Estructura de Datos y Algoritmos 2

Semestre 2023-1

Profesor: Jorge Alberto Solano Galvez

Práctica 01. Algoritmos de búsqueda. Parte 1.

Grupo 06

Integrantes:
Aguilar Martinez Erick Yair
Casillas Herrera Leonardo Didier

Práctica 4

Objetivo:
Identificar el comportamiento y características de algunos algoritmos de búsqueda por comparación de llaves.

Actividades:

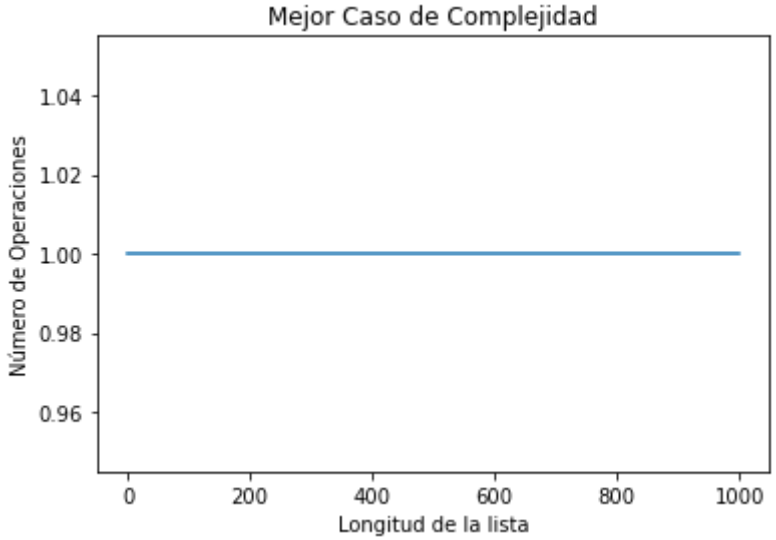
- Implementar el algoritmo iterativo y recursivo de búsqueda lineal en lenguaje Python para localizar alguna llave o valor en una secuencia de datos.
- Implementar el algoritmo iterativo y recursivo de búsqueda binaria en lenguaje Python para localizar alguna llave o valor en una secuencia de datos.

Instrucciones:

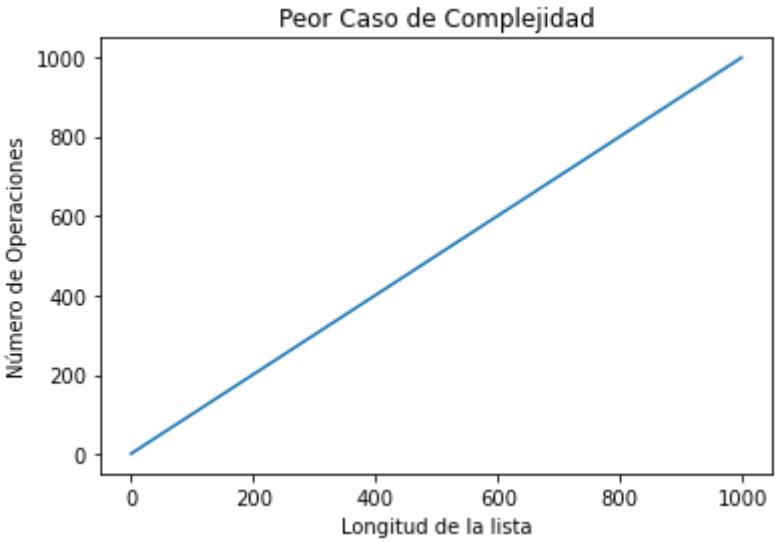
- Implementar búsqueda secuencial en lenguaje Python tanto iterativa como recursiva para encontrar un nodo (con el parámetro de búsqueda que desees).
- Implementar búsqueda binaria en lenguaje Python tanto iterativa como recursiva para encontrar un nodo (con el parámetro de búsqueda que desees). Antes de realizar la búsqueda se debe utilizar un método de ordenamiento directo (n^2) y uno logarítmico (n $*$ log(n)).
- Obtener la complejidad algorítmica para cada implementación (búsqueda secuencial, búsqueda binaria con ordenamiento directo y búsqueda binaria con ordenamiento logarítmico).
- Graficar el comportamiento de los algoritmos para el mejor, el peor y el caso promedio para cada implementación (búsqueda secuencial, búsqueda binaria con ordenamiento directo y búsqueda binaria con ordenamiento logarítmico).

Resultados Obtenidos.

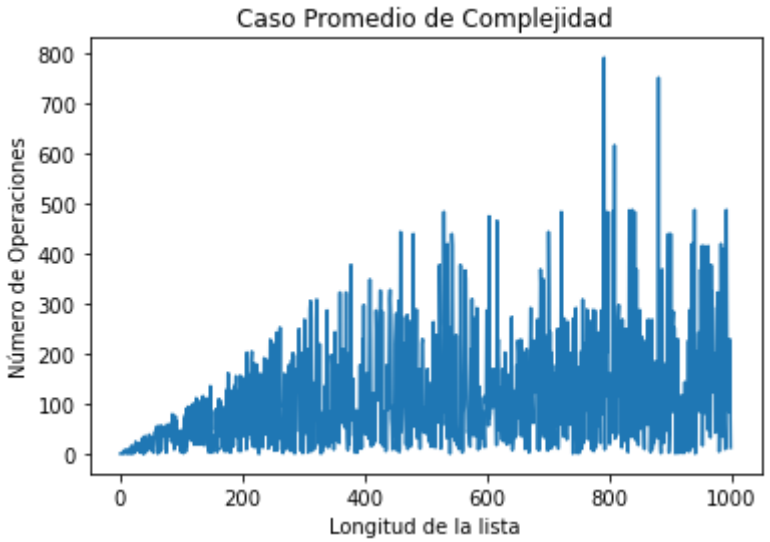
Búsqueda Lineal Iterativa.
Mejor caso de complejidad



