Twin GCD

Description

You are given an array A of size N. Define $f(i)$ as the number of ways to choose two subsets $B, C \subset [1 \dots N]$ such that:

1. $B \cap C = \emptyset$,
2. $gcd(A[B_1], A[B_2], \cdots, A[B_{|B|}]) = i$, and
3. $gcd(A[C_1], A[C_2], \cdots, A[C_{|C|}]) = i$.

In other words, the number of ways to choose two subsequences each having GCD equal to i and disjoint. Here, we define $GCD(\emptyset) = 0$.

Calculate $\sum_{i=1}^{N} i \cdot f(i)$ modulo 998 244 353.

Constraints

- $2 \leq N \leq 100\,000$
- $1 \leq A[i] \leq N$

Input Format

The input is given in the following format:

```
N
A[1] A[2] ... A[N]
```

Output Format

Output a single line containing $\sum_{i=1}^{N} i \cdot f(i)$ modulo 998 244 353.

Example Input 1

```
3
3 3 3
```

Example Output 1

```
36
```

Example Input 2

```
3
3 1 2
```

Example Output 2

```
2
```

Example Input 3

```
9
4 2 6 9 7 7 7 3 3
```

Example Output 3

```
10858
```

Explanation

In the first example, all subsequences of A have a GCD of 3. There are 12 number of ways to choose B and C, where 6 of them are:

- $B = \{1\}$, $C = \{2\}$.
- $B = \{1\}$, $C = \{2, 3\}$.
- $B = \{1\}$, $C = \{3\}$.
- $B = \{1, 2\}$, $C = \{3\}$.
- $B = \{1, 3\}$, $C = \{2\}$.
- $B = \{2\}$, $C = \{3\}$.

The remaining 6 can be obtained by swapping B and C from the above.

The next pages contain solution(s) and other things related to the problem.

Solution

While we have the constraint $A[i] \leq N$, we originally put it to make the problem description a bit simpler. In fact, our solution can be modified to remove that assumption, as long as we can bound $A[i]$ to be at most 100 000. To make a distinction, we shall denote $\max(A[i])$ as M.

First, there is a straightforward DP solution with $O(M^3 \log M)$ time complexity, which may work if $M \leq 100$.

Next, the most common way to solve counting problem related to GCD can be formulated as the following. First, we define the following two functions:

- $g(i)$: number of ways to count the object we want, with their GCD being a multiple of i.
- $f(i)$: number of ways to count the object we want, with their GCD being **exactly** i.

Usually $g(i)$ is easy to compute, but $f(i)$ is not. Yet, we can calculate $f(i)$ from $g(i)$ as shown below.

```
for i = M ; i >= 1 ; i -= 1:
    f[i] = g(i)
    for j = 2*i ; j <= M ; j += M:
        f[i] -= f[j]
```

After that, we can simply use the array $f[i]$ to calculate the answer.

Suppose that we define the object we want to be "two disjoint subsequences each with GCD = i", then we can do the following: Define c_i as the number A's element which is a multiple of i. Without going into detail, c_i for all $1 \leq i \leq M$ can be calculated in $O(N + M \log M)$. Then, using inclusion-exclusion, we can define $g(i)$ as

$$g(i) = 3^{c_i} - 2 \cdot 2^{c_i} + 1$$

While this works in the first example, it will fail in the second example. This is because the way we calculate $f(i)$, instead of getting the count of "two disjoint subsequences each with GCD = i", what we will get is "two disjoint subsequences, one having GCD = i, and the other having GCD a multiple of i".

Before going into the full solution, we will show a quadratic solution which can be improved into the full solution.

In a similar manner as before, we will define the function g and f but with two parameters.

- $g(i, j)$: number of ways to choose B and C such that they are disjoint, $GCD(B)$ is a multiple of i, and $GCD(C)$ is a multiple of j.
- $f(i, j)$: similar to $g(i, j)$, but instead of a multiple, both are exactly.

