*****Instituto Politécnico Nacional***

***Escuela Superior de Cómputo***

*Aplicaciones para Comunicaciones de Red*

***Practica 6: Algoritmo bucketsort***

***Integrantes:***

* *Ornelas García Luis Ángel*
* *Sampayo Hernández Mauro*

***Grupo:*** *3CM5*

***Profesor:*** *Moreno Cervantes Axel Ernesto*

**Introducción:**

El ordenamiento por cubetas (bucket sort en inglés) es un algoritmo de ordenamiento que distribuye todos los elementos a ordenar entre un número finito de cubetas. Cada cubeta sólo puede contener los elementos que cumplan unas determinadas condiciones. Las condiciones deben ser excluyentes entre sí, para evitar que un elemento pueda ser clasificado en dos cubetas distintas. Después cada una de esas cubetas se ordena individualmente con otro algoritmo de ordenación (que podría ser distinto según la cubeta), o se aplica recursivamente este algoritmo para obtener cubetas con menos elementos. Se trata de una generalización del algoritmo Pigeonhole sort. Cuando los elementos a ordenar están uniformemente distribuidos la complejidad computacional de este algoritmo es de O(n).

**Desarrollo:**

* **LinServer.c**

**Función main:**

**Función \*Servidor:**

**Función ShellSort:**

* **Hilo\_retorno.c**

Este programa actúa como el Cliente, el cual lleva a cabo la inicialización de cada una de las cubetas en donde se almacenarán todos los elementos que serán ordenados para posteriormente ser enviados mediante hilos al servidor, en donde serán ordenados.

Este Cliente cuenta con una estructura definida como “datos” de la cual serán inicializadas tantas estructuras como número de cubetas se tengan. Esta estructura contiene un arreglo de caracteres y un entero, en donde se guardará la dirección y el puerto al que se conectarán el cliente y el servidor respectivamente, un apuntador que apunta a el contenido de una cubeta, y un entero donde se guardará el tamaño de la cubeta.

El Cliente cuenta también con 2 arreglos de enteros globales con una capacidad máxima de 1000 números cada uno, en dónde se guardará el arreglo de números a ordenas, y el arreglo ya ordenado respectivamente. Además, se cuenta con un contador entero global inicializado en 0 definido como j.

**Función main:**

Esta es la función principal del Cliente. Al iniciar el programa, se le pedirá al usuario que elija el número de cubetas que desea inicializar para llevar a cabo el ordenamiento de el arreglo de números, para después ejecutar la función “GenerarNumeros()” la cual se encarga de crear el arreglo de números aleatorios.

Después se lleva a cabo la inicialización de un arreglo de n\_cubetas (número de cubetas) hilos y se ejecuta la función “CrearCubeta()”.

