收集一些偏僻但值得记住的知识点。

欧拉定理

若$gcd(a,m)=1$，则$a^{\varphi(m)}\equiv 1(\mod m)$

扩展欧拉定理

前置芝士

欧拉函数，欧拉定理（雾）、快速幂

欧拉定理

$$\text{欧拉函数：}\varphi(n)\text{表示小于等于}n\text{的正整数中与}n\text{互质的数的个数。}$$
$$a\text{与}m\text{互质时，}a^{\varphi(m)}\equiv 1 \mod m$$

用途

假设要求$a^b \mod m$

扩展欧拉定理可以在底数a与模数m不互质的情况下，将质数降至与模数同阶的大小，进而进行快速幂的计算。

公式

$$a^c\equiv\begin{cases}a^{c \mod \varphi(m)},gcd(a,m)=1\\a^c,gcd(a,c)\neq 1\cap c<\varphi(m)\\a^{\varphi(m)+(c \mod \varphi(m))},gcd(a,m)\neq 1\cap c\geq\varphi(m)\end{cases}$$

记忆：当$b\geq\varphi(m)$时，$a^b\equiv a^{b \mod \varphi(m)+\varphi(m)}$。

luoguP5091 【模板】扩展欧拉定理

```
#include<bits/stdc++.h>
#define int long long
using namespace std;
const int N=1e6+7;

inline int read(int mod){
    int x=0,f=0;char ch=getchar();
    while(ch<'0'||ch>'9') ch=getchar();
    while(ch>='0'&&ch<='9'){
        x=x*10+ch-'0';
        if(x>=mod) f=1,x%=mod; //改进快读,边读边模
        ch=getchar();
    }
    return x+(f==1?mod:0);
}

int phi(int x){ //根号复杂度求欧拉函数
    int ans=x,sq=sqrt(x);
    for(int i=2;i<=sq;i++){
        if(x%i==0){
            ans=ans/i*(i-1);
            while(x%i==0) x/=i;
        }
    }
    if(x>1) ans=ans/x*(x-1);
    return ans;
}

int fpow(int a,int b,int p){ //快速幂
    int res=1;
    while(b){
        if(b&1) res=res*a%p;
        a=a*a%p;
        b>>=1;
    }
    return res;
}

signed main(){
    int a,b,c,m;
    scanf("%lld%lld",&a,&m);
    b=read(phi(m)); //按照mod m意义下处理b
    printf("%lld\n",fpow(a,b,m));
    return 0;
}
```

模板

```
//luoguP5091 【模板】扩展欧拉定理
//给定a,m,b,求a^b mod m

inline int read(int mod){
```

```cpp
    int x=0,f=0;char ch=getchar();
    while(ch<'0'||ch>'9') ch=getchar();
    while(ch>='0'&&ch<='9'){
        x=x*10+ch-'0';
        if(x>=mod) f=1,x%=mod; //改进快读,边读边模
        ch=getchar();
    }
    return x+(f==1?mod:0);
}

int phi(int x){ //根号复杂度求欧拉函数
    int ans=x,sq=sqrt(x);
    for(int i=2;i<=sq;i++){
        if(x%i==0){
            ans=ans/i*(i-1);
            while(x%i==0) x/=i;
        }
    }
    if(x>1) ans=ans/x*(x-1);
    return ans;
}

int fpow(int a,int b,int p){ //快速幂
    int res=1;
    while(b){
        if(b&1) res=res*a%p;
        a=a*a%p;
        b>>=1;
    }
    return res;
}

signed main(){
    int a,b,c,m;
    scanf("%lld%lld",&a,&m);
    b=read(phi(m)); //按照mod m意义下处理b
    printf("%lld\n",fpow(a,b,m));
    return 0;
}
```

【模板】原根

有一些我不太会证明的结论，希望之后有人在题解区能证一下...

首先，什么样的数才有原根?

结论：$2,4,p^k,2\times p^k$，其中 p 为奇素数，k 为正整数。

怎么求一个数 n 的所有原根?

首先找到 n 的最小原根，设为 g，则 n 的所有原根可以由 g 的若干次乘方得到。

具体地，若 n 存在原根，则其原根个数为 $\varphi(\varphi(n))$，每一个原根都形如 g^k 的形式，要求满足 $\gcd(k,\varphi(n))=1$。

于是，我们在得到 n 的最小原根 g 后便可在 $O(\varphi(n)\log\varphi(n))$ 的时间复杂度内得到 n 的所有原根。

如何得到 n 的最小原根?

枚举即可。从小到大枚举 g 并检验是否是 n 的原根（我听说 n 的最小原根是 $O(n^{0.25})$ 级别的，不太确定对不对）。检验的方法如下：

根据原根的定义，如果 g 是 n 的原根，除了满足 $g^{\varphi(n)}\equiv 1$ 外，还需要满足对于任意更小的 k 有 $g^k\neq 1$。而我们不可能把小于 $\varphi(n)$ 的所有数都拿出来检验。

事实上，关于阶有一条强的性质：如果 $\gcd(a,n)=1$，且 $a^k\equiv 1 \bmod n$，则 $k|\varphi(n)$，所以我们事实上只需要检验 $\varphi(n)$ 的所有真因子即可，进一步地，设 n 的所有素因数为 $p_1,\dots,p_l$，则只需要检验所有的 $\frac{\varphi(n)}{p_i}$ 即可，因为他们涵盖了 $\varphi(n)$ 所有真因子的倍数。

