Programació

UT3.2 Copiar, modificar, buscar i ordenar

Assignació vs còpia

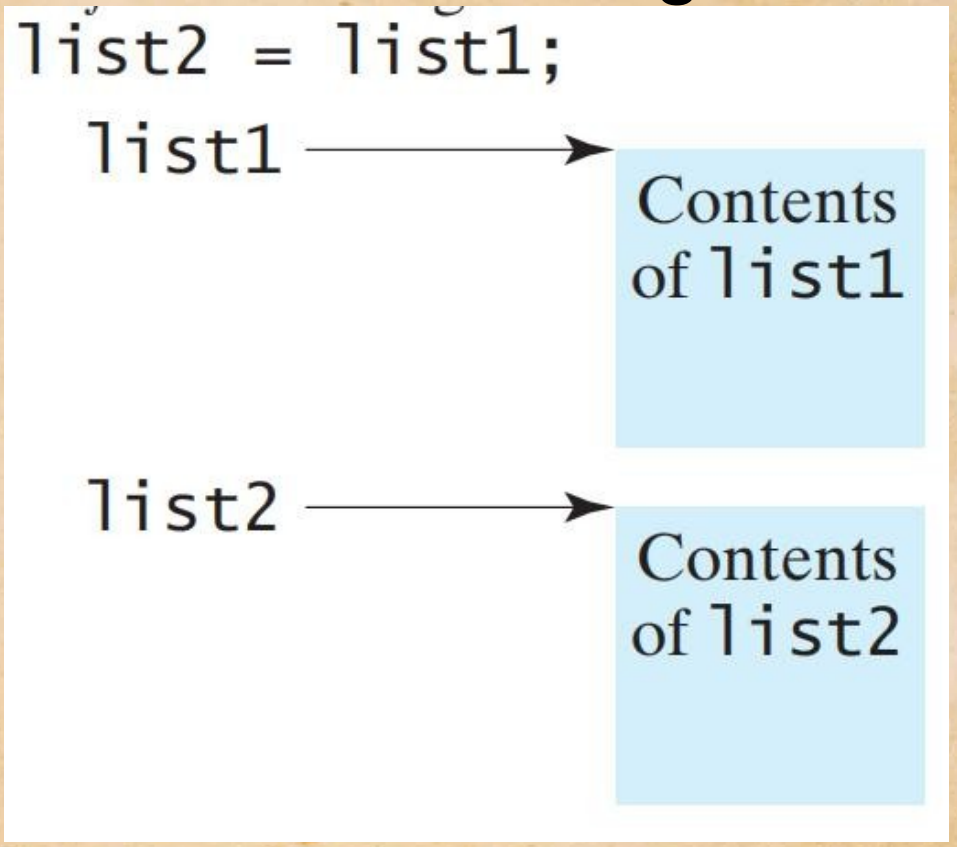
- **Podem arribar a pensar que l'assignació copiarà un array a un altre. Però en el cas dels arrays, no es realitza la còpia del contingut de l'array en un altre, sinó que duplica la referència a l'array original.**

```
...
  //Assignació amb arrays:
  int[] arrayA = {10, 20, 30, 40, 50};
  int[] arrayB = {60, 70, 80, 90, 100};
  //A la variable "arrayA" se li assigna el valor d'"arrayB".
  arrayA = arrayB;
  //Es modifica el valor de l'índex 2 només d'"arrayB".
  arrayB[2] = 60;
  //"arrayA" també ha vist modificat el seu valor a l'índex 2!
  System.out.println(arrayA[2]);
  //Es modifica el valor de l'índex 4 només d'"arrayA".
  arrayA[4] = 70;
  //"arrayB" també ha vist modificat el seu valor a l'índex 4!
  System.out.println(arrayB[4]);
  //De fet, "a" i "b" accedeixen exactament a les mateixes dades.
  ...
```

