1 Introduction

1.1 Algorithm's Complexity

Definition

It's a study of all the resources needed to execute the program. This can include time, memory, and bandwidth (if the program sends requests over a network). However, the main focus is often on time, because while memory and bandwidth can be upgraded with more advanced hardware, time cannot be bought.

Example :

Algorithm Sum of First N Integers

```
Var
n, sum, i integer;
Begin
sum ← 0;
i ← 1;
print('Input Integer N : ')
Read(n);
while i <= n do
    sum ← sum + i;
    i ← i + 1;
end while
print('Sum is ' ,sum);
End
```

Some Terminology

- **Frequency of Execution (F_r)**: The number of times an instruction is executed.
- **Execution Time of Basic Instructions (Δt)**: We assume that all basic instructions (such as print, read, assignment, arithmetic operations, etc.) have the same execution time, denoted as Δt.
- **Function ($f(n)$)**: Represents the total frequency of the program's instructions in relation to the data size n.
- **Execution Time Function ($T(n)$)**: The execution time function in relation to the data size n.

Time Complexity :

we have $f(n) = \sum fr$ since $f(n)$ is sum of frequency of execution (fr) we need to figure out the fr of each instruction and sum them :

sum ← 0	$fr = 1$ (one affectation)
i ← 1	$fr = 1$ (one affectation)
print('Input Integer N : ')	$fr = 1$ (one print)
Read(n)	$fr = 1$ (one read)
while i <= n **do**	$fr = n + 1$ (check the while condition n+1 times)
sum ← sum + i	$fr = 2n$ (one affectation and one arithmetic operation + (2) inside a while that loops n times (2n))
i ← i + 1	$fr = 2n$ (one affectation and one arithmetic operation + (2) inside a while that loops n times (2n))
print('Sum is ' ,sum)	$fr = 1$ (one print)

$$\begin{aligned} f(n) &= \sum fr \\ &= 1 + 1 + 1 + 1 + (n + 1) + 2n + 2n + 1 \\ &= 5n + 6 \\ &= \boxed{5n + 6} \end{aligned}$$

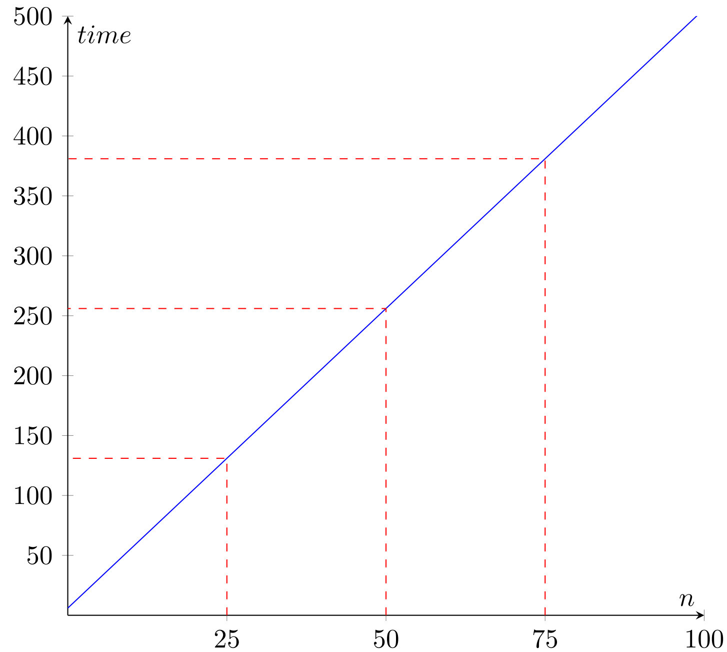
now that we have the complexity function $f(n)$ we need to find the $T(n)$ we have $T(n) = f(n) \times \Delta t$

$$\begin{aligned} T(n) &= f(n) \times \Delta t \\ &= (5n + 6) \times \Delta t \\ &= \underbrace{5\Delta t}_{a} n + \underbrace{\Delta t 6}_{b} \\ &= \boxed{an + b} \end{aligned}$$

Note

- **Exact Theoritical Complexity** : It's $T(n)$
- **Approximate Theoretical Complexity (Asymptotic)**: It approximates $T(n)$ by omitting all constants and taking the term with the highest growth rate.

In this example the exact theoritical complexity is $T(n) = an + b$ and its approximate theoritical complexity is $an + b \sim O(n)$ we notice that its time complexity is linear



Space Complexity :

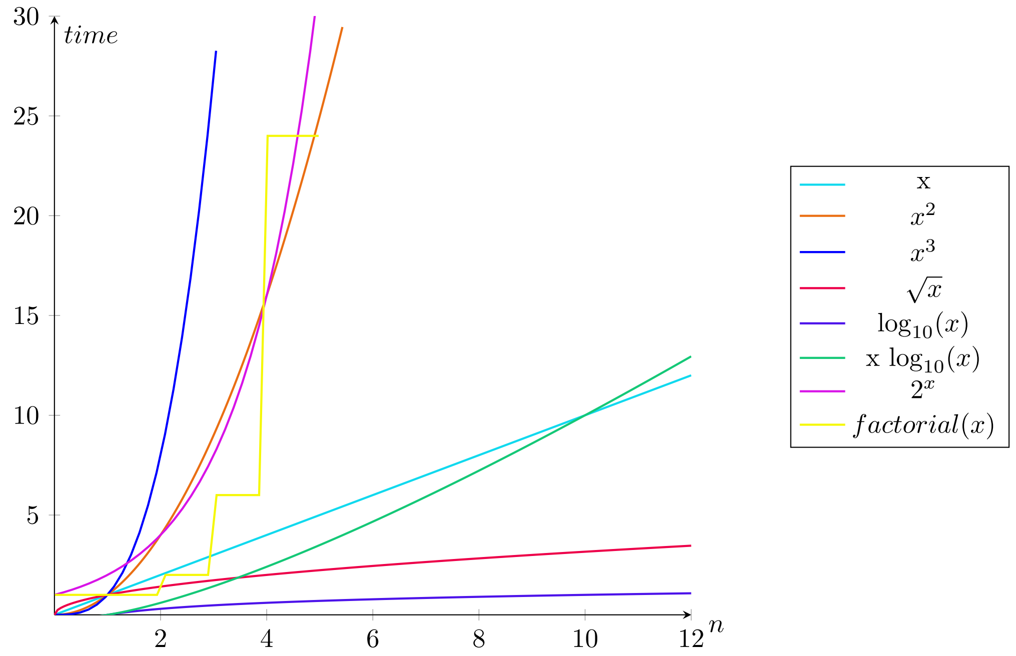
Before a program gets executed all instructions are loaded in memory and at execution time all variables are also stored in the memory so we need to find the number of instruction and number of variables and sum them let's suppose they all take same size 1 Byte

We have in the algorithm a total of 3 variables and 8 instructions in 3+8 = 11 Byte it's constant $\sim O(1)$

2 Complexity Analysis Basics

Some Terminology

- **Theoritical Complexity**: It's theoritical complexity calculated through $f(n)$ and $T(n)$ by supposing that each basic instruction has same execution time Δt
- **Experimental Complexity**: It's real life complexity calculated by executing a program in a machine and using a clock to get the exact time



Example :

suppose we have $f(n) = 2^n \quad , \quad \Delta t = 10^{-6}s$

$$T(n) = f(n) \times \Delta t$$
$$T(60) = 2^{60} \times 10^{-6}$$
$$= \boxed{36559 years}$$

Landau Notation

- **Big O** : O upper bound
- **Big Omega**: Ω lower bound
- **Big Theta** : Θ average

Landau Notation Mathematical Definition

$f(n) = \Omega(g(n)) \quad \exists c > 0 \,, \quad c.g(n) \leq f(n) \quad , \forall n \geq n_0$

$f(n) = O(g(n)) \quad \exists c > 0 \,, \quad c.g(n) \geq f(n) \quad , \forall n \geq n_0$

$f(n) = \Theta(g(n)) \quad \exists c_1 > 0 \,, \exists c_2 > 0 \,, \quad c_1.g(n) \leq f(n) \leq c_2.g(n) \quad , \forall n \geq n_0$

Example :

$f(n) = 2n + 5 => O(n)$

$$\begin{gathered} f(n) \leq c.g(n) \\ 2n + 5 \leq c.n \\ 2 + \frac{5}{n} \leq c \\ \frac{5}{n} \leq 1 \qquad \forall n \geq 5 \qquad \boxed{n_0 = 5} \\ 2 + \frac{5}{5} \leq c \quad 3 \leq c \qquad \boxed{c = 3} \end{gathered}$$

