# ADS Tutorial 3 – Solutions*

Instructor: Heng Guo Tutors: Georgios Kalantzis and Dhairya Mangla

October 15, 2026

## Problem 1

Evaluate each of the following formulas involving complex numbers. Recall that $i^2 = -1$.

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

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

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

### Solution

(a) $2i(3 - i) = 6i - 2(i^2) = 6i - 2(-1) = 2 + 6i$

(b) $2i(i + 1)^2 + 4(i + 1)^3$ There are two approaches we can take to solving this problem. The first is to factor the polynomials and substitute $i^2 = -1$ as appropriate. But we could also convert our numbers into polar coordinates, since exponentiation is simpler (and generalises better) in polar form.

The factor and simplify approach:

$$\begin{aligned}
2i(i+1)^2 + 4(i+1)^3 &= 2i(i+1)(i+1) + 4(i+1)(i+1)(i+1) \\
&= 2i(i^2 + 2i + 1) + 4(i+1)(i^2 + 2i + 1) \\
&= 2i^3 + 4i^2 + 2i + 4(i^3 + 2i^2 + i + i^2 + 2i + 1) \\
&= 6i^3 + 16i^2 + 14i + 4 \\
&= 6i(-1) + 16(-1) + 14i + 4 \\
&= -12 + 8i
\end{aligned}$$

Cartesian → polar → Cartesian approach:

$$\begin{aligned}
2i(i+1)^2 + 4(i+1)^3 &= 2e^{i\pi/2}\left(\sqrt{2}e^{i\pi/4}\right)^2 + 4\left(\sqrt{2}e^{i\pi/4}\right)^3 \\
&= 2e^{i\pi/2}\left(2e^{i\pi/2}\right) + 4\left(2\sqrt{2}e^{i3\pi/4}\right) \\
&= 4e^{i\pi} + 8\sqrt{2}e^{i3\pi/4} \\
&= (-4) + 8\sqrt{2}\left(\frac{1}{\sqrt{2}}(-1+i)\right) \\
&= -12 + 8i
\end{aligned}$$

*The solutions contain additional explanations that are not necessary, if you were to answer such a question in an exam.

(c) $3i/(1+i)$ To remove the imaginary number from the denominator of the fraction, we simply multiply the top and bottom by the *complex conjugate* of $1+i$:

$$\begin{aligned}\frac{3i}{1+i} &= \frac{3i}{1+i}\cdot\frac{1-i}{1-i}\\ &= \frac{3i-3i^2}{1-i^2}\\ &= \frac{3i-3(-1)}{1-(-1)}\\ &= \tfrac{3}{2}+\tfrac{3}{2}i\end{aligned}$$

# Problem 2

Recall the *Discrete Fourier Transform (DFT)*, i.e., a sequence of (complex) numbers obtained by evaluating a polynomial of degree $n-1$ on each of the $n$th roots of unity. To apply a Divide-and-Conquer approach, in the lectures we assumed that $n$ is a power of 2, i.e., that $n = 2^k$ for some non-negative integer $k$.

If we wanted to use the FFT on a polynomial of degree $n-1$ where $n$ is not a power of 2, we could use *padding*. We first find $n'$ which is the smallest power of 2 such that $n' > n$, and then we apply the FFT to the polynomial with $n'$ coefficients obtained by setting the coefficients of $x^n, \ldots, x^{n'-1}$ to 0. This evaluates the polynomial at the $n'$th roots of unity, not at the $n$th roots of unity, so it computes $\mathrm{DFT}_{n'}$ of the padded coefficients, not $\mathrm{DFT}_n$; this is enough for polynomial multiplication (Problem 3).

Provide an algorithm for the computation of $n'$ and an argument about its running time. Also, argue that the running time of the Divide-and-Conquer FFT algorithm on the padded polynomial is $O(n \log n)$.

## Solution

Recall that the Discrete Fourier Transform of a sequence of $n$ complex numbers $p_0$, $p_1, \cdots$, $p_{n-1}$ is defined to be the sequence of complex numbers

$$P(1),\ P(\omega_n),\ P(\omega_n^2), \cdots,\ P(\omega_n^{n-1})$$

obtained by evaluating the polynomial $P(x) = p_0 + p_1 x + p_2 x^2 + \cdots + p_{n-1}x^{n-1}$ on each of the $n$th roots of unity.

