INSA de Lyon
IMESI
**CS1**

# A week of algorithms from easy to easy

Final Version

Thomas Grenier

Lyon, le October 10, 2012

# Contents

# Introduction

The following exercises can be answer both using pseudo-code or *C++*. I recommend to do twice the exercises, one time for each language.

For each exercise, you have to

1. Write the algorithm
2. Give an example of execution (i.e. with a 5 elements array)
3. Prove the correctness of your algorithm
4. Give the order of grow and the asymptotic notation of the running time
5. Give the order of grow and the asymptotic notation of the memory used

Also, understand and apply this French sentence: *"C'est en forgeant qu'on devient forgeron"*.

# 1 How to work?

Create and write an algorithm is the most sensible task of the exercises. If you make a mistake your whole answers will be wrong. Then, write a algorithm is also the most difficult task. It is possible that you can not write it, or can solve only one part of the problem. This section will give you the basics of understanding, testing and building algorithm. I think that the use of exercises and exercises is the only solution when student can not create alone algorithm because of lack of knowledge, computer science practice, belief or simply do not have a 'good' method of work. Remember one thing: algorithms are not written in one shot but progressively, even for the craftiest programmer, except for basic and *cooking receipt* algorithms.

As there are so many algorithms, it is not possible to learn all of them. But having some cooking-receipt or global understanding will help you to solve all kind of problems. The only way to understand the cooking receipt is to train yourself at least one time on each kind of problem: sorting, algebra number, matrix manipulations, search, ... and also know the most important families of cooking-receipt as: linear programming, hashing table, divide and conquer, ...

There are many resources dealing about such knowledge and explanations. The books of T. Cormen (Introduction to Algorithm) and the one of C. H. Papadimitriou (Algorithms) are two essential readings.

These books have a good pedagogic approach about algorithm: algorithm solution are described explaining progressively the ideas (generally staring from the most inner one). In fact, just giving the algorithm solution of a problem without explanations is just a tool helping you to verify that your algorithm is quite close to the proposed one. As there is generally not only one algorithm that solve correctly (and optimally!) a given problem, having the algorithm solution:

- won't help you how to create another algorithm for a similar problem, because you do not have understood the cooking receipt and so you are not able to use it properly,
- needs that you also understand the processing way of the proposed algorithm and then you have to check if yours do the same. Some times the approaches used are drastically different... This a huge work but very pedagogic!

## 1.1 Prerequiste

**Unknown Code** Analyze the following code. The function CREATE$(\mathrm{B}, n)$ create (and allocate) a new array $B$ of the specified size $n$.

```
BADNAME(A, x, index)
    ▷
1   CREATE(B, Length[A]+1)
    ▷
2   for i ← 1 to index
3       do B[i] ← A[i]
    ▷
4   for i ← index to Length[A]
5       do B[i+1] ← A[i]
6   B[index] ← x
7   DESTROY(A)
8   A ← B
```

✍ Using a 5 elements array, give an example of use of the algorithm and detail the content of the array at each iteration

✍ Give a proper name to this algorithm.

✍ Complete the 3 missing comments

✍ Give the asymptotic running time of the algorithm

# 2 Playing with Variables

## 2.1 Monday Morning

**Swapping 2 variables** Write the algorithm SWAP(A,B) that swaps the content of the two variables $a$ and $b$.

**Return the square value** Write the algorithm SQUARE$(x)$ that returns the square value of $x$.

**Polynomial** Write the algorithm POLYNOMIAL$(a,b,c,x)$ that returns the value of $a.x^2+b.x+c$.

**Average2** Write the algorithm AVERAGE$(a,b)$ that returns the average value of $a$ and $b$.

**Average4** Write the algorithm AVERAGE$(a,b,c,d)$ that returns the average value of $a$, $b$, $c$ and $d$.

**Distance2** Write the algorithm AVERAGE$(a,b)$ that returns the distance between of $a$ and $b$.

**Distance2D** Write the algorithm AVERAGE$(x,y)$ that returns the distance of the point $P(x,y)$ to the origin $O(0,0)$.

**Distance2D2P** Write the algorithm AVERAGE$(x_1,y_1,x_2,y_2)$ that returns the distance between the point $P1(x_1,y_1)$ and $P2(x_2,y_2)$.

**Distance3D** Write the algorithm AVERAGE$(x,y,z)$ that returns the distance of the point $P(x,y,z)$ to the origin $O(0,0,0)$.

**Distance2D2P** Write the algorithm AVERAGE$(x_1, y_1, x_2, y_2)$ that returns the distance between the point $P1(x_1, y_1)$ and $P2(x_2, y_2)$.

**Suite** Write the algorithm SUITE$(a, b, u_t)$ that returns the value $u_{t+1} = a.u_t + b$.

**Modulo** Write the algorithm MODULO$(x, y)$ that returns the modulo of $x$ by $y$.
Remind that the modulo operator is % in *C++* and *mod* for pseudo code. Example 10 % 5 will give 2 as 10 *mod* 5.

