****

**Universidad Peruana de Ciencias Aplicadas**

Ciencias de la Computación - 9° Ciclo

Profesor: Carlos Jara García

**Programación concurrente y distribuida**

Trabajo Parcial

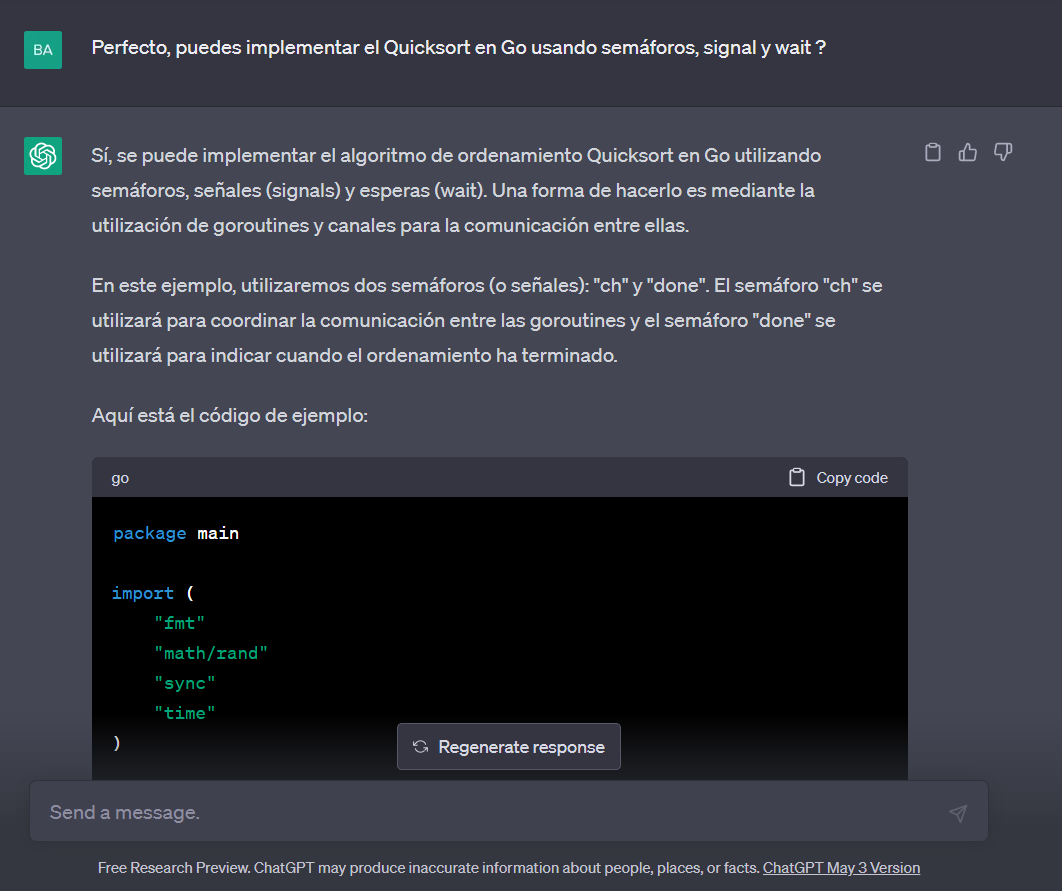
|  |  |  |
| --- | --- | --- |
| Basauri Quispe, Roberto Carlos | U20181C074 | Ciencias de la Computación |
| Arroyo Bonifaz, Luis Roberto | U201716094 | Ciencias de la Computación |