If $n$ is not a power of 2, then we can define $n' := 2^{\lceil \log_2 n \rceil}$ to be the smallest power of 2 greater than $n$. Note that this definition immediately implies that $n' < 2n$, since otherwise there must exist another power of 2 between $n$ and $n'$. We construct the polynomial $p'(x)$ as follows:

$$p'(x) := p(x) + p_n x^n + p_{n+1}x^{n+1} + \cdots + p_{n'-1}x^{n'-1},$$

where $p_n = p_{n+1} = \cdots = p_{n'-1} = 0$. This padding operation takes time $O(n')$, and $p'(x) = p(x)$ for any number $x$. (I.e., evaluating $p'(x)$ at any number $x$ gives you the same value as evaluating $p(x)$ at $x$.) From here, we can simply apply the divide-and-conquer approach outlined in lecture to $p'$. It returns $\mathrm{DFT}_{n'}(p_0, \ldots, p_{n'-1})$, the values of $p$ at the $n'$th roots of unity. This is not $\mathrm{DFT}_n(p_0, \ldots, p_{n-1})$, because the $n'$th roots of unity are different points from the $n$th roots of unity: for example, $1+x+x^2$ has $\mathrm{DFT}_3 = (3, 0, 0)$, while padding it to $n' = 4$ gives $(3, i, 1, -i)$. For polynomial multiplication this does not matter: we only need the values at enough points, and the inverse $\mathrm{DFT}_{n'}$ turns them back into coefficients.

In the original DFT, we evaluate a polynomial of degree $m-1$ at $m$ distinct points in time $O(m \log m)$. Here, we evaluate a polynomial with $n'$ coefficients at $n'$ points, which, for $n' < 2n$, should have runtime no more than $O(n' \log n') \leq O(2n \log(2n)) = O(2n \log n + 2n \log 2) = O(n \log n)$.

# Problem 3

**A.** Compute $\mathrm{DFT}_4(0,1,2,3)$ (i.e., evaluate the polynomial $x+2x^2+3x^3$ on each of the 4th roots of unity). You don't have to apply the FFT algorithm, you may do the calculation directly.

**B.** Use the FFT algorithm to multiply the polynomials $p(x) = x - 4$ and $q(x) = x^2 - 1$. To do this, use the following steps:

(a) First figure out what the degree of $p(x) \cdot q(x)$ is. Let $n = \deg(p(x) \cdot q(x)) + 1$. If this is not a power of two, work as in Problem 2 to pick an $n$ that is a power of 2 by padding. (Lecture 5 pads to the smallest power of 2 that is at least twice the number of coefficients of each factor; any power of 2 that is at least $\deg(p(x) \cdot q(x)) + 1$ works, since that is the number of coefficients of the product.)

(b) Write down each of the $n$th roots of unity.

(c) Evaluate both $p(x)$ and $q(x)$ at each of the $n$th roots of unity $x_j$, and then compute $p(x_j) \cdot q(x_j)$ for every $j$. To save time, you may do this directly as in Part A of this question, without using the FFT recurrence. We will use the FFT in the next step, for the polynomial interpolation.

(d) Recover the coefficients of the polynomial $p(x) \cdot q(x)$. To do that, use the inverse FFT as presented in the lectures: run the FFT on the values $p(x_j) \cdot q(x_j)$ with $\omega_n^{-1}$ in place of $\omega_n$, and divide by $n$.

## Solution

**A.** The 4th complex roots of unity are the solutions to the equation $z^4 = 1$. They are 1, $\omega$, $\omega^2$, $\omega^3$, where $\omega = e^{i2\pi/4} = e^{i\pi/2}$, and evaluate to $\{1, i, -1, -i\}$. Substituting each of these values into the polynomial $f(x) = x + 2x^2 + 3x^3$ gives the following:

$$
\begin{aligned}
f(\omega^0) &= f(1) = 1 + 2(1)^2 + 3(1)^3 = 6 \\
f(\omega^1) &= f(i) = i + 2(i)^2 + 3(i)^3 = i + 2(-1) + 3i(-1) = -2 - 2i \\
f(\omega^2) &= f(-1) = -1 + 2(-1)^2 + 3(-1)^3 = -1 + 2 - 3 = -2 \\
f(\omega^3) &= f(-i) = -i + 2(-i)^2 + 3(-i)^3 = -i + 2(-1) - 3i(-1) = -2 + 2i
\end{aligned}
$$

**B.** (a) The degree of the product of any two polynomials $pq$ is $\deg(pq) = \deg(p) + \deg(q)$. Since $p(x)$ and $q(x)$ have degrees 1 and 2, respectively, the degree of $pq$ is simply $1 + 2 = 3$. The number $n$ is defined to be one more than the degree of $pq$; thus, $n = 4$, which is indeed a power of 2.

