Compresor de Canciones

Gutierrez Vazquez Axel
Hernandez Macias Axxel Gael
Quintero Arreola Laura Vanessa

Introducción

¿Qué es un algoritmo voraz?

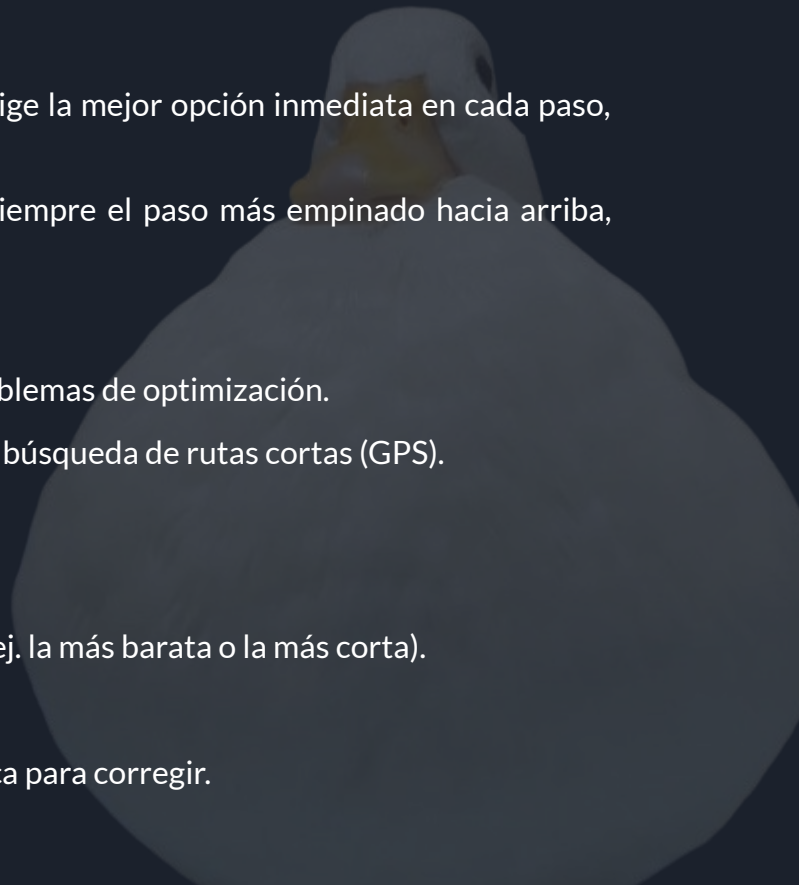
- Es una estrategia de resolución de problemas que elige la mejor opción inmediata en cada paso, sin pensar en las consecuencias futuras.
- Analogía: Es como escalar una montaña eligiendo siempre el paso más empinado hacia arriba, esperando llegar a la cima más alta.

¿Para qué sirve?

- Para encontrar soluciones rápidas y eficientes en problemas de optimización.
- Se usa en compresión de datos (ahorrar espacio) y en búsqueda de rutas cortas (GPS).

¿Cómo funciona?

- Analiza el conjunto de opciones disponibles.
- Selecciona la opción más atractiva en ese momento (ej. la más barata o la más corta).
- Añade esa opción a la solución final.
- Repite el proceso hasta terminar, sin retroceder nunca para corregir.



Objetivos

- **Implementación Algorítmica:** Demostrar el funcionamiento conjunto de **Prim** (reordenamiento) y **Huffman** (compresión) aplicados a archivos reales.
- **Interfaz Gráfica Funcional:** Desarrollar una GUI amigable en Python (Tkinter) que oculte la complejidad matemática al usuario.
- **Gestión de Audio:** Integrar un reproductor capaz de procesar y reproducir archivos desde un formato binario propio.
- **Eficiencia:** Lograr la compresión y descompresión de archivos multimedia utilizando programación concurrente (hilos) para mantener la fluidez de la aplicación.

Compresor de canciones

Decidimos abordar el proyecto final desarrollando un compresor de canciones, para analizar cómo funcionan algoritmos como huffman y prim a la hora de resolver el problema.

