Universidad de Burgos
Departamento de Ingeniería Informática
Lenguajes y Sistemas Informáticos

Algoritmia
3º Grado en Ingeniería Informática
Programación dinámica

Práctica Obligatoria 3
Programación Dinámica

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

Tras la puesta en producción de los sistemas de optimización de contenidos, se ha producido un aumento notable en el consumo y marcados cambios en la dinámica de streaming. Habéis propuesto a *Acme* un nuevo contrato para mejorar la situación.

1. Problemas a resolver

1.1. Optimizar carga de los servidores (II)

El rápido aumento en el número de servidores ha provocado que crezca en la misma medida el número de contenidos almacenados parcialmente. Esta fragmentación excesiva, junto con la cada vez mayor base de usuarios, ha provocado que se disparen los cambios de servidor durante el streaming, hecho que está empezando a afectar al rendimiento del sistema en su conjunto.

Se ha decidido sacrificar parte del espacio de almacenamiento para garantizar que las sesiones empiezan y acaban en la misma máquina. Por ello se pide:

1. Implementar un algoritmo que maximice el tiempo de emisión de cada servidor garantizando que no se parten ninguno de los vídeos.

1.2. Mapa de distancias entre servidores

Ahora la oferta de contenidos de *Acme* es mucho más amplia y dinámica. La Dirección Técnica ha decidido que no es viable mantener, con el sistema actual, copias para todos los contenidos en emisión en cada servidor. Por ello, se ha decidido sustituir la arquitectura de cachés simples por una de relays-caché. En este nuevo esquema, cada servidor almacena un conjunto exclusivo de contenidos actualizados y, a su vez, redirige los streams desde otros servidores para su puesta a disposición del usuario final, posiblemente almacenando copias temporales del contenido.

Cuando un usuario conectado a un servidor solicita un vídeo, debe determinarse la vigencia de los fragmentos locales para su emisión inmediata o actualización mediante una ruta al servidor origen del contenido. Para que este proceso sea óptimo se pide:

1. Implementar un algoritmo eficiente que calcule las rutas de menor latencia entre todos los servidores. Es decir, que dado un servidor A se pueda, a partir de un servidor arbitrario B llegar de A a B con la menor latencia total. Este método debe seguir el paradigma de **programación dinámica.**

2. En el caso de que haya dos caminos con la misma latencia, se debe seleccionar el que menos saltos tenga.

2. Informe

Junto con la solución planteada se pide un informe que contenga:

1. Análisis de la complejidad temporal y espacial de cada algoritmo implementado (Maximización del tiempo de emisión del servidor caché (1.1) y distancias mínimas entre servidores (1.2)

2. Responda a las preguntas:

   - Respecto a la maximización del tiempo de emisión del servidor caché (Justifica la respuesta):
     - ¿Cómo de buena es la solución? ¿Es óptima o aproximada?[1]

[1] No confundir una solución óptima (obtiene el máximo posible) frente a una solución eficiente (calcula el máximo con la menor complejidad posible).

    - ¿Cómo cambiaría la complejidad espacial si aumentase el tamaño del servidor? ¿Y si aumentase el número de vídeos?
  - Respecto a la búsqueda de distancias mínimas entre servidores (Justifica la respuesta):
    - Este problema tiene una aproximación voraz, ¿cuál sería la ventaja de usarlo frente a la solución que has planteado?
    - ¿Y cuál sería su desventaja?

3. Rúbrica

Si no se cumple con los requisitos, con el enunciado, se utilizan librerías no nativas de *Python* o no se presenta informe 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. Carga de los servidores (50 %)

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

3.1.2. Caminos y distancias entre los servidores (50 %)

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

3.2. Informe 4/10

- Análisis de complejidad temporal y espacial: 60 %
- Preguntas: 40 %
- Defectos formales: maquetación, ausencia de apartados necesarios (título, índice, autores...), faltas de ortografía, redacción: hasta -2 puntos.

4. Entrega

La entrega de la práctica consistirá en un fichero comprimido (*zip, tar, rar...*) con el informe en formato *pdf* y el código en *Jupyter Notebook (ipynb)* o *Python (py)*.

Los ficheros deberán tener el siguiente nombre:

`<primer apellido alumno><Segundo apellido alumno>_<primer apellido alumno><Segundo apellido alumno>`

Así si los alumnos fueran “José Luis Garrido Labrador” y “Daniel Setó Rey” la entrega sería:

- garridoLabrador_setoRey.zip
  - garridoLabrador_setoRey_informe.pdf
  - garridoLabrador_setoRey_codigo.ipynb