Merge Sort

Arturo Daza, Sebastian Samaniego, Giovanny Gonzalez

I. Introducción

Este taller se centra en el algoritmo de ordenamiento Merge Sort, conocido por su eficacia en grandes conjuntos de datos. El objetivo es implementar este algoritmo para ordenar más de 500 documentos en un array. Con la ayuda de herramientas modernas de desarrollo y colaboración en parejas, los participantes cargarán y organizarán estos documentos, aprendiendo tanto la aplicación práctica del Merge Sort como la importancia de la documentación estructurada en informes formales, preparándolos para enfrentar desafíos del mundo real en la organización de datos.

II. Resultados

Implementación del Algoritmo

Utilice una herramienta de desarrollo como GitHub Copilot, Easy Code o Phind para implementar el algoritmo de ordenamiento Merge Sort en el lenguaje de programación de su elección.

Carga de Documentos

Cree un programa que cargue al menos 500 documentos en un array. Estos documentos pueden ser nombres de archivos, títulos de documentos u otra información relevante. Puede generar datos aleatorios o utilizar datos reales si están disponibles.

```
documentos = [
"La programación en Python es clave para el trabajo con datos",
    "Los programadores en Java tienen un alto interés en pasar a Python",
    "La optimización de algoritmos es fundamental en el desarrollo de software",
    "Las bases de datos relacionales son esenciales para muchas aplicaciones",
    "El paradigma de programación funcional gana popularidad",
    "La seguridad informática es un tema crucial en el desarrollo de aplicaciones web",
    "Los lenguajes de programación modernos ofrecen abstracciones poderosas",
    "La inteligencia artificial está transformando diversas industrias",
    "El aprendizaje automático es una rama clave de la ciencia de datos",
    "Las interfaces de usuario intuitivas mejoran la experiencia del usuario",
    "La calidad del código es esencial para mantener un proyecto exitoso",
    "La agilidad en el desarrollo de software permite adaptarse a cambios rápidamente",
    "Las pruebas automatizadas son cruciales para garantizar la estabilidad del software",
    "La modularización del código facilita la colaboración en equipos de programadores",
    "El control de versiones es necesario para rastrear cambios en el código",
    "La documentación clara es fundamental para que otros entiendan el código",
    "La programación orientada a objetos promueve la reutilización de código",
    "La resolución de problemas es una habilidad esencial en la programación",
    "La optimización prematura puede llevar a código complicado y difícil de mantener",
    "El diseño de interfaces de usuario atractivas mejora la usabilidad de las aplicaciones",
    "El código limpio es esencial para facilitar el mantenimiento",
    "Los patrones de diseño son soluciones probadas para problemas comunes",
    "Las pruebas unitarias garantizan el correcto funcionamiento de las partes del código",
    "El desarrollo ágil prioriza la entrega continua de valor al cliente",
    "Los comentarios en el código deben ser claros y útiles",
    "La recursividad es una técnica poderosa en la programación",
    "Las bibliotecas de código abierto aceleran el desarrollo de software",
    "La virtualización permite una mejor utilización de los recursos de hardware",
    "La seguridad en la programación web es fundamental para prevenir ataques",
```

Aplicación de Merge Sort

Aplique el algoritmo Merge Sort para ordenar los documentos almacenados en el array. La ordenación se realizará en orden alfabético o según un criterio relevante.

```
def merge_sort(arr):
    if len(arr) <= 1:
        return arr

    mid = len(arr) // 2
    left_half = arr[:mid]
    right_half = arr[mid:]

    left_half = merge_sort(left_half)
    right_half = merge_sort(right_half)

    return merge(left_half, right_half)

def merge(left, right):
    result = []
    i = j = 0

    while i < len(left) and j < len(right):
        if left[i] < right[j]:
            result.append(left[i])
            i += 1
        else:
            result.append(right[j])
            j += 1

    result.extend(left[i:])
    result.extend(right[j:])
    return result

# Aplicar Merge Sort a la lista de documentos
documentos_ordenados = merge_sort(documentos)

# Imprimir los documentos ordenados
for documento in documentos_ordenados:
    print(documento)
```

```
El análisis de datos masivos (big data) abre oportunidades para obtener insights
El análisis predictivo utiliza datos históricos para predecir tendencias
El aprendizaje automático es una rama clave de la ciencia de datos
El aprendizaje profundo es una rama avanzada del machine learning
El control de versiones es necesario para rastrear cambios en el código
El código autodocumentado es claro y fácil de entender para otros programadores
El código limpio es esencial para facilitar el mantenimiento
El desarrollo full-stack abarca tanto el frontend como el backend
El desarrollo ágil prioriza la entrega continua de valor al cliente
El diseño de interfaces de usuario atractivas mejora la usabilidad de las aplicaciones
El diseño de interfaces de usuario es crucial para la experiencia del usuario
El diseño responsivo garantiza una experiencia consistente en diferentes dispositivos
El enfoque DevOps une el desarrollo y las operaciones para una entrega eficiente
El enfoque centrado en el usuario mejora la usabilidad de las aplicaciones
El machine learning permite a las máquinas aprender de los datos
El monitoreo de aplicaciones permite identificar y resolver problemas en tiempo real
El paradigma de programación funcional gana popularidad
El rendimiento de las aplicaciones es esencial para brindar una buena experiencia
La agilidad cultural es clave para adoptar prácticas ágiles de manera efectiva
La agilidad en el desarrollo de software permite adaptarse a cambios rápidamente
La agilidad en el desarrollo permite adaptarse a cambios del mercado
La analítica de datos ayuda a tomar decisiones basadas en información
La arquitectura de microservicios permite escalar componentes individualmente
```

III. Conclusiones

En base a los análisis de los ejercicios de la parte A y B, podemos concluir lo siguiente:

1. Merge Sort se ha demostrado como un algoritmo eficiente para ordenar grandes volúmenes de datos. En este taller, hemos aplicado este algoritmo a una lista de más de 50 documentos, destacando su capacidad para manejar conjuntos extensos de manera ordenada y rápida.

2. La estructura de Divide y Vencerás implementada por

Merge Sort es fundamental para su eficacia. Dividir la lista en mitades, ordenar cada mitad por separado y luego combinarlas ordenadamente ilustra la estrategia clave detrás del éxito del Merge Sort.

3. Merge Sort no solo es eficiente, sino también versátil. Puede aplicarse a diversos tipos de datos y ofrece resultados consistentemente ordenados, lo que lo convierte en una herramienta confiable en el arsenal del programador para tareas de ordenamiento.