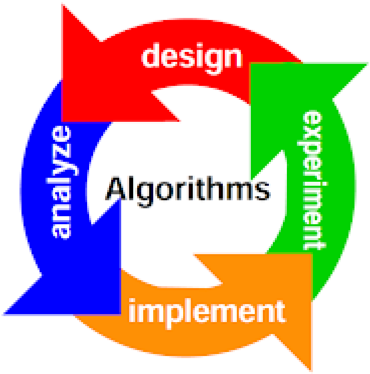

ডিভিসর এর সংখ্যা নির্নয় (Finding Number of Divisors)

সাধারনত আমরা কোন সংখ্যার কতগুলো ডিভিসর আছে তা নির্নয় এর জন্য লুপ চালায় কাউন্ট করি। কিন্তু n যদি 10^12 হয় তখন কি আমরা এত বার লুপ চালায় কাউন্ট করব?? না , কারন এর ফলে প্রোগ্রাম অনেক ধীরগতির হয়ে যাবে এবং অনলাইন জাজ এ time limit exceed আসবে। তাহলে উপায় কি??

সুত্রঃ ডিভিসর সংখ্যা নির্নয় এর জন্য প্রথমে n এর প্রাইম ফ্যাক্টর গুলো বের করে নিতে হবে। যে কোন সংখ্যা n হলে n কে আমরা প্রাইম সংখ্যার গুনফল আকারে লিখতে পারি। যেমনঃ n=১২ হলে,

12= 2 * 2 * 3

এখানে প্রাইম ফ্যাক্টরগুলো হলঃ 2, 2 , 3

এখন, $12= 2^{2} * 3^{1}$

তাহলে , Number Of Divisors=d

d= ((2 এর পাওয়ার)+1)*((3 এর পাওয়ার)+1)

=(2+1)*(1+1)

=6

সুতরাং , n= $P_1^{x1} * P_2^{x2} * P_3^{x3} *$ ....P_n^{xn} হলে

ডিভিসর সংখ্যা , d= (x1+1)*(x2+1)*(x3+1)*..................................(xn+1)

প্রাইম ফ্যাক্টর বের করার নিয়মঃ

প্রথমে 1 থেকে sqrt(10^12) পর্যন্ত সব প্রাইম নম্বর জেনারেট করে ফেলব সিভ sieve এলগরিদম দিয়ে।

এখানে sqrt পর্যন্ত নেয়া হয়েছে কারন 10^12 এর প্রাইম ফ্যাক্টর গুলো sqrt(10^12) এর মধ্যেই থাকবে যেমন ১২ এর প্রাইম ফ্যাক্টর sqrt(12)=3 এর মধ্যেই থাকে ।

এখন প্রাইম জেনারেট করার সময় অবশ্যই p এরেতে স্টোর করে রাখব। p এরের সাইজ বের করে ফেলব।

এখন প্রাইম ফ্যাক্টর বের করার জন্য চেক করতে হবে প্রাইম নম্বরগুলো দিয়ে n ভাগ যায় কি না। ভাগ গেলে আমরা প্রাইম ফ্যাক্টর কাউন্ট করব।

সর্বোচ্চ এরের সাইজ k পর্যন্ত লুপ চলবে কিন্তু আমরা তার আগেই লুপ শেষ করতে পারি কারন 12 এর ক্ষেত্রে এরের মান sqrt(12) কে অতিক্রম করবে না।

এরপর লুপ এ n%p[i]==0 কি না চেক করে বের করব ।

```
while(n%p[i]==0)

{

    count++;

}
```

যদি n=0 অথবা ১ হয়ে যায় তাহলে break করব লুপ ।

যেমনঃ

726= 2*3*11*11

n=(726/2)=363

2 এর জন্য count=1

n=(363/3)=121

3 এর জন্য count=1

n=(121/11)=11

n=(11/11)=1

11 এর জন্য count=2

n=1 তাই ব্রেক হবে ।

সুতরাং d=(1+1)*(1+1)*(2+1)=12

Code

```
#include <bits/stdc++.h>
using namespace std;
#define n 1000005
bool a[n];
long long int k=1;
long long int twin[n];
void sieve()
{
    long long int i,j;
    a[0]=a[1]=1;
    for(i=4;i<=n;i=i+2)
    {
        a[i]=1;
    }
    for(i=3;i<=sqrt(n);i=i+2)
    {
        for(j=i*i;j<=n;j=j+2*i)
        {
            a[j]=1;        //3*3, 3*5,3*7.....
        }
    }
    for(i=2;i<=n;i++)
    {
        if(a[i]==0)
        {
           twin[k]=i;
           k++;
        }
```

```
    }

}
int main()
{
    long long int m,g,c,r,s,t,l,h;
    cin>>t;
    sieve();
    for(l=1;l<=t;l++)
    {
        cin>>m;
        //in>>m;
        r=1;
        h=0;
        for(g=1;g<=k && twin[g]<=sqrt(m);g++)
        {
            c=0;
            if(m%twin[g]==0)
            {

            while(m%twin[g]==0)
            {
                c++;
                m=m/twin[g];
                if(m==0 || m==1)
                {
                    break;
                }

             }
              s=c+1;
              r=r*s;
            }



        }
        if(m!=1)
        {
            r=r*2;    //for prime numbers there are only 2 divisors.
        }

        printf("%lld\n",r);


    }

    return 0;
}
```

UVA 294 Divisors এবং **LightOj 1028 Trailing Zero (1)** এর প্রব্লেম এই নিয়মে সল্ভ করা যায়। কোডিং এ প্রব্লেম হলে UVA 294 Divisors এবং LightOj 1028 Trailing With Zeroes(I) এর সলুশন দেখতে পারো ।