La Transformada Rápida de Fourier

Punto 1.-

Al momento de copiar y ejecutar el código que se nos proporcionó me di cuenta de que en mi opinión DFT no es tan exacto como el DIFFFT ya que este último nos proporciona un valor más exacto ya que nos da el resultado usando notación científica esto en el caso en donde los resultados de DFT son un 0 exacto y el DIFFFT nos da valores muy próximos a 0 esto es algo que considero habla de mayor precisión en los cálculos, sin embargo me di cuenta de una gran diferencia en el pulso cuadrado, al ver el código me percate que el pulso cuadrado de la clase CalculadoraDFT era diferente al de la clase CalculadoraDIFFFT, al momento de cambiar el pulso y ejecutar de nuevo “coincidieron” los valores para esta onda, cabe aclarar que uso las comillas para expresar que los valores no son exactamente los mismos pero si son muy próximos. Es muy interesante ver como al cambiar la forma de atacar al problema que se quiere solucionar cambian los valores, ya que una forma puede ser más precisa que otra, aunque solo cambien los últimos dígitos del resultado.

Para el desarrollo del punto 3 se tomaron las funciones de entrada de DIFFFT, al inicio los resultados no se parecían a los resultados de la clase DIFFFT, investigando un poco descubrí que para el uso del algoritmo iterativo de DIFFFT y DITFFT iterativos hay varias formas de ocuparlos: con la entrada ordenada, con la salida desordenada, por lo que se debería ordenar invirtiendo bits o una versión donde la entrada y salida ya están en orden (es más complejo computacionalmente). El caso de DIFFFT ocupa la entrada en orden y la salida en desorden teniendo que ocupar la función invertirBits para ordenar, ahora en el caso de DITFFT fue complicado ver cómo eran las entradas y salidas, al inicio pensé que era con la entrada desordenada y salida era la que ya estaba ordenada pero me equivoque completamente, para este algoritmo es necesario que la entrada sea ordenada y la salida será desordenada por lo que se usa la función invertirBits, el gran cambio reside en los precálculos de w__{N}^{r} conjugados pero no en cómo se calcula, si no en la parte de obtener el valor. Ahí es necesario invertir ese bit específico para obtener el valor W correcto, para ello se creó la clase invertirBit. Algo importante a mencionar es que, con base en lo indicado en el documento de esta práctica, lo cual es que la complejidad de la Transformada Discreta de Fourier es de $\Theta(N^2)$, pero usando la Transformada Rápida de Fourier se logra reducir a $\Theta(Nlog(N))$ en donde N es el tamaño de entrada del algoritmo. En los documentos y paginas leídas se considera al algoritmo DITFFT más eficiente que el algoritmo DIFFFT aun a pesar de tener la misma complejidad.

Sanchez Mendez Edmundo Josué
Análisis de Algoritmos

Punto 2.-

Para la parte de generar las gráficas de las funciones de entrada y salida se optó por usar JavaFX debido a que tiene los componentes necesarios para la realización de una gráfica.

Es importante mencionar que a diferencia del punto 1 se respetó los valores de las funciones de entrada que se tenían, por lo que la gráfica de la función de pulso cuadro es diferente. La razón es muy simple, debido a que hay diferente cantidad de unos en las funciones difiere la salida ya que esa cantidad de números representa a la cantidad de veces - 1 que la transformada de Fourier de la función pulso cuadrado podrá formar arcos en la gráfica, es decir, las veces que forma un arco el pulso es cantidad de unos - 1

Se añade un ejemplo en la Imagen 1a de la entrada y un ejemplo de la salida en la Imagen 1b usando DFT.

Nota: El código para generar las gráficas se encuentra en la clase FFT.

