# Computer Science - Data Structures

## Topic: Algorithms Overview

### Approach: Encoding from Mathematical Notation

Miguel Angel Avila Torres - 

## Introduction

This project will be realized aiming to develop the skill of transforming some proper mathematical formula into an algorithm, by recursive, successive or loop definition.

### *Motivation*

Is useful to develop the skill of transcribing any formula into a piece of code, there are many applications.

When you are trying to solve a really hard problem you usually will use models, notations and different mathematical tools that allows to simplify the problem, finding some easy way to manage it properly.

Also, when some problem over a known science field is approached, one frequently recurs to use the related knowledge. It's sure that some or few answers to different parts of the problem in question will be found in mathematical notation (if there are no algorithms in which one can based for) and then this subject obtains fundamental importance.

## Reasoning

For the purpose of built an algorithm from its mathematical form one must see reasonable elements that composes the algorithm. These are:

- ⋆ Sum $\sum_{i=1}^{n}$ and multiplication $\prod_{i=1}^{n}$ signs becomes "for" loops from $i$ to $n$.

- ⋆ Nested sums and multiplications are turned into nested "for" loops.

- ⋆ Multiplications $\prod_{i=1}^{n} a^{b_i}$ over the same term are a sum over the exponents of that term $a^{\sum_{i=1}^{n} b_i}$.
  The proof may be done based on some algebra rules.[1]

- ⋆ Traversals trough a matrix $(a_{ij})_{n\times m}$ are in fact nested loops, which according with what you are trying to do it may have certain form

  For example, the sum of all matrix entries has the mathematical notation:

$$sum(A_{n\times m}) = \sum_{i=1}^{n}\sum_{j=1}^{m}(a_{ij})$$

- ⋆ Constant terms are supposed to be declared constants too in the programming language that you are going to use. It is not only a perspective question, it is for visualizing and reading better code, "Quality Code".[2]

- ⋆ From a mathematical formula the limitations of it are extremely important because those can guide us to avoid some errors on runtime; example, for this given partial derivative:

$$\frac{\partial f(x,y)}{\partial x} = \frac{2x\sqrt[n]{x^2+y^2}}{(n+1)(2ayx+by)e^{ayx^2+byx+n}}$$

  In the ordinate $(x,y) = (0,0)$ we have an in-determination zero over zero, and it will make the program fail. So, what we have to do is restrict the domain of $f$ in the program.

  The process of restricting/verifying some input values is often called "validation" by some engineers.

- ⋆ As last, remember when you are working with matrices, in general, the whole input length is equal to $n \times m = w$, therefore the complexity analysis for an algorithm that goes through a entire matrix should be written as $O(f(w)) = O(w)$.

  If you have noticed, it is because we are not looping the input two times its value is to say $(w^2)$, we are iterating it only once $(w)$, in a ratio $1:1$ element/processing. Do not lie to yourself!

[1] It may be used with some restrictions, is useful to indicate the result, moreover not for computing it. Will be formidable if you are able to explain the reason.

[2] Remember, is important to develop useful things but also maintainable ones, which are easy to read, easy to debug

## ALGORITHMS

From the algorithms bellow, is needed a conversion from its mathematical formula to code[3] [4]

† *Recursive Fibonacci* $[\star\star\star]$*:*

$$Fib(0) = 0, \tag{1}$$
$$Fib(1) = 1, \tag{2}$$
$$Fib(n) = Fib(n-1) + Fib(n-2) \tag{3}$$

† *Linear Fibonacci* $[\star\star\star]$*:*

$$Fib : 0,1,2,3,5,8,13,\ldots \tag{4}$$

Let $a,b$ be two contiguous numbers in the succession, its sum gives the next value on it.

† *Factorial* $[\star]$*:*

$$Fact(n) = \prod_{i=2}^{n} i \iff n \neq 0 \wedge n \in \mathbb{Z}^+ \tag{5}$$
$$Fact(1) = 1 \tag{6}$$
$$Fact(0) = 1 \tag{7}$$

† *Matrix' diagonals sum*$[\star]$*:*

$$D_s(A_{n\times n}) = \sum_{i=1}^{n} \left((a_{ii}) + (a_{i(n-i+1)})\right) \tag{8}$$

† *Regular parallelepiped' sum* $[\star]$*:*

$$P_{lmn}, K_{lmn} \in \mathfrak{M}_{l\times m\times n} \subset \mathbb{R}^n \tag{9}$$
$$P_{lmn} + K_{lmn} := S_{lmn} \tag{10}$$
$$:= (s_{lmn}) = (p_{lmn}) + (k_{lmn}) \tag{11}$$

$P,K$ are cubes in $\mathfrak{M}$, the space of all the cubes of size $l \times n \times m$ size. A cube is a "vector" of matrices.

