UD5.2- ALGORITMES D'ORDENACIÓ I CERCA

Programació – **1er DAW/DAM**

GENERALITAT VALENCIANA
Conselleria d'Educació, Cultura i Esport

CPFPBATOI

C/ Societat Unió Musical, 8 - 03802 Alcoi
Tel. 966 52 76 60 - Fax 966 52 76 61
03012165.secret@gva.es
www.cipfpbatoi.es

GOBIERNO DE ESPAÑA
MINISTERIO DE EDUCACIÓN Y FORMACIÓN PROFESIONAL

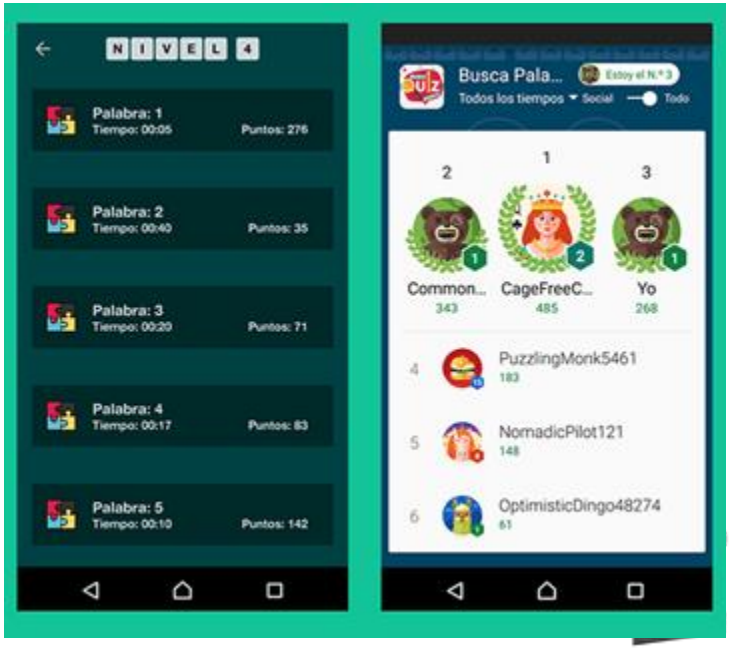
UNIÓ EUROPEA

0. CONTINGUTS

- INTRODUCCIÓ
- TIPUS D'ALGORITMES D'ORDENACIÓ
  - INTERCANVI
  - SELECCIÓ
- CERCA D'ELEMENTS A ARRAYS
- LA CLASSE `java.utils.Arrays`

