整除理论

如果把**组合数学**理解成对于**计数问题**的研究，那么**数论**就是对于**整除性问题**的研究，其中组合与数论是程序设计竞赛中的常见考点。

1. 整除的基本知识

**找寻环节：**

给定一个长度为n的字符串，求它的最小循环节长度，n<=10^5.例如输入“abbaabbaabba”，输出“4”.

**分析：**对于约数、倍数相关的题目，经常用到枚举约数和枚举倍数两个切入点。考虑循环节长度一定是n的一个约数，于是只需要枚举出n的所有约数，逐一检验即可，复杂度O(√n + Pn)，其中P是约数个数。用KMP算法可以做到线性复杂度。

1. 质数与合数

**欧拉线性筛法求素数，**时间复杂度:**O(n)**

#include <bits/stdc++.h>  
using namespace std;  
const int N=1e6+10;  
int primes[N];  
bool st[N];  
void get\_primes(){  
 int cnt=0;  
 //1不是质数也不是合数  
 for(int i=2;i<=N;i++){  
 if(!st[i])primes[++cnt]=i;//没有被筛去,说明是质数  
 for(int j=1;i\*primes[j]<=N;j++){  
 st[i\*primes[j]]=true;//筛去合数  
 if(i%primes[j]==0)break;//核心操作,保证了O(n)的复杂度  
 }  
 }  
}  
int main(){  
 //n以内的所有质数  
 int n,cnt;  
 cin>>n;  
 get\_primes();  
 for(int i=1;primes[i]<=n;i++){  
 cout<<primes[i]<<" ";  
 cnt++;  
 }  
 cout<<endl;  
 cout<<cnt<<endl;  
 return 0;  
}

1. 最大公约数与最小公倍数

a1和a2的最大公约数记作(a1,a2)

a1和a2的最小公倍数记作[a1,a2]

下面给出4个有关最大公约数和最小公倍数的常见性质与结论：

1. 对任意整数m，m(a1,…,ak)=(ma1,…,mak)，即整数同时成倍放大，最大公约数也放大相同倍数。该性质同样适用于最小公倍数情况。
2. 对任意整数x，(a1,a2)=(a1,a2+a1x)，即一个整数加上另一个整数的任意倍数，它们的最大公约数不变。这里要注意的是，(a,0)=a.该性质不适用于最小公倍数情况。
3. (a1,a2,a3,…,ak)=((a1,a2),a3,…,ak),以及推论(a1,a2,a3,…,ak+r)=((a1,…,ak),(ak+1,…,ak+r))。这是计算多元最大公约数的主要手段。更深刻一点，这个性质说明了最大公约数运算具有某种“结合律”。该性质同样适用于最小公倍数情况。
4. [a1,a2](a1,a2)=a1a2，最大公约数\*最小公倍数=原来两个数的乘积。

性质2可以给出一个高效的求两数最大公约数的算法：每次让较大的数对较小数取模，可以缩小问题规模而保持最大公约数不变，然后重复(递归)这个步骤。递归边界使某数变成了0，而此时另一个数即为所求答案，于是得到如下代码：

int gcd(int x,int y){  
 if(y==0)return x;//递归边界  
 else return gcd(y,x%y);  
}

这种利用两数相除(取模)求最大公约数的方法叫作**辗转相除法**或**Euclid算法**。

经过压行后，可以得到单行辗转相除法，代码如下:

int gcd(int x,int y){return y?gcd(y,x%y):x;}//greatest common divisor最大公约数

利用性质4，用两数之积除以它们的最大公约数，代码如下：

int lcm(int x,int y){return x/gcd(x,y)\*y;}//Least Common Multiple最小公倍数,要注意乘除的先后顺序

因为x一定是gcd(x,y)的倍数，所以这样先除后乘没有问题，而且可以避免可能的溢出事件。