Algorithms and Data Structures 2020/21
Week 5 tutorial sheet

1. In class we mostly worked with DFT for the case where n is a power of 2, and the polynomial being evaluated has degree $n-1$. However, we also showed that we can apply the DFT and Inverse DFT when n is not a power of 2, by taking n' to be the closest power of 2 satisfying $n \leq n'$, and adding some leading coefficients of value 0.

   Make this idea formal: First show how to compute n'. Also show that our DFT or Inverse DFT still takes $\Theta(n \lg(n))$ time in terms of the original value n.

2. This exercise asks you to do a few complex number calculations. Evaluate each of these. (See also rules for multiplication and division in the FFT notes.)

   (a) $2i(3-i)$.

   (b) $2i(i+1)^2 + 4(i+1)^3$.

   (c) $3i/(1+i)$.

3. Compute $\text{DFT}_4\langle 0, 1, 2, 3\rangle$. *(do this directly, rather than by FFT, if you prefer).*

   This is Exercise 30.2-2, p. 838 of [CLRS].

4. Use the FFT to efficiently multiply the two polynomials $p(x) = x-4$ and $q(x) = x^2-1$. Use the following steps:

   (a) First work out what will be the degree of the product polynomial pq. Take $\deg(pq)+1$ as our n (and if necessary round up to the nearest power of 2).

   (b) For this value of n (which we made sure was a power of 2), use trigonometry to write down each of the nth roots-of-unity (so we have them to work with).

   (c) Calculate the DFT for $p(x)$ for nth roots of unity.

   (d) Calculate the DFT for $q(x)$ for nth roots of unity.

   (e) Do pointwise multiplication of the two DFTs to get the DFT of $pq(x)$ for nth roots of unity.

   (f) Calculate the Inverse DFT of the DFT for $pq(x)$, to obtain the polynomial pq. It's a good idea to do this via DFT (and then swapping), like we saw in class.

   (g) Check your answer by straight multiplication.