El código que desarrollamos es una aplicación que permite al usuario ingresar una canción y poderla comprimir, así mismo se puede hacer una comparación de cuanto reduce su tamaña. Además dentro de la misma aplicación se implementa un botón para descomprimir, reproducir el resultado y ver los códigos.

Procesamiento de Audio

Manipulación de Datos (Backend)

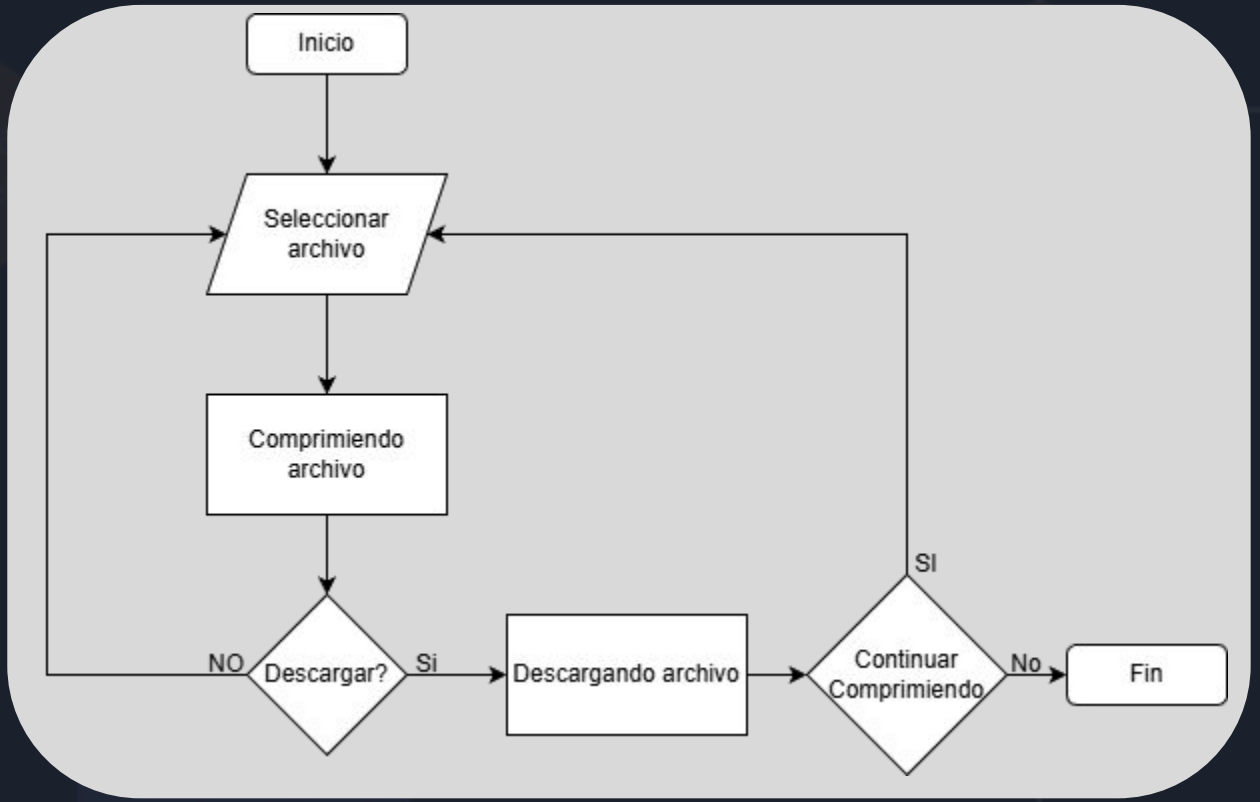
- El audio digital es una secuencia de bytes.
- En este proyecto, el audio se trata como "datos crudos" para ser reordenados (Algoritmo de Prim) y comprimidos, sin alterar su calidad sonora al descomprimir.

Reproducción e Interfaz (Frontend)

- Decodificación en Tiempo Real: Se convierten los bits comprimidos a un archivo .mp3 temporal (_temp_playback.mp3).
- Gestión de Recursos: Uso de hilos (threads) para no congelar la interfaz mientras se procesa el audio.
- Control de Volumen: Ajuste matemático de la ganancia de salida (0.0 a 1.0) mediante la librería pygame.mixer.

