ALGORITHMICS - Bachelor of Software Engineering, 2022

LAB GUIDE. SESSION 3

GOALS:

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

1. Basic recursive models

You are provided with the following 10 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 "by subtraction" 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 $a>b^k$.

`SumVector1.java` solves the simple problem of adding the elements of a vector in three different ways, and `SumVector2.java` measures times of these three algorithms for different sizes of the problem.

`Fibonacci1.java` solves the problem of calculating the Fibonacci number of order `n` in four different ways, and `Fibonacci2.java` measures times of these four algorithms for different sizes of the problem.

TO DO:

A. Work to be done

- An `algstudent.s3` **package** in your course project. The content of the package should be:
    - All the basic recursive models 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.

- A **PDF document** (`session3_1.pdf`) using the course template. The activity of the document should be the following:
    - **Activity 1. Basic recursive models**.
        - A brief explanation for each of the given classes indicating how you calculated the complexity of that class.
        - A brief explanation for each of the 2 new classes indicating how you calculate the complexity to get the requested one.

B. Delivery method

You should include in your Java project a new `algstudent.s3` package with the following content inside it:

- All the requested source files for that package.
- The requested PDF document called `session3.pdf` with the corresponding activity.

Deadlines:

- The deadline is one day before the next lab session of your group.

2. Divide and Conquer problem

Polyomino

There are lots of documents and information about the Polyomino problem on the Internet. For example at https://en.wikipedia.org/wiki/Polyomino (English), or http://culturacientifica.com/2014/07/16/embaldosando-con-l-triominos-un-ejemplo-de-demostracion-por-induccion/ (Spanish). From the readings, it is deduced that for **trominoes** there are two cases: **L-tromino** (piece of 3 squares in L) and I-tromino (piece of 3 squares in line or I).

In the attached document (TrominoDescripcion.pdf), we analyze how to place L-trominoes in a board of side `n` (with `n` being power of 2, that is: 2, 4, 8, 16, ...). The algorithm explained is clearly "divide and conquer".

Well, let's choose as a problem to solve that of placing L-trominoes in a board of side `n` (obligatorily power of 2). This problem is only a particular case of a much more general problem set out above.

TO DO:

You are asked to design and implement an efficient algorithm to find the solution of the problem using the Java programming language.

To test the algorithm, you can put simple cases (via program arguments) for small values of `n` (n = 2, 4, 8, 16) and verify that the solution found is correct.

For example, with the class `Tromino.java`, for a board 8*8 with the initial empty cell in the position row 3 and column 5 (3, 5), the invocation could be:

```
java Tromino 8 3 5
```

You should also implement a `TrominoTimes.java` class that increases the size of the problem (`n` = 16, 32, 64, 128, 256, ... until HEAP is overflowed) and measures execution times for each problem size. To measure the times, the initial position of the empty cell can be set at a random place and the solution should not be written on the screen, it simply should measure the time it takes to calculate the solution.

N	*T Tromino*
16	...
32	...
64	...

128	…
256	…
…	…
Heap overflow	

- What should be the time complexity of the algorithm?
- Check if the time obtained in the previous section does or does not meet the theoretical complexity of the algorithm.

OPTIONAL WORK

If you have time and interest, you can work on one of the following lines:

- Try to parallelize the algorithm presented, in line with what has been seen in theory class. Measure times to see what time improvement is achieved.
- Try to solve the same problem proposed above using another algorithm different than the proposed "divide and conquer" approach. Measure times to see what time improvement/deterioration is achieved.
- Try to solve the L-tromino problem for side boards that have a size other than power of 2 (*deficient*, according to the terminology).
- Try to solve the problem for different schemes. For example, the domino (shapes of 2 squares) or what would be even more ambitious: the tetromino (shapes of 4 squares) or pentamino (shapes of 5 squares).

A. Work to be delivered

- An `algstudent.s32` **package** in your course project. The content of the package should be:
  - `Tromino.java`. Class that solves this problem.
  - `TrominoTimes.java`. Class that takes times.
- A **PDF document** (`session3_2.pdf`) using the course template. The activity of the document should be the following:
  - **Activity 1. Tromino Times.**
    - Create the requested table with times and answer the two questions indicated in the document.

You should include in your Java project a new `algstudent.s32` package with the following content inside it:

- All the requested source files for that package.
- The requested PDF document called `session3_2.pdf` with the corresponding activity.

Deadlines:

- The deadline is one day before the next lab session of your group.