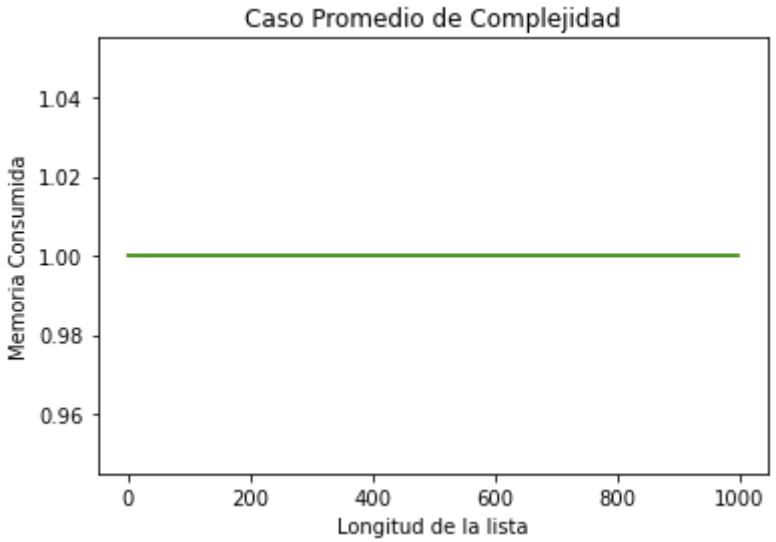
Peor caso de complejidad



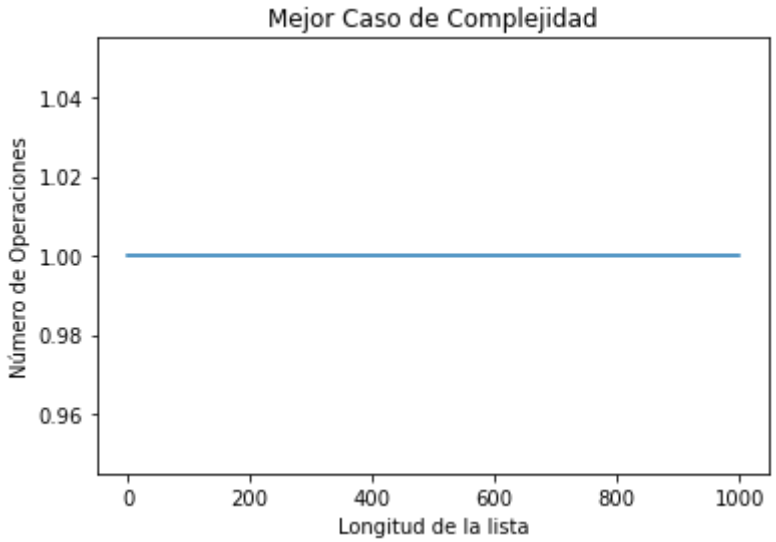
Caso promedio



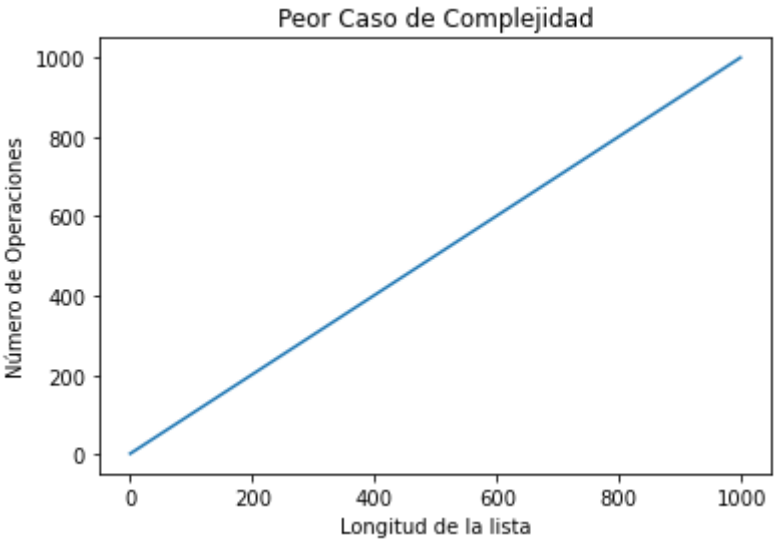
Complejidades Espaciales.
Caso promedio



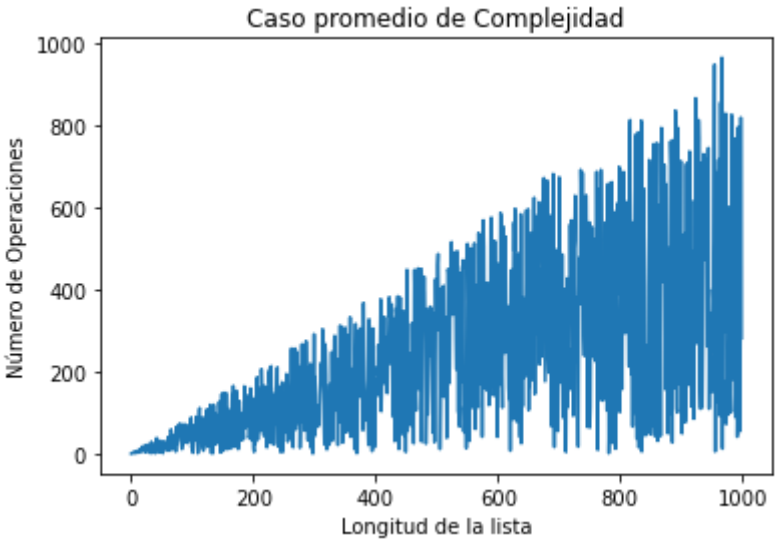
Búsqueda lineal recursiva.
Mejor caso de complejidad



