Problema 2

En el problema 2 se implementó el TAD lista doblemente enlazada como clase de Python con los distintas funciones, a continuación se detalla la función y su correspondiente metodología:

- esta_vacia(): Esta función se utiliza para verificar si la lista no contiene elementos, esto lo realiza al comparar el tamaño de lista con cero; en caso de que esta comparación se verifique, es decir, la lista está vacía, la función retorna "True".
- __len__(): Sobrecarga el método len() de Python, para poder ser utilizado en TADs lista doblemente enlazadas; de esta manera retorna el tamaño de la lista.
- agregar_al_inicio(item): Inserta un nodo al inicio de la lista, debe de especificarse el dato que se adiciona, la función modifica el tamaño de la lista y la referencia del que era anteriormente el primer elemento.
- agregar_al_final(item): Inserta un nodo al final de la lista, debe de especificarse el dato que se adiciona, la función modifica el tamaño de la lista y la referencia del que era anteriormente el último elemento.
- insertar(item, posicion): Inserta un nodo a la lista, debe de especificarse el dato que se adiciona y la posición, en caso de no especificarse posición, se considera que se agrega al inicio de la lista, la función modifica el tamaño de la lista y la referencia de los nodos adyacentes a la agregado.
- extraer(posicion): Extrae el nodo de la lista, el ubicado en la posición indicada, al realizarlo se disminuye el tamaño de la lista en uno, se retorna un ítem con el dato contenido en el nodo y se modifican las referencias de los nodos adyacentes al extraído.
- copiar(): Crea una lista doblemente enlazada que es idéntica a la lista original. Respeta el dato de los nodos y sus referencias.
- invertir(): Modifica la lista, de manera que su orden sea el inverso; se modifican las referencias de los nodos pero se mantiene el dato de cada nodo.
- concatenar(lista): Crea una nueva lista a partir de la unión de dos listas otorgadas; dicha conjunción respeta que el último nodo de la lista inicial tiene como referencia al primer nodo de la lista agregada.
- __add__(lista): Sobrecarga el método que permite utilizar el operador "+" para concatenar listas.

Dicho código pasó efectivamente las pruebas unitarias provistas por la cátedra. Posteriormente se realizaron gráficas de cantidad de elementos (n) vs tiempo de ejecución para los métodos invertir() (Figura 1), __len__() (Figura 2) y copiar() (Figura 3) con las cuales pudimos comprobar su eficiencia temporal. En la función __len__() se observa una complejidad de O(1). Por otro lado, en las funciones invertir() y copiar() obtenemos O(n), esto se corresponde con el resultado esperado, pues copiar e invertir requieren recorrer los n elementos de la lista.

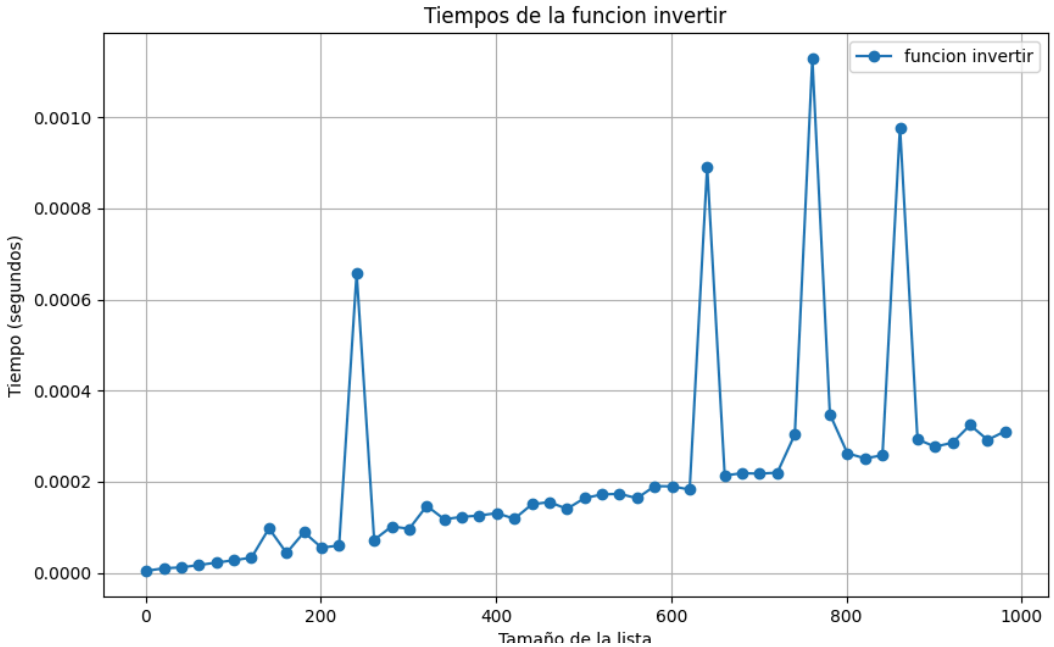

Figura (1): Gráfica tiempo/tamaño para la función invertir.

