

## Tiempos de ejecución

A continuación, se muestra cuánto se tardó el programa en ejecutar las acciones: mostrar artículos del inventario, y mostrar artículos del inventario por categoría.










Profiling results			
Results:  		View:   	
		Collected data:  Snapshot 	
Name	Total Time	Total Time (CPU)	Invocati...
▼  <b>main</b>	8.3 ms (100%)	7.87 ms (100%)	2
>  Tienda.mostrarInventario ()	5.61 ms (70%)	5.54 ms (70.4%)	1
⌚ Tienda.mostrarInventarioPorTipo ()	2.41 ms (30%)	2.33 ms (29.6%)	1

Figura 1 - Hash Map










Profiling results			
Results:  		View:   	
		Collected data:  Snapshot 	
Name	Total Time	Total Time (CPU)	Invocations
▼  <b>main</b>	7.84 ms (100%)	7.73 ms (100%)	2
>  Tienda.mostrarInventario ()	5.18 ms (66.1%)	5.12 ms (66.2%)	1
⌚ Tienda.mostrarInventarioPorTip	2.66 ms (33.9%)	2.61 ms (33.8%)	1

Figura 2 - Linked Hash Map










Profiling results			
Results:  		View:   	
		Collected data:  Snapshot 	
Name	Total Time	Total Time (CPU)	Invocations
▼  <b>main</b>	1.52 ms (100%)	1.51 ms (100%)	2
>  Tienda.mostrarInventario ()	0.933 ms (61.1%)	0.927 ms (61.2%)	1
⌚ Tienda.mostrarInventarioPorTipo	0.595 ms (39%)	0.588 ms (38.8%)	1

Figura 3 - Hash Tree

Se puede observar que el tipo de mapa que menos tiempo se tarda en ejecutar las acciones mencionadas (recorrer su contenido) es el **Tree Map**. El más tardado es el **Hash Map**.

## Complejidad de tiempo

```
/**
 * Muestra todos los productos del inventario
 * @return resultado
 */
public String mostrarInventario(){
    String resultado = "";

    resultado = resultado + "\nTODOS LOS PRODUCTOS DEL INVENTARIO: ";
    ArrayList<String> productosMostrados = new ArrayList<String>();

    for (Map.Entry<String, ArrayList<String>> entrada : inventario.entrySet()) {
        for (String producto : entrada.getValue()) {
            if(!productosMostrados.contains(producto)){
                resultado = resultado + "\n\n" + producto + "\n - Cantidad: " + Collections.frequency(inventario, producto);
            }
            productosMostrados.add(producto);
        }
    }
    return resultado;
}
```

Se puede observar que para mostrar los datos del Hash Map, la complejidad de tiempo que se necesita es  **$O(n^2)$** . Esto se puede saber porque dentro de un ciclo for se encuentra otro ciclo for.

Cuando hay bucles anidados como este, dependiendo de la cantidad de ciclos que haya se eleva la  $n$  a una potencia determinada. En este caso, como hay dos bucles anidados, la  $n$  se eleva al cuadrado (Mejia, 2020).

## Bibliography

Mejia, A. (3 de octubre de 2020). *How to find time complexity of an algorithm?* Obtenido de AdrianMejia.com: <https://adrianmejia.com/how-to-find-time-complexity-of-an-algorithm-code-big-o-notation/>