La Transformada Rápida de Fourier

En este documento se explicará cómo fue la elaboración del punto 5 para la practica de la transformada rápida de Fourier, en donde, se nos pide realizar lo siguiente:

5. **Este punto da sobre la calificación final de la Unidad de Aprendizaje dos puntos**. Lo que hay que hacer es grabar su voz en un archivo wav, mp3 o algún otro (no se admite texto con formato) medio minuto al menos, graficarla y obtener su Transformada de Fourier Rápida por ambos algoritmos DIF y DIT. Darle al usuario de calcularlo iterativamente o recursivamente. Reportar conclusiones.

Antes que nada, es importante mencionar que para la elaboración de este punto solo fue necesario de crear dos clases extra, las cuales llevan el nombre de VozGraficafft y ObtenerDatosAudio, las demás clases del proyecto ya fueron explicadas debidamente en el documento anterior correspondiente a esta práctica.

Así pues, expliquemos la clase ObtenerDatosAudio. Esta clase nos permite obtener las amplitudes de onda de cualquier audio que tenga el formato .wav (es importante mencionar que este programa fue diseñado exclusivamente para el uso de este formato de audio), el funcionamiento de esta clase es sencillo tenemos el método ObtenerDatosdelAudio el cual tiene la dirección del archivo .wav que se usara, después se llama al método obtenerWAVAmplitudes asignado le valor de retorno de ese método a AmplitudAudio para después procesar esas amplitudes con el método procesarAmplitudes.

El método obtenerWAVAmplitudes obtiene lo que su nombre indica que son las amplitudes del archivo .wav elegido, el proceso del como se genera esta explicado en el código mediante comentarios, es importante mencionar que hacemos uso de la Modulación por impulsos codificados o PCM por sus siglas en inglés. El PCM es un procedimiento de modulación utilizado para transformar una señal analógica en una secuencia de bits (señal digital) y que es la forma estándar de audio digital en computadoras, discos compactos, telefonía digital y otras aplicaciones similares. En un flujo PCM la amplitud de una señal analógica es muestreada regularmente en intervalos uniformes, y cada muestra es cuantizada al valor más cercano dentro de un rango de pasos digitales, en resumen, nos permite pasar de un audio a bits para poder computar la visualización del audio normal como el audio después de aplicar la FFT ya sea de forma discreta, en su forma DIT o en su forma DIF, en estos últimos dos casos ya sea de forma recursiva o iterativa, sin embargo, aquí no acaba la cosa tenemos que entender el ultimo método de esta clase el cual es procesarAmplitudes.

ProcesarAmplitudes, la cual recibe como parámetro el arreglo generado por obtenerWAVAmplitudes, nos permite ajustar las amplitudes tal que podamos observarlas en la ventana de una forma tal que permita que no se amontonen entre

ellas, su funcionamiento es sencillo, utiliza el ancho de la ventana y el tamaño del arreglo para poder obtener cuantas muestras por pulgada se mostraran en la ventana y serializa la información en un archivo de texto con el nombre “DatosOnda.txt” en forma de string separado mediante comas. Es importante mencionar que este .txt se crea para que el programa sea más eficiente y no tenga que calcular los datos de la onda de audio cada vez que se quiera usar un método en específico, al final del día cuando se cierra el programa este archivo es borrado.

