莫比乌斯反演在数论中占有重要的地位，许多情况下能大大简化运算。那么我们先来认识莫比乌斯反演公式。

**定理：**http://img.blog.csdn.net/20140416155848046和http://img.blog.csdn.net/20140416155900312是定义在非负整数集合上的两个函数，并且满足条件http://img.blog.csdn.net/20140416155727296，那么我们得到结论

http://img.blog.csdn.net/20140416160124562

在上面的公式中有一个http://img.blog.csdn.net/20140416160256500函数，它的定义如下：

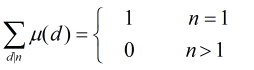
    （1）若http://img.blog.csdn.net/20140416160429140，那么http://img.blog.csdn.net/20140416160522250

    （2）若http://img.blog.csdn.net/20140416160632156，http://img.blog.csdn.net/20140416160716765均为互异素数，那么http://img.blog.csdn.net/20140416160909015

    （3）其它情况下http://img.blog.csdn.net/20140416160949437

对于http://img.blog.csdn.net/20140416160256500函数，它有如下的常见性质：

    （1）对任意正整数http://img.blog.csdn.net/20140416161324484有



        （2）对任意正整数http://img.blog.csdn.net/20140416161324484有

http://img.blog.csdn.net/20140416162025500

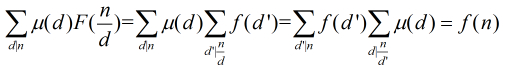
线性筛选求莫比乌斯反演函数代码。

void Init()

1. {
2. memset(vis,0,sizeof(vis));
3. mu[1] = 1;
4. cnt = 0;
5. for(int i=2; i<N; i++)
6. {
7. if(!vis[i])
8. {
9. prime[cnt++] = i;
10. mu[i] = -1;
11. }
12. for(int j=0; j<cnt&&i\*prime[j]<N; j++)
13. {
14. vis[i\*prime[j]] = 1;
15. if(i%prime[j]) mu[i\*prime[j]] = -mu[i];
16. else
17. {
18. mu[i\*prime[j]] = 0;
19. break;
20. }
21. }
22. }
23. }

有了上面的知识，现在我们来证明莫比乌斯反演定理。

**证明**

****

证明完毕！

嗯，有了莫比乌斯反演，很多问题都可以简化了，接下来我们来看看莫比乌斯反演在数论中如何简化运算的。

**题目：**[**http://bz.cdqzoi.com/JudgeOnline/problem.php?id=2818**](http://bz.cdqzoi.com/JudgeOnline/problem.php?id=2818)

**题意：**给一个正整数http://img.blog.csdn.net/20140416161324484，其中http://img.blog.csdn.net/20140416171309984，求使得http://img.blog.csdn.net/20140416171448562为质数的http://img.blog.csdn.net/20140416171546750的个数，http://img.blog.csdn.net/20140416171719562。

**分析：**对于本题，因为是使得http://img.blog.csdn.net/20140416171448562为质数，所以必然要枚举小于等于http://img.blog.csdn.net/20140416161324484的质数，那么对于每一个质数http://img.blog.csdn.net/20140416175101218，只

     需要求在区间http://img.blog.csdn.net/20140416175411265中，满足有序对http://img.blog.csdn.net/20140416171546750互质的对数。

     也就是说，现在问题转化为：在区间http://img.blog.csdn.net/20140416175931093中，存在多少个有序对使得http://img.blog.csdn.net/20140416171546750互质，这个问题就简单啦，因为

     是有序对，不妨设http://img.blog.csdn.net/20140416180108921，那么我们如果枚举每一个http://img.blog.csdn.net/20140416180159812，小于http://img.blog.csdn.net/20140416180159812有多少个http://img.blog.csdn.net/20140416180249093与http://img.blog.csdn.net/20140416180159812互素，这正是欧拉函数。所以

     我们可以递推法求欧拉函数，将得到的答案乘以2即可，但是这里乘以2后还有漏计算了的，那么有哪些呢？

     是http://img.blog.csdn.net/20140416180835484且为素数的情况，再加上就行了。

1. #include <iostream>
2. #include <string.h>
3. #include <stdio.h>
4. #include <bitset>
6. using namespace std;
7. typedef long long LL;
8. const int N = 10000010;
10. bitset<N> prime;
11. LL phi[N];
12. LL f[N];
13. int p[N];
14. int k;
16. void isprime()
17. {
18. k = 0;
19. prime.set();
20. for(int i=2; i<N; i++)
21. {
22. if(prime[i])
23. {
24. p[k++] = i;
25. for(int j=i+i; j<N; j+=i)
26. prime[j] = false;
27. }
28. }
29. }
31. void Init()
32. {
33. for(int i=1; i<N; i++)  phi[i] = i;
34. for(int i=2; i<N; i+=2) phi[i] >>= 1;
35. for(int i=3; i<N; i+=2)
36. {
37. if(phi[i] == i)
38. {
39. for(int j=i; j<N; j+=i)
40. phi[j] = phi[j] - phi[j] / i;
41. }
42. }
43. f[1] = 0;
44. for(int i=2;i<N;i++)
45. f[i] = f[i-1] + (phi[i]<<1);
46. }
48. LL Solve(int n)
49. {
50. LL ans = 0;
51. for(int i=0; i<k&&p[i]<=n; i++)
52. ans += 1 + f[n/p[i]];
53. return ans;
54. }
56. int main()
57. {
58. Init();
59. isprime();
60. int n;
61. scanf("%d",&n);
62. printf("%I64d\n",Solve(n));
63. return 0;
64. }