$f(n) = 2n + 5 => \Omega(n)$

$$\begin{gathered} f(n) \geq c.g(n) \\ 2n + 5 \geq c.n \\ 2 + \frac{5}{n} \geq c \\ \frac{5}{n} \leq 1 \qquad \forall n \geq 5 \qquad \boxed{n_0 = 5} \\ 2 + \frac{5}{5} \geq c \quad 3 \geq c \qquad \boxed{c = 2} \end{gathered}$$

$f(n) = 2n + 5 => \Theta(n)$

$$\begin{gathered} c_1.g(n) \leq f(n) \leq c_2.g(n) \\ c_1.n \leq 2n + 5 \leq c_2.n \\ c_1 \leq 2 + \frac{5}{n} \leq c_2 \\ \frac{5}{n} \leq 1 \qquad \forall n \geq 5 \qquad \boxed{n_0 = 5} \\ 2 + \frac{5}{5} \geq c_1 \quad 3 \geq c_1 \qquad \boxed{c_1 = 2} \\ 2 + \frac{5}{5} \leq c_2 \quad 3 \leq c_2 \qquad \boxed{c_2 = 3} \end{gathered}$$

Note

- **n is integer** : it represents the size of the inputed data
- **c is float** : it represents a coefficient
- **Why we didn't took $c_1 = 3$ even though $c_1 \leq 3$ when n = 5** : because the inequality $2 + \frac{5}{n} \geq c_1$ must be verified $\forall n \geq n_0 = 5$ if we take n = 6 , $2 + \frac{5}{6} \approx 2.8$ and $c_1 = 3$ isn't ≤ 2.8

Laudau Notation Limit Definition

if $\lim\limits_{n \to +\infty} \left| \frac{f(n)}{g(n)} \right| = k \quad , \quad k > 0 \quad => \quad f(n) \in \Theta(g(n))$

if $\lim\limits_{n \to +\infty} \left| \frac{f(n)}{g(n)} \right| = 0 \quad => \quad f(n) \in O(g(n))$

if $\lim\limits_{n \to +\infty} \left| \frac{f(n)}{g(n)} \right| = +\infty \quad => \quad f(n) \in \Omega(g(n))$

Exercice :

show that $f(n) = 5n^2 - 6n => \Theta(n^2)$

$$c_1.g(n) \leq f(n) \leq c_2.g(n)$$
$$c_1.n^2 \leq 5n^2 - 6n \leq c_2.n^2$$
$$c_1 \leq 5 - \frac{6}{n} \leq c_2$$
$$\frac{6}{n} \leq 1 \qquad \forall n \geq 6 \qquad \boxed{n_0 = 6}$$
$$5 - \frac{6}{6} \geq c_1 \quad 4 \geq c_1 \qquad \boxed{c_1 = 4}$$
$$5 - \frac{6}{6} \leq c_2 \quad 4 \leq c_2 \qquad \boxed{c_2 = 5}$$

$f(n) = 6n^3 \neq \Theta(n^2)$

$$c_1.g(n) \leq f(n) \leq c_2.g(n)$$
$$c_1.n^2 \leq 6n^3 \leq c_2.n^2$$
$$c_1 \leq 6n \leq c_2 \qquad\qquad c_1 \nexists, c_2 \nexists$$

$f(n) = n^2 => O(10^{-5}n^3)$

$$f(n) \leq c.g(n)$$
$$n^2 \leq c.10^{-5}n^3$$
$$\frac{10^5}{n} \leq c$$
$$\frac{10^5}{n} \leq 1 \qquad \forall n \geq 10^5 \qquad \boxed{n_0 = 10^5}$$
$$\frac{10^5}{10^5} \leq c \quad 1 \leq c \qquad \boxed{c = 1}$$

$f(n) = 10n^3 + 3n^2 + 5n + 1 => O(n^3)$

$$f(n) \leq c.g(n)$$

$$10n^3 + 3n^2 + 5n + 1 \leq c.n^3$$

$$10 + \frac{3}{n} + \frac{5}{n^2} + \frac{1}{n^3} \leq c$$

$$\frac{3}{n} \leq 1 \quad \forall n \geq 3$$

$$\frac{5}{n^2} \leq 1 \quad n^2 \leq 5 \quad n \leq \sqrt{5} \quad \forall n \geq 3$$

$$\frac{1}{n^3} \leq 1 \quad \forall n \geq 1$$

$$\forall n \geq 1 \cap \forall n \geq 3 \quad => \quad \forall n \geq 3 \quad \boxed{n_0 = 3}$$

$$10 + \frac{3}{3} + \frac{5}{3^2} + \frac{1}{3^3} \approx 11.59 \qquad \boxed{c = 13}$$