(b) As in part A, $n = 4$. Thus, our roots of unity are the same as before:

$$\{\omega^0, \omega^1, \omega^2, \omega^3\} = \{(e^{i\pi/2})^0, (e^{i\pi/2})^1, (e^{i\pi/2})^2, (e^{i\pi/2})^3\} = \{1, i, -1, -i\}.$$

(c) Evaluating both $p(x) = x - 4$ and $q(x) = x^2 - 1$ at each 4th root of unity gives:

$$
\begin{array}{lll}
p(1) = (1) - 4 = -3 & q(1) = (1)^2 - 1 = 0 & pq(1) = -3 \cdot 0 = 0 \\
p(i) = (i) - 4 = -4 + i & q(i) = (i)^2 - 1 = -2 & pq(i) = (-4 + i)(-2) = 8 - 2i \\
p(-1) = (-1) - 4 = -5 & q(-1) = (-1)^2 - 1 = 0 & pq(-1) = -5 \cdot 0 = 0 \\
p(-i) = (-i) - 4 = -4 - i & q(-i) = (-i)^2 - 1 = -2 & pq(-i) = (-4 - i)(-2) = 8 + 2i
\end{array}
$$

(d) Write $C(x) := p(x)q(x)$ and $y_k := C(\omega_4^k)$, so that $(y_0, y_1, y_2, y_3) = (0,\ 8 - 2i,\ 0,\ 8 + 2i)$ by part (c). By the inverse DFT (Lecture 5), the coefficients of $C$ are

$$c_j = \frac{1}{4} \sum_{k=0}^{3} y_k \, \omega_4^{-jk} \qquad (j = 0, 1, 2, 3),$$

that is, we run the FFT on $(y_0, y_1, y_2, y_3)$ with $\omega_4^{-1} = -i$ in place of $\omega_4$, and divide by 4.

**With the FFT.** The even half $(y_0, y_2) = (0, 0)$ gives $(0, 0)$. The odd half $(y_1, y_3) = (8-2i,\ 8+2i)$ gives $(y_1 + y_3,\ y_1 - y_3) = (16, -4i)$ (for $n = 2$ the root is $\omega_2^{-1} = -1$). The butterflies, with $\omega_4^{-1} = -i$, give

$$\big(0 + 16,\ 0 + (-i)(-4i),\ 0 - 16,\ 0 - (-i)(-4i)\big) = (16, -4, -16, 4),$$

and dividing by 4 gives $(c_0, c_1, c_2, c_3) = (4, -1, -4, 1)$.

**Check, directly from the formula.** Only $y_1$ and $y_3$ are non-zero, and $(-i)^2 = -1$, $(-i)^3 = i$, $(-i)^4 = 1$:

$$\begin{aligned}
c_0 &= \tfrac{1}{4}\big((8-2i) + (8+2i)\big) = 4,\\
c_1 &= \tfrac{1}{4}\big((8-2i)(-i) + (8+2i)(-i)^3\big) = \tfrac{1}{4}\big((-2-8i) + (-2+8i)\big) = -1,\\
c_2 &= \tfrac{1}{4}\big((8-2i)(-i)^2 + (8+2i)(-i)^6\big) = \tfrac{1}{4}\big(-(8-2i) - (8+2i)\big) = -4,\\
c_3 &= \tfrac{1}{4}\big((8-2i)(-i)^3 + (8+2i)(-i)^9\big) = \tfrac{1}{4}\big((2+8i) + (2-8i)\big) = 1.
\end{aligned}$$

This gives us an overall product of

$$\begin{aligned}
p(x)q(x) &= c_0 + c_1 x + c_2 x^2 + c_3 x^3\\
&= 4 - x - 4x^2 + x^3
\end{aligned}$$

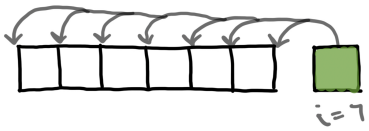
# Problem 4

Recall the InsertionSort algorithm from the lectures (see also CLRS Chapter 2.1). The algorithm maintains the following invariant: at round $i$, when element $A[i]$ is examined, elements $A[1], \ldots, A[i-1]$ are sorted. Then element $A[i]$ is compared with $A[i-1], A[i-2], \ldots$ until the correct position for it is found, and then it is inserted in that position. The worst-case running time of InsertionSort is $\Theta(n^2)$.

