Algoritmos de ordenación

Algoritmos de ordenación
Objetivos

- Comprender la diferencia entre ordenación rápida y lenta.
- Comprender el algoritmo de ordenación de la burbuja.
- Comprender el algoritmo de ordenación por inserción.
- Comprender el algoritmo de ordenación por selección.
- Comprender el algoritmo de ordenación Shellsort.
- Comprender el algoritmo de ordenación Quicksort.
- Comprender el algoritmo de ordenación Mergesort.
- Comprender el algoritmo de ordenación Heapsort.

Algoritmos de ordenación
Comprender la diferencia entre ordenación rápida y lenta

Vamos a estudiar brevemente algunos algoritmos de ordenación **lentos** $\Theta(n^2)$, y algunos algoritmos de ordenación **rápidos** $\Theta(n\log n)$.

Los algoritmos lentos:

- Son sencillos de implementar y comprender.
- Se comportan mal cuando la entrada es muy grande.

Los algoritmos rápidos:

- Son complejos.
- Se comportan bien cuando la entrada es muy grande.

Para **problemas pequeños** usaremos **algoritmos lentos**, por simplicidad, para **problemas grandes** usaremos **algoritmos grandes**, por eficiencia.

Algoritmos de ordenación
Comprender la diferencia entre ordenación rápida y lenta

Supondremos que cada valor queda identificado por una clave (contenida en un vector numérico), y que existe un método swap que realiza el intercambio entre dos valores.

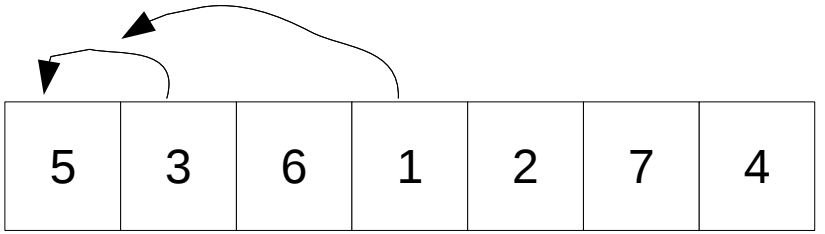
```
void swap(int v[],int a,int b) {
  int aux = v[a];
  v[a] = v[b];
  v[b] = v[a];
}
```

Algoritmos de ordenación
Comprender el algoritmo de ordenación de la burbuja

Ordenación de ordenación de la burbuja:

```
void ord_burbuja(int v[],int n) {
  int i,j;
  for (i = 0;i < n;i++)
    for (j = i+1;j < n;j++)
      if (v[i] > v[j])
        swap(v,i,j);
}
```

O(n²)

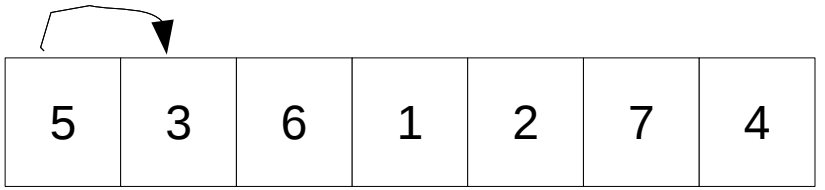


Algoritmos de ordenación
Comprender el algoritmo de ordenación por inserción

Algoritmo de ordenación por inserción:

```
void ord_insercion(int v[],int n) {
  int i,j,indice;
  for (i = 1;i < n;i++) {
    indice = v[i];
    j = i-1;
    while (j >= 0 && v[j] > indice) {
      v[j+1] = v[j];
      j--;
    }
    v[j+1] = indice;
  }
}
```

$O(n^2)$

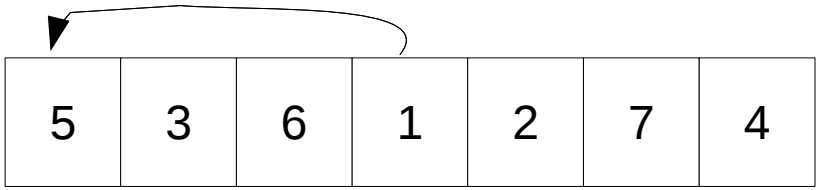


Algoritmos de ordenación
Comprender el algoritmo de ordenación por selección

Algoritmo de ordenación por selección:

```
void ord_seleccion(int v[],int n) {
  int i,j,minimo;
  for (i = 0;i < n-1;i++) {
    minimo = i;
    for (j=i+1;j < n;j++) {
      if (v[minimo] > v[j])
        minimo = j;
    }
    swap(v,minimo,i);
  }
}
```

