CHEFEQUA - EDITORIAL

general chefequa, divide-and-conq, generating_functions, interpolation, medium-hard, nov18, ntt, taran_1407

taran_1407 #1 November 23, 2018, 9:56pm

PROBLEM LINK:

Practice
Contest: Division 1
Contest: Division 2

Setter: Xiuhan Wang
Tester: Zhong Ziqian
Editorialist: Taranpreet Singh

DIFFICULTY:

Hard

PREREQUISITES:

Generating Functions, Multipoint Evaluation, and Interpolation using Number Theoretic Transformation.

PROBLEM:

Given M = 998244353 and sequence A and C of length N each, with it being known, that

$$
\begin{equation}
C_i = {\textstyle \sum^{N-1}_{j=0}} A^j_i B_j (\bmod M)
\end{equation}
$$

holds for each valid i with some coefficients B_i for all $0 \leq i \leq N-1$, $0 \leq B_i < M$. We need to find these coefficients.

SUPER QUICK EXPLANATION

- Represent sequence C as a generating function C(x) = C_0+C_1*x+C_2^2*x^2 \ldots and by substituting values of C_i and term shifting, we obtain \displaystyle C_0+C_1*x+C_2^2*x^2+\ldots = \sum_{i=0}^{N-1} \frac{B_i}{1-A_i*x}.
- Multiplying both sizes by \prod (1-A_i*x) both sides, and reducing Numberator on left side to get a polynomial P(x) of degree N-1, we divide back \prod (1-A_i*x) and we apply Cover up method for Partial Fractions to get B_i = -A_i*\frac{P(x)}{\prod (1-A_i*x)}.
- Apply multi-point evaluation for all values A_i using Number Theoretic Transformations and Divide and Conquer.

EXPLANATION

So, we have this relation

\begin{equation}
C_i = {\textstyle \sum^{N-1}_{j=0}} A^j_i B_j (\bmod M)
\end{equation}

Let us represent sequence C as generating function C(x) = C_0+C_1*x+C_2^2*x^2 \ldots . We have values of C_i for all 0 \leq i \leq N-1. Let us try to substitute these values into this function. We get

\begin{equation}
C(x) = C_0+C_1*x+C_2*x^2+\ldots = B_0*(1+A_0*x+A_0^2*x^2 +\ldots) + B_1*(1+A_1*x+A_1^2*x+\ldots)+B_2*(1+A_2*x+A_2^2*x^2+\ldots
\end{equation}

We know from properties of Generating functions, we can represent P(x) = 1+c*x+c^2*x^2 +\ldots in its closed form by infinite GP sum as \displaystyle \frac{1}{1-c*x}.

Hence, we can represent C(x) as:

\begin{equation}
\displaystyle C(x) = \sum \frac{B_i}{1-A_i*x}
\end{equation}

Now, if we expand the right side, we will get

\begin{equation}
\displaystyle C(x) = \frac{\sum_{i=0}^{N-1} B_i*\prod_{j \neq i} (1-A_j*x)}{\prod_{i=0}^{N-1} (1-A_i*x)}
\end{equation}

Let us take Q(x) = \prod_{i=0}^{N-1} (1-A_i*x) and multiply it both sides.

On left side we have C(x)*Q(x) which is a polynomial of degree 2*N-1 while on right side, we have \sum_{i=0}^{N-1} B_i*\prod_{j \neq i} (1-A_j*x) which is a polynomial of degree N-1.

Since both sides are equal, we have all coefficients of C(x)*Q(x) with power \geq N-1 as zero, getting a polynomial of degree N-1 having N coefficients C(x)*Q(x) = P(x).

Once again, dividing both sides by Q(x), we get

\begin{equation}
\displaystyle \frac{P(x)}{Q(x)} = \sum_{i=0}^{N-1} \frac{B_i}{1-A_i*x}
\end{equation}

This is a form of Partial Fraction Decoomposition. Interesting this about Q(x) is that due to A_i \neq A_j for all i \neq j, we have Q(x) as a product of distinct linear factors which allows us to use Heaviside's Cover-up Method, read here.

Let us differentiate Q(x). We get Q'(x) = \displaystyle \sum_{i=0}^{N-1}-A_i*\prod_{i\neq j}(1-A_j*x).

It can be seen that by using cover up method, we can evaluate \displaystyle B_i = \frac{P(x)}{Q(x)/(1-A_i*x)}.

Now, We can see, that Q'(1/A_i), we get -A_i*\prod_{j \neq i}(1-A_j*x), so, we can write Q(x)/(1-A_i*x) as \frac{Q'(x)}{-A_i}, leading to

\begin{equation}
\displaystyle B_i = \frac{-A_i*P(1/A_i)}{Q'(1/A_i)}
\end{equation}

We can see, that for any polynomial f(x) of degree N-1, f(1/x) = \frac {f(x)}{x^{N-1}} where g(x) is formed by reversing the coefficients of f(x).

This way, we can define polynomials R(x) = -P(1/x)*x^{N-1} (Negative sign to make our expression simpler) and S(x) = Q(1/x)*x^{N-1} to get

\begin{equation}
B_i = \frac{A_i*R(A_i)}{S(A_i)}
\end{equation}

All we are left to solve this problem is to compute R(x) and S(x) at N distinct values efficiently, which itself was present as a problem here, having a detailed editorial here, as well as a good resource here. Another link (though in Chinese, try to translate using google translate, can be found here on multi-point evaluation and interpolation.)

Implementation

Computing Q(x)

Suppose we represent P_{(l,r)}(x) = \prod_{i = l}^{r}(1-A_i*x). We want to compute P_{0,N-1}(x) which is same as P_{0,mid}(x)*P_{mid+1,N-1}(x) which can be computed this way called Divide and Conquer, having running time defined by Recurrence T(N) = 2*T(N/2)+O(N*logN) leading to overall O(N*log^2N) running time.

This has been discussed in detail in this editorial for problem Chef and Interval Painting Problem from February Long Challenge.

Resources

For core concepts of NTT, refer this. This two-part blog on FFT and NTT is nice for competitive Programming.

Time Complexity

Computing Q(x) can be done using divide and conquer with NTT in time O(N*log^2N), P(x) is just the convolution of Q(x) and C(x) taking O(N*logN) time, and multi-point evaluation of N points also take O(N*log^2N) time, leading to overall running time O(N*log^2N) with a high constant factor associated with Number Theoretic Transformations.

AUTHOR'S AND TESTER'S SOLUTIONS:

Setter's solution
Tester's solution

Feel free to Share your approach, If it differs. Suggestions are always welcomed. 🙂