Trabajo Práctico - Programación I

Título: Algoritmos de Búsqueda y Ordenamiento en Python aplicados a una lista de empleados

Alumno: Juan Manuel Gomez
Materia: Programación I
Profesor/a: Ariel Enferrel

Índice

---

1. Introducción

En este trabajo se desarrollan y analizan dos algoritmos fundamentales en el campo de la programación: los algoritmos de búsqueda y de ordenamiento. Se eligió este tema por su gran utilidad en la manipulación de datos, ya que permite organizar y acceder a la información de forma eficiente.

El objetivo principal es comprender el funcionamiento de estos algoritmos y aplicarlos en un lenguaje de programación como Python, en este caso usando una lista de empleados como ejemplo práctico.

Tambien se busca reflexionar sobre las ventajas y limitaciones de estos métodos básicos, evaluando su rendimiento y aplicabilidad en distintos contextos. El trabajo apunta no solo a resolver un problema técnico, sino también a formar una base sólida para afrontar futuros desafíos en programación.

---

2. Marco Teórico

Algoritmos de ordenamiento

Un algoritmo de ordenamiento tiene como propósito organizar los elementos de una lista según el criterio deseado (por ejemplo, orden alfabético o numérico ascendente). Entre los métodos más conocidos se encuentra el **Bubble Sort** o "ordenamiento burbuja", que funciona comparando elementos de a pares y desplazando el mayor hacia el final en cada pasada. Este proceso se repite hasta que toda la lista quede ordenada.

Algoritmos de búsqueda

Los algoritmos de búsqueda permiten localizar un valor dentro de una estructura de datos. En este trabajo se utiliza la **búsqueda lineal**, que consiste en recorrer la lista elemento por elemento hasta encontrar una coincidencia o hasta que se llega al final. Su principal ventaja es que no requiere que los datos estén ordenados, aunque su desventaja es que puede ser lento en listas largas.

Comparación y elección

Elegí estos dos algoritmos por su sencillez y claridad conceptual, ideales para comenzar a trabajar con estructuras básicas.

---

3. Caso Práctico

Planteo del problema

Se simula una situación real en una empresa: se cuenta con una lista de empleados que incluye su nombre, el sector donde trabajan y su número de legajo. Se necesitan realizar dos operaciones clave sobre esa lista:

1. Devuelve la lista de empleados ordenada alfabéticamente.
2. Busca un empleado específico por su nombre y obtiene tanto su antigüedad en la empresa, como su sector y su legajo.

Para resolver esto implemente en Python dos algoritmos básicos: Bubble Sort y búsqueda lineal.

Implementación en Python:

```python
empleados = [
    {"nombre": "Laura", "sector": "RRHH", "legajo": 1001, "antiguedad": 15},
    {"nombre": "Carlos", "sector": "Sistemas", "legajo": 1002, "antiguedad": 13},
    {"nombre": "Ana", "sector": "Contabilidad", "legajo": 1003, "antiguedad": 6},
    {"nombre": "Martín", "sector": "Logística", "legajo": 1004, "antiguedad": 24},
    {"nombre": "Bruno", "sector": "Ventas", "legajo": 1005, "antiguedad": 2},
    {"nombre": "Zoe", "sector": "Compras", "legajo": 1006, "antiguedad": 10},
    {"nombre": "Florencia", "sector": "Marketing", "legajo": 1007, "antiguedad": 9},
    {"nombre": "Emilia", "sector": "Atención al cliente", "legajo": 1008, "antiguedad": 6},
    {"nombre": "Santiago", "sector": "Sistemas", "legajo": 1009, "antiguedad": 7},
    {"nombre": "Valentina", "sector": "Contabilidad", "legajo": 1010, "antiguedad": 8},
    {"nombre": "Ignacio", "sector": "RRHH", "legajo": 1011, "antiguedad": 8},
    {"nombre": "Julieta", "sector": "Marketing", "legajo": 1012, "antiguedad": 12},
    {"nombre": "Nicolás", "sector": "Logística", "legajo": 1013, "antiguedad": 25},
    {"nombre": "Camila", "sector": "Compras", "legajo": 1014, "antiguedad": 8},
    {"nombre": "Marcos", "sector": "Ventas", "legajo": 1015, "antiguedad": 42},
    {"nombre": "Lucía", "sector": "Sistemas", "legajo": 1016, "antiguedad": 30},
    {"nombre": "Diego", "sector": "RRHH", "legajo": 1017, "antiguedad": 6},
    {"nombre": "Andrea", "sector": "Contabilidad", "legajo": 1018, "antiguedad": 21},
    {"nombre": "Franco", "sector": "Compras", "legajo": 1019, "antiguedad": 11},
    {"nombre": "Rocío", "sector": "Atención al cliente", "legajo": 1020, "antiguedad": 3},
    {"nombre": "Pedro", "sector": "Logística", "legajo": 1021, "antiguedad": 14},
    {"nombre": "Sofía", "sector": "Marketing", "legajo": 1022, "antiguedad": 12},
    {"nombre": "Leandro", "sector": "Ventas", "legajo": 1023, "antiguedad": 11},
    {"nombre": "Milagros", "sector": "RRHH", "legajo": 1024, "antiguedad": 14},
    {"nombre": "Tomás", "sector": "Sistemas", "legajo": 1025, "antiguedad": 2},
    {"nombre": "Natalia", "sector": "Contabilidad", "legajo": 1026, "antiguedad": 5},
    {"nombre": "Federico", "sector": "Compras", "legajo": 1027, "antiguedad": 3},
    {"nombre": "Daniela", "sector": "Marketing", "legajo": 1028, "antiguedad": 1},
    {"nombre": "Esteban", "sector": "Logística", "legajo": 1029, "antiguedad": 6},
    {"nombre": "Gimena", "sector": "Ventas", "legajo": 1030, "antiguedad": 2}
]
```

```python
def bubble_sort_empleados(lista):
    n = len(lista)
    for i in range(n):
        for j in range(0, n - i - 1):
            if lista[j]["nombre"] > lista[j + 1]["nombre"]:
                lista[j], lista[j + 1] = lista[j + 1], lista[j]

def buscar_empleado(lista, nombre):
    for empleado in lista:
        if empleado["nombre"].lower() == nombre.lower():
            return empleado
    return None

bubble_sort_empleados(empleados)
```

```python
while True:
    print("\nOpciones:")
    print("1. Mostrar lista de empleados.")
    print("2. Buscar empleado:")
    print("3. Salir")

    opcion = input("Elegí una opción: ")

    if opcion == "1":
        print("\nLista de empleados:")
        for emp in empleados:
            print(f"{emp['nombre']} - {emp['sector']} - Legajo {emp['legajo']} - {emp['antiguedad']} años")

    elif opcion == "2":
        nombre_buscado = input("Ingresá el nombre del empleado que querés buscar: ")
        resultado = buscar_empleado(empleados, nombre_buscado)
        if resultado:
            print(f"\nInformacion del empleado: {resultado['nombre']} trabaja en {resultado['sector']}, su legajo es {resultado['legajo']} y hace {resultado['antiguedad']} años traba
        else:
            print(f"\nEl nombre {nombre_buscado} no pertenece a ningun empleado de la empresa.")

    elif opcion == "3":
        print("Saliendo del programa.")
        break
    else:
        print("La opción no es válida. Intentalo de nuevo porfavor.")
```

