Writeup

1. 查看源码，注意到p-1和q-1存在512bit的素因子beta：

```
assert(is_prime(beta) and len(bin(beta)[2:]) == 512)
p = 2*x*beta + 1
q = 2*y*beta + 1
```

显然，这个beta是一个突破口。此外，题目源码中给出了tip:

```
assert(tip == 2*x*y*beta + x + y)
```

所以，不难看出这是考察某种特殊的N分解算法

2. 题目的分解算法来源于《Further Attacks On Server-Aided Rsa Cryptosystems》

We know that $p \equiv q \equiv 1 \bmod \beta$. We can now employ a variant of a method of Lehmer described in [6]. Write $p = x\beta + 1$ and $q = y\beta + 1$, so that $N = xy\beta^2 + (x+y)\beta + 1$. Then $\boxed{(N-1)/\beta = xy\beta + (x+y) = u\beta + v}$ where u and $0 \le v < \beta$ are known and x, y are unknown. We have $\boxed{x + y = v + c\beta,\ xy = u - c,}$ where c is the (unknown) carry in expressing $(N-1)/\beta$ in base β.

Finding x and y is equivalent to finding c, since given c we know $x+y$ and xy, and x, y are obtained as the roots of the quadratic equation $(Z-x)(Z-y) = Z^2 - (x+y)Z + xy$.

根据文章的描述，我们可以转换思路，先计算出x和y值。此外，文章通过巧妙的方法，令x+y=v+c*beta，xy=u-c。根据题目提供的`N`、`tip`和`beta`，我们可以计算出`u`和`v`：

```
h = (N-1)/(g)
u = h/(g) # 4 * xy
v = h%(g) # 2 * (x+y)
```

这样，我们就只需要求解一个未知数`c`，它可能的值范围为`(N-1)/g mod g`

3. 文章中给我们提供了一种求解c的算法：

We observe that $\lambda(N)$, the exponent of the multiplicative group modulo N, is $\lambda(N) = \mathrm{lcm}\{p-1, q-1\} = \mathrm{lcm}\{x\beta, y\beta\}$ and so $\lambda(N)$ divides $xy\beta$. Let a be prime to N. We have

$$a^{u\beta} = a^{xy\beta + c\beta} \equiv a^{c\beta} \bmod N$$

so putting $b = a^{\beta}$ we have $b^u \equiv b^c$. This equation determines c, which is of magnitude $C = \sqrt{N}/\beta^2$, modulo the order of b in the multiplicative group. With high probability the order of b will be nearly as large as xy, which is of magnitude N/β^2. Hence a solution c to $b^c \equiv b^u \bmod N$ with $c \le C$ is very likely to be the correct value.

We now solve this equation by the "baby-step giant-step" method of Shanks [10]. Let D be an integer larger than $\sqrt{C}$ and form the lists

$$b^0, b^D, b^{2D}, \dots, b^{D^2} \bmod N$$

and

$$b^u, b^{u-1}, \dots, b^{u-D} \bmod N.$$

We can sort these lists and find a common value $b^{rD} \equiv b^{u-s}$ in time $O(D^{1+\epsilon})$. Then we recover c as $rD+s$. A low-storage alternative is to use Pollard's λ method [9].

找到一个大于 `sqrt(C)` 的整数D， 然后组成数列**b^0,b^D,b^2D,......,b^D*D mod N** 和 **b^u,b^u-1,......,b^u-D**，我们将这些数列进行排序，然后遍历第一个数列，找到两者的共同值：**b^rD ≡ b^u-D mod N**。那么， c的值即为 `rD+s`。

依据这个，编写exp.py即可求得p、q，实现N的分解。