Course 1: Introduction in algorithm

1 Definition of algorithm

In general terms, an *algorithm* is a method of solving problems step by step. A problem is composed of input data and a expression that specifies existing relation between input data and problem’s *solution*. An algorithm also consists of processing needed for obtaining a solution using input data.

An algorithm is sequence of processings that are applied upon input data of a problem and allow obtaining in finite time the solution of that problem.

The term algorithm comes from the name of a Persian mathematician, al-Khowarizmi (al-Kwarizmi), who lived in the 9th century and who wrote a work on performing numerical calculations in an algebraic manner. The first algorithm is considered to be Euclid's algorithm (used to determine the greatest common divisor of two natural numbers). The term algorithm can be understood in a broad sense, not necessarily being related to solving a character problem scientifically, but only to describe in an orderly manner activities that consist of going through a sequence of steps (such as using a public telephone or an ATM).

In mathematics, there are a number of algorithms: that of solving the equation of the second degree, Eratosthenes' algorithm (for generating prime numbers smaller than a certain value), Horner's scheme (for determining the quotient and remainder of dividing a polynomial by a binomial), etc. .

The solution to the problem is obtained by executing the algorithm. The algorithm can be executed on a formal machine (in the design and analysis phase) or on a physical machine (computer) after it has been codified in a programming language. Unlike a program, which depends on a programming language, an algorithm is a mathematical entity that is independent of the machine on which it will be executed.

2 Course’s objective

The subject of the "Algorithmic" discipline is the study of algorithms from the perspective of their elaboration and analysis. Developing an algorithm requires:

- knowledge specific to the field where the problem to be solved comes from;
- general problem solving techniques;
- intuition and algorithmic thinking.

The analysis of algorithms involves checking their correctness and efficiency. An algorithm is considered correct if, by applying it to the data of the problem, it leads to its solution, and efficient if, through its execution, the result is obtained in a reasonable time interval. The analysis involves the application of demonstration techniques specific to mathematics.

Regardless of the complexity of an IT application, at its base are algorithms designed to solve the fundamental problems of the application. No matter how sophisticated the software technology used (interfaces, global structuring of the application) the efficiency of the application is essentially determined by the efficiency of the algorithms involved . A poorly designed algorithm cannot be "fixed" by programming artifices. For this reason, the acquisition of solid knowledge regarding the development and analysis of algorithms is a necessary condition in the formation of a good programmer.

3 Algorithm properties

An algorithm must have following properties:

Generality. An algorithm intended to solve a problem must allow obtaining the result for any input data and not only for particular input data.

Finitude. An algorithm must admit a finite description and each of the processes it contains must be able to be executed in finite time. Infinite structures cannot be processed by means of algorithms.

Rigor. The processing of the algorithm must be specified rigorously, without ambiguity. At any stage of the execution of the algorithm, it is necessary to know exactly what the next stage is that will be executed.

Efficiency. Algorithms can be effectively used only if they use computational resources in an acceptable volume. Computing resources mean the amount of memory and the time required for execution.

Examples.

*1.*Not every problem can be solved algorithmically. We consider a natural number n and the following two problems: (i) to construct the set of divisors of n; (ii) to construct the set of multiples of n. To solve the first problem, an algorithm can be easily developed, but for the second problem, an algorithm cannot be written as long as it is not known It is a criterion for stopping processing.

2.An algorithm must work for any given input. Let's consider the problem of ascending ordering of the sequence of values: (2, 1, 4, 3, 5).One way of ordering would be the following: the first element is compared with the second and if it is not in the right order, it is exchanged (in this way we get (1, 2, 4, 3, 5)); for the sequence thus transformed, the second element is compared with the third, and if it is not in the desired order, it is exchanged (at this stage the string remains unchanged); it continues the procedure up to the penultimate element is compared with the last one. In this way (1, 2, 3, 4, 5) is obtained.

Although the method described above allowed the ascending ordering of the sequence (2, 1, 4, 3, 5) it cannot be considered a sorting algorithm if it is applied to the sequence (3, 2, 1, 4, 5) leads to (2, 1, 3, 4, 5).

2. *An algorithm needs to stop. Consider the following sequence of processings:*

Step 1. Assign variable x value of 1;

Step 2. Increase value of x by 2;

Step 3. If x is equal to 100 then the process is stop, else it returns to Step 2.

It is easy to see that x will never take the value 100, so the processing sequence never ends. For this reason, it cannot be considered a correct algorithm.

3. *The processing in an algorithm must be unambiguous. We consider the following processing sequence:*

Step 1. Assign variable x value of 0;

Step 2. Either increase x by 1 or decrease x by 1;;

Step 3. If $x \in [-10, 10]$ resume from Step 2, otherwise the algorithm stops.

As long as a criterion is not established by which it is decided whether x increases or decreases, the above sequence cannot be considered an algorithm. Ambiguity can be avoided by using a rigorous algorithm description language. Consider that Step 2 is replaced by:

Step 2. A coin is tossed. If a head is obtained, x is increased by 1 and if a tail is obtained, x is decreased by 1;

In this case, the specification of processing is no longer ambiguous, even if different results are obtained at different executions. What can be said about the finiteness of this algorithm? If the head and page were obtained alternatively, the processing could be infinite. There is also the possibility to get heads (or tails) 11 times in a row, in which case the process would stop after 11 repetitions of step 2. As long as the chance that the algorithm ends is non-zero, the algorithm can be considered correct (in the case presented above it is a random algorithm).

