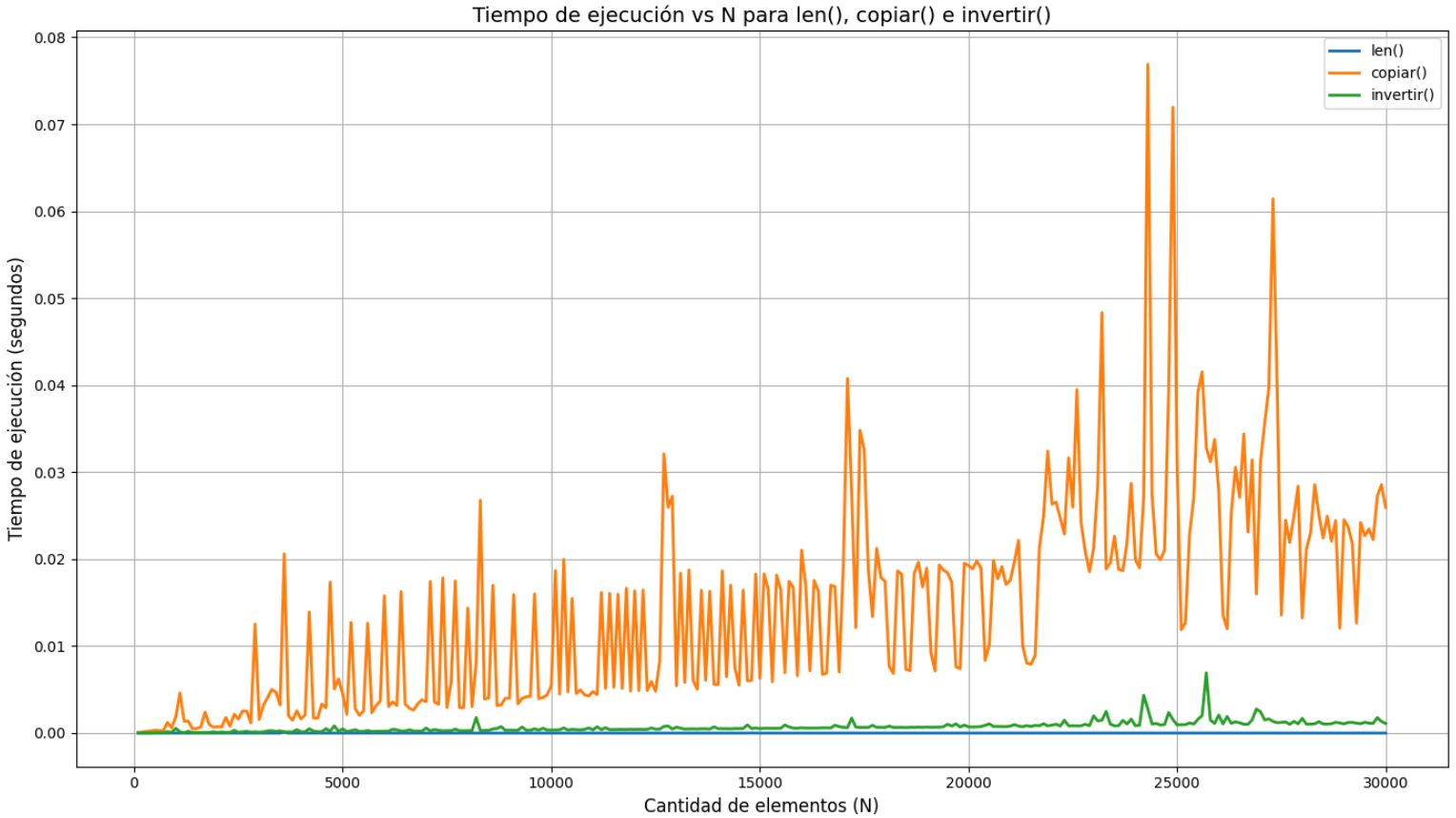
PROYECTO 1:



En esta parte del trabajo se implementó una estructura de datos dinámica basada en una Lista Doblemente Enlazada, la cual nos permite almacenar elementos que pueden compararse entre sí, contando con referencias hacia el nodo anterior y el siguiente.
Parte de la consigna era programar operaciones fundamentales como inserción, extracción, copia, concatenación, etc.

Métodos:

- esta_vacia(): Si len==0 devuelve true
- __len()__ : Devuelve la longitud de la lista
- agregar_al_inicio(item): insertar en la cabeza, teniendo en cuenta las relaciones de los nodos
- agregar_al_final(item): insertar en la cola, teniendo en cuenta las relaciones de los nodos
- insertar(item,posicion): inserta un dato en una posicion en especifico
- extraer(posicion): extrae un dato de una posicion en especifico
- copiar(): crea una copia elemento a elemento
- invertir(): invierte la lista en lugar, sin listas auxiliares
- concatenar(Lista): añade otra lista al final (Concatena)
- __add__(Lista): permite usar el operador **“+”** para sumar dos listas (reutiliza concatenar y copiar).

Analizamos la complejidad de tres de estos metodos y los comparamos su eficacia con listas de hasta 10000 elementos

Resultados:

len(): el tiempo de ejecución es constante, confirmando su complejidad $O(1)$.

copiar(): el tiempo crece linealmente con el tamaño, confirmando $O(n)$, aunque observamos irregularidades en la grafica podemos observar una clara tendencia.

invertir(): también mostró crecimiento lineal, como se esperaba.