浙江财經大學
Zhejiang University Of Finance & Economics

欧拉函数

信智学院　陈琰宏

欧拉函数

对于正整数n,欧拉函数是小于或等于n的正整数中与n互质的数的数目，记作$\varphi(n)$. $n = p_1^{a_1} * p_2^{a_2} * ... * p_x^{a_x}$

$\varphi(1) = 1$

$$\phi(n) = n \times \frac{p_1 - 1}{p_1} \times \frac{p_2 - 1}{p_2} \times ... \times \frac{p_k - 1}{p_k}$$

例如，对于 $n = 12$，其质因数分解为 $12 = 2^2 \cdot 3^1$。

应用欧拉函数的计算公式，我们有：

$$\varphi(12) = 12\left(\frac{2-1}{2}\right)\left(\frac{3-1}{3}\right) = 12 \times \frac{1}{2} \times \frac{2}{3} = 4$$

小于或等于 12 的正整数中与 12 互质的数有 $1, 5, 7, 11$，共 4 个。

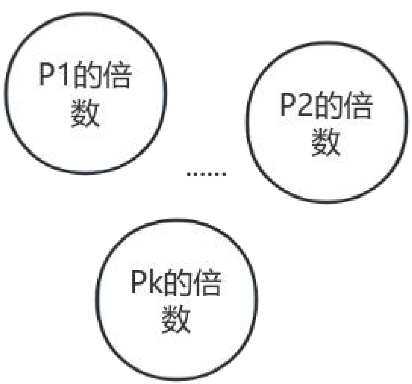
欧拉函数的证明1:

利用容斥原理来证明. 设sum为1~n中与n互质的数的个数基本思路是去掉1~n中所有p1, p2, …, pk的倍数

欧拉函数的具体公式:

$$\phi(n) = n \times \frac{p_1 - 1}{p_1} \times \frac{p_2 - 1}{p_2} \times \ldots \times \frac{p_k - 1}{p_k}$$

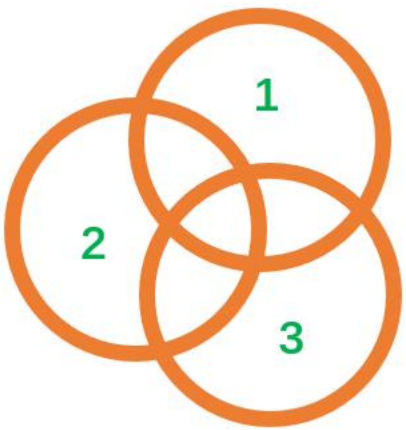
①当p1, p2, …, pk的倍数集合没有交集时



$$sum = n - \frac{n}{p_1} - \frac{n}{p_2} - \ldots - \frac{n}{p_k}$$

欧拉函数的证明1:

②当p1, p2, …, pk中的任意两个数的倍数集合拥有交集时
欧拉函数的具体公式:



这时在第①步时,会多减一次$p_i \times p_j$,所以需要加上一次$p_i \times p_j$

因此有

$$sum = n - \frac{n}{p_1} - \frac{n}{p_2} - \ldots - \frac{n}{p_k} + \frac{n}{p_1 \times p_2} + \frac{n}{p_1 \times p_3} + \ldots$$

依次类推有③,④,. . .

$$sum = n - \frac{n}{P1} - \frac{n}{P2} - \ldots - \frac{n}{Pk} + \frac{n}{P1*p2} + \frac{n}{P1*p3} + \ldots + \frac{n}{p_{k-1}*pk}$$
$$- \frac{n}{P1*p2*p3} - \frac{n}{P2*p3*p4} - \ldots - \frac{n}{p_{k-2}*p_{k-1}*p_k} + \ldots$$

最后将n提出来,就可出现

$$\phi(n) = n \times \frac{p_1 - 1}{p_1} \times \frac{p_2 - 1}{p_2} \times \ldots \times \frac{p_k - 1}{p_k}$$

欧拉函数的证明2:

对于正整数n,欧拉函数是小于或等于n的正整数中与n互质的数的数目，记作$\varphi(n)$.

