数学

- 细节操作
  - 精度
    - ceil
    - long long
  - 取模取余
- fast
  - 快速幂防炸ll
  - 逆元qpow mod
- 博弈论
  - easy nim1
  - easy nim2(黄金分割比)
  - SG 函数 拆分
  - SG 函数 集合
  - sg函数（群友版
  - 群友改编版
- 数论
  - 质因子cnt
  - gcd
  - 欧拉筛
  - 欧拉降幂
  - 欧拉函数phi jiangly
  - 欧拉函数O(n)
  - Min-25筛
  - exgcd
  - exgcd ax+by=c求解
    - 输入格式
    - 输出格式
  - exgcd通解
  - 逆元递推
  - 中国剩余定理CRT
  - EXCRT
  - 裴蜀定理
    - 题目描述
    - 输入格式
    - 输出格式
    - 输入输出样例
  - 数论分块
- 位操作
  - fast异或
  - lowbit
  - 常用 bit运算
- 组合数学
  - 鸽笼
  - 容斥(集合论)
  - 组合数 mod 小费马
  - 组合数学 杨辉三角
  - 康托展开
  - 斐波那契 O(logn)
  - 卢卡斯数列
  - 范德蒙德卷积

数学

细节操作

精度

ceil

long long

取模取余

```
    int n,x;
    cin>>n;
    cin>>x;
    cout<<x%n;
    cout<<(x%n+n)%n;//负数取模为正
```

fast

快速幂防炸ll

```
ll mul(ll a, ll b, ll p) {
    ll r = a * b - (ll)((long double)a / p * b + 0.5) * p;
    return r < 0 ? r + p : r;
}
ll fpow(ll a, ll b, ll p, ll x = 1) {
    for (; b; b >>= 1, a = mul(a, a, p))
        if (b & 1) x = mul(x, a, p);
    return x;
}
```

逆元qpow mod

```
#include <iostream>
#include <algorithm>
using namespace std;
#define ll long long
const ll N = 2e6+10;
const ll mod =911451407;
ll fac[N], invfac[N];
ll qpow(ll a,ll k,ll p)
{
    ll res=1;
    while(k)
    {
        if(k&1)res=res*a%p;
        a=a*a%p;
        k=k>>1;
    }
    return res;
}

int main()
{
    fac[0] = invfac[0] = 1;
    for(int i=1;i<N;i++)
    {
```

```
        fac[i]=fac[i-1]*i%mod;
        invfac[i]=invfac[i-1]*qpow(i,mod-2,mod)%mod;
    }
    ll n;
    cin>>n;
    while(n--)
    {
        ll x,l;
        cin>>l>>x;
        ll a=x+l-2;
        ll b=x-1;
        cout<<fac[a]*invfac[b]%mod*invfac[a-b]%mod<<"\n";
    }

    return 0;
}
```

博弈论

easy nim1

```
#include<iostream>
using namespace std;
int main()
{
    int t;
    cin>>t;
    while(t--)
    {
        int n;
        cin>>n;
    int ans=0;
    for(int i=1;i<=n;i++)
    {
        int x;
        cin>>x;
        ans=ans^x;
    }
    if(ans==0)cout<<"No"<<endl;
    else      cout<<"Yes"<<endl;
    }

    return 0;
}
```

easy nim2(黄金分割比)

```
#include <bits/stdc++.h>
int main()
{
    int a,b;
    std::cin>>a>>b;
    if(a>b)std::swap(a,b);//a为小的数字
    double temp;
    temp=(double)(b-a)*(double)((sqrt(5.0)+1.0)/2.0);
    int ans;
    ans=(int)temp;
    if(ans==a)std::cout<<0<<std::endl;
    else std::cout<<1<<std::endl;
    return 0;
}
```

SG 函数 拆分

```
#include <bits/stdc++.h>
using namespace std;
const int N=1e5+10;
int vis[N];
int n;
int sg(int x)
{
    if(vis[x]!=-1)return vis[x];//记忆化

    unordered_set<int>st;
    for(int i=0;i<x;i++)
    {
        for(int j=0;j<=i;j++)
        {
            st.insert(sg(i)^sg(j));
        }
    }
    for(int i=0; ;i++)  //mex
    {
        if(!st.count(i))
            return vis[x]=i;
    }
}
int main()
{
    memset(vis,-1,sizeof(vis));
    int n;
    cin>>n;
    int res=0;
    for(int i=1;i<=n;i++)
    {
```

```
        int x;
        cin>>x;
        res=res^sg(x);
    }
    if(res)cout<<"Yes";
    else cout<<"No";
    return 0;
}
```

SG 函数 集合

```
#include <bits/stdc++.h>
using namespace std;
const int N=1e5+10;
int a[N];
int s[N];
int h[N];
int f[N];
int k;int n;
int sg(int x)
{
    if(f[x]!=-1)return f[x];//记忆化

    unordered_set<int>st;
    for(int i=1;i<=k;i++)
    {
        if(x>=s[i])st.insert(sg(x-s[i]));
    }
    for(int i=0; ;i++)  //mex
    {
        if(!st.count(i))
            return f[x]=i;
    }
}
int main()
{
    cin>>k;
    for(int i=1;i<=k;i++)
    {
        cin>>s[i];
    }
    cin>>n;
    memset(f,-1,sizeof(f));
    int res=0;
    for(int i=1;i<=n;i++)
    {
        int x;
        cin>>x;
        res=res^sg(x);
    }
    if(res)cout<<"Yes";
```

```
    else cout<<"No";
    return 0;
}
```

sg函数（群友版

```
void solve(int n, int m)
{
    map<ar(2), int>mp;
    mp[{0, 0}] = 0;
    function<int(int, int)>dfs = [&] (int x, int y)->int
    {
        if (mp.count({ x,y })) return mp[{x,y}];
        unordered_set<int>st;
        if (x >= 1)st.insert(dfs(x - 1, y));
        if (x >= 2)    st.insert(dfs(x - 2, y));
        if (x >= 5)    st.insert(dfs(x - 5, y));
        for (int i = 1;i <= y;i += 1) {
            st.insert(dfs(x, y - i));
        }
        int sg = 0;
        while (st.count(sg)) sg += 1;
        return mp[{x, y}] = sg;
    };
    cout << dfs(n, m) << " ";
}
```

群友改编版

Problem - 1537D - Codeforces

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
constexpr int inf=0x3f3f3f3f;
constexpr int N=2e6+10;
constexpr int M=1e3+10;
const int MAXN=1e6+10;
int prime[MAXN];
bool vis[MAXN];
int cnt=0;
void Euler_prime(int n)
{
    for(int i=2;i<=n;++i)
    {
        if(!vis[i])
        {prime[cnt++]=i;vis[i]=true;}//vis[i]置为true或不置true都可以
        for(int j=0;j<cnt;++j)
        {
            if(i*prime[j]>n)//判断是否越界
```

```
                break;
            vis[i*prime[j]]=true;//筛数
            if(i%prime[j]==0)//时间复杂度为O(n)的关键!
                break;
        }
    }
}
map<int, int>mp;
map<int, int>ppp;
int dfs(int x)
{
    if(mp[x])return mp[x];

    unordered_set<int>st;
    for(int i=2;i<x;i++)
    {
        if(x%i==0)
        {
            st.insert(dfs(x-x/i));
        }
    }
    int sg = 0;
    while (st.count(sg)) sg += 1;
    return mp[x] = sg;
}
void solve()
{
    mp[1]=0;
    for(int i=2;i<=130;i++)
    {
        if(ppp[i])
        {
            mp[i]=0;
            if(i%2==0)
            cout<<0<<" "<<i<<endl;
        }
        else
        {
            int now=dfs(i);
            if(i%2==0)
            cout<<now<<" "<<i<<endl;
        }


    }
}
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);cout.tie(0);
    Euler_prime(1000009);
    for(int i=0;i<=10000;i++)
```

```
    {
        ppp[prime[i]]=1;
    }
    int t;
    // cin>>t;
    t=1;
    while(t--)
    {
      solve();
    }
    return 0;
}
```

数论

质因子cnt

```
#include<bits/stdc++.h>
#define ll long long
#define inf 0x7f7f7f7f
const int N=5e6+10;
using namespace std;

int yz[N];//质因子表，yz[i]表示 i的质因子个数

int main()
{
    int t;
    cin>>t;
    //生成质因子表
    for(int i=2;i<N;i++)
        if(yz[i]==0)
            for(int j=i;j<N;j+=i)
                yz[j]=yz[j/i]+1;

    for(int i=1;i<N;i++)
        yz[i]+=yz[i-1];

    while(t--)
    {
        int ans=0;
        int a,b;
        scanf("%d %d",&a,&b);
        printf("%d\n",yz[a]-yz[b]);//利用了a!/b!的特性
    }
}
```

```
    for (int i = n; i >= 1; i--) {
        for (int j = 2 * i; j <= n; j += i) {
            cnt[j] += cnt[i];
        }
    }
```

gcd

```
#include <iostream>
using namespace std;
int gcd(int a,int b)
{
    if(b==0)
        return a;
    return
        gcd(b,a%b);
}
int main()
{
    int n,m;
    int y;
    cin>>n>>m;
    y=gcd(max(n,m),min(n,m));
    cout<<n*m/y<<endl;
    return 0;
}
```

欧拉筛

```
#include<bits/stdc++.h>
using namespace std;
#define MAXN 1000
int prime[MAXN];
bool vis[MAXN];
int cnt=0;
void Euler_prime(int n)
{
    for(int i=2;i<=n;++i)
    {
        if(!vis[i])
        {prime[cnt++]=i;vis[i]=true;}//vis[i]置为true或不置true都可以
        for(int j=0;j<cnt;++j)
        {
            if(i*prime[j]>n)//判断是否越界
                break;
            vis[i*prime[j]]=true;//筛数
            if(i%prime[j]==0)//时间复杂度为O(n)的关键!
                break;
```

