Map遍历

#include<bits/stdc++.h>

using namespace std;

int main(){

map<int,int>Map;

int a[50000];

int i = 0;

while(cin>>a[i++]){

if(Map.count(a[i-1]))

Map[a[i-1]]++;

else Map[a[i-1]] = 0;

};

int ma = 0;

for(int j = 0;j<i;j++){

ma = max(ma,Map[a[j]]);

}

for(map<int,int>::iterator iter = Map.begin();iter != Map.end();++iter){

//cout << iter->first << " " << iter->second << endl;

ma = max(ma,Map[iter->first]);

}//Map.iterator a = Map.begin();

for(map<int,int>::iterator iter = Map.begin();iter != Map.end();++iter){

//cout << iter->first << " " << iter->second << endl;

if(ma == iter->second){

cout<<iter->first<<endl;

}

}

}

遍历set

#include<bits/stdc++.h>

using namespace std;

int main(){

set<int > S;

int t;

while(cin>>t){

S.insert(t);

S.const\_iterator

}

for(set<int >::iterator iter = S.begin();iter != S.end();iter++){

cout<<\*iter<<endl;

}

}

/\*不确定整数输入\*/

#include<bits/stdc++.h>

using namespace std;

map<int,bool> maps;

int main(){

int c[100000];

char a ;

while(cin.get()!= EOF){

cin.unget();

int i = 0;

maps.clear();

while((a = cin.get())!='\n')

{

cin.unget();

cin>>c[i++];

maps[c[i-1]] = 1;

}

for(int j = 0;j<i;j++){

if(maps[c[j]] == 1){

if(j != 0) cout<<" ";

cout<<c[j];

maps[c[j]] = 0;

}

}

cout<<endl;

}

}

MOBIWUSI

/\*

公因数为质数问题gcd（a,b） = prime

\*/

#include <iostream>

#include <bitset>

using namespace std;

typedef long long LL;

const int N = 10000010;

bitset<N> prime;

LL phi[N], f[N];

int p[N], k;

void isprime() { //求素数组

k = 0;

prime.set();

for(int i=2; i<N; i++) {

if(prime[i]) {

p[k++] = i;

for(int j=i+i; j<N; j+=i)

prime[j] = false;

}

}

}

void Init() { //求(i)

int i,j;

for( i=1; i<N; i++) phi[i] = i;

for( i=2; i<N; i+=2) phi[i] >>= 1;

for( i=3; i<N; i+=2) {

if(phi[i] == i) {

for( j=i; j<N; j+=i)

phi[j] = phi[j] - phi[j] / i;

}

}

f[1] = 0;

for( i=2;i<N;i++)

f[i] = f[i-1] + (phi[i]<<1);

}

LL Solve(int n){

LL ans = 0;

for(int i=0; i<k&&p[i]<=n; i++)

ans += 1 + f[n/p[i]];

return ans;

}

int main() {

Init();

isprime();

int n;

scanf("%d",&n);

printf("%I64d\n",Solve(n));

return 0;

}

GCD

/\*给定a,b,c,d,k五个数，其中a=c=1固定的，让你从[a,b]中找出x，

[c,d]中找出y，使gcd(x,y) = k，注意gcd(x,y) 与gcd(y,x)归为同一种，

问一共能找到多少组x，y;\*/

/\*这题求[1,n],[1,m]gcd为k的对数。而且没有顺序。

转化之后就是[1,n/k],[1,m/k]之间互质的数的个数。

用莫比乌斯反演就很容易求了。

为了去除重复的，去掉一部分就好了；

这题求的时候还可以分段进行优化的。

具体看我的下一篇博客吧！\*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Author :kuangbin

Created Time :2013/8/21 19:32:35

File Name :F:\2013ACM练习\专题学习\数学\莫比乌斯反演\HDU1695GCD.cpp

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

#include <stdio.h>

#include <string.h>

#include <iostream>

#include <algorithm>

#include <vector>

#include <queue>

#include <set>

#include <map>

#include <string>

#include <math.h>

#include <stdlib.h>

#include <time.h>

using namespace std;

const int MAXN = 1000000;

bool check[MAXN+10];

int prime[MAXN+10];

int mu[MAXN+10];

void Moblus()

{

memset(check,false,sizeof(check));

mu[1] = 1;

int tot = 0;

for(int i = 2; i <= MAXN; i++)

{

if( !check[i] )

{

prime[tot++] = i;

mu[i] = -1;

}

for(int j = 0; j < tot; j++)

{

if(i \* prime[j] > MAXN) break;

check[i \* prime[j]] = true;

if( i % prime[j] == 0)

{

mu[i \* prime[j]] = 0;

break;

}

else

{

mu[i \* prime[j]] = -mu[i];

}

}

}

}

int main()

{

//freopen("in.txt","r",stdin);

//freopen("out.txt","w",stdout);

int T;

int a,b,c,d,k;

Moblus();

scanf("%d",&T);

int iCase = 0;

while(T--)

{

iCase++;

scanf("%d%d%d%d%d",&a,&b,&c,&d,&k);

if(k == 0)

{

printf("Case %d: 0\n",iCase);

continue;

}

b /= k;

d /= k;

if(b > d)swap(b,d);

long long ans1 = 0;

for(int i = 1; i <= b;i++)

ans1 += (long long)mu[i]\*(b/i)\*(d/i);

long long ans2 = 0;

for(int i = 1;i <= b;i++)

ans2 += (long long)mu[i]\*(b/i)\*(b/i);

ans1 -= ans2/2;

printf("Case %d: %I64d\n",iCase,ans1);

}

return 0;

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

这题就是求gcd(a,b,c) = 1 a,b,c <=N 的对数。

用莫比乌斯反演可以求解。

设g(n)为gcd(x,y,z)=n的个数，f(n)为n | g(i\*n)的个数，那么有f(n)=sigma(n|d,g(d))，那么g(n)=sigma(n|d, mu(d/n)\*f(d))，我们要求g(1)，则g(1)=sigma(n|d, mu(d)\*f(d))，

因为f(d)=(n/d)\*(n/d)\*(n/d)，所以g(1)=sigma( mu(d)\*(n/d)\*(n/d)\*(n/d) ).

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

#include <stdio.h>

#include <string.h>

#include <iostream>

#include <algorithm>

#include <vector>

#include <queue>

#include <set>

#include <map>

#include <string>

#include <math.h>

#include <stdlib.h>

#include <time.h>

using namespace std;

const int MAXN = 1000000;

bool check[MAXN+10];

int prime[MAXN+10];

int mu[MAXN+10];

void Moblus()

{

memset(check,false,sizeof(check));

mu[1] = 1;

int tot = 0;

for(int i = 2; i <= MAXN; i++)

{

if( !check[i] )

{

prime[tot++] = i;

mu[i] = -1;

}

for(int j = 0; j < tot; j++)

{

if(i \* prime[j] > MAXN) break;

check[i \* prime[j]] = true;

if( i % prime[j] == 0)

{

mu[i \* prime[j]] = 0;

break;

}

else

{

mu[i \* prime[j]] = -mu[i];

}

}

}

}

int main()

{

//freopen("in.txt","r",stdin);

//freopen("out.txt","w",stdout);

int T,n;

Moblus();

scanf("%d",&T);

while(T--)

{

scanf("%d",&n);

long long ans = 3;

for(int i = 1;i <= n;i++)

ans += (long long)mu[i]\*(n/i)\*(n/i)\*((n/i)+3);

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

}

return 0;

}

/\*互质的对数\*/

#include<cstdio>

#include<cstring>

#include<algorithm>

using namespace std;

const int MAXN = 1e5+10;

typedef long long LL;

LL F[MAXN],f[MAXN];

int pri[MAXN],pri\_num;

int mu[MAXN];//莫比乌斯函数值

int vis[MAXN],a[MAXN];

void mobius(int N) //筛法求莫比乌斯函数

{

pri\_num = 0;//素数个数

memset(vis, 0, sizeof(vis));

vis[1] = mu[1] = 1;

for(int i = 2; i <=N; i++)

{

if(!vis[i])

{

pri[pri\_num++] = i;

mu[i] = -1;

}

for(int j=0; j<pri\_num && i\*pri[j]<N ;j++)

{

vis[i\*pri[j]]=1;//标记非素数

//eg:i=3,i%2,mu[3\*2]=-mu[3]=1;----;i=6,i%5,mu[6\*5]=-mu[6]=-1;

if(i%pri[j])mu[i\*pri[j]] = -mu[i];

else

{

mu[i\*pri[j]] = 0;

break;

}

}

}

}

inline LL get(int x)

{

return (LL)((x\*(x-1))/2);

}

int main()

{

mobius(100005);

int n;

while(~scanf("%d",&n))

{

memset(F,0,sizeof(F));

memset(f,0,sizeof(f));

int mmax = -1;

for(int i = 1; i <= n; i++)

{

scanf("%d",&a[i]);

f[a[i]]++;

mmax = max(mmax, a[i]);

}

//求F[N]

for(int i=1;i<=mmax;i++)

{

for(int j=i;j<=mmax;j+= i)

{

F[i]+=f[j];//个数

}

F[i]=get(F[i]);//C(N,2),表示对数；保证了gcd(a,b);(a<b)

}

LL ans = 0;

for(int i=1; i<=mmax; i++)

ans+=F[i]\*mu[i];

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

}

return 0;

}

/\*欧拉函数\*/

#include<bits/stdc++.h>

using namespace std;

const int maxn = 100000000;

/\*线性筛O(n)时间复杂度内筛出maxn内欧拉函数值\*/

int m[maxn],phi[maxn],p[maxn],pt;//m[i]是i的最小素因数，p是素数，pt是素数个数

int make()

{

phi[1]=1;

int N=maxn;

int k;

phi[1]=1;

for(int i=2;i<N;i++)

{

if(!m[i])//i是素数

p[pt++]=m[i]=i,phi[i]=i-1;

for(int j=0;j<pt&&(k=p[j]\*i)<N;j++)

{

m[k]=p[j];

if(m[i]==p[j])//为了保证以后的数不被再筛，要break

{

phi[k]=phi[i]\*p[j];

/\*这里的phi[k]与phi[i]后面的∏(p[i]-1)/p[i]都一样（m[i]==p[j]）只差一个p[j]，就可以保证∏(p[i]-1)/p[i]前面也一样了\*/

break;

}

else

phi[k]=phi[i]\*(p[j]-1);//积性函数性质，f(i\*k)=f(i)\*f(k)

}

}

}

int main(){

make();

/\* for(int i = 1;i<maxn;i++)\*/cout<<" "<<phi[maxn-1]<<endl;

}

数论

//ax === b mod m.cpp

/\*

ax === b(mod m)相当于ax +my = b

根据前面求 ax+my = b的步骤：

（1）求x,y,t = gcd(a,m)

（2）如果b%d!=0无解

（3） 由ax+my = t,得a(x\*b/t)+m(y\*b/t) = b

于是ax===b(mod m)的一个解为x = x\*b/t

（4）ax === b(mod m)的所有解为：

x = (x\*b/t+ i\*m/t)mod m,i = 0,1,2,...d-1

\*/

#include<iostream>

using namespace std;

int main(){

long long a,b,x,m,y;

while(cin>>a>>b>>m){

long long t = extend\_gcd(a,m,x,y);//拓展欧几里得

if(b%t == 0){

x =(x\*b/t)%m;

while(x<0)x = (x+m)%m;

for(int i = 0;i<t;i++){

cout<<(x+i\*(m/t))%m<<endl;

}

}

else cout<<"no solutions"<<endl;

}

}

//ax+by=c.cpp

/\*ax+by=c的整数解算法 \*/

/\*设t=gcd(a, b)，记a=mt，b=nt

求特解：求整系数方程ax+by=t的一个整数解x0，y0，

求一般解：若 t不是c的因数，则整系数方程ax+by=c无整数解；

若 t是c的因数，记c=gt，则整系数方程ax+by=c一般解为：

x=gx0+bk, y=gy0-ak, k为任何整数

如果x1<=x<=x2&&y1<=y<=y2;

存在这样的x,y输出“have”反之“nothing”

\*/

#include<iostream>

using namespace std;

int main()

{

long long a,b,c,x,y,x0,y0,x1,x2,y1,y2,kx1,kx2,ky1,ky2;

while(cin>>a>>b>>c>>x1>>x2>>y1>>y2){

long long t = extend\_gcd(a,b,x0,y0);//拓展欧几里得

if(c%t == 0){

long long g = c/t;

long long m = a/t;

long long n = b/t;

kx1 = (x1- g\*x0-1)/n;

kx2 = (x2- g\*x0+1)/n;

ky2 = (y1- g\*y0+1)/(-m);

ky1 = (y2- g\*y0-1)/(-m);

for(int k = kx1+1;k<kx2;k++)

if(k>ky1&&k<ky2){

cout<<"have"<<endl;

x = g\*x0+n\*k;

y = g\*y0-m\*k;

cout<<x<<" "<<y<<endl;

}

}

else cout<<"nothing"<<endl;

}

}

//fast\_mod.cpp

/\*

n^m(mod c)

A^13 = A^8\*A^4\*A^1用二进制表示A^(1101) = A^(1000)\*A^(100)\*A^(1)

例如：：N = 13 = 1101（2），N>>1 = 110（2） ，

\*/

long long fast\_mod(long long n,long long m,long long c){ //求快速幂

long long ret=1;

while(m){

if(m&1)ret = ret\*n%c;

n=n\*n%c;//这里保证m的移位和n的幂数对等，右移x位就为n的2^x次幂

m>>=1;

}

return ret;

}  
//gcd.cpp

#include<iostream>

using namespace std;

int gcd(int a,int b) {

int c;

if(a ==0) return b;

while(b!=0) c=b,b=a%b,a=c;

return a;

}

//gcd.cpp

long long gcd(long long a, long long b){

if ((b==0)&(a==0)) //表示无最大公因数

return -1;

return b == 0?a:gcd(b, a%b);

}

//extend\_gcd.cpp（非递归）

long long extend\_gcd(long long a, long long b, long long &x, long long &y){

long long x0,x1,x2, y0,y1,y2;

long long r0,r1,r2, q;

if((a== 0)&&(b==0)){

x=0; y=0;

return -1;

}

if((a== 0)&&(b!=0)){

x = 0; y = 1 ;

return b;

}

if((b == 0) &&(a!=0)) {

x = 1; y = 0 ;

return a;

}

if((a!= 0) &&(b!=0)){

x0=0; x1=1; r0= a;

y0 = 1; r1 = b;

r2=r0 % r1;y1 =0- r0/r1;

x2=1; y2=y1;

while((r1%r2) != 0) {

r0=r1; r1=r2;

q= r0/r1;

x2=x0-x1\*q; y2=y0-y1\*q;

x0=x1; x1=x2; y0=y1; y1=y2; r2=r0 % r1;

}

x = x2; y = y2;

return r2;

}

}

//extend\_gcd.cpp（递归）

long long extend\_gcd(long long a, long long b, long long &x,long long &y){

long long t,m;

if ((b==0)&(a==0)) return -1;

if (b==0){

x=1;y=0;

return a;

}

else {

m=extend\_gcd(b,a % b,x,y);

t=x;x=y;y=t-(a/b)\*y;

}

return m;

}

int main()

{

long long a,b,x,y;

while(cin>>a>>b){

long long t1 = gcd(a,b);

long long t2 = extend\_gcd(a,b,x,y);

cout<<t1<<" "<<x<<" "<<y<<endl;

}

}

//niyuan.cpp

/\*

a m 互质

ax === 1 mod m相当于ax +my = 1

ax+my = 1的步骤：

（1）求x,y,t = gcd(a,m)

（2）如果t != 1无解

（3）于是ax===1(mod m)的一个解为x = x

\*/

#include<iostream>

using namespace std;

//extend\_gcd(a, m, &x, &y); x为所求解 when gcd(a,m) == 1

long mod\_reverse(long long a, long long m){

long long y=0,x=1,r=a%m, q, t, mm=m;//初始化

if(r<0)r=r+m;

while((m%r) != 0) {

a=m; m=r;

q= a/m; r=a % m;

t=x; x=y-x\*q; y=t;

}

if(r!=1) return 0;

if(x<0) x=x+mm;

return x;