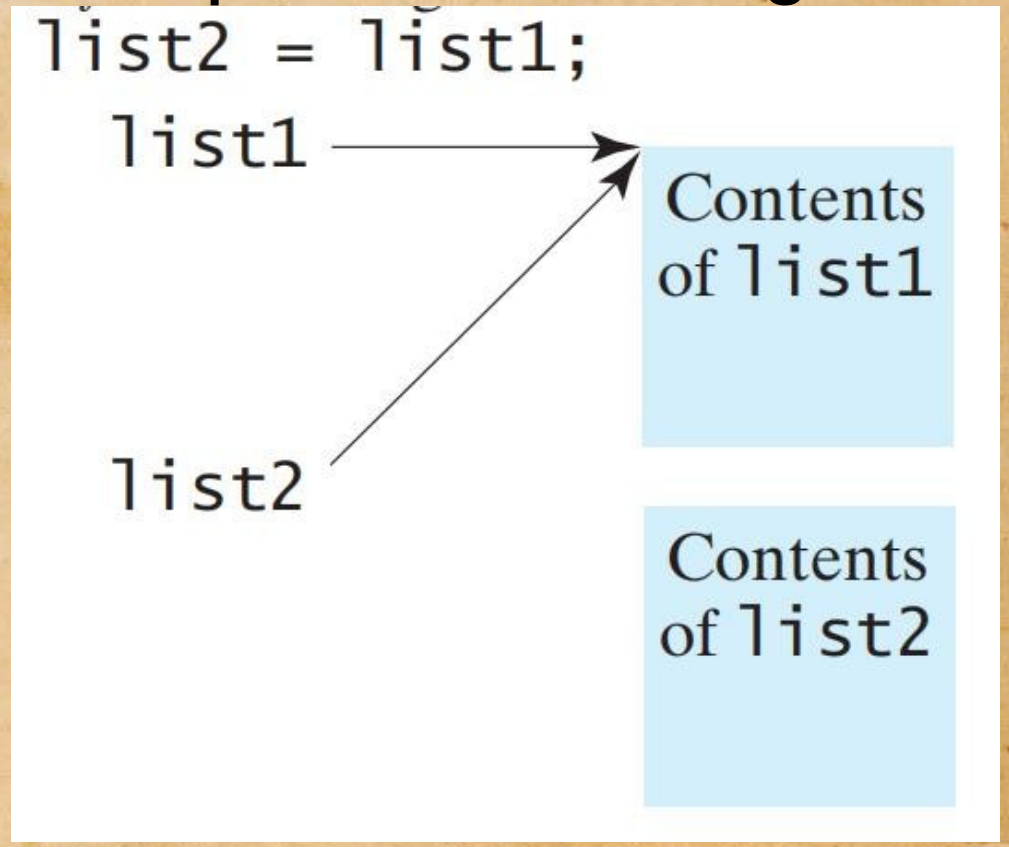
¿I què passa amb les dades inicials de l'arrayB?

Assignació (modificació de referències)

Abans de l'assignació



Després de l'assignació



Duplicar arrays

- Per a duplicar un array, cal copiar-lo **valor a valor** a un array nou, per exemple, amb un bucle for.

Amb "duplicar arrays" ens referim a tindre DOS arrays independents que contenen els mateixos valors

Pràctica 3.6

- Realitza un programa que copie un array de 10 nombres reals anomenat arrayA en un altre anomenat arrayB i els imprimisca per pantalla.

Canviar la longitud d’un array

- En este punt cal tindre clar que la longitud de l'array, una volta declarat, **NO ES POT CANVIAR**.
- Davant d’este problema, existeixen dos solucions:

  1) Copiar les dades de l'array ple a un altre més gran.

  2) Definir un array amb grandària suficient com per no arribar a omplir-lo mai.

Recerca en arrays

- La recerca és el procés de buscar un element específic en un array. Per exemple, buscar si una determinada qualificació està en una llista de notes.
- La recerca és una tasca comuna a la programació.
- Molts algoritmes i estructures realitzen operacions de recerca.
- Els dos enfocaments més comunament utilitzats són la **recerca lineal** i la **recerca binària**.

Recerca lineal

L'enfocament de recerca lineal compara la dada a buscar amb cada element de la matriu. Continua fent-ho fins que la dada coincidisca amb un element de l'array o bé l'array s'esgote sense trobar cap coincidència.

Si s'obté una **coincidència**, la recerca lineal **retorna l'índex de l'element de l'array** que coincidisca amb la dada a buscar (clau).

Si no es troba **cap coincidència**, la recerca retorna **-1**.