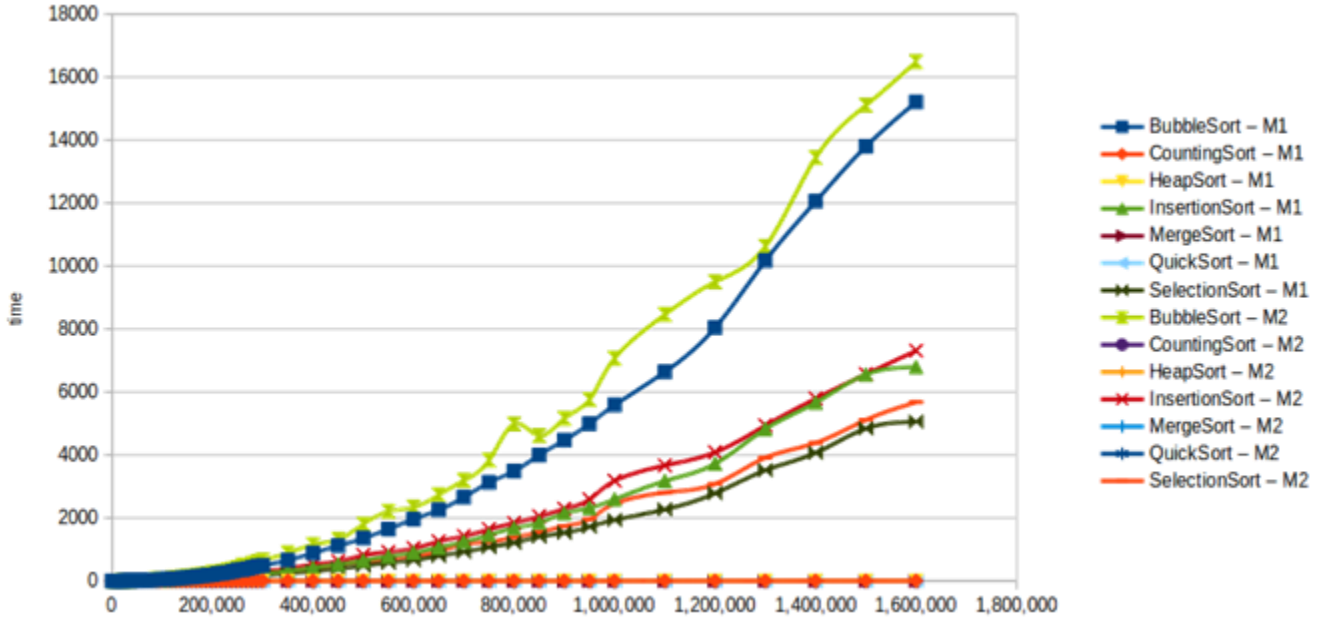
1. INTRODUCCIÓ

- A moltes de les activitats que es realitzen diàriament la informació tractada mitjançant estructures de dades **ha d'estar ordenada.**
  - Rànquing d'un joc **(puntuació aconseguida)**
  - Agendes i números de telèfon **(Identificador del contacte)**
  - Llistats d'alumnes **(Cognoms, ...)**



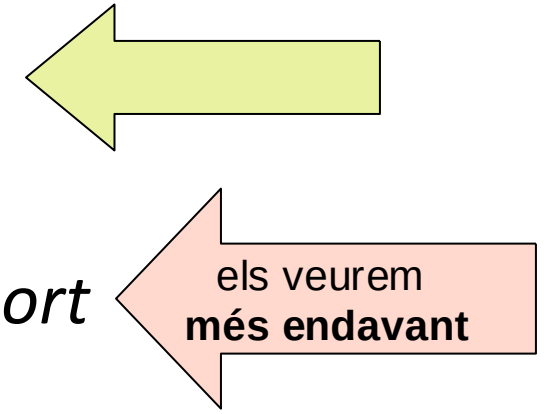
1. INTRODUCCIÓ

- Per tal assolir la **màxima eficiència**, S'han desenvolupat **una sèrie d'algorismes**, Amb diferents nivells de dificultat, que permeten **l'ordenació d'elements** dins d'una **estructura de dades**.
  - Una mesura de l'eficiència **d'un algorisme** és el nombre de vegades que fa una acció (p.e una comparació).



