VNIVERSITAT DE VALÈNCIA

Laboratorio de ESTRUCTURAS DE DATOS Y ALGORITMOS
Grado en Ciencia de Datos (1º)
Curso 2024-25

Práctica Nº 2: Algoritmos de ordenación

Periodo de realización: Semana del **24/02/2025 a 28/02/2025**

Metodología de trabajo

Fases	Tipo	Dedicación
1. Resolver las tareas planteadas en este guion antes de iniciar la sesión presencial en el laboratorio.	No presencial	Máx. 4,5 h.
2. Resolver en el laboratorio un nuevo ejercicio basado en la resolución de las tareas previas.	Presencial	Máx. 3 h.

Introducción

Si en la primera práctica se trataba el problema de la búsqueda de información en una lista de datos, en esta segunda práctica se va a trabajar la ordenación de una lista con objeto de poder realizar de manera eficiente otros procesos como, por ejemplo, la propia búsqueda.

En esta práctica vamos a abordar el cálculo empírico de costes de diversos algoritmos de ordenación. En concreto, se pretende obtener resultados experimentales del coste real en pasos de la implementación en Python de los dos algoritmos de ordenación analizados en las clases de teoría:

1. Ordenación por selección.
2. Quicksort.

Las listas de datos (y los archivos) con los que se van a trabajar son las mismas que en la anterior práctica, listas de personas vacunadas de Covid-19. Pero en esta ocasión se ha modificado el tipo de datos definido para introducir una representación mejorada de la clase *PersonaVacunada*, en la que se reflejan los nuevos conceptos de programación orientada a objetos (clases) vistos en teoría. El archivo "**`pr2_persona.py`**" (Aula Virtual) incluye la definición del tipo de datos a utilizar (clase *PersonaVacunada*). En este archivo se puede observar que se ha cambiado la definición de las fechas utilizadas, que ahora corresponden a un nuevo tipo *Fecha*, declarado en el archivo "**`pr2_fecha.py`**".

El material proporcionado para la práctica se complementa con el archivo "**`pr2_algoritmos.py`**", que incluye la implementación de los 2 algoritmos de ordenación que se van a utilizar y comparar.

Antes de empezar a trabajar es **muy importante** leer la documentación del código proporcionado para interpretarlo y saber cómo utilizarlo correctamente en la resolución de los ejercicios que se plantean. En las dos clases que ya están construidas (*PersonaVacunada* y *Fecha*) se ha incluido comentarios (*docstring*) que resumen las operaciones públicas de cada clase. Este resumen tiene como objeto facilitar la utilización de los tipos en los ejercicios.

Ejercicios (Fase 1)

Tarea 1

Se debe escribir un programa[1] "**`pr2_v1.py`**" que realice las siguientes tareas:

1. Leer y cargar en una lista la información de personas vacunadas disponible en el archivo "**`vacunaciones_small.dat`**".
2. Mostrar por pantalla los datos cargados en la lista y verificar que corresponden con los disponibles en el archivo.

Para hacer este programa se puede usar como referencia el utilizado en la práctica 1, puesto que se debe leer el mismo archivo de datos y, por tanto, el formato no cambia y el mecanismo de lectura tampoco debería cambiar. El único detalle a considerar es la forma de introducir los datos de una persona. Ahora ya no es posible hacerlo como en la práctica anterior, puesto que el tipo *PersonaVacunada* es una clase donde los datos son privados y no accesibles. Por contra, la clase dispone de una operación para asignar datos que facilita mucho esta tarea.

Tarea 2

Modifica el programa anterior para que ordene la lista de vacunaciones utilizando como criterio la fecha de la dosis 1 de las personas vacunadas. Para ello, hay que tener en cuenta que los algoritmos proporcionados (Selección y Quicksort) están planteados de manera genérica y en su estado actual comparan elementos completos de la lista (todo el registro/objeto). Sin embargo, lo que nos interesa ahora es comparar exclusivamente las fechas de la primera dosis de las personas que componen la lista. Por eso, comparaciones del estilo `v[j] < v[pos_min]` no tendrán sentido puesto que se están comparando personas completas y no solo las fechas. Revisa la definición de la clase *PersonaVacunada* para modificar este tipo de comparaciones y adaptarlas al criterio de ordenación requerido.