```
        }
    }
}
int main()
{
    int k;
    cin>>k;
    Euler_prime(k);
    for(int i=0;i<=20;i++)
    {
        cout<<prime[i]<<endl;
    }
    return 0;
}
```

欧拉降幂

$$a^b \bmod c = a^{b \bmod \varphi(c) + \varphi(c)} \bmod c$$

$$a^b \equiv \begin{cases} a^{b\%\phi(p)} & gcd(a,p)=1 \\ a^b & gcd(a,p)\neq 1, b<\phi(p) \\ a^{b\%\phi(p)+\phi(p)} & gcd(a,p)\neq 1, b\geq \phi(p) \end{cases} \quad (mod\ p)$$

```
#include<iostream>
#include<cstring>
#include<cmath>
using namespace std;
typedef long long ll;
const int MAX=1000100;
ll  fastPow(ll  a,ll b,ll mod)
{
    ll ans=1;
    a %= mod;
    while(b)
    {
        if(b&1)
        {
            ans = (ans*a)%mod;
        }
        b >>= 1;
```

```
        a = (a*a)%mod;
    }
    return ans;
}
ll  eulerFunction(ll x)
{
    ll eulerNumbers = x;
    for(ll i = 2; i*i <= x; i++)
    {
        if(x % i == 0)
        {
            eulerNumbers = eulerNumbers / i * (i-1);
            while(x % i == 0)
            {
                x /= i;
            }
        }
    }
    if(x > 1)
    {
        eulerNumbers = eulerNumbers / x * (x-1);
    }
    return eulerNumbers;
}
ll eulerDropPow(ll a,char b[],ll c)
{
    ll eulerNumbers = eulerFunction(c);
    ll descendingPower=0;
    for(ll i=0,len = strlen(b); i<len; ++i)
    {
        descendingPower=(descendingPower*10+b[i]-'0') % eulerNumbers;
    }
    descendingPower += eulerNumbers;
    return fastPow(a,descendingPower,c);
}
int main()
{
    ll a,c;
    char b[MAX];
    while(~scanf("%lld%s%lld",&a,b,&c))
    {
        printf("%lld\n",eulerDropPow(a,b,c));
    }
    return 0;
}
```

欧拉函数phi jiangly

1~N 中与N 互质的数的个数叫欧拉函数

```
int phi(int n) {
    int res = n;
    for (int i = 2; i * i <= n; i++) {
        if (n % i == 0) {
            while (n % i == 0) {
                n /= i;
            }
            res = res / i * (i - 1);
        }
    }
    if (n > 1) {
        res = res / n * (n - 1);
    }
    return res;
}
```

欧拉函数O(n)

Solution

首先我们可以根据**扩展欧拉定理**:

$$\text{当 } b \geq \varphi(p) \text{ 时，有 } a^b \equiv a^{b \bmod \varphi(p)+\varphi(p)} \pmod p$$

得到:

$$2^{2^{2\cdots}} \bmod p = 2^{(2^{2\cdots} \bmod \varphi(p)+\varphi(p))} \bmod p$$

很显然这是一个递归式子，边界条件为 $p = 1$，此时式子的值为 0。

对于 $\varphi(p)$，我们可以线性筛预处理。

时间复杂度：$O(P + T \log p)$

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
constexpr int inf=0x3f3f3f3f;
constexpr int N=1e7+10;
constexpr int M=N/10;
int n,tot,prime[M],phi[N];
bool vis[N];
//线性 欧拉函数
void sieve(int n)
{
    phi[1]=1;
    for(int i=2;i<=n;++i)
```

```
    {
        if(!vis[i]) prime[++tot]=i,phi[i]=i-1;
        for(int j=1;j<=tot&&i*prime[j]<=n;++j)
        {
            vis[i*prime[j]]=1;
            if(i%prime[j]==0)
            {
                phi[i*prime[j]]=phi[i]*prime[j];
                break;
            }
            else
            {
                phi[i*prime[j]]=phi[i]*phi[prime[j]];
            }
        }
    }
}
ll qpow(ll x,ll p,ll mod)
{
    ll res=1;
    while(p)
    {
        if(p&1)res=res*x%mod;
        x=x*x%mod;
        p>>=1;
    }
    return res;
}
// 优化快速幂 防炸ll
ll mul(ll a, ll b, ll p) {
    ll r = a * b - (ll)((long double)a / p * b + 0.5) * p;
    return r < 0 ? r + p : r;
}
ll fpow(ll a, ll b, ll p, ll x = 1) {
    for (; b; b >>= 1, a = mul(a, a, p))
        if (b & 1) x = mul(x, a, p);
    return x;
}
//
ll solve(ll p)
{
    if(p==1) return 0;
    return qpow(2,solve(phi[p])+phi[p],p);
}
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);cout.tie(0);
    sieve(N-5);
    int t;
    cin>>t;
    while(t--)
```

```
    {
        int p;
        cin>>p;
        cout<<solve(p)<<"\n";
    }
    return 0;
}
```

Min-25筛

```
/* 「LOJ #6053」简单的函数 */
#include <algorithm>
#include <cmath>
#include <cstdio>

const int maxs = 200000;  // 2sqrt(n)
const int mod = 1000000007;

template <typename x_t, typename y_t>
inline void inc(x_t &x, const y_t &y) {
  x += y;
  (mod <= x) && (x -= mod);
}

template <typename x_t, typename y_t>
inline void dec(x_t &x, const y_t &y) {
  x -= y;
  (x < 0) && (x += mod);
}

template <typename x_t, typename y_t>
inline int sum(const x_t &x, const y_t &y) {
  return x + y < mod ? x + y : (x + y - mod);
}

template <typename x_t, typename y_t>
inline int sub(const x_t &x, const y_t &y) {
  return x < y ? x - y + mod : (x - y);
}

template <typename _Tp>
inline int div2(const _Tp &x) {
  return ((x & 1) ? x + mod : x) >> 1;
}

// 以上目的均为防负数和取模
template <typename _Tp>
inline long long sqrll(const _Tp &x) {  // 平方函数
  return (long long)x * x;
```

```
}

int pri[maxs / 7], lpf[maxs + 1], spri[maxs + 1], pcnt;

inline void sieve(const int &n) {
  for (int i = 2; i <= n; ++i) {
    if (lpf[i] == 0) {  // 记录质数
      lpf[i] = ++pcnt;
      pri[lpf[i]] = i;
      spri[pcnt] = sum(spri[pcnt - 1], i);  // 前缀和
    }
    for (int j = 1, v; j <= lpf[i] && (v = i * pri[j]) <= n; ++j) lpf[v] = j;
  }
}

long long global_n;
int lim;
int le[maxs + 1],  // x <= \sqrt{n}
    ge[maxs + 1];  // x > \sqrt{n}
#define idx(v) (v <= lim ? le[v] : ge[global_n / v])

int G[maxs + 1][2], Fprime[maxs + 1];
long long lis[maxs + 1];
int cnt;

inline void init(const long long &n) {
  for (long long i = 1, j, v; i <= n; i = n / j + 1) {
    j = n / i;
    v = j % mod;
    lis[++cnt] = j;
    (j <= lim ? le[j] : ge[global_n / j]) = cnt;
    G[cnt][0] = sub(v, 1ll);
    G[cnt][1] = div2((long long)(v + 2ll) * (v - 1ll) % mod);
  }
}

inline void calcFprime() {
  for (int k = 1; k <= pcnt; ++k) {
    const int p = pri[k];
    const long long sqrp = sqrll(p);
    for (int i = 1; lis[i] >= sqrp; ++i) {
      const long long v = lis[i] / p;
      const int id = idx(v);
      dec(G[i][0], sub(G[id][0], k - 1));
      dec(G[i][1], (long long)p * sub(G[id][1], spri[k - 1]) % mod);
    }
  }
  /* F_prime = G_1 - G_0 */
  for (int i = 1; i <= cnt; ++i) Fprime[i] = sub(G[i][1], G[i][0]);
}

inline int f_p(const int &p, const int &c) {
```

```
  /* f(p^{c}) = p xor c */
  return p xor c;
}

int F(const int &k, const long long &n) {
  if (n < pri[k] || n <= 1) return 0;
  const int id = idx(n);
  long long ans = Fprime[id] - (spri[k - 1] - (k - 1));
  if (k == 1) ans += 2;
  for (int i = k; i <= pcnt && sqrll(pri[i]) <= n; ++i) {
    long long pw = pri[i], pw2 = sqrll(pw);
    for (int c = 1; pw2 <= n; ++c, pw = pw2, pw2 *= pri[i])
      ans +=
          ((long long)f_p(pri[i], c) * F(i + 1, n / pw) + f_p(pri[i], c + 1)) %
          mod;
  }
  return ans % mod;
}

int main() {
  scanf("%lld", &global_n);
  lim = sqrt(global_n);  // 上限

  sieve(lim + 1000);  // 预处理
  init(global_n);
  calcFprime();
  printf("%lld\n", (F(1, global_n) + 1ll + mod) % mod);

  return 0;
}
```

exgcd

```
#include<bits/stdc++.h>
#define int long long
using namespace std;
int exgcd(int a,int b,int &x,int &y){
    if(!b){
        x=1,y=0;
        return a;
    }
    int d=exgcd(b,a%b,x,y);
    int temp=x;
    x=y;
    y=temp-a/b*y;
    return d;
}
void solve(){
    int a,b;cin>>a>>b;
```

```
    int x,y;
    int d=exgcd(a,b,x,y);
    if(d==1){
        cout<<((x%b)+b)%b<<'\n';
    }
    else cout<<-1<<'\n';//无解时

}
signed main(){
    ios::sync_with_stdio(false);
    cin.tie(0);cout.tie(0);
    int t;cin>>t;
    while(t--)solve();
    return 0;
}
```

