# Algorithms

Alessandro Londero
Junior Consultant, ADDACTIS® Worldwide
alessandro.londero@addactis.com

ADDACTIS Worldwide
Risk & Insurance Solutions
+32 (0)2 526 13 10

Argentina - Brazil - Belgium - Chile - Colombia - Costa Rica - Ecuador - France - Germany - Guatemala - Italy - Luxembourg - Mexico
The Netherlands - Panama - Peru - Poland - Senegal - Singapore - Spain - Switzerland

# Introduction

ADDACTIS® Expertises' program gathers news from every business unit of ADDACTIS® Worldwide. Each consultant gives his point of view on a trend which is influencing his job.

Today, discover ADDACTIS® Worldwide Junior Consultant Alessandro Londero, thinking about algorithm.

## 1 Could you please define what is an algorithm? What are good practices in algorithmic?

**"A good documentation could save you time when coming back on some code that you wrote several times."**

An algorithm could be seen as a series of instructions in the aim to obtain one or more results. For example, if I ask you to find the Greatest Common Divisor ('gcd') of 72 and 63, you will test 2, then 3, then 4, and so on... to finally conclude that it's 9. You have unconsciously applied the following algorithm:

*« While I'm sufficiently small to be the gcd*

- *I check if I could be the gcd (if I divide 72 and 63)*
  - *If yes, I keep it in a corner of my memory;*
  - *If no, I go to the next one.*

*Once every number sufficiently small checked, the number in the corner of my memory is the GCD. »*

I will overall take some general advice from INRIA available at the end of the next page: *https://caml.inria.fr/resources/doc/guides/guidelines.en.html*

- Don't hesitate to comment your code.

A good documentation could save you time when coming back on some code that you wrote several times. Commenting on your code will explain to others your code while they read it. You might also have to reread it later for several reasons such as:

- To improve it;
- To validate it;
- To explain to someone how the algorithm works;
- To reuse it.

For all these reasons the code that you are writing must be as clear as possible, and if doubts remain on its intelligibility do not hesitate to comment on the lines that seem unclear.

A similar point is to name all the variables you are using as clearly as possible (for example, in our previous example a variable named « gcd_candidate » will be clearer than « gcdc »).

- Reread your program to check its readability.

It's the perfect time to:

- Rename variables to make the code clearer;
- Comment unclear lines;
- Check that the code could not be simplified.

## 2 Must an algorithm be always adjusted to be efficient? Can you give us an example?

Today most algorithms are already efficient but all new algorithms could be optimized. For example, let's take the search of a word inside a dictionary where words aren't sorted (a non-sorted table) we will have to look at every word until:

- We find the searched word;
- We scrolled through the whole dictionary.

However it is clear that word processing program couldn't work like that!

A good optimization for them is to use a "sorted" dictionary. It means to put the word in a predefined order (alphabetically for example). Now it's sufficient to take the word in the middle of the dictionary and compare it to the searched word, there are three options:

- The current word is the searched word ⟶ we have finished and the word is present;
- The current word is before the searched word ⟶ we remove all the words after the current word;
- The current word is after the searched word ⟶ we remove all the words before the current word.

We do this step until we find the word we looked for, or until we have excluded the whole dictionary.

The advantage of this algorithm is that if we haven't found the word we remove the half of the remaining dictionary at each step.

But sorting the dictionary to search only one word will take more time than searching directly inside the non-sorted dictionary.

The same kind of consideration could be done for the memory used by an algorithm.

Generally the optimization of an algorithm is always the research of a flimsy equilibrium between memory consumption and a number of simple operations to execute. One and the other being a factor that can improve or reduce the performance of an algorithm.

***"Generally the optimization of an algorithm is always the research of a flimsy equilibrium between memory consumption and a number of simple operations to execute."***

## 3 Why is the efficiency of an algorithm crucial in the Big Data context?

The Big Data is the management and analysis of a massive amount of data. The efficiency of the algorithm is there crucial, so that the time used to process the amount of data must be minimal.

If a decision must be quickly taken, the algorithm must have run quickly enough to give the result before it's too late.

For example stock exchange are now quoted live, so if the algorithm arrive a tenth second too late the quotation could have changed and the order (to buy/sell) given by the algorithm may have significant financial consequences.

To conclude we won't be able to improve some processing algorithm because by nature they must browse through the whole data, but what we can do now is to improve the processing capacity of these computers.

***"We won't be able to improve some processing algorithm because by nature they must browse through the whole data, but what we can do now is to improve the processing capacity of these computers."***


© 2017 ADDACTIS Worldwide – All rights reserved; any reproduction without written permission from ADDACTIS Worldwide is prohibited.

# ADDACTIS Worldwide

Risk & Insurance Solutions


www.addactis.com
contact@addactis.com
+32 (0)2 526 13 10

Argentina - Brazil - Belgium - Chile - Colombia - costa Rica - Ecuador - France - Germany - Guatemala - Italy - Luxembourg - Mexico - The Netherlands - Panama - Peru - Poland - Senegal - Singapore - Spain - Switzerland