4. *An algorithm must stop after a reasonable time interval. Let us consider that solving a problem involves the processing of n data and that the number of processings T (n) depends*

on n. We assume that time of execution of one processing is 10^{-3} s and problem has the size of n=100. If we a characterised algorithm T(n) = n then execution time will be $100 \times 10^{-3} = 10^{-1}$ secunde. If else another characterised algorithm is used, T(n) = 2^n then exeuction time will be about 10^{27}second or 10^{19} years.

4 Data

The processing performed within an algorithm is performed on some data. These are entities carrying information (relevant to the problem to be solved). We consider data as containers that contain information, the current value of a data being the information it contains at a given moment. Depending on the role played within the algorithm, the data can be constant (their value remains unchanged during the algorithm) or variables (their value can be changed).

From the point of view of the information it carries, the data can be:

- *Simple: that contain a single value* (a number, a truth value or a character).
- *Structured: contain more simple data between which exist a relational structure.*

If all the component data have the same nature then the structure is homogeneous, otherwise it is a heterogeneous structure.

The homogeneous structured data that will be used in the following are those intended to represent simple algebraic structures: finite set (set of distinct values for which the order in which they are kept is not important), ¸ finite string (set of not necessarily distinct values for which the order in which they are stored is important) and matrix (two-dimensional table of values).

To represent this data, we will use the table structure characterized by the fact that each component value can be specified by specifying one or more indexes. One-dimensional tables are most frequently used (for representing strings and sets) and two-dimensional ones (for the representation of matrices).

5 Types of processing

Similarly, data and processing can be classified into simple and structured. The simple processings are:

- *Assignment*. Allows assigning a value to a variable. The assigned value can be the result of evaluating an expression. An expression describes a calculation performed on some data and contains operands (specifies the data on which the calculations are performed) and operators (specifies the processing to be performed).

- *Transfer.* They allow taking the input data of the problem and providing the result.
- *Control.* Normally, the processing in the algorithm is carried out in the order in which they are specified. If you want to change the natural order, the execution control is transferred to a certain processing.

Processing structures are:

- *Sequential.* It is a sequence of processing (simple or structured). The execution of the sequential structure consists in the execution of the component processing in the order in which they are specified.
- *Conditional (alternative).* It allows the specification of situations in which, depending on the fulfillment or non-fulfillment of a condition, one or another processing is carried out. The condition is usually an expression whose result is a logical value (true or false). Such processing occurs, for example, in the evaluation of a function defined by:

$$f(x) = \begin{cases} -1 & \text{if} \quad x < 0 \\ 0 & \text{if} \quad x = 0 \\ 1 & \text{if} \quad x > 0 \end{cases}$$

- *Loop (repetitive).* It allows the modeling of situations when a processing must be repeated. It is characterized by the existence of a repeated processing and a stop (or continue) condition. Depending on the time when the condition is analyzed, there are previously conditioned repetitive processing (the condition is analyzed before performing the processing) and posterior conditioned processing (condition (the data is analyzed after processing).
  Such processing occurs, for example, in the calculation of a finite sum. Such processing occurs, for example, in the calculation of a finite amount $\sum_{i=1}^{n} 1/i^2$. In this case, the processing that is repeated is the gathering, and the stopping condition is the fact that all those n terms have been gathered.

6 Exercises.

1. Consider the following method of multiplying two integers x and y. Write x next to y (on the same line). Divide x by 2 and write the quotient below x (ignore the rest). Multiply y by 2 and write the product under y. The process continues thus building two columns of numbers. The calculations stop when the value 1 is obtained on the first column. Add all the values on the second column that correspond to odd values on the first column.
   Is the described method an algorithm? Is it correct (determine the product of the two

numbers)?
Indication. As a result of successive divisions by 2, the values on the first column decrease until a value equal to 1 is reached. Therefore, the processing is finished. The correctness of the processing derives from the fact that the method actually performs the base 2 conversion of the first number and the product is obtained by successively multiplying the second number by powers of his 2 and by summing those products that correspond to non-zero digits in the binary representation of the first number.

2. Propose an algorithm for determining the integer part of a real number. *Indication*. The sign of the number will be taken into account. If it is positive, the value of 1 is successively reduced until a value strictly lower than 1 is reached. The number of reductions performed indicates the value of the whole part. For negative numbers, 1 is successively added until a value greater than or equal to 0 is obtained.

3. We consider that a seller only has coins of 2 units and 5 units. Propose an algorithm to determine how to pay a change (the change is at least 4). *Indication*. If the change is an even number then the seller can only give coins of 2. If the number is odd the seller can give a coin of 5 and what remains (an even number) in coins of 2.

4. We consider the following two problems:
   (a) A housewife has made a series of purchases and has two bags at her disposal. The problem is to distribute the purchases in the two bags so that the difference between the weights of the two bags is as small as possible.
   (b) Consider two storage devices (for example magnetic disks) and a set of files whose total size does not exceed the global capacity of the two disks. An attempt is made to distribute the files on the two disks so that the difference between the remaining free spaces is as small as possible.

   Is there any connection between the two problems? Find solutions.

   Indication. Both problems can be formalized as follows: a set of numbers is considered positive, A = {a1, a2, . . . , a_n} and it is required to determine two disjoint subsets B and C

   thus $B \cup C = A$ *and* $|\sum_{a \in B} a - \sum_{a \in C} a|$ *is minimal*. The simplest method is that of "brute force" by which all pairs of subsets (B, C) are generated and the one that minimizes the specified difference is chosen. The number of distinct partitions is $2^{n-1} - 1$ if n is odd and $2^{n-1} - 1 + C_n^{\frac{n}{2}}/2$ if n is even. For large n, the number of partitions to be tested becomes large (for n = 10 it is 637 and for n = 100 it is 10^{29}. A more efficient method should be found..