Tarea 1: Complejidad Computacional

1. **O(1):** El tiempo de ejecución del orden 1 (o constante) no depende del tamaño de entrada, siempre toma la misma cantidad de pasos.

Ejemplo: Acceder al elemento de un arreglo según el índice:

*public int obtenerPrimerElemento(){*

*return array[0];*

*}*

1. **O(N):** El tiempo de ejecución del orden N (o lineal) crece de forma proporcional al número de elementos.

Ejemplo: Recorrer un arreglo y sumar sus elementos:

*public int sumarArray(int [] array){*

*int suma = 0;*

*for (int i = 0; I < array.length; i++){*

*suma += array[i];*

*}*

*return suma;*

*}*

Nótese que dentro de la función solo hay 1 “for”, lo que indica que la complejidad computacional es de orden N.

1. **O(N2):** El tiempo de ejecución del orden N2 (o cuadrática) crece con el cuadrado del número de elementos.

Ejemplo: Comparar todos los pares de elementos de un arreglo:

*public void mostrarTodosLosPares(int [] array){*

*for (int i=0; i < array.length; i++){*

*for (int j = 0; j < array.length; j++){*

*System.out.println(“[”+array[i]+”,”+”array[j]+”]”);*

*}*

*}*

*}*

Nótese que dentro de la función hay más de 1 “for”, lo que indica que la complejidad computacional es de orden N2.