$\varphi(1)=1 \qquad n=p_1^{a_1}*p_2^{a_2}*\ldots*p_x^{a_x}$

$$\phi(n)=n\times\frac{p_1-1}{p_1}\times\frac{p_2-1}{p_2}\times\ldots\times\frac{p_k-1}{p_k}$$

首先， 欧拉函数是一个积性函数，当m,n互质时，

$\varphi(mn)=\varphi(m)*\varphi(n)$

根据唯一分解定理知 $n=p_1^{a_1}*p_2^{a_2}*\ldots*p_x^{a_x}$

因此 $\varphi(n)=\varphi(p_1^{a_1})*\ldots*\varphi(p_x^{a_x})$

对于任意一项 $\varphi(p_s^{a_s})=p_s^{a_s}-p_s^{(a_s-1)}$

欧拉函数的证明2:

对于任意一项 $\varphi(p_s^{a_s}) = p_s^{a_s} - p_s^{(a_s-1)}$

从定义出发 $\varphi(p_s^{a_s})$等于小于或等于$p_s^{a_s}$的正整数中与$p_s^{a_s}$互质的数的数目

从1到$p_s^{a_s}$中共有$p_s^{a_s}$个数字

其中与$p_s^{a_s}$不互质的有$p_s, 2p_s, \ldots, {p_s}^{a_s-1} * p_s$,共${p_s}^{a_s-1}$项

所以 $\varphi(p_s^{a_s})$ = $p_s^{a_s}$ - ${p_s}^{a_s-1} = p_s^{a_s} * (1 - \frac{1}{p_s})$

欧拉函数的证明2:

因此

$$
\begin{aligned}
\varphi(n) &= \varphi(p_1^{a_1}) * \ldots * \varphi(p_x^{a_x}) \\
&= (p_1^{a_1} - p_1{}^{a_1-1}) * \ldots * (p_x^{a_x} - p_x{}^{a_x-1}) \\
&= p_1^{a_1} * (1 - \frac{1}{p_1}) * p_2^{a_2} * (1 - \frac{1}{p_2}) * \ldots * p_x^{a_x} * (1 - \frac{1}{p_x}) \\
&= p_1^{a_1} * p_2^{a_2} * \ldots * p_x^{a_x} * (1 - \frac{1}{p_1}) * (1 - \frac{1}{p_2}) * \ldots * (1 - \frac{1}{p_x}) \\
&= n * \prod_{i=1}^{x} (1 - \frac{1}{p_i})
\end{aligned}
$$

1 [6380] 欧拉函数

给定 n 个正整数 a_i，请你求出每个数的欧拉函数。

欧拉函数的定义

> $1 \sim N$ 中与 N 互质的数的个数被称为欧拉函数，记为 $\phi(N)$。
> 若在算数基本定理中，$N = p_1^{a_1} p_2^{a_2} \ldots p_m^{a_m}$，则：
> $\phi(N) = N \times \frac{p_1-1}{p_1} \times \frac{p_2-1}{p_2} \times \ldots \times \frac{p_m-1}{p_m}$

输入格式

第一行包含整数 n。

接下来 n 行，每行包含一个正整数 a_i。

输出格式

输出共 n 行，每行输出一个正整数 a_i 的欧拉函数。

数据范围

$1 \leq n \leq 100$,
$1 \leq a_i \leq 2 \times 10^9$

代码

```cpp
#include <iostream>
using namespace std;
int main()
{
    int T;
    cin >> T;
    while(T--)
    {
        int n;
        cin >> n;
        // 公式最前面的n
        int res = n;
        //求质因子
        for(int i = 2; i <= n / i; i++)
        {
            if(n % i == 0)//找到质因子
            {
                // (p - 1) / p
                // 先除后乘
                res = res / i * (i - 1) ;
                // 对 n 进行约分
                while(n % i == 0) n = n / i;
            }
        }
        // 如果有剩余，则剩余是个质因子
        if( n > 1) res = res /n * (n - 1) ;
        cout << res << endl;
    }
    return 0;
}
```