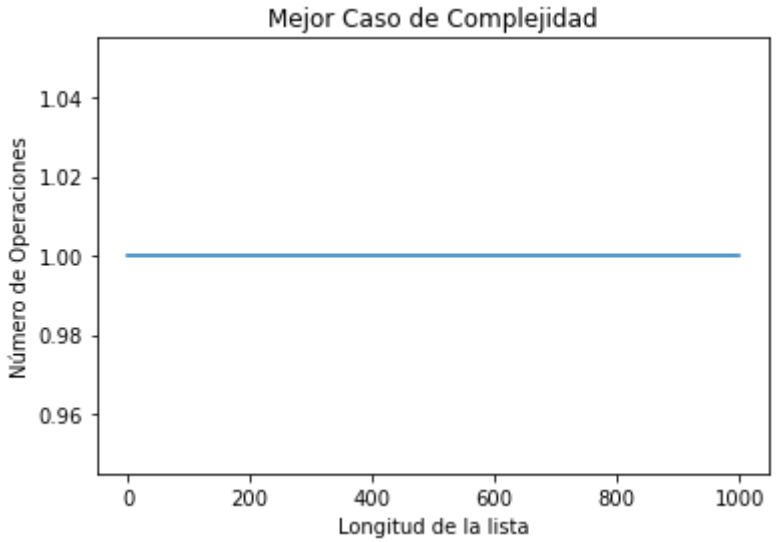
Peor caso de complejidad



Caso promedio

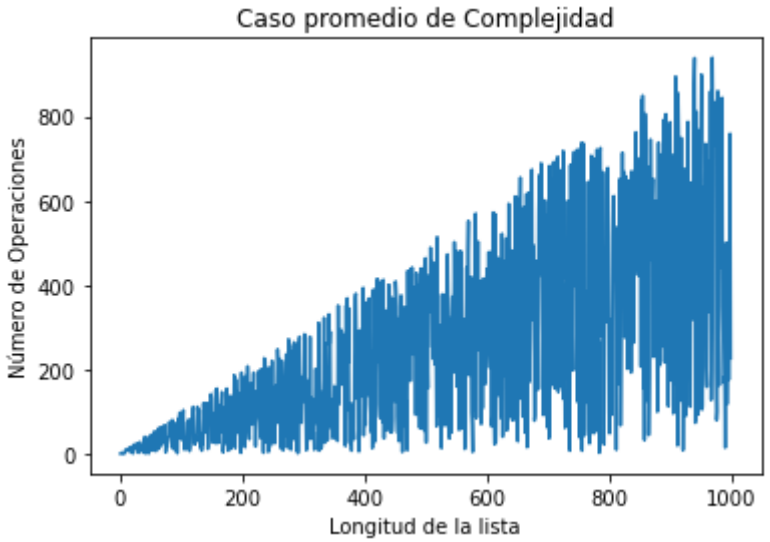


Mejor caso de complejidad

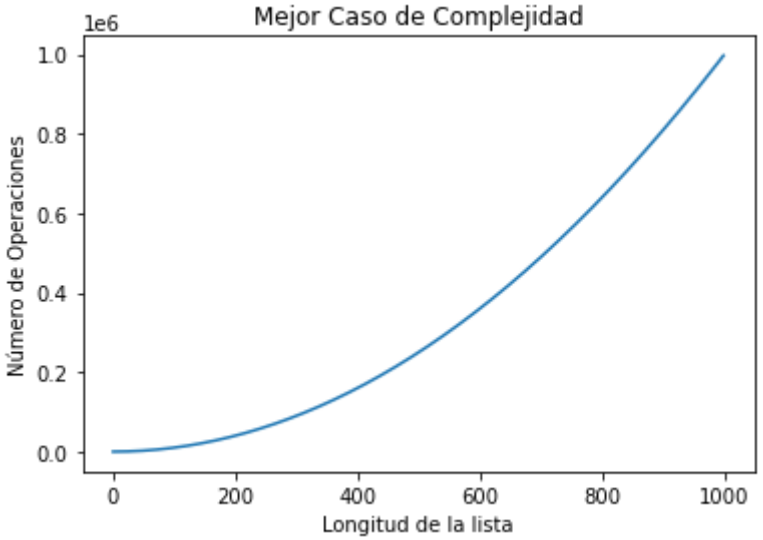


Peor caso de complejidad

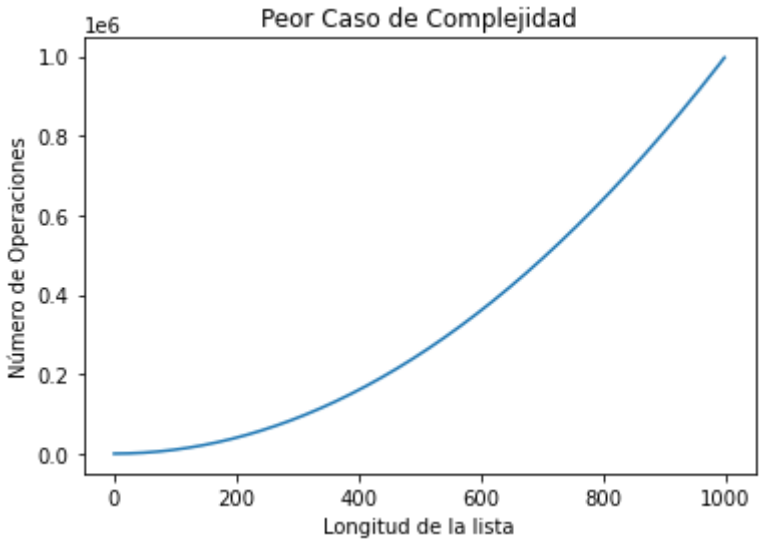
Caso promedio de complejidad



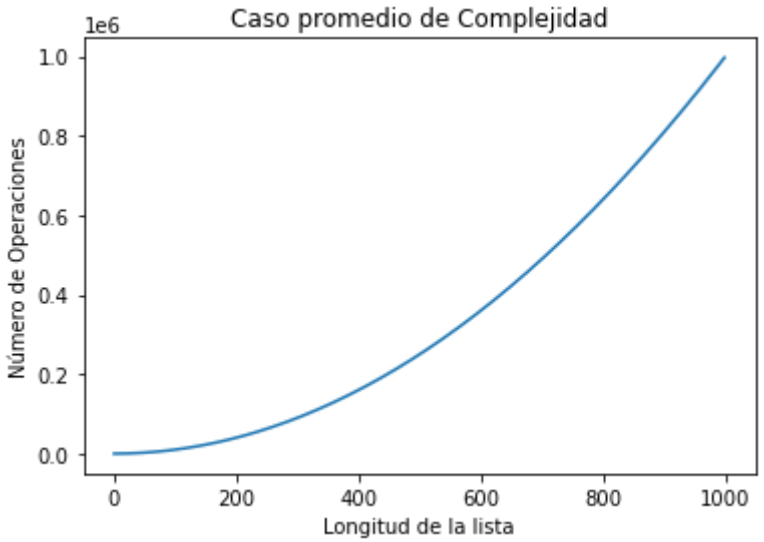
Búsqueda Binaria con ordenamiento burbuja.
Mejor caso de complejidad