Recerca lineal

ATENCIÓ: No cal memoritzar el codi, només comprendre el seu funcionament

```java
int[] llista = {6,7,2,1,33,2};
int clau = 3;

boolean trobat = false;
for(int i = 0; i < llista.length && !trobat; i++){
    if(clau == llista[i]){
        System.out.println(i);
        trobat = true;
    }
}

if(!trobat){
    System.out.println(-1);
}
```

Recerca binària

La recerca binària és l'altre enfocament de recerca comuna per una llista de valors.

Per a que la recerca binària funcione, **els elements de l'array han d'estar ordenats (sorted)**.

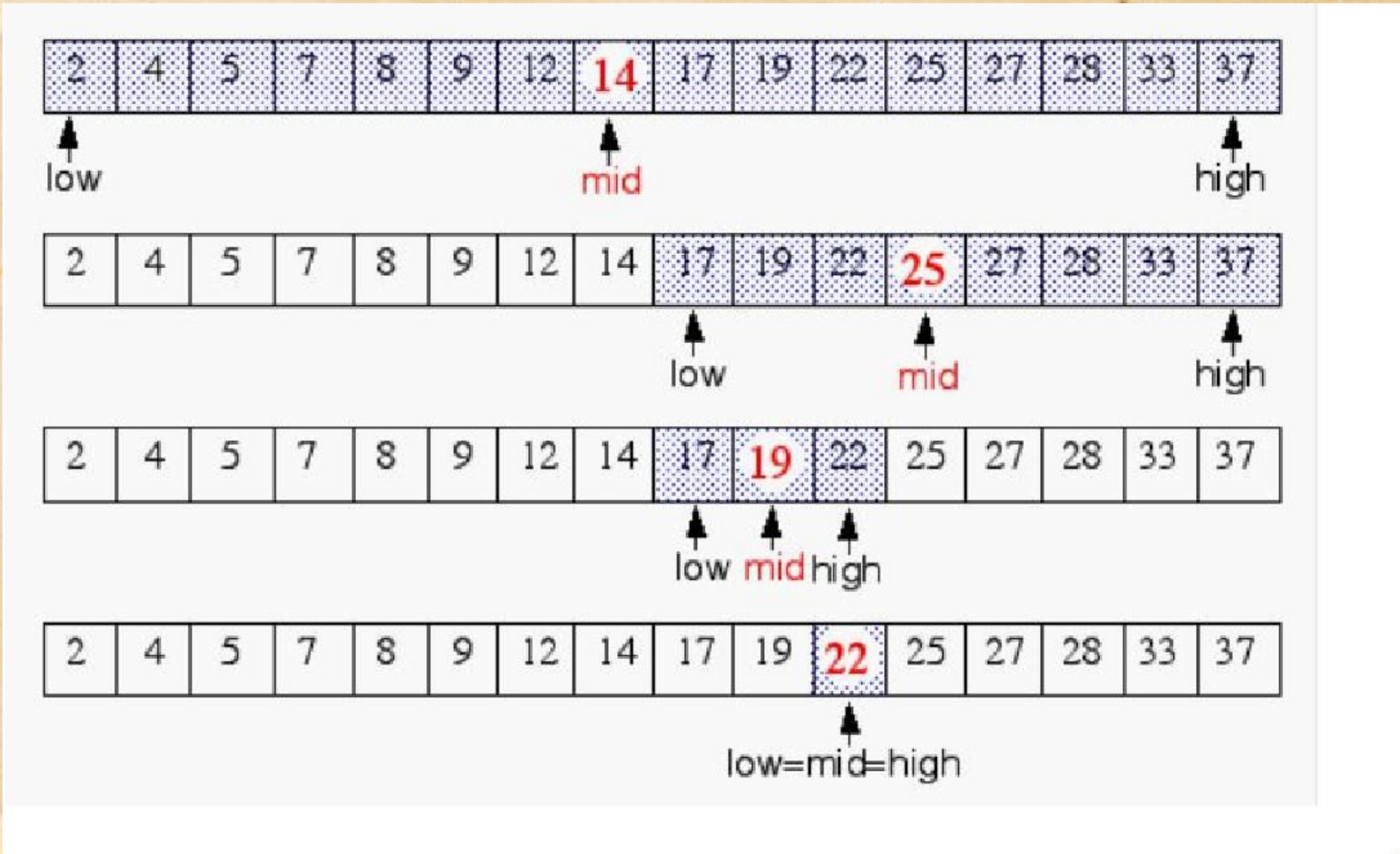
Suposem que l'array està en ordre ascendent. La recerca binària primer compara la clau amb l'element central de l'array.