}

int main(){

long long a,m;

while(cin>>a>>m){

cout<<mod\_reverse(a,m)<<endl;

}

}

//中国剩余定理.cpp

/\*设 m1，m2,m3...mn是两两互素的n个正整数，

记m = m1m2m3...mn ，Mi = n/mi

，1≤i≤n，那么，下列模线性方程组

x === b1(mod m1)

x === b2(mod m2)

...

x === bn(mod mn)

的解为x0 = (b1m1y1+b2m2y2+...+bnmnyn)，

其中yi 是Miyi ===1(mod mi) 的解，i=1，2，3，…，n

\*/

#include<iostream>

using namespace std;

const int maxn = 1000;

long long m[maxn];

long long b[maxn];

long long ChinaRemainder(int n,long long m0[], long long b[]){

long long d,x,y,t,m=1,a=0,Mi;

int i;

for (i=0; i< n ; i++) m=m\*m0[i];

for (i=0; i< n ; i++) {

Mi=m / m0[i];

extend\_gcd(Mi, m0[i], x, y); //求逆元拓展欧几里得

//或x=mod\_reverse(MM, m0[i]);

a=(a+Mi\*x\*b[i]) % m;

}

while(a<0)a = a+m;

return a;

}

int main(){

int n;

while(cin>>n){

for(int i = 0;i<n;i++)

cin>>b[i]>>m[i];

cout<<ChinaRemainder(n,m,b)<<endl;

}

}

**RSA加密**

/\*两个大素数p,q n = p\*q

n 的欧拉数 b(n) = (p-1)(q-1)

e 与 b(n)互质且0<=e<=b(n)-1

解密密匙d 满足d\*e===1(mod b(n)) (n d互质)

e n 为公钥 d为私钥

加密信息 m

加密 ci = mi^e(mod n)

解密 mi = ci^d(mod n)

\*/

图论

#include<iostream>

#include<vector>

using namespace std;

/\*第一次搜索到边（u，v）时v是如果白色，为树边，灰色，后向边，黑色，前向或横向边\*/

/\*拓扑排序：没搜完一个节点，加到桟中，全部搜完，出栈就是拓扑排序\*/

/\*强连通分量：1.原图dfs。2.将图反转。3.按照原图节点完成时间的降序dfs，pa为-1的个数为强分量的个数\*/

vector<int >VZ[1000];//记录图

vector<int >VT[1000];

int id[1000];//id为完成的时间

int time\_d[1000];//发现节点的时间

int time\_f[1000];//完成节点的时间

int color[1000];// 0发现的节点 1发现未完成的节点2完成的节点

int pa[1000];//父亲节点

void dfs\_visit(vector<int> \*V,int v,int &time){

time++;

time\_d[v] = time;

color[v] = 1;

int \_size = V[v].size();

for(int i = 0;i<\_size;i++)

if(color[V[v][i]] == 0){

pa[V[v][i]] = v;

dfs\_visit(V,V[v][i],time);

}

color[v] = 2;

time++;

time\_f[v] = time;

}

void dfs(vector<int> \*V,int m,int n){

//初始化

for(int i = 1;i<=m;i++){

color[i] = 0;pa[i] = -1;

}

//对每个节点深搜

int time = 0;

for(int i = 1;i<=m;i++){

if(color[i] == 0){

dfs\_visit(V,i,time);

}

}

}

int main(){

int m,n;

while(cin>>m>>n){

//建图

for(int i = 0;i<n;i++){

int a,b;

cin>>a>>b;

id[i] = 0;

VZ[a].push\_back(b);

VT[b].push\_back(a);

}

//生成深度优先搜索树

dfs(VZ,m,n);

for(int i = 1;i<=m;i++){

cout<<i<<" "<<time\_f[i]<<endl;

}cout<<endl;

int ans = 0;

for(int i = 1;i<=m;i++){

if(pa[i] == -1)ans++;

}

cout<<ans<<endl;

}

}

/\*

8 14

3 4

3 7

7 6

7 8

6 7

8 8

4 3

4 8

2 3

2 5

2 6

5 1

5 6

1 2

6 8

1 2

2 5

3 5

3 6

4 2

5 4

6 6

\*/