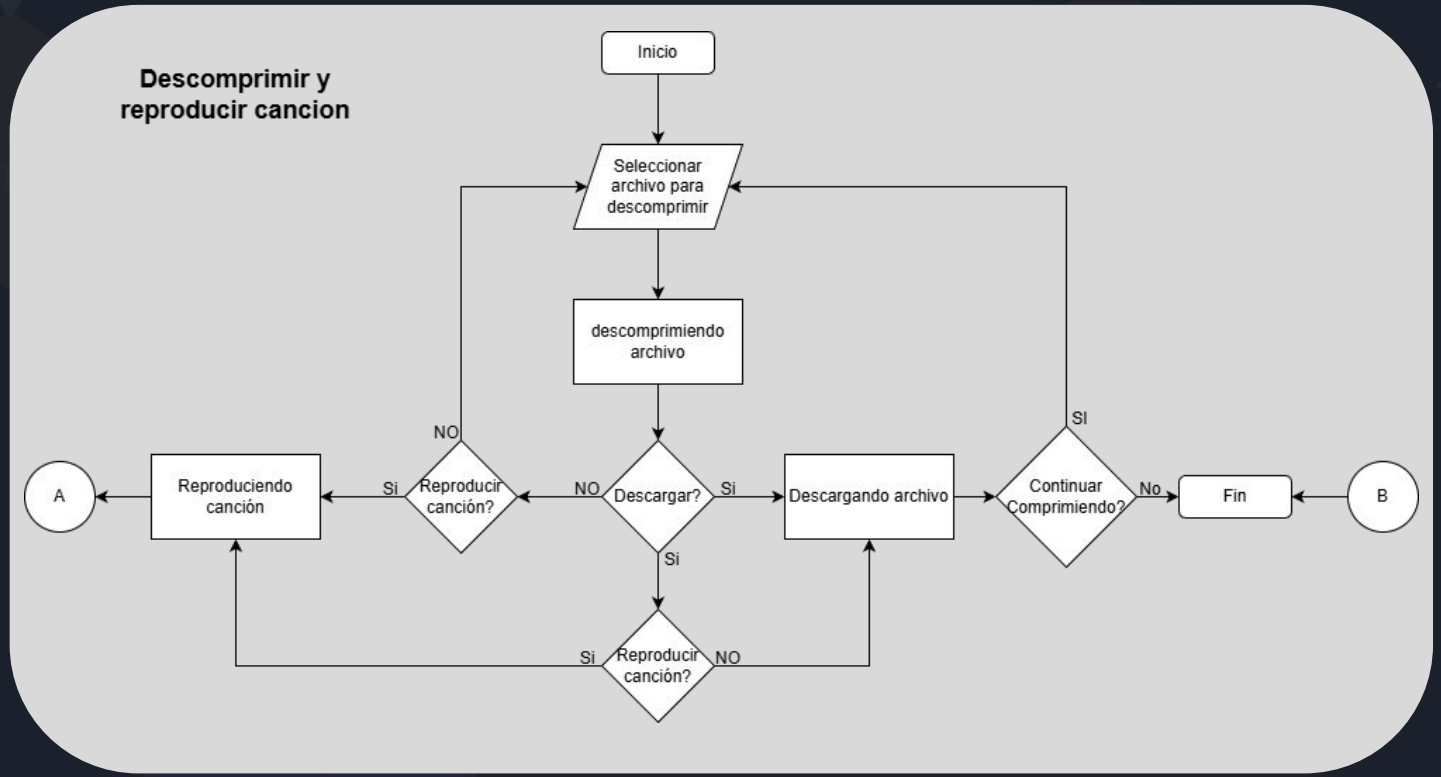
Diagramas de flujo

El siguiente diagrama explica el funcionamiento al seleccionar un archivo y comprimirlo.



Diagramas de flujo

El diagrama a continuación muestra el proceso de seleccionar un archivo para descomprimir, para posteriormente reproducir la canción o descargar el archivo descomprimido.



