Universidad de Burgos
Departamento de Ingeniería Informática
Lenguajes y Sistemas Informáticos

Algoritmia
3º Grado en Ingeniería Informática
Divide y vencerás

Práctica Obligatoria 2
Divide y Vencerás

Créditos: 0.4 (10 horas); **Peso: 6,67 %**

El despliegue de la empresa de *streaming* ACME en Europa ha sido todo un éxito. La utilización de los servidores caché es óptima, lo que ha permitido alcanzar un nivel de servicio competitivo manteniendo los costes de infraestructura dentro del presupuesto previsto.

Todos los departamentos están muy satisfechos, con una excepción. El área de marketing (cuyo responsable ha recalcado en el último comité de dirección que "esas métricas están muy bien" pero también que "la competencia ha ganado más cuentas en febrero"), reclama mejores herramientas para comprender a los usuarios y planificar la compra de nuevos contenidos. Así las cosas, la dirección ejecutiva de ACME se ha visto obligada a prolongar vuestro contrato.

1. Problemas a resolver

1.1. Ranking de contenidos

hay q usar el algorit de ordenacion q queramos. orden por campo dado

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 su tamaño.

1.2. Optimizar la compra de contenidos

El Departamento de Marketing de ACME dispone de un *forecast* consistente en previsiones semanales de la demanda en Europa (en número de potenciales espectadores) de diferentes contenidos audiovisuales. Para cada contenido, ACME puede decidir adquirir la licencia para su emisión en exclusiva, empezando y terminando en dos semanas cualesquiera. Se dispone también de una estimación de los ingresos semanales que cada espectador proporciona a ACME por cada contenido que visualiza.

1. Conocido el coste semanal en euros de la licencia para un contenido audiovisual, diseñar un algoritmo que permita determinar cuál es el máximo beneficio que puede obtenerse adquiriendo la licencia para un periodo de emisión consecutivo óptimo (es decir, la mejor selección posible de semana de inicio y semana de finalización dentro del periodo considerado en las previsiones). Para ello, calcúlese el mejor resultado obtenible comprobando todas las combinaciones posibles de semana de inicio y fin.

   Por ejemplo, considérese la siguiente tabla con la previsión de espectadores para una película P_i que ACME está considerando adquirir, y cuyo coste de licenciamiento es $c_i = 1500€$ por semana. Supóngase también

	Semana								
	0	1	2	3	4	5	6	7	8
Previsión de espectadores	500	1000	2000	150	2000	1500	200	1200	700

[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.

que se estiman unos ingresos de $I = 2€$ por semana por cada espectador. A la vista de los datos, podemos comprobar que si ACME adquiriese la película para emitirla solamente en la semana 1, obtendría unos ingresos de $1000 \cdot 2€$, lo que le dejaría un beneficio de 500 € tras abonar el coste de licencia. Incluir la semana 2 aumentaría aún más el beneficio. Por el contrario, emitir desde la semana 1 a la 3 es peor opción, salvo que se incluyan semanas posteriores; puede comprobarse que el mejor periodo consecutivo de emisión se logra adquiriendo los derechos de emisión desde la semana 1 a la 5, que reporta un beneficio óptimo de 5800€.

Los ingresos por usuario I, la previsión semanal de espectadores para un contenido P_i, con un número arbitrario de semanas, y el coste semanal de licenciamiento c_i se consideran argumentos de entrada del algoritmo. El algoritmo debe devolver únicamente el máximo beneficio obtenible (no es necesario devolver el intervalo de emisión correspondiente).

2. Para mejorar la complejidad temporal de la solución anterior, resolver el problema mediante un enfoque basado en el paradigma Divide y Vencerás. Para ello, téngase en cuenta lo siguiente:

   - Considerando dos mitades de la previsión semanal de espectadores, la secuencia óptima se encuentra, o bien en la mitad derecha, o bien en la mitad izquierda, o bien cruza el centro de la previsión.
   - Si conocemos la solución al problema en la mitad derecha y en la mitad izquierda, podemos compararlas con la posible solución que cruza el centro y devolver la mejor de las tres.
   - Obtener la mejor solución que cruza el centro puede resolverse con complejidad temporal lineal.

3. Dado un conjunto de contenidos con sus correspondientes costes de licencia y previsiones de espectadores, utilice el algoritmo diseñado en 1.1 para seleccionar el mejor subconjunto de k contenidos cuya adquisición resultaría rentable (o un número menor que k si no hay suficientes contenidos rentables).

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 cada algoritmo implementado (ordenación de 1.1, las dos versiones del optimizador y el ranking de contenidos rentables de 1.2)

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?
   - Respecto a la optimización de la compra de contenidos (Justifica la respuesta):
     - ¿Has conseguido mejorar la complejidad temporal mediante la solución Divide y Vencerás? ¿Por qué ha mejorado? ¿Cuánto ha mejorado? Pon algunos ejemplos numéricos especulativos que relacionen el tamaño del problema con el tiempo de ejecución en ambas versiones. Contrástalos de forma empírica empleando para ello tu implementación.

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.

3.1. Código 6/10

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

3.1.1. Ranking de contenidos (40 %)

- Funcionalidad correcta (supera las pruebas del profesor): 7 puntos
- Complejidad temporal óptima: 2 puntos
- Complejidad espacial óptima: 1 punto

3.1.2. Optimizar la compra de contenidos(60 %)

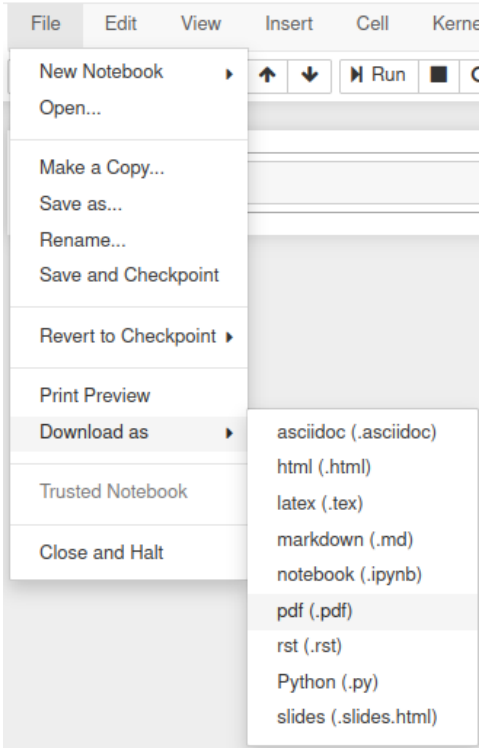
- Funcionalidad correcta (supera las pruebas del profesor): 7 puntos
- Complejidad temporal óptima: 2 puntos
- Complejidad espacial óptima: 1 punto

3.2. Informe 4/10

- Análisis de complejidad temporal: 60 %
- Preguntas: 40 %

4. Entrega

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



Los ficheros deberán tener el siguiente nombre:

- `<ApellidosPrimerAlumno>_<ApellidosSegundoAlumno>_dyv.ipynb`
- `<ApellidosPrimerAlumno>_<ApellidosSegundoAlumno>_dyv.pdf`

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

- GarridoLabrador_RamosPerez_dyv.ipynb
- GarridoLabrador_RamosPerez_dyv.pdf

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.