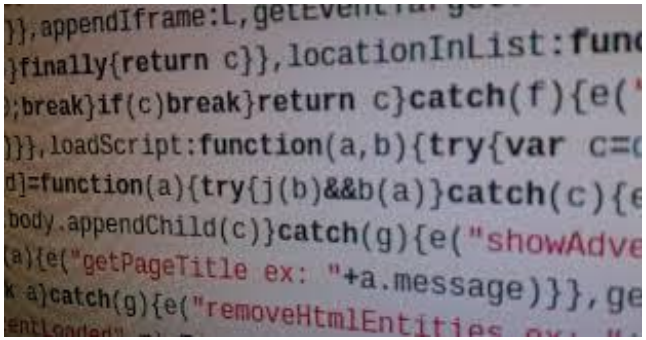


UT 8

OPERACIONES CON ARRAYS.

ORDENACIÓN Y BÚSQUEDA

IES JUAN DE LA CIERVA
DPTO. INFORMÁTICA

CONTENIDOS DE LA PRESENTACIÓN

Objetivos

En esta unidad aprenderás a manejar la búsqueda y ordenación en arrays unidimensionales y multidimensionales, que luego podrás aplicar las técnicas con ficheros.

Contenidos

1. Métodos de búsqueda:
   - ❑ Lineal
   - ❑ Búsqueda binaria (datos ordenados).
2. Métodos de ordenación (Intercambio, Selección e Inserción).
3. Métodos de Inserción (Inserción Directa, Shell y Quick Sort).

MÉTODOS DE BÚSQUEDA

Búsqueda en Arrays

BÚSQUEDA, podemos hablar de dos tipos de búsqueda:

- **Búsqueda lineal**, consiste en recorrer y examinar todos los elementos del array, desde el principio al final. El array no necesita esta ordenado.

- **Búsqueda binaria**, debe estar ordenado el array y vamos preguntando por mitades, eliminando cada vez la mitad del array o subarray analizado.