Ahora expliquemos la última clase la cual es VozGraficafft, esta clase crea toda la GUI usando JavaFX, se usa JavaFX debido a la comida de uso y a que las GUI que se generan quedan estéticamente aceptable, en resumen es una clase que cada vez que se ejecuta llama a la clase ObtenerDatosAudio obteniendo lo que ya se mencionó, como vemos hay 13 métodos los cuales no todos son necesarios de explicar como lo son el main, start, Cerrar el cual le da la facultad al programa de finalizar el proceso de forma definitiva y borrar el archivo .txt generado e init, para este último solo hay que mencionar que es donde se crea las GUI y llama a la clase ObtenerDatosAudio, los métodos con el prefijo GraficarAudioSalida son los métodos que calculan la FFT correspondiente a su nombre, esto se hace pasando de un arreglo de números float a un arreglo de números complejos que tiene el siguiente formato Complejo(número float, 0), se calcula la correspondiente FFT y se muestra el resultado de forma gráfica, el método GraficarAudioEntrada solo grafica el audio de entrada, esto para poder ver la gráfica antes de hacer uso de cualquier FFT, después de lo explicado nos damos cuenta que nos quedan 3 métodos por explicar el más sencillo es createGrid lo único que hace este método es crear los botones de la ventana que se ve al ejecutar el programa, el método ObtenerArregloFloat obtiene el arreglo de float del archivo .txt generado su funcionamiento es simple, obtiene el string almacenado en el archivo y lo transforma a un arreglo de números float. Y por último el método llamado CompletarArreglo mismo que llama mucho la atención por su nombre e importancia para el correcto funcionamiento de cualquier programa que pretenda usar DITFFT o DIFFFT ya sea mediante su forma recursiva o iterativa, se puede dar el caso de querer usar el método de DFT esto provoca que el uso del método CompletarArreglo no sea necesario, sin embargo, lo utilizo para obtener los mismos valores y graficas que resultan, ahora veamos el porque es útil, pero antes recordemos la teoría de esta practica tanto para DIT como DIF, en los cuales se hacen uso de las operaciones mariposa las cuales para su funcionamiento es necesario usar una potencia de dos (2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, etc.), en caso de no usar cualquier potencia de dos el algoritmo fallara rotundamente ya que se está incumpliendo una condición necesaria, después de recordar la operación mariposa y su uso hablemos plenamente del método CompletarArreglo, este método antes que nada calcula la potencia de dos más cercana y grande al tamaño del arreglo que contiene los datos de la onda de audio, es decir, si el tamaño del arreglo fuera 50 el método obtendría 64 ya que es la más cercana y grande al número 50, por lo cual este método nos dice el tamaño que debe tener nuestro arreglo para que pueda funcionar correctamente los algoritmos DIT y DIF, por

ejemplo, para pasar de un arreglo de tamaño 50 o de cualquier número que no es potencia de 2 hacemos lo siguiente: tenemos el tamaño del arreglo con el que funcionara el programa y el tamaño del arreglo con los datos, estos dos valores se restan de la forma $DatosFaltantes = 2^n - tamaño\ del\ arreglo\ datos\ de\ la\ onda$, en donde 2^n es el valor de la potencia de 2 encontrada, después obtenemos la mitad de $DatosFaltantes$ para de esta manera agregar al inicio y al final del arreglo de los datos de la onda original para de esta forma llegar al tamaño requerido en el ejemplo anterior tendríamos que $DatosFaltantes = 64 - 50 = 14$ la mitad de 14 es 7 entonces agregaríamos 7 ceros al inicio del arreglo, después los 50 datos y al final otros 7 ceros, dándonos un total de 64 elementos en un arreglo, es decir obtenemos lo deseado, una forma matemática de definir lo anterior seria: $ArregloFinal = n + Tamaño\ arreglo\ original + n$, en donde n tiene el mismo valor tal que el tamaño del $ArregloFinal$ resulta ser una potencia de 2, obviamente en caso de que $Tamaño\ arreglo\ original$ resulte ser potencia de 2 no es necesario el procedimiento.

Conclusiones

Hay mucho que rescatar sobre la práctica, una parte se rescató en el documento anterior de esta práctica y es que después de leer varios documentos en los cuales se habla sobre la FFT obtuve mucho conocimiento, en especial ya que en ESCOM aún no he visto la transformada de Fourier. Es un tema sumamente interesante ya que tiene una implementación que hoy en día es una novedad y es el reconocimiento de voz, para lograr el reconocimiento de voz se aplica la FFT ya que se obtienen patrones similares, lo único que cambia es la intensidad con la que se habla pero en esencia la FFT que se obtiene de un “no” o de un “si” es similar en todos los casos, obviamente cambia respecto al idioma en el que se habla pero en esencia se obtiene de esa manera. Es muy interesante esta aplicación y porque no en esta larga cuarentena intentar realizar un reconocimiento básico de voz que detecte si se dijo un “si” o un “no” pero eso ya será un proyecto personal.