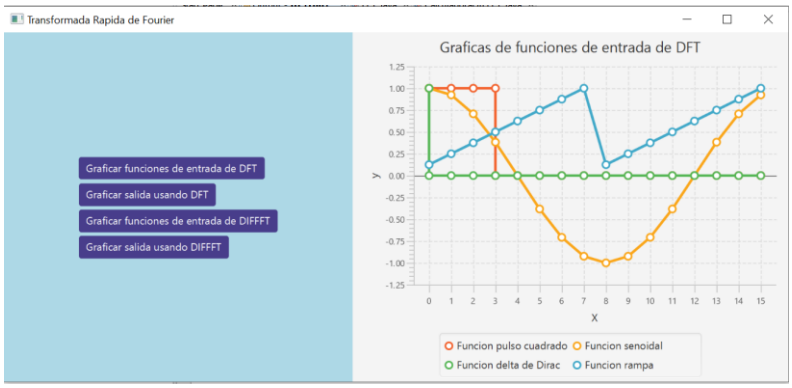

Imagen 1a. Graficas de funciones de entrada DFT

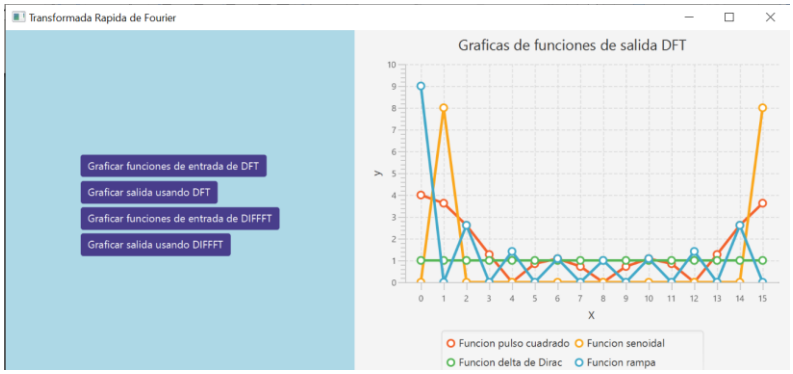

Imagen 1b. Graficas de funciones de salida DFT

Punto 3.-

La conclusión de este punto se anexo al **Punto 1**, en este punto se mostrará una captura de pantalla de los resultados que arroja la clase TransformadaDITFFT (ver Imagen 2) y también unas capturas de pantalla de la GUI incluyendo las gráficas de entrada y salida haciendo uso DITFFT (ver Imagen 3a y 3b).

```
run:

Pulso cuadrado
 (1.0, i 0.0) (1.0, i 0.0) (1.0, i 0.0) (1.0, i 0.0) (1.0, i 0.0) (1.0, i 0.0) (1.0, i 0.0) (0.0, i 0.0) (0.0, i 0.0) (0.0, i 0.0) (0.0, i 0.0) (0.0, i 0.0)
FFT Pulso cuadrado
 (8.0, i 0.0) (1.0, i -5.027339492125847) (0.0, i 0.0) (1.0, i -1.4966057626654892) (0.0, i 0.0) (1.0, i -0.6681786379192989) (0.0, i 0.0) (1.0, i -0.19891236737965778) (0.0, i 0.0)
FFT Magnitud pulso
 8.0 5.125830895483012 0.0 1.7999524462728316 0.0 1.2026897738700906 0.0 1.0195911582083184 0.0 1.0195911582083184 0.0 1.2026897738700906 0.0 1.7999524462728316 0.0 5.125830895483012

Funcion senoidal
 (1.0, i 0.0) (0.9238795325112867, i 0.0) (0.7071067811865476, i 0.0) (0.38268343236508984, i 0.0) (6.123233995736766E-17, i 0.0) (-0.3826834323650897, i 0.0) (-0.7071067811865475, i
FFT senoidal
 (-8.996207971523 45E-16, i 0.0) (8.0, i -1.133107779529596E-15) (4.396005698054655E-17, i 4.710277376051326E-16) (-8.879442133834921E-16, i 1.3576491606600733E-16) (9.957992501029601
FFT Magnitud Senoidal
 8.996207971523 45E-16 8.0 4.730746412598571E-16 8.982633458600204E-16 8.937432890299474E-16 8.924931928192026E-16 5.121091585599618E-16 1.9915985002059193E-16 2.1060222747281167E-16

Impulso delta
 (1.0, i 0.0) (0.0, i 0.0) (0.0, i 0.0) (0.0, i 0.0) (0.0, i 0.0) (0.0, i 0.0) (0.0, i 0.0) (0.0, i 0.0) (0.0, i 0.0) (0.0, i 0.0) (0.0, i 0.0) (0.0, i 0.0)
FFT Impulso delta
 (1.0, i 0.0) (1.0, i 0.0) (1.0, i 0.0) (1.0, i 0.0) (1.0, i 0.0) (1.0, i 0.0) (1.0, i 0.0) (1.0, i 0.0) (1.0, i 0.0) (1.0, i 0.0) (1.0, i 0.0) (1.0, i 0.0)
FFT Magnitud delta
 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0

Rampa
 (0.125, i 0.0) (0.25, i 0.0) (0.375, i 0.0) (0.5, i 0.0) (0.625, i 0.0) (0.75, i 0.0) (0.875, i 0.0) (1.0, i 0.0) (0.125, i 0.0) (0.25, i 0.0) (0.375, i 0.0) (0.5, i 0.0) (0.625, i
