ALGORITMIA

Grado en Ingeniería Informática del Software - Curso 2015-2016

# GUIÓN DE LA PRÁCTICA 4

## OBJETIVO:

- **Algoritmos ávidos o devoradores**

## Minimizar el intercambio de dinero

En muchas ocasiones, por ejemplo, cuando viajamos en grupo, tenemos que hacer intercambios de dinero cuando varias de las personas han realizado algún pago y llega el momento de ajustar cuentas. De hecho, existen ya múltiples aplicaciones informáticas que sirven para gestionar ese tipo de tareas: *Group Expense Calculator*, *MissPlitty*, *PayPool Share Costs Calculator*, etc.

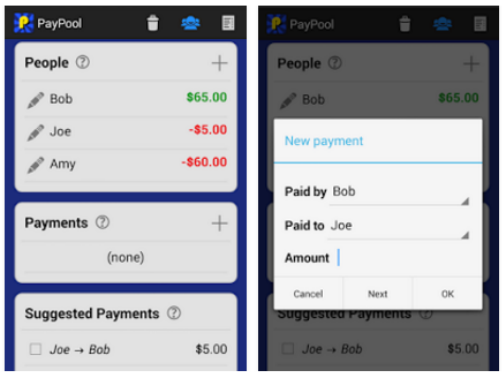
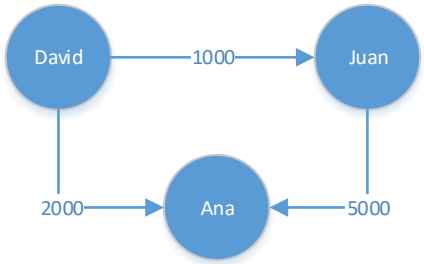


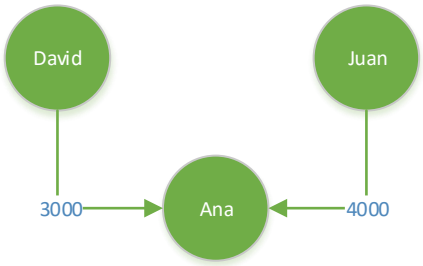

*Ilustración 1. Imagen de PayPool Share Costs Calculator*

La clave de dichas aplicaciones es saber cómo minimizar el flujo de dinero entre las diferentes personas. Veámoslo con dos ejemplos:

**Ejemplo 1**

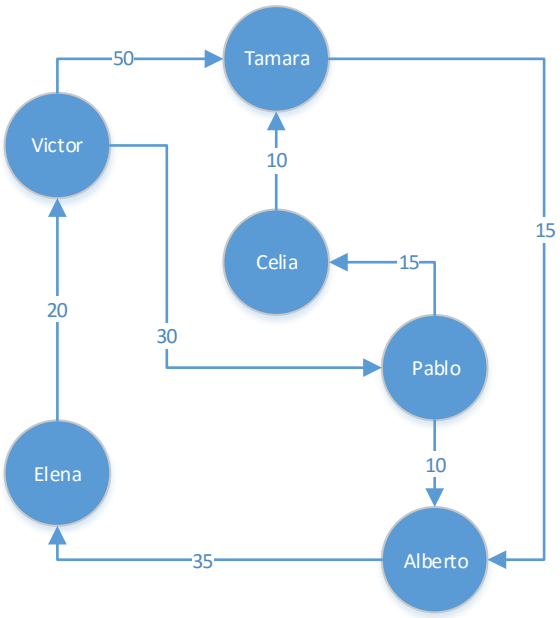



En este caso David le debe a Juan 1000 (podrían ser, por ejemplo, euros), Juan le debe a Ana 5000 y David le debe a Ana 2000. Podrían hacer esos 3 movimientos de dinero, pero sería más óptimo realizar los siguientes:

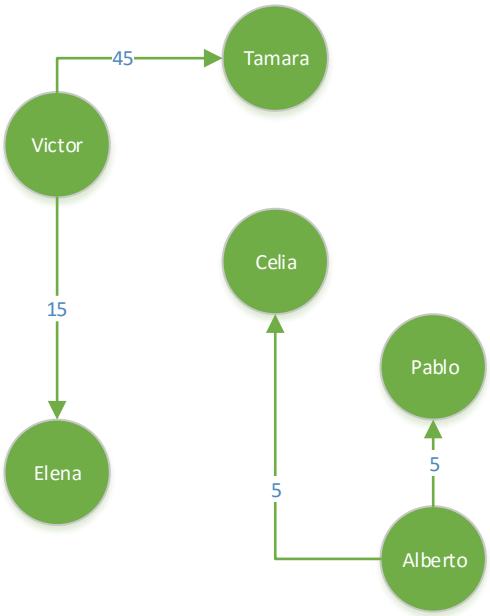



Es decir, si David y Juan le pagan 3000 y 4000 euros respectivamente a Ana, las 3 personas habrán ajustado cuentas minimizando el flujo de dinero entre ellas (el número de movimientos de dinero).

**Ejemplo 2**




Este caso es un poco más complejo, pero la idea es exactamente la misma. A continuación, se muestra un flujo de dinero mucho más óptimo:




## Trabajo pedido

*Si utiliza Eclipse, se creará el proyecto* ***prac04_Dev<3 últimas cifras dni>*** *con todas las clases necesarias.*
*Si utiliza JDK, las clases necesarias se crearán dentro del paquete* ***alg<dnipropio>.p4***

SE PIDE diseñar e implementar un algoritmo capaz de resolver el citado problema.

Probar el algoritmo implementado para comprobar, que efectivamente, proporciona la solución correcta, al menos para los dos ejemplos antes citados (`caso1.txt` y `caso2.txt` respectivamente). Se cargarán desde fichero de la siguiente forma:

- `MinimizarFlujoDinero [RUTA_FICHERO]`

El programa deberá mostrar los datos leídos originalmente (por ejemplo, para `caso2.txt`):

```
**Computando balance de cuentas:
Victor tiene que pagar 50 a Tamara
Victor tiene que pagar 30 a Pablo
Tamara tiene que pagar 15 a Alberto
Pablo tiene que pagar 10 a Alberto
Pablo tiene que pagar 15 a Celia
Alberto tiene que pagar 35 a Elena
Elena tiene que pagar 20 a Victor
Celia tiene que pagar 10 a Tamara
```

…y los resultados finales (por ejemplo, para `caso2.txt`):

```
**Resultados finales:
Victor tiene que pagar 45 a Tamara
```

```
Victor tiene que pagar 15 a Elena
Alberto tiene que pagar 5 a Celia
Alberto tiene que pagar 5 a Pablo
```

¿Qué complejidad tiene el algoritmo diseñado? ¿Se podría cambiar el diseño para mejorar la complejidad?

Si se dispone de varios algoritmos para resolver el problema, comparar los tiempos obtenidos para los mismos casos de prueba.

*Se entregará:*

- *Los ficheros fuente de las clases, que se hayan programado, dentro del paquete o del proyecto Eclipse.*
- *Un documento Word con una pequeña explicación del o de los algoritmos utilizados y de sus complejidades.*

*Se habilitará una tarea en el campus virtual para realizar la entrega. El plazo de entrega será de 1 semana y 1 día a partir de finalizar la sesión.*