exgcd ax+by=c求解

输入格式

第一行一个正整数 T，代表数据组数。

接下来 T 行，每行三个由空格隔开的正整数 a,b,c。

输出格式

T 行。

若该行对应的询问无整数解，一个数字 −1。
若该行对应的询问有整数解但无正整数解，包含 2 个由空格隔开的数字，依次代表整数解中，x 的最小正整数值，*y* 的最小正整数值。
否则包含 5 个由空格隔开的数字，依次代表正整数解的数量，正整数解中，*x* 的最小值，y*的最小值*，*x* 的最大值，*y* 的最大值。

```
#include<bits/stdc++.h>
using namespace std;
long long gcd(long long n,long long m)
{
    return (n%m==0)?m:gcd(m,n%m);
}
void exgcd(long long a,long long b,long long &x,long long &y)
{
    if(!b)
    {
        x=1;
        y=0;
        return;
    }
    long long p;
    exgcd(b,a%b,x,y);
    p=x;
    x=y;
```

```
    y=p-(a/b)*y;
    return;
}
int t;
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);cout.tie(0);
    cin>>t;
    while(t--)
    {
        long long x=0,y=0,a,b,c,g,xin,yin,xax,yax,npa=0,k;
        cin>>a>>b>>c;
        g=gcd(a,b);
        if(c%g!=0)
        cout<<-1<<"\n";
        else
        {
            a/=g;   b/=g;   c/=g;
            exgcd(a,b,x,y);
            x*=c;   y*=c;
            xin=x>0&&x%b!=0?x%b:x%b+b;//若x>0且x%b!=0，x的最小正整数解就等于x%b，否则则需要在x%b的基础上加b。
            yax=(c-xin*a)/b;//求出对应的y的最大正整数解。
            yin=y>0&&y%a!=0?y%a:y%a+a;
            xax=(c-yin*b)/a;//同理于x。
            if(xax>0)//判断有无正整数解。
            npa=(xax-xin)/b+1;//求出正整数解的数量。
            if(!npa)
            cout<<xin<<" "<<yin<<"\n";
            else cout<<npa<<" "<<xin<<" "<<yin<<" "<<xax<<" "<<yax<<"\n";
        }
    }
    return 0;
}
```

exgcd通解

```
#include<bits/stdc++.h>

using namespace std;
typedef long long ll;

const ll inf = 1e18;

void exgcd(ll a, ll b, ll &d, ll &x, ll &y) {
    if (!b) d = a, x = 1, y = 0;
    else exgcd(b, a % b, d, y, x), y -= x * (a / b);
}
```

```
ll wk() {
    ll a, b, c, g, x, y;
    cin >> a >> b >> c;
    exgcd(a, b, g, x, y);
    if (c % g) return -1;
    a /= g, b /= g, c /= g, x *= c, y *= c;

    ll ans = inf;
    // 求次数
    auto wk = [&](ll t) {
        ll r = x + b * t, s = y - a * t;
        if (r >= 0 && s >= 0) ans = min(ans, 2 * (r + s));
        else ans = min(ans, 2 * (abs(r) + abs(s)) - 1);
    };

    // 附近找最小值
    ll t0 = -x / b;
    for (ll t = t0 - 1; t <= t0 + 1; ++t) wk(t);
    t0 = y / a;
    for (ll t = t0 - 1; t <= t0 + 1; ++t) wk(t);
    return ans;
}

int main() {
    int T;
    cin >> T;
    while (T--) cout << wk() << endl;
    return 0;
}
```

逆元递推

```
#include <iostream>
#define ll long long
const ll N=3000010;
ll arr[N];
ll a,p;
int main()
{
  std::cin>>a>>p;
  arr[1]=1;
  for(int i=2;i<=a;i++)
  {
    arr[i]=(p-p/i)*arr[p%i]%p;
  }
  for(int i=1;i<=a;i++)
```

```
  {
      std::cout<<arr[i]<<"\n";
  }
  return 0;
}
```

中国剩余定理CRT

中国剩余定理是一种用于求解诸如

$$\begin{cases} x \equiv a_1 (mod\ m_1) \\ x \equiv a_2 (mod\ m_2) \\ \quad\quad ..... \\ x \equiv a_K (mod\ m_K) \end{cases}$$

形式的同余方程组的定理，其中$m_1, m_2 ... m_k$为**两两互质**的整数，我们的目的，是找x的**最小非负整数**解。

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long
const int N=1e6+10;
int m[N], a[N];
void exgcd(ll a,ll b,ll &x,ll &y)
{
    if(!b)
    {
        x=1,y=0;
        return;
    }
    exgcd(b,a%b,x,y);
    ll t=x;
    x=y,y=t-(a/b)*y;
    return ;
}
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);cout.tie(0);
    int n;
    ll ans=0;
    ll mul=1;
    cin>>n;
    for (int i=1;i<=n;i++){
        cin>>m[i]>>a[i];
        mul=mul*m[i];
    }
    for(int i=1;i<=n;i++){
```

```
        ll s=mul/m[i];
        ll x=0,y=0;
        exgcd(s,m[i],x,y);
        if(x<0)
            x+=m[i];
        ans+=s*x*a[i];
    }
    ans=ans%mul;
    cout<<ans;
    return 0;
}
```

EXCRT

```
#include <bits/stdc++.h>
using namespace std;
typedef __int128 ll;
const int N = 1e5 + 10;
ll x, y, d;
int n;
ll a,b,A,B;
void exgcd(ll &x,ll &y,ll a,ll b)
{
    if(!b)d=a,x=1,y=0;
    else exgcd(y,x,b,a%b),y-=a/b*x;
}
ll gcd(ll a,ll b)
{
    return b?gcd(b,a%b):a;
}
ll lcm(ll a,ll b)
{
    return a/gcd(a,b)*b;
}
void merge()
{
    exgcd(x, y, a, A);
    ll c = B - b;
    if(c % d) puts("-1"), exit(0);
    x = x * c / d % (A / d);
    if(x < 0) x += A / d;
    ll mod = lcm(a, A);
    b = (a * x + b) % mod; if(b < 0) b += mod;
    a = mod;
}

int main()
```

```
{
    scanf("%d",&n);
    for(int i = 1 ; i <= n ; ++ i) {
        long long _A, _B;
        scanf("%lld%lld", &_A, &_B), A = _A, B = _B;
        if(i > 1) merge();
        else a = A, b = B;
    }
    printf("%lld\n", (long long)(b % a));
}
```

L-Three Permutations “范式杯”2023牛客暑期多校训练营1 (nowcoder.com)

```
#include<bits/stdc++.h>

using namespace std;
typedef long long ll;
typedef pair<ll, ll> pll;

const ll inf = 1e18;

void exgcd(ll a, ll b, ll &d, ll &x, ll &y) {
    if (!b) d = a, x = 1, y = 0;
    else exgcd(b, a % b, d, y, x), y -= x * (a / b);
}

pll excrt(pll l, pll r) {
    auto[r1, m1] = l;
    auto[r2, m2] = r;
    if (r1 == -1 || r2 == -1) return {-1, -1};
    ll d, l1, l2;
    exgcd(m1, m2, d, l1, l2);
    if ((r2 - r1) % d) return {-1, -1};
    ll L = m1 * m2 / d;
    ll R = ((r1 + (r2 - r1) / d * l1 % L * m1) % L + L) % L;
    return {R, L};
}

int main() {
    ll n;
    cin >> n;
    vector<ll> a(n + 1), b(n + 1), c(n + 1);
    vector<ll> ia(n + 1), ib(n + 1), ic(n + 1);
    for (int i = 1; i <= n; ++i) cin >> a[i], ia[a[i]] = i;
    for (int i = 1; i <= n; ++i) cin >> b[i], ib[b[i]] = i;
    for (int i = 1; i <= n; ++i) cin >> c[i], ic[c[i]] = i;

    // 计算三个置换
    vector<ll> abc(n + 1), bca(n + 1), cab(n + 1);
    for (int i = 1; i <= n; ++i) abc[i] = a[b[c[i]]];
    for (int i = 1; i <= n; ++i) bca[i] = b[c[a[i]]];
```

```
    for (int i = 1; i <= n; ++i) cab[i] = c[a[b[i]]];
    ll lena = 0, lenb = 0, lenc = 0;
    // 计算到每个点的时间距离+周期
    vector<ll> disa(n + 1, -1), disb(n + 1, -1), disc(n + 1, -1);
    for (ll u = 1; disa[u] == -1; u = abc[u], ++lena) disa[u] = lena;
    for (ll u = 1; disb[u] == -1; u = bca[u], ++lenb) disb[u] = lenb;
    for (ll u = 1; disc[u] == -1; u = cab[u], ++lenc) disc[u] = lenc;

    // EXCRT
    auto solve = [&](ll x, ll y, ll z) -> ll {
        if (disa[x] == -1 || disb[y] == -1 || disc[z] == -1) return inf;
        pll A(disa[x], lena);
        pll B(disb[y], lenb);
        pll C(disc[z], lenc);
        A = excrt(A, excrt(B, C));
        return A.first == -1 ? inf : A.first;
    };

    int T;
    cin >> T;
    while (T--) {
        ll x, y, z;
        cin >> x >> y >> z;
        ll m0 = solve(x, y, z);
        ll m1 = solve(ic[z], ia[x], ib[y]);
        ll m2 = solve(ic[ib[y]], ia[ic[z]], ib[ia[x]]);
        ll ans = min({m0 * 3, m1 * 3 + 1, m2 * 3 + 2});
        printf("%lld\n", ans >= inf ? -1 : ans);
    }
    return 0;
}
```

裴蜀定理

注意:逆元不一定存在，当且仅当 $gcd(a,m)=1$时逆元存在考察方程 $xy+km=1$，如果 $gcd(x,m)\neq 1$，则方程左侧一定是 $gcd(x,m)$的倍数，而右侧不是，等式不成立。这就是表蜀定理。

题目描述

小凯手中有两种面值的金币，两种面值均为正整数且彼此互素。每种金币小凯都有无数个。在不找零的情况下，仅凭这两种金币，有些物品他是无法准确支付的。现在小 凯想知道在无法准确支付的物品中，最贵的价值是多少金币？注意：输入数据保证存在 小凯无法准确支付的商品。

输入格式

两个正整数 a 和 b，它们之间用一个空格隔开，表示小凯中金币的面值。

输出格式

一个正整数 N，表示不找零的情况下，小凯用手中的金币不能准确支付的最贵的物品的价值。

输入输出样例

输入 #1

