

Q1

We can handle this problem by a divide and conquer recursion and use repeated squaring.

When we want to compute M^n , we could compute $y = M^{\lfloor \frac{n}{2} \rfloor}$. Here, we compute $\frac{n}{2}$ by flooring. Then assign $\lfloor \frac{n}{2} \rfloor$ as the new N and do next recursion. Every recursion we return a value, if n is even, we return y^2 , otherwise n is odd, we return $y^2 * M$ as we do floor operation for $n/2$.

Doing those, until $n = 0$ and this is a boundary condition and return 1. Since each recursion reduces the exponent by half, the number of recursive layers is $O(\log n)$, and the algorithm can get results in a very short time.

The Pseudocode shows below:

Algorithm 1: An algorithm

```

1 Function Main( $M, n$ ):
2   | return quickMul( $M, n$ )
3 end
4 Function quickMul ( $M, n$ ):
5   | if  $n$  equals 0 then
6     | return 1
7   | end
8   |  $y = \text{quickMul}(n/2)$ 
9   | if  $n$  is even then
10    | return  $y * y$ 
11  | end
12  | if  $n$  is odd then
13    | return  $y * y * M$ 
14  | end
15 end

```

Q2

Note that using the substitution $y = x^{100}$ reduces $P(x)$ to $P^*(y) = A_0 + A_1y + A_2y^2$. The product of $R^*(y) = P^*(y)P^*(y)$ of these two polynomials is of degree 4 so to uniquely determine $R^*(y)$ we need 5 of its values. Thus, we evaluate $P^*(y)$ at 5 values of its argument x , by letting $x = -2, -1, 0, 1, 2$. We then obtain from these 5 values of $R^*(y)$ its coefficients, by solving the corresponding system of linear equation in coefficients r_0, \dots, r_4 such that $R^*(j) = r_0 + r_1x + \dots + r_4x$. Thus we solve the system $\sum_{j=0}^4 r_j i^j = R^*(i) : -2 \leq i \leq 2$. We now form the polynomial $R^*(j) = r_0 + r_1x + \dots + r_4x$ with thus obtained r_j and finally substitute back y with x^{100} obtaining $R(x) = P(x)P(x)$.

Q3

First, we should know if we do continuously inner product, time complexity would be $O(n^2)$. Look at the figure and combine the lecture slides, we use convolution.

Let N' be the net sequence N in the reverse order. Here we should notice that the flipping of the array is not an actually flipping of the net and it doesn't change anything. It is purely an approach to help

is visualise how we obtain the different coefficients which comprise the convolution which in the end is what we are trying to calculate. So we can do convolution of the sequence $C = A * N'$, in order to do convolution, we first transfer sequence form to $P_A(x)$, $P'_N(x)$, then compute the DFT followed by multiplication, and then use inverse transformation for DFT to recover the coefficients of the product polynomial $P_C(x)$, thus, we got the sequence of C . The sequence like

$$C_0 = A_0 * N_0$$

$$C_1 = A_0N_1 + A_1N_0$$

...

$$C_{k+m} = A_kN_m + A_{k+1}N_{m-1} + \dots + A_{k+m}N_0$$

...

$$C_{m+n} = A_nN_m$$

So that we can get a sequence of fish numbers with length $100n + 10$ (the length of net). What we do is just use *for* loop to find the largest possible number of fish in time $O(n)$. And we know convolution in time $O(n \log n)$.

Thus, total time runs in $O(n \log n) + O(n)$ which is $O(n \log n)$.

Q4

(a)

Denote $A = \langle 1, \underbrace{0, 0, \dots, 0}_k, 1 \rangle$, the corresponding polynomial is $P_A(x) = 1 + x^{k+1}$. So the question can be thought as compute the convolution $P_A(x) * P_A(x)$. Thus, we simply multiply them because the form of two polynomials are easily to compute:

$$P_A(x)P_A(x) = 1 + 2x^{k+1} + x^{2k+2}$$

So the convolution of these two sequences is

$$(1, \underbrace{0, 0, \dots, 0}_k, 2, \underbrace{0, 0, \dots, 0}_k, 1)$$

(b)

As the above question mentioned, this sequence produces polynomial $P_A(x) = 1 + x^{k+1}$. Consequently, the DFT is equal to

$$\begin{aligned} DFT(A) &= \langle 1 + \omega_{k+2}^{0*(k+1)}, 1 + \omega_{k+2}^{k+1}, 1 + \omega_{k+2}^{2(k+1)}, \dots, 1 + \omega_{k+2}^{(k+1)(k+1)} \rangle \\ &= \langle 2, 1 + \omega_{k+2}^{k+1}, 1 + \omega_{k+2}^{2(k+1)}, \dots, 1 + \omega_{k+2}^{(k+1)(k+1)} \rangle \end{aligned}$$

Q5

The result sequence of $x * \langle 1, 1, -1 \rangle$ (denote P_B) is $\langle 1, 0, -1, 2, -1 \rangle$, and the corresponding polynomial is $P_C(x) = 1 - x^2 + 2x^3 - x^4$.

First, the resulting polynomial $P_C(x)$ length $(n-1) + (m-1) + 1 = 5$ as $m = 3$, so n should be 3.

Now, assume x has the sequence like $\langle x_1, x_2, x_3 \rangle$ and the first coefficient of $P_C(x)$ is 1 which equals to $x_1 * 1$ (first coefficient of P_B), so $x_1 = 1$.

And, the first coefficient of $P_C(x)$ is 0 which equals to x_1 multiply the second coefficient of P_B plus x_2 multiply the first coefficient of P_B which is $1 * 1 + x_2 * 1 = 0$ so, $x_2 = -1$.

Last, the fifth coefficient of $P_C(x)$ equals to x_3 multiply the third coefficient of P_B is -1 which is $x_3 * -1 = -1$, so $x_3 = 1$.

Actually, I use part of convolution coefficients formula to solve this problem, also we can use convolution to prove that sequence s is correct.

Thus, the sequence $\langle 1, -1, 1 \rangle$ can be satisfied.