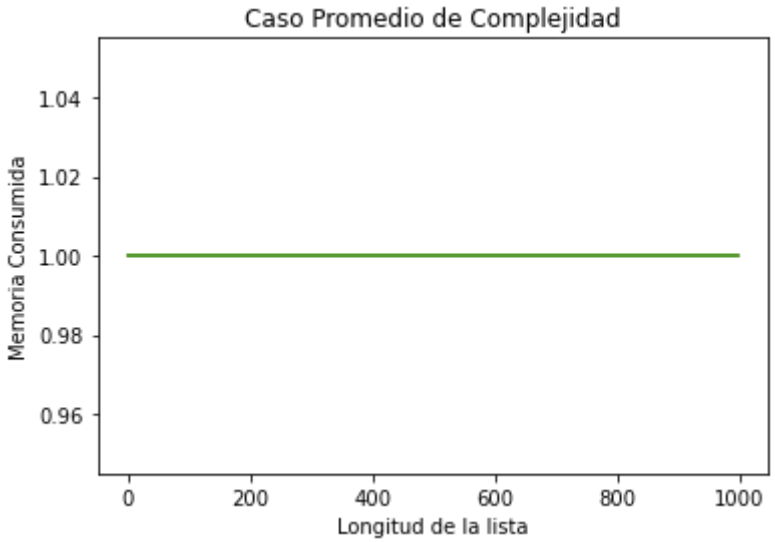
Peor caso de complejidad



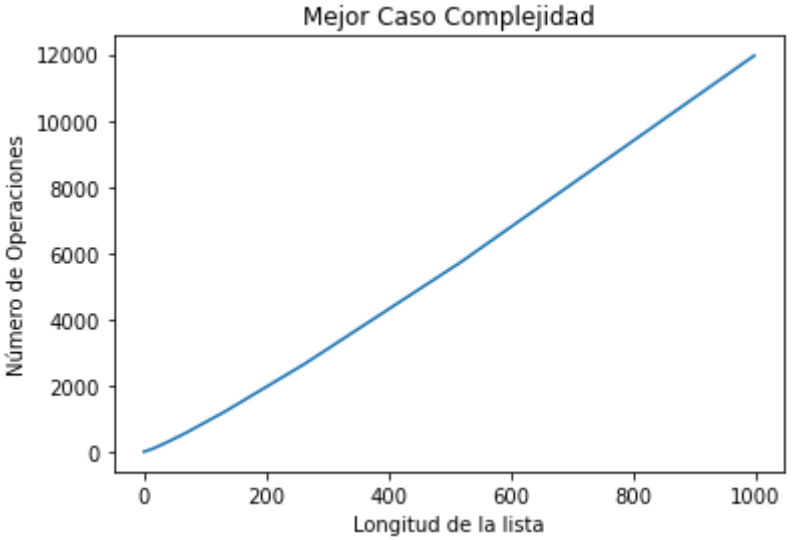
Caso promedio



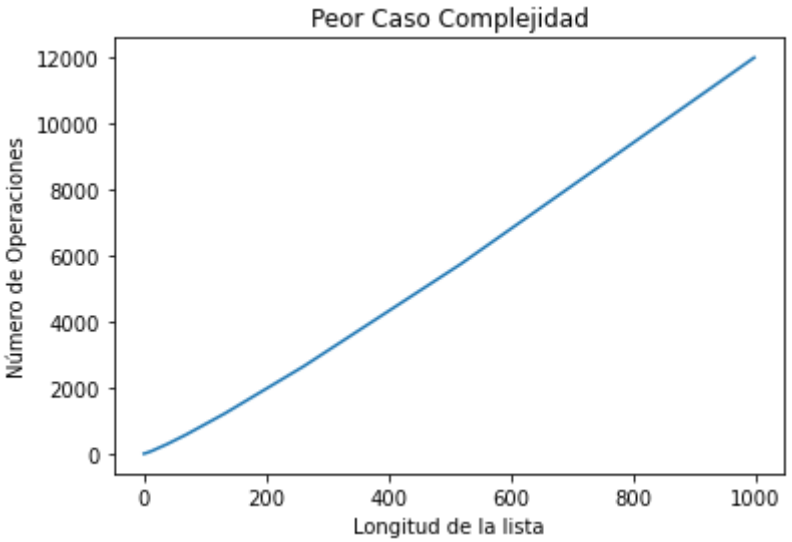
Caso promedio de complejidad



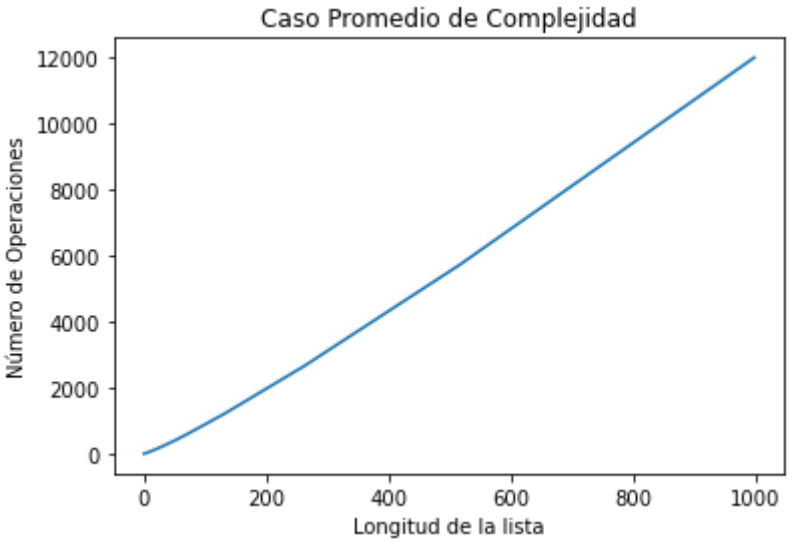
Búsqueda binaria iterativa con ordenamiento MergeSort.
Mejor caso de complejidad



