Standard Code Library

stick16

Your Huazhong University of Science and Technology

November 29, 2024

Contents

一切的开始

宏定义

- 需要 C++11

```cpp
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define mid ((l+r)>>1)
// #define mod (998244353)
#define mod (1000000007)
#define ull unsigned long long
#define eps (1e-8)
#define mk make_pair
#define tim (double)clock()/CLOCKS_PER_SEC
#define For(i,a,b) for(int i=(a);i<=(b);++i)
#define rep(i,a,b) for(int i=(a);i>=(b);--i)
inline namespace IO{
    inline int read(){
        int x=0,f=1;char ch;
        while((ch=getchar())<'0'||x>'9')if(ch=='-')f=-1;
        while(ch>='0'&&ch<='9'){x=((x<<1)+(x<<3)+(ch^48)),ch=getchar();}
        return x*f;
    }
    void write(char x){putchar(x);}
    void write(const char *x){for(;*x;++x)putchar(*x);}
    void write(char *x){for(;*x;++x)putchar(*x);}
    void write(signed x){
        if(x<0)putchar('-'),x=-x;
        if(x>9)write(x/10); putchar('0'+x-x/10*10);
    }
    void write(long long x){
        if(x<0)putchar('-'),x=-x;
        if(x>9)write(x/10); putchar('0'+x-x/10*10);
    }
    void write(unsigned long long x){
        if(x>9)write(x/10);
        putchar('0'+x-x/10*10);
    }
    void write(double x){printf("%0.5lf",x);}
    template<typename type1,typename type2,typename ...typen>
    void write(type1 a1,type2 a2,typen ...an){
        write(a1);
        write(a2,an...);
    }
}using namespace IO;
inline int gcd(int x,int y){return y==0?x:gcd(y,x%y);}
inline int lcm(int x,int y){return x/gcd(x,y)*y;}
inline int lowbit(int x){return x&(-x);}
const int N=1900005;
```

数据结构

ST 表

- 二维

```cpp
int f[maxn][maxn][10][10];
inline int highbit(int x) { return 31 - __builtin_clz(x); }
inline int calc(int x, int y, int xx, int yy, int p, int q) {
    return max(
        max(f[x][y][p][q], f[xx - (1 << p) + 1][yy - (1 << q) + 1][p][q]),
        max(f[xx - (1 << p) + 1][y][p][q], f[x][yy - (1 << q) + 1][p][q])
    );
}
void init() {
    FOR (x, 0, highbit(n) + 1)
    FOR (y, 0, highbit(m) + 1)
        FOR (i, 0, n - (1 << x) + 1)
```

```cpp
        FOR (j, 0, m - (1 << y) + 1) {
            if (!x && !y) { f[i][j][x][y] = a[i][j]; continue; }
            f[i][j][x][y] = calc(
                i, j,
                i + (1 << x) - 1, j + (1 << y) - 1,
                max(x - 1, 0), max(y - 1, 0)
            );
        }
}
inline int get_max(int x, int y, int xx, int yy) {
    return calc(x, y, xx, yy, highbit(xx - x + 1), highbit(yy - y + 1));
}
```

fhq

```cpp
mt19937 rnd(time(NULL)^clock());
int siz[N],pri[N],val[N];
int tot;
int ls[N],rs[N];
#define lc ls[now]
#define rc rs[now]
int root;
inline int New(int k){
    int z=++tot;
    val[z]=k,pri[z]=rnd(),siz[z]=1;
    return z;
}
inline void push_up(int now){siz[now]=siz[lc]+siz[rc]+1;}
void split(int now,int k,int &x,int &y){
    if(!now){x=y=0;return;}
    if(val[now]<=k){x=now;split(rc,k,rc,y);}
    else {y=now;split(lc,k,x,lc);}
    push_up(now);
}
int merge(int x,int y){
    if(!x||!y)return x|y;
    if(pri[x]<pri[y]){rs[x]=merge(rs[x],y);push_up(x);return x;}
    else {ls[y]=merge(x,ls[y]);push_up(y);return y;}
}
inline void insert(int k){
    int x,y;
    split(root,k,x,y);
    root=merge(merge(x,New(k)),y);
}
inline void del(int k){
    int x,y,z;
    split(root,k-1,x,y);
    split(y,k,y,z);
    y=merge(ls[y],rs[y]);
    root=merge(merge(x,y),z);
}
inline int getrnk(int k){
    int x,y;
    split(root,k-1,x,y);
    int res=siz[x]+1;
    root=merge(x,y);
    return res;
}
inline int getkth(int now,int k){
    while(1){
        if(!now)return -1;
        if(siz[lc]+1==k)return val[now];
        if(siz[lc]>=k)now=lc;
        else k-=siz[lc]+1,now=rc;
    }
}
inline int getpre(int k){
    int x,y;
    split(root,k-1,x,y);
    int res=getkth(x,siz[x]);
    root=merge(x,y);
```

```
    return res;
}
inline int getnxt(int k){
    int x,y;
    split(root,k,x,y);
    int res=getkth(y,1);
    root=merge(x,y);
    return res;
}
```

数学

MinMax 容斥

$$\min(S) = \sum_{T\subseteq S, T\neq\emptyset} (-1)^{|T|+1} \max(S)$$

$$\max(S) = \sum_{T\subseteq S, T\neq\emptyset} (-1)^{|T|+1} \min(S)$$

excrt

```
int n,m;
int a[N],b[N];
int exgcd(int a,int b,int &x,int &y){
    if(b==0){x=1,y=0;return a;}
    int d=exgcd(b,a%b,x,y);
    int z=x;x=y,y=z-a/b*y;

    return d;
}
inline int mul(int a,int b,int mod){
    int res=0;
    for(;b;b>>=1){
        if(b&1)res=(res+a)%mod;
        a=(a<<1)%mod;
    }
    return res;
}
inline int excrt(){
    int ans=b[1];m=a[1];
    For(i,2,n){
        int t,y,c=((b[i]-ans)%a[i]+a[i])%a[i];
        int d=exgcd(m,a[i],t,y);
        // t=t*(c/d);
        t=mul(t,c/d,a[i]);
        ans+=t*m;
        m=lcm(m,a[i]);
        ans=(ans%m+m)%m;
    }
    int res=(ans%m+m)%m;
    return res;
}
signed main()
{
    n=read();
    For(i,1,n)a[i]=read(),b[i]=read();
    write(excrt());
    return 0;
}
/*
x=b[i](mod a[i])
*/
```

gauss

```
int n,m;
double a[N][N];
```

```cpp
inline void gauss(){
    For(i,1,n){
        int F=0;
        For(j,i,n)if(a[j][i]){
            For(k,1,m)swap(a[i][k],a[j][k]);
            F=1; break;
        }
        if(!F)For(j,1,i-1)if(a[j][i]&&a[i][i]==0){
            For(k,1,m)swap(a[i][k],a[j][k]);
            F=1; break;
        }
        if(!F)continue;
        double tt=a[i][i];
        For(k,i,m)a[i][k]/=tt;
        For(j,1,n)if(j!=i){
            tt=a[j][i];
            For(k,i,m)a[j][k]-=tt*a[i][k];
        }
    }
    For(i,1,n)if(a[i][i]==0&&a[i][m]){write(-1,'\n');return;}
    For(i,1,n)if(a[i][i]==0&&a[i][m]==0){write(0,'\n');return;}
    For(i,1,n)printf("x%lld=%0.2lf\n",i,a[i][m]);
}
signed main()
{
    n=read(),m=n+1;
    For(i,1,n)For(j,1,m)cin>>a[i][j];
    gauss();
    return 0;
}
/*
a[1][1]*x1+a[1][2]*x2+...+a[1][n]*xn=b1
a[2][1]*x1+a[2][2]*x2+...+a[2][n]*xn=b2
无解-1
无穷解 0
*/
```

线性筛

```cpp
phi[1]=1,mu[1]=1;
For(i,2,N-1){
  if(!vis[i]){p[++tot]=i,phi[i]=i-1,mu[i]=-1;}
  for(int j=1;j<=tot&&i*p[j]<N;++j){
    vis[i*p[j]]=1;
    if(i%p[j]!=0){
      phi[i*p[j]]=phi[i]*(p[j]-1);
      mu[i*p[j]]=-mu[i];
    }else{
      phi[i*p[j]]=phi[i]*p[j],mu[i*p[j]]=0;
      break;
    }
  }
}
```

二元一次方程 Ax+By=C

```cpp
auto clac = [&](int a, int b, int c) {
    int u = 1, v = 1;
    if (a < 0) { // 负数特判，但是没用经过例题测试
        a = -a;
        u = -1;
    }
    if (b < 0) {
        b = -b;
        v = -1;
    }

    int x, y, d = exgcd(a, b, x, y), ans;
    if (c % d != 0) { // 无整数解
        cout << -1 << "\n";
```

```
        return;
    }
    a /= d, b /= d, c /= d;
    x *= c, y *= c; // 得到可行解

    ans = (x % b + b - 1) % b + 1;
    auto [A, B] = pair{u * ans, v * (c - ans * a) / b}; // x 最小正整数 特解

    ans = (y % a + a - 1) % a + 1;
    auto [C, D] = pair{u * (c - ans * b) / a, v * ans}; // y 最小正整数 特解

    int num = (C - A) / b + 1; // xy 均为正整数 的 解的组数
};
```

求解连续按位异或

O(1) $0^1 2 \cdots n$

```
unsigned xor_n(unsigned n) {
    unsigned t = n & 3;
    if (t & 1) return t / 2u ^ 1;
    return t / 2u ^ n;
}
```

```
i64 xor_n(i64 n) {
    if (n % 4 == 1) return 1;
    else if (n % 4 == 2) return n + 1;
    else if (n % 4 == 3) return 0;
    else return n;
}
```

Miller - Rabin 素数测试

$O(4log^3x)$，这里可以看错 O(1)

```
int mul(int a, int b, int m) {
    int r = a * b - m * (int)(1.L / m * a * b);
    return r - m * (r >= m) + m * (r < 0);
}
int mypow(int a, int b, int m) {
    int res = 1 % m;
    for (; b; b >>= 1, a = mul(a, a, m)) {
        if (b & 1) {
            res = mul(res, a, m);
        }
    }
    return res;
}

int B[] = {2, 3, 5, 7, 11, 13, 17, 19, 23};
bool MR(int n) {
    if (n <= 1) return 0;
    for (int p : B) {
        if (n == p) return 1;
        if (n % p == 0) return 0;
    }
    int m = (n - 1) >> __builtin_ctz(n - 1);
    for (int p : B) {
        int t = m, a = mypow(p, m, n);
        while (t != n - 1 && a != 1 && a != n - 1) {
            a = mul(a, a, n);
            t *= 2;
        }
        if (a != n - 1 && t % 2 == 0) return 0;
    }
    return 1;
}
```

Pollard - Rho 因式分解

以单个因子 O(logX) 的复杂度输出数字 X 的全部质因数，由于需要结合素数测试，总复杂度会略高一些。如果遇到超时的情况，可能需要考虑进一步优化，例如检查题目是否强制要求枚举全部质因数等等

```
int PR(int n) {
    for (int p : B) {
        if (n % p == 0) return p;
    }
    auto f = [&](int x) -> int {
        x = mul(x, x, n) + 1;
        return x >= n ? x - n : x;
    };
    int x = 0, y = 0, tot = 0, p = 1, q, g;
    for (int i = 0; (i & 255) || (g = gcd(p, n)) == 1; i++, x = f(x), y = f(f(y))) {
        if (x == y) {
            x = tot++;
            y = f(x);
        }
        q = mul(p, abs(x - y), n);
        if (q) p = q;
    }
    return g;
}
vector<int> fac(int n) {
    #define pb emplace_back
    if (n == 1) return {};
    if (MR(n)) return {n};
    int d = PR(n);
    auto v1 = fac(d), v2 = fac(n / d);
    auto i1 = v1.begin(), i2 = v2.begin();
    vector<int> ans;
    while (i1 != v1.end() || i2 != v2.end()) {
        if (i1 == v1.end()) {
            ans.pb(*i2++);
        } else if (i2 == v2.end()) {
            ans.pb(*i1++);
        } else {
            if (*i1 < *i2) {
                ans.pb(*i1++);
            } else {
                ans.pb(*i2++);
            }
        }
    }
    return ans;
}
```

数论常见结论及例题

常见结论

球盒模型（八种）

参考链接。给定 n 个小球 m 个盒子。

- 球同，盒不同、不能空

  隔板法：N 个小球即一共 $N-1$ 个空，分成 M 堆即 $M-1$ 个隔板，答案为 $\dbinom{n-1}{m-1}$。

- 球同，盒不同、能空

  隔板法：多出 $M-1$ 个虚空球，答案为 $\dbinom{m-1+n}{n}$。

- 球同，盒同、能空

  $\dfrac{1}{(1-x)(1-x^2)\dots(1-x^m)}$ 的 x^n 项的系数。

- 球同，盒同、不能空

  $\dfrac{x^m}{(1-x)(1-x^2)\dots(1-x^m)}$ 的 x^n 项的系数。

- 球不同，盒同、不能空

  第二类斯特林数 `Stirling2`(n,m) - 球不同，盒同、能空

  第二类斯特林数之和 $\displaystyle\sum_{i=1}^{m}$ `Stirling2`(n,m)，答案为 $\sum_{i=0}^{m} dp[n][i]$。

- 球不同，盒不同、不能空

  第二类斯特林数乘上 m 的阶乘 $m!\cdot$ `Stirling2`(n,m)，答案为 $dp[n][m]*m!$。

- 球不同，盒不同、能空

  答案为 m^n。

麦乐鸡定理

给定两个互质的数 n,m，定义 $x=a*n+b*m\ a\geq 0,b\geq 0$，当 $x>n*m-n-m$ 时，该式子恒成立。

一些结论：

> 对于给定的 X 和序列 $[a_1,a_2,...,a_n]$，有：$\boldsymbol{X=(X\&a_1)or(X\&a_2)or...or(X\&a_n)}$。原理是 and 意味着取交集，or 意味着取子集。来源 - 牛客小白月赛 49C

调和级数近似公式

```
log(n) + 0.5772156649 + 1.0 / (2 * n)
```

欧拉函数常见性质

- $1-n$ 中与 n 互质的数之和为 $n*\varphi(n)/2$。
- 若 $a\ b$ 互质，则 $\varphi(a*b)=\varphi(a)*\varphi(b)$。实际上，所有满足这一条件的函数统称为积性函数。
- 若 f 是积性函数，且有 $n=\displaystyle\prod_{i=1}^{m}p_i^{c_i}$，那么 $f(n)=\displaystyle\prod_{i=1}^{m}f(p_i^{c_i})$。
- 若 p 为质数，且满足 $p\mid n$，
  - $p^2\mid n$，那么 $\varphi(n)=\varphi(n/p)*p$。
  - $p^2\nmid n$，那么 $\varphi(n)=\varphi(n/p)*(p-1)$。
- $\displaystyle\sum_{d|n}\varphi(d)=n$。> 如 $n=10$，则 $d=10/5/2/1$，那么 $10=\varphi(10)+\varphi(5)+\varphi(2)+\varphi(1)$。
- $\displaystyle\sum_{i=1}^{n}\gcd(i,n)=\sum_{d|n}\left\lfloor\frac{n}{d}\right\rfloor\varphi(d)$ （欧拉反演）。

组合数学常见性质

- $k*C_n^k=n*C_{n-1}^{k-1}$；
- $C_k^n*C_m^k=C_m^n*C_{m-n}^{m-k}$；
- $C_n^k+C_n^{k+1}=C_{n+1}^{k+1}$；
- $\sum_{i=0}^{n}C_n^i=2^n$；
- $\sum_{k=0}^{n}(-1)^k*C_n^k=0$。
- 拉格朗日恒等式：$\displaystyle\sum_{i=1}^{n}\sum_{j=i+1}^{n}(a_ib_j-a_jb_i)^2=(\sum_{i=1}^{n}a_i)^2(\sum_{i=1}^{n}b_i)^2-(\sum_{i=1}^{n}a_ib_i)^2$。

范德蒙德卷积公式

在数量为 $n+m$ 的堆中选 k 个元素,和分别在数量为 n m 的堆中选 i $k-i$ 个元素的方案数是相同的,即 $\sum_{i=0}^{k}\binom{n}{i}\binom{m}{k-i}=\binom{n+m}{k}$;

变体:

- $\sum_{i=0}^{k} C_{i+n}^{i}=C_{k+n+1}^{k}$;
- $\sum_{i=0}^{k} C_{n}^{i}*C_{m}^{i}=\sum_{i=0}^{k} C_{n}^{i}*C_{m}^{m-i}=C_{n+m}^{n}$ 。

卡特兰数

是一类奇特的组合数，前几项为 $1,1,2,5,14,42,132,429,1430,4862$ 。如遇到以下问题，则直接套用即可。

- 【括号匹配问题】n 个左括号和 n 个右括号组成的合法括号序列的数量，为 Cat_n 。
- 【进出栈问题】$1,2,...,n$ 经过一个栈，形成的合法出栈序列的数量，为 Cat_n 。
- 【二叉树生成问题】n 个节点构成的不同二叉树的数量，为 Cat_n 。
- 【路径数量问题】在平面直角坐标系上，每一步只能**向上**或**向右**走，从 $(0,0)$ 走到 (n,n) ，并且除两个端点外不接触直线 $y=x$ 的路线数量，为 $2Cat_{n-1}$ 。

计算公式：$Cat_n=\frac{C_{2n}^{n}}{n+1}$ ，$C_n=\frac{C_{n-1}*(4n-2)}{n+1}$ 。

斐波那契数列

通项公式：$F_n=\frac{1}{\sqrt{5}}*\left[\left(\frac{1+\sqrt{5}}{2}\right)^n-\left(\frac{1-\sqrt{5}}{2}\right)^n\right]$ 。

直接结论:

- 卡西尼性质：$F_{n-1}*F_{n+1}-F_n^2=(-1)^n$;
- $F_n^2+F_{n+1}^2=F_{2n+1}$;
- $F_{n+1}^2-F_{n-1}^2=F_{2n}$ （由上一条写两遍相减得到）;
- 若存在序列 $a_0=1,a_n=a_{n-1}+a_{n-3}+a_{n-5}+...(n\geq 1)$ 则 $a_n=F_n(n\geq 1)$;
- 齐肯多夫定理：任何正整数都可以表示成若干个不连续的斐波那契数（F_2 开始）可以用贪心实现。

求和公式结论:

- 奇数项求和：$F_1+F_3+F_5+...+F_{2n-1}=F_{2n}$;
- 偶数项求和：$F_2+F_4+F_6+...+F_{2n}=F_{2n+1}-1$;
- 平方和：$F_1^2+F_2^2+F_3^2+...+F_n^2=F_n*F_{n+1}$;
- $F_1+2F_2+3F_3+...+nF_n=nF_{n+2}-F_{n+3}+2$;
- $-F_1+F_2-F_3+...+(-1)^nF_n=(-1)^n(F_{n+1}-F_n)+1$;
- $F_{2n-2m-2}(F_{2n}+F_{2n+2})=F_{2m+2}+F_{4n-2m}$ 。

数论结论:

- $F_a\mid F_b\Leftrightarrow a\mid b$;
- $\gcd(F_a,F_b)=F_{\gcd(a,b)}$;
- 当 p 为 $5k\pm 1$ 型素数时，$\begin{cases}F_{p-1}\equiv 0\pmod p\\F_p\equiv 1\pmod p\\F_{p+1}\equiv 1\pmod p\end{cases}$;
- 当 p 为 $5k\pm 2$ 型素数时，$\begin{cases}F_{p-1}\equiv 1\pmod p\\F_p\equiv -1\pmod p\\F_{p+1}\equiv 0\pmod p\end{cases}$;
- $F(n)\%m$ 的周期 $\leq 6m$ （$m=2\times 5^k$ 时取到等号）;
- 既是斐波那契数又是平方数的有且仅有 $1,144$ 。

杂

- 负数取模得到的是负数，如果要用 $0/1$ 判断的话请取绝对值；
- 辗转相除法原式为 $\text{gcd}(x,y)=\text{gcd}(x,y-x)$ ，推广到 N 项为 $\text{gcd}(a_1,a_2,\dots,a_N)=\text{gcd}(a_1,a_2-a_1,\dots,a_N-a_{N-1})$ ，
  - 该推论在 "四则运算后 gcd " 这类题中有特殊意义，如求解 $\text{gcd}(a_1+X,a_2+X,\dots,a_N+X)$ 时See；
- 以下式子成立：$\text{gcd}(a,m)=\text{gcd}(a+x,m)\Leftrightarrow\text{gcd}(a,m)=\text{gcd}(x,m)$ 。求解上式满足条件的 x 的数量即为求比 $\dfrac{m}{\text{gcd}(a,m)}$ 小且与其互质的数的个数，即用欧拉函数求解 $\varphi\left(\dfrac{m}{\text{gcd}(a,m)}\right)$ 。
- 已知序列 a ，定义集合 $S=\{a_i\cdot a_j\mid i<j\}$ ，现在要求解 $\text{gcd}(S)$ ，即为求解 $\text{gcd}(a_j,\text{gcd}(a_i\mid i<j))$ ，换句话说，即为求解后缀 gcd 。
- 连续四个数互质的情况如下,当 n 为奇数时,$n,n-1,n-2$ 一定互质;而当 n 为偶数时,$\begin{cases}n,n-1,n-3\text{互质} & \text{gcd}(n,n-3)=1\text{时}\\ n-1,n-2,n-3\text{互质} & \text{gcd}(n,n-3)\neq 1\text{时}\end{cases}$ See；
- 由 $a\mod b=(b+a)\mod b=(2\cdot b+a)\mod b=\dots=(K\cdot b+a)\mod b$ 可以推广得到 $(a\mod b)\mod c=((K\cdot bc+a)\mod b)\mod c$ ，由此可以得到一个 bc 的答案周期See；
- 对于长度为 $2\cdot N$ 的数列 a ，将其任意均分为两个长度为 N 的数列 p,q ，随后对 p 非递减排序、对 q 非递增排序，定义 $f(p,q)=\sum\limits_{i=1}^{n}|p_i-q_i|$ ，那么答案为 a 数列前 N 大的数之和减去前 N 小的数之和See。
- 令 $\begin{cases}X=a+b\\ Y=a\oplus b\end{cases}$ ，**如果**该式子**有解**，那么存在前提条件 $\begin{cases}X\geq Y\\ X,Y\text{同奇偶}\end{cases}$ ；进一步，此时最小的 a 的取值为 $\dfrac{X-Y}{2}$ See。

  然而，上方方程并不总是有解的，只有当变量增加到三个时，才**一定有解**，即：**在保证上方前提条件成立的情况下**，求解 $\begin{cases}X=a+b+c\\ Y=a\oplus b\oplus c\end{cases}$ ，则一定存在一组解 $\{\dfrac{X-Y}{2},\dfrac{X-Y}{2},Y\}$ See。
- 已知序列 p 是由序列 a_1 、序列 a_2 、……、序列 a_n 合并而成，且合并过程中各序列内元素相对顺序不变，记 $T(p)$ 是 p 序列的最大前缀和，则 $T(p)=\sum\limits_{i=1}^{n}T(a_i)$ See 。
- $x+y=x|y+x\&y$ ，对于两个数字 x 和 y ，如果将 x 变为 $x|y$ ，同时将 y 变为 $x\&y$ ，那么在本质上即将 x 二进制模式下的全部 1 移动到了 y 的对应的位置上 See 。
- 一个正整数 x 异或、加上另一个正整数 y 后奇偶性不发生变化：$a+b\equiv a\oplus b(\text{mod}2)$ See 。

常见例题

题意：将 1 至 N 的每个数字分组，使得每一组的数字之和均为质数。输出每一个数字所在的组别，且要求分出的组数最少 See 。

考察哥德巴赫猜想，记全部数字之和为 S ，分类讨论如下：

- 为 S 质数时，只需要分入同一组；
- 当 S 为偶数时，由猜想可知一定能分成两个质数，可以证明其中较小的那个一定小于 N ，暴力枚举分组；
- 当 $S-2$ 为质数时，特殊判断出答案；
- 其余情况一定能被分成三组，其中 3 单独成组，$S-3$ 后成为偶数，重复讨论二的过程即可。

题意：给定一个长度为 n 的数组，定义这个数组是 BAD 的，当且仅当可以把数组分成两个子序列，这两个子序列的元素之和相等。现在你需要删除**最少的**元素，使得删除后的数组不是 BAD 的。

最少删除一个元素——如果原数组存在奇数，则直接删除这个奇数即可；反之，我们发现，对数列同除以一个数不影响计算，故我们只需要找到最大的满足 $2^k\mid a_i$ 成立的 2^k ，随后将全部的 a_i 变为 $\dfrac{a_i}{2^k}$ ，此时一定有一个奇数（换句话说，我们可以对原数列的每一个元素不断的除以 2 直到出现奇数为止），删除这个奇数即可 See 。

题意：设当前有一个数字为 x，减去、加上最少的数字使得其能被 k 整除。

最少减去 $x \bmod k$ 这个很好想；最少加上 $\left(\left\lceil\frac{x}{k}\right\rceil * k\right) \bmod k$ 也比较好想，但是更简便的方法为加上 $k - x \bmod k$，这个式子等价于前面这一坨。

题意：给定一个整数 n，用恰好 k 个 2 的幂次数之和表示它。例如：$n = 9, k = 4$，答案为 $1 + 2 + 2 + 4$。

结论 1：k 合法当且仅当 `__builtin_popcountll(n) <= k && k <= n`，显然。

结论 2：$2^{k+1} = 2 \cdot 2^k$，所以我们可以将二进制位看作是数组，然后从高位向低位推，一个高位等于两个低位，直到数组之和恰好等于 k，随后依次输出即可。举例说明，$\{1,0,0,1\} \to \{0,2,0,1\} \to \{0,1,2,1\}$，即答案为 0 个 2^3、1 个 2^2、……。

```
signed main() {
    int n, k;
    cin >> n >> k;

    int cnt = __builtin_popcountll(n);

    if (k < cnt || n < k) {
        cout << "NO\n";
        return 0;
    }
    cout << "YES\n";

    vector<int> num;
    while (n) {
        num.push_back(n % 2);
        n /= 2;
    }

    for (int i = num.size() - 1; i > 0; i--) {
        int p = min(k - cnt, num[i]);
        num[i] -= p;
        num[i - 1] += 2 * p;
        cnt += p;
    }

    for (int i = 0; i < num.size(); i++) {
        for (int j = 1; j <= num[i]; j++) {
            cout << (1LL << i) << " ";
        }
    }
}
```

题意：n 个取值在 $[0, k)$ 之间的数之和为 m 的方案数

答案为 $\sum_{i=0}^{n} -1^i \cdot \binom{n}{i} \cdot \binom{m - i \cdot k + n - 1}{n - 1}$

图论

LCA

```
vector<int> edge[N];
int fa[N],dep[N],siz[N],son[N];
int top[N],dfn[N],tot;
void dfs1(int x,int f){
    fa[x]=f,dep[x]=dep[f]+1;
    siz[x]=1,son[x]=-1;
    for(int y:edge[x]){
        if(y==f)continue;
        dfs1(y,x);
        siz[x]+=siz[y];
        if(son[x]==-1||siz[y]>siz[son[x]])son[x]=y;
    }
```

```
}
void dfs2(int x,int t){
    dfn[x]=++tot;
    top[x]=t;
    if(son[x]==-1)return;
    dfs2(son[x],t);
    for(int y:edge[x]){
        if(y==fa[x]||y==son[x])continue;
        dfs2(y,y);
    }
}
inline int Lca(int x,int y){
    int fx=top[x],fy=top[y];
    while(fx!=fy){
        if(dep[fx]>=dep[fy])x=fa[fx];
        else y=fa[fy];
        fx=top[x],fy=top[y];
    }
    if(dfn[x]<dfn[y])return x;
    else return y;
}
```

prim

```
const int N = 550, INF = 0x3f3f3f3f;
int n, m, g[N][N];
int d[N], v[N];
int prim() {
    ms(d, 0x3f); //这里的 d 表示到“最小生成树集合”的距离
    int ans = 0;
    for (int i = 0; i < n; ++ i) { //遍历 n 轮
        int t = -1;
        for (int j = 1; j <= n; ++ j)
            if (v[j] == 0 && (t == -1 || d[j] < d[t])) //如果这个点不在集合内且当前距离集合最近
                t = j;
        v[t] = 1; //将 t 加入“最小生成树集合”
        if (i && d[t] == INF) return INF; //如果发现不连通，直接返回
        if (i) ans += d[t];
        for (int j = 1; j <= n; ++ j) d[j] = min(d[j], g[t][j]); //用 t 更新其他点到集合的距离
    }
    return ans;
}
int main() {
    ms(g, 0x3f); cin >> n >> m;
    while (m -- ) {
        int x, y, w; cin >> x >> y >> w;
        g[x][y] = g[y][x] = min(g[x][y], w);
    }
    int t = prim();
    if (t == INF) cout << "impossible" << endl;
    else cout << t << endl;
} //22.03.19 已测试
```

图的连通性:

无向图的割点与桥:

给定无向图 $G=(V,E)$:

若对于 $x \in V$，从图中删去节点 x 以及所有相连边后，G 分裂为两个及以上不相连子图，则称 x 为 G 的**割点**。

若对于 $e \in E$，从图中删去边 e 后，G 分裂为两个不相连子图，则称 x 为 G 的**桥**或**割边**。

追溯值:

设 dfn_x 为 x 在搜索树上的时间戳，low_x 为 x 的追溯值。

- 追溯值定义:

设 $subtree(x)$ 表示无向图搜索树中以 x 为根的子树。

- $low[x] = dfn[x] \leftarrow$ 初值。
- $low[x] = \min(low[x], low[y]) \leftarrow$ 若 x 为 y 父节点。
- $low[x] = \min(low[x], dfn[y]) \leftarrow$ 若边 (x, y) 不是搜索树上的边。

设 x 为 y 的父节点，$low[y]$ 的意义为：从 $subtree(y)$ 出发，在不经过边 (x, y) 的前提下，能到达的时间戳最小的点。

无向图中的简单环:

搜索树上每条额外边都构成一个简单环，且由 $dfs $ 的性质，该边为环上 dfn 最大的点，连到该环 dfn 最小的点上。判定方式:

$$dfn[x] < dfn[y]$$

割边判定法则:

无向边 (x, y) 是桥，当且仅当搜索树上存在 x 的一个子节点 y，满足:

$$dfn[x] < low[y]$$

根据定义，$dfn[x] < low[y]$ 说明从 $subtree(y)$ 出发，在不经过 (x, y) 的前提下，无法到达 x 或比 x 访问更早的节点。则显然边 (x, y) 为桥。

性质：桥是**搜索树**上的边，且一个**简单环**中的边不是桥。

割点判定法则:

若 x 非搜索树根节点，则 x 是割点当且仅当搜索树上存在一个子节点 y, 满足:

$$dfn[x] \leq low[y]$$

特别的，若 x 是搜索树根节点，则 x 是割点当且仅当其存在至少两个子节点 y_1, y_2 满足上述条件。证明通同割边类似。

无向图求点双联通分量:

Tarjan 时维护一个栈：1. 当一个节点第一次被访问时，入栈

2. 当割点判定条件成立时，无论 x 是否为根:
   - 不断弹出栈顶节点，直到节点 y 被弹出。
   - 刚才弹出的所有节点与 x 一起构成一个 `v-DCC`

对于点双的缩点，上述的每个 x 拆成多个点，分别连向对应的虚点 X。

有向图求强联通分量:

若对于途中任意两个节点 x, y，既存在从 x 到 y 的路径，也存在从 y 到 x 的路径，则称该有向图为“强连通图”。有向图的极大强联通子图称为“强连通分量”，即为 SCC。

$dfn[x] = low[x]$ 的点，必定是一个 SCC 中最先被访问到的点，此时在栈中的所有点必定满足 $dfn[x] > low[x]$，直接弹出元素直到 x 被弹出为止。细节: 不在栈中的点已经不在这个 SCC 中了，所以不能更新该点的 $low[x]$。

