|  |
| --- |
| **PROYECTO 1 IPC** |
| **202300645– Carlos Heraldo Quiná Corona** |

**Resumen**

**Novedad/Vigencia:** Si bien el procesamiento de datos matriciales es un tema recurrente en diversas disciplinas, la aplicación específica de este algoritmo, combinada con la utilización de XML como formato de entrada y salida, aporta una solución novedosa para la gestión de datos estructurados en contextos donde la flexibilidad y extensibilidad de XML son valoradas.

**Posturas y Impactos:** El trabajo se centra en la eficiencia computacional y la legibilidad del código. El impacto se observa principalmente a nivel técnico, al ofrecer una herramienta versátil para la manipulación de datos matriciales, y a nivel de productividad, al automatizar tareas repetitivas de consolidación de datos.

**Conclusiones:** El algoritmo desarrollado demuestra la capacidad de Python para abordar problemas de procesamiento de datos de manera eficiente y flexible. La utilización de XML como formato de entrada y salida facilita la integración de este programa en entornos heterogéneos, donde la interoperabilidad de datos es crucial.

**Palabras clave**

Python, XML, procesamiento de matrices, consolidación de datos, algoritmo

***Abstract***

***Novelty/Currency****: While matrix data processing is a recurrent theme in various disciplines, the specific application of this algorithm, combined with the use of XML as input and output format, provides a novel solution for managing structured data in contexts where the flexibility and extensibility of XML are valued.*

***Positions and Impacts:*** *The work focuses on computational efficiency and code readability. The impact is observed mainly at the technical level, by offering a versatile tool for manipulating matrix data, and at the productivity level, by automating repetitive data consolidation tasks.*

***Conclusions:*** *The developed algorithm demonstrates Python's ability to address data processing problems efficiently and flexibly. The use of XML as input and output format facilitates the integration of this program into heterogeneous environments, where data interoperability is crucial.*

***Keywords***

*Python, XML, matrix processing, data consolidation, algorithm.*

**Introducción**

**¿Cómo simplificar grandes conjuntos de datos matriciales almacenados en XML?** Este ensayo presenta una solución innovadora: un algoritmo en Python capaz de sumar filas similares en matrices XML. La gestión eficiente de datos estructurados es fundamental en diversos campos, desde la ciencia de datos hasta las finanzas. El formato XML, por su flexibilidad, es ideal para almacenar matrices, pero su procesamiento puede ser complejo.

Nuestro algoritmo automatiza la tarea de sumar filas con características similares, generando una matriz más compacta y fácil de analizar. Exploraremos las bases teóricas de las operaciones matriciales y las razones por las que Python es una herramienta poderosa para este tipo de tareas. Los resultados obtenidos demostrarán la eficiencia y versatilidad de nuestra solución.

En resumen, este trabajo tiene como objetivo:

* Ofrecer una herramienta práctica para un problema común en el manejo de datos.
* Mostrar las capacidades de Python para el procesamiento de datos estructurados.
* Destacar las ventajas del formato XML en el contexto de matrices.
* Contribuir al campo de la ciencia de datos y la ingeniería de software.

A lo largo del ensayo, responderemos preguntas clave como: ¿Cómo definimos la similitud entre filas? ¿Cuál es la eficiencia computacional de nuestro algoritmo? ¿Qué beneficios ofrece en comparación con otros métodos?

En conclusión, este ensayo proporcionará una solución detallada, respaldada por resultados experimentales, y una discusión sobre las posibles aplicaciones y mejoras futuras de nuestro algoritmo.

**Desarrollo del tema**

**Representación de la Matriz en XML**

* **Estructura XML:** Se define una estructura XML simple para representar la matriz, donde cada fila es un elemento y cada celda es un atributo de ese elemento.
* **Lectura del XML:** Se utiliza una librería de parsing de XML (como ElementTree) para leer el archivo y construir una representación en memoria de la matriz.

**Estructura de Datos Personalizada: Listas Enlazadas**

* **Nodo:** Se define una clase Nodo para representar una celda de la matriz, conteniendo el valor de la celda y un puntero al siguiente nodo en la fila.
* **Fila:** Una fila se representa como una lista enlazada de nodos.
* **Matriz:** La matriz completa se representa como una lista de filas.

