Reporte - Actividad 08, Parte 1: Huffman, FrontEnd

Rivera Reos Fernando de Jesus

Materia: Análisis de Algoritmos

Maestro: Jorge Ernesto Lopez Arce Delgado

Sección: D01

Introducción

Técnica voraz (Greedy technique):

La técnica voraz es un enfoque de resolución de problemas en algoritmos y programación que se caracteriza por tomar decisiones locales óptimas en cada paso con la esperanza de llegar a una solución global óptima. En cada paso del algoritmo, se elige la mejor opción disponible en ese momento sin considerar el panorama completo. Esta técnica es útil en problemas donde se pueden tomar decisiones paso a paso y no se necesita retroceder en las decisiones tomadas previamente. La técnica voraz generalmente es eficiente en términos de tiempo de ejecución, pero no siempre garantiza la mejor solución global.

Algoritmo de Huffman:

El algoritmo de Huffman es un método de compresión sin pérdida que se basa en la técnica voraz. Fue desarrollado por David A. Huffman en 1952. Este algoritmo se utiliza comúnmente para la compresión de datos, como en la compresión de archivos de texto, imágenes y audio.

El algoritmo de Huffman funciona asignando códigos de longitud variable a los símbolos en función de su frecuencia de aparición en los datos que se están comprimiendo. Los símbolos que ocurren con mayor frecuencia reciben códigos más cortos, mientras que aquellos que ocurren con menos frecuencia reciben códigos más largos. Esto permite una representación más eficiente de los datos, ya que los símbolos más comunes se representan con menos bits.

El proceso de compresión con el algoritmo de Huffman generalmente sigue estos pasos:

1. Calcular la frecuencia de aparición de cada símbolo en los datos.
2. Construir un árbol binario de Huffman utilizando una cola de prioridad (generalmente implementada como una estructura de datos de montículo).
3. Asignar códigos binarios a cada símbolo basados en el árbol de Huffman generado.
4. Comprimir los datos utilizando los códigos asignados.
5. Almacenar el árbol de Huffman junto con los datos comprimidos para que se pueda usar durante la descompresión.
6. Durante la descompresión, el árbol de Huffman se reconstruye utilizando la misma información almacenada junto con los datos comprimidos, y luego se utiliza para decodificar los datos binarios en la secuencia original de símbolos.

El algoritmo de Huffman es eficiente y suele proporcionar una buena compresión para datos con distribuciones de frecuencia no uniformes, lo que lo hace ampliamente utilizado en diversas aplicaciones de compresión de datos.

Objetivos

Para esta parte de la actividad, se busca implementar y documentar la parte de frontend para el algoritmo de compresión Huffman.

Debe ser una ventana con tres botones: Examinar, Comprimir, Descomprimir, el botón examinar deberá de permitir buscar algún archivo de su equipo abrir el archivo y calcular la frecuencia de los caracteres, posteriormente crear la LISTA de frecuencia de caracteres e imprimirla.

Desarrollo

El código es una aplicación básica de interfaz gráfica de usuario (GUI) escrita en Python utilizando la biblioteca Tkinter. La aplicación tiene la funcionalidad de examinar archivos, calcular la frecuencia de caracteres en el archivo seleccionado y mostrar esta frecuencia en una lista en la interfaz gráfica.

1. Librerías Utilizadas:

- **tkinter**: Para crear la interfaz gráfica.
- **filedialog**: Para proporcionar una interfaz de exploración de archivos.
- **Collections:** Para manejar la lista de frecuencia de caracteres de manera más eficiente.

2. Funciones:

openFile(): Esta función se activa cuando se hace clic en el botón "Examinar". Abre un cuadro de diálogo para que el usuario seleccione un archivo. Luego, lee el contenido del archivo, calcula la frecuencia de los caracteres en el contenido y muestra esta frecuencia en la ventana de la aplicación.

3. Interfaz de Usuario:

- Se crea una ventana principal (root) con el título "Compresión Huffman".
- Se crea un marco para los botones (buttonFrame) que contiene tres botones: "Examinar", "Comprimir" y "Descomprimir".
- Se crea otro marco (frequencyFrame) para mostrar la frecuencia de caracteres del archivo seleccionado.
- Se utiliza una etiqueta para indicar el propósito de la lista que muestra la frecuencia.

- Se utiliza una Listbox para mostrar la frecuencia de caracteres.
- La lista de frecuencia se actualiza cada vez que se selecciona un nuevo archivo.

4. Ejecución:

El programa se ejecuta mediante root.mainloop(), lo que inicia el bucle principal de la interfaz gráfica.

Conclusiones

El código implementa de manera fluida la forma de examinar y recuperar un archivo, leerlo y calcular la frecuencia de caracteres que se busca.

Bibliografía

- Bratley, P., & Brassard, G. (1997). Fundamentos de algoritmia.
- Levitin, A. (2008). Introduction to design and analysis of algorithms, 2/E. Pearson Education India.
- Sedgewick, R. (2013). An introduction to the analysis of algorithms. Pearson Education India.
- Documentacion Oficial de Python, Modulo ‘Collections’ https://docs.python.org/3/library/collections.html#counter-objects
- Documentacion Oficial de Python, Modulo ‘tkinter’ https://docs.python.org/es/3/library/tkinter.html