2 [6381] 筛法求欧拉函数

给定一个正整数 n，求 1~ n 中每个数的欧拉函数之和共一行，包含一个整数 n。

输出共一行，包含一个整数，表示 1~ n 中每个数的欧拉函数之和。$1 \leqslant n \leqslant 10^6$

输入	输出
6	12

分析

如果 `x` 是一个素数 `p`

$$x = p \quad \varphi(x) = x * (1 - \frac{1}{p}) = p * (\frac{p-1}{p}) = p - 1$$

```
for (int i = 2; i <= x; i ++)
{
    if(!st[i])//st[]标记某个数是不是质数
    {  //cnt记录已经找到的质数数量
        p[cnt ++] = i; //p[]存放已经找到的质数
        phi[i] = i - 1;//// 欧拉函数 Φ (x)
    }
}
```

分析

$$\phi(n) = n \times \frac{p_1 - 1}{p_1} \times \frac{p_2 - 1}{p_2} \times \ldots \times \frac{p_k - 1}{p_k}$$

如果 x 不是素数，假设 $x = p[j] * i$, 且 $i\%p[j] = 0$，根据欧拉函数

可以推出：$p[j] * i$ 的所有质因子与 i的质因子是完全相同的。
根据欧拉函数的定义，欧拉函数值只与质因子有关，跟其的指数无关。
如此，$p[j] * i$ 与 i 的 $\frac{p1-1}{P1} * \frac{p2-1}{P2} * \ldots * \frac{pk-1}{Pk}$ 部分相同，则：

$$\varphi(i * p[j]) = (i * p[j]) * \frac{p_1 - 1}{P_1} * \frac{p_2 - 1}{P_2} * \ldots * \frac{p_k - 1}{P_k} = p[j] * \varphi(i)$$

```
for (int j = 0; p[j] * i <= x; j ++)
{
    st[p[j] * i] = 1;// 标记为不是质数
    if (i % p[j] == 0)//保证是最小质数的
    {  //计算 i * p[j] 的 欧拉函数
        phi[i * p[j]] = p[j] * phi[i];
        break;
    }
    ...  ...
}
```

分析

$$\phi(n) = n \times \frac{p_1 - 1}{p_1} \times \frac{p_2 - 1}{p_2} \times \ldots \times \frac{p_k - 1}{p_k}$$

如果 x 不是素数 ，假设 $x = p[j] * i$, 且 $i\%p[j]! = 0$。根据欧拉函数

因为$i\%p[j]! = 0$ 且 $p[j]$是质数，所以 i与 $p[j]$ 互质，根据欧拉函数的积性 性质，可以推出：

$$\varphi(i * p[j]) = \varphi（p[j]） * \varphi(i) = （p[j] - 1) * \varphi(i)$$

```
for (int j = 0; p[j] * i <= x; j ++)
{
    ... ...
     phi[i * p[j]] = (p[j] - 1) * phi[i];
}
```

代码

```
const int N = 1000010;
int n;
int p[N];//筛选质数 p[]存放已经找到的质数
int st[N];//st[]标记某个数是不是质数,
int cnt;// cnt记录已经找到的质数数量
int phi[N]; // 欧拉函数 Φ (x)
long long res;

void ola_primes(int x)
{

int main()
{
    cin >> n;
    phi[1] = 1;
    ola_primes(n);
    for (int i = 1; i <= n; i ++) res += phi[i];
    cout << res;

    return 0;
}
```

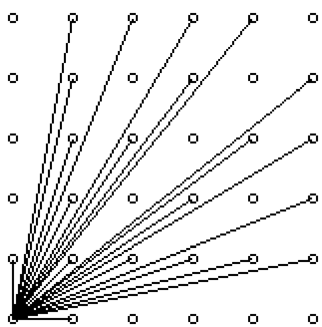
```
void ola_primes(int x)
{
    for (int i = 2; i <= x; i ++)
    {
        if(!st[i])//是质数
        {
            p[cnt ++] = i;
            phi[i] = i - 1;//// 欧拉函数 Φ (x)
        }
        //线性筛       筛选的数不能越界
        for (int j = 0; p[j] * i <= x; j ++)
        {
            st[p[j] * i] = 1;// 标记为不是质数
            if (i % p[j] == 0)//保证是最小质数筛
            {
                phi[i * p[j]] = p[j] * phi[i];
                break;
            }
            // i % p[j] != 0 表示 i,p[j]互质, 符合 积性函数
            phi[i * p[j]] = (p[j] - 1) * phi[i];
        }
    }
}
```