**Función GenerarNumeros:**

**Función CrearCubeta:**

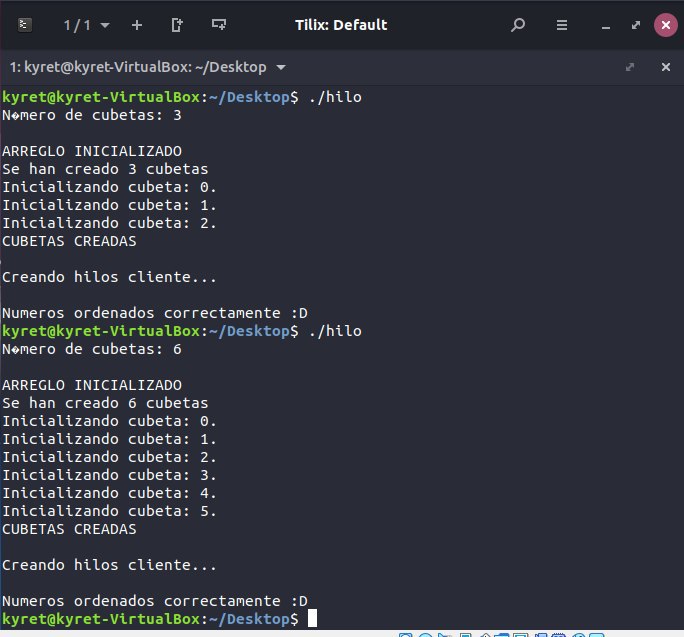
**Función LlenarCubeta:**

**Función \*Cliente:**

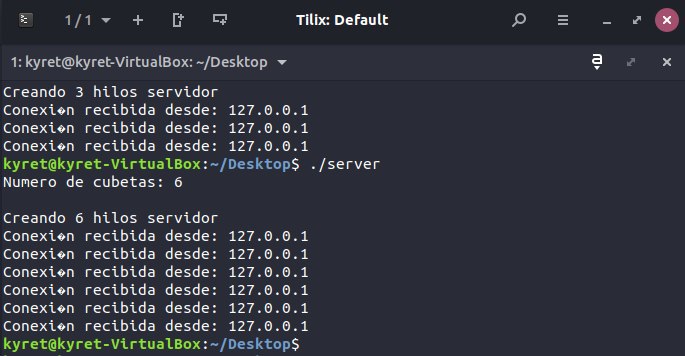
**Pruebas:**

Se realizaron dos pruebas inicializando 3 y 6 cubetas respectivamente, para realizar el ordenamiento del arreglo de números generados aleatoriamente

Aquí se muestra la inicialización de las cubetas en ambas pruebas en el lado del Cliente:



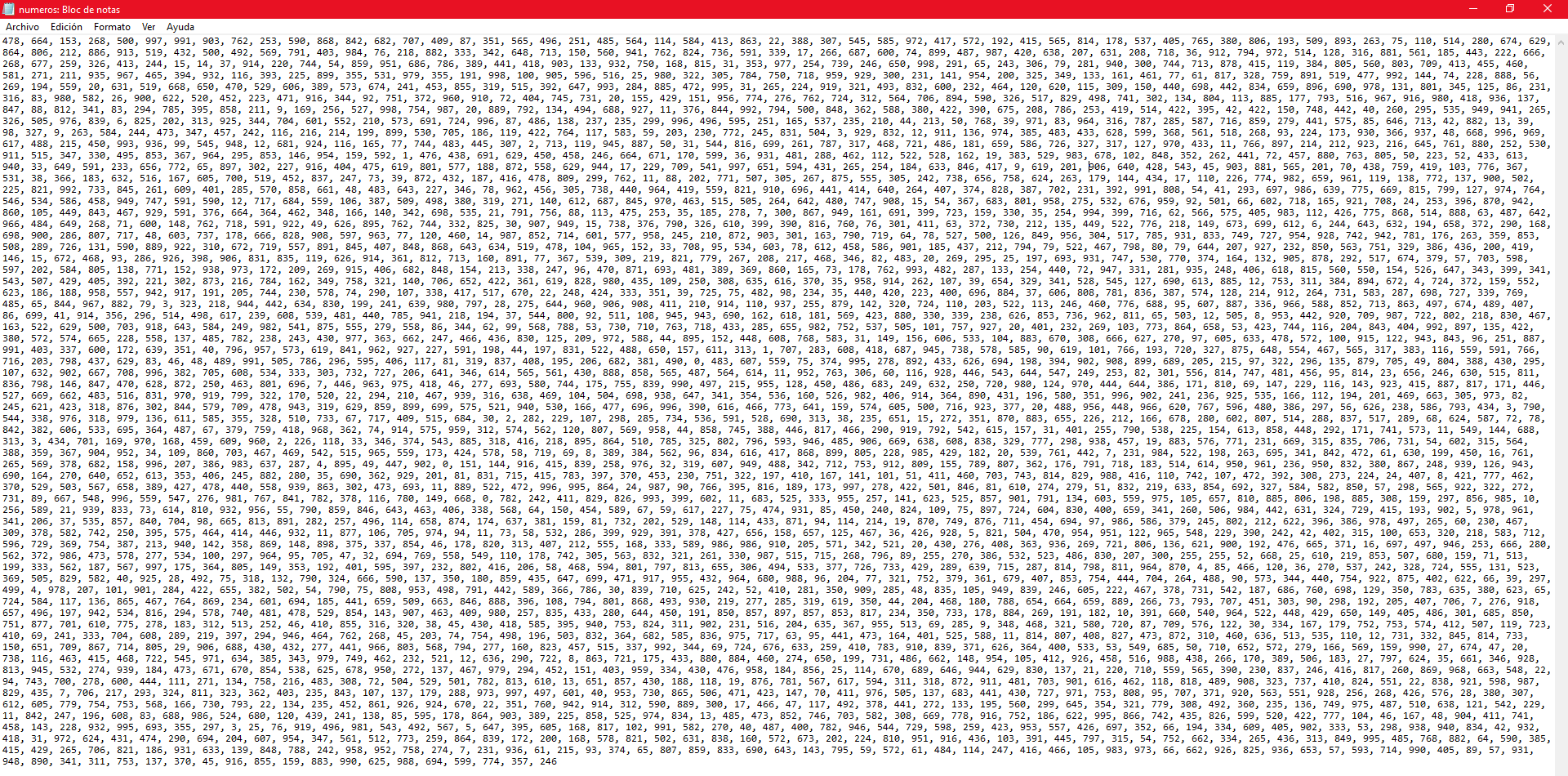
Aquí se muestra el recibimiento de las cubetas en ambas pruebas en el lado del Servidor:



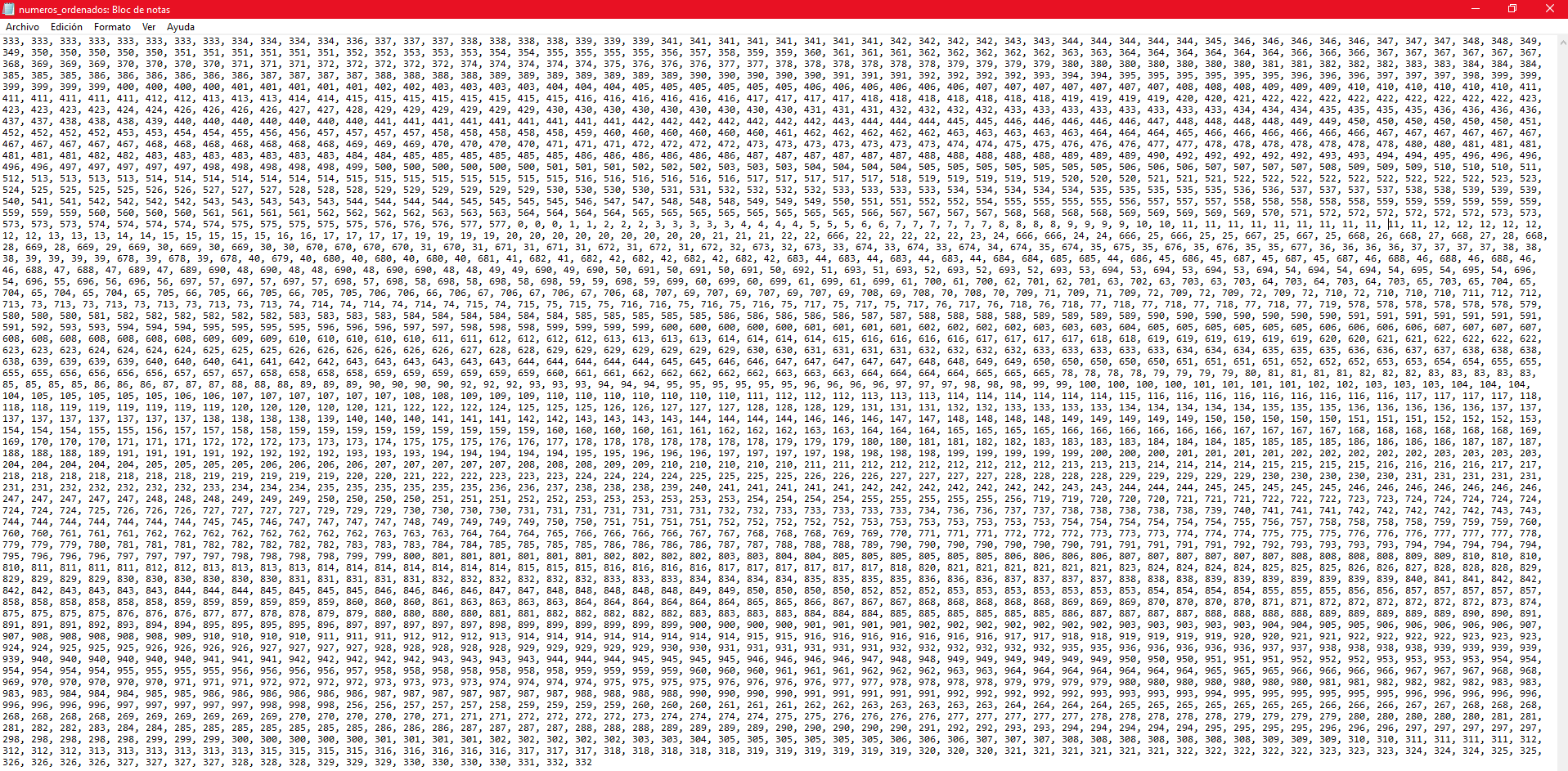
Finalmente, se muestran los dos archivos de texto (uno con el arreglo de números aleatorios, y el otro con el mismo arreglo ya ordenado) generados en cada una de las pruebas:

* **Prueba con 3 cubetas:**

Arreglo de números aleatorios:

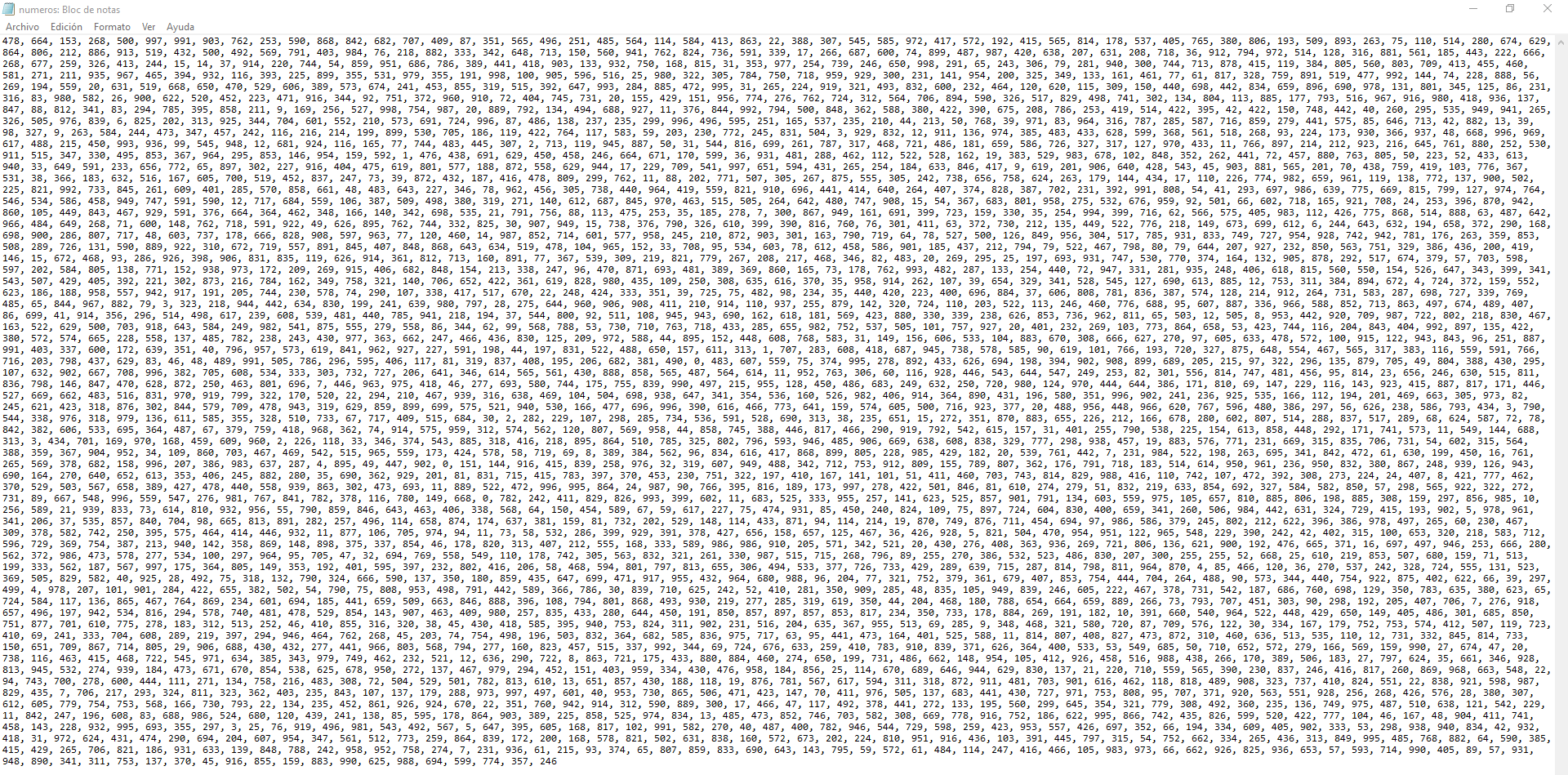


Arreglo ordenado:

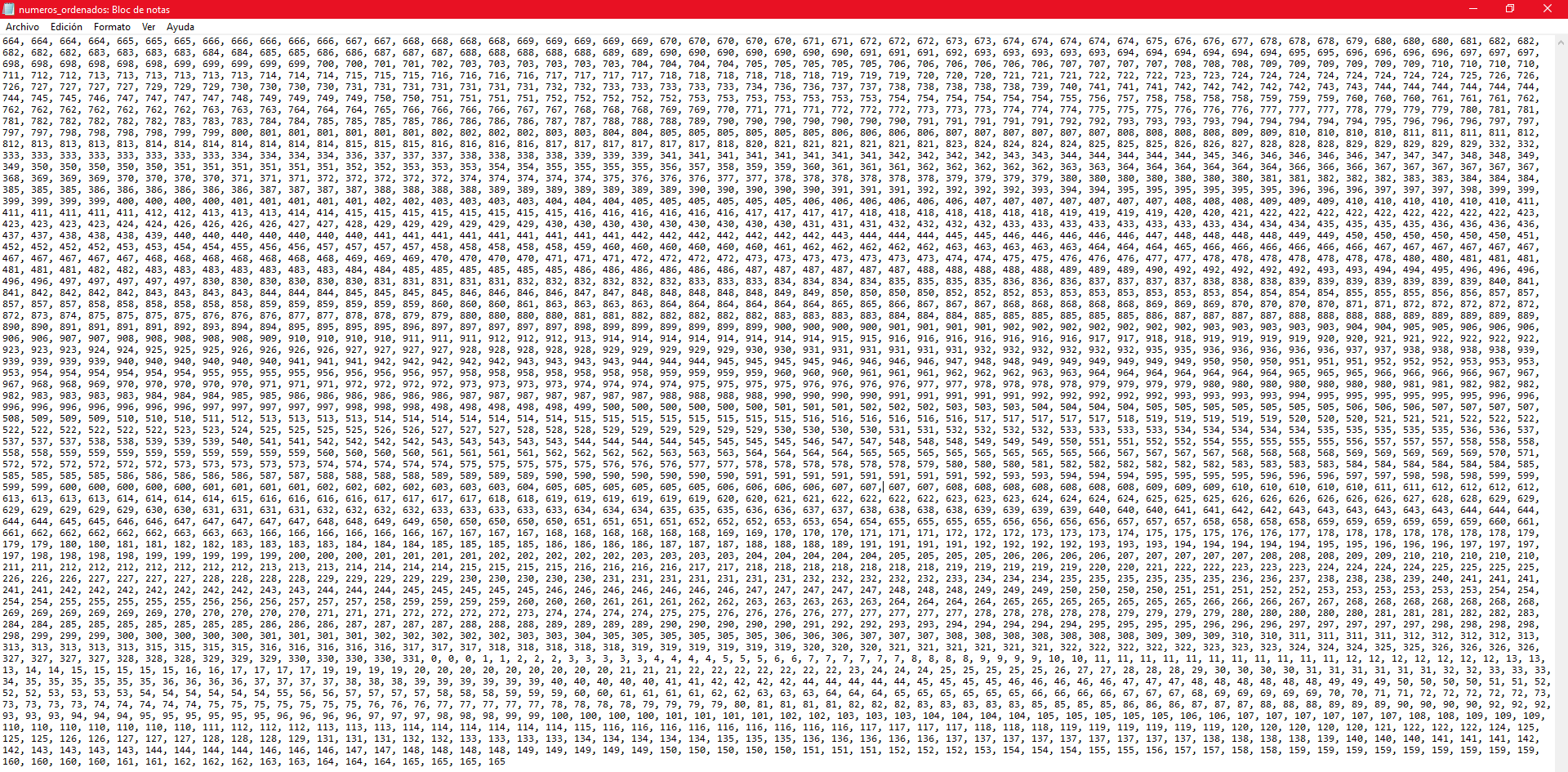


* **Prueba con 6 cubetas:**

Arreglo de números aleatorios:



Arreglo ordenado:



**Conclusión:**

El algoritmo bucketsort es un algoritmo de ordenamiento que resulta ser bastante eficaz al tener una complejidad computacional de O(n), y con la aplicación de hilos en cliente servidor para llevar a cabo el ordenamiento en cada una de las cubetas, este proceso se realiza con un poco más de rapidez.

**Bibliografía:**

* <https://pubs.opengroup.org/onlinepubs/007908799/index.html>