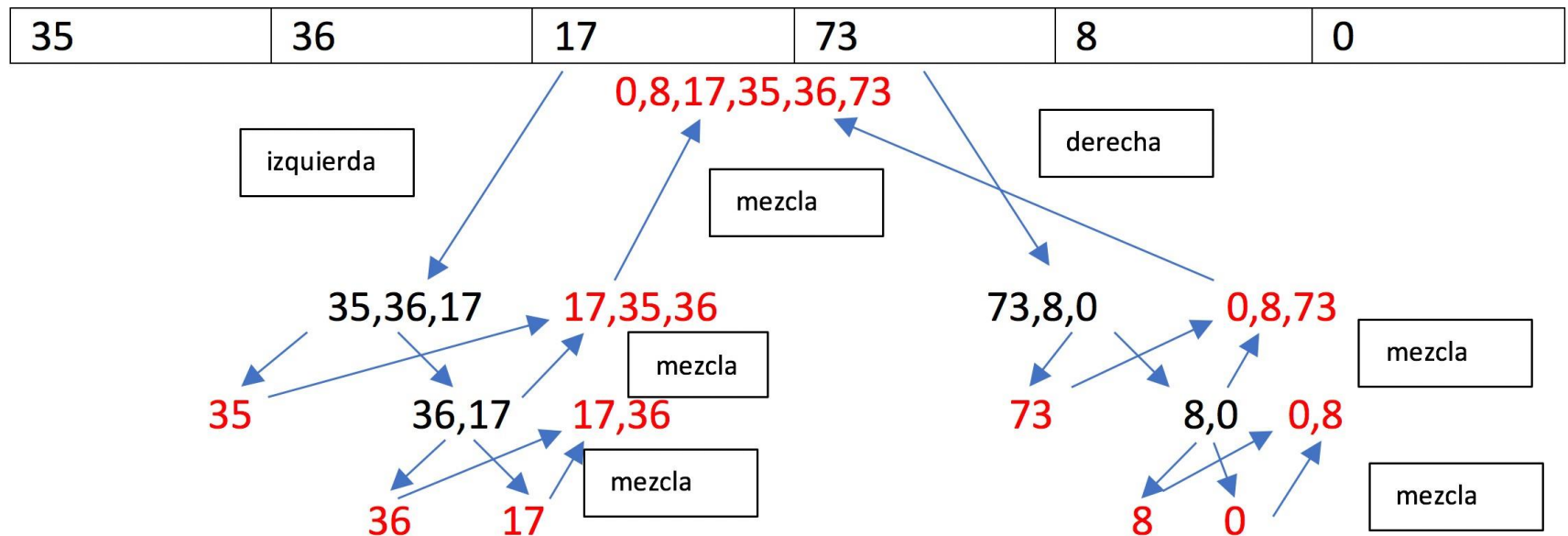
```

for k in range(0, len(lista)):
    numComparaciones += 1
    if listaA[i] < listaB[j]:
        lista[k] = listaA[i]
        i += 1
    else:
        lista[k] = listaB[j]
        j += 1

```

# Divide y vencerás.

## Ordenación por mezcla





# Algoritmos de búsqueda

## Binaria recursiva

$$t(n) = t\left(\frac{n}{2}\right) + g(n) \quad g(n) \in O(1)$$

$$O(\log n)$$

```
if primero>ultimo:
    return -1;
medio = (primero+ultimo) // 2
if lista[medio] < x:
    return busquedabinariarec(lista,x,medio+1,ultimo)
elif (lista[medio] > x):
    return busquedabinariarec(lista,x,primero,medio-1)
else:
    return medio
```

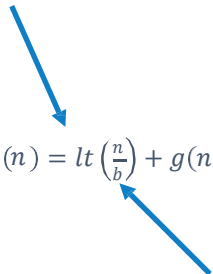
### Lista ordenada

- Valor central=buscado termina
- Sino busca derecha o izquierda

$k = 0$   
 $1 = 2^1$

1

2

$$t(n) \in \begin{cases} \theta(n^k) & \text{si } l < b^k \\ \theta(n^k \log n) & \text{si } l = b^k, \text{ siendo } t(n) = lt\left(\frac{n}{b}\right) + g(n), \\ \theta(n^{\log_b l}) & \text{si } l > b^k \end{cases}$$


UNIVERSIDAD  
INTERNACIONAL  
DE LA RIOJA

**unir**

[www.unir.net](http://www.unir.net)