Universidad de Burgos
Departamento de Ingeniería Informática
Lenguajes y Sistemas Informáticos

Algoritmia
3º Grado en Ingeniería Informática
Métodos voraces

Práctica Obligatoria 1
Algoritmos voraces y divide y vencerás

Créditos: 0.6 (15 horas aproximadamente); **Peso: 10 %**

La empresa de *streaming* ACME está desplegando servidores caché por Europa para poder garantizar una velocidad adecuada a sus clientes. Para abaratar costes ha hecho un estudio sobre qué películas y series son más demandadas en cada país. Se os ha contratado para resolver algunos problemas en este despliegue.

1. Problemas a resolver

1.1. Ranking de contenidos

Se ha planteado como prioritario disponer de herramientas eficientes de ordenación de los contenidos. Esto permitirá elaborar varios tipos de informes en tiempo real y realizar recomendaciones a los usuarios. Se pide:

1. Implementar un algoritmo de ordenación eficiente[1], basado en el paradigma Divide y Vencerás, que permita ordenar ascendentemente una lista de vídeos según un criterio arbitrario. Debe disponerse de un método con los siguientes parámetros: una función que devuelve un valor numérico para cada elemento (clave de ordenación), una secuencia de vídeos (por ejemplo, una lista) y un parámetro opcional para invertir el sentido de la ordenación.

2. Comprobar que pueden ordenar diferentes conjuntos de vídeos por criterios como el número de espectadores, el tamaño, o cualquier otro valor clave. Por defecto deben ordenarse por su tamaño.

1.2. Optimizar carga de los servidores

ACME se plantea optimizar la utilización de los servidores desplegados en cada país. Se pide:

- Dada la capacidad de un servidor y el número de espectadores de cada vídeo y su tamaño, obtener, mediante un planteamiento voraz, qué vídeos deberían ser almacenadas en el servidor para maximizar el *tiempo de emisión*.

El tiempo de emisión es la duración total emitida desde un servidor, sumando los streams individuales a todos los espectadores. Se asume que la duración es proporcional al espacio de almacenamiento de cada vídeo. Por ejemplo, si en un servidor se aloja una película de 10 GB con 1000 espectadores y media película de 20 GB con 300 espectadores, el tiempo de emisión para ese servidor será proporcional a $10 \cdot 1000 + \frac{20}{2} \cdot 300$ (esta es la cantidad a maximizar).

Nótese que en cada servidor no se tiene por qué almacenar el vídeo completo. Un espectador puede comenzar a ver una película en un servidor y terminarla en otro (la tecnología de streaming de ACME permite gestionar este cambio de manera transparente.)

[1]Dado que el ejercicio consiste en implementar un algoritmo de ordenación, no se permite emplear las funciones de ordenación disponibles en las bibliotecas nativas de Python ni en ninguna otra externa.

1.3. Optimizar conexión entre servidores

Existe un servidor central con la información de la conexión entre servidores. Cuando un usuario solicita una serie/película el servidor central le manda al servidor de caché más cercano, pero si no está disponible solicita al siguiente servidor más cercano. Se conoce el *ping* medio entre cada par de servidores. Se pide:

- Diseñar un método voraz para simplificar la red de relaciones entre los servidores, de manera que todos ellos mantengan la conectividad, para cada uno pueda conocerse de inmediato cuál es el más cercano y se almacene la mínima cantidad de información necesaria.

2. Informe

Junto con la solución planteada se pide que se respondan las siguientes preguntan que se plantean en el *Notebook* de la práctica:

1. Análisis de la complejidad temporal de la solución de cada problema.
2. Responda a las preguntas:
   - Respecto al ranking de contenidos (Justifica la respuesta):
     - ¿Cómo afecta la función clave, que proporciona el criterio de ordenación, a la complejidad temporal del algoritmo?
     - ¿Has usado alguna estructura de datos adicional?
   - Respecto a la carga de datos en los servidores (Justifica la respuesta):
     - ¿Cómo de buenas son las soluciones encontradas? ¿Son óptimas o aproximadas?
     - ¿Qué ocurriría si solo se admitiese almacenar vídeos completos en cada servidor?
     - ¿Qué estructuras de datos has utilizado?
   - Respecto a la conexión entre servidores (Justifica la respuesta):
     - ¿Cómo de buenas son las soluciones encontradas? ¿Son óptimas o aproximadas?
     - ¿Cómo afecta el número de conexiones entre servidores a la complejidad espacial y temporal del algoritmo empleado?
     - ¿Qué estructuras de datos has utilizado?

3. Rúbrica

Si no se cumple con los requisitos (no pasa ninguno de los test), con el enunciado, se utilizan librerías no nativas de *Python* o no se contestan las preguntas, se tendrá una puntación de **cero** en toda la práctica.

No se pueden cambiar las cabeceras de las clases y funciones del *Notebook* provisto, pero sí se puede y se recomienda crear funciones y clases adicionales para facilitar el buen diseño de la solución.

$$(0{,}7 * \text{código} + 0{,}3 * \text{complejidad temporal}) * \text{informe}$$

3.1. Código

- Código repetido, inútil o no general[2]: -0.25 puntos por cada ocurrencia.

3.2. Informe

- Análisis temporal: 4 puntos
- Preguntas: 6 puntos

[2]Un código no general es aquel que no funcione en las versiones de Python de 3.10 a 3.13

4. Entrega

La entrega de la práctica consistirá en un fichero .ipynb con el *Notebook* con el código y las preguntas respondidas.

El fichero deberá tener el siguiente nombre:

`<ApellidosPrimerAlumno>_<ApellidosSegundoAlumno>_po1.ipynb`

Así, si los alumnos fueran "José Luis Garrido Labrador" e "Ismael Ramos Pérez", la entrega sería:

- GarridoLabrador_RamosPerez_po1.ipynb

Nota importante 1: si el documento no tiene este formato de nombre, la práctica entera tendrá una penalización de **dos puntos** sobre el total.

Nota importante 2: en caso de que se entregue por parejas **ambas personas** deberán hacer la entrega y deberán entregar **el mismo fichero**, si uno de la pareja no hace la entrega se evaluará como "No presentado". En caso de que estos ficheros sean diferentes (bajo verificación por SHA256 y manual) se tendrá la penalización de **dos puntos** y se tendrá en cuenta cada práctica por separado.

Nota importante 3: en el caso de que se suba un fichero que no es un ***Jupyter Notebook***, **esté corrupto** o **tenga fallos de sintaxis** alguna parte del código, la práctica tendrá una nota de 0. Por tanto, verificar que la entrega ha sido correcta. Tampoco se permite en el código llamadas a funciones del sistema, de tener alguna de estas llamadas la práctica tendrá también una nota de 0.