《Queries》解题报告

—POI XIV Stage.1 ZAP

By Kwc-Oliver

【问题简述】

有N个询问，每次询问a, b, d，问有多少个(x, y)满足 $1 \le x \le a$, $1 \le y \le b$，且$Gcd(x, y) = d$。

$1 \le d \le a, b \le 50000$, $1 \le N \le 50000$

【引言】

BZOJ 2045是本题的简化版。唯一的不同就是本题有很多组询问。当问题量化之后如何处理、如何去除冗余，将是本文所要讨论的。最后利用此方法解决拓展问题—NOI 2010 day1 《能量采集》一题。

【分析】

先将问题转化为:对于给定的a, b, d，求有多少组(x, y)满足[1]$1 \le x \le a/d$, $1 \le y \le b/d$，且$Gcd(x, y) = 1$。

那么下文的a, b分别为原题中的$a/d, b/d$。

设$G(x, y)$表示有多少组$x \le a, y \le b$，且x, y互质。

设$F(a, b, k)$表示有多少组$x \le a, y \le b$，且$Gcd(a, b) \ge k$。

那么显然有：

$$F(a, b, k) = (a/k) * (b/k)$$

根据容斥原理有：

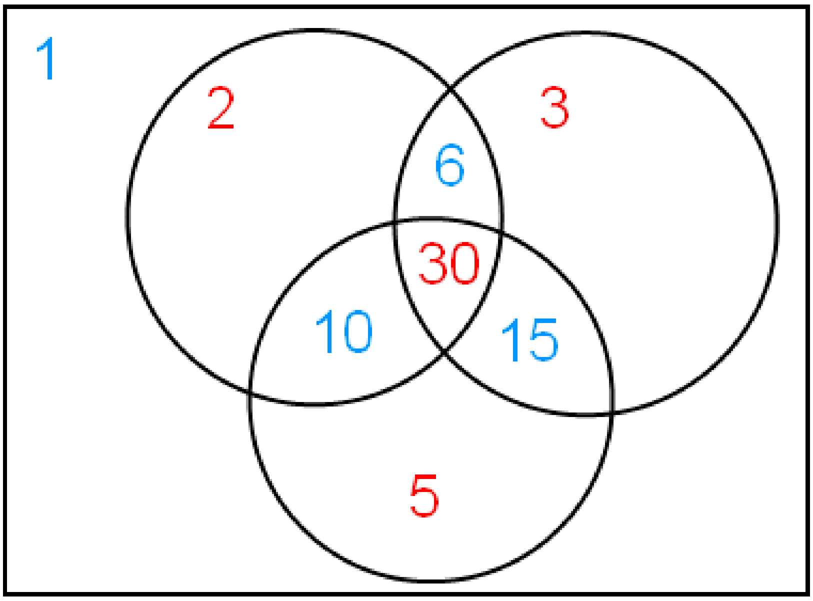
$$Ans = G(x, y) = P_1 * F(a, b, 1) + P_2 * F(a, b, 2) + P_3 * F(a, b, 3) + ... + P_x * F(a, b, x)$$

其中

[1]本文所有”/”均表示整除

$$P_i = \begin{cases} 0 & i\ have\ some\ equal\ prime\ factors \\ 1 & i\ have\ odd\ prime\ factors \\ -1 & i\ have\ even\ prime\ factors \end{cases}$$

这个可以自己画个图，根据容斥原理是可以得到的，下面附上一个简图。



注意到P_i与a, b, d无关。所以可以预处理出来。

那么现在，对于每个询问我们可以做到$O(a/d)$。但由于有50000组，所以还需优化。

由于预处理不影响时间复杂度，所以我们仍然考虑

$$Ans = G(x, y) = \sum P_x * (a/x) * (b/x)$$

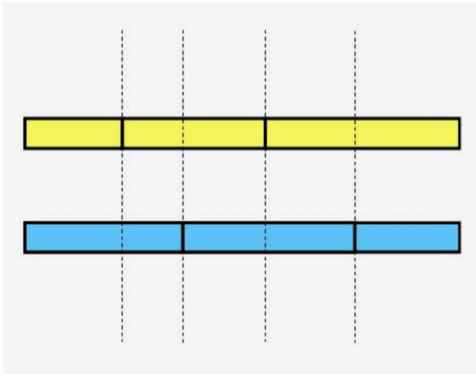
对于其中连续的k项，有可能有$a/x = a/(x+k), b/x = b/(x+k)$。那么对于这$k$项，我们可以直接算出，即为