**Algoritmo de Procesamiento**

1. **Lectura de la Matriz:** Se lee el archivo XML y se construye la representación en memoria de la matriz utilizando las listas enlazadas.
2. **Identificación de Filas Similares:**
   * Se define un criterio de similitud entre filas (por ejemplo, basado en una cierta tolerancia en los valores de las celdas).
   * Se itera sobre todas las filas, comparando cada fila con las siguientes y marcando las filas similares.
3. **Suma de Filas Similares:**
   * Para cada conjunto de filas similares, se crea una nueva fila sumando los valores correspondientes de cada celda.
   * La fila original y las filas sumadas se eliminan de la matriz.
4. **Generación de la Matriz Resultante:**
   * La nueva matriz, que contiene las filas originales y las filas sumadas, se convierte nuevamente a formato XML y se guarda en un nuevo archivo.

**Escritura en XML**

* **Generación del XML:** Se utiliza la librería de parsing de XML para crear los elementos XML correspondientes a la matriz resultante.
* **Escritura del Archivo:** Se escribe el XML generado en un nuevo archivo.

**Ventajas del Enfoque Utilizado**

* **Flexibilidad:** La estructura de datos personalizada permite adaptar el programa a diferentes tipos de matrices y criterios de similitud.
* **Aprendizaje:** La implementación manual de las listas enlazadas proporciona un entendimiento profundo de las estructuras de datos y los algoritmos de manipulación.
* **Optimización:** Al evitar el uso de librerías de alto nivel, se pueden realizar optimizaciones específicas para el problema en cuestión.

**Desafíos y Consideraciones**

* **Complejidad:** La implementación manual de las listas enlazadas puede resultar más compleja que el uso de librerías especializadas.
* **Eficiencia:** Para matrices grandes, la eficiencia del algoritmo puede verse afectada por la búsqueda de filas similares.
* **Mantenimiento:** El código puede ser más difícil de mantener debido a la mayor cantidad de código personalizado.

Anexos:

Imagen 1

Imagen que contiene Escala de tiempo

Descripción generada automáticamente

Diagrama de actividades de la opción 2

Imagen 2

Diagrama

Descripción generada automáticamente

Diagrama de actividades de la opción 2

Imagen 3

Imagen que contiene Diagrama

Descripción generada automáticamente

Diagrama de clases del programa

**Conclusiones**

El programa desarrollado demuestra la viabilidad de procesar matrices XML utilizando una estructura de datos personalizada basada en listas enlazadas. Aunque este enfoque requiere un mayor esfuerzo de implementación, ofrece una mayor flexibilidad y un mejor entendimiento de los procesos involucrados. Futuras investigaciones podrían explorar la optimización del algoritmo para matrices de gran tamaño y la implementación de criterios de similitud más sofisticados.

**Referencias bibliográficas**

1. Algar Díaz, M. J. & Fernández de Sevilla Vellón, M. (2019). Introducción práctica a la programación con Python. [Universidad de Alcalá1](https://biblioteca.unapec.edu.do/media/smjjs13i/bibliograf%C3%ADa-programaci%C3%B3n-python.pdf). Este libro ofrece una introducción práctica al lenguaje Python y sus fundamentos.
2. Arboledas Brihuega, D. (2017). Criptografía sin secretos con Python. [RA-MA1](https://biblioteca.unapec.edu.do/media/smjjs13i/bibliograf%C3%ADa-programaci%C3%B3n-python.pdf). Si te interesa la criptografía y cómo aplicarla en Python, este libro es una excelente opción.
3. Arévalo Ovalle, D. (2021). Métodos numéricos con Python. [Politécnico Grancolombiano1](https://biblioteca.unapec.edu.do/media/smjjs13i/bibliograf%C3%ADa-programaci%C3%B3n-python.pdf). Aquí encontrarás información sobre cómo utilizar Python para resolver problemas numéricos y matemáticos.
4. Arnáez Braschi, E. (2015). Enfoque práctico de la teoría de robots: Con aplicaciones en Matlab. [Universidad Peruana de Ciencias Aplicadas1](https://biblioteca.unapec.edu.do/media/smjjs13i/bibliograf%C3%ADa-programaci%C3%B3n-python.pdf). Aunque no es exclusivamente sobre Python, este libro aborda conceptos relevantes para la programación en general.
5. Arroyo Guardeño, D., Gayoso Martínez, V. & Hernández Encinas, L. (2020). Ciberseguridad. [CSIC Consejo Superior de Investigaciones Científicas1](https://biblioteca.unapec.edu.do/media/smjjs13i/bibliograf%C3%ADa-programaci%C3%B3n-python.pdf). La ciberseguridad es un tema crucial en la actualidad, y este libro explora cómo Python se utiliza en este campo.