```
3 7
```

输出 #1

```
11
```

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    long long n,m;
    cin>>n>>m;
    cout<<n*m-n-m;
    return 0;
}
```

设自然数 a、b 和整数 n。a 与 b 互素。考察不定方程：

$$ax+by=n$$

其中 x 和 y 为自然数。如果方程有解，称 n 可以被 a、b 表示。

记 $C=ab-a-b$

由 a 与 b 互素，C 必然为奇数。则有结论：

对任意的整数 n，n与C-n中有且仅有一个可以被表示。

即：可表示的数与不可表示的数在区间 对称（关于 C 的一半对称)。0 可被表示，C 不可被表示；负数不可被表示，大于 C 的数可被表示

数论分块

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
const int inf=0x3f3f3f3f;
```

```
const int N=2e5+10;
const int M=1e3+10;
int a[N];
int n;
ll solve()
{
    cin>>n;
    ll r=0;
    ll m=sqrt(n);
    for(ll i=1;i<=m;i++)
    {
        r+=n/i;
    }
    return (r<<1)-m*m;
    // return ;
}
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);cout.tie(0);
    int t;
    cin>>t;
    while(t--)
    {
        cout<<solve()<<"\n";
    }
    return 0;
}
```

位操作

fast异或

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    long long n;
    cin>>n;
    if(n%4==0)cout<<n;
    if(n%4==1)cout<<1;
    if(n%4==2)cout<<n+1;
    if(n%4==3)cout<<0;
    return 0;
}
```

lowbit

```
#include <bits/stdc++.h>
int lowbit(int x)
{
   return x&(-x);
}
int main()
{
   int a;
   std::cin>>a;
   std::cout<<lowbit(a);
   return 0;
}
```

常用 bit运算

```
一、取出x的第i位：y = ( x>> ( i-1 ) ) & 1

二、将x的第i位取反：x = x ^ ( 1<< ( i-1 ) )

三、将x的第i位变为1：x = x | ( 1<< ( i-1 ) )

四、将x的第i位变成0：x = x & ~（ 1<< ( i-1 ) ）

五、将x最靠右的1变成0：x = x & (x-1)

六、取出最靠右的1：y=x&(-x)

七、把最靠右的0变成1：x | = (x-1)
```

组合数学

鸽笼

```

```

容斥(集合论)

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
const ll inf=0x3f3f3f3f;
const ll N=1e5+1;
ll a[N];
void solve()
```

```
{
    return ;
}
int main()
{
    ios::sync_with_stdio(false);
    ll n,m;
    cin>>n>>m;
    for(ll i=0;i<m;i++)
    {
        cin>>a[i];
    }
    ll ans=0;
    for(ll i=1;i<(1<<m);i++)
    {
        ll temp=1;ll cnt=0;
        for(ll j=0;j<m;j++)
        {
            if((i>>j)&1)
            {
                if(temp*a[j]>n)
                {
                    temp=-1;
                    break;
                }
                cnt++;
                temp=temp*a[j];
            }

        }
        if(temp==-1)continue;
        if(cnt%2)ans+=n/temp;
        else ans-=n/temp;
    }
    cout<<ans;
    return 0;
}
```

组合数 mod 小费马

```
#include <iostream>
#include <algorithm>
using namespace std;
#define ll long long
const ll N = 2e6+10;
const ll mod =911451407;
ll fac[N], invfac[N];
ll qpow(ll a,ll k,ll p)
{
    ll res=1;
    while(k)
```

```
    {
        if(k&1)res=res*a%p;
        a=a*a%p;
        k=k>>1;
    }
    return res;
}

int main()
{
    fac[0] = invfac[0] = 1;
    for(int i=1;i<N;i++)
    {
        fac[i]=fac[i-1]*i%mod;
        invfac[i]=invfac[i-1]*qpow(i,mod-2,mod)%mod;
    }
    ll n;
    cin>>n;
    while(n--)
    {
        ll x,l;
        cin>>l>>x;
        ll a=x+l-2;
        ll b=x-1;
        cout<<fac[a]*invfac[b]%mod*invfac[a-b]%mod<<"\n";
    }

    return 0;
}
```

组合数学 杨辉三角

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
const ll N = 1010;
ll a[N];
const ll MOD =1e9+7;
ll comb[N][N];//comb[n][m]就是C(n,m)，n中选m个数
void solve()
{
    for(ll i=0;i<N;i++)
    {
        comb[i][0]=comb[i][i] = 1;
        for(ll j=1;j<i;j++)
        {
            comb[i][j]=comb[i-1][j]+comb[i-1][j-1];
            comb[i][j]%=MOD;
        }
    }
}
```

```
int main()
{
    ll n;
    cin>>n;
    for(ll i=1;i<=n/2;i++)
    {
        a[i]=i*2-1;a[n-i+1]=i*2;
    }
    if(n&1)a[n/2+1]=n;
    solve();
    ll sum=0;
    for(ll i=1;i<=n;i++)
    {
        sum=sum+(comb[n-1][i-1]*a[i]%MOD)%MOD;
    }
    cout<<sum%MOD<<endl;
    for(ll i=1;i<=n;i++)
    {
        cout<<a[i]<<" ";
    }
    return 0;
}
```

康托展开

```
/***** 这里以字符串进行展示  字符串可泛化性好 ******/

#include<iostream>
#include<cstdio>
#include<vector>
#include<algorithm>
using namespace std;

/*******打出1-n的阶乘表*******/
int f[20];
int x, num;

void jie_cheng(int n)
{
    f[0] = f[1] = 1; // 0的阶乘为1
    for(int i = 2; i <= n; i++) f[i] = f[i - 1] * i;
}

/**************康托逆展开**************/

vector<char> vec; //存需要排列的字符
void rev_kangtuo(int k) //输出序号为 k 的字符序列
{
    int n = vec.size(), len = 0;
    string ans = "";
    k--; // 算的时候是按 12345 是第0位
```

```
    for(int i = 1; i <= n; i++){
        int t = k / f[n - i]; // 第 i 位需要 第 t + 1 大的数
        k %= f[n - i];        //剩下的几位需要提供的排列数
        ans += vec[t] ; //  vec[t] 就是第 t + 1 大的数
        vec.erase(vec.begin() + t);
//用过就删了，不用vector用暴力也可以，就是说枚举，然后一个一个的比较大小，并记录有几个没用过的字符且字典序比它小
    }
    cout << ans << '\n';
}

/***************************************/
// 假设展开后不超过10位
int main()
{
    jie_cheng(10); // 预处里好阶乘
    scanf("%d", &x); // 输入需要逆展开的数字
    /************康托逆展开***********/
    for(int i = 1; i <= 10; i++)
    {
        if(x / f[i] == 0) // 求出 x 逆展开所需的最小的位数，方便下面的初始化
        {
            num = i;
            break;
        }
    }
    for(int i = 1; i <= num; i++) vec.push_back(i + '0'); //输入的位数只要不小于num就可以
    rev_kangtuo(x);
    return 0;
}
```

斐波那契 O(logn)

$1 \quad 1 \quad 2 \quad 3 \quad 5 \quad 8 \quad 13 \quad 21$

$$F_n = \frac{(\frac{1+\sqrt{5}}{2})^n - (\frac{1-\sqrt{5}}{2})^n}{\sqrt{5}}$$

卢卡斯数列

$1 \quad 3 \quad 4 \quad 7 \quad 11 \quad 18 \quad 29 \quad 47$

$$L_n^2 - 5F_n^2 = -4$$

$$L_n = (\frac{1+\sqrt{5}}{2})^n + (\frac{1-\sqrt{5}}{2})^n$$

任意两个或两个以上斐波那契—卢卡斯数列之和或差仍然是斐波那契—卢卡斯数列

范德蒙德卷积

范德蒙德卷积公式

$$\sum_{i=0}^{k}\binom{n}{i}\binom{m}{k-i}=\binom{n+m}{k}$$

证明

考虑用二项式定理证明：

$$\begin{aligned}\sum_{k=0}^{n+m}\binom{n+m}{k}x^k&=(x+1)^{n+m}\\&=(x+1)^n(x+1)^m\\&=\sum_{r=0}^{n}\binom{n}{r}x^r\sum_{s=0}^{m}\binom{m}{s}x^s\\&=\sum_{k=0}^{n+m}\sum_{r=0}^{k}\binom{n}{r}\binom{m}{k-r}x^k\end{aligned}$$

即有：

$$\binom{n+m}{k}=\sum_{r=0}^{k}\binom{n}{r}\binom{m}{k-r}$$

推论 1 及证明

$$\sum_{i=-r}^{s}\binom{n}{r+i}\binom{m}{s-i}=\binom{n+m}{r+s}$$

证明与原公式证明相似。

推论 2 及证明

$$\sum_{i=1}^{n}\binom{n}{i}\binom{n}{i-1}=\binom{2n}{n-1}$$

根据基础的组合数学知识推导，有：

$$\sum_{i=1}^{n}\binom{n}{i}\binom{n}{i-1}=\sum_{i=0}^{n-1}\binom{n}{i+1}\binom{n}{i}=\sum_{i=0}^{n-1}\binom{n}{n-1-i}\binom{n}{i}=\binom{2n}{n-1}$$

推论 3 及证明

$$\sum_{i=0}^{n}\binom{n}{i}^2=\binom{2n}{n}$$

根据基础的组合数学知识推导，有：

$$\sum_{i=0}^{n}\binom{n}{i}^2=\sum_{i=0}^{n}\binom{n}{i}\binom{n}{n-i}=\binom{2n}{n}$$

推论 4 及证明

$$\sum_{i=0}^{m}\binom{n}{i}\binom{m}{i}=\binom{n+m}{m}$$

根据基础的组合数学知识推导，有：

$$\sum_{i=0}^{m}\binom{n}{i}\binom{m}{i}=\sum_{i=0}^{m}\binom{n}{i}\binom{m}{m-i}=\binom{n+m}{m}$$

卡特兰数

H_0	H_1	H_2	H_3	H_4	H_5	H_6	...
1	1	2	5	14	42	132	...

