

## LOJ 1161 - Extreme GCD & CodeChef - COPRIME3

Given N positive integers  $a_1, a_2, \dots, a_N$ , not necessarily distinct, how many ways you can take 3 integers from the N numbers such that their GCD is 1 (They are Co-Prime). So Count the number of triples  $(i, j, k)$  such that  $1 \leq i < j < k \leq N$  and  $\text{GCD}(a_i, a_j, a_k) = 1$ .

Given an integer  $N (1 \leq N \leq 10^5)$ . The next line contains  $N$  integers  $a_1, a_2, \dots, a_N$ , separated by spaces. The integers will be positive and not greater than  $1000000 (1 \leq a_i \leq 10^6)$ .

Print the number of ways you can take the integers as mentioned above.

### Sample input:

```
5
1 2 4 6 8
4
2 4 6 1
```

### Sample Output:

```
6
3
```

Idea-

- From N elements we can select any three in  ${}^N C_3$  ways. In all of these triple's GCD may 1, or 2, or 3 or 4 ... or  $10^6$ .
- So, to calculate the result we have to subtract all the triples whose GCD is "exactly" 2,3,4,... $10^6$  from  ${}^N C_3$  ways.
- But it is difficult to calculate numbers of triples of GCD 2,3,4,... $10^6$  efficiently.
- Rather we calculate how many triples are there whose GCD are "atleast" 2,3,4,..., $10^6$ . That is how many triples are divisible by 2, divisible by 3, divisible by 4 ... divisible by  $10^6$ . Then subtract( GCD-2,3,5,30 etc, number of odd prime factors) or add(GCD-6,10,14,15, etc number of odd prime factors) all the triples from  ${}^N C_3$  ways. and do not add or subtract for (GCD-4,12,18 etc divisible by any perfect square numbers i.e. have a prime factor  $p^k$  where  $k>1$ ).
- We can use inclusion exclusion principle and Mobius function to solve the problems.

---

### Sample Code:

```
#define ll long long
ll a[100005], c[100005][4];
ll d[1000005], mob[1000005], vis[1000005];
void nCr()
{
    c[0][3]=c[1][3]=c[2][3]=0;
    for(ll i=3; i<=100000; i++)
    {
        ll v = (i*(i-1)*(i-2)) / (1*2*3);
        c[i][3]=v;
    }
}
```

```

void mobius()
{
    for(int i=1; i<=1000000; i++)mob[i]=1;
    for(ll i=2; i<=1000000; i++)
    {
        if(vis[i]==0)
        {
            mob[i]=-1;
            for(ll j=i+i; j<=1000000; j+=i)
            {
                if(j%(i*i)==0)mob[j]=0;
                mob[j] *= (-1);
                vis[j]=1;
            }
        }
    }
}

int main()
{
    mobius();
    nCr();

    int n; scanf("%d", &n);
    for(int i=1; i<=n; i++)scanf("%lld", &a[i]);

    for(int i=1; i<=n; i++)
    {
        int v = a[i];
        for(int j=1; j*j<=v; j++)
        {
            if(v%j==0)
            {
                d[j]++;
                if(j!=(v/j))d[v/j]++;
            }
        }
    }
    ll ans = c[n][3];
    for(int i=2; i<=1000000; i++)
    {
        ans += (mob[i]*c[d[i]][3]);
    }
    printf("%lld\n", ans);
    return 0;
}

```