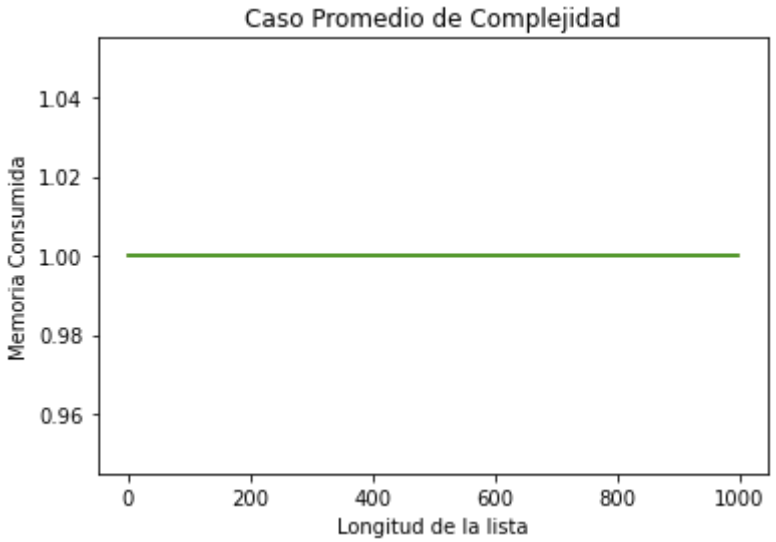
Peor caso de complejidad



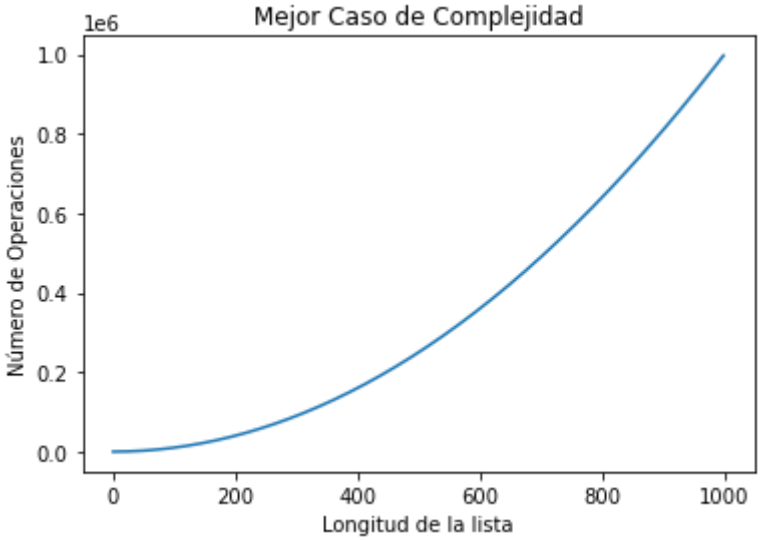
Caso promedio de complejidad



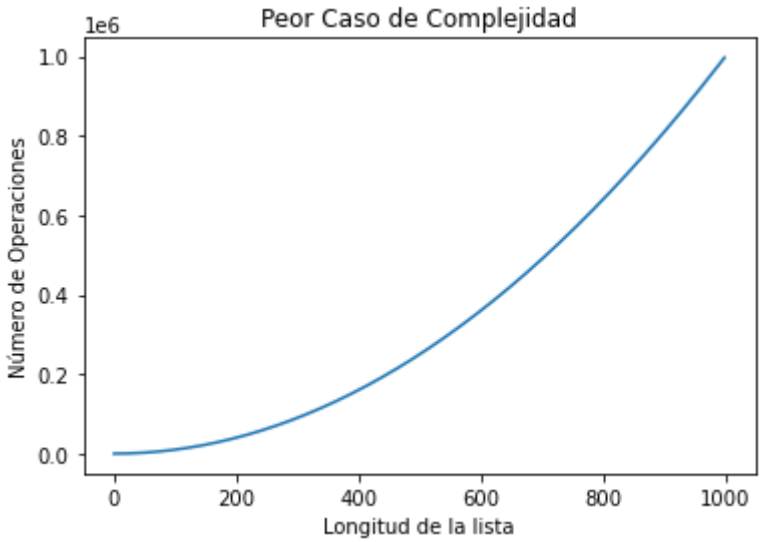
Complejidad Espacial.
Caso promedio



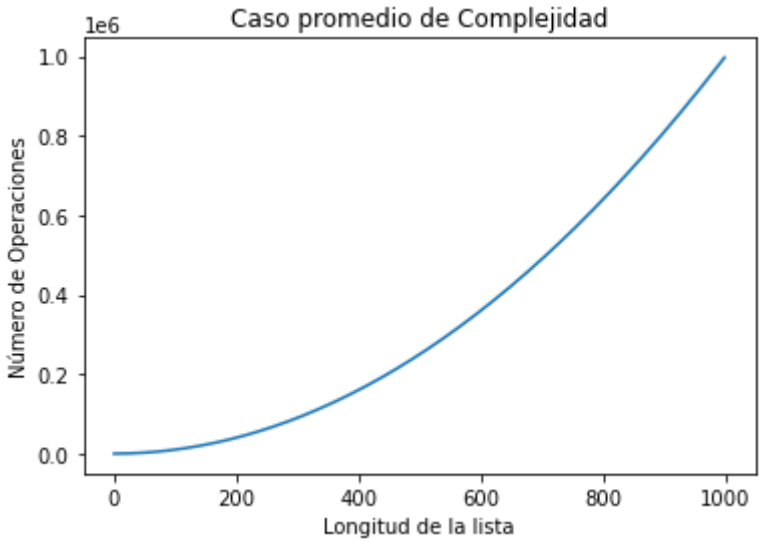
Busqueda binaria recursiva con ordenamiento burbuja.
Mejor caso de complejidad



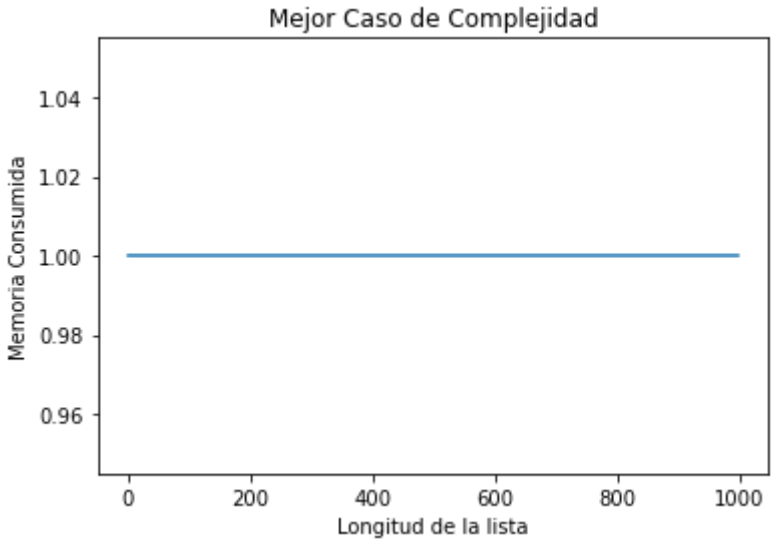
Peor caso de complejidad.



