Algoritmo de Búsqueda y Ordenamiento

Alumnos:

Lautaro Pez- lautapez@gmail.com
Cristian Paolucci- cristianodixit@gmail.com

Materia: Programación I

Profesor: Julieta Trapé

Tutor: Sofia Raia

Fecha de entrega:

Comisión: 4

Indice

Introducción

En programación, los algoritmos de ordenamiento y búsqueda son fundamentales p ara manejar datos de forma eficiente. Estos algoritmos permiten organizar la información de manera lógica y buscar elementos dentro de estructuras con mayor rapidez.
Este trabajo analiza el uso de Insertion Sort, búsqueda lineal y búsqueda binaria en una lista de nombres de autores. Se incluye un código que demuestra su funcionamiento y utilidad.

Marco Teórico

Algoritmos de ordenamiento

Los algoritmos de ordenamiento reorganizan elementos según un criterio (alfabéticamente, por número, etc) En este trabajo se utiliza:

Insertion Sort: Construye progresivamente una lista ordenada. Toma un elemento a la vez y lo inserta en la posición correcta dentro de los ya ordenados. Es eficiente para listas pequeñas o casi ordenadas.

Algoritmo de Búsqueda

Sirven para encontrar un valor específico dentro de una colección. Se aplican dos métodos:

Búsqueda Lineal: Recorre uno a uno todos los elementos hasta hallar el buscado. Funciona sobre cualquier lista, ordenada o no.

Búsqueda binaria: Solo para listas previamente ordenadas. Divide el espacio de búsqueda a la mitad en cada iteración, siendo más eficiente que la búsqueda lineal en listas grandes.

Funcionamiento de Búsqueda binaria:

1. Se identifica el elemento central de la lista
2. Se compara con el objetivo:
   Si es igual, se devuelve su posición
   Si es menor, se busca en la mitad derecha
   Si es mayor, en la mitad izquierda
3. El proceso se repite hasta encontrar el elemento o agotar la lista

Caso Práctico

Se desarrollo un programa que permite:

Ordenar una lista de nombres de autores utilizando Insertion Sort.

Buscar un autor específico mediante Búsqueda Lineal y Búsqueda Binaria.

```
# Insertion Sort para ordenar nombres
def insertion_sort(lista):
    for i in range(1, len(lista)):
        actual = lista[i]
        j = i - 1
        while j >= 0 and actual.lower() < lista[j].lower():
            lista[j + 1] = lista[j]
            j -= 1
        lista[j + 1] = actual
    return lista

# Búsqueda lineal
def busqueda_lineal(lista, objetivo):
    for i in range(len(lista)):
        if lista[i].lower() == objetivo.lower():
            return i
    return -1

# Búsqueda binaria
def busqueda_binaria(lista, objetivo):
    izquierda = 0
    derecha = len(lista) - 1
    objetivo = objetivo.lower()
```

```
    while izquierda <= derecha:
        medio = (izquierda + derecha) // 2
        if lista[medio].lower() == objetivo:
            return medio
        elif lista[medio].lower() < objetivo:
            izquierda = medio + 1
        else:
            derecha = medio - 1
    return -1

# Lista de autores
autores = ["Cortázar", "Borges", "Neruda", "Paz", "García Márquez"]

print("Lista original de autores:")
print(autores)

# Ordenar
autores_ordenados = insertion_sort(autores.copy())
print("\nLista ordenada alfabéticamente:")
print(autores_ordenados)

# Búsqueda
objetivo = "Paz"

# Lineal
indice_lineal = busqueda_lineal(autores, objetivo)
print(f"\nResultado búsqueda lineal de '{objetivo}':")
if indice_lineal != -1:
    print(f"Encontrado en la posición {indice_lineal} (lista original)")
else:
    print("No encontrado.")

# Binaria
indice_binaria = busqueda_binaria(autores_ordenados, objetivo)
print(f"\nResultado búsqueda binaria de '{objetivo}':")
if indice_binaria != -1:
    print(f"Encontrado en la posición {indice_binaria} (lista ordenada)")
else:
    print("No encontrado.")
```

Metodología Utilizada

El desarrollo del proyecto se llevó a cabo en las siguientes etapas:

Revisión teórica de algoritmos de búsqueda y ordenamiento
Implementación del código
Pruebas con una lista de autores
Análisis de los resultados obtenidos para validar el funcionamiento de los algoritmos
Redacción del informe final con la documentación del proceso

Resultados Obtenidos

1. La lista de autores fu ordenada correctamente de forma alfabética usando Insertion Sort
2. La búsqueda lineal identificó correctamente la posición del autor “Paz” en la lista original
3. La búsqueda binaria localizó correctamente a “Paz” en la lista ya ordenada.
4. Se observaron diferencias en la forma de operar de ambas búsquedas, destacando la necesidad de una lista ordenada para usar búsqueda binaria.

Conclusiones

Este trabajo permitió comprender de manera práctica la implementación y utilidad de tres algoritmos:

Insertion Sort efectivo para listas pequeñas como la utilizada

Búsqueda Lineal útil para listas sin ordenar

Búsqueda Binaria muy eficiente, pero depende de una lista previamente ordenada

Bibliografía

Material de la Cátedra
Búsquedas en Google
Youtube
IA

Anexo

```
/usr/local/bin/python3 "/Users/mercedeslobo/Desktop/TP Integrador Programacion Perez-Paolucci/Tp-Integrador-Busqueda-Ordenamiento.py"
mercedeslobo@MacBook-Pro-de-Mercedes ~ % /usr/local/bin/python3 "/Users/mercedeslobo/Desktop/TP Integrador Programacion P
erez-Paolucci/Tp-Integrador-Busqueda-Ordenamiento.py"
Lista original de autores:
['Cortázar', 'Borges', 'Neruda', 'Paz', 'García Márquez']

Lista ordenada alfabéticamente:
['Borges', 'Cortázar', 'García Márquez', 'Neruda', 'Paz']

Resultado búsqueda lineal de 'Paz':
Encontrado en la posición 3 (lista original)

Resultado búsqueda binaria de 'Paz':
Encontrado en la posición 4 (lista ordenada)
mercedeslobo@MacBook-Pro-de-Mercedes ~ %
```

Resultado en consola del programa: