Estructura de Datos y Algoritmos 2

Semestre 2023-1

Profesor: Jorge Alberto Solano Galvez

# Práctica 03. Algoritmos de ordenamiento. Parte 3.

Grupo 06

Integrantes:
Aguilar Martinez Erick Yair
Casillas Herrera Leonardo Didier

**Práctica 3**

**Objetivo:**
Conocer e identificar la estructura de los algoritmos de ordenamiento Counting Sort y Radix Sort.
Actividades:

- Implementar el algoritmo counting sort en lenguaje Python para ordenar una secuencia de datos.
- Implementar el algoritmo radix sort en lenguaje Python para ordenar una secuencia de datos.

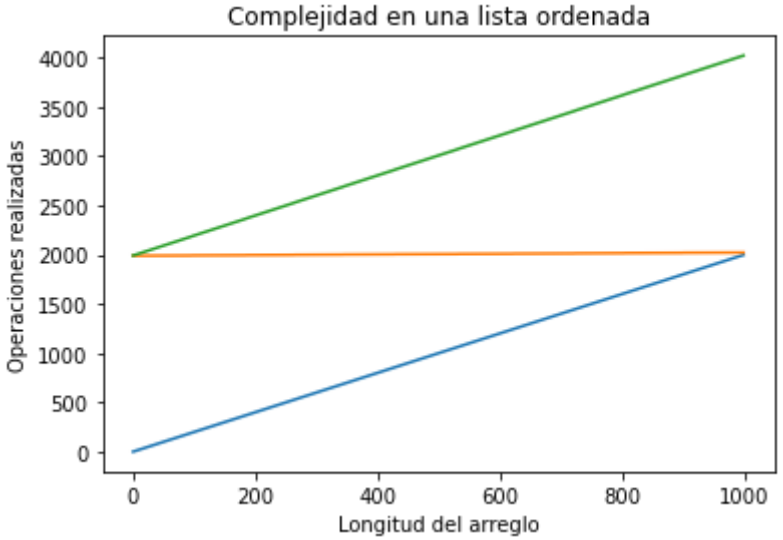
**Instrucciones:**

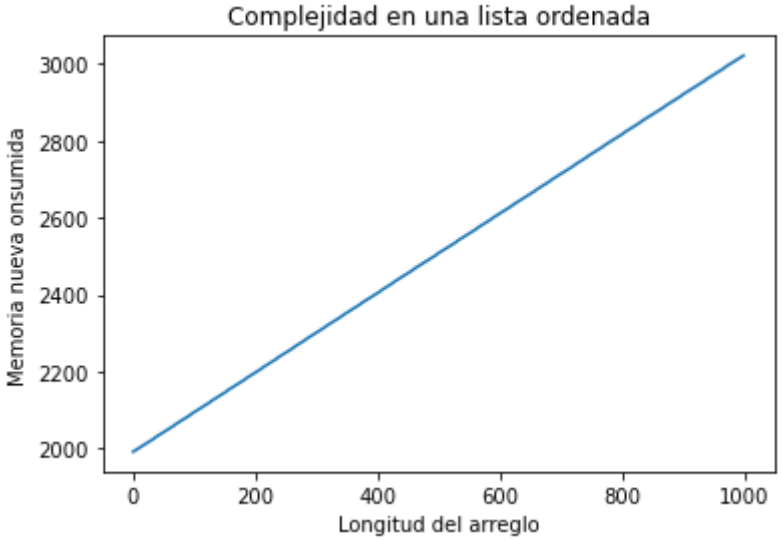
- Implementar en lenguaje Python los algoritmos de ordenamiento counting sort y de radix sort que permitan ordenar objetos
- A partir de los algoritmos implementados en Python, obtener los polinomios del mejor, el peor y el caso promedio de complejidad de counting sort y de radix sort
- A partir del algoritmo implementado en Python, generar la gráfica del tiempo de ejecución para los algoritmos de counting sort y de radix sort para diferentes instancias de tiempo (listas de 1 a 1000 elementos) para los siguientes casos:
  – El mejor caso: con una lista ordenada
  – El peor caso: lista ordenada en forma inversa
  – El caso promedio: una lista aleatoria

La práctica se puede realizar de manera individual, en parejas o en equipo de 3 integrantes. Cada persona debe subir su práctica a Google Classroom porque las conclusiones son individuales. Durante la revisión de la práctica se le puede cuestionar a cualquier integrante del equipo, la calificación es única para todo el equipo.

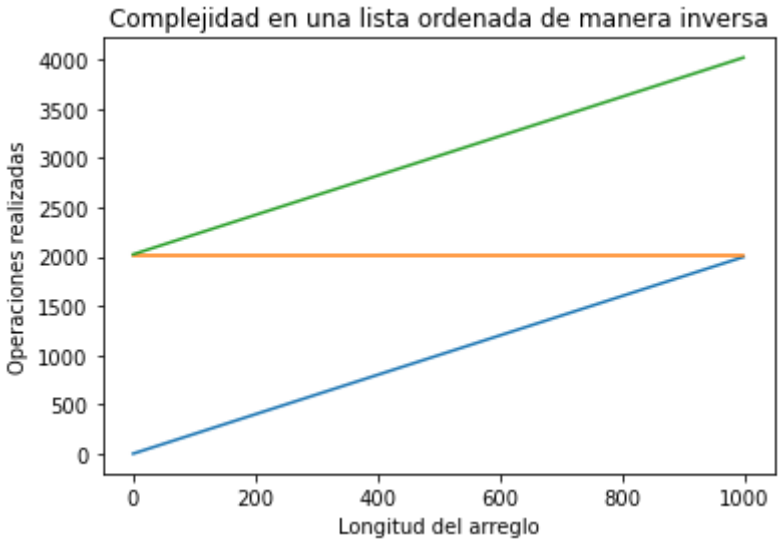
**Resultados Obtenidos.**

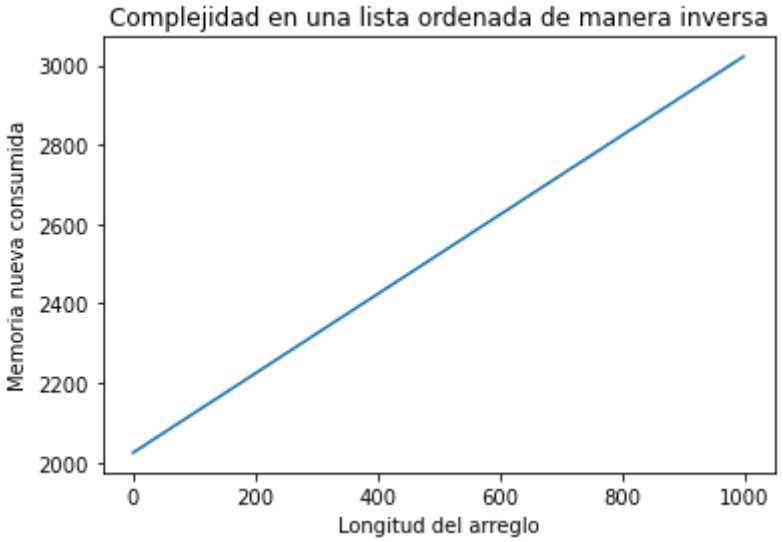
## CountingSort - Gráficas.
## Mejor caso de complejidad.