Recerca binària

Considerem els següents tres casos:

- **Si la clau és menor que l'element central**, ha de continuar buscant la clau només a la primera meitat de l'array.
- **Si la clau és igual que l'element central**, la recerca acaba amb una coincidència.
- **Si la clau és major que l'element central**, ha de continuar buscant la clau només a la segona meitat de l'array.

Recerca binària



Cerca binària

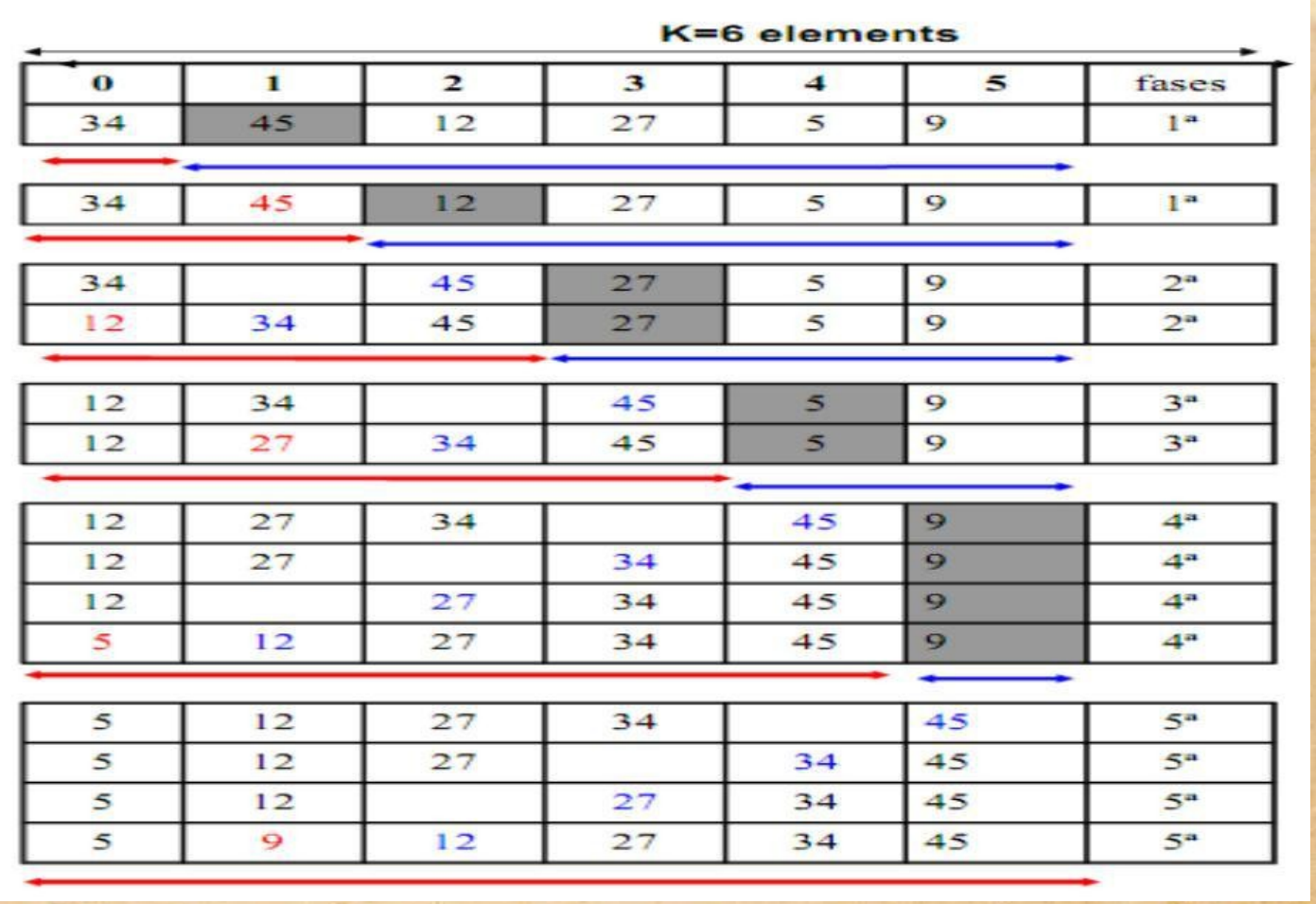
```java
int[] llista = {2,4,5,7,8,9,12,14,17,19,22,25,27,28,33,37};
int clau = 22;

boolean trobat = false;
int menor = 0;
int central = 0;
int major = llista.length - 1;
while(major >= menor && !trobat){
    central = (menor + major) / 2;
    if(clau < llista[central]){
        major = central - 1;
    }
    else if(clau == llista[central]){
        System.out.println(central);
        trobat = true;
    }
    else{
        menor = central + 1;
    }
}

if(!trobat){
    System.out.println(-1);
}
```

