



Algoritmia
Grado en Ingeniería Informática del Software
Escuela de Ingeniería Informática – Universidad de Oviedo

Ordenación

Juan Ramón Pérez Pérez

jrpp@uniovi.es

En qué consisten los algoritmos de ordenación

Dado un **conjunto de n elementos** (a_1, a_2, \dots, a_n) y una **relación de orden total** (\leq) sobre ellos, el problema de la **ordenación** consiste en encontrar una **permutación** de esos elementos que cumpla la relación establecida.

Métodos de ordenación

Vamos a estudiar distintos **métodos**, para ello consideramos que todos los elementos pueden estar en **memoria**. Realizaremos los ejemplos sobre un vector de **enteros** (esto último es fácilmente generalizable).

Criterios para estudiar los distintos métodos

- Número de **intercambios**
- Número de **comparaciones**
- Número total de pasos
 - **Complejidad**
- Estabilidad del método
- Estudiar caso **mejor**, **peor** y **medio**. ¿Cuándo se produce?

Ordenación por Inserción directa

```
public static void insercion(int[] a)
{
    int n= a.length;
    for (int i=1; i<n; i++)
    {
        int x=a[i];
        int j=i-1;
        while (j>=0 && x<a[j])
        {
            a[j+1]=a[j]; // desplaza el elemento del array
            j=j-1;
        }
        a[j+1]=x; // mete elemento a ordenar en el hueco
    }
}
```



Video inserción directa (otra versión):

<https://youtu.be/gTxFxgvZmQs>

Análisis del algoritmo

- Comparaciones

- Gran número

- Intercambios

- Gran número. No tantos como otros métodos

Complejidades

- Caso mejor:
 - $O(n)$
- Caso peor:
 - $O(n^2)$
- Caso medio:
 - $O(n^2)$

Ordenación por Selección

```
public static void seleccion (int[] v)
{
    int n= v.length;
    int posmin;
    for (int i=0; i<n-1; i++)
    {
        // buscar posición del mas pequeño
        posmin= i;
        for (int j= i+1; j < n; j++)
            if (v[j] < v[posmin])
                posmin= j;
        intercambiar(v,i,posmin);
    } // for
}
```

Método Intercambiar

```
private static void intercambiar (int[] v, int i,  
int j)  
{  
    int t;  
    t= v[i]; v[i]= v[j]; v[j]= t;  
}
```



Video selección: <https://youtu.be/boOwArDShLU>

Análisis del algoritmo

- Número de intercambios
 - Mínimo.
- Comparaciones
 - Gran número
 - Es fácilmente predecible el número de intercambios y comparaciones a partir del número de elementos

Complejidades

- Caso mejor:
 - $O(n^2)$
- Caso peor:
 - $O(n^2)$
- Caso medio:
 - $O(n^2)$

Ordenación por Burbuja

```
public static void burbuja (int[] a)
{
    int n= a.length;
    for (int i=0;i<=n-2;i++)
        for (int j=n-1;j>i;j--)
            if (a[j-1]>a[j])
                intercambiar(a,j-1,j);
}
```



Video burbuja: <http://youtu.be/t-igeo1xxEg?t=26s>

https://youtu.be/1JvYAXT_o64

Análisis del algoritmo

- Comparaciones:
 - Gran número
- Intercambios
 - Gran número

Complejidad

- Caso mejor:
 - $O(n^2)$
- Caso peor:
 - $O(n^2)$
- Caso medio:
 - $O(n^2)$

Rápido (Quicksort)

- Introduce la idea de particiones
- Para ello se elige un elemento como pivote
- Esto permite independizar al método de la disposición de los datos iniciales
- Es un método recursivo

Rápido

```
private static void rapido(int[] v,  
                           int iz, int de)  
{  
    int m;  
    if (de>iz)  
    {  
        m=particion(v,iz,de);  
        rapido(v,iz,m-1);  
        rapido(v,m+1,de);  
    }  
}
```

Variantes en la elección del pivote

- Primer elemento
- Último elemento
- Elemento central
- Elemento aleatorio
- Mediana de todos los elementos
- Mediana a 3

Elección del pivote y creación de particiones

```
private static int particion(int[] v,int iz,int de)
{
    int i, pivote;
    intercambiar(v, (iz+de)/2,iz);
    //el pivote es el de centro y se cambia con el
    primero
    pivote= v[iz];
    i= iz;
    for (int s= iz+1; s <= de; s++)
        if (v[s] <= pivote)
        {
            i++;
            intercambiar(v,i,s);
        }
    intercambiar(v,iz,i); //se restituye pivote
    return i; // posición en que queda el pivote
}
```



Video Quicksort (comparado con Burbuja):

<https://youtu.be/aXXWXz5rF64>

Ej. 2.1: Realizar la traza para ordenar la secuencia utilizando el algoritmo Rápido (Quicksort)

- Partiendo de la siguiente secuencia de enteros:
 - (8, 3, 2, 1, 7, 5)
- Utilizar el elemento central como pivote.
- Indicar en cada paso, cuales son las particiones obtenidas.

Análisis del algoritmo

- Revisaremos este algoritmo en el siguiente tema ya que es del tipo *Divide y Vencerás*.
- Caso mejor: $O(n \log n)$
- Caso peor: $O(n^2)$
- Caso medio: $O(n \log n)$

Otros métodos de ordenación

- Mezcla (*Mergesort*) (<https://youtu.be/es2T6KY45cA>)
- Ordenación mediante montículos (*Heapsort*) (<https://youtu.be/H5kAcmGOn4Q>)
- Shellsort (<https://youtu.be/QTtHQVRiDo4>)
- ...

Ejercicios propuestos

- Detectar las diferencias entre el código de cada uno de los métodos y los videos que muestran su realización
- Utilizar la implementación proporcionada para **contar** el número de **comparaciones** e **intercambios** que se producen alguno de los métodos, para el caso mejor, peor y un caso intermedio. Representar estos valores en una tabla función de los elementos del array.