Arrays: Búsqueda Lineal

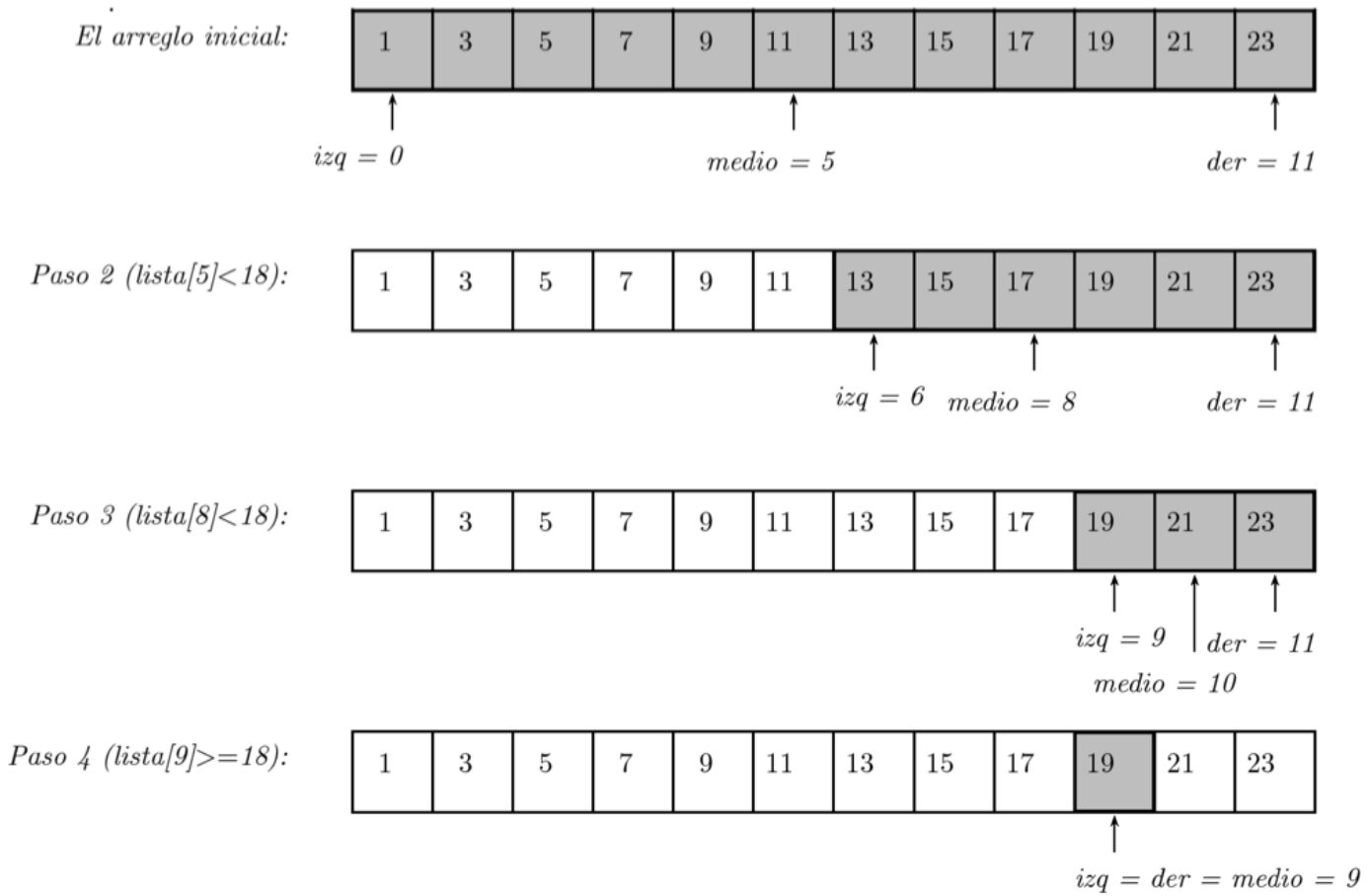
```
public static void main(String[] args) {
    int[] vector= {2,6,35,43,1,0,33,12,98};
    boolean encontrado=false;
    int i=0;     //índice para recorrer el vector;
    int t=33;    //elemento a buscar

    while(encontrado==false && i<vector.length) {
        if(vector[i]==t) encontrado=true;
        else i++;
    }

    if(encontrado) System.out.println("Elemento encontrado"
            + " en la posición "+i);
    else System.out.println("Elemento no encontrado"
            + " en el vector");
}
```

Arrays : Búsqueda Binaria

Buscar el valor 18 en el vector lista que tienes a continuación



Arrays : Búsqueda Binaria

- Funcionamiento-Ejemplo (busca el número 81)
- [1,2,3,.....,48,49,**50**,51,52,.....,98,99,100]
- [51,52,53,.....,73,74,**75**,76,77,.....,98,99,100]
- [76,77,78,.....,85,86,**87**,88,89,.....,98,99,100]
- [76,77,78,79,80,**81**,82,83,84,85,86]