ATENCIÓ: No cal memoritzar el codi, només comprendre el seu funcionament

Ordenació d'arrays

- **Existeixen diferents algoritmes d'ordenació. Els més habituals són:**
  - **Inserció directa**
  - **Selecció directa**
  - **Intercanvi directe (bombolla)**

Algorisme d'inserció directa



0	1	2	3	4	5	fases
34	45	12	27	5	9	1ª
34	45	12	27	5	9	1ª
34		45	27	5	9	2ª
12	34	45	27	5	9	2ª
12	34		45	5	9	3ª
12	27	34	45	5	9	3ª
12	27	34		45	9	4ª
12	27		34	45	9	4ª
12		27	34	45	9	4ª
5	12	27	34	45	9	4ª
5	12	27	34		45	5ª
5	12	27		34	45	5ª
5	12		27	34	45	5ª
5	9	12	27	34	45	5ª

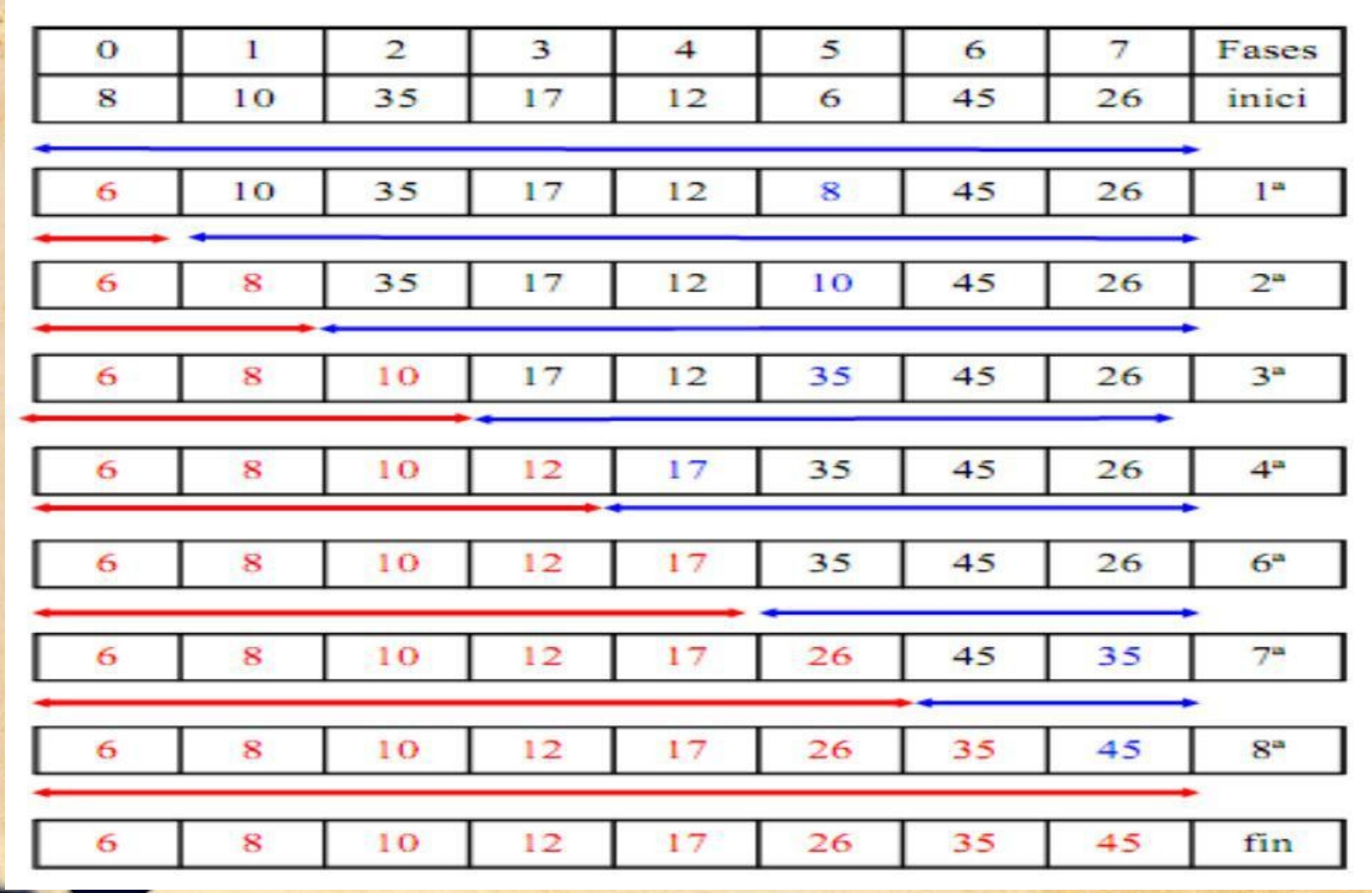
https://es.wikipedia.org/wiki/Ordenamiento_por_inserci%C3%B3n

