# Complejidad Algorítmica

Antes de hablar sobre la complejidad algorítmica, debemos saber que es un algoritmo. Un algoritmo implica la descripción precisa de los pasos a seguir para solucionar un problema, sabiendo esto, es lógico que se quiere obtener la opción mas eficiente para el algoritmo, ahí es donde entra la complejidad algorítmica.

La complejidad algorítmica representa la cantidad de recursos que necesita un algoritmo para resolver un problema y, por ende, determina la eficiencia del algoritmo. No se proporcionan medidas absolutas sino medidas relativas en base al tamaño del problema dado. Está se expresa en base al tamaño de la entrada (N)

Existen varios tipos de complejidad constante, la mas pequeñas que vendría siendo la constante, que se representa como O(1), la logarítmica O(log n), la líneas O(n), cuasi lineal O(n log n), cuadrática O(n^2), cúbica O(n^3), polinómica O(n^a), exponencial O(2^n), las polinomiales que son proporcionales a n^a, y las exponenciales que no son factibles, excepto para tamaño de entrada n.

Aquí hay un ejemplo de complejidad constante:

```
x=x+1;
```

Esta no depende de un tamaño de entrada y es solo una asignación, por lo tanto tiene una complejidad de O(1) (constante).