(Catalan 数列)

递推式 ¶

该递推关系的解为:

$$H_n = \frac{\binom{2n}{n}}{n+1} (n \geq 2, n \in \mathbf{N}_+)$$

关于 Catalan 数的常见公式:

$$H_n = \begin{cases} \sum_{i=1}^{n} H_{i-1} H_{n-i} & n \geq 2, n \in \mathbf{N}_+ \\ 1 & n = 0, 1 \end{cases}$$

$$H_n = \frac{H_{n-1}(4n-2)}{n+1}$$

$$H_n = \binom{2n}{n} - \binom{2n}{n-1}$$

$$\begin{aligned} H(x) &= \frac{1 - \sqrt{1-4x}}{2x} \\ &= \frac{1}{2x} \sum_{n \geq 1} \binom{2n-1}{n} \frac{1}{(2n-1)} 2x^n \\ &= \sum_{n \geq 1} \binom{2n-1}{n} \frac{1}{(2n-1)} x^{n-1} \\ &= \sum_{n \geq 0} \binom{2n+1}{n+1} \frac{1}{(2n+1)} x^n \\ &= \sum_{n \geq 0} \binom{2n}{n} \frac{1}{n+1} x^n \end{aligned}$$

```
#include <iostream>
using namespace std;
int n;
long long f[25];

int main() {
  f[0] = 1;
```

```
  cin >> n;
  for (int i = 1; i <= n; i++) f[i] = f[i - 1] * (4 * i - 2) / (i + 1);
  // 这里用的是常见公式2
  cout << f[n] << endl;
  return 0;
}
```

$$f(n) = C_{2n}^{n} - C_{2n}^{n-1}$$

```
#include<iostream>
#include<cstring>
#include<cstdio>
using namespace std;
long long cat[50];
long long tatal;
void Cat4(int n)
{
    tatal=1;
    for(int i=0;i<n;++i)//求c(2n,n);
        tatal=tatal*(2*n-i)/(i+1);
    tatal=tatal-tatal*n/(n+1);
}
int main()
{
    int n;
    scanf("%d",&n);
    Cat4(n);
    printf("%lld\n",tatal);
    return 0;
}
```

数值积分

普通辛普森法

```
const int N = 1000 * 1000;

double simpson_integration(double a, double b) {
  double h = (b - a) / N;
  double s = f(a) + f(b);
  for (int i = 1; i <= N - 1; ++i) {
    double x = a + h * i;
    s += f(x) * ((i & 1) ? 4 : 2);
  }
  s *= h / 3;
  return s;
}
```

自适应辛普森法

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long
double a,b,c,d;
double l,r;
double f(double x)
{
    return (c*x+d)/(a*x+b);
}
double simpson(double l, double r) {
    double mid=(l+r)/2;
    return (r-l)*(f(l)+4*f(mid)+f(r))/6;  // 辛普森公式
}

double asr(double l, double r, double eps, double ans, int step) {
    double mid=(l+r)/2;
    double fl=simpson(l, mid),fr=simpson(mid,r);

    if(abs(fl+fr-ans)<=15*eps&&step<0)
    return fl+fr+(fl+fr-ans)/15;  // 足够相似的话就直接返回

    return
        asr(l,mid,eps/2,fl,step-1)+
        asr(mid,r,eps/2,fr,step-1);  // 否则分割成两段递归求解
}

double calc(double l,double r,double eps) {
  return asr(l,r,eps,simpson(l, r),12);
}
void solve()
{
    cin>>a>>b>>c>>d;
    cin>>l>>r;
    printf("%.6lf",calc(l,r,1e-6));
}
int main()
```

```
{
    ios::sync_with_stdio(false);
    cin.tie(0);cout.tie(0);
    solve();
    return 0;
}
```

假如有一段图像已经很接近二次函数的话，直接带入公式求积分，得到的值精度就很高了，不需要再继续分割这一段了

线性代数

矩阵快速幂

YYF是一个英勇的侦查员。现在他正在执行打入到敌方内部的危险任务。在解决了一系列的险情后，YYF到达了敌方著名的"地雷路"起始点。这条路非常长，上面被精心排布了不少地雷。一开始，YYF站在1的位置。对于后面的路程，YYF有p的概率向前走一步，或者有1-p的概率向前跳两步。现在问题来了。非常喜欢坑队友的情报部得到了每个地雷的位置，但他们不准备告诉YYF，反而请你计算YYF能安全走过整条“地雷路”的概率。

```
#include<bits/stdc++.h>
using namespace std;
int n,a[11];
double ans,p;
struct data
{
    double a[2][2];
}t,none;
data operator *(data a,data b)
{
    data c=none;
    for(int i=0;i<=1;i++)
    {
        for(int j=0;j<=1;j++)
        {
            for(int k=0;k<=1;k++)
            {
                c.a[i][k]+=a.a[i][j]*b.a[j][k];
            }
        }
    }
    return c;
}
data poww(data a,int x)//矩阵快速幂
{
    data sum=none,num=a;
    sum.a[1][1]=sum.a[0][0]=1.0;
    while(x)
    {
        if(x&1)
        {
```

```
            sum=(sum*num);
        }
        x/=2;
        num=(num*num);
    }
    return sum;
}
int main()
{
    while(scanf("%d%lf",&n,&p)!=EOF)
    {
        ans=1.0;
        for(int i=1;i<=n;i++)
        {
            scanf("%d",&a[i]);
        }
        sort(a+1,a+n+1);
        t.a[0][0]=p;//初始矩阵
        t.a[0][1]=1-p;
        t.a[1][0]=1.0;
        t.a[1][1]=0.0;
        for(int i=1;i<=n;i++)
        {
            data tmp;
            if(i==1)
            {
                tmp=poww(t,a[i]-1);
            }else{
                tmp=poww(t,a[i]-a[i-1]-1);
            }
            ans=(ans*(1-tmp.a[0][0]));//乘上不踩雷的概率
        }
        printf("%.7lf\n",ans);
    }
    return 0;
}
/*
1 0.5
2
2 0.5
2 4
*/
```

斐波那契快速幂

```
const int mod = 1000000007;

struct Matrix {
  int a[3][3];

  Matrix() { memset(a, 0, sizeof a); }
```

```
  Matrix operator*(const Matrix &b) const {
    Matrix res;
    for (int i = 1; i <= 2; ++i)
      for (int j = 1; j <= 2; ++j)
        for (int k = 1; k <= 2; ++k)
          res.a[i][j] = (res.a[i][j] + a[i][k] * b.a[k][j]) % mod;
    return res;
  }
} ans, base;

void init() {
  base.a[1][1] = base.a[1][2] = base.a[2][1] = 1;
  ans.a[1][1] = ans.a[1][2] = 1;
}

void qpow(int b) {
  while (b) {
    if (b & 1) ans = ans * base;
    base = base * base;
    b >>= 1;
  }
}

int main() {
  int n = read();
  if (n <= 2) return puts("1"), 0;
  init();
  qpow(n - 2);
  println(ans.a[1][1] % mod);
}
```

高斯消元

行列式计算

```
const double EPS = 1E-9;
int n;
vector<vector<double> > a(n, vector<double>(n));

double det = 1;
for (int i = 0; i < n; ++i) {
  int k = i;
  for (int j = i + 1; j < n; ++j)
    if (abs(a[j][i]) > abs(a[k][i])) k = j;
  if (abs(a[k][i]) < EPS) {
    det = 0;
    break;
```