† *External lateral projections of a parallelepiped* $[\star^\star]$*:*

$$Lp(K_{lmn}, \text{left}) := A_{lm} \tag{12}$$
$$:= k_{lmc};\ \text{c=n} \tag{13}$$
$$Lp(K_{lmn}, \text{right}) := A_{lm} \tag{14}$$
$$:= k_{lmc};\ \text{c=1} \tag{15}$$

Here is showed the external lateral projection of a parallelepiped on its left and right sides, you must realize the algorithms plus up and down projections.

† *Regular parallelepiped's diagonals' sum* $[\star]$*:*

$$DP_s(K_{nnn}) = \sum_{i=1}^{n} (k_{iii} + k_{i(n-i+1)(n-i+1)} + k_{i(n-i+1)i} + k_{ii(n-i+1)}) \tag{16}$$

† *Vertical sum in a cube* $[\star]$*:*

$$VC_s(K_{lmn}) = A_{mn} := a_{ij} = \sum_{l'=1}^{l} k_{l'ij} \tag{17}$$

† *Symmetric Matrix Partition* $[\star\star\star]$*:*

$$k,l \in \mathbb{Z}^+; \tag{18}$$
$$mod(m,k), mod(n,l) = 0 \tag{19}$$
$$P_t(A_{mn},k,l) = B := \begin{bmatrix} A_{vw} & \ldots & A_{v'w} \\ \vdots & \ddots & \vdots \\ A_{vw'} & \ldots & A_{v'w'} \end{bmatrix} \tag{20}$$

Example:

$$A := \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

$$P_t(A,2,2) := \begin{bmatrix} A_{11\to 22} & A_{13\to 14} \\ A_{31\to 42} & A_{33\to 44} \end{bmatrix}$$

$$A_{11\to 22} := \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}; A_{13\to 14} := \begin{bmatrix} a_{13} & a_{14} \\ a_{23} & a_{24} \end{bmatrix}$$
$$A_{31\to 42} := \begin{bmatrix} a_{31} & a_{32} \\ a_{41} & a_{42} \end{bmatrix}; A_{33\to 44} := \begin{bmatrix} a_{33} & a_{34} \\ a_{43} & a_{44} \end{bmatrix}$$

Note that $P_t$ is a function from matrices to "matrices of matrices".

† *Internal diagonal product in symmetric attached systems of symmetric linear equations* $[\star]$*:*

$$D_p(K_{nnn}) = \prod_{i=1}^{n} (k_{iii} \times k_{i(n-i+1)(n-i+1)} \times k_{i(n-i+1)i} \times k_{ii(n-i+1)}) \tag{21}$$

† *Differential vector (average measure)* $[\star^\star]$*:* Let $A_{mn}$ be a matrix over $\mathscr{H}_{m\times n}$ and $\langle \mathbf{e}_k \rangle$ be the standard basis over $\mathscr{H}^n$ then:

$$|\psi\rangle \in \mathscr{H}^n; \langle \mathbf{e}_{1\leq i\leq n} \rangle \subset \mathscr{H}^n \tag{22}$$
$$\nabla : \mathbb{K}_{m\times n} \subseteq (\mathscr{H}^n) \to \mathscr{H} \tag{23}$$
$$A_{mn} \mapsto \nabla A_{mn} \tag{24}$$
$$\nabla A_{mn} = m^{-1} \left( \sum_{i=1}^{m} \left\langle \frac{\partial A_i}{\partial x_1}, \frac{\partial A_i}{\partial x_2}, \ldots, \frac{\partial A_i}{\partial x_n} \right\rangle \right)^T \tag{25}$$
$$\nabla A_{mn} = m^{-1} \left( \sum_{i=1}^{n} \left( \sum_{j=1}^{m} (a_{ji}) \langle \mathbf{e}_i \rangle \right) \right)^T = |\psi\rangle \tag{26}$$

Where $|\psi\rangle$ is a vector in $\mathscr{H}^n$ the complex space. Note that the mapping from $\mathscr{H}_{m\times n}$ to $\mathscr{H}^n$ is closed because the source space is the same one in which lies the result.

