# Combinatorial Optimization

Written by HADIOUCHE Azouaou.

## Disclaimer

That typed course follows the course Pr.YAGOUNI.

> To seperate the contents of the course to actual additions or out of context information, a black band will be added by its side like the one englobing this comment.

## Contents

Chapter 1

# Introduction

**Definition 1.1 (Information):** *An* information *is the set of data, formulas, rules, results… etc., which results in knowledge and should be interesting, logical, complete, premise… etc.*

An information goes through the a treatment, for example if we take a rectangle with width $W = 50m$ and height $H = 20m$, to calculate the surface with the formula $S = W \cdot H$ and using it we get $S = 1000m^2$, and each step is information of different types

| | | |
|---|---|---|
| $W = 50m$<br>$H = 20m$ | $S = W \cdot H$ | $S = 1000m^2$ |
| info. to treat | treating info. | treated info. |

**Definition 1.2 (Algorithm):** *An* algorithm *is a finite sequence of steps that are: elementary, complementary, logical, chronological and non-ambiguous, that transforms a string representing the data to a string representing a solution, a set of solutions or the non-existence of solutions.*

Each algorithm has one or many stop condition, taking for example a search algorithm on a list to find the minimum value, the stop condition would be that we passed through all the items of the list, or if we look for some value val, its either passing through all the items of the list or finding the value val.

For a given problem $(\mathcal{P})$, we make an algorithm $\mathcal{A}$ to solve $(\mathcal{P})$, that is an algorithm given any instance $I$ of $(\mathcal{P})$, it returns a solution. The algorithm has the multiple components which include:

- Constants & Variables:
  - Constant: a value that stays unchanged throughout the treatment.
  - Variable: a memory case that can save a value and admits a change through the treatment.
- Affectations (:=): changing the value of a variable.
- Tests:
  - Simple Test (if block): a condition only containing the if block

```
if (condition)
  then <action-1>;
       <action-2>;
       ...
       <action-n>
end
```

  - Complete Test (if-else block): a condition containing both if and else

```
if (condition)
  then <action1>;
  else <action2>;
end
```

  - Complete Composed Test (if-else if-else block): a condition containing if, else, and a set of else if statements

```
if (condition1)
  then <action1>;
else
  if (condition2)
    then <action2>;
  else
    <action3>;
  end
end
```

- Loops:
  - For Loop

    ```
    for i = 1 to n do
      <action>;
    end
    ```