Universidad Nacional de Entre Ríos
Facultad de Ingeniería

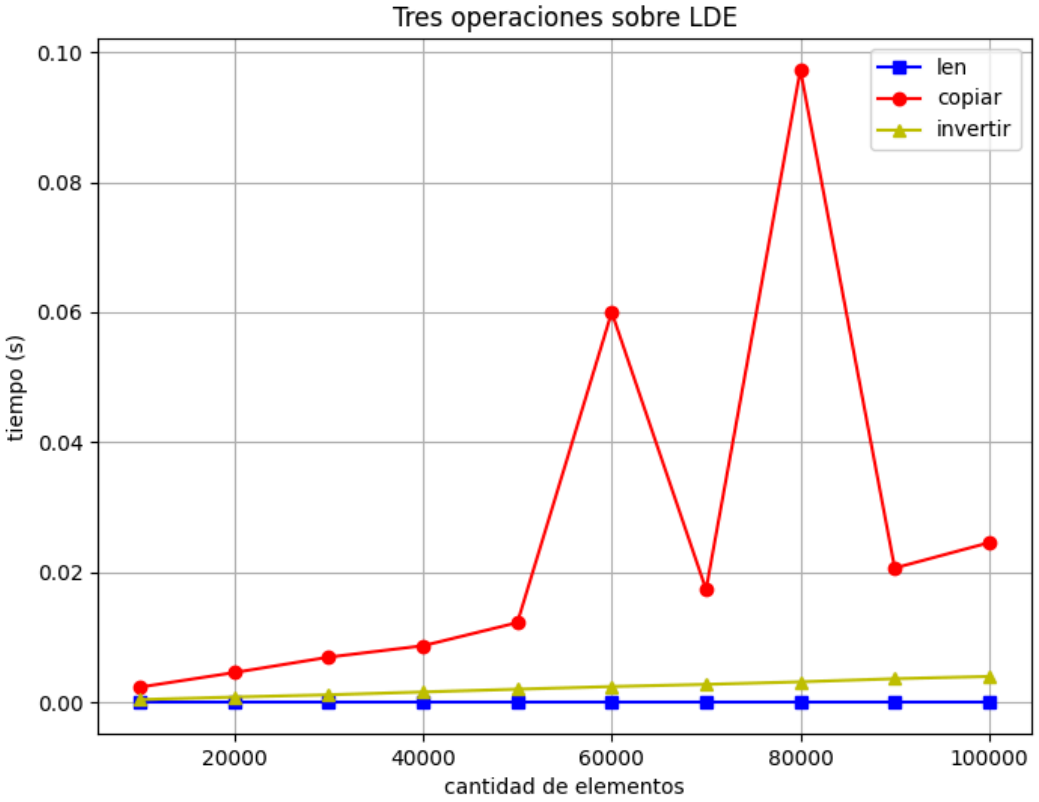
Algoritmos y estructuras de datos

Informe general del Trabajo Práctico N°1
“Aplicaciones de los tipos abstractos de datos”

Alumnos:

- Tarabini Melina
- Rodríguez Esteban

Problema 1:



En la lista doblemente enlazada, la operación `len` tiene complejidad **O(1)** porque solo consulta un contador interno. En cambio, **invertir** y **copiar** son lineales: invertir es más rápido ya que sólo intercambia punteros, mientras que copiar requiere crear nuevos nodos, lo que lo hace más costoso.

La estructura se basa en **nodos** con un valor y dos referencias (anterior y siguiente). La clase `ListaDobleEnlazada` gestiona estos nodos mediante tres atributos:

- **__ cabeza** : referencia al primer nodo (su atributo anterior siempre es None).
- **__cola**: referencia al último nodo (su atributo siguiente siempre es None).

- **__cantidad_elementos**: contador que lleva la cantidad de nodos, inicializado en cero.

Entre las funciones implementadas se encuentran:

- **__str__(self)**: devuelve una cadena con los elementos separados por " - ". Si no hay nodos, retorna "lista sin elementos".

- **esta_vacia(self)**: indica si la lista está vacía.

- **__len__(self)**: permite utilizar len(lista) para conocer el total de nodos.

- **agregar_al_inicio(self, item)**: inserta un nodo al inicio. Si estaba vacía, el nodo se vuelve cabeza y cola a la vez.

- **agregar_al_final(self, item)**: incorpora un nodo al final, con manejo especial si la lista aún no tiene elementos.

- **insertar(self, item, posicion=-1)**: inserta en cualquier posición válida. En 0 agrega al inicio, en negativo o al final coloca el nodo al final, y en posiciones intermedias ajusta los punteros vecinos.

- **extraer(self, posicion=-1)**: elimina un nodo y devuelve su valor. Si no se pasa posición, quita el último. Contempla los casos de cabeza y cola y actualiza referencias. Incluye validaciones de errores.

- **copiar(self)**: genera una lista nueva con los mismos valores, preservando el orden. Se crean nodos nuevos con insertar.

- **invertir(self)**: invierte el orden de los nodos intercambiando sus punteros y luego actualiza cabeza y cola.

- **concatenar(self, p_lista)**: agrega al final de la lista todos los elementos de otra lista, sin modificar la original.

- **__add__(self, p_lista)**: redefine el operador +, creando una lista nueva que combina primero la actual y luego la pasada como parámetro, usando internamente concatenar.