FFT Rampa
 (9.0, i 0.0) (0.0, i 0.0) (-1.0, i 2.4142135623730951) (0.0, i 0.0) (-1.0, i 1.0) (0.0, i 0.0) (-1.0, i 0.41421356237309515) (0.0, i 0.0) (-1.0, i 0.0) (0.0, i 0.0) (-1.0, i -0.41421
FFT Magnitud Rampa
 9.0 0.0 2.613125929752753 0.0 1.4142135623730951 0.0 1.082392200292394 0.0 1.0 0.0 1.082392200292394 0.0 1.4142135623730951 0.0 2.613125929752753 0.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

Imagen 2. Resultados de la clase TransformadaDITFFT

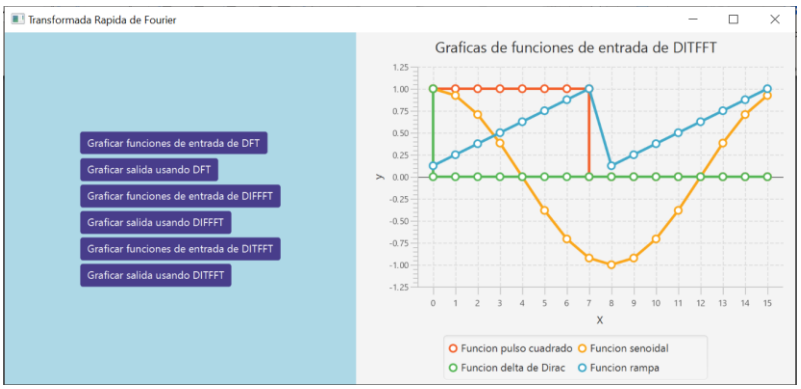

Imagen 3a. Graficas de funciones de entrada DITFFT

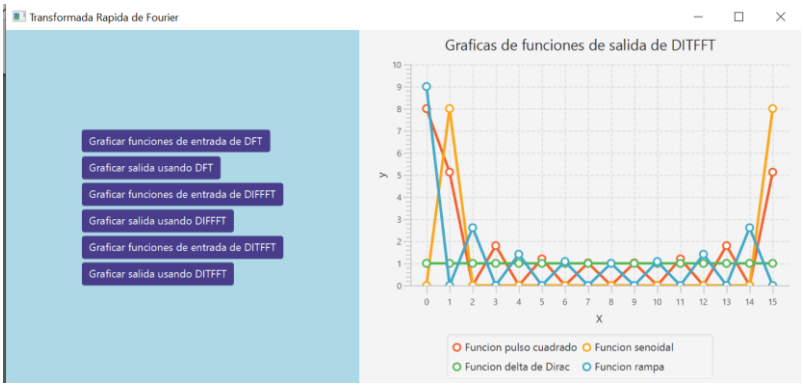

Imagen 3b. Graficas de funciones de salida DITFFT

Sanchez Mendez Edmundo Josué
Análisis de Algoritmos

Punto 4.-

Para poder realizar esta parte de la practica fue necesario recordar la práctica de dividir y vencer para hacer lo mismo con los algoritmos iterativos de DIT Y DIF FFT fue complicado definir los algoritmos recursivos ya que al inicio se trataron de hacer usando la clase invertirBits en la entrada y salida respectivamente, sin embargo, esto no era necesario ya que el algoritmo ordena la entrada y salida automáticamente debido al paradigma en uso (dividir y vencer), para el caso de los cálculos de w_{N}^{r} conjugados se hacen durante el proceso usando la clase calcularraicesdivyvencer en ambos casos, si se compara el código de la clase anteriormente mencionada con la clase calcularRaices es más pequeño ya que genera solo las raíces que se usaran en la correspondiente recursión y no más, provocando así que sea más eficiente que los algoritmos originales.

Si comparamos el algoritmo iterativo y recursivo correspondiente nos damos cuenta de que la idea es parecida dividir la suma en dos partes, obtener el valor de w_{N}^{r} conjugados correspondiente con el cálculo, hacer las operaciones correspondientes con los números complejos y obtener un resultado.

Nota: El código recursivo para DITFFT y DIFFFT se encuentran en las mismas clases que los iterativos y el código para probar el algoritmo recursivo usando dividir y vencer se encuentra en CalculadoraDITFFTDivideyvenceras y CalculadoraDIFFFTDivideyvenceras respectivamente