Finding GCD of a set of numbers?

So, I was asked this question in an interview. Given a group of numbers (not necessarily distinct), I have to find the multiplication of GCD's of all possible subsets of the given group of numbers.

My approach which I told the interviewer:

```
1. Recursively generate all possible subsets of the given set.
2a. For a particular subset of the given set:
2b. Find GCD of that subset using the Euclid's Algorithm.
3. Multiply it in the answer being obtained.
```

Assume GCD of an empty set to be 1. However, there will be 2^n subsets and this won't work optimally if the n is large. How can I optimise it?

c++ algorithm

edited Oct 11 at 12:05

asked Oct 11 at 11:39
rohansingh
108 6

1 What do you mean by "modulus it"? What is the divisor? – Rhymoid Oct 11 at 11:42

with your 3 steps you construct **a** subset. How is this supposed to find **all** subsets? – tobi303 Oct 11 at 11:43

Also, the interviewer might want to adjust their terminology. An unordered list that allows duplicates is usually called "multiset" or "bag". – Rhymoid Oct 11 at 11:44

@Rhymoid, my bad. I was trying to divide it in terms of even and odd numbers but couldn't form anything. – rohansingh Oct 11 at 11:45

1 This is from the ongoing contest : ACM-ICPC Asia-Amritapuri regionals. s3.amazonaws.com/codechef_shared/download/ICPC/2015/… second problem from here. The contest is currently going on. So I doubt you were asked this in an interview. – sasha Oct 11 at 12:27

2 Answers

Assume that each array element is an integer in the range 1..U for some U.

Let f(x) be the number of subsets with GCD(x). The solution to the problem is then the sum of d^f(d) for all distinct factors 1 <= d <= U.

Let g(x) be the number of array elements divisible by x.

We have

```
f(x) = 2^g(x) - SUM(x | y, f(y))
```

We can compute g(x) in O(n * sqrt(U)) by enumerating all divisors of every array element. f(x) can be computed in O(U log U) from high to low values, by enumerating every multiple of x in the straightforward manner.

answered Oct 11 at 17:07
Niklas B.
48.4k 5 101 146

Pre - Requisite :

Fermat's little theorem (there is a generalised theorem too) , simple maths , Modular exponentiation

Explanation : Notations : A[] stands for our input array

Clearly the constraints 1<=N<=10^5 , tell me that either you need a O(N * LOG N) solution , dont try to think DP as its complexity according to me will be N * max(A[i]) i.e. approx. 10^5 * 10 ^ 6 . Why? because you need the GCD of the subsets to make a transition.

Ok , moving on

We can think of clubbing the subsets with the same GCD so as to make the complexity.

So , lets decrement an iterator i from 10^6 to 1 trying to make the set with GCD i !

Now to make the subset with GCD(i) I can club it with any i*j where j is a non negative Integer. Why ?

GCD(i , i*j) = i

Now ,

We can build a frequency table for any element as the number is quite reachable!

Now , during the contest what I did was , keep the number of subsets with gcd(i) at f2[i]

hence what we do is sum frequency of all elements from j*i where j varies from 1 to floor(i/j) now the subsets with a common divisor(and not GCD) as i is (2^sum - 1) .

Now we have to subtract from this sum the subsets with GCD greater than i and having i as a common divisor of gcd as i.

This can also be done within the same loop by taking summation of f2[i*j] where j varies from 1 to floor(i/j)

Now the subsets with GCD i equal to 2^sum -1 - summation of f2[ij] Just multiply i (No . of subsets with GCD i times) i.e. power (i , 2^sum -1 - summation of f2[ij]) . But now to calculate this the exponent part can overflow but you can take its % with given MOD-1 as MOD was prime! (Fermat little theorem) using modular exponentiation

Here is a snippet of my code as I am unsure that can we post the code now!

```
for(i=max_ele; i >= 1;--i)
            {
                to_add=F[i];
                to_subtract = 0 ;
                for(j=2 ;j*i <= max_ele;++j)
                    {
                        to_add+=F[j*i];
                        to_subtract+=F2[j*i];
                        to_subtract>=(MOD-1)?(to_subtract%=(MOD-1)):0;
                    }

                subsets = (((power(2 , to_add , MOD-1) ) - 1) - to_subtract)%(MOD-
1) ;

            if(subsets<0)
                subsets = (subsets%(MOD-1) +MOD-1)%(MOD-1);

            ans  = ans * power(i , subsets , MOD);
            F2[i]= subsets;
            ans %=MOD;
        }
```

I feel like I had complicated the things by using F2, I feel like we can do it without F2 by not taking j = 1. but it's okay I haven't thought about it and this is how I managed to get AC .

answered 2 days ago
Shubham Sharma
350 1 18