Algorisme d'inserció directa

```
int j;
int clave;
for (int i = 1; i < lista.length; i++) {

    clave = lista[i]; //clave a reubicar
    j = i - 1;

    while (j >= 0 && lista[j] > clave ) {
        lista[j+1] = lista[j]; // intercambio de claves
        j = j - 1;
    }

    lista[j+1] = clave; //inserta elemento actual
}
```

ATENCIÓ: No cal memoritzar el codi, només comprendre el seu funcionament

Algorisme de selecció directa



0	1	2	3	4	5	6	7	Fases
8	10	35	17	12	6	45	26	inici
6	10	35	17	12	8	45	26	1ª
6	8	35	17	12	10	45	26	2ª
6	8	10	17	12	35	45	26	3ª
6	8	10	12	17	35	45	26	4ª
6	8	10	12	17	35	45	26	6ª
6	8	10	12	17	26	45	35	7ª
6	8	10	12	17	26	35	45	8ª
6	8	10	12	17	26	35	45	fin

https://es.wikipedia.org/wiki/Ordenamiento_por_selecci%C3%B3n

Algorisme de selecció directa

```java
for (int i = 0; i < lista.length - 1; i++) {
    // Encuentra el valor mínimo de la lista
    int valorMinimoActual = lista[i];
    int indiceValorMinimoActual = i;

    for (int j = i + 1; j < lista.length; j++) {
        if (valorMinimoActual > lista[j]) {
            valorMinimoActual = lista[j];
            indiceValorMinimoActual = j;
        }
    }

    // Intercambia lista[i] con lista[indiceValorMinimoActual] si procede
    if (indiceValorMinimoActual != i) {
        lista[indiceValorMinimoActual] = lista[i];
        lista[i] = valorMinimoActual;
    }
}
```

ATENCIÓ: No cal memoritzar el codi, només comprendre el seu funcionament

Algorisme d'intercanvi directe (algoritme de la bombolla)

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

https://es.wikipedia.org/wiki/Ordenamiento_de_burbuja

Algorisme d'intercanvi directe

```
for (int i = 1; i < lista.length; i++) {

    for (int j = 0; j < lista.length - 1; j++) {

        if (lista[j] > lista[j+1]) {
            int aux = lista[j];
            lista[j] = lista[j+1];
            lista[j+1] = aux;
        }
    }
}
```

ATENCIÓ: No cal memoritzar el codi, només comprendre el seu funcionament