Mayo, 2023

QUICKSORT

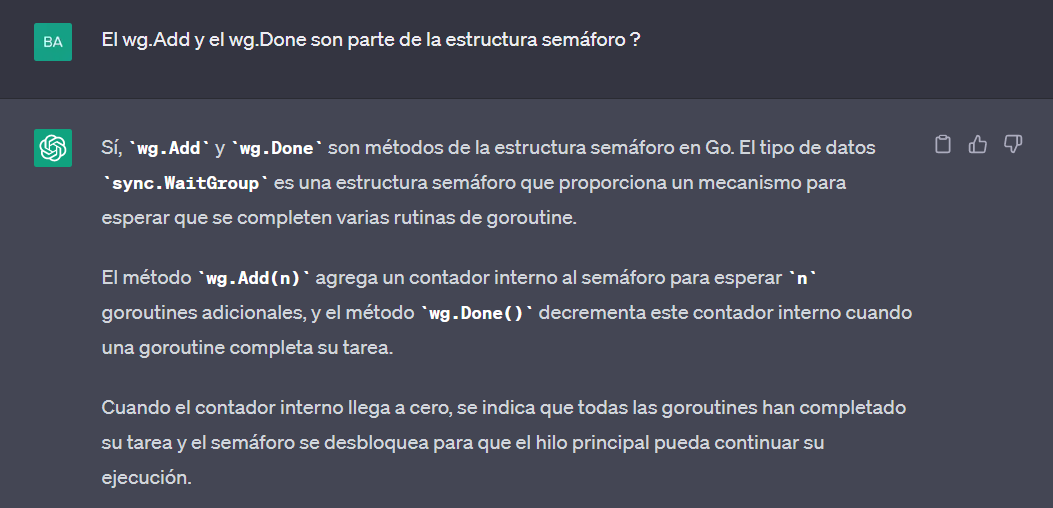
Quicksort es un algoritmo del tipo divide y vencerás. En el que se selecciona un pivote y en base a este, divide el arreglo restante en dos grupos. Cada arreglo se va a comparar con el pivote y se ordenará; sin embargo, no sin antes de realizar el proceso mencionado con el arreglo restante de manera recursiva.

Para realizar el Quicksort en go se le realizaron las siguientes consultas a chatgpt:



**Fuente: ChatGpt  
 Imagen 1: Consulta de código**

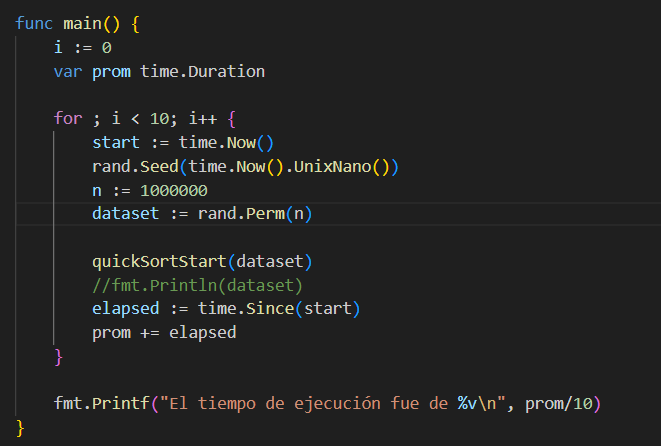
Esta primera versión de código contenía múltiples errores que volvían negativo el contador de acciones o goroutine. Para ello se consultó como operaban los “signal” y “wait” en Go, ya que los aprendidos pertenecían a Promela.



**Fuente: ChatGpt  
 Imagen 2: Consulta**

Teniendo esto en cuenta. Se corrigió el código que por consiguiente tuvo el comportamiento esperado. El de ordenar los arreglos.

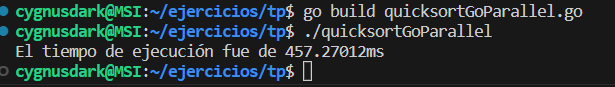
Antes de todo lo mencionado, se hizo la implementación del quicksort de manera serial en código Go. Debido a que el objetivo no es recordar los algoritmos de ordenamiento. Se utilizó el código QuickSort para Go de Boot.Dev. Y se le agregó la lectura de tiempo de ejecución y la de ejecuciones múltiples.



**Fuente: ChatGpt  
 Imagen 3: Código Go**

Este proceso se aplicó tanto para el quicksort.go, quicksorGot.go y quicksortGoParallel.go, el primer archivo (todos los archivos están en el repositorio github adjunto) pertenece al algoritmo en forma serial, el siguiente archivo aplica principios de concurrencia con channels y goroutines, y el tercer archivo aplicaría los patrones signaling y los mecanismos de sincronización, con los wg.Add y wg.Done, en conjunto con el semáforo Struct, que contendría un semáforo para cada valor en el arreglo.

