Ministry of Higher Education and Scientific Research
Mohamed Khider University - Biskra
Faculty of Exact Sciences and Natural and Life Sciences
Department of Computer Science

2nd year LMD

Course of Algorithmic and Data Structures: ALGO 3

Translated and Revised by: **Pr. Laid KAHLOUL**
l.kahloul@univ-biskra.dz
(based on the french version proposed by **Pr. Abdelhamid DJEFFAL**)

College Year 2023/2024

Plan of the course

References

[1] D. Beauquier, J. Berstel, and P. Chrétienne. *Éléments d'algorithmique*. Masson, Version 6, 2005.

[2] T.H. Cormen, C.E. Leiserson, R.L. Rivest, T.H. Cormen, and T.H. Cormen. *Introduction à l'algorithmique*. Dunod, 1996.

[3] J. Courtin and I. Kowarski. *Initiation à l'algorithmique et aux structures de données*. Dunod, 1990.

[4] M.C. Gaudel, M. Soria, and C. Froidevaux. Types de données et algorithmes 2: Recherche, tri, algorithmes sur les graphes. 1987.

[5] Guillaume Poupard. *Algorithmique*. Ecole Nationale Supérieure des Techniques Avancées, 2000.

Chapter 1

Introduction

Using a computer to solve a problem requires a lot of preparation work. Having no capacity for invention, the computer can only execute the orders provided to it via a program. The latter must therefore be established in such a way as to consider all the eventualities (i.e., cases) of treatment.

Example: the Div(a,b) problem, don't forget the case b=0!

1.1 Solving a problem in computer science

Indeed, in computer science we have several steps to follow when solving a problem:

- step 1: **Problem definition:**
  This involves determining all available information and the form of the desired results.
- step 2: **Problem analysis:**
  It involves finding a way to move from data to results. In certain cases it may be necessary to carry out a theoretical study. The result of the analysis step is an algorithm. A first definition of an algorithm can be the following:
  "An algorithm is a finite sequence of instructions uniquely indicating the order in which a set of operations must be performed to solve all problems of a given type."

  Also note that there are problems for which we cannot find a solution and therefore it is impossible to give the corresponding algorithm.
- step 3: **Writing an algorithm** with an algorithmic description language:
  Once you find a way to move from data to results, you must be able to write a clear

and unambiguous solution. As it is impossible to do this in natural language, the existence of an algorithmic language is essential.

- step 4: **Translation of the algorithm** into a programming language:
  Steps 1, 2 and 3 are done without the use of a machine. If we want to make the algorithm concrete or practical, we would have to translate it into a programming language. We will then say that a program is an algorithm expressed in a programming language.

- step 5: **Program debug:**

  When you submit the program to the machine, the latter processes it in two stages:

  1. The machine (can) assist the correction of the program, this is called "lexical/syntax checking" in programming jargon.
  2. The machine translates the meaning expressed by the program into an executable version (called: object program, machine-code program, etc).

  If the results obtained are those expected, the development of the program ends. If we do not obtain results, we will say that there are logical errors. The program either gives no results, unexpected results or partial results. In this case, we must first review whether the algorithm has been translated well, or whether there has been a good analysis.

1.2 Algorithm concept

1.2.1 Definition

We can define an algorithm as follows:

Result of a logical approach to solving a problem. This is the result of the analysis.

Or:

A sequence of calculation steps that takes a set of values as input and produces a set of values as output.

1.2.2 Properties

We can state the following five properties that an algorithm must satisfy:

1. Generality: an algorithm must always be designed in such a way as to **consider all the eventualities of a treatment**.
2. Termination: An algorithm **must stop** after a finite time.
3. Fully Defined: all operations of an algorithm must be **unambiguously** defined (no ambiguity)
4. Repetitiveness: generally, an algorithm contains **several iterations**, i.e. actions that are repeated several times.
5. Efficiency: Ideally, an algorithm should be designed in such a way that it **runs in minimum time and consumes minimum resources**.

1.2.3 Examples

- GCD (Greatest Common Divisor) of two numbers u and v.
  - Naive algorithm: we successively test whether each integer is a common divisor.
  - Decomposition into prime numbers.
- Sorting algorithms
- Search algorithms
  - Search for a character string in a text (Word processing software).
  - Dictionary search.
  - ...etc.

1.2.4 Remark

Please note, some problems do not admit an exact and usable algorithmic solution. In this case, we use heuristic algorithms which provide approximate solutions.

1.3 A conventional algorithmic language

During this course, we will use a conventional algorithmic language to describe the different solutions to the problems. The following algorithm summarizes the general form of an algorithm and most of the declarations and instructions that can be used.