[1940] 可见的点

在一个平面直角坐标系的第一象限内，如果一个点（x, y）与原点（0, 0）的连线中没有通过其他任何点，则称该点在原点处是可见的。

例如，点（4, 2）就是不可见的，因为它与原点的连线会通过点（2, 1）。

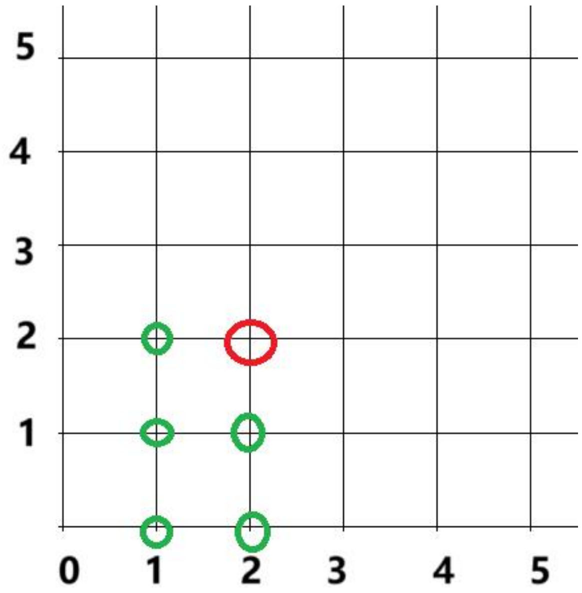
部分可见点与原点的连线如下图所示：

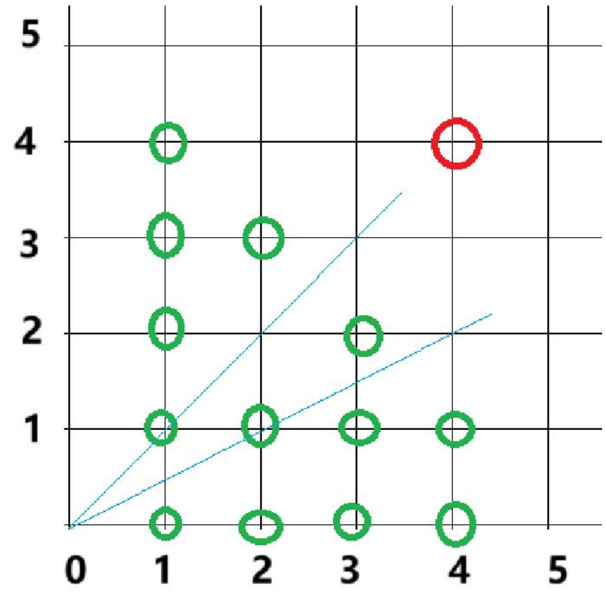


编写一个程序，计算给定整数 N 的情况下，满足 0≤x，y≤N的可见点（x，y）的数量（可见点不包括原点）。

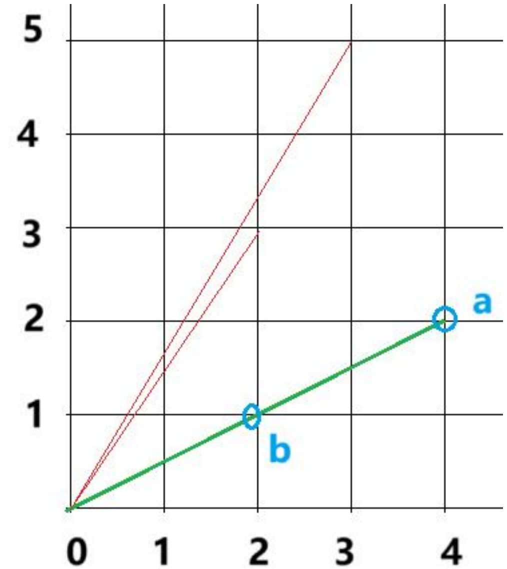
[1940] 可见的点

输入	输出
4 2 4 5 231	 1 2 5//　　测试数据的编号（从 1 开始） 2 4 13//　该组测试数据对应的 N 以及 3 5 21//　可见点的数量。 4 231 32549





分析



深入考虑一下：一个点a不可见就代表它和原点之间有一个点b。这时a. x/a. y一定等于b. x/b. y。

所以以一个点可见代表它的x坐标和y坐标互质。这也是欧拉函数的引入点。

phi[i]就可以代表x为i，y小于i的可见点数量

同时也可以代表y为i，x小于i的可见点数量。

所以把1到N之间所有数的欧拉函数值加起来，就是以中间线（射线(0,0)(1,1)）为分界，其中一侧的可见点数目。最终结果为 2*phi[i]+1

代码

```
int primes[N], cnt;
bool st[N];
int phi[N];

void init(int n)
{
    phi[1]=1;
    for(int i = 2;i<=n;i++)
    {
        if(!st[i])
        {
            primes[cnt++] = i;
            phi[i] = i-1;
        }
        for(int j=0;primes[j]*i<=n;j++)
        {
            st[i*primes[j]]=true;
            if(i%primes[j]==0)
            {
                phi[i*primes[j]] = phi[i]*primes[j];
                break;
            }
            //情况2
            phi[i*primes[j]] = phi[i]*(primes[j]-1);
        }
    }
}
```

```
int main()
{
    init(N);//预处理求出所有小于N的欧拉函数值
    int n,m;
    cin >> m;
    for(int T=1;T<=m;T++)
    {
        cin >> n;
        int res = 1;//2倍+1，初始为1
        for(int i = 1;i<=n;i++) res+=phi[i]*2;
        cout << T << ' ' << n << ' ' << res << endl;
    }
    return 0;
}
```

[1605] 最大公约数

给定整数 N，求 $1 \leqslant x, y \leqslant N$ 且 GCD(x, y) 为素数的数对 (x, y)有多少对。

GCD(x, y) 即求 x，y的最大公约数。

分析

不同于题[1940 可见的点],不是求互质对的数,而是求x,y最大公约数gcd(x,y)是素数的对数

$$gcd(x,y)=p \iff gcd(\frac{x}{p},\frac{y}{p})=1$$

$$\text{统计}gcd(x,y)=p\in[0,N]\text{对数} \iff \text{统计}gcd(\frac{x}{p},\frac{y}{p})=1\in[0,\frac{N}{p}]\text{对数}$$

$$\iff x',y'\in[0,\frac{N}{p}]\text{互质数的对数}$$

问题转化为题[1940],只不过N变成了N/p

```
typedef long long LL;
const int N = 1e7 + 10;
int primes[N], cnt;
bool st[N];int phi[N];LL s[N];

void init(int n)
{
    for(int i=2;i<=n;i++)
    {
        if(!st[i])
        {
            primes[cnt++]=i;
            phi[i] = i-1;
        }
        for(int j =0;primes[j]*i<=n;j++)
        {
            st[primes[j]*i]=true;
            if(i%primes[j]==0)
            {
                phi[i*primes[j]] = phi[i]*primes[j];
                break;
            }
            phi[i*primes[j]] = phi[i]*(primes[j]-1);
        }
    }
    for(int i =1;i<=n;i++)s[i]=s[i-1]+phi[i];
}
```

```
int main()
{
    int n;
    cin >> n;
    init(n);
    LL res = 0;
    for(int i=0;i<cnt;i++)
    {
        int p=primes[i];
        res+=s[n/p]*2+1;
    }
    cout << res << endl;
    return 0;
}
```

[lq2018 9205]矩阵求和

经过重重笔试面试的考验，小明成功进入 Macrohard 公司工作。
今天小明的任务是填满这么一张表：
表有 n 行 n 列，行和列的编号都从 1 算起。
其中第 i 行第 j 个元素的值是 gcd(i, j) 的平方，
gcd 表示最大公约数，以下是这个表的前四行的前四列：

1	1	1	1
1	4	1	4
1	1	9	1
1	4	1	16

小明突然冒出一个奇怪的想法，他想知道这张表中所有元素的和。
由于表过于庞大，他希望借助计算机的力量。输入一行一个正整数 n （n<=1e7）， 表示表格大小。输出一行一个数，表示所有元素的和。由于答案比较大，请输出模 1000000007后的结果。