Nota: Mientras de menor duración sea el audio la visualización de las FFT será mejor, esto debido a la cantidad de los datos que se procesan, se intento optimizar lo mejor posible para el audio que se grabó, sin embargo, se anexa un audio en la carpeta audio con el nombre de “audio” el cual es de utilidad ya que al hacer un audio tan pequeño la FFT se ve de manera perfecta. El audio por defecto del programa es la grabación que realice, el cual tiene el nombre de “Grabacion”. Para poder modificar el nombre del audio a ocupar se hace en la clase VozGraficafft en la variable NombreArchivo solo se requiere poner el nombre y dejar la extensión del archivo la cual es .wav

Se anexan imágenes de como se ve el audio del archivo audio.wav:

Imagen 1: Grafica del audio de entrada

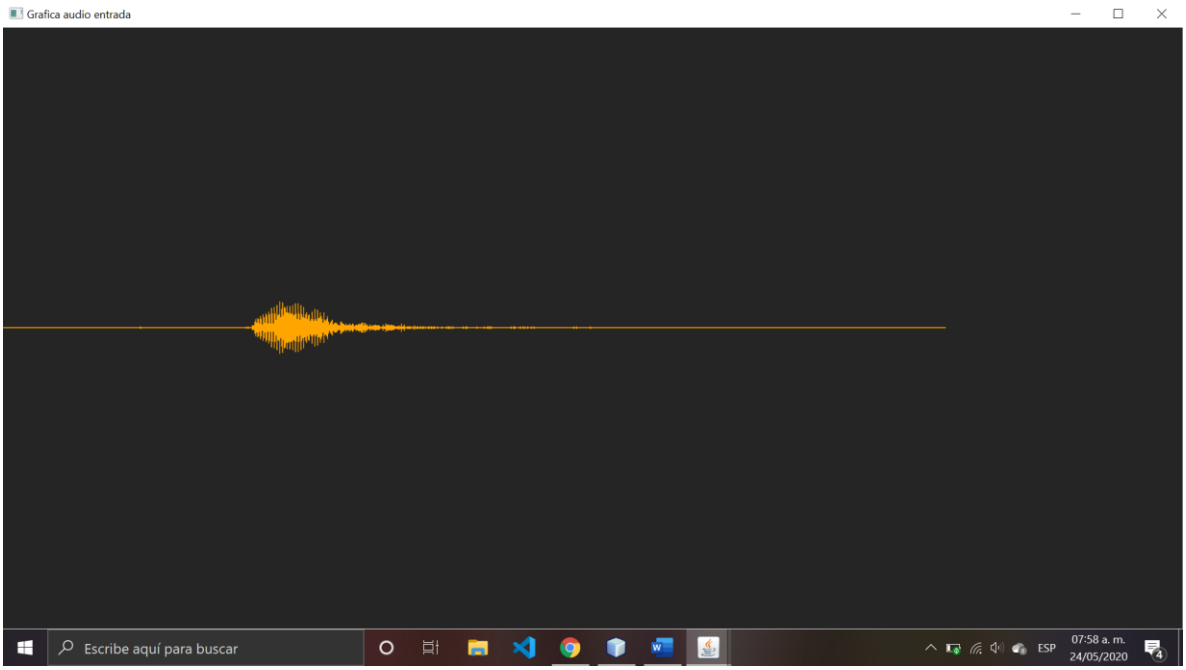


Imagen 2: Grafica del audio usando DFT

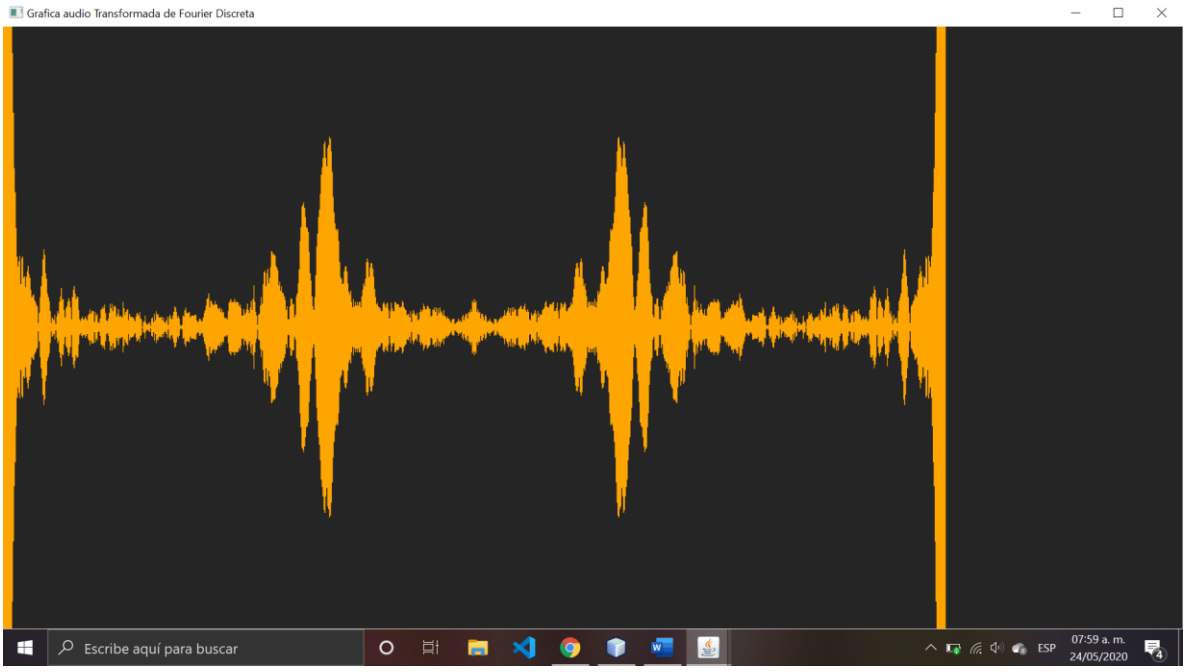


