L16 : Number Theory 2

1-Tut : Sachin And Varun Problem

Send Feedback

Varun and you are talking about a movie that you have recently watched while Sachin is busy teaching Number Theory. Sadly, Sachin caught Varun talking to you and asked him to answer his question in order to save himself from punishment. The question says:

Given two weights of a and b units, in how many different ways you can achieve a weight of d units using only the given weights? Any of the given weights can be used any number of times (including 0 number of times).

Since Varun is not able to answer the question, he asked you to help him otherwise he will get punished.

Note: Two ways are considered different if either the number of times a is used or a number of times b is used is different in the two ways.

Input Format:

The first line of input consists of an integer
T denoting the number of test cases.
Each test case consists of only one line containing three space-separated integers a, b and d.

Output Format:

For each test case, print the answer in a separate line.

Constraints:

$1 \leq T \leq 10^5$

$1 \leq a < b \leq 10^9$

$0 \leq d \leq 10^9$

Sample Input 1

```
4
2 3 7
4 10 6
6 14 0
2 3 6
```

Sample Output 1

```
1
0
1
2
```

Explanation

Test case 1: 7 can only be achieved by using 2 two times and 3 one time.

Test case 2: 6 can't be achieved by using 4 and 10. So, 0 ways are there.

```
#include<bits/stdc++.h>
using namespace std;
```

```
typedef long long int ll;
class triplet
{
public:
    ll x, y, gcd;
};
triplet extended_euclid(ll a, ll b)
{
    if (b == 0)
    {
        triplet ans;
        ans.x = 1;
        ans.y = b;
        ans.gcd = a;
        return ans;
    }
    triplet small_ans = extended_euclid(b, a % b);
    triplet ans;
    ans.gcd = small_ans.gcd;
    ans.x = small_ans.y;
    ans.y = small_ans.x - small_ans.y * (a / b);
    return ans;
}
ll Multiplicative_Modulo_Inverse(ll a, ll m)
{
    ll ans=extended_euclid(a, m).x;
    return (ans%m+m)%m;
}
ll count_ways(ll a, int b, ll d)
{
    ll y1=((d%a)*Multiplicative_Modulo_Inverse(b, a))%a;
    ll first_value=d/b;
    if((d/b)<y1)
    {
        return 0;
    }
    ll n=(first_value-y1)/a;
    return n+1;
}
int main()
{
    ll t;
    cin>>t;
    while(t--)
    {
        ll a, b, d;
        cin>>a>>b>>d;
        ll g=__gcd(a, b);
        if(d%g!=0)
        {
            cout<<0<<endl;
            continue;
        }
        a/=g;
        b/=g;
        d/=g;
        cout<<count_ways(a, b, d)<<endl;
    }
}
```

2-Tut : Advanced GCD

Send Feedback

Varun explained its friend Sanchit the algorithm of Euclides to calculate the GCD of two numbers. Then Sanchit implements the algorithm

```
int gcd(int a, int b)
{
  if (b==0)
  return a;
  else
  return gcd(b,a%b);
}
```

and challenges to Varun to calculate gcd of two integers, one is a little integer and another integer can have 10^4 digits.

Your task is to help Varun an efficient code for the challenge of Sanchit.

Input Format

The first line of input will contain T(number of the test case), each test case follows as.
Each test case consists of two number A and B.

Output Format:

Print for each pair (A,B) in the input one integer representing the GCD of A and B.

Constraints:

1 <= T <= 100
1 <= A <= 4*10^5
1 <= |B| <= 10^4
where |B| is the length of the integer B

Sample Input:

```
2
2 6
10 11
```

Sample Output:

```
2

1
```

```
#include<bits/stdc++.h>
using namespace std;

int gcdmain(int a ,int b){
    if(b==0){
        return a;
    }
    return gcdmain(b,a%b);
}

int mod(string num, int a)
{

    int res = 0;

    for (int i = 0; i < num.length(); i++)
        res = (res * 10 + (int)num[i] - '0') % a;

    return res;
}
int gcd(int a,string b)
{

 int d=mod(b,a);
 int maxi=max(a,d);
 int mini=min(a,d);
 return gcdmain(maxi,mini);
}
int main(){

  int t;
    cin>>t;
    while(t--){
        int a;
        cin>>a;
        string b;
        cin>>b;
        cout<<gcd(a,b)<<endl;
    }
    return 0;
}
```

3-Tut : Divisors Of Factorial

Send Feedback

Given a number, find the total number of divisors of the factorial of the number.

Since the answer can be very large, print answer modulo 10^9+7.

Input Format:

The first line contains T, number of test cases.
T lines follow each containing the number N.

Output Format:

Print T lines of output each containing the answer.

Constraints

1 <= T <= 500
0 <= N <= 50000

Sample Input:

```
3
2
3
4
```

Sample Output:

```
2
4

8
```

```
#include<bits/stdc++.h>
using namespace std;
#define m 1000000007
void fact_divisor(int n){
    bool *prime=new bool[n+1];
    for(long long i=0;i<n+1;i++){
        prime[i]=true;
    }

    prime[0]=false;
    prime[1]=false;
    for(long long i=2;i<=sqrt(n);i++){
        if(prime[i]){
            for(long long j=i;j*i<=n;j++){
                prime[j*i]=false;
            }
        }
    }



    long long divisor=1;
    for(long long i=0;i<=n;i++){
        if(prime[i]){
            long long sum=0;
            for(long long j=1;pow(i,j)<=n;j++){
                sum+=n/pow(i,j);
            }
            divisor=(divisor%m*(sum+1)%m)%m;
        }
    }
    cout<<divisor<<endl;

}
int main(){

    // write your code here
     long long t;
    cin>>t;
    while(t--){
        long long n;
        cin>>n;
        fact_divisor(n);
    }
    return 0;
}
```

4-Ass : Find The Cube Free Number

Send Feedback

A cube free number is a number who's none of the divisor is a cube number (A cube number is a cube of a integer like 8 (2 * 2 * 2) , 27 (3 * 3 * 3)). So cube free numbers are 1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, 18 etc (we will consider 1 as cube free). 8, 16, 24, 27, 32 etc are not cube free number. So the position of 1 among the cube free numbers is 1, position of 2 is 2, 3 is 3 and position of 10 is 9. Given a positive number you have to say if its a cube free number and if yes then tell its position among cube free numbers.

Note: we will consider 1 as the 1st cube free number

Input Format:

First line of the test case will be the number of test case T

Each test case contain an integer N

Output Format:

For each test case print the position of N in cube free numbers and if its not a cube free number print "Not Cube Free" in a newline.

Constraints:

```
1 <= T <= 10^5
1 <= N <= 10^6
```

Sample Input:

```
10
1
2
3
4
5
6
7
8
9
10
```

Sample Output:

```
1
2
3
4
5
6
7
Not Cube Free
8

9
```

```
#include<bits/stdc++.h>
using namespace std;
void cubic(int arr[]){
    for(int i=0;i<1000001;i++){
        arr[i]=0;
    }
    arr[1]=1;
    for(int i=2;i<=100;i++){
        int cube=i*i*i;
        for(int j=1;cube*j<1000001;j++){
            arr[cube*j]=-1;

        }
    }


    int k=1;
    for(int i=1;i<1000001;i++){
        if(arr[i]!=-1){
            arr[i]=k;
            k++;
        }
    }
}
int main(){

    // write your code here
    int n;
    cin>>n;
    int arr[1000001];
    cubic(arr);
    for(int i=1;i<=n;i++){
        int val;
        cin>>val;
        if(arr[val]!=-1){
            cout<<arr[val]<<endl;
        }
        else{
            cout<<"Not Cube Free"<<endl;
        }
    }

    return 0;
}
```

5-Ass : Number Of Factors

Send Feedback

A number is called n-factorful if it has exactly n distinct prime factors. Given positive integers a, b, and n, your task is to find the number of integers between a and b, inclusive, that are n-factorful. We consider 1 to be 0-factorful.

Input Format:

First line of input will contain T(number of test cases), each test case follows as.
Each test cases consists of a single line containing integers a, b, and n as described above.

Output Format:

Output for each test case one line containing the number of n-factorful integers in [a, b].

Constraints:

1 <= T <= 10000
1 ≤ a ≤ b ≤ 10^6
0 <= b - a <= 100

0 ≤ n ≤ 10

Sample Input

```
4
1 3 1
1 10 2
1 10 3
1 100 3
```

Sample Output

```
2
2
0

8
```

```
#include<bits/stdc++.h>
using namespace std;
void makeSieve(int *isprime, int n)
{
    for (int i = 0; i < n + 1; i++)
    {
        isprime[i] = 0;
    }
    for (int i = 2; i <= n; i++)
    {
        if (isprime[i] == 0)
            for (int j = 1; j * i <= n; j++)
            {
                isprime[j * i] += 1;
            }
    }
}
int main(){

    // write your code here
     int t;
    cin >> t;
    int *arr = new int[1000001];
    int **table=new int* [11];
    for(int i=0; i<11; i++)
    {
        table[i]=new int[10000001];
    }
    makeSieve(arr, 1000001);
    for(int i=0; i<11; i++)
    {
        table[i][0]=0;
        table[i][1]=0;
        for(int j=2; j<1000001; j++)
        {
            if(i==arr[j])
            {
                table[i][j]=table[i][j-1]+1;
            }
            else
            {
                table[i][j]=table[i][j-1];
            }
        }
    }
    for (int test_case = 1; test_case <= t; test_case++)
    {
        int a, b, n;
        cin>>a>>b>>n;
        if(a==1&&n==0)
        {
            cout<<1<<endl;
            continue;
        }
        cout<<table[n][b]-table[n][a-1]<<endl;
    }
    return 0;
}
```

6-Ass : Find the good sets!

Send Feedback

You are given array a consisting of n distinct integers. A set s of numbers is called good if you can rearrange the elements in such a way that each element divides the next element in the order, i.e. 'si' divides 'si + 1', such that i < |s|, where |s| denotes size of set |s|.

Find out number of distinct good sets that you can create using the values of the array. You can use one value only once in a particular set; ie. a set cannot have duplicate values. Two sets are said to be different from each other if there exists an element in one set, which does not exist in the other.

As the answer could be large, print your answer modulo 10^9 + 7.

Input Format:

First line of the input contains an integer T denoting the number of test cases. T test cases follow.
First line of each test case contains an integer n denoting number of elements in array a.
Next line contains n space separated integers denoting the elements of array a.

Output Format:

For each test case, output a single line containing the corresponding answer.

Constraints

1 ≤ T ≤ 10

1 ≤ n ≤ 10^5

1 ≤ ai ≤ 10^5

All the elements of array a are distinct.

Input

```
2
2
1 2
3
6 2 3
```

Output:

```
3
5
```

Explanation

Test case 1. There are three sets which are good {1}, {2}, {1, 2}.

Test case 2. There are five sets which are good {6}, {2}, {3}, {6 2}, {6, 3}.

```
#include<bits/stdc++.h>
using namespace std;
#define m 1000000007
#define ll long long int
ll size_of_sieve = 750001;
void Seive(ll *arr, ll n)
{
    ll *sieve = new ll[size_of_sieve];
    for (ll i = 0; i < size_of_sieve; i++)
    {
        sieve[i] = 0;
    }
    for (ll i = 0; i < n; i++)
    {
        sieve[arr[i]] = 1;
    }
    for (ll i = 1; i < size_of_sieve; i++)
    {
        ll current_element = i;
        if (sieve[current_element] != 0)
            for (ll j = 2; j * current_element < size_of_sieve; j++)
            {
                if (sieve[j * current_element] != 0)
                {
                    sieve[j * current_element] = (sieve[j * current_element] % m + sieve[i] % m)
% m;
```

```
                }
            }
    }
    ll total_sum = 0;
    for (ll i = 0; i < size_of_sieve; i++)
    {
        total_sum = (total_sum % m + sieve[i] % m) % m;
    }
    cout << total_sum << endl;
    delete[] sieve;
}
int main(){

    // write your code here
    ll t;
    cin >> t;
    ll *arr = new ll[750001];
    while (t--)
    {
        ll n;
        cin >> n;
        for (ll i = 0; i < n; i++)
        {
            cin >> arr[i];
        }
        Seive(arr, n);
    }
    delete[] arr;
    return 0;
}
```

7-Ass : Card Game

Send Feedback

Vova again tries to play some computer card game.

The rules of deck creation in this game are simple. Vova is given an existing deck of n cards and a magic number k. The order of the cards in the deck is fixed. Each card has a number written on it; number ai is written on the i-th card in the deck.

After receiving the deck and the magic number, Vova removes x (possibly x = 0) cards from the top of the deck, y (possibly y = 0) cards from the bottom of the deck, and the rest of the deck is his new deck (Vova has to leave at least one card in the deck after removing cards). So Vova's new deck actually contains cards x + 1, x + 2, ... n - y - 1, n - y from the original deck.

Vova's new deck is considered valid iff the product of all numbers written on the cards in his new deck is divisible by k. So Vova received a deck (possibly not a valid one) and a number k, and now he wonders, how many ways are there to choose x and y so the deck he will get after removing x cards from the top and y cards from the bottom is valid?

Input
The first line contains two integers n and k (1 ≤ n ≤ 100 000, 1 ≤ k ≤ 10^9).
The second line contains n integers a1, a2, ..., an (1 ≤ ai ≤ 10^9) — the numbers written on the cards.
Output
Print the number of ways to choose x and y so the resulting deck is valid.
Sample Input 1
```
3 4
6 2 8
```
Sample Output 1
```
4
```
Sample Input 2
```
3 6
9 1 14
```
Sample Output 2
```
1
```

```
#include<iostream>
using namespace std;
#include<utility>
#include<vector>
#define ll long long int
#define PII pair<int, int>
#define f first
#define s second
#define mk make_pair
#define pb push_back
int main(){
            // Write your code here
     int n, i, j;
    ll k, temp_for_k, ans = 0;
    cin >> n >> k;
    temp_for_k = k;
    vector<ll> a(n);
    for (i = 0; i < n; i++)
    {
        cin >> a[i];
    }
    vector<PII> vp; //store prime factors of k with respective powers
    for (i = 2; i * i <= temp_for_k; i++)
    {
        if (temp_for_k % i == 0)
        {
            int count = 0;
            while (temp_for_k % i == 0)
            {
                temp_for_k /= i;
                count++;
            }
            vp.push_back(make_pair(i, count));
        }
    }
    if (temp_for_k != 1)
    {
        vp.push_back(make_pair(temp_for_k, 1));
    }
    //vq contains all prime factors of k with their respective powers 0
    vector<PII> vq = vp;
    for (i = 0; i < vq.size(); i++)
    {
        vq[i].s = 0;
    }
    //j is back pointer and i is front pointer
    //We are checking if j..i is divisible by k by adding prime factors of a[i] to vq
    //If it is divisible, then j...z for all z > i is also divisible by k
    //So we add n-i to answer
    //Then we remove j's contribution by subtracting it's prime factors from vq
    j = 0;
    for (i = 0; i < n; i++)
    {
        //add factors of a[i]
        for (int z = 0; z < vp.size(); z++)
        {
            if (a[i] % vp[z].f == 0)
            {
                temp_for_k = a[i];
                int cn = 0;
                while (temp_for_k % vp[z].f == 0)
                {
                    temp_for_k /= vp[z].f;
                    cn++;
                }
                vq[z].s += cn;
            }
        }
        while (j <= i)
        {
            int z;
            //check if divisible by k from j to i
            for (z = 0; z < vp.size(); z++)
            {
                if (vp[z].s > vq[z].s)
                    break;
            }
            if (z != vp.size())
                break;
            //if divisible update ans
            ans += n - i;
            //remove factors of a[j] from vq
            for (int z = 0; z < vp.size(); z++)
            {
                if (a[j] % vp[z].f == 0)
                {
                    temp_for_k = a[j];
                    int cn = 0;
                    while (temp_for_k % vp[z].f == 0)
                    {
                        temp_for_k /= vp[z].f;
                        cn++;
                    }
                    vq[z].s -= cn;
                }
            }
            j++;
        }
    }
    cout << ans;
    return 0;
}
```

8-Ass : Strange order

Send Feedback

Given an integer n . Initially you have permutation p of size n : p[i] = i . To build new array a from p following steps are done while permutation p is not empty:

Choose maximum existing element in p and define it as x ; Choose all such y in p that gcd (x , y) ≠ 1 ; Add all chosen numbers into array a in decreasing order and remove them from permutation. Your task is to find a after p became empty.

Note: gcd (a , b) is the greatest common divisor of integers a and b .

Input format

Input contains single integer n — the size of permutation p p.

Output format

Print n integers — array a .

Constraints:

1 <= N <= 10^5

Sample Input:

5

Sample Output:

5 4 2 3 1

Explanation

It's needed 4 iterations to create array a:

Add 5

Add 4 and 2

Add 3

Add 1

```
#include<iostream>
#include<vector>
#include<algorithm>
using namespace std;
#define MaxSize 2000001
int main()
{
    int *sieve = new int[MaxSize];
    for (int i = 0; i <= MaxSize; i++)
    {
        sieve[i] = i;
    }
    for (int i = 2; i * i <= MaxSize; i++)
    {
        for (int j = i * i; j <= MaxSize; j += i)
        {
            if (sieve[j] > i)
            {
                sieve[j] = i;
            }
        }
    }
    int n, k = 0;
    cin >> n;
    int *finalans = new int[n];
    bool *marked = new bool[n + 1];
    for (int i = 0; i <= n; i++)
```

```cpp
    {
        marked[i] = false;
    }
    for (int i = n; i >= 1; i--)
    {
        if (!marked[i])
        {
            int lpd = sieve[i];
            int x = i;
            vector<int> v;
            marked[i] = true;
            v.push_back(i);
            while (x != 1)
            {
                for (int j = i - lpd; j >= 1; j = j - lpd)
                {
                    if (!marked[j])
                    {
                        marked[j] = true;
                        v.push_back(j);
                    }
                }
                while (x % lpd == 0)
                {
                    x = x / lpd;
                }
                lpd = sieve[x];
            }
            sort(v.begin(), v.end(), greater<int>());
            for (int i = 0; i < v.size(); i++)
            {
                finalans[k] = v[i];
                k++;
            }
        }
    }
    finalans[n - 1] = 1;
    for (int i = 0; i < n; i++)
    {
        cout << finalans[i] << " ";
    }
    cout << endl;
}
```