2. ALGORITMES D'ORDENACIÓ. TIPUS

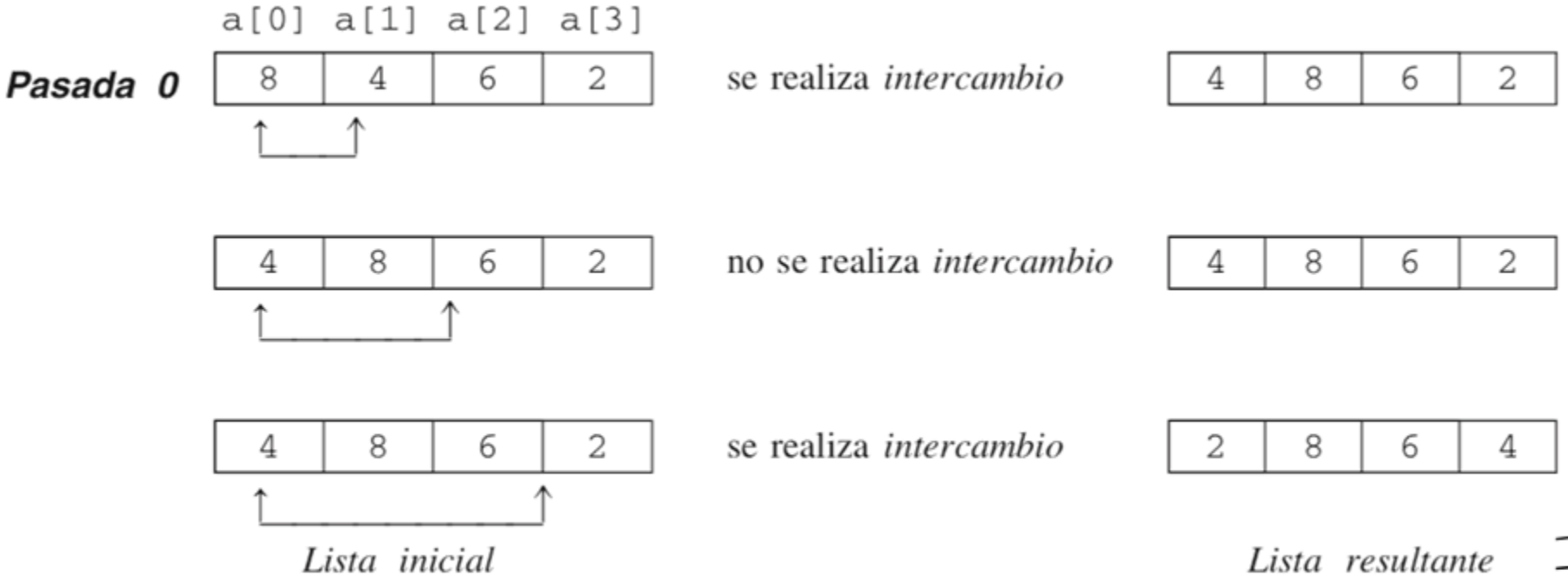
- Podem classificar els algorismes de cerca en 2 grans grups:
  - **Directes o bàsics:**
    - *Selecció, intercanvi, inserció*
  - **Indirectes o avançats**
    - *QuickSort, mergesort, Heapsort*



En el cas de llistats petits, amb els algoritmes directes serà suficient, en llistes grans aquests algoritmes són massa ineficients

