21779. Algorísmia i Estructures de Dades II
Curs 2023-24
Ordenant elements: *Selection sort*

22 de febrer de 2024

Imagineu que voleu ordenar el següent array d'elements:

1	30	41	28	5

Com ho faríeu (mentalment)?

Selection sort

L'algorisme d'ordenació *selection sort* ordena un array trobant repetidament l'element mínim (si volem ordenar en ordre ascendent) de la part no ordenada i posant-lo al principi. Per això, l'algorisme gestiona l'array a ordenar dividit en dues parts:

- La part de l'array que ja es troba ordenada.
- La part restant de l'array (desordenada).

I, en cada iteració, l'element mínim de la part de l'array desordenada es selecciona i es mou al principi de la part desordenada (formant part de la part ordenada):

```
int[] array = {1, 30, 41, 28, 5};
// Selecciona l'element minim de array[0, 4]
// i el posa al principi
1, 30, 41, 28, 5
// Selecciona l'element minim de array[1, 4]
// i el posa al principi de array[1, 4]
// (fent un swap entre els dos elements)
1, 5, 41, 28, 30
// Selecciona l'element minim de array[2, 4]
// i el posa al principi de array[2, 4]
// (fent un swap entre els dos elements)
1, 5, 28, 41, 30
// Selecciona l'element minim de array[3, 4]
// i el posa al principi de array[3, 4]
// (fent un swap entre els dos elements)
1, 5, 28, 30, 41
```

En aquesta sessió pràctica:

1. Implementarem l'algorisme d'ordenació *selection sort* en Java.
2. Analitzarem el seu cost computacional des del punt de vista del temps d'execució.
3. Integrarem el mètode de l'algorisme d'ordenació *selection sort* en una aplicació Android simple.

1 Implementació de l'algorisme *selection sort*

1. Creau un projecte Netbeans anomenat *EfficientSorting* amb la seva corresponent classe principal.

2. Dins la classe principal, implementeu els següents mètodes:

```
public static void printArray(int[] arr)
```

   S'encarrega d'imprimir per consola tots els elements d'un array d'enters.

```
public static int[] generateRandomArray(int n)
```

   S'encarrega de generar un array d'enters de longitud n emplenat amb nombres aleatoris.

   - **NOTA**: Per tal de facilitar posteriors comparacions, assegureu-vos d'inicialitzar el vostre generador de nombres aleatoris amb la mateixa llavor per poder obtenir arrays amb els mateixos elements.

3. Creeu una nova classe anomenada *Sorting*. Dins aquesta classe heu d'implementar el mètode:

```
public static void selectionSort(int[] arr)
```

   S'encarrega d'ordenar l'array d'enters *arr* utilitzant l'algorisme *selection sort*.

4. Dins la vostra classe principal:

   (a) Creeu un array de nombres aleatoris de longitud 10.

   (b) Imprimiu el contingut del vostre array sense ordenar.

   (c) Ordeneu el vostre array utilitzant l'algorisme *selection sort*.

   (d) Imprimiu el vostre array ordenat per assegurar-vos que l'algorisme funciona correctament.

2 Què ens costa utilitzar l'algorisme *selection sort*?

Una vegada implementat l'algorisme, analitzarem el seu cost computacional des del punt de vista del temps d'execució.

Per això, generarem arrays aleatoris de diferents longituds i mesurarem el temps necessari (en mil·lisegons) per a realitzar-ne l'ordenació, emplenant la Taula 1:

n	**10**	**100**	**1000**	**10000**	**100000**	**200000**	**300000**	**500000**	**600000**
Temps (ms)									

Taula 1: Temps d'execució de l'algorisme *selection sort*.

NOTA: Per calcular el temps d'execució podeu utilitzar la funció *System.currentTimeMillis()*, la qual retorna l'hora actual del sistema en mil·lisegons.

Una vegada realitzades totes les mesures, observarem les dades a través d'un gràfic on l'eix horitzontal serà la longitud de l'array (n) i l'eix vertical el temps (en mil·lisegons) necessari per a ordenar-lo (veure Figura 1).

NOTA: A Aula digital podeu trobar un excel que us pot servir de plantilla per a generar el gràfic del temps d'execució.

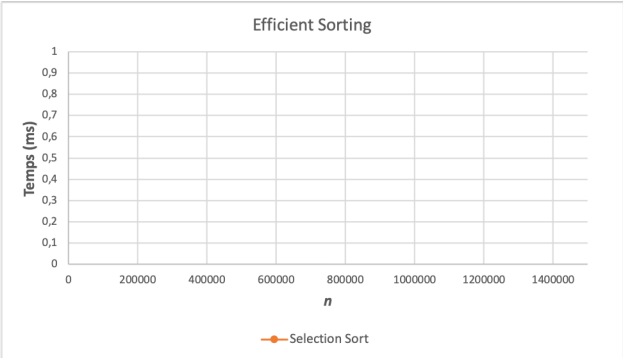


Figura 1: Gràfic del temps d'execució en funció de la longitud de l'array.

3 Quin és l'ordre de complexitat de l'algorisme *selection sort*?

Observant el gràfic generat a partir de les dades de la Taula 1 i el gràfic de la Figura 2, quin és l'ordre de complexitat temporal de l'algorisme *selection sort*?

