ALGORITHMICS - Bachelor of Software Engineering, 2020

LAB GUIDE. SESSION 3

GOALS:

- **Divide and Conquer: recursive models and examples**

1. Basic recursive models

You are provided with the following 6 classes that you should try to understand one by one:

`Subtraction1.java` and `Subtraction2.java` classes have a scheme by subtraction with `a=1`, which involves a large expenditure of stack memory. In fact, it overflows when the problem size grows to several thousands. Fortunately, that type of problems will be better solved with an iterative solution (with loops) than with this solution (subtraction with `a=1`).

`Subtraction3.java` class has a scheme by subtraction with `a>1`, which involves a large time of execution (exponential, not polynomial). This means that for a relatively large size problem the algorithm does not end (untreatable time). The consequence is that we must try not to use subtraction type solutions with multiple calls (`a>1`).

`Division1.java`, `Division2.java` and `Division3.java` classes have a recursive scheme by division, being the first of type $a<b^k$, the second of type $a=b^k$ and the remaining one $a>b^k$.

2. Uncle Scrooge (In Spanish: El tío Gilito)

Uncle Scrooge (uncle of Donald Duck) is undoubtedly the wealthiest in Disney, since based on his financial operations (his detractors classify them within usury), he has accumulated a huge wealth, of which he does not lose the opportunity to show off.

The reality is that his Disney colleagues have no alternative but to ask him for money. Therefore, they ask for credits (**equal gold coins**), to which Uncle Scrooge, of course, applies a great interest rate. Debtors agreed so that when a credit is returned (**n gold coins**), they always put **one and only one fake coin**, which weighs **less** than the real ones.

Given that provocation and to detect the false coins, Uncle Scrooge bought a state-of-the-art electronic balance (It tells us if it weighs less on the left side, if it weighs less on the right side, or if both weigh the same).

Every time he uses the balance, he spends 1 watt of energy, regardless of the weight he puts on each plate.

He currently has the algorithm staged in the **Gilito1** class, which is a bad one and implies an expense in the electricity bill higher than the scam itself.

TO DO:

A. Work to be done

- An Eclipse `session3` **package** in your course project. The content of the package should be:
  - Package `session3.recursive`: All the files that were given with the instructions for this session together with two new classes:
    - `Substraction4.java`. You should create a new class and implement a recursive method by subtraction with a complexity $O(3^{n/2})$.
    - `Division4.java`. You should create a new class and implement a recursive method by division with a complexity $O(n^2)$ and a number of subproblems = 4.
  - Package `session3.gilito`: All the files that were given with the instructions for this session (`Gilito1.java` and `Gilito1Times.java`) together with three new classes:
    - `Gilito2.java`. Class with an algorithm that **minimizes** energy expenditure, and therefore the electricity bill.
    - `Gilito2Times.java`. Class to calculate the runtime and the energy in the worst case of the designed algorithm. The worst case involves placing the false coin in a position that assumes that the algorithm behaves as badly as possible. The time must be calculated for a number of coins `n` = 100, 200, 400, 800, etc.
    - `GilitoComparison.java`. Class that combines `Gilito1Times.java` and `Gilito2Times.java` to calculate and compare the runtime and the energy between the two alternatives. However, instead of the worst case, in this test the fake coin should be placed in a random position. That way, it will be possible to check if the two algorithms find it in the same position for the same case. The time must be calculated for a number of coins `n` = 100, 200, 400, 800, etc.
  - Package `session3.closest`: All the files that were given with the instructions for this session with all classes fully implemented.
- A **PDF document** using the course template. The activities of the document should be the following:
  - **Activity 1. Basic recursive models**.
    - A brief explanation (a paragraph) for each of the two new classes indicating how you calculate the complexity to get the requested one.
  - **Activity 2. Uncle Scrooge.**
    - Fill in a table with those energy expenditure values previously obtained using `Gilito2Times.java`. From the theoretical point of view, explain: what function models the energy expenditure previously calculated?

- Fill in a table with those values of times previously obtained using `Gilito2Times.java`. From the theoretical point of view, explain: what function models the execution time calculated before?

B. Delivery method

You should **commit and push** your project with your new `session3` package in your Github repository with the following content inside it:

- All the requested source files.
- The requested PDF document called `session3.pdf` with the corresponding activities.

Important:

- Make sure your Github course project is up to date after the delivery and that the last commit is strictly before the deadline. If not, the mark for this session will be 0 without any exception.
- Your mark will be eventually included in the `Marks.xlsx` file in your repository. Please, do not modify anything in that file to avoid conflicts between versions.

Deadlines:

- Group L.I-01 (Thursday): March 11, 2020 at 11:55pm.
- Group L.I-02 (Wednesday): March 10, 2020 at 11:55pm.
- Group L.I-03 (Tuesday): March 9, 2020 at 11:55pm.
- Group L.I-04 (Monday): March 15, 2020 at 11:55pm.