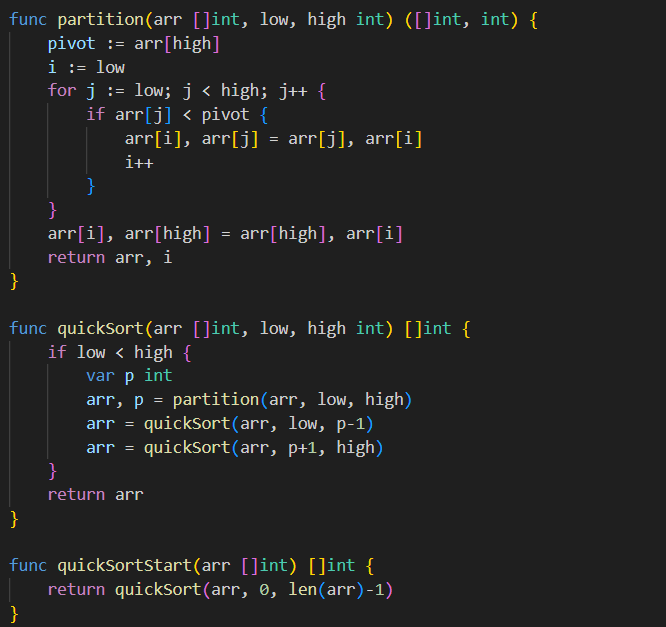
Ahora para el tiempo de ejecución se tiene lo siguiente:



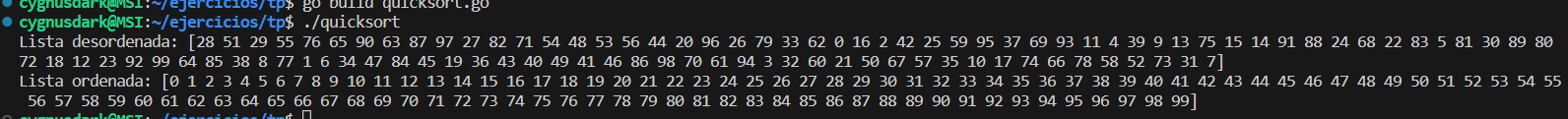
**Fuente: ChatGpt  
 Imagen 4: Ejecución**

Este vendría a ser el tiempo promedio de ejecución. El cual es de 0.5 s en 10 ejecuciones de código. (1000 ejecuciones demoraba demasiado, y el plazo se acababa).

Implementación de forma tradicional:

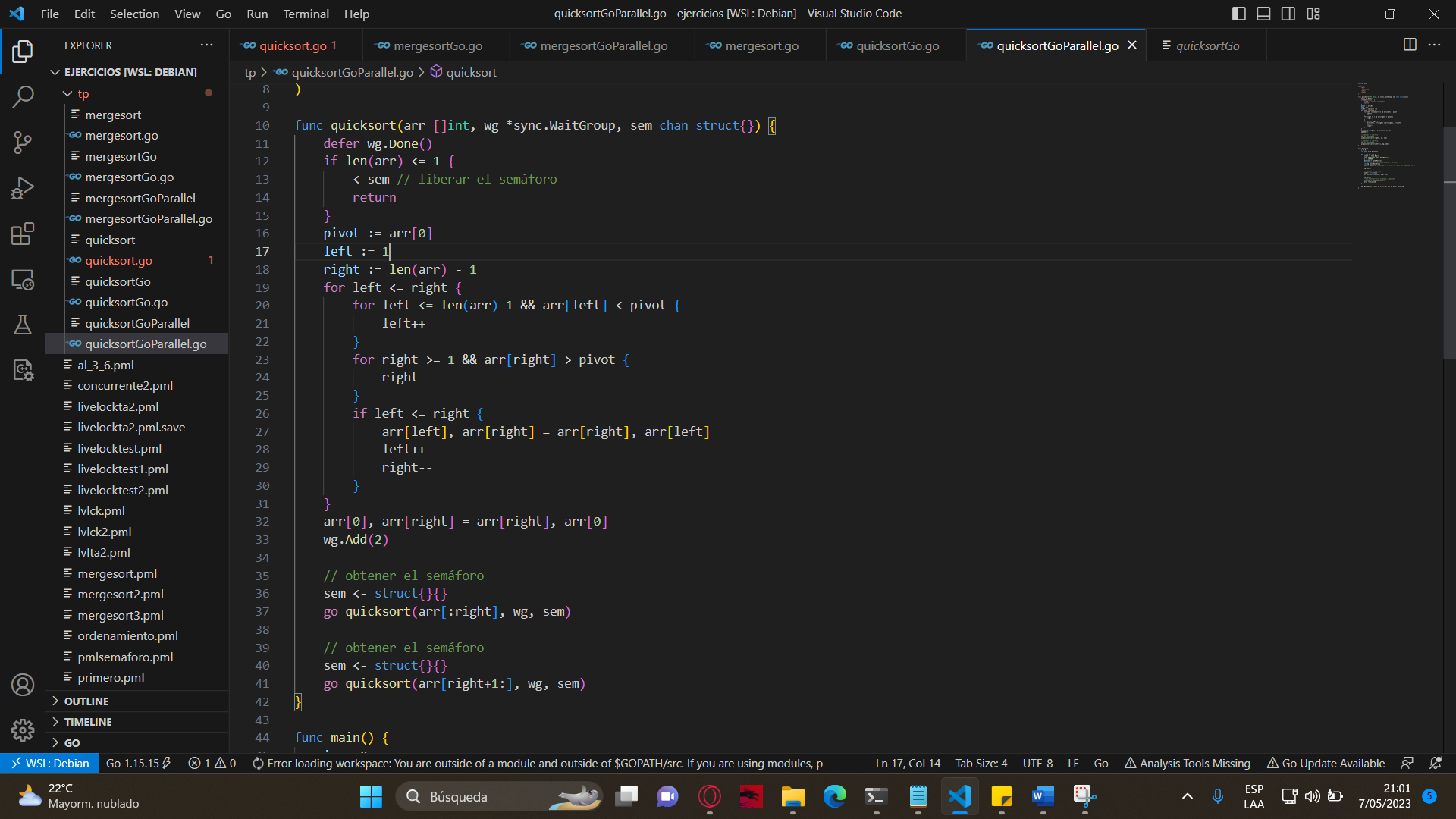


**Fuente: VSCode  
 Imagen 5: Código**

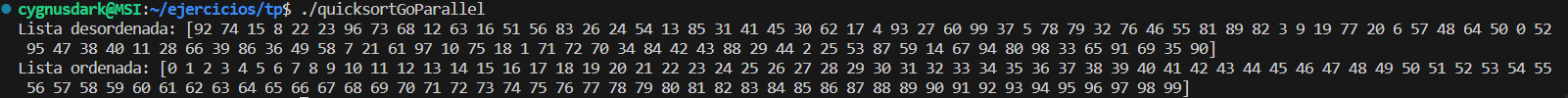
****

**Fuente: VSCode  
 Imagen 6: Resultado**

**Implementación concurrente Quicksort**

****

**Fuente: VSCode  
 Imagen 7: Código**

****

**Fuente: VSCode  
 Imagen 8: Resultado del Parallel**

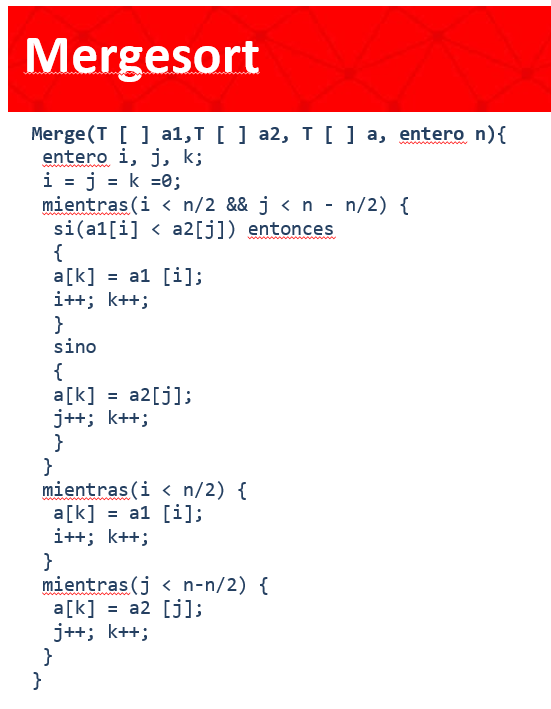
MERGESORT

El algoritmo mergesort consiste en la división de un arreglo en dos subarreglos de longitud igual a la mitad del original. Esta operación se realizará de forma recursiva hasta que no se pueda dividir más, es decir el arreglo tenga longitud de 1. Finalmente se mezclaran las sublistas de menor a mayor hasta formar un único arreglo que contendrá todos los valores ordenados. La complejidad algorítmica es igual a O(n log n).