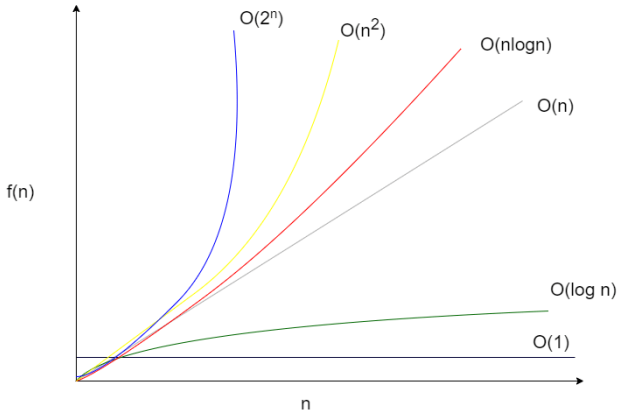


Figura 2: Ordre de complexitat temporal.

4 Implementació de l'algorisme *selection sort* en Android

1. Creau un projecte Android anomenat també *EfficientSorting* amb la plantilla *Empty views activity*.

2. Editau l'arxiu *"activity_main.xml"* i afegiu els següents elements per tal d'aconseguir una pantalla similar a la que es mostra a la figura 4(utilitzau les constraints necessàries):

   - Un camp de text (*EditText*) per indicar la longitud de l'array que es vol ordenar.
   - Un *Button* amb el text *Selection Sort* que s'encarrega d'inicialitzar la funcionalitat de l'aplicació.
   - Un label *TextView* que serveix per visualitzar el contingut de l'array original.
   - Un label *TextView* que serveix per visualitzar el contingut de l'array una vegada ordenat.

- Un label *TextView*, a la part inferior de la pantalla, per mostrar els milisegons que tarda l'ordenació.

3. Quan es pitgi el botó *Selection Sort*, ha de cridar-se un mètode de la nostra activitat principal que:

   - Crei un array de n números enters aleatoris, on n és el valor especificat al *EditText* de la longitud.
   - Mostri l'array de números enters generat dins l'element *TextView* corresponent.
   - Ordeni l'array generat amb el mètode de *Selection Sort*.
   - Mostri els resultat de l'ordenació dins el *TextView* corresponent.
   - Mostri, al *TextView* de la part inferior, quants de milisegons ha tardat l'execució.

   Per evitar que una execució massa llarga de l'ordenació col·lapsi la interfície d'usuari (podria fer que Android mostri un missatge de que l'aplicació tarda massa), hauríeu d'utilitzar una execució a un nou fil (*thread*).

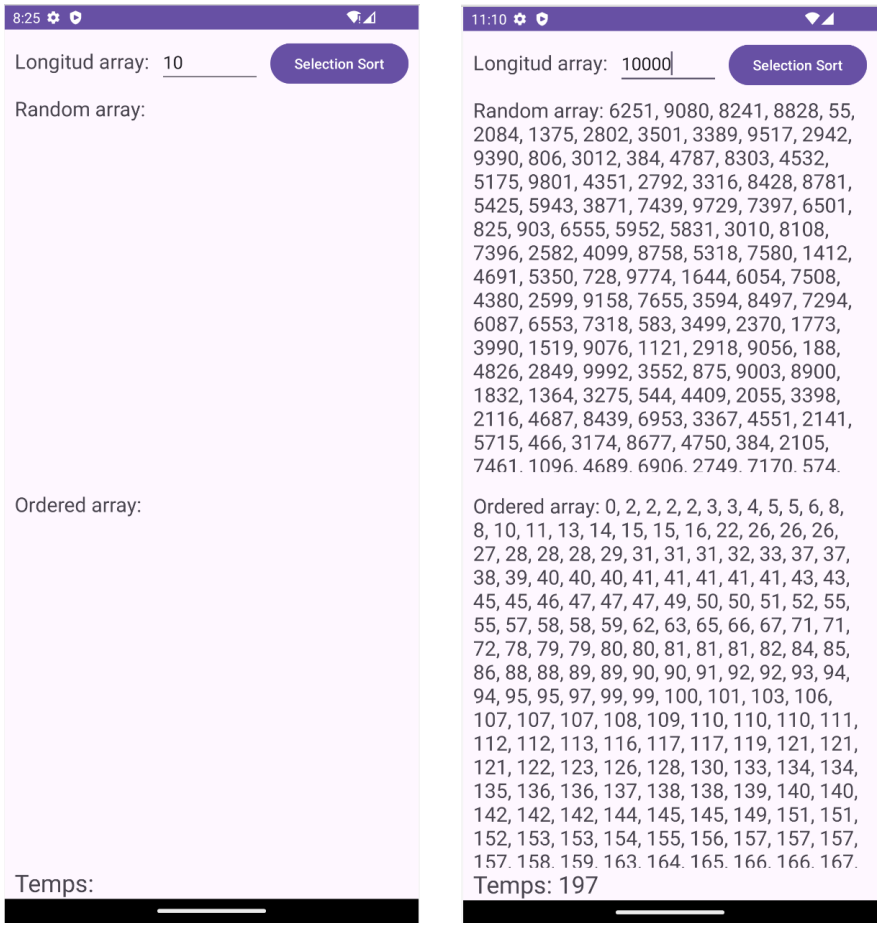
   En java, podeu utilitzar el següent codi per llençar un fil nou d'execució:

```
        Thread thread = new Thread() {
            @Override
            public void run() {
                // La vostra crida
            }
        };
        thread.start();
```

   Si, en un moment donat, dins aquest codi, voleu modificar el valor d'un text de la interfície (pista: per exemple, mostrar el número de milisegons que tarda l'execució), haureu de fer la crida perquè es modifiqui dins el *thread* principal:

```
        Thread thread = new Thread() {
            @Override
            public void run() {
                // La vostra crida
                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        // Modificar el TextView
                    }
                });
```

```
11                }
12            };
13            thread.start();
```



(a) (b)

Figura 3: (a) El nostre layout inicial i (b) el resultat després de pitjar el botó *Selection Sort*