Algorithms and Data Structures 2023/24
Notes for week 5 tutorial

1. There are two parts to this question.

   (i) First we show how to compute n'. This can be done in $\Theta(\lg(n))$ steps:

   Initialise x to 1;
   Keep multiplying x by 2 until we find $x \geq n$.
   Then set n' to be this final value of x.

   It is likely that some of them will have the solution $n' =_{\mathsf{def}} 2^{\lceil \lg(n) \rceil}$. This is also fine.

   Observe that we are guaranteed that $n \leq n' < 2n$.

   (ii) Once n' has been computed, we add $n' - n$ 0-coefficients for the higher-order powers $x^n, \ldots x^{n'-1}$ to the original polynomial of consideration. That is, we map $p \to p'$ by adding all these 0-terms, or more likely, we map the original vector representation

   $$\langle a_0, \ldots, a_{n-1} \rangle \to \langle a_0, \ldots, a_{n-1}, 0, \ldots, 0 \rangle.$$

   Then we compute $\mathrm{FFT}_{n'}$ in $\Theta(n' \lg(n'))$ time.

   The time taken to pad the original polynomial with extra 0s is linear in $n' - n$, and therefore negligible in regard to $\Theta(n' \lg(n'))$. Now suppose that c_1 is the constant of the Ω side of the $\Theta(n' \lg(n'))$ and c_2 is the constant of the O side. So the running-time $T(\mathrm{FFT}_{n'})$ of FFT on the padded-polynomial satisfies

   $$c_1 \cdot n' \lg(n') \leq \; T(\mathrm{FFT}_{n'}) \leq \; c_2 \cdot n' \lg(n')$$

   Well, then by $n \leq n'$ we also have $c_1 n \lg(n) \leq T(\mathrm{FFT}_{n'})$, hence our overall algorithm is $\Omega(n \lg(n))$ with respect to the constant c_1.

   Now applying $n' < 2n$, we can write

   $$\begin{aligned} T(\mathrm{FFT}_{n'}) &\leq c_2 \cdot n' \lg(n') \\ &< c_2 \cdot 2n(\lg(2n)) \\ &= c_2 \cdot 2n(\lg(n) + 1) \quad \leq \quad 4c_2 n(\lg(n))) \end{aligned}$$

   assuming $n \geq 2$. Therefore our algorithm is also $O(n \lg(n))$ (in this case via the constant $4c_2$).

   Hence our overall algorithm is $\Theta(n \lg(n))$ (regardless of power-of-2), as required.

2. Will just do (b) as an example.

   We start with the expression $2i(i+1)^2 + 4(i+1)^3$.

   Let's consider $2i(i+1)^2$ first. If we calculate $(i+1)^2$, we get $(i^2 + 2i + 1)$, which is $-1 + 2i + 1$, which is $2i$. If we then calculate $2i(2i)$, this is $4i^2$, which is -4.

   For the second expression, we already know $(i+1)^2 = 2i$, so $4(i+1)^3$ is $4(i+1)2i$, which is $8(i^2 + i) = 8(-1+i) = 8i - 8$.

   Then adding these two together, we have $-4 + 8i - 8$, which is $8i - 12$.

3. Tutors, just apply the formula.

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

   Answer: Following the suggested steps:

   (a) Degree of product poly will be 3, so we need 4th roots of unity.

   (b) Roots are $\omega_4^0 = 1, \omega_4^1 = i, \omega_4^2 = -1, \omega_4^3 = -i$.

   (c) Evaluate $p(x)$ at $x = 1 \Rightarrow p(1) = -3$. For $x = \omega_4^1 = i$, I get $p(i) = i - 4$. For $x = -1$, we get $p(-1) = -5$. For $x = -i$ we get $p(-i) = -i - 4$.
   So the DFT$_4$ of p is $\langle -3, i-4, -5, -i-4 \rangle$.

   (d) Same way we get $\langle 0, -2, 0, -2 \rangle$ for the DFT$_4$ of q.

   (e) Multiply these DFTs in a pointwise fashion to get the following DFT$_4$ for pq.

   $$\langle 0, 8-2i, 0, 8+2i \rangle$$

   (f) Finally compute the inverse DFT to recover the coefficients of pq. We follow the rules for DFT^{-1} from slide 22 of Lectures 5 & 6.

   (i) Compute $z =$DFT$_4(\langle 0, 8-2i, 0, 8+2i \rangle)$, (yes, that's right: forwards DFT, not inverse DFT).
   Now, because the vector z is only length 4 and quite a simple one, it would not be too difficult to just compute DFT$_4(z)$ by substitution. But for practice we will use the FFT recurrence to compute it.
   As we know the vector z defines a polynomial

   $$Z(x) = z_0 + z_1 x + z_2 x^2 + z_3 x^3,$$

   and that the DFT is defined in terms of this polynomial.

So, back to slide 9; and we know that $Z_{even}(y) = z_0 + z_2 y$ and $Z_{odd}(y) = z_1 + z_3 y$. That means that we will want $\mathsf{DFT}_2(\langle z_0, z_2 \rangle)$ and also $\mathsf{DFT}_2(\langle z_1, z_3 \rangle)$. (note that we could also have determined this is what we need by checking the Algorithm (lines 4 and 5) of slide 13 of the lectures 5-6).

Now $\langle z_0, z_2 \rangle = \langle 0, 0 \rangle$.

Also $\langle z_1, z_3 \rangle = \langle 8 - 2i, 8 + 2i \rangle$.

It is trivial that $\mathsf{DFT}_2(\langle 0, 0 \rangle) = \langle 0, 0 \rangle$.

By substitution (of 1 and of $\omega_2^1 = -1$), $\mathsf{DFT}_2(\langle 8 - 2i, 8 + 2i \rangle) = \langle 16, -4i \rangle$.

Next we apply the rules of the FFT recurrence on slide 10 (for $n = 4$):

- $w_4^1 = 1$:

$$\begin{aligned} Z(1) &= Z_{even}(1) + 1 \cdot Z_{odd}(1) \\ &= 0 + 1 \cdot 16 = 16. \end{aligned}$$

- $w_4^1 = i$:

$$\begin{aligned} Z(i) &= Z_{even}(i^2) + i \cdot Z_{odd}(i^2) \\ &= Z_{even}(-1) + i \cdot Z_{odd}(-1) = 0 + i(-4i) = 4. \end{aligned}$$

- $w_4^2 = -1$:

$$\begin{aligned} Z(-1) &= Z_{even}((-1)^2) - 1 \cdot Z_{odd}((-1)^2) \\ &= Z_{even}(1) - 1 \cdot Z_{odd}(1) = 0 - (16) = -16. \end{aligned}$$

- $w_4^3 = -i$:

$$\begin{aligned} Z(-i) &= Z_{even}((-i)^2) - i \cdot Z_{odd}((-i)^2) \\ &= Z_{even}(-1) - i \cdot Z_{odd}(-1) = 0 - i(-4i) = -4. \end{aligned}$$

So

$$z = \langle 16, 4, -16, -4 \rangle.$$

(ii) Next (following the rule on slide 22 of lectures 5-6) we must compute $z' = \langle z_0/4, z_3/4, z_2/4, z_1/4 \rangle$. z' will be the list of coefficients for pq in least significant order.

Note we are *deliberately* reversing the **final 3 entries** (this would be **final $n - 1$ entries** for larger n, first entry is the only one that stays there). Check your notes to see why this is the case.

With our vector z above, this gives

$$z = \langle 4, -1, -4, 1 \rangle.$$

So the polynomial $pq(x)$ is

$$pq(x) = 4 - x - 4x^2 + x^3.$$

If we check back to the original polynomials p, q and multiply direct, we verify that our DFT computation is correct