Justificación de la elección

Decidi trabajar con una lista de diccionarios para reflejar mejor un entorno real, donde cada persona tiene datos asociados. Esto permite aplicar los mismos algoritmos pero en un contexto más profesional.

---

4. Metodología Utilizada

1. Se investigó el funcionamiento de los algoritmos de ordenamiento y búsqueda.
2. Se definió una estructura de datos adecuada para representar empleados (lista de diccionarios).
3. Se codificaron las funciones de ordenamiento Bubble Sort y búsqueda lineal.
4. Se realizo la lista de 30 empleados con datos variados.
5. Se agregó un menú interactivo para que el usuario elija entre ver la lista, buscar un nombre o finalizar con el programa.
6. Se realizaron pruebas con distintos nombres y opciones del menú.

---

5. Resultados Obtenidos

- El programa ordena correctamente la lista de empleados por nombre.
- Se puede buscar cualquier empleado y obtener su sector, legajo y antigüedad.
- El menú mejora la interacción con el usuario, haciendo el programa más amigable.
- Se realizaron pruebas con varios nombres y no se detectaron errores.

---

6. Conclusiones

Este trabajo me ayudó un montón a entender cómo aplicar la teoría de los algoritmos en un caso concreto. Al principio parecía sencillo, pero al ir armando todo (los empleados, la búsqueda, el ordenamiento y después el menú), fui viendo que cada parte tenía su lógica y su desafío.

Sumar un menú con opciones también me sirvió porque sentí que estaba armando algo mas de verdad, no solo un ejemplo para hacer una vez. Además, pude probar qué pasa si hay muchos empleados, si se busca uno que no existe, o si el usuario pone mal el nombre. Todo eso me hizo pensar cómo mejorar el código a futuro tambien.

Aunque Bubble Sort no es el algoritmo más eficiente, me pareció perfecto para practicar porque se entiende bien lo que hace paso a paso. Y con la búsqueda lineal pasa algo parecido: no será la más rápida, pero es muy clara para empezar.

En resumen, siento que hice un laburo completo, que me sirvió tanto para practicar Python como para pensar en la lógica detrás de cada decisión que tomé. Fue un buen ejercicio para crecer como programador.

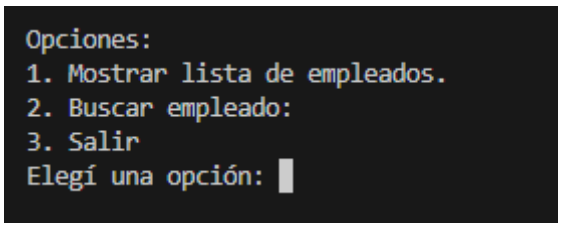
---

7. Bibliografía

- Python Software Foundation. (2024). *Python 3 Documentation*. https://docs.python.org/3/
- Sweigart, A. (2019). *Automate the Boring Stuff with Python*. No Starch Press.
- Khan Academy. *Algoritmos de búsqueda y ordenamiento*. https://es.khanacademy.org/
- W3Schools. *Python Lists and Dictionaries*. https://www.w3schools.com/python/
- Stack Overflow. Consultas realizadas sobre estructuras de datos y control de errores.

---

8. Anexos



```
Elegí una opción: 1

Lista de empleados:
Ana - Contabilidad - Legajo 1003 - 6 años
Andrea - Contabilidad - Legajo 1018 - 21 años
Bruno - Ventas - Legajo 1005 - 2 años
Camila - Compras - Legajo 1014 - 8 años
Carlos - Sistemas - Legajo 1002 - 13 años
Daniela - Marketing - Legajo 1028 - 1 años
Diego - RRHH - Legajo 1017 - 6 años
Emilia - Atención al cliente - Legajo 1008 - 6 años
```

```
Elegí una opción: 2
Ingresá el nombre del empleado que querés buscar: bruno

Informacion del empleado: Bruno trabaja en Ventas, su legajo es 1005 y hace 2 años trabaja en la empresa.
```

```
Elegí una opción: 3
Saliendo del programa.
```