In this case, $g(i,j)$ is

$$g(i,j) = 2^{c_i - c_{LCM(i,j)}} 2^{c_j - c_{LCM(i,j)}} 3^{c_{LCM(i,j)}} - 2^{c_i} - 2^{c_j} + 1$$

Meanwhile, $f(i,j)$ can be computed as the following:

```
for i = M ; i >= 1 ; i -= 1
    for j = M ; j >= 1 ; j -= 1
        f[i, j] = g(i, j)
        for k = i ; k <= M ; k += i
            for l = j ; l <= M ; l += j
                if i == k and l == j: continue
                f[i, j] -= f[k, l]
```

After that, notice that $f(i)$ in the problem statement is exactly $f(i,i)$ here. Thus, we can use it to calculate the final answer. The time complexity of this solution is $O(M^2 \log^2 M)$, which may pass if $M \leq 1\ 000$.

Now, a natural follow-up from above solution is "can we somehow use Möbius function to solve this?" In one dimension the Möbius function, $\mu(i)$, can be used to solve it as below:

```
for i = 1 ; i <= M ; i += 1
    f[i] = 0
    for j = i ; j <= M ; j += i
        f[i] += g(j) * mu(j/i)
```

One way to see it, $\mu(j/i)$ is the contribution of $g(j)$ to $f(i)$.

As we cannot make a one dimensional solution to work, we shall define Möbius function in two dimension. Let's define $\mu(i,j)$ as the contribution of $g(i,j)$ to $f(1,1)$. We can prove the following claim:

Claim: $\mu(i,j) = \mu(i)\mu(j)$.

The proof (sketch) is deferred to appendix.

Next, $\mu(i)$ for all $1 \leq i \leq M$ can be precomputed in $O(M \log M)$. Finally, we can use $\mu(i,j)$ to make the following solution:

```
for gcd = 1 to M:
  f[gcd] = 0
  for a = 1 to M / gcd:
    for b = 1 to M / gcd:
      f[gcd] += g(a*gcd, b*gcd) * mu(a, b)
```

The time complexity of the nested loops will be $O(\sum_{i=1}^{M}(M/i)^2) = O(M^2)$. However, within the function $g(a \cdot gcd, b \cdot gcd)$ we need to invoke $GCD(a,b)$ to calculate their LCM because a and b may not be coprime. Thus, the overall time complexity of this solution will be $O(M^2 \log M)$, which may pass if $M \leq 2\ 000$.

We note that it is possible to kick the $O(\log M)$ factor by memoizing the result of $GCD(a, b)$. Hence, it is possible to improve the previous solution into $O(M^2)$, which may pass if $M \leq 5\ 000$.

After this, we will show how to improve the previous solution. In a nutshell, previously we calculate $f(i)$ in $O((M/i)^2)$. We will show a method to calculate $f(i)$ in $O((M/i) \log (M/i))$, which, when summed for all $1 \leq i \leq M$, will gives us $O(M \log^2 M)$.

First, notice that in the previous solutions, we implicitly transform the calculation of $f(i)$ into a calculation of $f(1)$ of the following instance:

- Remove all elements of A which are not divisible by i.
- Divide the remaining elements by i.
- (Note) the maximum value in this instance is M/i.

Hence, to make discussion easier, from here on we assume that we are dealing with the calculation of $f(1)$. Abusing the notation, denote M as the maximum value in the current instance we are dealing with.

