Chapter 2  PROGRAMMING EXERCISE

Comparison of DFT and FFT in C++

The discrete Fourier transform (DFT) is a function that takes audio data in the time domain and transforms it to the frequency domain. Given a vector of $N$ audio samples (called $f$), the DFT returns $N/2$ valid frequency components (stored in output vector $F$). The frequency components are complex numbers whose magnitudes are distributed between 0 and the Nyquist frequency. The equation for the DFT is this:

$$F_n = \frac{1}{N} \left( \sum_{k=0}^{N-1} f_k \cos \frac{2\pi nk}{N} - if_k \sin \frac{2\pi nk}{N} \right)$$

Equation 1 Formulation of the discrete Fourier transform (DFT)

This equation can be implemented “literally” in a programming language like C++, yielding an algorithm with a computational complexity which is $O(N^2)$. Consider how the transform is applied. Because frequency components change over time (unless you’re dealing with a single note or chord or some other unchanging-frequency sound), in order to evaluate the frequency components of a segment of sound, you have to apply the Fourier transform repeatedly over small windows of sound samples. For sound sampled at 44.1 kHz, a window size of 1024 or 2048 samples (0.023 and 0.046 seconds, respectively) is reasonable. For just three minutes of sound, with a window size of 1024 samples, and moving the window over a full 1024 samples to do the next transform, you’d have to apply the transform over 129 times. (An exercise in Chapter 7 has you do a more complete computational complexity analysis.

The fast Fourier transform (FFT) is an alternative implementation of the Fourier transform which has a complexity of $O(N \log N)$. The algorithm benefits from not repeating the same calculations. One simple version of this algorithm is called the radix-2 DIT. It operates by first computing the DFTs of the even indexed inputs and then of the odd-indexed inputs and then combining these results, recursively to produce the DFT of the entire sequence. So that the input can be divided in half at each recursive level, $N$ must be a power of 2. The equation representing this operation is this:

$$F_n = \sum_{k=0}^{N/2-1} f_{2k}e^{-2\pi(2k)n/N} + \sum_{k=0}^{N/2-1} f_{2k+1}e^{-2\pi(2k+1)n/N}$$

Equation 2 Formulation of the radix-2 DIT algorithm for the FFT
Your assignment is to implement both of these algorithms and compare their run-times. You can easily find additional sources concerning the FFT that turn Equation 1 into an algorithm. There are numerous variations and optimizations of the FFT that you may want to explore.