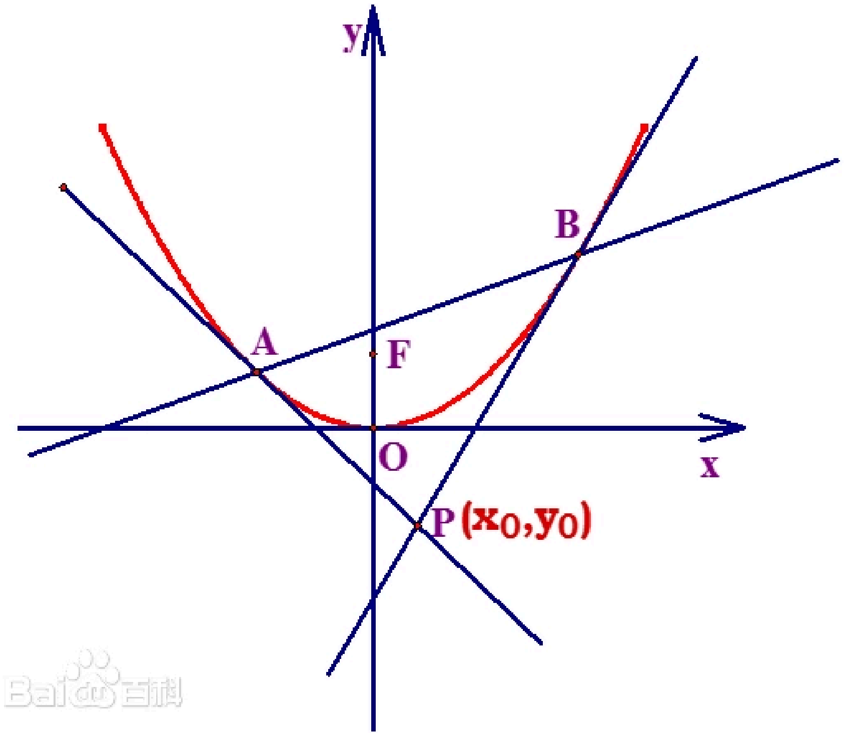
杂七杂八的数论

1.定义F为斐波那契数列,如果F_i能被k整除,那么对于每一个j,F_{i*j}也能被k整除 (CF2033F)

2.斐波那契数列 MOD k,会得到一个周期数列

3.一定存在整数x,y,满足 $ax+by=\gcd(a,b)$ (裴属定理)

4.抛物线与直线所围成的面积S_{AOB}为阿基米德三角形$\triangle ABP$的$\frac{2}{3}$ (阿基米德三角形)



5.从0到x所有数的异或满足以下公式

$$XOR(0,x)=\begin{cases} x & \text{if } x \equiv 0 \pmod 4 \\ 1 & \text{if } x \equiv 1 \pmod 4 \\ x+1 & \text{if } x \equiv 2 \pmod 4 \\ 0 & \text{if } x \equiv 3 \pmod 4 \end{cases}$$

6.**逆元** 给定整数a，满足$gcd(a,m)=1$，方程&ax \mod m = 1&，x的所有解是a在模m意义下的逆，记作a^{-1}
$(a+b) \mod p=(a \mod p+b \mod p) \mod p$
$(a/b) \mod p=(a \mod p*b^{-1} \mod p) \mod p$

7.**筛法**

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 1e8+10;

int sieve[N];
int prime[N];

/*
埃拉托斯特尼筛法
对于任意一个大于 1 的正整数 n，
那么它的 x 倍就是合数（x > 1）。
利用这个结论，我们可以避免很多次不必要的检测。

如果我们从小到大考虑每个数，
然后同时把当前这个数的所有（比自己大的）倍数记为合数，
那么运行结束的时候没有被标记的数就是素数了。
*/
int Era(int n){
    int k = 0;
    for (int i=2;i*i<=n;i++){
        if (!sieve[i]){
            for (int j=i*i;j<=n;j+=i){
                sieve[j] = 1;
            }
        }
    }
    for (int i=2;i<=n;i++){
        if (!sieve[i]){
            prime[k++] = i;
        }
    }
    return k;
}

/*
欧拉筛法
*/
int Eular(int n){
    int k = 0;
    for (int i=2;i<=n;i++){
        if (!sieve[i]){
            prime[k++] = i;
        }
        for (int j=0;j<k;j++){
            if (i * prime[j] > n) break;
            sieve[i * prime[j]] = 1;
            if (i % prime[j] == 0) break;
        }
    }
    return k;
}

int main(){
    int n,len1,len2;
    cin >> n;
    len1 = Era(n);
```

```
    len2 = Eular(n);
}
```

8.快速幂

```
#include <bits/stdc++.h>
using namespace std;
using LL = long long;
LL qpow(LL a,LL x,LL mod){
    LL res = 1;
    while (x){
        if (x&1) res = res * a % mod;
        a = a * a % mod;
        x >>= 1;
    }
    return res;
}
int main(){
    LL a,x,mod;
    cin >> a >> x >> mod;
    cout << qpow(a,x,mod) << '\n';
}
```

9.欧几里得算法求GCD

```
int gcd(int a, int b){
    return b == 0 ? a : gcd(b, a % b);
}
```