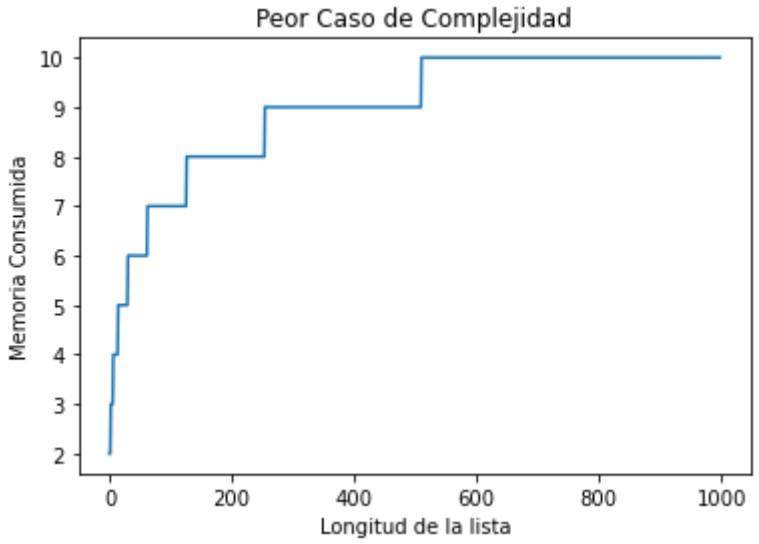
Caso promedio



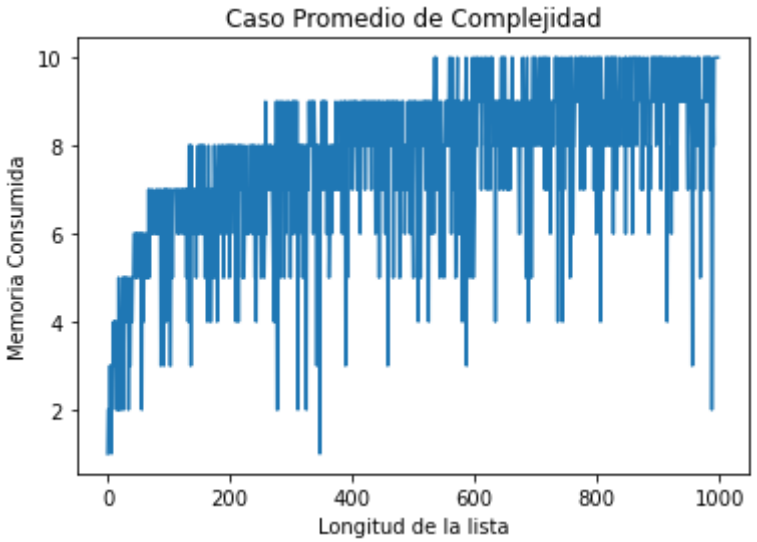
Complejidad Espacial.
Mejor caso de complejidad



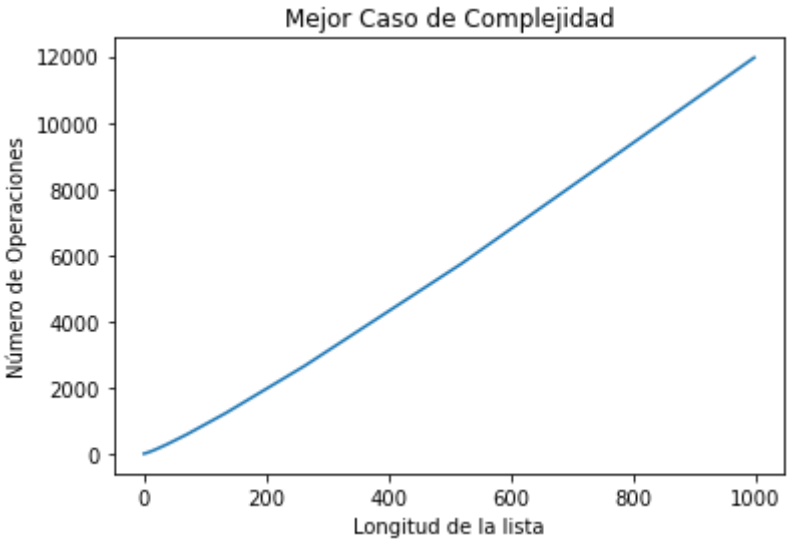
