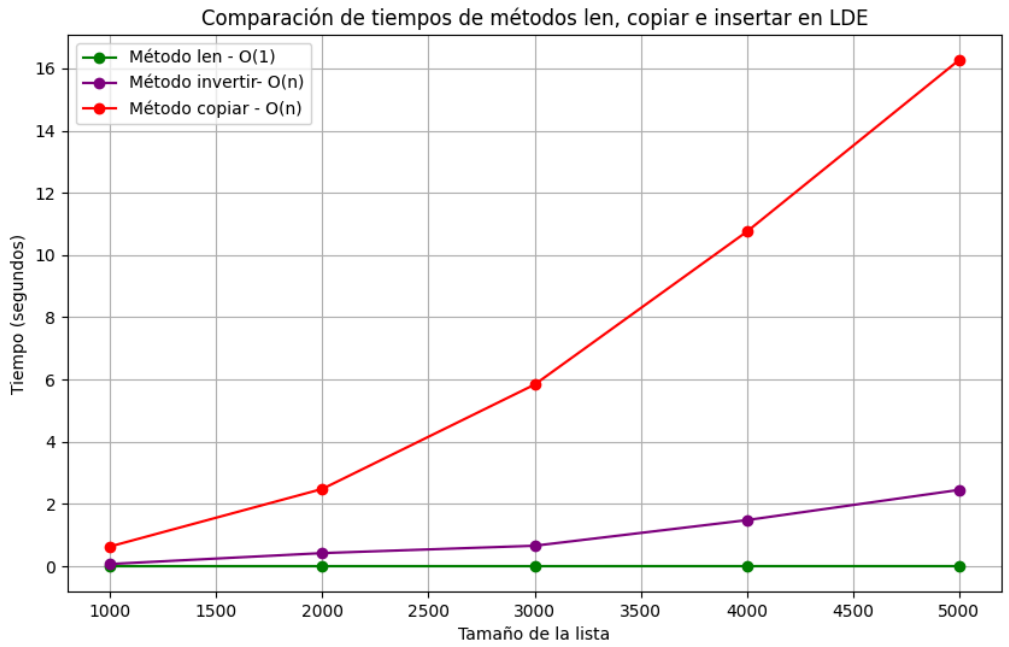
Problema 1

La clase que implementamos cumple con la especificación lógica solicitada para el TAD Lista Doblemente Enlazada, incluyendo los métodos para insertar, extraer, copiar, invertir, concatenar, calcular el tamaño y recorrer la lista con un ciclo. Durante las pruebas vimos que cada operación se comporta como esperábamos: el método len tarda siempre lo mismo sin importar el tamaño de la lista (tiempo constante), lo que confirma que es O(1). En cambio, los métodos copiar e invertir muestran un tiempo que aumenta a medida que crece la lista, lo cual corresponde a una complejidad O(n) ya que recorren todos los nodos una sola vez.

La gráfica refleja estas diferencias: la línea verde de len es prácticamente plana, mientras que las líneas de copiar e invertir crecen de manera lineal, con copiar tardando un poco más porque además de recorrer debe ir creando los nuevos nodos. Esto confirma que la implementación respeta las eficiencias esperadas y que los algoritmos se resolvieron de forma correcta.

**Gráfica**




En la gráfica de N vs tiempo de ejecución se observa que:

El método `__len__` *(línea verde)* mantiene un tiempo prácticamente constante a medida que aumenta el tamaño de la lista. Esto confirma que su complejidad es $O(1)$.

El método `invertir` *(línea violeta)* muestra un crecimiento lineal, al igual que el método copiar, aunque con tiempos inferiores al método copiar. Esto concuerda con una complejidad $O(n)$, ya que el algoritmo tarda un tiempo que crece directamente proporcional al tamaño de los datos. También se justifica porque el método sólo emplea un bucle `while`.

El método `copiar` *(línea roja)* presenta un crecimiento lineal del tiempo, con respecto al número de elementos. El hecho de que la pendiente aumenta de manera proporcional al tamaño de la lista valida que su complejidad es $O(n)$. Dado que, al igual que invertir emplea sólo el método `while`, la diferencia es que además de recorrer debe crear nuevos nodos, por esto la diferencia de pendiente.

**Conclusiones**

Se logró cumplir con las implementaciones solicitadas, pasando los test correspondientes. Se verificó el orden de complejidad de cada método solicitado mediante las gráficas, aunque estás pueden arrojar errores debido a que estas comparan respecto del tiempo, y no es un método 100% eficaz, por esto se constató lo observado en la figura 1 principalmente por la observación del código, examinando si poseen o no bucles for o while, y confirmando así el orden de complejidad.