Se debe prestar una especial atención a la definición del pivote del algoritmo Quicksort. En este algoritmo, el pivote es la referencia para hacer la partición (y las comparaciones) y ahora este pivote debe ser una fecha, no una persona.

Una vez adaptados los algoritmos de ordenación debes hacer que el programa ordene la lista de personas vacunadas leídas del archivo "`vacunaciones_small.csv`", la ordene por el algoritmo de *Selección* y muestre por pantalla los pasos realizados y toda la lista resultante para comprobar que realmente se han ordenado los datos correctamente. Los resultados que se deben obtener son:

- Pasos = 54
- Orden (solo se muestran identificadores):
  1. 0002MDM
  2. 0007LRV
  3. 0014FGV
  4. 0002JTX
  5. 0005PZF
  6. 0009LFP
  7. 0001WQH
  8. 0001TQS
  9. 0006HYR
  10. 0012VRY

[1] Se deben importar adecuadamente los archivos que se proporcionan como material de la práctica.

Modifica el programa para que ordene la lista con el algoritmo *Quicksort*, y no por Selección, comprueba que se obtiene el mismo orden y que el número de pasos realizados ahora es 19.

Como se puede observar, el algoritmo Quicksort es aproximadamente 3 veces más rápido que el algoritmo de Selección para ordenar solo 10 elementos, considerando que en los dos algoritmos solo se han contabilizado los pasos que implican comparaciones o modificaciones de elementos de la lista.

Tarea 3

Realiza la última modificación del programa para que ordene la lista de personas vacunadas con los dos métodos de ordenación en la misma ejecución, primero por *Selección* y después por *Quicksort*. En este caso, hay que tener en cuenta que tras la primera ordenación la lista quedará ordenada y, por lo tanto, el segundo algoritmo se aplicaría sobre una lista ordenada (mejor caso) y la comparación de resultados no tendría sentido porque se estarían realizando ordenaciones en situaciones diferentes de la lista. Para resolver este problema se debe hacer lo siguiente:

- Leer el archivo de datos y almacenar las personas en una lista (*vacunas*).
- Copiar la lista anterior en otra lista con la operación *copy*, `copia1 = vacuna.copy()`
- Realizar una segunda copia de la lista original, `copia2 = vacuna.copy()`
- Ordenar por Selección la primera copia.
- Ordenar por Quicksort la segunda copia.
- Mostrar resultados: número de pasos de cada algoritmo, la relación entre los pasos de Selección y Quicksort y la lista ordenada (solo una vez) para comprobar la ordenación.

La salida del programa debería ser algo como esto:

```
Seleccion = 54 pasos
Quicksort = 19 pasos
ratio Seleccion/Quicksort: 2.84

Lista ordenada por fecha 1a. dosis:
    - ID: 0002MDM
    - Nombre: Ferris Rovira, Nuria
    - Vacuna: Moderna
    - Dosis recibidas: 3
    - Fecha dosis 1: 20/1/2021
    - Fecha dosis 2: 10/3/2021
    - Fecha dosis 3: 20/9/2021
    - Fecha proxima dosis: 31/12/9999

    - ID: 0007LRV
    - Nombre: Santos Trujillo, Victor
    - Vacuna: Moderna
    - Dosis recibidas: 3
    - Fecha dosis 1: 5/3/2021
    - Fecha dosis 2: 3/5/2021
    - Fecha dosis 3: 9/11/2021
    - Fecha proxima dosis: 31/12/9999

    - ID: 0014FGV
    - Nombre: Ferris Sobrino, Oscar
    - Vacuna: Pfizer
    - Dosis recibidas: 3
    - Fecha dosis 1: 16/4/2021
    - Fecha dosis 2: 25/6/2021
    - Fecha dosis 3: 22/12/2021
    - Fecha proxima dosis: 31/12/9999
```

etc…

Si pruebas a ejecutar ahora el programa con el archivo “vacunaciones_big.csv” (no muestres la lista ordenada) las diferencias entre algoritmos son mucho más apreciables. Los resultados serían:

```
Seleccion = 12920985 pasos
Quicksort = 57782 pasos
ratio Seleccion/Quicksort: 223.62
```

Ahora, con más de 5000 elementos en la lista, el algoritmo *Quicksort* es más de **200** veces más rápido que el algoritmo *Selección*.

Fase 2: Tarea final

Resolución de un nuevo ejercicio planteado durante la sesión presencial en el laboratorio.