Peor caso de complejidad



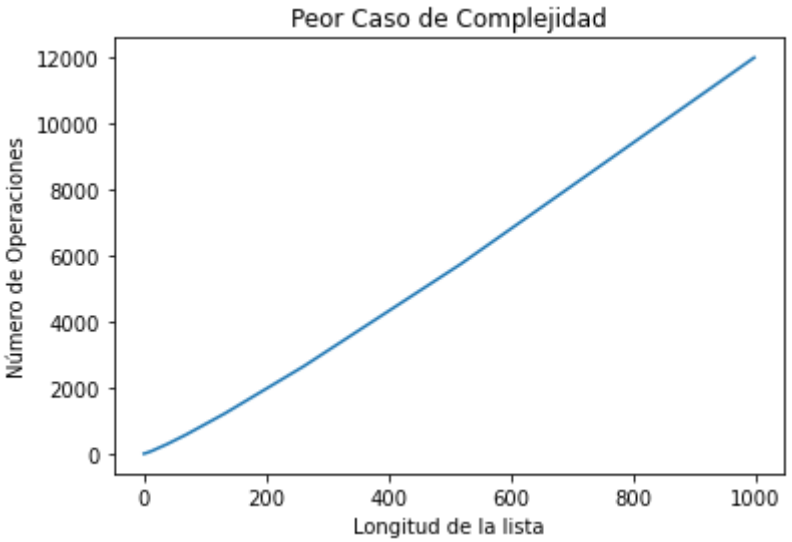
Caso promedio



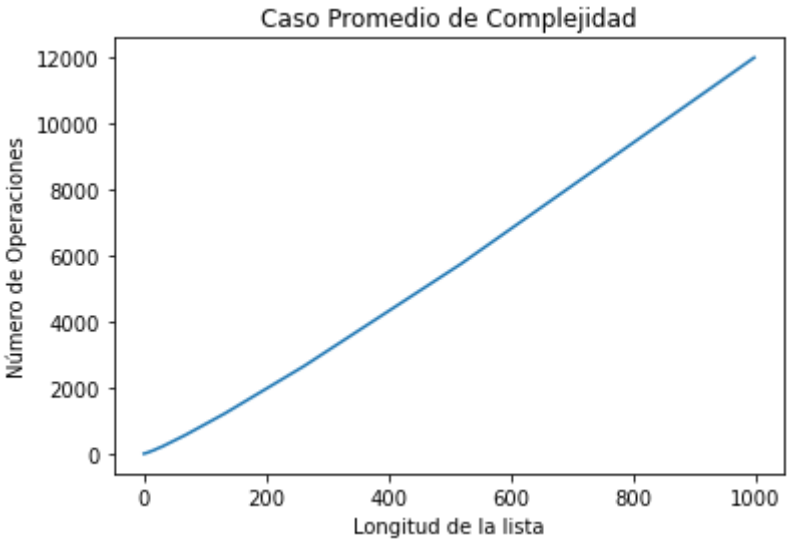
Busqueda binaria recursiva con ordenamiento MergeSort
Mejor caso de complejidad



