# Homework 2

February 5, 2020

## 1 Introduction

For this homework, you will implement a FFT function that applies Fast Fourier Transform (FFT) to a list of numbers. Your solution should include the solution of homework 1 and the following functions

1. `split :: [a] -> ([a], [a])` that splits a list into two lists that contains the even and odd indexed elements of the input list.
2. `fft :: [Double] -> [(Double, Double)]` that computes the FFT of the input list. This function should be recursive and should utilize the `split` function.

## 2 FFT Algorithm

Given a list of doubles $X = [x_0, x_1, x_2, \dots x_{N-1}]$, if $N \leq 16$, you should call `dft(X)`. Otherwise, split $X$ into an even list $X_1 = [x_0, x_2, \dots, x_{n-2}]$ and an odd list $X_2 = [x_1, x_3, \dots, x_{n-1}]$ using the `split` method. Let $E = \texttt{fft}(X_1)$ and $O = \texttt{fft}(X_2)$. Then, we define $Y = \texttt{fft}(X)$ by

$$\begin{aligned} Y_k &= E_k + e^{-\frac{2\pi i}{N}k} O_k \quad \text{where } 0 \leq k < \tfrac{N}{2} \\ Y_{k+\frac{N}{2}} &= E_k - e^{-\frac{2\pi i}{N}k} O_k \quad \text{where } 0 \leq k < \tfrac{N}{2}. \end{aligned}$$

Let 2-tuple $(x, y)$ represent complex number $x + iy$.

If $e^{-\frac{2\pi i}{N}k} = (\cos(-\frac{2\pi}{N}k), \sin(-\frac{2\pi}{N}k)) = (u, v)$, $E_k = (a, b)$ and $O_k = (c, d)$, then

$$\begin{aligned} Y_k &= (a + uc - vd, b + ud + vc) \\ Y_{k+\frac{N}{2}} &= (a - uc + vd, b - ud - vc) \end{aligned}$$

## 3 Testing

You can test the implementation using the following `main`

```
import Data.Time

-- your implementation goes here

main = do
         let n = 2^8
         let s1 = map (\x -> sin(20*pi*x) + sin(40*pi*x)/2) $ range 0 1 n
         -- print(rd 3 s1)

         start <- getCurrentTime
         let dft1 = map (\x -> x/n) $ absolute $ dft s1
         print(rd 2 dft1)
         end <- getCurrentTime
         print (diffUTCTime end start)

         start2 <- getCurrentTime
         let fft1 = map (\x -> x/n) $ absolute $ fft s1
         print(rd 2 fft1)
         end2 <- getCurrentTime
         print (diffUTCTime end2 start2)
```

You should expect FFT to be about 10 times faster than DFT. This performance gain should increase if you increase the size of `n`. There are peak values of 0.5, 0.25 at index 10 and 20. The results of DFT and FFT should be the same.

## 4 Submission

Please write your solution in a file – `hwk2.hs` and submit it to the dropbox.