Plantilla para Trabajos Integradores – Programación I

Datos Generales

Título del trabajo : "Algoritmos de Búsqueda y Ordenamiento con listas y diccionarios en Python"

- **Alumnos**: Luciano Olmedo

  Saul Hillar (saulhillar@outlook.es)
- **Materia**: Programación I
- **Profesor/a**: Ariel Enferrel
- **Fecha de Entrega**: 09/06/2025

Índice

---

1. Introducción

El trabajo se centra en la temática de búsqueda y ordenamiento, que son conceptos fundamentales dentro del área de la programación. El grupo decidió hacer este tema debido a la importancia que tiene en el desarrollo de software, ya que ambos procesos permiten la optimización de la gestión de datos, mejora la eficiencia de los programas y facilita la interacción con el usuario. Además, se trata de uno de los ejes trabajados en profundidad durante la cursada, lo que hace que tenga una aplicación de los conocimientos adquiridos.

La búsqueda y el ordenamiento son herramientas clave para resolver problemas cotidianos de la informática, como organizar información, acceder rápidamente a registros, o filtrar resultados. Desde sistemas de archivos hasta motores de búsqueda, estas operaciones son

parte esencial del funcionamiento interno de un montón de aplicaciones reales que usamos cotidianamente.

El objetivo general de este trabajo es desarrollar una solución práctica en Python que permita aplicar distintos algoritmos de ordenamiento (como burbuja, selección o inserción) y búsqueda (lineal y binaria) sobre una base de datos de videojuegos. Se busca que el desarrollo sea funcional, interactivo y que demuestre, mediante ejemplos concretos, cómo varía el comportamiento y la utilidad de cada algoritmo según el tipo de dato y la operación deseada.

---

2. Marco Teórico

Este apartado contiene la fundamentación conceptual del tema tratado. Debe incluir definiciones, clasificaciones, jerarquías, estructuras y sintaxis si corresponde.

- Incluir gráficos, esquemas o tablas si son útiles para la comprensión.
- Mencionar autores o documentación oficial si se toman definiciones o explicaciones.
- Para temas técnicos, se recomienda describir cómo se implementa el concepto en el lenguaje de programación trabajado (ej. Python, Java, etc.).

---

3. Caso Práctico

Se planteó la necesidad de ordenar una lista de videojuegos según su año de lanzamiento, de forma ascendente. Esta operación es fundamental para que luego pueda aplicarse búsqueda binaria y para ofrecer al usuario una visión cronológica clara de los juegos disponibles. Se decidió implementar el algoritmo de ordenamiento burbuja (Bubble Sort), y adicionalmente ordenamiento rápido (QuickSort).

```python
#metodo de ordenamiento por burbuja
def bubble_sort(juegos, key):
    n = len(juegos)
    for i in range(n):
        # Últimos i elementos ya están en su lugar
        for j in range(0, n - i - 1):
            if juegos[j][key] > juegos[j + 1][key]:
                temp = juegos[j]
                juegos[j] = juegos[j + 1]
                juegos[j + 1] = temp
    # Retornamos la lista ordenada
    return juegos
```

```python
#ordenamiento quick sort
def quick_sort(juegos, key):
    if len(juegos) <= 1:
        return juegos
    left = []
    middle = []
    right = []
    # Elegimos un pivote, en este caso el elemento del medio
    pivot = juegos[len(juegos) // 2][key]
    for x in juegos:
        if x[key] < pivot:
            left.append(x)
        elif x[key] > pivot:
            right.append(x)
        else:
            middle.append(x)
    return quick_sort(left, key) + middle + quick_sort(right, key)
```

Se eligió BubbleSort, y QuickSort por su simplicidad y porque permite visualizar claramente cómo se intercambian los elementos. Aunque existen algoritmos más eficientes como QuickSort que también incorporamos, para listas pequeñas. Además, su implementación permite trabajar con el uso de estructuras repetitivas (for) y condicionales (if), pilares básicos de la programación.

