和与积

calc

【题意简述】

给出 N，统计满足下面条件的数对(a,b)的个数：

1.$1<=a<b<=N$
2.$a+b$ 整除 ab

$N<=2^{31}-1$

【算法分析】

算法一：

题意再简单明了不过了，再配上一个轻松暴力又好写的算法---枚举 a,b，可以拿到 20 分。

时间复杂度：$O(N^2)$

期望得分：20

算法二：

想要拿到更可观的分数，我们要深入地做一些数学推导，从而发现问题的本质。

设$d=\gcd(a,b),a=dx,b=dy$

那么$(a+b)\mid ab \Leftrightarrow d(x+y)\mid d^2xy \Leftrightarrow (x+y)\mid dxy \Leftrightarrow (x+y)\mid d \Leftrightarrow d=k(x+y),k\in N$

这是因为 x 与 $x+y$ 以及 y 与 $x+y$ 都没有共同的因子

因此，原问题所求的二元组(a,b)的个数，与三元组(k,x,y)(x<y,gcd(x,y)=1,ky(x+y)<=n)

的个数是相等的，而对于一对合法的(x,y)，对于$1\sim\left\lfloor\frac{n}{y(x+y)}\right\rfloor$，k 都是可以取的，所以

我们可以通过下面的公式得到答案：

$$Ans=\sum_{x=1}^{\gcd(x,y)=1}\sum_{y=x+1}\left\lfloor\frac{n}{y(x+y)}\right\rfloor$$

为了方便起见，我们可以设 $p=y,q=x+y$，那么进一步可以得到：

$$Ans=\sum_{p=2}^{2p-1,gcd(p,q)=1}\sum_{q=p+1}\left\lfloor\frac{n}{pq}\right\rfloor$$

由此我们可以枚举 p,q 计算答案，由于 p,q 都只需要在$\sqrt{n}$的范围内枚举，再加上判断 p,q 是否互质的操作，于是我们得到了一个$O(\sqrt{n}\sum_{i=2}^{\sqrt{n}}L_i)$的算法,其中$L_i$是 i 的质因子个数。

时间复杂度：$O(\sqrt{n}\sum_{i=2}^{\sqrt{n}}L_i)$

期望得分：$40\sim60$

算法三：

我们将算法二中的公式继续化简，将 p 与 q 的关系剥离开可以得到:

$$Ans=\sum_{p=2}^{2p-1,gcd(p,q)=1}\sum_{q=p+1}\left\lfloor\frac{\left\lfloor\frac{n}{p}\right\rfloor}{q}\right\rfloor$$

这一点给了我们一些启发，当我们枚举了 p 之后，$\left\lfloor\frac{n}{p}\right\rfloor$部分的值将不会改变，不妨设它为 T，说不定不用枚举 q，而采用其他的方法也可以一起计算所有的值之和，因为 p 与 q 之间的关系已经被剥离了。事实上，我们是有办法的，这是因为$\left\lfloor\frac{T}{q}\right\rfloor$所得到的值对于连续的 q 有很多都是相同的，$\forall q\in\left[\left\lfloor\frac{T}{x+1}\right\rfloor+1,\left\lfloor\frac{T}{x}\right\rfloor\right]$有$\left\lfloor\frac{T}{q}\right\rfloor=x$。当然，有一点不能忽视，那就是我们要求 p 与 q 互质，所有$\gcd(p,q)\neq1$的 q 都是不能纳入计算的，这一点使用容斥原理能够在$O(2^{L_p})$的复杂度内很好地解决。我们可以枚举所有的 x，找出使得$\left\lfloor\frac{T}{q}\right\rfloor=x$且$\gcd(p,q)=1$的 q 的个数，这样就得到了一个$O(\sqrt{n}\sum_{i=2}^{\sqrt{n}}2^L)$的算法。虽然这个复杂度似乎看起来不如上一个算法，但是在实践中由于不同的$\left\lfloor\frac{T}{q}\right\rfloor$比较少，实际效果比上一个好了很多。

时间复杂度：$O(\sqrt{n}\sum_{i=2}^{\sqrt{n}}2^L)$

期望得分： $60 \sim 70$

算法四：

对于极限数据 $N <= 2^{31}-1$的巨大范围，上面两个算法似乎还是无能为力，但是我们可以看到，这两种算法各有各的优势，之所以拉慢算法三的原因是$\left\lfloor \frac{T}{q} \right\rfloor$的取值过多，而这一点由于 q 的取值比较少，正是算法二所能弥补的。

我们可以确定一个分界点 k 将这两个算法有机地结合起来，使它们结合之后的复杂度达到一个平衡---在枚举了 p 之后，对于 $q \in [p+1,k]$我们使用算法二解决，而对于 $q \in [k+1,2p-1]$使用算法三解决。可是这个分界点又取在哪里呢？我们同样可以通过精确的数学计算算出最平衡的 k 的取值：

对于 $q \in [p+1,k]$使用算法二的复杂度是 $(k-p)L$

对于 $q \in [k+1,2p-1]$使用算法三的复杂度是 $(\frac{T}{k}-\frac{T}{2p})2^L$

因此总的复杂度 $F=(k-p)L+(\frac{T}{k}-\frac{T}{2p})2^L$

对其求导得到 $F'=L-\frac{2^L T}{k^2}$

令 $F'=L-\frac{2^L T}{k^2}=0$

得 $k=\sqrt{\frac{2^L T}{L}}$

得到了最优的 k 的取值，我们回过头来看看这个解法的时间复杂度：

将 $k=\sqrt{\frac{2^L T}{L}}$ 带入 $F=(k-p)L+(\frac{T}{k}-\frac{T}{2p})2^L$ 得:

$$F=(\sqrt{\frac{2^L T}{L}}-p)L+(\frac{T}{\sqrt{\frac{2^L T}{L}}}-\frac{T}{2p})2^L=2\sqrt{2^L LT}-Lp-\frac{T2^L}{2p}$$

$\because T=\frac{N}{p}$，$\therefore$ 总的时间复杂度约为 $O(\sum_{p=2}^{\sqrt{n}}\sqrt{\frac{2^{L_p}L_p N}{p}})$

对于这个怪异的复杂度，恕我不能把它化成一个更加简单明了的形式，但是这个算法

的性能比上两个算法优越了一个档次，实际效果十分出色，能飞速地通过所有的测试数据。

时间复杂度： $O(\sum_{p=2}^{\sqrt{n}}\sqrt{\frac{2^{L_p}L_pN}{p}})$

期望得分：100

【附】

calc.cpp
C++ Source File
2 KB