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

```
1:  Var
2:  n, sum, i integer;
3:  Begin
4:  sum ← 0;
5:  i ← 1;
6:  print('Input Integer N : ')
7:  Read(n);
8:  while i <= n do
9:      sum ← sum + i;
10:     i ← i + 1;
11: end while
12: print('Sum is ' ,sum);
13: 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)

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

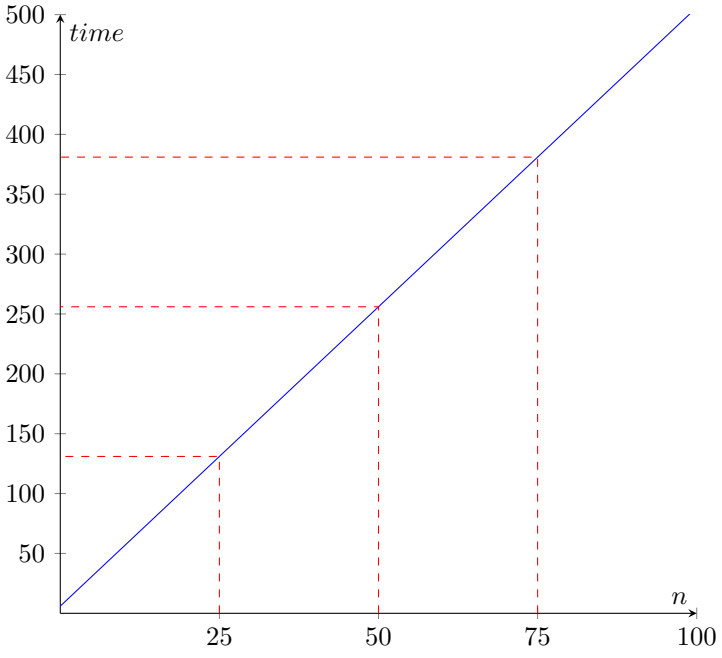
now that we have the complexity function $f(n)$ we need to find the $T(n)$ we have $T(n) = f(n) \times \Delta t$

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

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 $\sim O(1)$