Se probó la función con una lista real de videojuegos desordenados. Tras ejecutar la función, se observó que los juegos aparecían correctamente ordenados. También se imprimió la lista antes y después del ordenamiento para verificar visualmente el efecto del algoritmo. Este orden luego permitió ejecutar correctamente una búsqueda binaria mediante la clave que pasó el usuario, validando así su utilidad práctica dentro del programa.

---

4. Metodología Utilizada

Describe los pasos seguidos durante el desarrollo del trabajo.

El trabajo comenzó con una breve investigación teórica sobre algoritmos de búsqueda y ordenamiento, utilizando como fuentes documentación oficial de Python y sitios como YouTube para ver casos en práctica. Se seleccionaron los métodos de burbuja, ordenamiento rápido (como métodos de ordenamiento) búsqueda binaria por su relevancia y relación con los contenidos vistos en clase, e incorporamos la búsqueda lineal para evaluar velocidades de búsqueda, etc.

Para el desarrollo, se creó una lista de videojuegos y se trabajó en archivos separados: uno para simular una base de datos, otro para las funciones y un tercero para el menú interactivo. El código fue probado por etapas, trabajando sobre distintas ramas y verificando que cada algoritmo funcionara correctamente antes de integrarlo antes de mergearlo.

Se utilizó Visual Studio Code como IDE, y Git para el control de versiones y GitHub para el repositorio. El proyecto fue realizado en equipo, dividiendo tareas como programación, testeo y documentación.

---

5. Resultados Obtenidos

Detalla qué se logró con el caso práctico, qué aspectos funcionaron correctamente y qué dificultades se presentaron.

El caso práctico permitió aplicar correctamente los algoritmos de búsqueda y ordenamiento en una lista de videojuegos. Se implementaron funciones para ordenar por título, año y puntaje, y se realizaron búsquedas tanto lineales como binarias. Los algoritmos funcionaron como se esperaba.

Durante las pruebas se corrigieron algunos errores, como problemas de scope, no ordenar antes de usar búsqueda binaria o comparar textos sin pasar a minúsculas. También se notaron diferencias de rendimiento entre burbuja y quicksort en listas más grandes.

---

6. Conclusiones

Reflexión final del grupo de trabajo. Aquí se recomienda incluir:

A lo largo del trabajo, el grupo pudo aplicar los conceptos de búsqueda y ordenamiento en un proyecto concreto. Se aprendió a implementar y comparar distintos algoritmos, entendiendo sus ventajas y limitaciones según el tipo de datos y el contexto.

Este tema resulta muy útil en programación, ya que ordenar y buscar información es algo que aparece en casi cualquier proyecto real, desde juegos hasta sistemas de gestión o aplicaciones web.

Como posibles mejoras, se podría implementar una interfaz gráfica o agregar filtros combinados (por género, año y puntaje). Las principales dificultades fueron al integrar las funciones con el menú y asegurar que la búsqueda binaria funcionara correctamente, lo cual se resolvió con ordenamiento previo y pruebas controladas.

---

7. Bibliografía

Listado de fuentes consultadas, utilizando normas básicas APA u otro formato consistente.

Python Software Foundation. (s.f.). *Sorting HOW TO*. Disponible en: https://docs.python.org/3/howto/sorting.html

---

8. Anexos

Material adicional que no va en el cuerpo principal del trabajo pero que aporta valor.

Pueden ser:

- Capturas del programa funcionando.
- Enlace al video explicativo.
- Código completo como archivo externo o adjunto.
- Cuadros comparativos.
- Documentos auxiliares (como diagramas de flujo).

Recomendaciones para la presentación

- Formato del archivo: **.docx** o **.pdf**
- Tipografía: Arial o Calibri, tamaño 11 o 12
- Interlineado: 1,5
- Márgenes estándar (2.5 cm)
- Portada opcional con el logo de la institución (si se requiere)
- Entrega digital mediante plataforma institucional o por correo