嗯，上题不算太难，普通的欧拉函数就可以搞定，接下来我们来看看它的升级版。

**题意：**给定两个数http://img.blog.csdn.net/20140417194053968和http://img.blog.csdn.net/20140417194128859，其中http://img.blog.csdn.net/20140417194225734，http://img.blog.csdn.net/20140417194323328，求http://img.blog.csdn.net/20140417194427609为质数的http://img.blog.csdn.net/20140417194508812有多少对？其中http://img.blog.csdn.net/20140417194053968和http://img.blog.csdn.net/20140417194128859的范围是http://img.blog.csdn.net/20140417194716859。

**分析：**本题与上题不同的是http://img.blog.csdn.net/20140417194053968和http://img.blog.csdn.net/20140417194128859不一定相同。在这里我们用莫比乌斯反演来解决，文章开头也说了它能大大简化

     运算。我们知道莫比乌斯反演的一般描述为：

http://img.blog.csdn.net/20140417200055140

     其实它还有另一种描述，本题也是用到这种。那就是：

http://img.blog.csdn.net/20140417200609375     好了，到了这里，我们开始进入正题。。。     对于本题，我们设

http://img.blog.csdn.net/20140417200802406为满足http://img.blog.csdn.net/20140417200847703且http://img.blog.csdn.net/20140417194225734和http://img.blog.csdn.net/20140417194323328的http://img.blog.csdn.net/20140417194508812的对数

http://img.blog.csdn.net/20140417201039703为满足http://img.blog.csdn.net/20140417201126828且http://img.blog.csdn.net/20140417194225734和http://img.blog.csdn.net/20140417194323328的http://img.blog.csdn.net/20140417194508812的对数

     那么，很显然http://img.blog.csdn.net/20140417201441703，反演后得到http://img.blog.csdn.net/20140417202124531

     因为题目要求是http://img.blog.csdn.net/20140417194427609为质数，那么我们枚举每一个质数http://img.blog.csdn.net/20140417202506515，然后得到

http://img.blog.csdn.net/20140417202926687

     如果直接这样做肯定TLE，那么我们必须优化。

     我们设http://img.blog.csdn.net/20140417203936468，那么继续得到http://img.blog.csdn.net/20140417204525703。

     到了这里，可以看出如果我们可以先预处理出所有的http://img.blog.csdn.net/20140417204714281对应的http://img.blog.csdn.net/20140417204813281的值，那么本题就解决了。

     我们设http://img.blog.csdn.net/20140418092417984，注意这里http://img.blog.csdn.net/20140417202506515为素数，http://img.blog.csdn.net/20140418092530828。

     那么，我们枚举每一个http://img.blog.csdn.net/20140418092622046，得到http://img.blog.csdn.net/20140418092705046，现在分情况讨论：

     （1）如果http://img.blog.csdn.net/20140418092622046整除http://img.blog.csdn.net/20140418093612203，那么得到

http://img.blog.csdn.net/20140418093258359

     （2）如果http://img.blog.csdn.net/20140418092622046不整除http://img.blog.csdn.net/20140418093612203，那么得到

http://img.blog.csdn.net/20140418094003625

1. #include <iostream>
2. #include <string.h>
3. #include <stdio.h>
5. using namespace std;
6. typedef long long LL;
7. const int N = 10000005;
9. bool vis[N];
10. int p[N];
11. int cnt;
12. int g[N],u[N],sum[N];
14. void Init()
15. {
16. memset(vis,0,sizeof(vis));
17. u[1] = 1;
18. cnt = 0;
19. for(int i=2;i<N;i++)
20. {
21. if(!vis[i])
22. {
23. p[cnt++] = i;
24. u[i] = -1;
25. g[i] = 1;
26. }
27. for(int j=0;j<cnt&&i\*p[j]<N;j++)
28. {
29. vis[i\*p[j]] = 1;
30. if(i%p[j])
31. {
32. u[i\*p[j]] = -u[i];
33. g[i\*p[j]] = u[i] - g[i];
34. }
35. else
36. {
37. u[i\*p[j]] = 0;
38. g[i\*p[j]] = u[i];
39. break;
40. }
41. }
42. }
43. sum[0] = 0;
44. for(int i=1;i<N;i++)
45. sum[i] = sum[i-1] + g[i];
46. }
48. int main()
49. {
50. Init();
51. int T;
52. scanf("%d",&T);
53. while(T--)
54. {
55. LL n,m;
56. cin>>n>>m;
57. if(n > m) swap(n,m);
58. LL ans = 0;
59. for(int i=1,last;i<=n;i=last+1)
60. {
61. last = min(n/(n/i),m/(m/i));
62. ans += (n/i)\*(m/i)\*(sum[last]-sum[i-1]);
63. }
64. cout<<ans<<endl;
65. }
66. return 0;
67. }