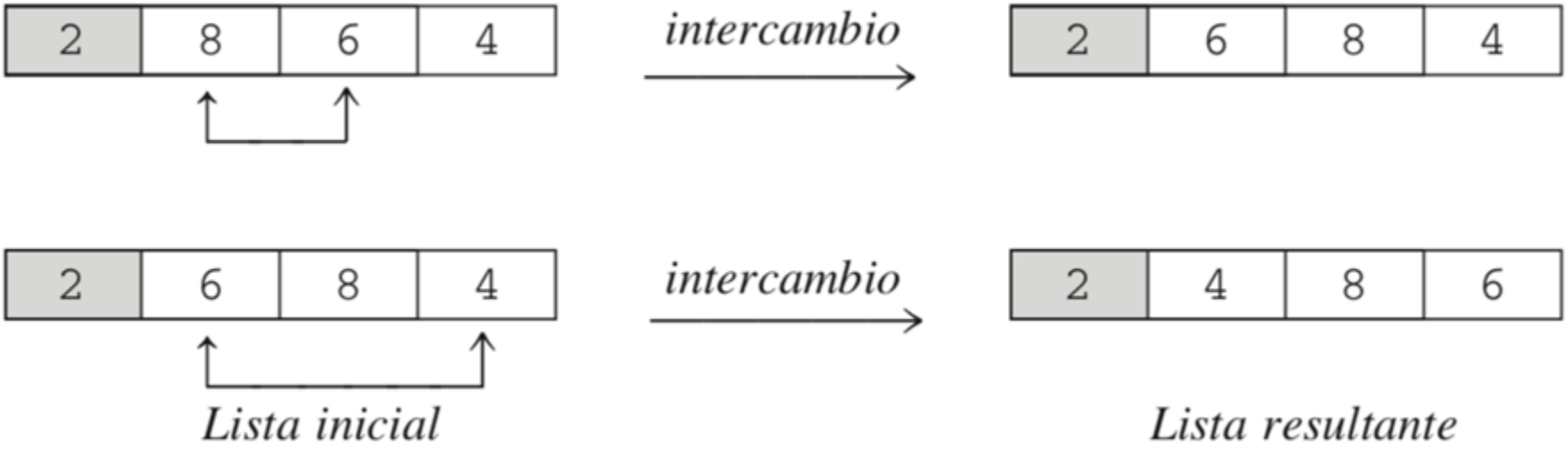
2.1 ORDENACIÓ PER INTERCANVI I

- Es tracta de l'algorisme més senzill (i menys recomanable).
- Aquest **algoritme** es fonamenta en **llegir de manera successiva** i **iterativa** una llista d'elements i anar **intercanviant-los** perquè ocupen la posició.



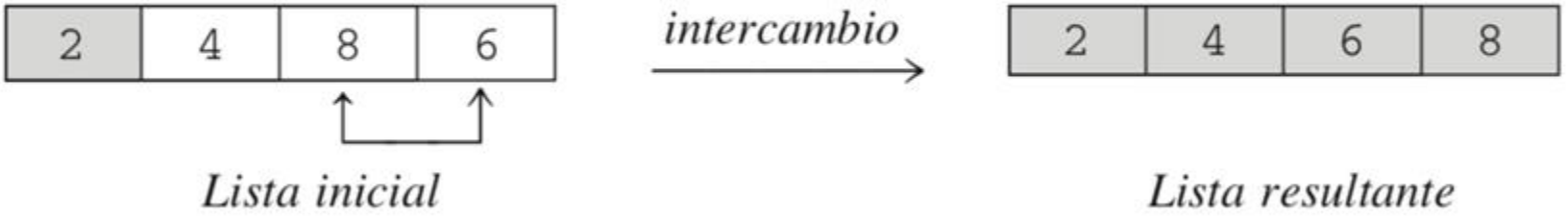
2.1 ORDENACIÓ PER INTERCANVI II

- A la **primera passada** tindrem el **primer element** de l'array ordenat.
  - En aquesta passada considerarem només la sublista **{8, 6, 4}**
  - Com podem veure requereix de 2 intercanvis



2.1 ORDENACIÓ PER INTERCANVI III

- A la tercera passada, els elements a considerar són **{8,6}.**
  - Com podem veure haurem d'intercanviar els elements 8 i 6 de manera que el nostre llistat quedarà ordenat



2.1 ORDENACIÓ PER INTERCANVI IV

Algorisme

1. A la primera iteració considerem tots els elements de la llista **A[0]**...**A[N]**.
2. Seleccionem l'element que actuarà **com a pivot** que és el primer element de la llista A[**A0**]
3. Busquem a la sublista **A[1]** ... **A[n-1] l'element més petit** i ho **intercanviem** amb el primer element **A[0]** (ara l'entrada més petita és a la primera posició del vector)**.**
4. Repetim el pas 2 i 3 però considerant la subllista **A[1]**...**A[N-1]**

2.1 ORDENACIÓ PER INTERCANVI V

Algorisme

```
PER i = 0 FINS i < (mida(vector)-1) AMB PAS i++ FER
     PER j = i + 1 FINS j < (mida(vector)) AMB PAS j++ FER
         SI (vector[j] < vector[i]) LLAVORS
             aux = vector[i]
             vector[i] = vector[j]
             vector[j] = aux
         FIN_SI
    FIN_PER
FIN_PER
```

