

ALGORITMOS DE ORDENAMIENTO

1. Burbuja
2. Selección
3. Inserción
4. Otros

[Ejercicios](#)

Definición

- Debido a que las estructuras de datos son utilizadas para almacenar información, es necesario poder recuperarla de manera eficiente, por ende es deseable que aquella esté ordenada.
- Existen varios métodos para ordenar las diferentes estructuras de datos básicas.
- Los algoritmos de ordenamiento nos permiten ordenar los valores de vectores o matrices que han sido cargados o inicializados aleatoriamente.
- Existen distintos métodos, los cuales varían en la cantidad de comparaciones que suceden, el tiempo que demoran y la cantidad de código necesario para implementarlo.

Tipos de Algoritmos

- Para poder ordenar una cantidad determinada de números almacenadas en un vector o matriz, existen distintos métodos (algoritmos) con distintas características y complejidad.
- La clasificación de los mismos se basa en el mecanismo utilizado para realizar el ordenamiento, ya sea a través de operaciones simples, como las iteraciones o más complejos, como la utilización de algoritmos recursivos.

1. Métodos Iterativos

- Estos métodos son simples de entender y de programar ya que son iterativos, simples ciclos y sentencias que hacen que el vector pueda ser ordenado.
- Dentro de los Algoritmos iterativos encontramos:
 1. Burbuja
 2. Inserción
 3. Selección
 4. Shellsort

2. Métodos Recursivos

- Estos métodos son aún más complejos, requieren de mayor atención y conocimiento para ser entendidos.
- Son rápidos y efectivos, utilizan generalmente la técnica Divide y Vencerás, que consiste en dividir un problema grande en varios pequeños para que sea más fácil resolverlos.
- Mediante llamadas recursivas, es posible que el tiempo de ejecución y de ordenamiento sea más óptimo.
- Dentro de los algoritmos recursivos encontramos:
 1. Ordenamiento por Mezclas (merge)
 2. Ordenamiento Rápido (quick)

Método Burbuja

- El método de la Burbuja es uno de los más simples, es tan fácil como comparar todos los elementos de una lista contra todos, si se cumple que uno es mayor o menor a otro, entonces los intercambia de posición.
1. Comenzando desde el inicio del arreglo, se compara cada par de elementos adyacentes.
 2. Si ambos no están ordenados (el segundo es menor que el primero), se intercambian sus posiciones.
 3. En cada iteración, un elemento menos necesita ser evaluados (el último), ya que no hay más elementos a su derecha que necesiten ser comparados, puesto que ya están ordenados.

6 5 3 1 8 7 2 4

Implementación de Algoritmo Burbuja

```
private static void burbuja() {  
  
    int[] arreglo = new int[] { 5, 6, 1, 0, 3 };  
    int aux = -1;  
  
    for (int i = 1; i <= arreglo.length; i++) {  
        for (int j=0 ; j< arreglo.length - 1; j++) {  
            if (arreglo[j] > arreglo[j+1]) {  
                aux = arreglo[j];  
                arreglo[j] = arreglo[j + 1];  
                arreglo[j + 1] = aux;  
            }  
        }  
    }  
}
```

Método de Inserción

1. El bucle principal del ordenamiento por inserción va examinando sucesivamente todos los elementos del vector o arreglo desde el segundo hasta el n-ésimo;
2. Toma cada valor y lo inserta en el lugar adecuado entre sus predecesores dentro del vector o arreglo.

6 5 3 1 8 7 2 4

ALGORITMOS DE ORDENAMIENTO - Inserción

Implementación de Algoritmo de Inserción

```
private static void insercion() {  
  
    int[] arreglo = new int[] { 5, 6, 1, 0, 3 };  
    int aux = -1;  
    int j;  
  
    // recorre el arreglo desordenado  
    for (int i = 1; i < arreglo.length; i++)  
    {  
        aux = arreglo[i];  
  
        j = i - 1;  
  
        // Mueve los elementos del arreglo [0..i-1] que  
        // sean mayores al 'aux', una posición adelante  
        // de su posición actual.  
        while (j >= 0 && arreglo[j] > aux)  
        {  
            arreglo[j + 1] = arreglo[j];  
            j--;  
        }  
  
        arreglo[j + 1] = aux;  
    }  
}
```

Método de Selección

1. El ordenamiento por Selección funciona seleccionando buscando el menor elemento del vector o matriz.
2. Intercambia con el primero de la lista.
3. A continuación selecciona el siguiente menor y lo pone en la segunda posición de la matriz.


Y en general

- Buscar el mínimo elemento entre una posición i y el final de la lista
- Intercambiar el mínimo con el elemento de la posición i

	8
	5
	2
	6
	9
	3
	1
	4
	0
	7

Implementación de Algoritmo de Selección

```
private static void seleccion() {  
    int[] arreglo = new int[] { 5, 6, 1, 0, 3 };  
  
    int i, j, aux1, aux2;  
  
    // imprime el arreglo al inicio del algoritmo  
    imprimirArreglo(arreglo);  
  
    // recorre el arreglo desordenado  
    for(i = 0; i < arreglo.length - 1; i++)  
    {  
        aux1 = i;  
  
        // encuentra el minimo valor dentro del arreglo desordenado  
        for(j = i + 1; j < arreglo.length; j++)  
        {  
            if(arreglo[j] < arreglo[aux1])  
            {  
                aux1 = j;  
            }  
        }  
    }
```



```
        // intercambia el minimo valor encontrado  
        // con el primer objeto de la lista  
        if(aux1 != i)  
        {  
            aux2 = arreglo[aux1];  
            arreglo[aux1] = arreglo[i];  
            arreglo[i] = aux2;  
        }  
    }
```

Variaciones

- Existen muchos más algoritmos de ordenamiento.
- El por qué de su existencia radica en su capacidad de ordenar listas de valores extensas, así también como el tiempo y el insumo de memoria RAM.
- Muchos de los algoritmos más simples han sido mejorados a fin de mantener su simplicidad, pero a su vez, agregar mayor capacidad de procesamiento o disminución del tiempo de ejecución.
- A continuación un ejemplo de varios tipos.

<https://www.youtube.com/watch?v=kPRA0W1kECg>