Imagen 3: Grafica del audio usando DIF en forma recursiva

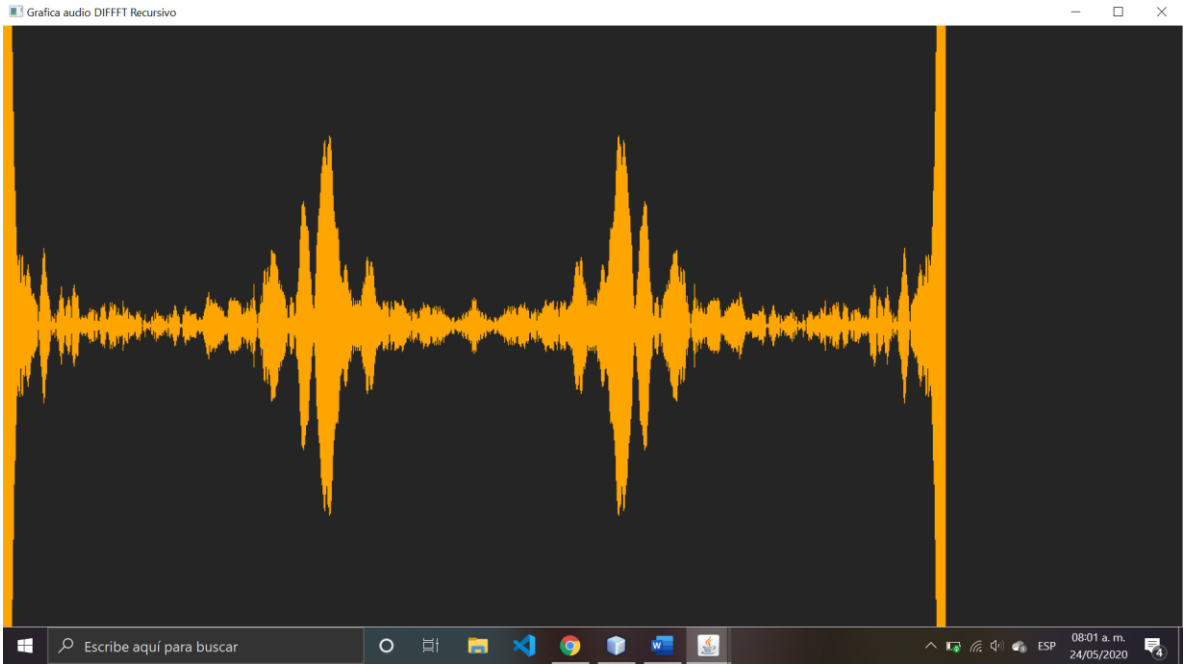
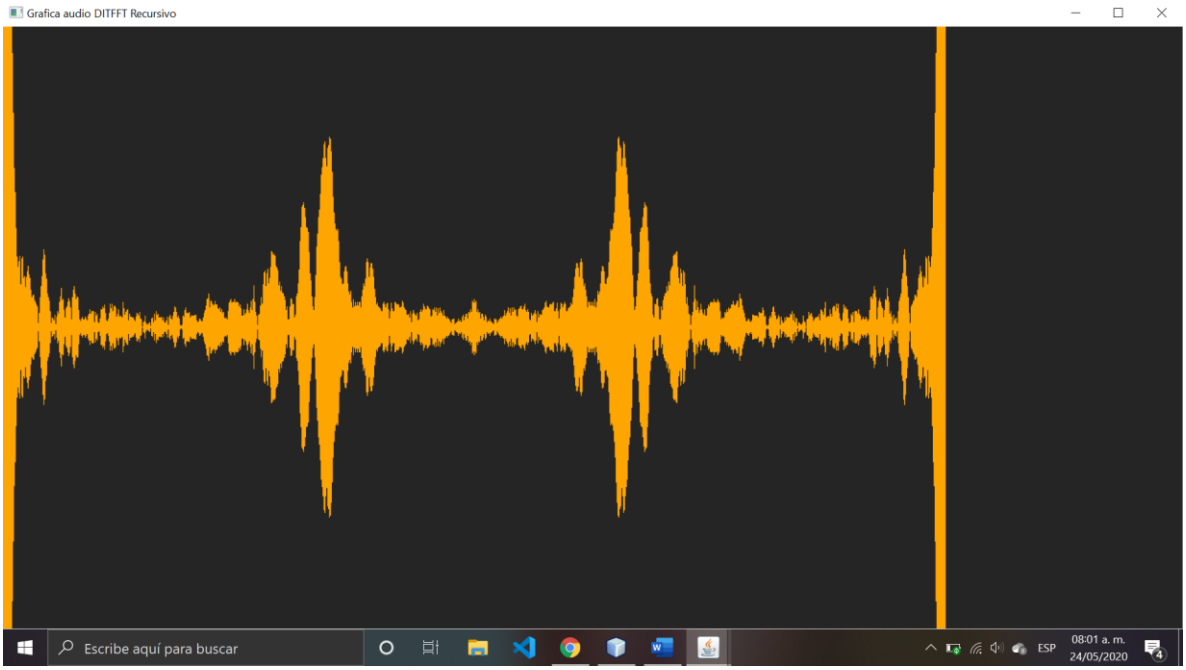


Imagen 4: Grafica del audio usando DIT en forma recursiva



Nota: Se eligieron esos colores debido al contraste que generan

Nota 2: Se excluyeron las imágenes DIT y DIF iterativos debido a que los recursivos dan casi los mismos valores, sin embargo, están disponibles en la GUI.

Sanchez Mendez Edmundo Josué
Análisis de Algoritmos