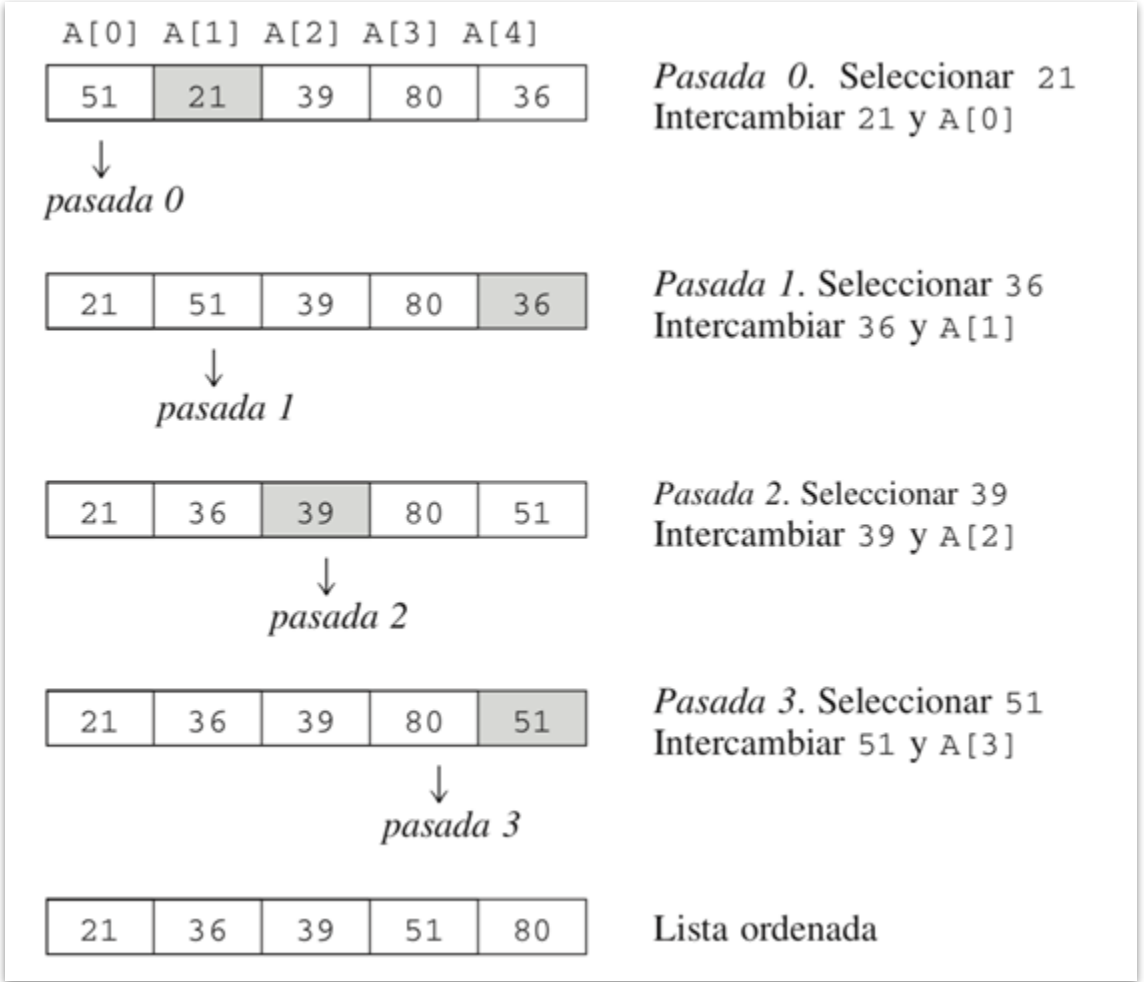
2.1.1 ACTIVITAT PRÈVIA

- **Activitat 10.-** Implementa un mètode `ordenarPerIntercanvi` que reba com a paràmetre un array de sencers i l'ordene.