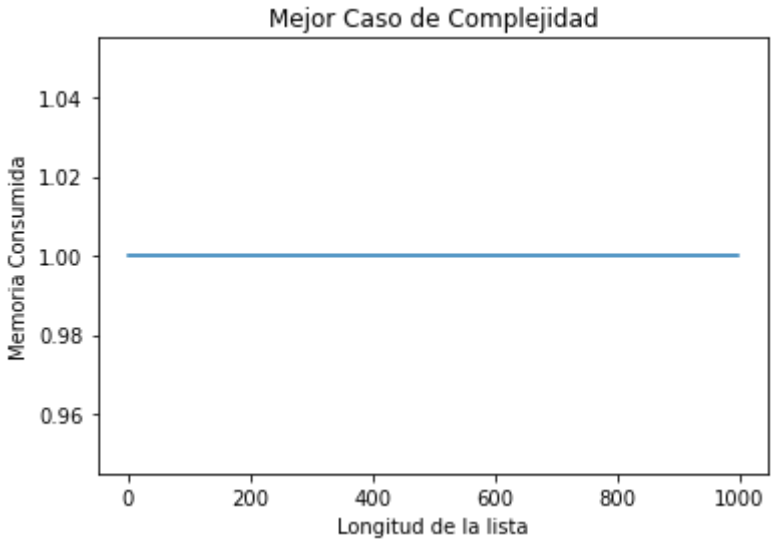
Peor caso de complejidad



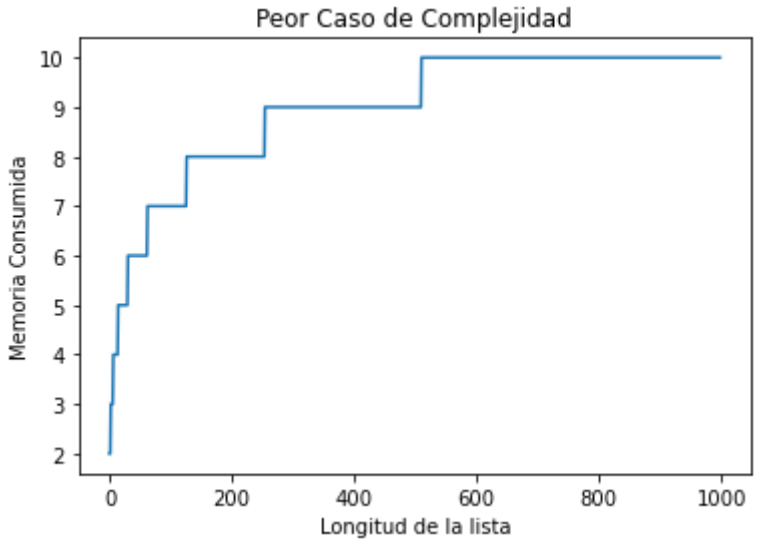
Caso promedio



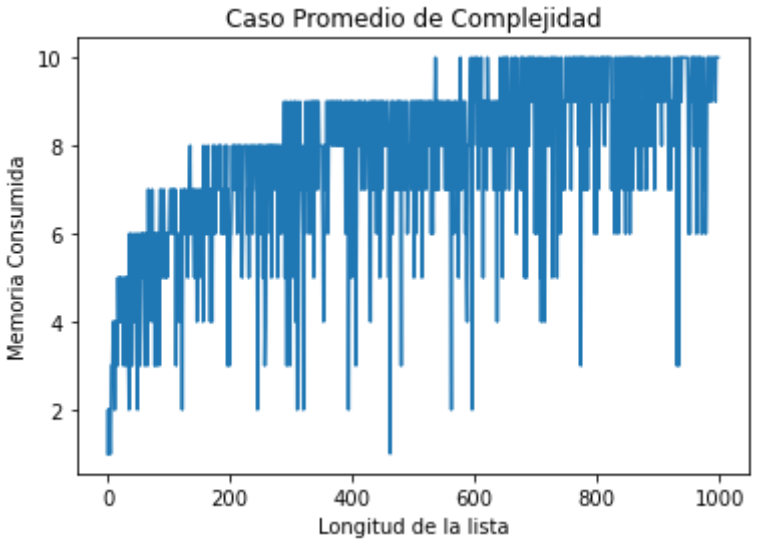
Complejidad Espacial.
Mejor caso de complejidad



Peor caso de complejidad



Caso promedio



Conclusiones.

Aguilar Martinez Erick Yair.

Muchas de las veces hay que considerar si es viable o no utilizar la búsqueda binaria ya que es un algoritmo muy eficiente pero el costo que representa el ordenamiento puede ser contraproducente.

La recursividad genera siempre una complejidad espacial mayor o igual a la temporal, esto porque las llamadas se acumulan en una pila lo cual puede ser estresante para las computadoras.

Casillas Herrera Leonardo Didier.

En estas implementaciones de los algoritmos de búsqueda lineal y binaria, las complejidades cambiaban dependiendo del caso, por eso debemos analizar nuestro conjunto para saber cuál algoritmo de búsqueda usar y así elegir el que sea más eficiente tanto en tiempo como en uso de memoria; dependiendo de lo que necesitemos debemos elegir un algoritmo lineal o binario y hacerlo iterativo o recursivo.