EXERCISE FOR CSE202 – WEEK 3

The variant of FFT seen in class is called "decimation-in-frequency". Another variant, called "decimation-in-time" leads to the following algorithm, with the same input/output.

(1) If $n = 1$, return a_0.
(2) Split A into $A^{(e)} = (a_0, a_2, \dots, a_{n-2})$ and $A^{(o)} = (a_1, a_3, \dots, a_{n-1})$.
(3) Compute recursively $y^{(e)} := \mathrm{DFT}_{\omega^2}(A^{(e)})$ and $y^{(o)} := \mathrm{DFT}_{\omega^2}(A^{(o)})$.
(4) For $j = 0$ to $n/2-1$: compute $y_j := y_j^{(e)} + \omega^j y_j^{(o)}$ and $y_{j+n/2} = y_j^{(e)} - \omega^j y_j^{(o)}$.
(5) Return $(y_0, \dots, y_{n-1})$.

Question 1. *Prove the correctness of that algorithm (i.e., it terminates and computes the DFT of its input A.)*

Solution. Termination is clear since the power of 2 is reduced by 1 at each recursive call.

The proof that the algorithm computes the DFT is by induction.

For $n = 1$ the correctness is clear. Otherwise, we just have to check that the formulas for the y_j's compute the DFT for n assuming the algorithm to be correct for $n/2$. From

$$\begin{aligned} y_j^{(e)} &= a_0 + a_2\omega^{2j} + a_4\omega^{4j} + \cdots + a_{n-2}\omega^{(n-2)j}, \\ y_j^{(o)} &= a_1 + a_3\omega^{2j} + a_4\omega^{4j} + \cdots + a_{n-1}\omega^{(n-2)j}, \end{aligned}$$

it follows that for $j \in \{0, \dots, n/2-1\}$,

$$\begin{aligned} y_j &= y_j^{(e)} + \omega^j y_j^{(o)} \\ &= a_0 + a_1\omega^j + a_2\omega^{2j} + a_3\omega^{3j} + \cdots = A(\omega^j), \\ y_{j+n/2} &= y_j^{(e)} - \omega^j y_j^{(o)} \\ &= a_0 + a_1\omega^{n/2+j} + a_2\omega^{2(n/2+j)} + a_3\omega^{3(n/2+j)} + \cdots = A(\omega^{j+n/2}), \end{aligned}$$

where, in the last line, we use the fact that $\omega^{n/2} = -1$, which comes from ω being a primitive nth root of 1. □

Question 2. *Analyse its asymptotic complexity.*

Solution. Let $C(n)$ be the number of arithmetic operations performed by the algorithm for n a power of 2. The algorithm performs two recursive calls in size $n/2$ (ie, $2C(n/2)$ operations), and $3n/2$ operations in Step (4) ($n/2$ multiplications $\omega^j y_j^{(o)}$ followed by n additions or subtractions). This leads to

$$C(n) \le 2C(n/2) + 3n/2,$$

which is exactly the same inequality as the one satisfied by the complexity of the variant of FFT seen in class. Thus the same consequence follows,

$$C(n) \le \frac{3}{2} n \log_2 n + O(n).$$

□