  Des del mètode main() genera 2 arrays amb 10 números aleatoris, mostra'ls sense ordenar i posteriorment mostra'ls ordenats fent ús del mètode `ordenarPerIntercanvi`

2.2 ORDENACIÓ SELECCIÓ

- Aquest algorisme es basa a dur a terme diferents passades, de manera que, a cadascuna, seleccionarem l'element més petit i posarem a la posició **a[X]** on **x** és el número d'iteració.



2.2 ORDENACIÓ SELECCIÓ

Algorisme

1. Seleccionar l'element **més petit** de la **llista a**; e **intercanviar** amb el primer element a**[0]**. Ara l'entrada més petita és a la primera posició del vector.
2. **Considera** les **posicions** de la **sub-llista a[1], a[2], a[3],...a[n-1]**
3. **Selecciona** l'element **més petit** e **intercanvia'l** pel primer element de la **sub-llista.** Ara les dues primeres entrades estan en ordre.
4. Repetir el **pas 3,** però ara considerant la sub-llista **a[2] ... a[n-1],** després a**[3]...a[n-1]** i així successivament fins a arribar al final.

2.2 ORDENACIÓ SELECCIÓ

Algorisme

```
PER i = 0 FINS i < (mida(vector)-1) AMB CADA PAS i++ FER
   indiceElementMenor = i;
   PER j = i + 1 FINS j < mida(vector) AMB CADA PAS j++ FER
         SI (vector[j] < vector[indiceElementoMenor]) LLAVORS
             indiceElementMenor = j
         FIN_SI
   FIN_PER
   SI (indiceElementoMenor != i) LLAVORS
         aux = vector[indiceElementoMenor]
         vector[indiceElementoMenor] = vector[i]
         vector[i] = aux
   FIN_SI
FIN_PER
```

2.2.1 ACTIVITAT PRÈVIA

- **Activitat 11.-** Implementa un mètode `ordenarPerSeleccio` que reba com a paràmetre un array de `String` i ordene aquest array.
- *Des del mètode `main()` genera 1* ***array*** *amb els noms de tots els companys del grup, mostra'ls sense ordenar i posteriorment mostra'ls ordenats fent ús del mètode `ordenarPerSeleccio`*

Nota: Recorda que amb els tipus String no podem fer ús dels operadors de comparació **==, != , <, >** ja que estaríem comparant les referències i no la cadena que conté

3. CERCA D'ELEMENTS EN UN ARRAY

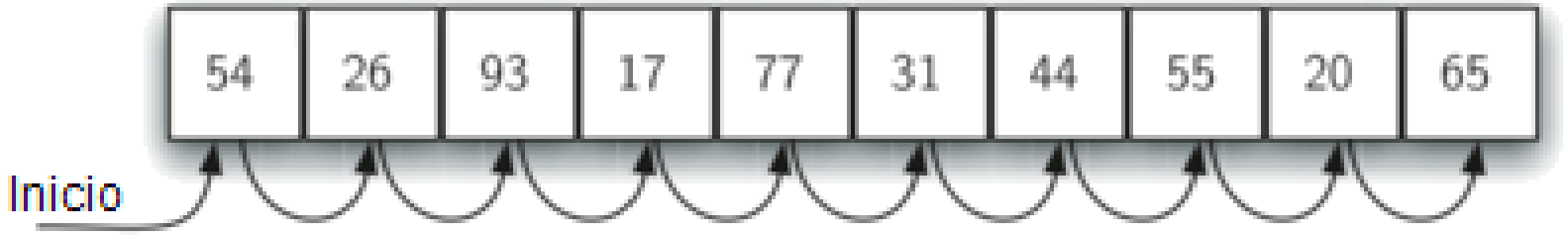
Cerca seqüencial o lineal

- Els **elements** de la llista **exploren un darrere l'altre** comparant-los amb l'element cercat o **element clau.**

  Ex. *Cerca de la persona a qui correspon un número de telèfon a l'agenda del mòbil.*

- En el cas **mitjà** haurà d'explorar, **la meitat** dels elements per trobar-lo.

- **No cal** que el llistat estigui **ordenat**



*Funciona bé amb **arrays petits***

3. CERCA D'ELEMENTS EN UN ARRAY

- La **cerca d'un element** específic sobre un conjunt d'elements, és una tasca molt comuna a qualsevol programa.
- Quan es treballa amb grans quantitats de dades, es necessiten **d'algorismes eficients**.
- Podem trobar dues tècniques fonamentals:
  - Cerca **seqüencial**:
  - Cerca **binària** o **dicotòmica**:

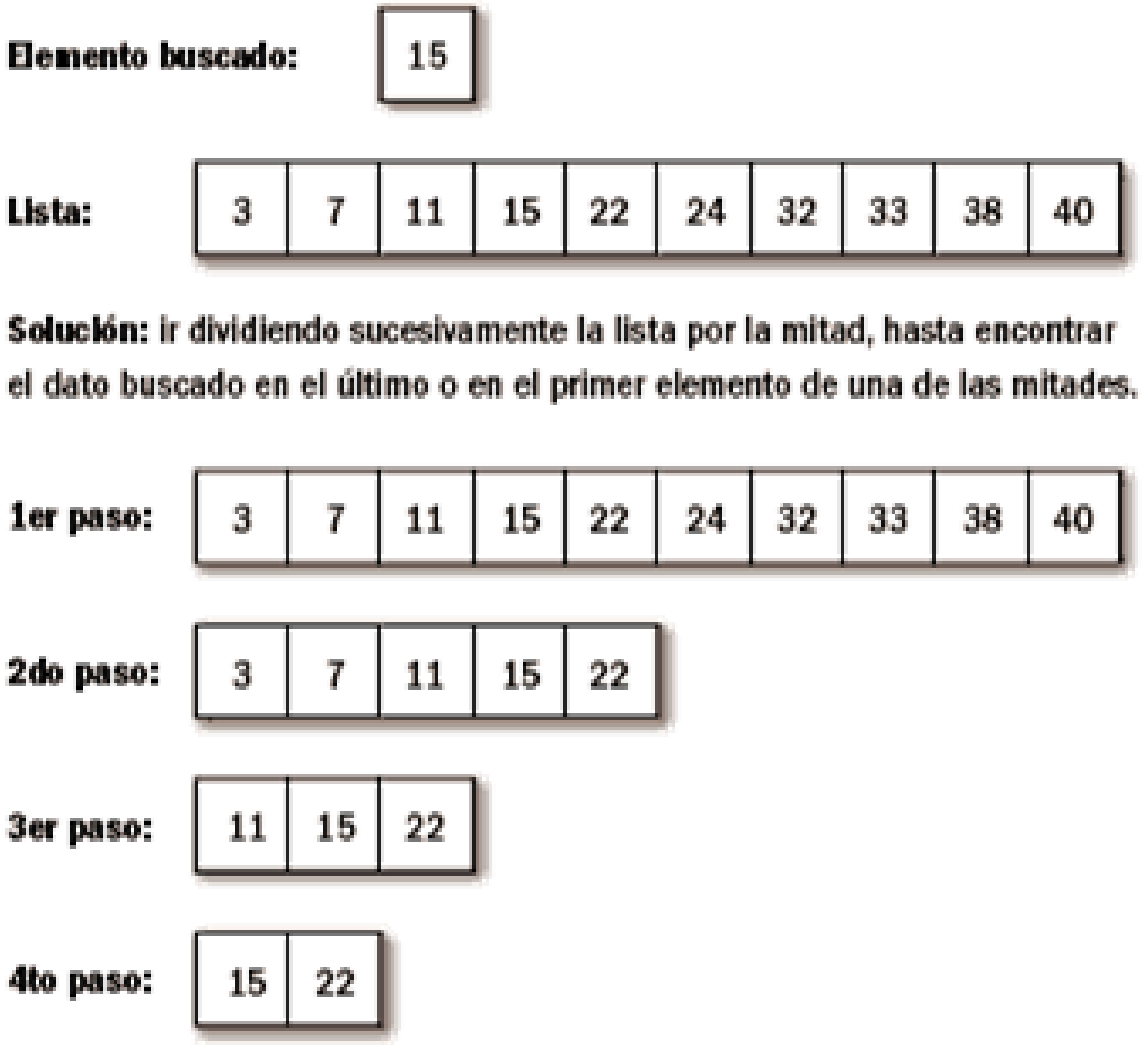
3. CERCA D'ELEMENTS EN UN ARRAY

Cerca binària o dicotòmica

- Aquest tipus de cerca, necessita que els **elements** de l'array estiguen *situats* atenent a una **relació d'ordre**. *(p.e. de menor a major).*
- *La* estratègia consisteix en **seleccionar l'element central** de la llista, si aquest no coincideix se seguirà buscant, repetint el algoritme, a la **sub-llista esquerra** (si és més **petit**) o a la **sub-llista dreta** (si és més **gran**)
- Es tracta d'un mètode **eficient** i àmpliament utilitzat en **programació**
  - **Sistema de fitxers, cerca d'una paraula en un diccionari, etc. . .**

3. CERCA D'ELEMENTS EN UN ARRAY

- **Cerca binària o dicotòmica**



ACTIVITAT PRÈVIA

Activitat 12.- Implementa els mètodes de cerca de vectors següents:

```
public static int cercarEnVectorNoOrdenat(int[] numeros, int numBuscar)

public static int cercarEnVectorOrdenat(int[] numeros, int numBuscar)
```

*Des del mètode `main()` Genera 2 arrays amb **10 números aleatoris**. Un ordenat i un altre no. demana un número a l'usuari per teclat i busca el número a l'array utilitzant els dos mètodes de cerca. Si no trobem l'element a l'array, es tornarà un -1,*

