Estructuras de Datos y Algoritmos

Práctica II - Curso 2021/22

La Red Social (2)

1. Introducción

Aunque en la primera práctica hemos conseguido una aplicación que resuelve el problema planteado, el hecho de que tarde cientos de años en obtener el resultado para situaciones reales (1.000.000.000 de usuarios) nos ha hecho plantearnos que quizás sea conveniente dedicar algo de esfuerzo a la optimización del tiempo empleado por el programa.

Las etapas de lectura del fichero, ordenación-selección de grumos y escritura del fichero de nuevas relaciones no parece que planteen ningún problema y no vamos a intentar optimizarlas. El problema está en las otras dos etapas, que tienen un orden de $O(n^2)$. Nuestro objetivo va a ser conseguir reducir ese orden a $O(n)$ mediante el uso de estructuras de datos avanzadas, de forma que pasemos de tardar cientos de años a tardar minutos en resolver el problema.

2. Descripción Técnica

Hay dos maneras de conseguir el objetivo deseado:

1. Cambiando varias de las estructuras utilizadas (basadas en ArrayList de Java o listas de Python, que utilizan una implementación contigua) por implementaciones basadas en tablas de dispersión. Con este enfoque no es necesario cambiar la estructura del programa y se siguen teniendo las mismas etapas basadas en la funciones uber_amigos, crea_usuarios y crea_grumos (adaptadas a las nuevas estructuras de datos). Si se sigue éste enfoque (el camino fácil) se puede obtener **como máximo 6.67 puntos sobre 10** en la nota de la práctica.
2. Basar la solución en una estructura de datos principal que se adapta perfectamente al tipo de problema que queremos resolver. También se necesitaría una estructura de datos secundaria para que la principal pueda desplegar todo su potencial. En este caso si que cambia la estructura del programa, las etapas de crear usuarios y crear grumos se sustituirían por una única etapa y ya no sería necesaria la función uber_amigos. Si se sigue éste enfoque (el camino profesional) se puede obtener **como máximo 11 puntos sobre 10** en la nota de la práctica.

   Pista: La estructura de datos principal se puede encontrar buscando en Internet alguno de los siguientes términos clave: *treaps*, *k-d trees*, *fibonacci heap*, *suffix tree*, *disjoint sets*, *van Emde Boas priority queue*, *Shor algorithm*.

En la evaluación también se tendrá en cuenta si las estructuras de datos están codificadas por el alumno (adaptadas por tanto a las características del problema y definiendo únicamente las operaciones necesarias) o bien se utilizan estructuras de datos ya predefinidas en el entorno de programación. **En éste último caso la nota se multiplica por el factor 0.75**.

Por ejemplo, si se elige el primer enfoque y no se define ninguna estructura de datos sino que se utilizan las estructuras basadas en tablas de dispersión del entorno (por ejemplo HashMap/HashSet

de Java o conjuntos/diccionarios de Python), la nota máxima posible en la práctica sería de 0.67 × 0.75 = 5 puntos sobre 10. Si se elige el segundo enfoque y tanto la estructura de datos principal como la secundaria están definidas por vosotros, la nota máxima posible sería de 11 puntos sobre 10.

Nota: En alguno o ambos de los enfoques puede ser importante el poder representar los grumos de la forma más compacta posible, por eso se permite que a la hora de calcular las nuevas relaciones de amistad necesarias sea posible usar un único usuario del grumo (de esa forma la única información que es necesario almacenar sobre cada grumo es su tamaño y el identificador de un usuario cualquiera). Ejemplo: En el recuadro de la izquierda se muestra la salida habitual (primer usuario de un grumo relacionado con segundo usuario del siguiente grumo), también es válido proporcionar una salida como la del recuadro de la derecha (solo se usa un usuario de cada grumo):

```
...
Nuevas relaciones de amistad..
29917185 <-> 85819403
32800774 <-> 84738093
95715370 <-> 60358727
30179397 <-> 39682073
```

```
...
Nuevas relaciones de amistad..
29917185 <-> 85819403
85819403 <-> 84738093
84738093 <-> 60358727
60358727 <-> 39682073
```

3. Objetivos

Además de crear la aplicación, el otro objetivo de la práctica será el **evaluar su eficiencia** respecto al tiempo, realizando una serie de medidas del tiempo empleado para distintos tamaños del fichero de entrada. Esta medida se va a utilizar para **estimar** el tiempo que tardaría la aplicación con el caso real de 1.000.000.000 de usuarios.

En esta segunda práctica no se va a pedir el análisis teórico.

4. Presentación y Evaluación de la práctica

Las prácticas se deben realizar de forma **individual**. Se pueden codificar en Java o Python.

Para una correcta evaluación de la práctica el alumno deberá:

1. Presentar electrónicamente (por el Aula Virtual de la Escuela o por correo electrónico), antes de la fecha límite el código fuente y los documentos que se indiquen. En el código fuente debe aparecer (como comentario) en las primeras líneas el nombre del alumno que ha realizado la práctica.
2. Presentarse a la sesión de evaluación que le corresponda según su grupo de laboratorio en la semana establecida.

Es en la defensa de la práctica donde se produce la evaluación, la presentación electrónica es simplemente un requisito previo para garantizar la equidad entre subgrupos y la comprobación preliminar de autoría.