Algorithms and Data Structures

Seminars Guide, 2023–24

José Luis Balcázar

Departament de Ciències de la Computació, UPC

Session 1, 11/01: Review of various notions, including recursion

Today:

- Short briefing
- Higher order
- Graphs and trees
- Recursion
- Evaluable exercise
- Exercises for the week

Short, partial briefing including Jutge courses.

- Most info, including evaluation, deferred to the theory class on Monday.
- Mandatory lab exercises to do *in class* most Thursdays and to be submitted through the Aula Virtual are part of the evaluation, including *today*'s.
- Attendance to the Lab Sessions is *mandatory*. Only Aula submissions made from the classroom will be considered. *I want to see you making the submission from here*.
- Jutge courses:
  - Specific, *mandatory*, non-public (invitations sent round):
    - Algorithms and Data Structures Winter 2023-24 ESCI.
    - This course will be expanded along the quarter with the addition of some new lists or new problems;
    - further, exams will refer to it.
  - Optional, open, *recommended*: Ad computationem vIA reGIA.
  - Optional on demand: ask me for invitations to the first-year courses of 2023-24 if you believe they can be helpful to you.

Higher order

1. Functions as arguments *or results* of other functions. Examples.
2. Use today: reading items with `pytokr`.

The problems in the first three lists of the Jutge course will give you practice with `pytokr` and with general higher order programming. They are also part of the current version of Programming and Algorithms 1.

Graphs and trees

Review and terminology:

- graphs, directed and undirected,
- free trees,
- rooted general trees,
- binary trees.

How are we representing binary trees in our programs?

And in our input data?

Recursion

The fourth list in the Jutge course contains problems to be solved *recursively*. You are likely to have iterative solutions of many of them. They are no longer what is required now! Work *only* recursively, *no loops*.

Goal for our first couple of weeks: proficiency with recursive programs with *more than one* recursive call per call.

Examples:

- reading binary trees,
- reading and traversals of binary trees (P98436).

Specific programming exercises:

1. Write first a program that reads binary trees from `stdin` as per P98436, but assuming just one integer per line, so that the standard function `input()` will keep providing you with the next element to be read.
2. Test it out, e. g. by asking for specific components of the tree just read within the Python interpreter.
3. Move on into a program that reads the tree without assuming anymore that the values come in different lines, using `pytokr()`, and test it out in the same way.
4. Then, complete the program with traversals and submit to P98436.

Evaluable exercise

After completing P98436 up to green light, solve P61120 and upload to the Aula a screen capture of your best attempt before leaving the room. *This exercise is to be solved individually.*

Exercises for the week

1. Generating (of course, *recursively*, no loops) all subsets of a given set of words (P18957).
2. Evaluating prefix expressions (P20006).
3. Size-based encoding of binary trees (X30150, midterm exam problem last year).
4. Reading and traversal of general trees (P66413). Feel free to choose to ignore the `C++` code suggestions and work instead fully in Python, if you prefer that option.
5. Once you have working Python solutions, consider trying `C++` (P57669).

We will get back to all this later on with alternative programs for tree traversals and with the *graph variant* of tree preorder traversal, namely, *depth-first search* (DFS).