† *Semi-symmetric Outer Product* $[\star^{\star^\star}]$*:*

$$|\psi\rangle = [\psi_1, \psi_2, \psi_3, \cdots, \psi_n]^T; \langle\psi| = \left(|\psi\rangle^T\right)^* \tag{27}$$
$$\Delta : \mathscr{H}^n \to \mathscr{H}^n \tag{28}$$
$$|\psi\rangle \mapsto \Delta|\psi\rangle \tag{29}$$
$$\Delta|\psi\rangle = \nabla \left( (|\psi\rangle \langle\psi|)^{-1} \right) \tag{30}$$

(Note 1: Use the previous definition of $\nabla$)
(Note 2: Those operations were defined here arbitrarily)

[3] The following algorithms has matched its difficulty according to its stars number, being $\star^\star$ the highest and $\star$ the lowest

[4] Sums and multiplications avoids stack-overflow exceptions over recursive algorithms.

## Algorithms Design, Mathematical Notation, Asymptotic Complexity Analysis

The exercises bellow, are intended to be developed by the student, those must be written in its corresponding mathematical notation, transcript to an algorithm and analyzed using Big O notation.[5]

⋆⋆⋆⋆⋆ To define a linear function with an attached discrete sequence of values[6] that gives an image of the function.

⋆⋆ Unit n-sphere around the solution coordinate of a linear equations system of size $n \times n$.

⋆⋆ Orthogonality demonstration of two vectors in $\mathbb{R}^n$.

(Note: Remember, some demonstrations ends in a contradiction, however the conclusion we obtain is that the initial statement was false)

Now, for the unit sphere we would get the following elements:

$$\sum_{i=1}^{n}(x_i - h_i)^2 = 1 \tag{31}$$

where $h_i$ are the coordinates of the center. Now, what this problems ask you is:

> "Find the solution of an $n \times n$ augmented matrix and with such values build the unit sphere surrounding them. It means that by solving a given linear system you have fould the values for all $h_i$ and you just have to show the concrete formula"

A concrete case is:

$$\begin{cases} 2x_1 & 1x_2 & -1x_3 & = & 8 \\ -3x_1 & -1x_2 & 2x_3 & = & -11 \\ -2x_1 & 1x_2 & 2x_3 & = & -3 \end{cases}$$

$$\left[\begin{array}{ccc|c} 2 & 1 & -1 & 8 \\ -3 & -1 & 2 & -11 \\ -2 & 1 & 2 & -3 \end{array}\right] \rightarrow \left[\begin{array}{ccc|c} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & -1 \end{array}\right]$$

$$\Longrightarrow \underbrace{(x_1-2)^2 + (x_2-3)^2 + (x_3+1)^2 = 1}_{\text{This is the result and is displayed in console}}$$

[5] At least 3 of the exercises must be converted to an algorithm

[6] Both of them defined by the user; a linear function (On the scalar field) has the form: $f(x) = ax + b$

## Notes

⋆ You are not allowed to use any native class from any std library present in the program language that you are going to use. Given the case, your real qualification will be recalculated deducting 20% from the original one.

⋆ All the mentioned algorithms must have attached a complexity worst case analysis made in Big O notation (not only the last ones, sometimes is good to read all).

⋆ Questions with the instructor (me). **Do not be shy**, if you do not know how to do something then ask. But first try to make a brief supposition about how to solve the problem.

## Support Material


[1] D. Poole, Linear Algebra: A Modern Introduction. Cengage Learning, 2014.

[2] S. Lang, Introduction to Linear Algebra. Springer, 1997.

[3] J. Hefferon, Linear Algebra. Orthogonal Publishing L3c, 2017.

[4] K. Hoffman and R. Kunze Alden, Linear Algebra. Prentice Hall, 1971.

[5] S. Grossman I., Elementary Linear Algebra. Brooks/Cole Publishing Company, 1994.

[6] S. Boyd and L. Vandenberghe, Introduction to Applied Linear Algebra. Cambridge University Press, 2018.