$$(P_{x+k} - P_{x-1}) * (a/x) * (b/x)$$

那么将这些冗余去除后，时间复杂度是否有改善呢？

对于a/x，其不同的取值最多只有$2\sqrt{a}$，对于b/x也一样。

但是二元组$(a/x, b/x)$的取值却不是$2\sqrt{a} * 2\sqrt{b}$。



如图所示，可以发现这是一个用$2\sqrt{a}+2\sqrt{b}$点来划分区间的模型，所以二元组最多只有$2\sqrt{a}+2\sqrt{b}$个。

所以这道题，我们可以在$O(N*(2\sqrt{a/d}+2\sqrt{b/d})+a*\sqrt{a})$的时间内完成。

最后说一点关于实现的问题：

1，预处理P时要注意大于$\sqrt{a}$的质数，此时应处理好其为有一个质因子的情况。

2，如何求得二元组$(a/x,b/x)$取值范围相同的一段：对于当前点x，其向右延伸最远为$min\{a/(a/x),b/(b/x)\}$。其原理为a/x为下界，那么$a/(a/x)$就是上界了。

【拓展】

NOI 2010 day1 《能量采集》与本题类似，现将其问题模型抽象出来，即求：

$$2*\sum_{i=1}^{N}\sum_{j=1}^{M}Gcd(i,j)-N*M$$

$N,M\leq 10^5$

分析：对于相同的$d=Gcd(i,j)$，我们利用上文方法，可以在$O(\sqrt{N})$的时间内做出来。但是我们要枚举d，这样就是$O(N\sqrt{N})$，实际测试中也能AC。

但利用上文讲述的去除冗余的方法，我们发现，对于相邻的d，如果存在$N/d=N/(d+k),M/d=M/(d+k)$，那么是不必重复计算的，并且根据上文分析，也可以得出这个枚举的效率降为$O(\sqrt{N})$，那么这里的复杂度就降为$O(N)$了。而本题最后的瓶颈则是分解质因数的$O(N\log N)$了。

【总结】

我们首先转化问题，运用容斥原理，得到了一个很好的线性$O(a/d)$的并且有一部分可以预处理的算法。而面对大量数据，我们则进一步去除冗余，并分析得到确实会降低复杂度。

优化是无止境的。

【参考资料】

1,《POI XVI Stage I Report》陈丹琦

2，http://hi.baidu.com/wjbzbmr/blog/item/e341110873cd9033b1351d81.html

【附录】

代码：

```
#include <iostream>
#include <cstdio>
#include <algorithm>

using namespace std;

int P[51000];
int a,b,d,N;

void Prepare(){
	int _i,f;
	P[1]=1;
	for (int i=2;i<=50000;i++){
		_i=i;f=0;
		for (int j=2;j<=i&&j<=300;j++){
			if (!(i%j)){
				i/=j;
				P[_i]++;
				if (!(i%j)){
					f=1;
					break;
				}
			}
		}
		if (i>1) P[_i]++;
		if (f) P[_i]=0;
		else P[_i]&1?P[_i]=-1:P[_i]=1;
		i=_i;
		P[i]=P[i-1]+P[i];
	}
}

void solve(int A,int B,int d){
	if(A>B) swap(A,B);
	int l=1,Ans=0,r,c=0,p1,p2;
	while (l<=A){
		c++;
		r=min(min(A/(p1=A/l),B/(p2=B/l)),A);
		Ans+=p1*p2*(P[r]-P[l-1]);
		l=r+1;
	}
	printf("%d\n",Ans);
}

int main(){
	freopen("idiotic.in","r",stdin);
```

```
    freopen("idiotic.out","w",stdout);
    cin>>N;
    Prepare();
    while (N--){
        scanf("%d%d%d",&a,&b,&d);
        a/=d;b/=d;
        solve(a,b,d);
    }
}
```