What is the average-case running time of the algorithm, assuming that the $n$ elements are distinct and all $n!$ orderings of them are equally likely (as in Lecture 6)? Prove an appropriate asymptotic bound. You may assume that the running time of the algorithm is the number of comparisons that it performs during its execution.

## Solution

Let's start with an example. Suppose we examine an execution of InsertionSort on a list $A$ with $i = 7$. Assume that elements $A[1], \cdots, A[6]$ are already sorted, and we're trying to figure out where $A[7]$ should be inserted. There are seven possible locations where $A[7]$ might be inserted; see the image below.




Since the $n$ elements are distinct and all $n!$ orderings are equally likely, the relative order of $A[1], \ldots, A[7]$ is equally likely to be any of the 7! orderings of these seven elements. So $A[7]$ is equally likely to be the smallest, the second smallest, …, or the largest of them, and the probability that $A[7]$ is inserted into any particular location is $1/7$. How many comparisons are required for each insert position? The first position (counting from the right) requires only a single comparison, with $A[6]$. The second position requires two comparisons, one with $A[6]$ and one with $A[5]$. In general, the $k$th position (counted from the right) requires $k$ comparisons – *except* when $k = 7$, which only requires $k - 1 = 6$ comparisons. (After all, there are only 6 elements to which $A[7]$ *can* be compared.) Let $X_i$ be a random variable indicating the number of comparisons

made when inserting element $A[i]$. Then the expected number of comparisons made when inserting $A[7]$ is hence as follows:

$$\begin{aligned}\mathbb{E}[X_7] &= \left(\sum_{k=1}^{6} k \cdot \mathbb{P}(\text{position } k)\right) + 6 \cdot \mathbb{P}(\text{position } 7) \\ &= \frac{1}{7}\left(1+2+3+4+5+6+6\right) \\ &= \frac{1}{7}\left(\left(\sum_{j=1}^{7} j\right) - 1\right)\end{aligned}$$

And more generally, we have that

$$\mathbb{E}[X_i] = \frac{1}{i}\left(\left(\sum_{j=1}^{i} j\right) - 1\right)$$

(As a sanity check, we can confirm by hand that our formula outputs the correct number of expected comparisons for $i = 1$ and $i = 2$: no comparison for $A[1]$, and exactly one for $A[2]$.) By linearity of expectation, the expected number of comparisons in an execution of InsertionSort is thus:

$$\begin{aligned}\mathbb{E}\left[\sum_{i=1}^{n} X_i\right] &= \sum_{i=1}^{n} \mathbb{E}[X_i] \\ &= \sum_{i=1}^{n} \frac{1}{i}\left(\left(\sum_{j=1}^{i} j\right) - 1\right) \\ &= \sum_{i=1}^{n} \frac{1}{i}\left(\frac{i}{2}(i+1) - 1\right) \\ &= \sum_{i=1}^{n} \left(\frac{i+1}{2} - \frac{1}{i}\right) \\ &= \frac{1}{2}\sum_{i=1}^{n} (i+1) - \sum_{i=1}^{n} \frac{1}{i} \\ &= \frac{1}{2}\left(\frac{n}{2}(2+(n+1))\right) - \sum_{i=1}^{n} \frac{1}{i} \\ &= \frac{n}{4}(n+3) - \sum_{i=1}^{n} \frac{1}{i}\end{aligned}$$

From here, we note that the first term is quadratic, while the second term is $-H(n)$, where $H(n)$ is the $n$th harmonic number. For any $n \geq 1$, $1 \leq H(n) \leq n$. Thus, we have:

$$\frac{n(n-1)}{4} = \frac{n}{4}(n+3) - n \leq \mathbb{E}\left[\sum_{i=1}^{n} X_i\right] = \frac{n}{4}(n+3) - H(n) \leq \frac{n}{4}(n+3) - 1,$$

so the expected number of comparisons is $\Theta(n^2)$. Thus, the average-case runtime of InsertionSort is asymptotically no better than its worst-case runtime.