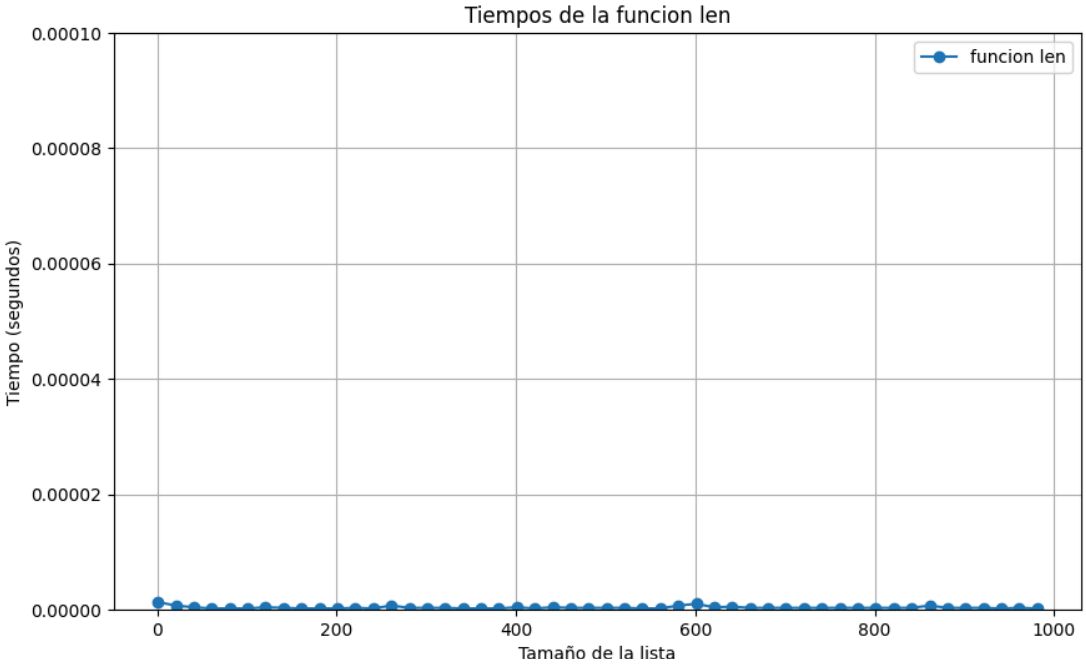

Figura (2): Gráfica tiempo/tamaño para la función len.

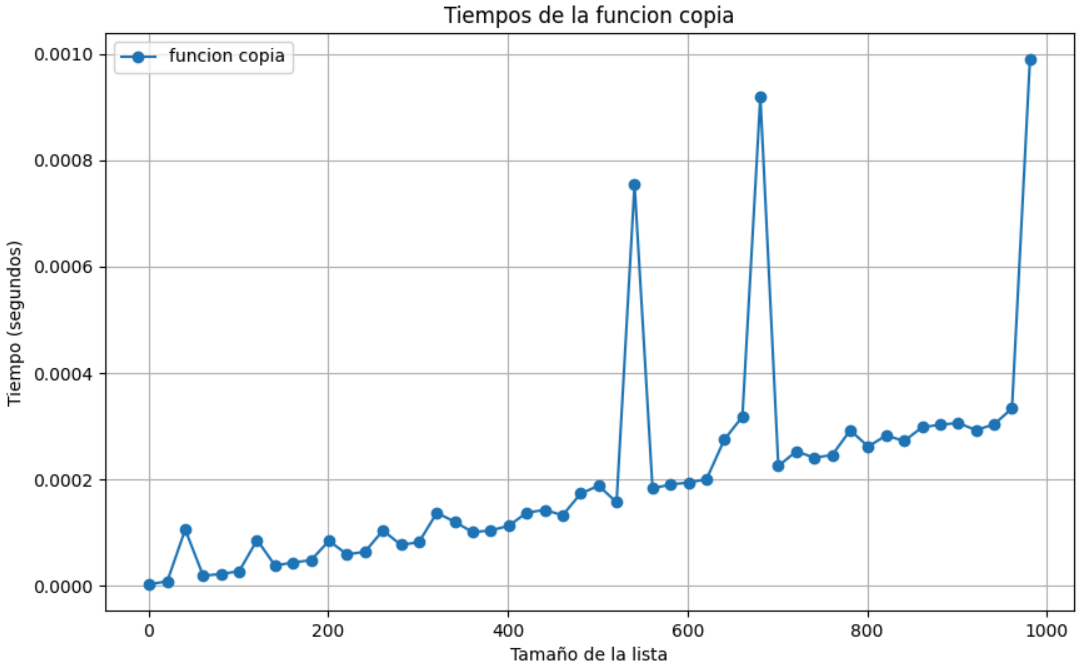

Figura (3): Gráfica tiempo/tamaño para la función copia