HQ5

Name: Debnath Kundu (MT22026) & Manvendra Kumar Nema (MT22038)

Answer (1) :

1. def **Evaluate**(A[0…d], v): //returns the value of the polynomial represented by the coefficients in array A at value=v
2.   If A.size == 1:
3.   return A[0]
4.   for(i=1 to d/3) // degree d=3^k-1 for some known k.
5.   D1[i]=A[3*i]
6.   D2[i]=A[3*i + 1]
7.   D3[i]=A[3*i + 2]
8.   **return (Evaluate** (D1,v^3) + **v** * **Evaluate**(D2,v^3) + $\mathbf{v^2}$ * **Evaluate**(D3,v^3)**)**

The **Evaluate** function is called with the array **A**[0…d] & the value **v. (v** is the value at which the polynomial has to be evaluated.**)**

Line 2-3 *(Base Condition)***:** If the size of the polynomial reduces to one single coefficient, then the function returns it.

Line 4-7 *(Pre-processing & Dividing)***:** The input array is divided into 3 sub-arrays D1,D2 & D3 of degree at most d/3, since the degree **d** is guaranteed to be a multiple of 3.

D1 contains those **A**[i], where (i mod 3) == 0, hence: D1[A_0 , A_3 , A_6 ...A_{d-2}]
D2 contains those **A**[i], where (i mod 3) == 1, hence: D2[A_1 , A_4 , A_7 …A_{d-1}]
D3 contains those **A**[i], where (i mod 3) == 2, hence: D3[A_2 , A_5 , A_8 …A_d]

Line 8 (*Delegate & Combine*)**:** It recursively evaluates the polynomial at **v** by dividing it into 3 smaller polynomials each of size one-third of the original size.

Let $P_d(x) = a_0 + a_1 x^1 + a_2 x^2 + \cdots + a_{d-1} \cdot x^{d-1} + a_d \cdot x^d$

We defined $D1(x) = a_0 + a_3 x^3 + a_6 x^6 + \cdots + a_{d-2}\, x^{d-2}$

Let $D1'(y) = a_0 + a_3 \cdot y^1 + a_6 x^2 + \cdots + a_{d-2} \cdot x^{(d-2)/3}$

If we put $y = x^3$, then $D1'(y)$ evaluates same as $D1(x)$

$\therefore \cancel{D1'(y) = D}\ D1'(y) = D1'(x^3) = D1(x)$ — (1)

Similarly, $D2(x) = a_1 x^1 + a_4 x^4 + a_7 x^7 + \cdots + a_{d-1} \cdot x^{d-1}$

Let $D2'(x) = a_1 + a_4 x^3 + a_7 \cdot x^6 + \cdots + a_{d-1} \cdot x^{d-2}$ (2)

Now, $x \cdot D2'(x) = x \cdot (a_1 + a_4 x^3 + a_7 \cdot x^6 + \cdots + a_{d-1} \cdot x^{d-2}) = D2(x)$

Similarly, $D3(x) = a_2 x^2 + a_5 x^5 + a_8 x^8 + \cdots + a_d \cdot x^d$

Let $D3'(x) = a_2 + a_5 x^3 + a_8 x^6 + \cdots + a_d\, x^{d-2}$

$\therefore x^2 \cdot D3'(x) = x^2 \cdot (a_2 + a_5 x^3 + a_8 x^6 + \cdots + a_d \cdot x^{d-2})$ — (3)

From st. (1), (2) & (3):—

$P_d(x) = D1(x) + D2(x) + D3(x)$

$= D1'(x^3) + x \cdot D2'(x^3) + x^2 \cdot D3'(x^3)$.

Thus the larger **d**-degree polynomial is reduced to **3** smaller **d/3** degree sub-polynomials, thereby implementing *divide & conquer* approach, as well as reducing multiplication overhead as we are evaluating the polynomial at $\mathbf{v^3}$ & not at **v.**

Proof of Complexity:

Let **T(d)**: Time complexity of **Evaluate**(A[0…d], v)

Line 3 either takes constant time operation or is a no-op when the conditional at **Line 2** returns False.

Base Case: When the size of the polynomial reduces to one single coefficient, then the function returns it. Hence, $T(1) = 1 = c(say)$

The for-loop through **Line 4-7** iterates d/3 times & hence takes **O(d)** time.

Line 8 recursively calls the same method thrice with a reduced input size of d/3. Hence, it takes **3 * T(d/3)** time.

Therefore, $T(d) = 3 * T(d/3) + O(d)$**.**

The solution of the recurrence is **O(d * log(d)).** [By Master's Theorem]

Answer (2a) :

1. def **PolynomialEvaluation**(A[0…d]): //returns the DFT of the polynomial of degree d represented by the coefficients in array A.
2. if A.size == 1 :
3.   return A[0]
4. $\omega = e^{(2*\pi*i)/d}$ // ω a d-th root of unity.
5. X = 1
6. Y = $\omega^{d/3}$
7. Z = $\omega^{(2*d)/3}$
8. for(i=0 to d/3) // degree d=3^k-1 for some known k.
9.   temp1[i]=A[3*i]
10.   temp2[i]=A[3*i + 1]
11.   temp3[i]=A[3*i + 2]
12. D1[0,1.2…(d/3)] = **PolynomialEvaluation** (temp1)
13. D2[0,1.2…(d/3)] = **PolynomialEvaluation** (temp2)
14. D3[0,1.2…(d/3)] = **PolynomialEvaluation** (temp3)
15. for(i=0 to d/3):
16.   A [i] = D1[i] + X*D2[i] + X*ω*D3[i]
17.   A [i +d/3] = D1[i] + Y*D2[i] + Y*ω*D3[i]
18.   A [i +(2*n)/3] = D1[i] + Z*D2[i] + Z*ω*D3[i]
19.   X = X*ω
20.   Y = Y*ω
21.   Z = Z*ω
22. return A

The **PolynomialEvaluation** function is called with the array **A**[0…d] & it returns the DFT of A(x), i.e, the values { A(a) : a is a d-th root of unity }.

Line 2-3 *(Base Condition)***:** If the size of the polynomial reduces to one single coefficient, then the function returns it.

Line 4-7 *(Initialisation)***:** Initializing the values of ω to **X,Y** & **Z**.

These variables will update the values of ω_d accordingly, which will be further used in the polynomial evaluation.

Line 8-11 *(Dividing)***:** The input array is divided into 3 sub-arrays temp1, temp2 & temp3 of degree at most d/3, since the degree **d** is guaranteed to be a multiple of 3.

temp1 contains those **A**[i], where (i mod 3) == 0, hence: D1[A_0 , A_3 , A_6 ...A_{d-2}]
temp2 contains those **A**[i], where (i mod 3) == 1, hence: D2[A_1 , A_4 , A_7 …A_{d-1}]
temp3 contains those **A**[i], where (i mod 3) == 2, hence: D3[A_2 , A_5 , A_8 …A_d]

Line 12-14 *(Delegating)***:** It recursively evaluates the polynomial at the above points (temp1, temp2 & temp3) by dividing it into 3 smaller polynomials each of size one-third of the original size & stores them respectively into the arrays D1, D2 & D3.

Line 15-21 *(Combining)***:**
In line 16, A is updated at indexes A_0 , A_3 , A_6 ...A_{d-2} in respective iterations through $\omega^0, \omega^3, \omega^6, \ldots \omega^{d-2}$
In line 17, A is updated at indexes A_1 , A_4 , A_7 …A_{d-1} in respective iterations through $\omega^1, \omega^4, \omega^7, \ldots \omega^{d-1}$
In line 18, A is updated at indexes A_2 , A_5 , A_8 …A_d in respective iterations through $\omega^2, \omega^5, \omega^8, \ldots \omega^d$.

The general updation formula in the above 3 lines is: $P(x) = D1(x^3) + x * D2(x^3) + x^2 * D3(x^3)$ [From question 1 (ref. previous question)]
X,Y & Z are being updated accordingly to vary the index of ω**.**

<u>Proof of Complexity</u>:
Let **T(d)**: Time complexity of **PolynomialEvaluation**(A[0…d]) where d is the degree of polynomial.
Line 3 either takes constant time operation or is a no-op when the conditional at **Line 2** returns False. Hence, **O(1)** time.
Base Case: When the size of the polynomial reduces to one single coefficient, then the function returns it. Hence, $T(1) = 1 = c(say)$
Line 4-7: Initialisation of variables, hence it takes constant time operation.
The for-loop through **Line 8-11** iterates d/3 times & hence takes **O(d)** time.
Line 12-14 recursively calls the same method thrice with a reduced input size of d/3. Hence, it takes **3 * T(d/3)** time.
The for-loop through **Line 15-21** iterates d/3 times to update the array A, hence takes **O(d)** time

Hence, T(d) = O(1) + O(d) + 3*T(d/3) + O(d)

Therefore**,** $T(d) = 3 * T(d/3) + O(d)$**.**

The solution of the recurrence is **O(d * log(d)).** [By Master's Theorem]