于是我们可以在 $O(n^{0.25}\log n)$ 的复杂度内得到 n 的最小原根，进而计算所有原根。

```
#include <bits/stdc++.h>
using namespace std;
const int MAXN=1000010;
int t,p,cnt,tot,ctans,fc[MAXN],ans[MAXN],pri[MAXN],rt[MAXN],q[MAXN],phi[MAXN];
void init () {
    phi[1]=1;
    for (int i=2;i<=MAXN-10;i++) {
        if (!q[i]) {pri[++tot]=i,phi[i]=i-1;}
        for (int j=1;j<=tot&&pri[j]*i<=MAXN-10;j++) {
            q[i*pri[j]]=1;
            if (i%pri[j]==0) {
                phi[i*pri[j]]=phi[i]*pri[j];
                break;
            }
            phi[i*pri[j]]=phi[i]*(pri[j]-1);
        }
    }
    rt[2]=rt[4]=1;
    for (int i=2;i<=tot;i++) {
        for (int j=1;(1ll*j*pri[i])<=MAXN-10;j*=pri[i]) {rt[j*pri[i]]=1;}
        for (int j=2;(1ll*j*pri[i])<=MAXN-10;j*=pri[i]) {rt[j*pri[i]]=1;}
    }
}
int gcd (int a,int b) {return (b==0?a:gcd(b,a%b));}
int qpow (int a,int b,int p) {
    int res=1;
    while (b) {
        if (b&1) {res=(1ll*res*a)%p;}
        a=(1ll*a*a)%p;
        b>>=1;
    }
```

```
    return res;
}
void proc (int p) {
    for (int i=2;i*i<=p;i++) {
        if (p%i==0) {
            fc[++cnt]=i;
            while (p%i==0) {p/=i;}
        }
    }
    if (p>1) {fc[++cnt]=p;}
    return;
}
bool chk (int x,int p) {
    if (qpow(x,phi[p],p)!=1) {return 0;}
    for (int i=1;i<=cnt;i++) {
        if (qpow(x,phi[p]/fc[i],p)==1) {return 0;}
    }
    return 1;
}
int findrt (int p) {
    for (int i=1;i<p;i++) {
        if (chk(i,p)) {return i;}
    }
    return 0;
}
void getrt (int p,int x) {
    int prod=1;
    for (int i=1;i<=phi[p];i++) {
        prod=(1ll*prod*x)%p;
        if (gcd(i,phi[p])==1) {
            ans[++ctans]=prod;
        }
    }
}
int main () {
    init();scanf("%d",&t);
    for (int ii=1;ii<=t;ii++) {
        int wtf;
        scanf("%d%d",&p,&wtf);
        if (rt[p]) {
            ctans=cnt=0;
            proc(phi[p]);
            int mn=findrt(p);
            getrt(p,mn);
            sort(ans+1,ans+ctans+1);
            printf("%d\n",ctans);
            for (int i=1;i<=ctans/wtf;i++) {printf("%d ",ans[i*wtf]);}
            printf("\n");
        } else {
            printf("0\n\n");
        }
    }
    return 0;
}
```

反素数

定义g(x)为正整数x的约数个数。如果某个正整数x满足：$\forall 0 < i < x,$，都有$g(x) > g(i)$，则称 x 为**反质数。**

luoguP1463 [POI2001][HAOI2007]反素数

求不超过N的最大反质数。

结论：

①因子个数为$(p_1+1)*(p_2+1)*...*(p_n+1)$，$p_i$为每个因子的个数

②每个因子个数不超过前一个

③最多使用不到10个因子（2e9范围内）

```
#include<bits/stdc++.h>
#define int long long
using namespace std;
const int N=2e5+7;

int a[]={0,2,3,5,7,11,13,17,19,23,29};
int n,res,Ans;

void dfs(int num,int step,int lim,int ans){
    if(step>10) return;
    if(ans>Ans) Ans=ans,res=num;
    if(ans==Ans) res=min(res,num);
    int tmp=1;
    for(int i=1;i<=lim;i++){
        tmp*=a[step];
        if(tmp*num>n) return;
        dfs(num*tmp,step+1,i,ans*(i+1));
    }
}

signed main(){
    cin>>n;
    dfs(1,1,31,1);
    cout<<res<<endl;
    return 0;
}
```

调和级数

Harmonic Number(调和级数+欧拉常数)

求$f(n)=1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+...+\frac{1}{n}(1\le n\le 10^8)$,精确到$10^{-8}$

自今没有一个完全正确的公式，但是欧拉给出了一个近似公式：

$f(n)\approx ln(n)+C+\frac{1}{2n}, n\to\infty$

欧拉常数值：$C\approx 0.57721566490153286060651209$

欧拉常数的表示方法

$\gamma = lim_{n\to\infty}(\sum_{k=1}^{n}\frac{1}{k} - log(n)) = -\int_0^\infty \frac{\log x}{\exp x}$

一些推论

一个整数N的约数个数上界为$2\sqrt{N}$

1~N每个数的约数个数的总和大约为NlogN

第n个质数p(n)的渐近估计$p(N) \approx NlnN$

约瑟夫问题

n个人，每次跳k。求最后一个被淘汰的人的位置

```
int Josephus(int n, int k) {
    if (n == 1) return 0;
    int res = 0;
    if (n < k) {
        For (i, 2, n)
            res = (res + k) % i;
        return res;
    }
    res = Josephus(n - n / k, k);
    if (res < n % k)
        res = res - n % k + n;
    else
        res = res - n % k + (res - n % k) / (k - 1);
    return res;
}
```

参考资料

OI-wiki

《信息学一本通提高篇》——林厚从

https://www.luogu.com.cn/blog/ouuan/solution-p5091