Note that $f(1)$ can be written as the following:

$$f(1) = \sum_{i=1}^{M} \sum_{j=1}^{M} \mu(i, j) \cdot g(i, j)$$

We will rewrite the formula into a summation of the following four terms:

$$\sum_{i=1}^{M} \sum_{j=1}^{M} \mu(i, j) \cdot 2^{c_i - c_{LCM(i,j)}} 2^{c_j - c_{LCM(i,j)}} 3^{c_{LCM(i,j)}} \quad (1)$$

$$- \sum_{i=1}^{M} \sum_{j=1}^{M} \mu(i, j) \cdot 2^{c_i} \quad (2)$$

$$- \sum_{i=1}^{M} \sum_{j=1}^{M} \mu(i, j) \cdot 2^{c_j} \quad (3)$$

$$+ \sum_{i=1}^{M} \sum_{j=1}^{M} \mu(i, j) \quad (4)$$

First, let's simplify the 4th term.

$$
\begin{aligned}
\sum_{i=1}^{M}\sum_{j=1}^{M}\mu(i,j) &= \sum_{i=1}^{M}\sum_{j=1}^{M}\mu(i)\mu(j) \\
&= \sum_{i=1}^{M}\mu(i)\sum_{j=1}^{M}\mu(j) \\
&= (\sum_{i=1}^{M}\mu(i))^2
\end{aligned}
$$

Because $\sum_{i=1}^{M}\mu(i)$ can be calculated in $O(M)$, the 4th term can be calculated in $O(M)$.

Next, we will simplify the 2nd term. Note that the 3rd term is basically the same as the 2nd term.

$$
\begin{aligned}
\sum_{i=1}^{M}\sum_{j=1}^{M}\mu(i,j)\cdot 2^{c_i} &= \sum_{i=1}^{M}\sum_{j=1}^{M}\mu(i)\mu(j)\cdot 2^{c_i} \\
&= \sum_{i=1}^{M}(\mu(i)\cdot 2^{c_i})\sum_{j=1}^{M}\mu(j) \\
&= (\sum_{i=1}^{M}\mu(i)\cdot 2^{c_i})(\sum_{i=1}^{M}\mu(i))
\end{aligned}
$$

Because $\sum_{i=1}^{M}\mu(i)\cdot 2^{c_i}$ and $\sum_{i=1}^{M}\mu(i)$ can be calculated in $O(M)$, thus the 2nd term and the 3rd term can be calculated in $O(M)$.

Finally, the tedious part. We will attempt to simplify the 1st term. First, we can rewrite it into the following:

$$
\sum_{i=1}^{M}\sum_{j=1}^{M}\mu(i,j)2^{c_i}2^{c_j}3^{c_{LCM(i,j)}}2^{-2c_{LCM(i,j)}}
$$

We will split it into 2 cases: When $LCM(i,j) \le M$ and when $LCM(i,j) > M$. We first focus on the first case.

$$= \sum_{i=1}^{M}\sum_{j=1}^{M} \mu(i,j) 2^{c_i} 2^{c_j} 3^{c_{LCM(i,j)}} 2^{-2c_{LCM(i,j)}}$$

$$= \sum_{k=1}^{M} 3^{c_k} 2^{-2c_k} \sum_{i|k} \sum_{LCM(i,j)=k} \mu(i)\mu(j) 2^{c_i} 2^{c_j}$$

We will now calculate the inner part of the formulation (i.e the summation over all pair i and j). Define:

- $d(k)$ as the sum of $\mu(j)2^{c_j}$ for all j which divides k.
- $l(k)$ as $d(k)^2$.

Observe that $l(k)$ "almost" correctly calculates the inner part of the formula. $l(k)$ will be the sum for all pair i, j such that $LCM(i,j)$ divides k. Meanwhile, what we really need is the sum for all pair i, j such that $LCM(i,j) = k$. However, we can actually obtain this from $l(k)$ and $l(i)$ for all i which divides k. Basically, it will be similar to the calculation of `f[]` before, just in the reversed order. Hence, we can calculate the first case in $O(M \log M)$.

For the second case, observe that when $LCM(i,j) > M$, then:

- $c_{LCM(i,j)} = 0$, thus $3^{c_{LCM(i,j)}} 2^{-2c_{LCM(i,j)}} = 1$.
- We already know the answer for all i, j such that $lcm(i,j) \leq M$ from the first case. Thus, we can just calculate $((\sum_{i=1}^{N} d(i))^2)$ and then substract the answer of the first case from it.

Hence, after solving the first case, we can also calculate the second case in $O(M)$.

As we can calculate all four terms in $O(M \log M)$, then $f(1)$ can be calculated in $O(M \log M)$. Hence, $f(i)$ can be calculated in $O((M/i)\log(M/i))$, giving us a solution with $O(M \log^2 M)$ time complexity, which may pass $M \leq 100\,000$. Do note that some parts of the implementation are not optimized, e.g the calculation of the powers of 2 and 3. Hence, the actual time complexity of the example implementation is closer to $O(M \log M(\log MOD + \log M))$, which is still good enough.

Discussions

Related to the claim that $\mu(i,j) = \mu(i)\mu(j)$: this looks relatively simple, so it may be well-known? Note that I actually do not have that much knowledge in this area. However, when I googled I hardly see discussions on two dimensional case like this. The closest property I know is that $\mu(ij) = \mu(i)\mu(j)$ when i and j are coprime. However, the claim in this problem works for all pair i and j.

Author

- Muhammad Ayaz Dzulfikar, National University of Singapore

Appendix

$\mu(i, j) = \mu(i)\mu(j)$ Proof Sketch

First, let's see one way to calculate Möbius function in one dimension.

```
let m be an array of size i
m[i] = 1
for j = i ; j >= 1 ; j--:
  m[j] = 0
  for k = 2*j ; k <= i ; k += j:
    m[j] -= m[k]
```

Observe that in the end, `m[1]` will be $\mu(i)$. More generally, for each $1 \leq j \leq i$, `m[j]` is $\mu(i/j)$ if j divides i and 0 otherwise.

Claim 1: for each j, the sum of `m[k]` for all k multiple of j is 0.

Proof Sketch: well-known from the property of Möbius function.

Claim 2: if we start with `m[i]` being x, then `m[j]` will be $\mu(i/j) \cdot x$.

Proof Sketch: clear from the algorithm.

Next, we will calculate $\mu(i, j)$. Similar to the one dimensional case, we can do the following:

```
let m be an array of size i x j
m[i][j] = 1
for a = i ; k >= 1 ; k--:
  for b = j ; b >= 1 ; b--:
    if a == i and b == j: continue
    m[a][b] = 0
    for k = a ; k <= i ; k += a:
      for l = b ; l <= j ; l += b:
        if a == k and b == l: continue
        m[a][b] -= m[k][l]
```

Like the one dimensional case, `m[1][1]` will be $\mu(i, j)$ and for each $1 \leq a \leq i, 1 \leq b \leq j$, `m[a][b]` will be $\mu(i/a, j/b)$. Note that from here, we assume that a divides i and b divides j (as otherwise, `m[a][b]` must be 0).

We want to note that here we already have a way to calculate $\mu(i, j)$, but its time complexity is pretty big (i.e superquadratic). This is another motivation why we want to show $\mu(i, j) = \mu(i)\mu(j)$, as this way is easier to compute.

Claim 3: `m[i][b]` will be $\mu(j/b)$ and `m[a][j]` will be $\mu(i/a)$.

Proof Sketch: clear from the algorithm, which reduces into the one dimensional case.

Now, we will prove our claim that $\mu(i, j) = \mu(i)\mu(j)$. We shall use induction: on row a, the value of `m[a][b]` will be $\mu(i/a)\mu(j/b)$. Our base case starts from row i, which is proven by Claim 3.

Do induction on row $a < i$. Our loop will process row $a, 2a, \cdots, i$. By our induction hypothesis and Claim 1, the summation on row $2a, \cdots, i$ must be 0. Thus, we only need to calculate the summation on row a. However, notice that this is basically the one dimensional case, with `m[j]` being `m[a][j]` which is $\mu(i/a)$. Due to Claim 2, `m[a][b]` will be $\mu(i/a)\mu(j/b)$. Thus, the induction works.

Hence, `m[1][1]` will be $\mu(i)\mu(j)$. Thus, our claim is proven.