**Even or odd?** Write the algorithm EVEN$(x)$ that returns *true* if $x$ is even.
You can use the modulo operator or the integer results of the division $x$ by 2. This last thing is noted $\lfloor x/y \rfloor$ for the floor value of the integer division of $x/y$ and $\lceil x/y \rceil$ for the ceil value. As example $\lfloor 9/2 \rfloor \Rightarrow 4$ and $\lceil 9/2 \rceil \Rightarrow 5$ and $x/y \Rightarrow 4.5$.
Using *C++* language, $\lfloor x/y \rfloor$ is $floor(x/y)$, and $\lceil x/y \rceil$ is $ceil(x/y)$

**Pow** Write the algorithm POW$(x, n)$ that returns the value of $x^n$.

**Factorial** Write the algorithm FACTORIAL$(x)$ that computes $x!$ the factorial of $x$ a positive integer value.

**Binomial Coefficient with Factorial** Write the algorithm BINCOEF$(n, k)$ that computes the binomial coefficient $\binom{n}{k}$ using the following formula:

$$\binom{n}{k} = \frac{n!}{k!\,(n-k)!} \quad \text{for } 0 \leq k \leq n. \tag{1}$$

**Binomial Coefficient, recursive formula** Write the algorithm BINCOEFRECURSIVE$(n, k)$ that computes the binomial coefficient using the following formula:

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k} \quad \text{for all integers } n, k > 0, \tag{2}$$

with initial values

$$\begin{array}{rcll} \binom{n}{0} & = & 1 & \text{for all integers } n \geq 0 \\ \binom{0}{k} & = & 0 & \text{for all integers } k > 0 \end{array} \tag{3}$$

**Pascal Triangle** Write the algorithm PASCALTRIANGLE$(l)$ that prints the line $l$ of the Pascal triangle. A given line $l$ of the Pascal triangle gives the coefficients of the expansion of $(x+y)^l$.

**Printing or displaying values**

1. Using a **for** loop, write the algorithm COUNT1$(n)$ that prints the values from 1 to $n$.
2. Write the algorithm COUNT2$(n)$ that prints the values from 0 to $n-1$.
3. Write the algorithm COUNT3$(n)$ that prints the values from $n$ to 1.
4. Write the algorithm COUNT4$(n)$ that prints the values from $2n$ to 0.
5. Write the algorithm COUNT5$(n)$ that prints only the odd values from 1 to $n$.
6. Write the algorithm COUNT6$(n)$ that prints only the even values from $4n$ to 1.
7. Write the algorithm COUNT7$(n)$ that prints the squared values of $i$ such as $i = 1 \ldots n$.
8. Write the algorithm COUNT8$(n)$ that prints first the values of $1 \ldots n$ and then $n \ldots 1$.

## 2.2 Arrays, a good afternoon

**Playing with one array**

1. Write the algorithm ZEROFILL$(A)$ that initializes all values of $A$ to zero.
2. Write the algorithm INIT2KEY$(A, key)$ that initializes all values of $A$ to the value $key$.
3. Write the algorithm INIT2INDEX$(A, index)$ that initializes all values of $A$ to the value of $A[index]$.
4. Write the algorithm INITLINEARASCENDING$(A)$ that initializes the elements of $A$ to their index value.
   i.e. $A[1] \leftarrow 1, A[2] \leftarrow 2, \ldots A[n] \leftarrow n$, with $n$ the length of $A$.
5. Write the algorithm INITLINEARDESCENDING$(A)$ that initializes the elements of $A$ such that:
   $A[1] \leftarrow n, A[2] \leftarrow n-1, \ldots A[n] \leftarrow 1$, with $n$ the length of $A$.
6. Write the algorithm INITNO3$(A)$ that initializes the elements of $A$ to the index value excepted if the index can be divided by 3. In this case, the value of the element is set to 3.
7. Write the algorithm INITLONG$(A)$ that initializes the elements of $A$ in ascending ordered values such that there is one value equal to 1, two values equal to 2 and so on until the end of the array $A$.
   Example, with $A$ a five-elements array: $A = \{1,2,2,3,3\}$
8. Write the algorithm INITFIBONACCI$(A)$ that initializes the elements of $A$ such that:
   $A[1] \leftarrow 1, A[2] \leftarrow 1, A[3] \leftarrow A[2] + A[1], \ldots A[n] \leftarrow A[n-1] + A[n-2]$, with $n$ the length of $A$.

**Average** Write the algorithm AVERAGE$(A)$ that returns the average value of the elements of $A$.

**DistanceND** Write the algorithm DISTANCEND$(X)$ that returns the distance value of the n-dimensional point $X$ to the origin.

**Insertion of an element** Write the algorithm INSERTAFTER$(A, x, i)$ that returns a new array having the value $x$ inserted in $A$ after the index $i$. Example: with $A = \{1,2,4,5\}$ then INSERTAFTER$(A,3,2)$ returns $\{1,2,3,4,5\}$.

**Playing with two arrays**

1. Write the algorithm COPY$(A)$ that returns B initialized with the same content of $A$.
2. Write the algorithm INSERTAFTER$(A, B, i)$ that returns a new array consisting of the values of $A$ until the index $i$, then the values of $B$ and the values of $A$ from $i+1$ to the end. The arrays have different sizes. Example: using $A = \{1,2,5,6\}$ and $B = \{3,4\}$, the function INSERTAFTER$(A,B,3)$ returns $\{1,2,3,4,5,6\}$.
3. Write the algorithm SCALARPRODUCT$(X,Y)$ that returns the scalar product of $X$ by $Y$.
4. Write the algorithm WEIGHTEDAVERAGE$(P,X)$ that returns the weighted average of the values of $X$ by the weights of $P$. The arrays have the same length $n$. This algorithm computes:

$$\frac{1}{\sum_{i=1}^{n} P[i]} \cdot \sum_{i=1}^{n} X[i].P[i] \tag{4}$$

5. Coming soon.

**DistanceND2P** Write the algorithm DISTANCEND$(X,Y)$ that returns the distance value between the n-dimensional points $X$ and $Y$.

# 3 Data Structures

## 3.1 Struc

## 3.2 Lists

## 3.3 Stacks and Queues

## 3.4 Trees

## 3.5 Graph