```
  }
  swap(a[i], a[k]);
  if (i != k) det = -det;
  det *= a[i][i];
  for (int j = i + 1; j < n; ++j) a[i][j] /= a[i][i];
  for (int j = 0; j < n; ++j)
    if (j != i && abs(a[j][i]) > EPS)
      for (int k = i + 1; k < n; ++k) a[j][k] -= a[i][k] * a[j][i];
}

cout << det;
```

高斯消元法解异或方程组

```
std::bitset<1010> matrix[2010];  // matrix[1~n]：增广矩阵，0 位置为常数

std::vector<bool> GaussElimination(
    int n, int m)  // n 为未知数个数，m 为方程个数，返回方程组的解（多解 /
                   // 无解返回一个空的 vector）
{
  for (int i = 1; i <= n; i++) {
    int cur = i;
    while (cur <= m && !matrix[cur].test(i)) cur++;
    if (cur > m) return std::vector<bool>(0);
    if (cur != i) swap(matrix[cur], matrix[i]);
    for (int j = 1; j <= m; j++)
      if (i != j && matrix[j].test(i)) matrix[j] ^= matrix[i];
  }
  std::vector<bool> ans(n + 1, 0);
  for (int i = 1; i <= n; i++) ans[i] = matrix[i].test(0);
  return ans;
}
```

线性基

异或最小值、最大值、第k大、rank

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
ll p[64];
ll d[64];
ll cnt=0;
bool zero=false;
//插入线性基
void ins(ll x)
{
```

```
    if(x==0)zero=true;

    for(ll i=62;i>=0;i--)
    {
        if(x&(1ll<<i))
        {
            if(!p[i]){ p[i]=x,cnt++; break; }
            else x^=p[i];
        }
        if(x==0)zero=true;
    }
}
//ending

//查询一个元素是否可以被异或出来
ll ask(ll x) {
    for(ll i=62;i>=0;i--)
    {
        if(x&(1ll<<i)) x^=p[i];
    }
    return x==0;
}
//ending

//查询异或最大值
ll askmx()
{
    ll ans=0;
    for(ll i=62;i>=0;i--)
        if((ans^p[i])>ans) ans^=p[i];
    return ans;
}
//ending
//查询异或最小值
ll askmn() {
    if(zero) return 0;
    for(ll i=0;i<=62;i++)
    {
        if(p[i])return p[i];
    }
    return 0;
}
//ending

//重建
void rebuild() {
    cnt=0;
    for(ll i=62;i>=0;i--)
        for(ll j=i-1;j>=0;j--)
            if(p[i]&(1ll<<j)) p[i]^=p[j];
    for(ll i=0;i<=62;i++) if(p[i]) d[cnt++]=p[i];
}
```

```
//查询异或第k小
ll kth(ll k) {
    k=k-zero;
    if(k>=(1ll<<cnt)) return -1;
    ll ans=0;
    for(ll i=62;i>=0;i--)
        if(k&(1ll<<i)) ans^=d[i];
    return ans ;
}
//查询排名
ll rrank(ll x) {
      ll ans = 0;
      for(ll i = cnt - 1; i >= 0; i --)
                  if(x >= d[i]) ans += (1 << i), x ^= d[i];
      return ans + zero;
}
//因为我们并没有考虑0的情况，所以还要去考虑一下0的情况，特判即可。

//ending
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);cout.tie(0);
    int n;
    cin>>n;
    for(int i=1;i<=n;i++)
    {
        ll x;
        cin>>x;
        ins(x);
    }
    rebuild();
    cout<<askmx()<<"\n";
    cout<<askmn()<<"\n";
    cout<<rrank(14)<<"\n";
    cout<<kth(8)<<"\n";
    return 0;
}
```

贪心法

```
#include <bits/stdc++.h>
using namespace std;
#define ll unsigned long long
ll p[64];
void insert(ll x)
{
    for(int i=63;~i;i--)
    {
```

```
        if(!(x>>i))
        {
            continue;
        }
        if(!p[i])
        {
            p[i]=x;
            break;
        }
        x^=p[i];
    }
}
int main(){
    int n;
    cin>>n;
    ll a;
    for(int i=1;i<=n;i++)
    {
        cin>>a;
        insert(a);
    }
    ll ans=0;
    for(int i=63;~i;i--)
    {
        ans=max(ans,ans^p[i]);
    }
    cout<<ans;
    return 0;
}
```

高斯消元法

```
#include <bits/stdc++.h>
using ull = unsigned long long;
const int MAXN = 1e5 + 5;

inline ull deg(ull num, int deg) { return num & (1ull << deg); }

ull a[MAXN];

int main() {
  int n;
  scanf("%d", &n);
  for (int i = 1; i <= n; ++i) scanf("%llu", &a[i]);
  int row = 1;
  for (int col = 63; ~col && row <= n; --col) {
    for (int i = row; i <= n; ++i) {
      if (deg(a[i], col)) {
        std::swap(a[row], a[i]);
        break;
```

```
      }
    }
    if (!deg(a[row], col)) continue;
    for (int i = 1; i <= n; ++i) {
      if (i == row) continue;
      if (deg(a[i], col)) {
        a[i] ^= a[row];
      }
    }
    ++row;
  }
  ull ans = 0;
  for (int i = 1; i < row; ++i) {
    ans ^= a[i];
  }
  printf("%llu\n", ans);
  return 0;
}
```

高数

泰勒展开

如果函数$f(x)$在x_0处有n阶导数，那么存在x_0的一个邻域，对于该邻域内的任一x，有

$$f(x)=f(x_0)+f^{'}(x_0)(x-x_0)+\frac{f^{''(x_0)}}{2!}(x-x_0)^2+\cdots+\frac{f^{(n)}(x_0)}{n!}(x-x_0)^n+R_n(x) \quad (1\text{-}1)$$

其中

$$R_n(x)=o((x-x_0)^n) \quad (1\text{-}2)$$

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + o(x^3)$$

$$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} + o(x^3)$$

$$(1+x)^\alpha = 1 + \alpha x + \frac{\alpha(\alpha-1)}{2!}x^2 + \frac{\alpha(\alpha-1)(\alpha-2)}{3!}x^3 + o(x^3)$$

$$\frac{1}{1-x} = 1 + x + x^2 + x^3 + o(x^3)$$

$$\frac{1}{1+x} = 1 - x + x^2 - x^3 + o(x^3)$$

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} + o(x^5)$$

$$\arcsin x = x + \frac{1}{2} \times \frac{x^3}{3} + \frac{1 \times 3}{2 \times 4} \times \frac{x^5}{5} + \frac{1 \times 3 \times 5}{2 \times 4 \times 6} \times \frac{x^7}{7} + o(x^7)$$

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} + o(x^4)$$

$$\sqrt{1+x} = 1 + \frac{x}{2} - \frac{x^2}{8} + \frac{x^3}{16} + o(x^3)$$

$$\tan x = x + \frac{x^3}{3} + \frac{2x^5}{15} + o(x^5)$$

$$\arctan x = x - \frac{x^3}{3} + \frac{x^5}{5} + o(x^5)$$

$$e^x = 1 + \frac{1}{1!}x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + o\left(x^3\right)$$

$$\ln(1+x) = x - \frac{1}{2}x^2 + \frac{1}{3}x^3 + o\left(x^3\right)$$

$$\sin x = x - \frac{1}{3!}x^3 + \frac{1}{5!}x^5 + o\left(x^5\right)$$

$$\arcsin x = x + \frac{1}{2} \times \frac{x^3}{3} + \frac{1 \times 3}{2 \times 4} \times \frac{x^5}{5} + \frac{1 \times 3 \times 5}{2 \times 4 \times 6} \times \frac{x^7}{7} + o\left(x^7\right)$$

$$\cos x = 1 - \frac{1}{2!}x^2 + \frac{1}{4!}x^4 + o\left(x^4\right)$$

$$\frac{1}{1-x} = 1 + x + x^2 + x^3 + o\left(x^3\right)$$

$$(1+x)^a = 1 + \frac{a}{1!}x + \frac{a(a-1)}{2!}x^2 + \frac{a(a-1)(a-2)}{3!}x^3 + o\left(x^3\right)$$

$$e^x = 1 + x + \frac{x^2}{2!} + \cdots + \frac{x^n}{n!} + o(x^n)$$

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots + (-1)^{k-1}\frac{x^{2k-1}}{(2k-1)!} + o(x^{2k-1})$$

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \cdots + (-1)^k \frac{x^{2k}}{(2k)!} + o(x^{2k})$$

$$\frac{1}{1+x} = 1 - x + x^2 - \cdots + (-1)^n x^n$$

$$\ln(1+x) = x - \frac{x^2}{2} + \cdots + (-1)^{n-1}\frac{x^n}{n} + o(x^n)$$

二、常用函数的泰勒展开式及其记忆技巧

由泰勒公式（麦克劳林公式）可得如下常用函数的展开式：

$$\frac{1}{1-x}=1+x+x^2+\cdots+x^n+o(x^n) \quad \cdots(1)$$

$$e^x=1+x+\frac{1}{2!}x^2+\cdots+\frac{1}{n!}x^n+o(x^n) \quad \cdots(2)$$

$$\sin x=x-\frac{1}{3!}x^3+\frac{1}{5!}x^5+\cdots+(-1)^n\frac{1}{(2n+1)!}x^{2n+1}+o(x^{2n+1}) \quad \cdots(3)$$

$$\cos x=1-\frac{1}{2!}x^2+\frac{1}{4!}x^4+\cdots+(-1)^n\frac{1}{(2n)!}x^{2n}+o(x^{2n}) \quad \cdots(4)$$


```
Series[Tan[x], {x, 0, 15}]
```

$$x+\frac{x^3}{3}+\frac{2x^5}{15}+\frac{17x^7}{315}+\frac{62x^9}{2835}+\frac{1382x^{11}}{155\,925}+\frac{21\,844x^{13}}{6\,081\,075}+\frac{929\,569x^{15}}{638\,512\,875}+O[x]^{16}$$

$$e^x=\sum_{n=0}^{\infty}\frac{x^n}{n!} \quad \forall x$$

$$\ln(1+x)=\sum_{n=1}^{\infty}\frac{(-1)^{n+1}}{n}x^n \quad \forall x\in(-1,1]$$

- 三角函数：

$$\sin x=\sum_{n=0}^{\infty}\frac{(-1)^n}{(2n+1)!}x^{2n+1} \quad \forall x$$

$$\cos x=\sum_{n=0}^{\infty}\frac{(-1)^n}{(2n)!}x^{2n} \quad \forall x$$

$$\tan x=\sum_{n=1}^{\infty}\frac{B_{2n}(-4)^n(1-4^n)}{(2n)!}x^{2n-1} \quad \forall x:|x|<\frac{\pi}{2}$$

$$\sec x=\sum_{n=0}^{\infty}\frac{(-1)^nE_{2n}}{(2n)!}x^{2n} \quad \forall x:|x|<\frac{\pi}{2}$$

$$\arcsin x=\sum_{n=0}^{\infty}\frac{(2n)!}{4^n(n!)^2(2n+1)}x^{2n+1} \quad \forall x:|x|<1$$

$$\arctan x=\sum_{n=0}^{\infty}\frac{(-1)^n}{2n+1}x^{2n+1} \quad \forall x:|x|<1$$

