Yuan Gao z5239220 Q3

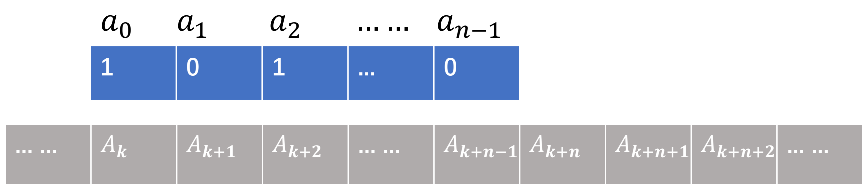
Assume you are given a map of a straight sea shore of length $100n$ meters as a sequence on $100n$ numbers such that A_i is the number of fish between i^{th} meter of the shore and $(i+1)^{th}$ meter, $0 \leq i \leq 100n - 1$. You also have a net of length n meters but unfortunately it has holes in it. Such a net is described as a sequence N of n ones and zeros, where 0's denotes where the holes are. If you throw such a net starting at meter k and ending at meter $k+n$, then you will catch only the fish in one meter stretches of the shore where the corresponding bit of the net is 1. Find the spot where you should place the left end of your net in order to catch the largest possible number of fish using an algorithm which runs in time $O(nlogn)$.

According to the question, there are:

Straight seashore (vector): $A = (A_0, A_1, A_2, A_3 \ldots \ldots, A_{100n-1})$

Net (step length): n

Weight: $a = (a_0, a_1, a_2 \ldots \ldots, a_{n-1})$ where $a_i = 0\ or\ 1$



Let C as catch the number of fish:

$$C_k = \sum_{i=0}^{n-1} a_i A_{k+i}$$

That is the number of caught fish started A_k and end of A_{k+n-1}.

Thus, we can think of A and a as two polynomials and do convolution.

$$\boldsymbol{C} = (C_0, C_1, C_2 \ldots \ldots C_{101n-2})$$

Because we need the forward match, net vector should be inverted. (a' as reverse order of net array).

Hence, we can get a vector $\boldsymbol{C}$ from $A * a'$, and the maximum value of $\boldsymbol{C}$ can be the largest possible number of fish.

The brute force algorithm of $A * a'$ would be $O(n^2)$, therefore we use FFT to reduce time complexity.

Firstly, there are the formula of DFT and IDFT

$$DFT \rightarrow y_i = \sum_{j=0}^{n-1} w_n^{ij} a_j$$

$$IDFT \rightarrow a_i = \frac{1}{n} \sum_{j=0}^{n-1} w_n^{-ij} y_j$$

Let the net $\boldsymbol{a}$ as an example:
Assume $a(x) = a_0 + a_1x + a_2x^2 \dots\dots, a_{n-1}x^{n-1}$
There is vector $\boldsymbol{a} = [a_0, a_1, a_2 \dots\dots, a_{n-1}]^T$ can be spilt to

$$\begin{cases} even(i\%2 = 0)\colon a^{[0]}[a_0, a_2, a_3 \dots\dots a_{n-2}]^T \to a^{[0]}(x) = a_0 + a_2x + a_4x \dots\dots a_{n-2}x^{\frac{n}{2}-1} \\ odd(i\%2 = 1)\colon a^{[1]}[a_1, a_3, a_5 \dots\dots a_{n-1}]^T \to a^{[1]}(x) = a_1 + a_3x + a_5x \dots\dots a_{n-1}x^{\frac{n}{2}-1} \end{cases}$$

Square x, we have

$$\begin{cases} a^{[0]}(x^2) = a_0 + a_2x^2 + a_4x^4 \dots\dots a_{n-2}x^{n-2} \\ a^{[1]}(x^2) = a_1 + a_3x^2 + a_5x^4 \dots\dots a_{n-1}x^{n-2} \end{cases}$$

Then,$xa^{[1]}(x^2) = a_1x + a_3x^3 + a_5x^5 \dots\dots a_{n-1}x^{n-1}$
We can get: $a(x) = a^{[0]}(x^2) + a^{[1]}(x^2)$
Now we change x to complex roots of unity w_n^k

$$a(w_n^k) = a^{[0]}((w_n^k)^2) + w_n^k a^{[1]}((w_n^k)^2)$$

There are

$$\begin{cases} a(w_n^k) = a^{[0]}((w_n^k)^2) + w_n^k a^{[1]}((w_n^k)^2) \\ a(w_n^{k+\frac{n}{2}}) = a^{[0]}((w_n^{k+\frac{n}{2}})^2) + w_n^{k+\frac{n}{2}} a^{[1]}((w_n^{k+\frac{n}{2}})^2) \end{cases}$$

According to Cancelation Lemma and Elimination Lemma

$$(w_n^{k+\frac{n}{2}})^2 = (w_n^k)^2 = w_{\frac{n}{2}}^k \to w_n^{k+\frac{n}{2}} = -w_n^k$$

$$\begin{cases} a(w_n^k) = a^{[0]}(w_{\frac{n}{2}}^k) + w_n^k a^{[1]}(w_{\frac{n}{2}}^k) \\ a(w_n^{k+\frac{n}{2}}) = a^{[0]}(w_{\frac{n}{2}}^k) + w_n^{k+\frac{n}{2}} a^{[1]}(w_{\frac{n}{2}}^k) = a^{[0]}(w_{\frac{n}{2}}^k) - w_n^k a^{[1]}(w_{\frac{n}{2}}^k) \end{cases}$$

We can start $DFT_n(a)$ by divide and conquer

$$DFT_n(a) \begin{cases} DFT_{\frac{n}{2}}(a^{[0]}) \\ DFT_{\frac{n}{2}}(a^{[1]}) \end{cases}$$

This step $T(n) = 2T\left(\frac{n}{2}\right) + O(n) = O(nlogn)$

And by the same logic
The time complexity $DFT_{100n}(A)$ is $O(nlogn)$

According to the formula of IDFT
Changing the i to $-i$ of complex roots of unity the IDFT can be done

Hence,

$$DFT_{101n-1}(a') \to (a_{n-1}, a_{n-2}, a_{n-3} \dots \dots, a_0, \underbrace{0 \dots, 0}_{100n-1})$$

$$DFT_{101n-1}(A) \to (A_0, A_1, A_2 \dots \dots, A_{100n-1}, \underbrace{0 \dots, 0}_{n-1})$$

$$A * a' = IDFT_{101n-1}(DFT_{101n-1}(A) \cdot DFT_{101n-1}(a'))$$

Then, we just cost O(n) to find the maximum value in vector $\boldsymbol{C} = A * a'$

The total time complexity is $O(nlogn)$