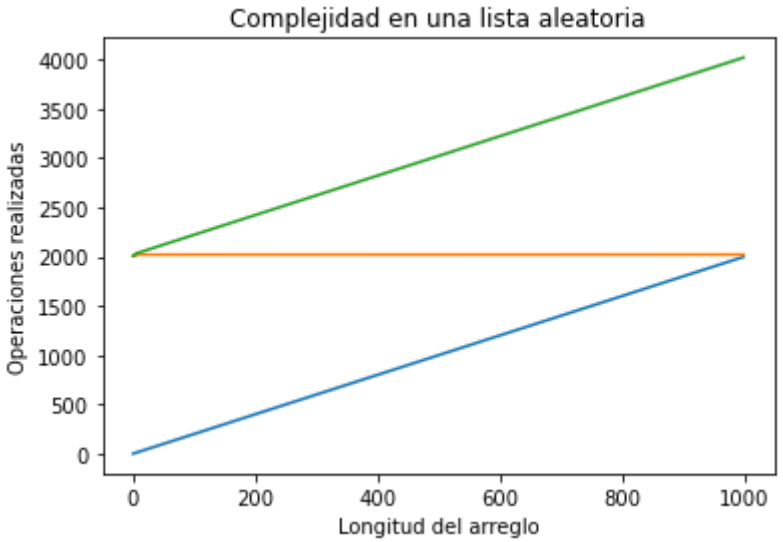



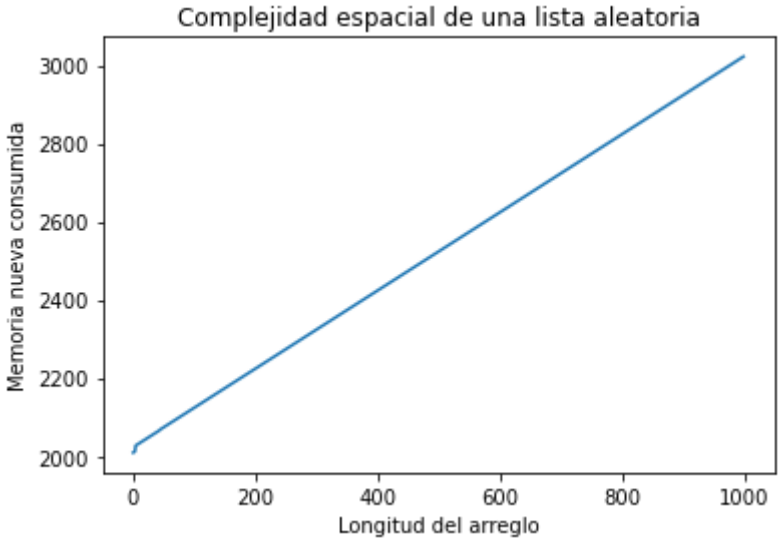
**Peor caso de complejidad.**



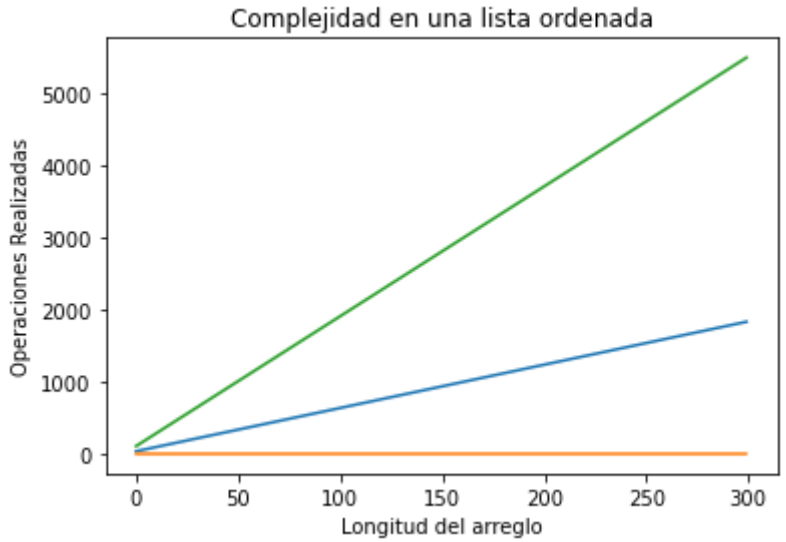



**Caso promedio.**



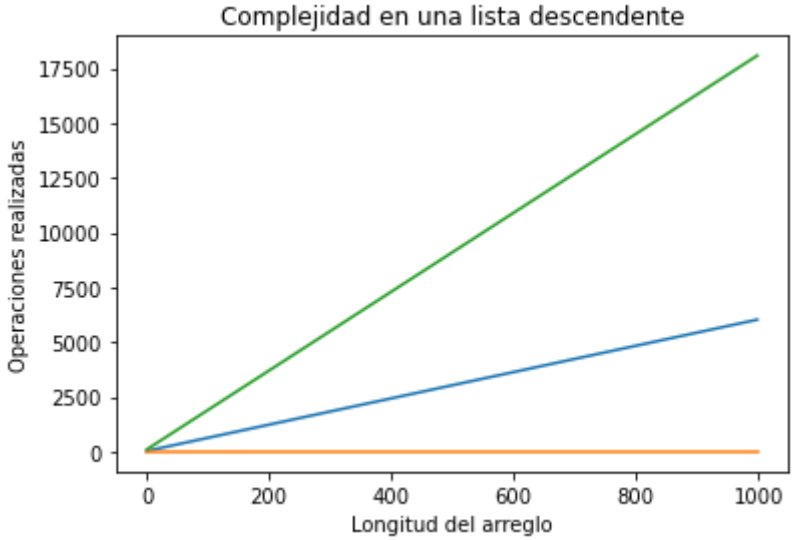



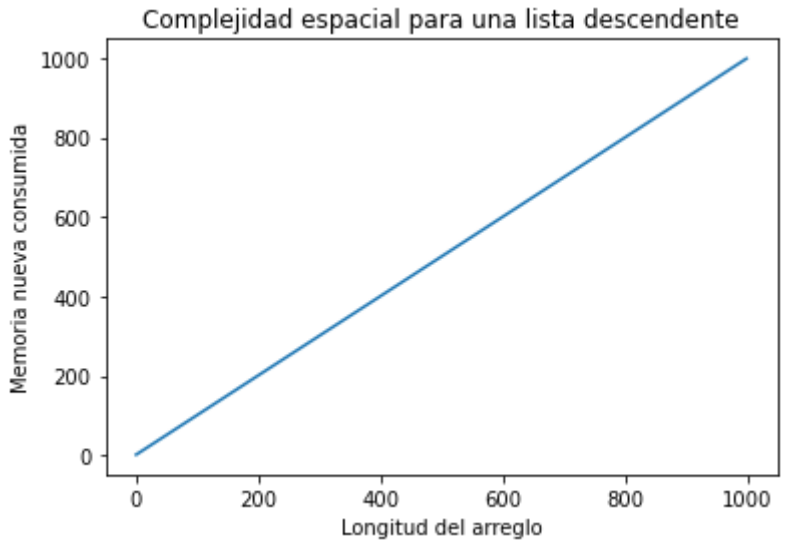
**RadixSort - Gráficas.**
**Mejor de complejidad.**



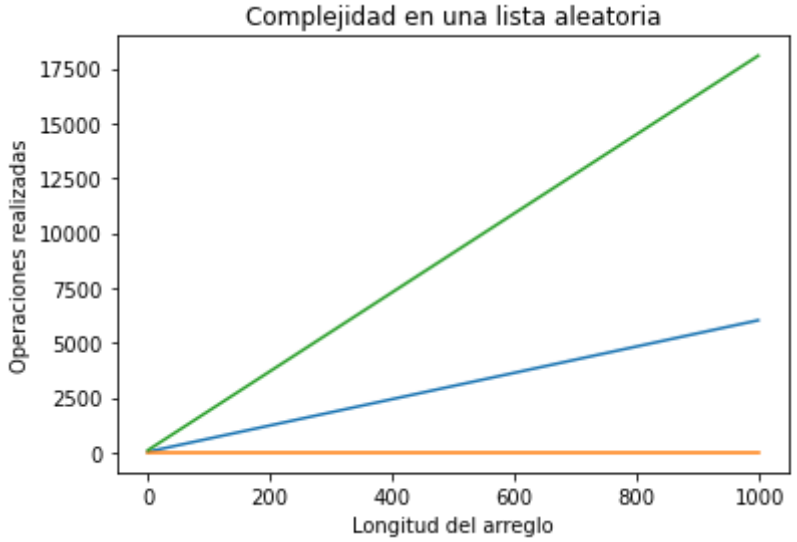


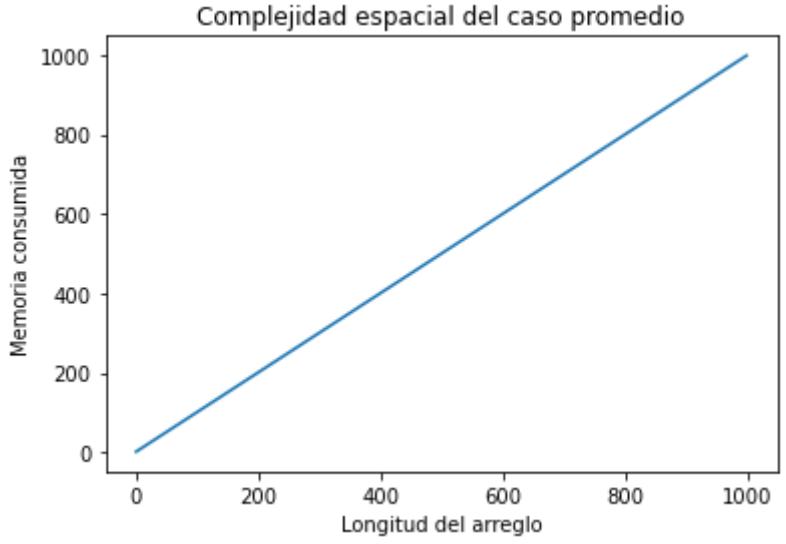
**Peor caso de complejidad.**






**Caso promedio.**



**Conclusiones.**

- Aguilar Martinez Erick Yair: Si se comprende a la perfección un algoritmo o a lo menos su esencia tenemos la oportunidad de adaptarlo a nuestros requerimientos, en nuestro caso ordenar objetos en base a diferentes parámetros.

- Casillas Herrera Leonardo Didier: En esta práctica sacamos la complejidad de CountingSort y RadixSort en donde las complejidades estaban en función de 2 variables diferentes que no tenían correlación y eso nos dio una complejidad lineal en donde las variables se sumaban además sacamos la complejidad espacial de cada algoritmo en el que vimos cuánta memoria se ocupaba para crear los arreglos necesarios al ordenar los elementos de nuestros conjuntos.