3. LA CLASSE `java.utils.Arrays`

Ja hem vist un mètode d'aquesta classe anomenat `toString()` (*retorna el contigut d' un array en format* String).

Ordenació d'arrays

➔ Aquesta classe estàtica també ens proporciona nombroses implementacions per a la **ordenació d'arrays**.

```
public static void sort(int[] a)
public static void sort(float[] a)
public static void sort(double[] a)
....
```

Utilitza l'algoritme **quicksort**

3. LA CLASSE `java.utils.Arrays`

Cerca d'elements en un array

- També ens proporciona diferents **mètodes sobrecarregats,** per fer cerques d'elements en un array.

```
binarySearch(char[] a, char key)
binarySearch(byte[] a, byte key)
binarySearch(float[] a, float key)
```

3. LA CLASSE java.utils.Arrays

- **Cerca d'elements en un array**

```
public class Examen2 {
    public static void main(String[] args) {
     String[] alumnes = {"Pepe", "Juan", "Luis", "Ana", "Luisa", "Carmen"};
     System.out.println(Arrays.toString(alumnes));
     Arrays.sort(alumnes);
     System.out.println(Arrays.toString(alumnes));
     int indiceAlumno = Arrays.binarySearch(alumnes, "Pepe");
     System.out.printf("\"Pepe\" se encuentra en la posicion %d que corresponde al
índice %d %n", indiceAlumno + 1, indiceAlumno);
   }
}
```

```
[Pepe, Juan, Luis, Ana, Luisa, Carmen]
[Ana, Carmen, Juan, Luis, Luisa, Pepe]
"Pepe" se encuentra en la posicion 6 que corresponde al
índice 5
```

3. LA CLASSE `java.utils.Arrays`

Això és tot... de moment :-)