Arrays : Búsqueda Binaria

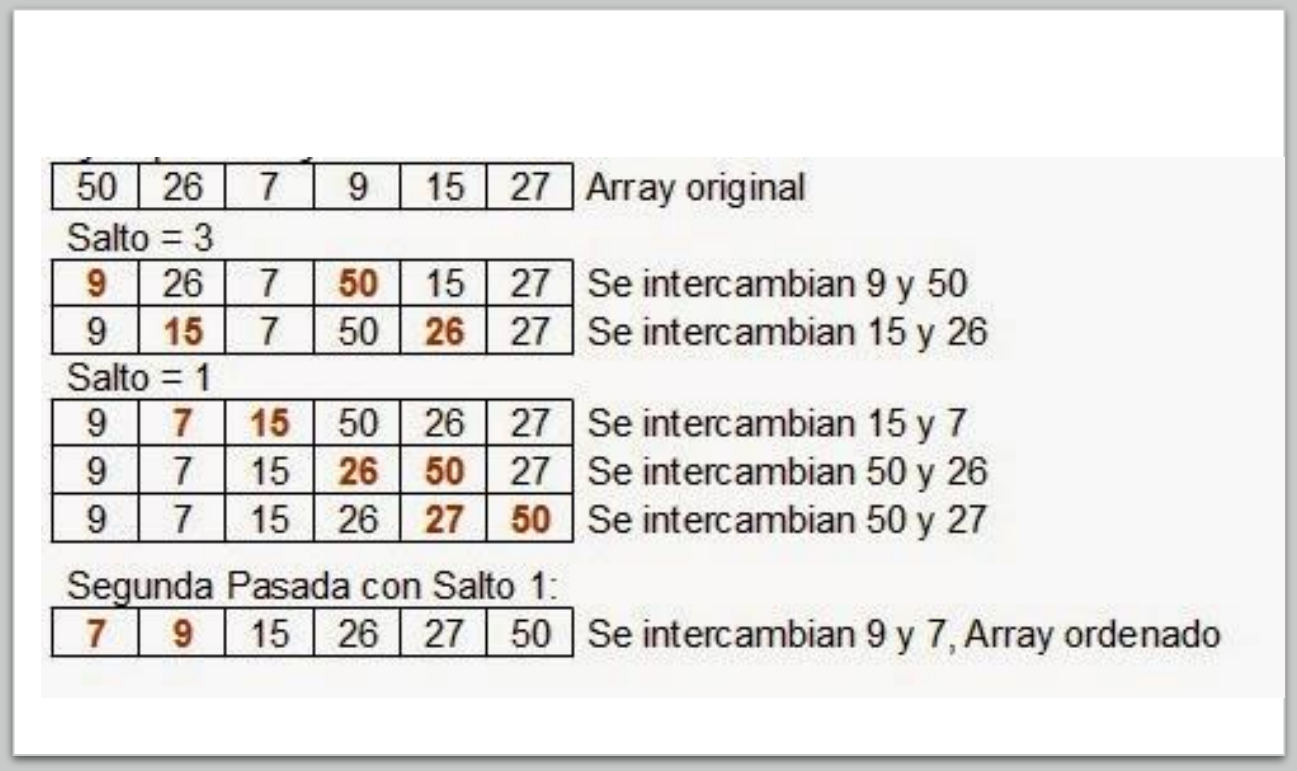
```
int centro,posicion=0, min=0, max=vector.length-1;
boolean encontrado=false;

while(min<=max && !encontrado){
     centro=(max+min)/2;
     if(vector[centro]==datobuscado) {
          encontrado=true;
          posicion=centro;
     }
     else if(dato < vector [centro] ) max=centro-1;
          else min=centro+1;
}

if (encontrado) System.out.println("El dato se ha "
          + "encontrado en la posición"+posicion);
else System.out.println("El elemento no se ha"
          + " encontrado");
```

Arrays : Búsqueda Binaria Recursiva

```
FUNCION BUSCAR( primero, ultimo, tabla, elemento)
    mitad = (primero + ultimo) / 2
    Si tabla[mitad]= elemento
        visualizar "ELEMENTO ENCONTRADO"
    sino
        Si tabla[mitad]> elemento
                ultimo = mitad
        sino
                primero = mitad
        Finsi
        Si  (( primero + ultimo ) / 2) = mitad
           visualizar  " EL ELEMENTO NO SE ENCUENTRA"
        sino
           BUSCAR(primero, ultimo, tabla, elemento)
        Fin_Si
    Fin_Si
FIN_FUNCION
```



MÉTODOS DE ORDENACIÓN

Arrays: Ordenación

Existen distintos tipos de ordenación de array que son:

- **SELECCIÓN.**
- **BURBUJA.**
- **INSERCIÓN.**
- **INSERCIÓN BINARIA.**
- **SHELL.**
- **ORDENACIÓN RÁPIDA Ó QUICK SORT.**

MÉTODO DE INTERCAMBIO o MÉTODO DE LA BURBUJA

50	26	7	9	15	27	Array original

Primera pasada:

26	**50**	7	9	15	27	Se intercambian el 50 y el 26
26	7	**50**	9	15	27	Se intercambian el 50 y el 7
26	7	9	**50**	15	27	Se intercambian el 50 y el 9
26	7	9	15	**50**	27	Se intercambian el 50 y el 15
26	7	9	15	27	**50**	Se intercambian el 50 y el 27

Segunda pasada:

7	**26**	9	15	27	50	Se intercambian el 26 y el 7
7	9	**26**	15	27	50	Se intercambian el 26 y el 9
7	9	15	**26**	27	50	Se intercambian el 26 y el 15

MÉTODO DE INTERCAMBIO o MÉTODO DE LA BURBUJA

Consiste en comparar elementos consecutivos e intercambiarlos si no cumplen el orden deseado.

De esta forma vamos arrastrando un valor hasta el final de la lista.

El proceso se irá repitiendo excluyendo los últimos elementos de la lista que ya están ordenados, hasta que esta se encuentre ordenada en su totalidad.

Es el método más sencillo y conocido, pero también es poco eficiente al realizar demasiadas operaciones, esto hace que sea más lento que otros.

MÉTODO DE INTERCAMBIO o MÉTODO DE LA BURBUJA ORDENACIÓN ASCENDENTE

```
int pasadas=0;
int aux=0;
for(pasadas=1; pasadas<vector.length;pasadas++){
    for(int i=0; i<(vector.length-pasadas);i++){
        if(vector[i]>vector[i+1]){
            aux=vector[i+1];
            vector[i+1]=vector[i];
            vector[i]=aux;
        }
    }
}
```

MÉTODO DE INTERCAMBIO o MÉTODO DE LA BURBUJA ORDENACIÓN DESCENDENTE

```
int aux;
for (int p = 0; p < vector.length; p++) {
    for (int i = 0; i < vector.length-1-p; i++) {
        if(vector[i]<vector[i+1]) {
            aux=vector[i];
            vector[i]=vector[i+1];
            vector[i+1]=aux;
        }
    }
}
```

MÉTODO DE INTERCAMBIO o MÉTODO DE LA BURBUJA MEJORADO

El método de la burbuja puede mejorarse.

Si en la estructura repetitiva interna no se produce ningún cambio, nuestro proceso debe detenerse al estar la lista ya ordenada.

De este modo no se realizarán más intercambios de los necesarios.

MÉTODO DE INTERCAMBIO o MÉTODO DE LA BURBUJA

```
int pasadas=0;
int aux=0;
boolean sw=true;
for(pasadas=1;pasadas<vector.length && sw==true;pasadas++){
    sw=false;
    for(int i=0;i<(vector.length-pasadas);i++){
        if(vector[i]>vector[i+1]){
            sw=true;
            aux = vector[i+1];
            vector[i+1] = vector[i];
            vector[i] = aux;
        }
    }
}
```

50	26	7	9	15	27	Array original
Salto = 3						
9	26	7	**50**	15	27	Se intercambian 9 y 50
9	**15**	7	50	**26**	27	Se intercambian 15 y 26
Salto = 1						
9	**7**	**15**	50	26	27	Se intercambian 15 y 7
9	7	15	**26**	**50**	27	Se intercambian 50 y 26
9	7	15	26	**27**	**50**	Se intercambian 50 y 27
Segunda Pasada con Salto 1:						
7	**9**	15	26	27	50	Se intercambian 9 y 7, Array ordenado

INSERCIÓN DE DATOS

INSERCIÓN EN VECTOR ORDENADO REEMPLAZANDO EL DATO EXISTENTE

```
//dato: valor a insertar
int i=0;
if (vector[i]> dato) vector[i]=num2;
else {
    while(vector[i]<num2 && i<vector.length-1){
        i++;
        if(vector[i]>num2){
            vector[i]=num2;

        }
    }
}
```

INSERCIÓN EN VECTOR ORDENADO DESPLAZANDO LOS ELEMENTOS SIGUIENTES

Ejercicio:

Modifica el ejemplo anterior para insertar el elemento conservando los datos desplazándolos hacia el final del vector.

El último dato se pierde.