Para realizar el algoritmo se tomó de referencia lo aprendido en el curso Algoritmos y Estructura de Datos, el pseudocódigo se escribió en el lenguaje GoLang.

**Fuente: UPC**

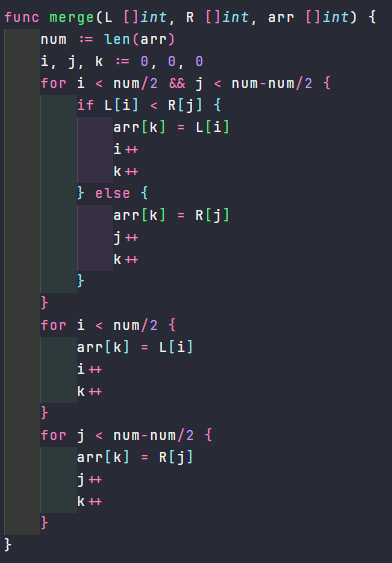
**Imagen 7 : Consulta**



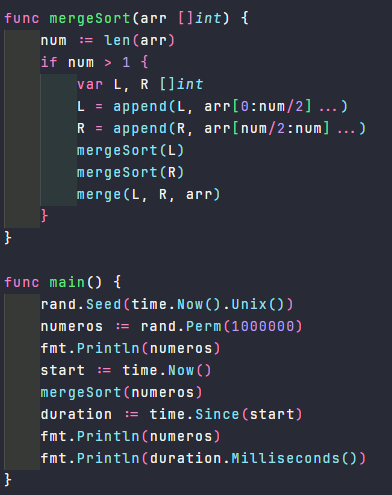
**Fuente: UPC**

**Imagen 8 : Consulta**

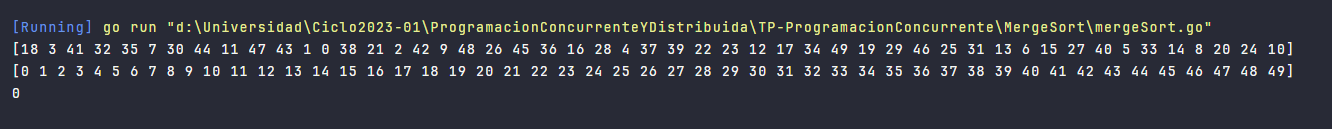
Implementación de manera tradicional:



**Fuente: VSCode  
 Imagen 9: Código**

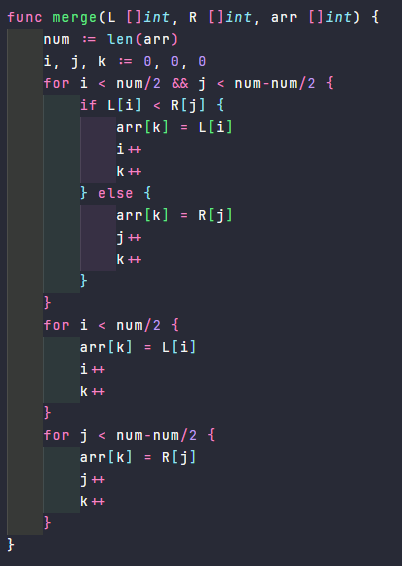


**Fuente: VSCode  
 Imagen 10: Código**

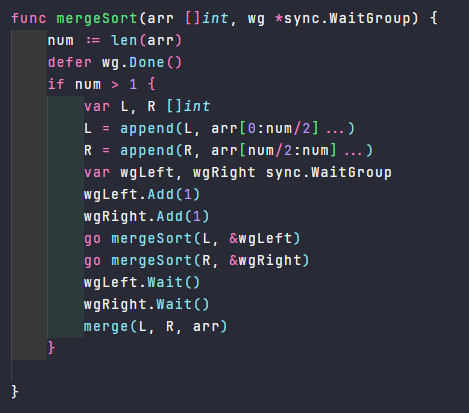


**Fuente: VSCode  
 Imagen 11: Resultado**

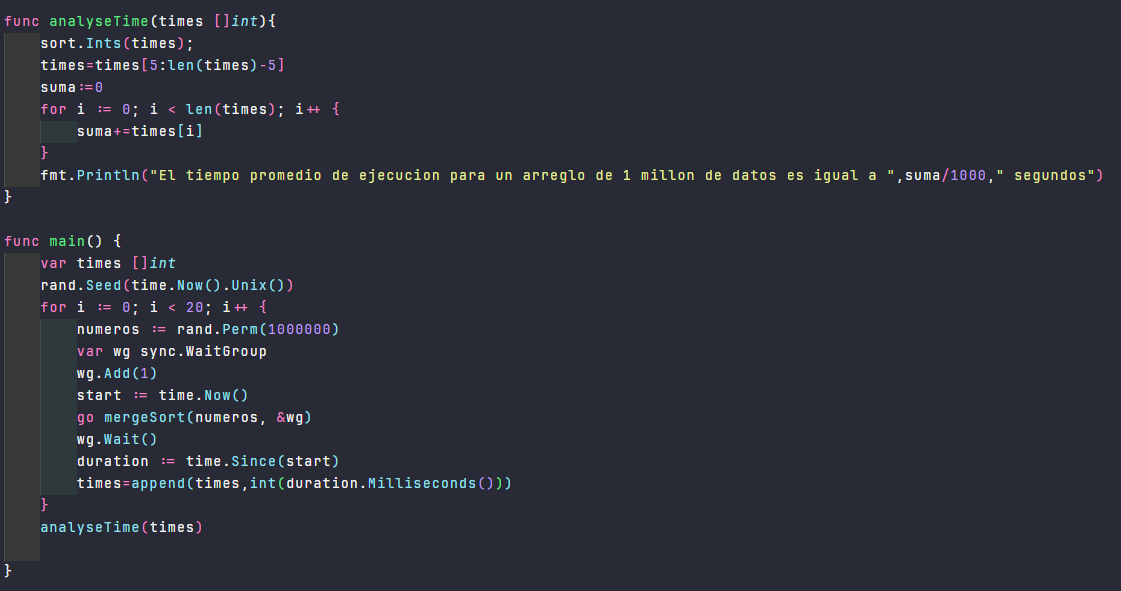
Implementación de forma concurrente



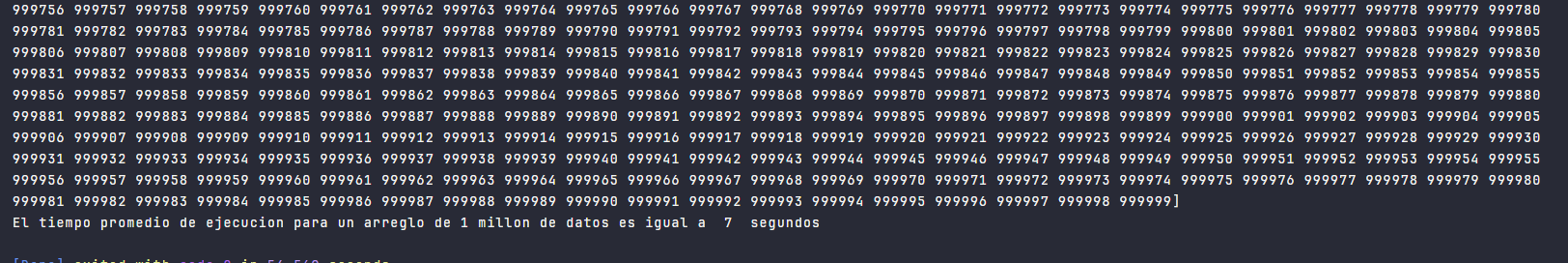
**Fuente: VSCode  
 Imagen 12: Código**



**Fuente: VSCode  
 Imagen 13: Código**



**Fuente: VSCode  
 Imagen 14: Código**

****

**Fuente: VSCode  
 Imagen 15: Resultado**

Anexos:

Video: <https://www.youtube.com/watch?v=rgpoQiSHPeY>

Repositorio: <https://github.com/Larbz/TP-ProgramacionConcurrente>

Bibliografía:

* Wagner, L. (2021, 17 junio). Quick Sort in Golang. *Boot.dev*. <https://blog.boot.dev/golang/quick-sort-golang/>
* *Introducing ChatGPT*. (s. f.). <https://openai.com/blog/chatgpt>
* *Algoritmos y Estructura de Datos*. (Canaval, Jara, Mendiolaza, Namay)