Complejidad Computacional

Las operaciones que dominan el costo total son:
- Recorrer todos los bytes (contar frecuencias)
- Reasignarlos con el orden del árbol MST
- Codificar cada byte en Huffman
- Decodificar cada bit en la descompresión

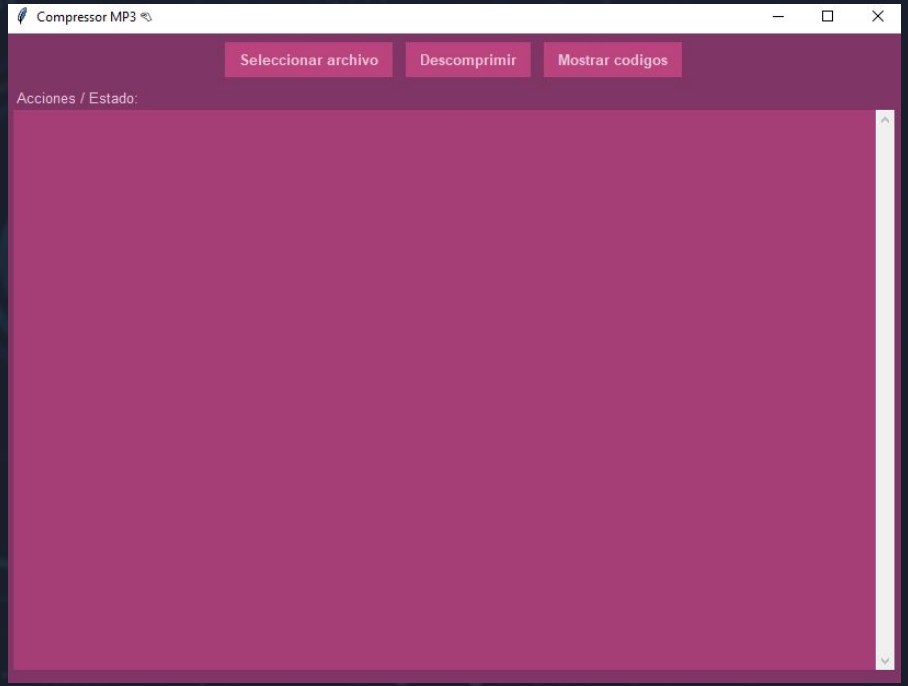
Todas estas operaciones requieren leer cada byte del archivo, por lo que son:

$$O(n)$$

La complejidad temporal como espacial también son O(n)

Resultados

Ahorita que terminemos de presentar puede usar la GUI profe :D



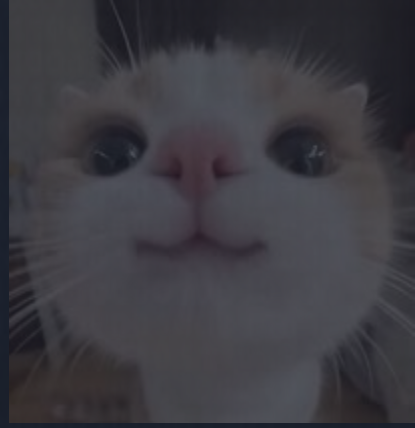
Conclusión

El proyecto demostró que la combinación de Prim y Huffman permite crear un descompresor eficiente y lineal al procesar archivos de audio. Prim reorganiza los símbolos de forma óptima y Huffman reduce la redundancia al generar códigos compactos.
La reasignación de símbolos basada en el MST generado por Prim redujo la variabilidad del conjunto de Bytes, mientras que Huffman aprovechó dichas frecuencias para producir una codificación más compacta.

Juntos ofrecen un método simple y eficaz para manejar grandes volúmenes de datos, destacando la utilidad de los algoritmos clásicos en aplicaciones modernas de compresion y descompresion.

Referencias

- Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to algorithms (3.ª ed.). MIT Press.
- Huber, D. M., & Runstein, R. E. (2018). Modern recording techniques (9.ª ed.). Routledge.
- KeepCoding. (12 de Agosto de 2024). Algoritmos voraces: ¿Qué son y cómo funcionan? KeepCoding Bootcamps. https://keepcoding.io/blog/que-son-los-algoritmos-voraces/
- Martínez Sandoval, L. I. (22 de agosto de 2022). Árboles de peso mínimo: Algoritmos de Prim y Kruskal. MADI NekoMath. https://madi.nekomath.com/P5/ArbolPesoMin.html



Gracias por su atencion