$$\arctan x=\frac{\pi\operatorname{sgn}x}{2}-\frac{1}{x}+\sum_{k=1}^{\infty}\frac{(-1)^k}{(2k+1)\,x^{2k+1}} \quad \forall x:|x|>1$$

$$e^x = 1+\frac{x}{1!}+\frac{x^2}{2!}+\frac{x^3}{3!}+\cdots \qquad -\infty<x<\infty$$

$$e^{-x^2} = 1-x^2+\frac{x^4}{2!}-\frac{x^6}{3!}+\frac{x^8}{4!}-\cdots \qquad -\infty<x<\infty$$

$$a^x = e^{x\ln a} = 1+\frac{x\ln a}{1!}+\frac{(x\ln a)^2}{2!}+\frac{(x\ln a)^3}{3!}+\cdots \qquad -\infty<x<\infty$$

$$e^{\sin x} = 1+x+\frac{x^2}{2}-\frac{x^4}{8}-\frac{x^5}{15}+\cdots \qquad -\infty<x<\infty$$

$$e^{\cos x} = e\left(1-\frac{x^2}{2}+\frac{x^4}{6}-\frac{31x^6}{720}+\cdots\right) \qquad -\infty<x<\infty$$

$$e^{\tan x} = 1+x+\frac{x^2}{2}+\frac{x^3}{2}+\frac{3x^4}{8}+\cdots \qquad |x|<\frac{\pi}{2}$$

$$e^x\sin x = x+x^2+\frac{x^3}{3}-\frac{x^5}{30}-\frac{x^6}{90}+\cdots+\frac{\left(\sqrt{2}\right)^n\sin\left(\frac{n\pi}{4}\right)x^n}{n!}+\cdots \qquad -\infty<x<\infty$$

$$e^x\cos x = 1+x-\frac{x^3}{3}-\frac{x^4}{6}+\cdots+\frac{\left(\sqrt{2}\right)^n\cos\left(\frac{n\pi}{4}\right)x^n}{n!}+\cdots \qquad -\infty<x<\infty$$

三角函数泰勒级数展开式(公式)

$$\sin x = x-\frac{x^3}{3!}+\frac{x^5}{5!}-\frac{x^7}{7!}+\cdots \qquad -\infty<x<\infty$$

$$\cos x = 1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}+\cdots \qquad -\infty<x<\infty$$

$$\tan x = x+\frac{x^3}{3}+\frac{2x^5}{15}+\frac{17x^7}{315}+\frac{62x^9}{2835}\cdots+\frac{2^{2n}\left(2^{2n}-1\right)B_n x^{2n-1}}{(2n)!}+\cdots \qquad |x|<\frac{\pi}{2}$$

$$\sec x = 1+\frac{x^2}{2}+\frac{5x^4}{24}+\frac{61x^6}{720}+\cdots+\frac{E_n x^{2n}}{(2n)!}+\cdots \qquad |x|<\frac{\pi}{2}$$

$$\csc x = \frac{1}{x}+\frac{x}{6}+\frac{7x^3}{360}+\frac{31x^5}{15120}+\cdots+\frac{2\left(2^{2n-1}-1\right)B_n x^{2n-1}}{(2n)!}+\cdots \qquad 0<|x|<\pi$$

$$\cot x = \frac{1}{x}-\frac{x}{3}-\frac{x^3}{45}-\frac{2x^5}{945}-\cdots-\frac{2^{2n}B_n x^{2n-1}}{(2n)!}-\cdots \qquad 0<|x|<\pi$$

反三角函数泰勒级数展开式

$$\sin^{-1} x = x + \frac{1}{2}\frac{x^3}{3} + \frac{1\cdot 3}{2\cdot 4}\frac{x^5}{5} + \frac{1\cdot 3\cdot 5}{2\cdot 4\cdot 6}\frac{x^7}{7} + \cdots \quad |x| < 1$$

$$\cos^{-1} x = \frac{\pi}{2} - \sin^{-1} x$$

$$= \frac{\pi}{2} - \left(x + \frac{1}{2}\frac{x^3}{3} + \frac{1\cdot 3}{2\cdot 4}\frac{x^5}{5} + \frac{1\cdot 3\cdot 5}{2\cdot 4\cdot 6}\frac{x^7}{7} + \cdots \right) \quad |x| < 1$$

$$\tan^{-1} x = \begin{cases} x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \cdots & |x| < 1 \\ \pm\frac{\pi}{2} - \frac{1}{x} + \frac{1}{3x^3} - \frac{1}{5x^5} + \cdots & \begin{cases} + \text{if } x \geq 1 \\ - \text{if } x \leq -1 \end{cases} \end{cases}$$

$$\sec^{-1} x = \cos^{-1}\left(\frac{1}{x}\right)$$

$$= \frac{\pi}{2} - \left(\frac{1}{x} + \frac{1}{2\cdot 3x^3} + \frac{1\cdot 3}{2\cdot 4\cdot 5x^5} + \frac{1\cdot 3\cdot 5}{2\cdot 4\cdot 6\cdot 7x^7} + \cdots \right) \quad |x| > 1$$

$$\csc^{-1} x = \sin^{-1}(1/x)$$

$$= \frac{1}{x} + \frac{1}{2\cdot 3x^3} + \frac{1\cdot 3}{2\cdot 4\cdot 5x^5} + \frac{1\cdot 3\cdot 5}{2\cdot 4\cdot 6\cdot 7x^7} + \cdots \quad |x| > 1$$

$$\cot^{-1} x = \frac{\pi}{2} - \tan^{-1} x$$

$$= \begin{cases} \frac{\pi}{2} - \left(x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \cdots \right) & |x| < 1 \\ p\pi + \frac{1}{x} - \frac{1}{3x^3} + \frac{1}{5x^5} + \cdots & \begin{cases} p = 0 \text{ if } x \geq 1 \\ p = 1 \text{ if } x \leq -1 \end{cases} \end{cases}$$

对数函数泰勒级数展开式

$$\ln x=(x-1)-\frac{(x-1)^2}{2}+\frac{(x-1)^3}{3}-\cdots \qquad 0<x\le 2$$

$$\ln x=2\left[\left(\frac{x-1}{x+1}\right)+\frac{1}{3}\left(\frac{x-1}{x+1}\right)^3+\frac{1}{5}\left(\frac{x-1}{x+1}\right)^5+\cdots\right] \qquad x>0$$

$$\ln x=\left(\frac{x-1}{x}\right)+\frac{1}{2}\left(\frac{x-1}{x}\right)^2+\frac{1}{3}\left(\frac{x-1}{x}\right)^3+\cdots \qquad x\ge\frac{1}{2}$$

$$\ln(1+x)=x-\frac{x^2}{2}+\frac{x^3}{3}-\frac{x^4}{4}+\frac{x^5}{5}-\cdots \qquad -1<x\le 1$$

$$\ln(a+x)=\ln a+2\left[\frac{x}{2a+x}+\frac{1}{3}\left(\frac{x}{2a+x}\right)^3+\frac{1}{5}\left(\frac{x}{2a+x}\right)^5+\cdots\right] \qquad a>0,\ -a<x$$

$$\ln\left(\frac{x+1}{x}\right)=2\left[\frac{1}{2x+1}+\frac{1}{3}\left(\frac{1}{2x+1}\right)^3+\frac{1}{5}\left(\frac{1}{2x+1}\right)^5+\cdots\right] \qquad x>0$$

$$\ln\left(\frac{1+x}{1-x}\right)=2\left(x+\frac{x^3}{3}+\frac{x^5}{5}+\frac{x^7}{7}+\cdots\right) \qquad |x|<1$$

$$\ln\left(\frac{x+1}{x-1}\right)=2\left[\frac{1}{x}+\frac{1}{3}\left(\frac{1}{x}\right)^3+\frac{1}{5}\left(\frac{1}{x}\right)^5+\frac{1}{7}\left(\frac{1}{x}\right)^7+\cdots\right] \qquad |x|>1$$

$$\frac{\ln(1+x)}{1+x}=x-\left(1+\frac{1}{2}\right)x^2+\left(1+\frac{1}{2}+\frac{1}{3}\right)x^3-\cdots \qquad |x|<1$$

$$\ln|\sin x|=\ln|x|-\frac{x^2}{6}-\frac{x^4}{180}-\frac{x^6}{2835}-\cdots-\frac{2^{n-1}B_n x^{2n}}{n(2n)!}+\cdots \qquad 0<|x|<\pi$$

$$\ln|\cos x|=-\frac{x^2}{2}-\frac{x^4}{12}-\frac{x^6}{45}-\frac{17x^8}{2520}-\cdots-\frac{2^{2n-1}\left(2^{2n}-1\right)B_n x^{2n}}{n(2n)!}+\cdots \qquad |x|<\frac{\pi}{2}$$

$$\ln|\tan x|=\ln|x|+\frac{x^2}{3}+\frac{7x^4}{90}+\frac{62x^6}{2835}+\cdots+\frac{2^{2n}\left(2^{2n-1}-1\right)B_n x^{2n}}{n(2n)!}+\cdots \qquad 0<|x|<\frac{\pi}{2}$$

$$\begin{aligned}\frac{1}{(1-x)^2} &= \frac{d(\frac{1}{1-x})}{dx} \\ &= \sum_{n=1}^{\infty} nx^{n-1} \\ &= 1 + 2x + 3x^2 + 4x^3 + \cdots + nx^{n-1}\end{aligned}$$

伯努利数、欧拉数

可以发现，在 $S_m(n)$ 中 n^{m+1} 的系数总是 $\frac{1}{m+1}$，n^m 的系数总是 $-\frac{1}{2}$，n^{m-1} 的系数总是 $\frac{m}{12}$，n^{m-3} 的系数是 $-\frac{m(m-1)(m-2)}{720}$，n^{m-4} 的系数总是零等。