O(n²)

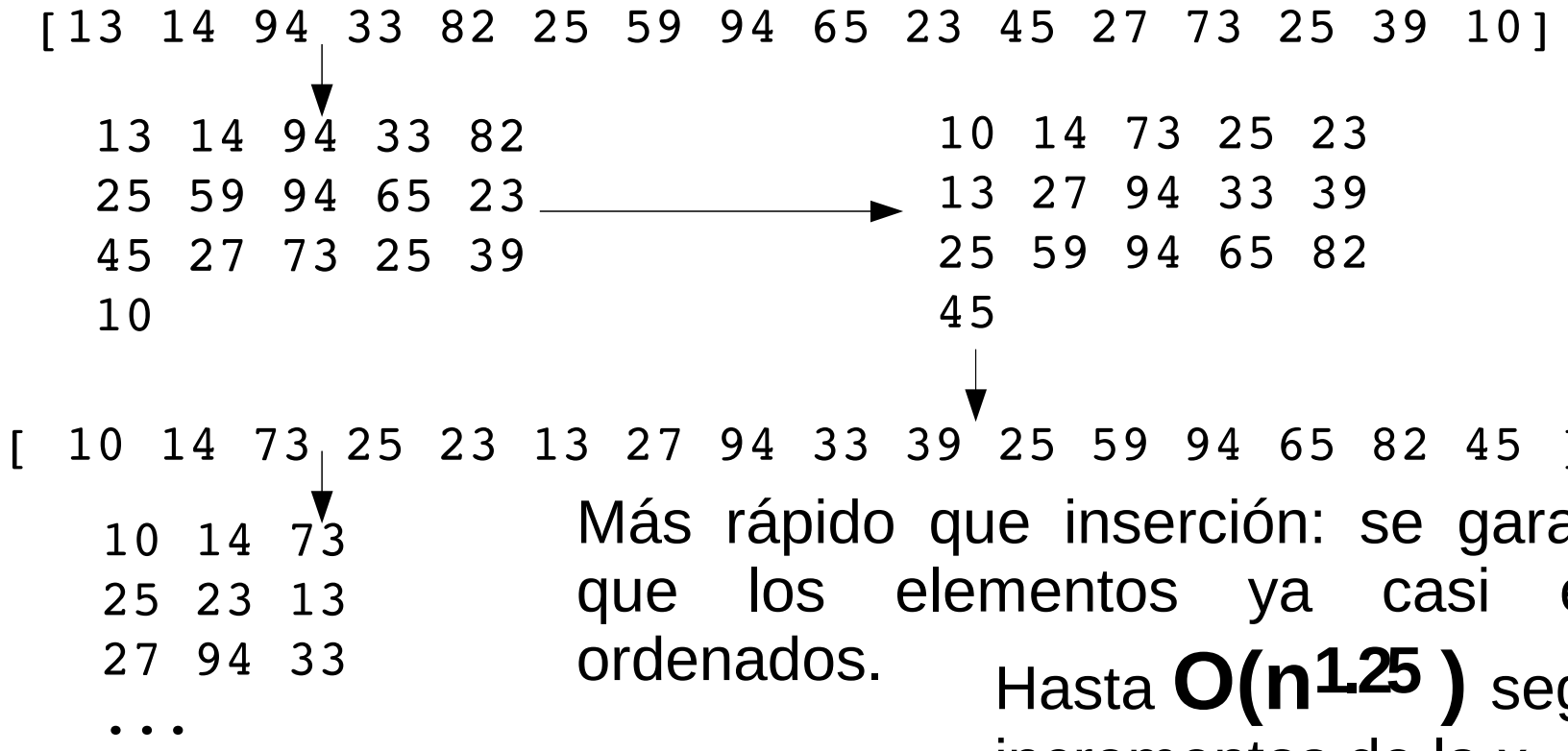


Algoritmos de ordenación
Comprender el algoritmo de ordenación Shellsort

Algoritmo de ordenación Shellsort:

Considera el vector una tabla de x columnas.

Ordena por columnas usando inserción, divide x entre 2 y repite.
La última iteración (x==1) equivale a **ordenación por inserción**.



Más rápido que inserción: se garantiza que los elementos ya casi están ordenados.

Hasta $\mathbf{O(n^{1.25})}$ según incrementos de la x

Algoritmos de ordenación
Comprender el algoritmo de ordenación Quicksort

Algoritmo de ordenación Quicksort:

Escoge un elemento pivote (idealmente la mediana).

`[13 14 94 33 82 25 59 94 65 23 45 27 73 25 39 10]`

A la izquierda se pasan los elementos <= que él, a la derecha >

`[13 14 25 23 27 25 10 33 94 82 59 94 65 45 73 39]`

Se aplica quicksort a la izquierda del pivote y a la derecha.

`[13 14 25 23 27 25 10 33 94 82 59 94 65 45 73 39]`

Para los casos más pequeños (n=2) se resuelven con condicionales.

Promedio: **O(nlogn)**

Peor (evitable mediante elección de pivote): **$O(n^2)$**

Algoritmos de ordenación
Comprender el algoritmo de ordenación Mergesort

Algoritmo de ordenación Mergesort:

Partiendo de un vector:

[13 14 94 **33** 82 25 59 94 65 23 45 27 73 25 39 10]

Se divide en dos de aproximadamente mismo tamaño.

[13 14 25 23 27 25 10 **33**] [94 82 59 94 65 45 73 39]

Se aplica mergesort a ambos vectores (¡recursivamente!)

[10 13 14 23 25 25 27 **33**] [39 45 59 65 73 82 94 94]

Se juntan estos dos vectores, iterando por ambos a la vez y escogiendo el elemento más pequeño de entre los dos apuntados.

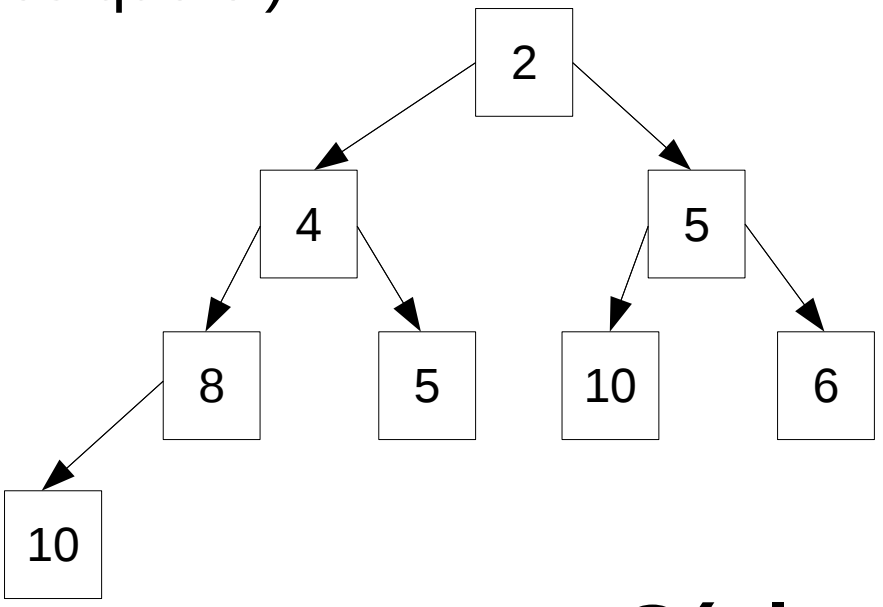
[**10** 13 14 23 25 25 27 **33**] [39 45 59 65 73 82 94 94]

Peor caso: **O(nlogn)**

Algoritmos de ordenación
Comprender el algoritmo de ordenación Heapsort

Algoritmo de ordenación Heapsort:

Inserta los elementos en un árbol parcialmente ordenado. Se almacenan ordenados (un nodo solo puede ser padre de elementos mayores que él).



Peor caso: **O(nlogn)**

Al extraer siempre la raíz, obtenemos el vector ya ordenado.

Algoritmos de ordenación
¿Qué hemos aprendido?

- **Diferencia entre ordenación rápida y lenta.**
- **Algoritmo de ordenación de la burbuja.**
- **Algoritmo de ordenación por inserción.**
- **Algoritmo de ordenación por selección.**
- **Algoritmo de ordenación Shellsort.**
- **Algoritmo de ordenación Quicksort.**
- **Algoritmo de ordenación Mergesort.**
- **Algoritmo de ordenación Heapsort.**