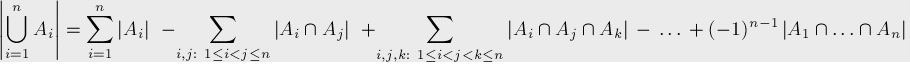
容斥原理的二进制实现模版

//http://www.cnblogs.com/xin-hua/p/3213050.html

最近学习容斥原理，实现容斥原理大致有三种方法：dfs，队列数组，二进制。

今天主要讲下二进制实现容斥原理：

有一个集合{A1……An}，求集合的子集？很显然答案为



也就是2^n个，也就是每一个子集有唯一标志符 i (0<i<2^n,空集除外),也就是说有唯一的二进制表示

代码看下面的：

#include<iostream>

#include<stdio.h>

#include<algorithm>

#include<iomanip>

#include<cmath>

#include<cstring>

#include<vector>

#include<stdlib.h>

using namespace std;

int prime[40000],m;

bool f[40000];

vector<int>p;//存放质因数

//用筛法初始化40000以内的质数，将质数存放在prime数组中，m记录大小

int init(){

m=0;

for(int i=2; i<40000; i++){

if (f[i]==0) prime[m++]=i;//质数

//筛去合数

for (int j=0; j<m&&i\*prime[j]<40000; j++){

f[i\*prime[j]]=1;

if (i%prime[j]==0) break;//保证每个数只筛去一次

}

}

}

//对n分解质因数

void factor(int n){

p.clear();

for (int i=0; i<m&&prime[i]\*prime[i]<=n; i++){

if (n%prime[i]==0){

p.push\_back(prime[i]);

n/=prime[i];

while (n%prime[i]==0)

n/=prime[i];

}

}

if(n>1) p.push\_back(n);

}

//用二进制实现容斥原理,求区间[1,r]内与n互素的数的个数

int solve(int r){

int sum = 0;

//i的范围是1-2^p.size()，空集除外，每一个子集所对应的

//二进制都不一样，也就是i

for (int i=1; i<(1<<p.size()); ++i){

int mult = 1,bits = 0;

for (int j=0; j<p.size(); ++j)

if (i&(1<<j)){//与i的二进制的第j位比较，看是否为1，是则选中

bits++;//计算i中1的个数，也就是质因数的个数

mult \*= p[j];

}

int cur = r / mult;

if (bits & 1)//若1的个数是奇数则进行加法，否则进行减法

sum += cur;

else sum -= cur;

}

return r - sum;//用总的数目-与n不互素的个数

}

int main(){

init();

int n,r;

while(cin>>n>>r){

factor(n);

cout<<solve(r)<<endl;

}

return 0;

}