The running time of an algorithm depends on how long it takes a computer to run the lines of code of the algorithm—and that depends on

the speed of the computer,

the programming language, and

the compiler or interpreter that translates the program (source program) from the programming language into runnable code that runs directly on the computer, among other factors.

***Running time of an algorithm-***

1. **Running Time** in terms of the size of its input. [How long the algorithm takes, in terms of the size of its input].
2. Function growth with the input size. (**Rate of growth** of the running time.) [How fast a function grows with the input size].

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

For example, suppose that an algorithm, running on an input of size n, takes 6n^2 + 100n + 300 & 0.6n^2 + 1000n + 3000 machine instructions.

By dropping the less significant terms and the constant coefficients, we can focus on the important part of an algorithm's running time—its **rate of growth**

Big O Notation (O) => Worst Case (upper bound)

Omega Notation (Ω) => Best Case (lower bound)

Theta Notation (Φ) => Average Case (between best & average case) (tight bound)

Here's a list of functions in asymptotic notation that we often encounter when analyzing algorithms, ordered by slowest to fastest growing:

Graphical user interface, application, Excel

Description automatically generated

Space Complexity   
Auxilary Space + Input Space  
Auxilary Space = Space taken to solve the problem  
Input Space = Space taken to store the Input