ALGORITHMICS

Bachelor of Software Engineering - Course 2015-2016

SESSION 4

GOAL:

- **Greedy algorithms**

Minimize cash flow

On many occasions, for example, when traveling in groups, we have to exchange money when several people have made payments and it's time to settle accounts. In fact, there are already multiple software applications used to manage such tasks: *Group Expense Calculator*, *MissPlitty*, *PayPool Share Costs Calculator*, etc.

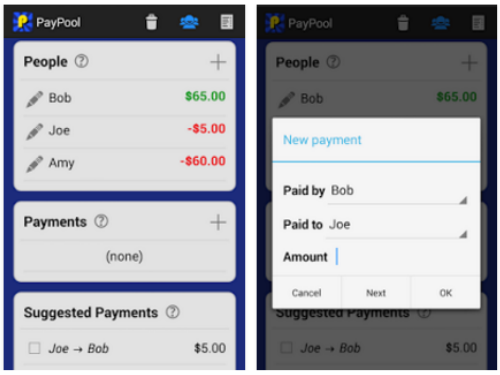
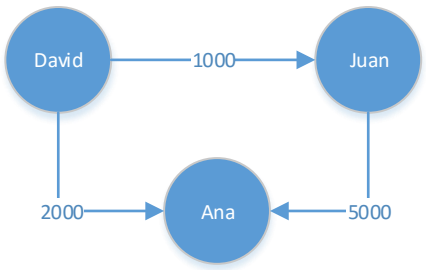


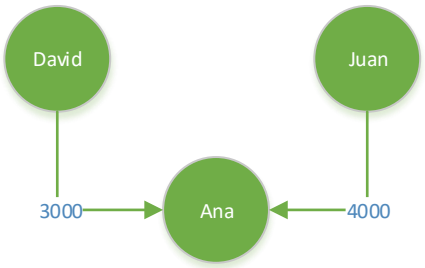
Figure 1. Picture of PayPool Share Costs Calculator

The key to those applications is to know how to minimize the flow of money among different people. Let's see two examples:

Example 1

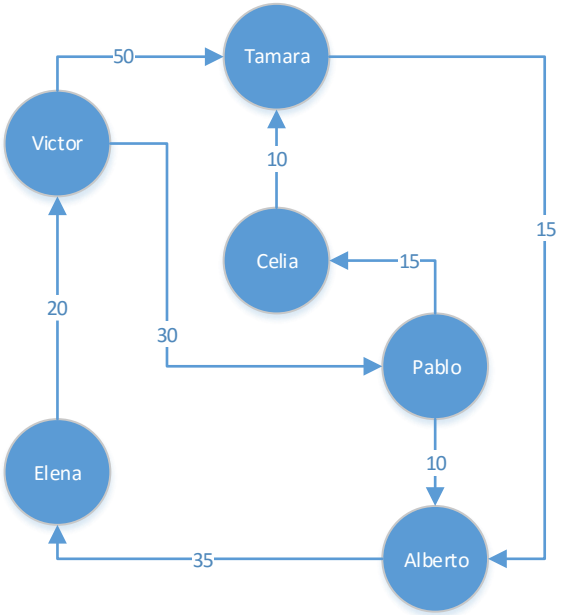


In this case David owes Juan 1000 (they could be, for example, euros), Juan owes Ana 5000 and David owes Ana 2000. They could make those 3 movements of money, but it would be more optimal to perform the following ones:

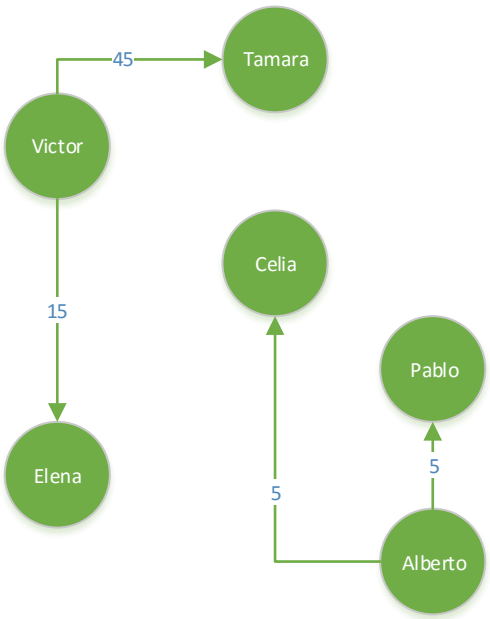


That is, if David and Juan pay 3000 and 4000 euros respectively to Ana, the 3 people have settled accounts minimizing the flow of money between them (the number of movements of money).

Example 2



This case is a bit more complex, but the idea is exactly the same. The following is a more optimal flow of money:



DEADLINE AND WORK TO BE DONE

You should create an Eclipse Project with the name: ***lab04_Greedy<UO>*** *with all the needed files inside*

You are asked to design and implement an algorithm capable of solving the above problem.

Test the algorithm implemented to check, which effectively provides the right solution, at least for the two examples cited above (`case1.txt`, `case2.txt` respectively as well as for `case3.txt`). Those tests are included in the `MinimizeCashFlowTest.java` class, in which Junit test cases are provided. You only should modify the path for the provided text files (`case1.txt`, `case2.txt` and `case3.txt`) and build a program to make tests run successfully. That is usually called Test-Driven Development and is a best practice for software development you should take into account from now on.

The program will display the data originally read (for example, for `case2.txt`):

```
**Compute balance:
Victor has to pay 50 to Tamara
Victor has to pay 30 to Pablo
Tamara has to pay 15 to Alberto
Pablo has to pay 10 to Alberto
Pablo has to pay 15 to Celia
Alberto has to pay 35 to Elena
Elena has to pay 20 to Victor
Celia has to pay 10 to Tamara
```

…and the final results (for example, for `caso2.txt`):

```
**Final results:
Victor has to pay 45 to Tamara
Victor has to pay 15 to Elena
Alberto has to pay 5 to Celia
```

```
Alberto has to pay 5 to Pablo
```

What is the complexity of the designed algorithm? Could you change its design to improve the complexity?

If you have multiple algorithms to solve the problem, compare the times obtained for the same test cases.

You should deliver:

- *The source files for the classes, which have been programmed within the Eclipse project.*
- *A document (.docx or .pdf) with a brief explanation of the algorithm or algorithms used and their complexities.*

A task will be enabled in the virtual campus to upload the exercise. The deadline is 1 week and 1 day after the session ends.