而 n^{m-k} 的系数总是某个常数乘以 $m^{\underline{k}}$，$m^{\underline{k}}$ 表示下降阶乘幂，即 $\frac{m!}{(m-k)!}$。

递推公式

$$\begin{aligned}S_m(n) &= \frac{1}{m+1}(B_0 n^{m+1} + \binom{m+1}{1} B_1 n^m + \cdots + \binom{m+1}{m} B_m n) \\ &= \frac{1}{m+1}\sum_{k=0}^{m}\binom{m+1}{k} B_k n^{m+1-k}\end{aligned}$$

伯努利数由隐含的递推关系定义：

$$\sum_{j=0}^{m}\binom{m+1}{j} B_j = 0, (m > 0)$$

$$B_0 = 1$$

例如，$\binom{2}{0}B_0 + \binom{2}{1}B_1 = 0$，前几个值显然是

n	0	1	2	3	4	5	6	7	8	…
B_n	1	$-\frac{1}{2}$	$\frac{1}{6}$	0	$-\frac{1}{30}$	0	$\frac{1}{42}$	0	$-\frac{1}{30}$	…

伯努利生成函数

伯努利数是指数生成函数的系数:

$$\frac{x}{e^x-1}=\sum_{k=0}^{\infty}\frac{B_k x^k}{k!}$$

很自然地我们需要将 $\frac{x}{e^x-1}$ 级数展开,

$$\begin{aligned}
f(x)&=\frac{x}{e^x-1} && \text{with } \lim_{x\to 0} f(x)=1\\
f'(x)&=\frac{e^x x-e^x+1}{(e^x-1)^2} && \text{with } \lim_{x\to 0} f'(x)=-\frac{1}{2}\\
f''(x)&=\frac{e^{2x}x-e^x x-2e^{2x}+2e^x}{(e^x-1)^3} && \text{with } \lim_{x\to 0} f''(x)=\frac{1}{6}\\
&\vdots && \vdots
\end{aligned}$$

进而,

$$\begin{aligned}
\frac{x}{e^x-1}&=\sum_{n=0}^{\infty}\frac{f^{(n)}x^n}{n!}\\
&=1+\left(-\frac{1}{2}\right)x+\left(\frac{1}{6}\right)\frac{x^2}{2!}+\left(-\frac{1}{30}\right)\frac{x^4}{4}+\left(\frac{1}{42}\right)\frac{x^6}{6!}+\cdots\\
&=1-\frac{x}{2}+\frac{x^2}{12}+\frac{x^4}{720}+\frac{x^6}{30240}+\cdots
\end{aligned}$$

$k=0$

所以得到伯努利数的递推表达式:

$$\begin{cases} B_0 = 1 \\ B_{n+1} = \sum\limits_{k=0}^{n+1} C_{n+1}^k B_k \end{cases}$$

另外一种表达递推形式为

$$\begin{cases} B_0 = 1 \\ \sum\limits_{k=0}^{n} C_{n+1}^k B_k = 0 \end{cases} \text{即} \begin{cases} B_0 = 1 \\ B_n = -\frac{1}{n+1} \sum\limits_{k=0}^{n-1} C_{n+1}^k B_k \end{cases}$$

同时我们得到一个副产品,即以下函数的泰勒展开式

$$\frac{x}{e^x-1} = \sum_{n=0}^{\infty} B_n \frac{x^n}{n!}$$

由递推表达式可以知道前几项的值 $B_0 = 1, B_1 = -\frac{1}{2}, B_2 = \frac{1}{6}, B_3 = 0 \ldots$,由表达式我们猜想 $B_{2n+1} = 0$ 知道,下面证明这一结果的正确性.

$$\cosh x \cdot \operatorname{sech} x = \left(\sum_{n=0}^{\infty} \frac{1}{(2n)!} x^{2n} \right) \left(\sum_{n=0}^{\infty} \frac{(-1)^n E_n}{(2n)!} x^{2n} \right)$$

$$= \left[1 + \frac{1}{2!} x^2 + \cdots + \frac{1}{(2n)!} x^{2n} + P_n(x) \right] \left[E_0 - \frac{E_1}{2!} x^2 + \cdots + \frac{(-1)^n E_n}{(2n)!} x^{2n} + Q_n(x) \right]$$
$$= 1$$

其中 $P_n(x), Q_n(x)$ 为余项.对照乘积中的 $2n$ 次幂的系数,可以得到

$$\frac{1}{0!} \cdot \frac{(-1)^n E_n}{(2n)!} + \cdots + \frac{1}{(2k)!} \cdot \frac{(-1)^{n-k} E_{n-k}}{[2(n-k)]!} + \cdots + \frac{1}{(2n)!} \cdot \frac{(-1)^0 E_0}{(0)!} = 0$$

$$\sum_{k=0}^{n} (2n)! \cdot \frac{1}{(2k)!} \cdot \frac{(-1)^{n-k} E_{n-k}}{[2(n-k)]!} = \sum_{k=0}^{n} \frac{(2n)!}{(2k)!(2n-2k)!} \cdot (-1)^{n-k} E_{n-k}$$

$$= \sum_{k=0}^{n} (-1)^{n-k} C_{2n}^{2k} E_{n-k} = \sum_{k=0}^{n} (-1)^k C_{2n}^{2n-2k} E_k = \sum_{k=0}^{n} (-1)^k C_{2n}^{2k} E_k = 0$$

其中 $n \geq 1, E_0 = 1$.所以我们就会得到欧拉数的一般递推公式

$$\begin{cases} E_0 = 1 & , n = 0 \\ \sum\limits_{k=0}^{n} (-1)^k C_{2n}^{2k} E_k = 0 & , n \geq 1 \end{cases}$$

或者表示为

$$\begin{cases} E_0 = 1 & , n = 0 \\ E_n = -\sum\limits_{k=0}^{n-1} (-1)^k C_{2n}^{2k} E_k & , n \geq 1 \end{cases}$$

离散/集合论

Bell Number（划分集合的方法的数量）

```
class Bell
{
    // Function to find n'th Bell Number
    static int bellNumber(int n)
    {
        int[][] bell = new int[n+1][n+1];
        bell[0][0] = 1;

        for (int i=1; i<=n; i++)
        {
            // Explicitly fill for j = 0
            bell[i][0] = bell[i-1][i-1];

            // Fill for remaining values of j
            for (int j=1; j<=i; j++)
                bell[i][j] = bell[i-1][j-1] + bell[i][j-1];
        }
        return bell[n][0];
    }
}
```

其他

逆序对

对逆序对个数有影响的就是区间内的数。设x为区间$[l,r]$中的逆序对个数。
由于没有相同的数，容易发现这样的等式关系:

$$x+\text{顺序对个数}=\text{总对数}$$

而反转之后，顺序对将变成逆序对，逆序对将变成顺序对。设原先的总逆序对个数为ans
那么反转区间$[l,r]$后答案将变成

$$ans-x+\frac{(r-l+1)*(r-l)}{2}-x$$

减去原先逆序对个数再加上顺序对个数即是新的对数化简得

$$newans=ans-2*x+\frac{(r-l+1)*(r-l)}{2}$$

考虑奇偶性的变化，$-2x$为偶数，对奇偶性无影响，于是对奇偶性有影响的只和总对数有关

因此n^2计算出原序列的逆序对个数，判断奇偶性，对于一个操作$O(1)$维护ans的奇偶性

math库

绝对值函数。

C++中fabs()函数位于< cmath >中，其余绝对值函数位于< cstdlib >中。

int abs(int i) 返回整型参数i的绝对值
double fabs(double x) 返回双精度参数x的绝对值
long labs(long n) 返回长整型参数n的绝对值
double cabs(struct complex znum) 求复数的绝对值

三角函数

double acos(double x) 返回x的反余弦arccos(x)值,x为弧度
double asin(double x) 返回x的反正弦arcsin(x)值,x为弧度
double atan(double x) 返回x的反正切arctan(x)值,x为弧度
double atan2(double x，double y) 带两个参数的反正切函数
double cos(double x) 返回x的余弦cos(x)值,x为弧度
double sin(double x) 返回x的正弦sin(x)值,x为弧度
double tan(double x) 返回x的正切tan(x)值,x为弧度

幂函数

double fmod (double x,double y); 返回两参数相除x/y的余数
double sqrt (double x) 返回x的开平方
double cbrt(double x) 计算x的立方根

对数函数

double log(double x) 返回logex的值
double $log10(x)$ 返回log10x的值
double $log2(x)$ x的二进制对数

double $log_m n$ log(n)/log(m)

指数函数

double exp(double x) 返回指数函数e^x的值
double exp2(double x) 返回2的x次方
double pow(double x,double y) 返回x^y的值
double pow10(int p) 返回10^p的值
frexp(param，n) 二进制浮点数表示方法 x=param*2^n*
*double ldexp(double x,int exp);这个函数刚好跟上面那个frexp函数功能相反，它的返回值是*x2^exp

返回小数

double modf(double value,double *iptr);拆分value值，返回它的小数部分，iptr指向整数部分（可返回）。
double frexp(double value,int * exp);这是一个将value值拆分成小数部分f和（以2为底的）指数部分exp，并返回小数部分f，即f* 2^exp。其中f取值在0.5~1.0范围或者0

取整

double ceil (double x); 取上整，返回比x大的最小整数

double floor (double x); 取下整，返回比x小的最大整数，即高斯函数[x]
double round(double x) 返回x的四舍五入值

最值

double fmax(double x,double y) 两个参数中的最大值
double fmin(x, y) 两个参数中的最小值

补充

double hypot(double x,double y);已知直角三角形两个直角边长度，求斜边长度

double poly(double x,int degree,double coeffs []);计算多项式

int matherr(struct exception *e);数学错误计算处理程序