对于大多有向图上的 DP，通用做法是：缩点 $\rightarrow$ 拓扑排序 $\rightarrow$ DAG 上 DP。

dcc:

```cpp
int n,m;
vector<pair<int,int>> edge[N];
int dfn[N],low[N],tot;
int vis[N],dcc[N],cnt;
vector<int> vec[N];
void dfs(int x,int ID){
    dfn[x]=low[x]=++tot;
    for(auto yy:edge[x]){
        int y=yy.first,id=yy.second;
        if(id==ID)continue;
        if(!dfn[y]){
            dfs(y,id);
            low[x]=min(low[x],low[y]);
        }else low[x]=min(low[x],dfn[y]);
        if(low[y]>dfn[x])vis[id]=1;
```

```
    }
}
void dfsc(int x,int ID){
    dcc[x]=cnt; vec[cnt].push_back(x);
    for(auto yy:edge[x]){
        int y=yy.first,id=yy.second;
        if(id==ID||vis[id])continue;
        if(!dcc[y])dfsc(y,id);
    }
}
signed main()
{
    n=read(),m=read();
    For(i,1,m){
        int u=read(),v=read();
        edge[u].push_back(mk(v,i));
        edge[v].push_back(mk(u,i));
    }
    For(i,1,n)if(!dfn[i])dfs(i,0);
    For(i,1,n)if(!dcc[i])cnt++,dfsc(i,0);
    write(cnt,'\n');
    For(i,1,cnt){
        write((signed)vec[i].size(),' ');
        for(int x:vec[i])write(x,' ');
        write('\n');
    }
    return 0;
}
```

有向图缩点：

```
int n,m;
int a[N];
int dfn[N],low[N],tot;
int stk[N],vis[N],top;
int col[N],V[N],cnt;
vector<int> edge[N],e[N];
void dfs(int x){
    dfn[x]=low[x]=++tot;
    stk[++top]=x,vis[x]=1;
    for(int y:edge[x]){
        if(!dfn[y]){
            dfs(y);
            low[x]=min(low[x],low[y]);
        }else if(vis[y])low[x]=min(low[x],dfn[y]);
    }
    if(dfn[x]==low[x]){
        ++cnt;
        while(stk[top]!=x){
            int p=stk[top--];
            V[cnt]+=a[p];
            col[p]=cnt,vis[p]=0;
        }int p=stk[top--];
        V[cnt]+=a[p];
        col[p]=cnt,vis[p]=0;
    }
}
int deg[N];
int dp[N];
signed main()
{
    n=read(),m=read();
    For(i,1,n)a[i]=read();
    For(i,1,m){
        int u=read(),v=read();
        edge[u].push_back(v);
        // edge[v].push_back(u);
    }
    For(i,1,n)if(!dfn[i])dfs(i);
    For(i,1,n)for(int y:edge[i]){
        if(col[i]==col[y])continue;
```

```cpp
        e[col[i]].push_back(col[y]);
        deg[col[y]]++;
    }
    queue<int> q;
    For(i,1,cnt)if(!deg[i])q.push(i),dp[i]=V[i];
    while(!q.empty()){
        int x=q.front();q.pop();
        for(int y:e[x]){
            deg[y]--;dp[y]=max(dp[y],dp[x]+V[y]);
            if(!deg[y])q.push(y);
        }
    }
    int res=0;
    For(i,1,cnt)res=max(res,dp[i]);
    write(res,'\n');
    return 0;
}
/*

*/
```

二分图

最大独立集 = 总点数 − 最小点覆盖 (集合) 最大权闭合子图 = 正点权和 − 最小割 (构造) 对于正价点, 连源, 边权为点权。对应地, 负价点连汇, 边权为 (负) 点权。原图中的有向边保留, 边权置为 INF

最小路径覆盖 = 总点数 − 拆点二分图最大匹配

HopcroftKarp 算法（基于最大流）解

```cpp
struct HopcroftKarp {
    int n, m;
    vector<array<int, 2>> ver;
    vector<int> l, r;

    HopcroftKarp(int n, int m) : n(n), m(m) { // 左右半部
        l.assign(n, -1);
        r.assign(m, -1);
    }
    void add(int x, int y) {
        x--, y--; // 这个板子是 0-idx 的
        ver.push_back({x, y});
    }
    int work() {
        vector<int> adj(ver.size());

        mt19937 rgen(chrono::steady_clock::now().time_since_epoch().count());
        shuffle(ver.begin(), ver.end(), rgen); // 随机化防卡

        vector<int> deg(n + 1);
        for (auto &[u, v] : ver) {
            deg[u]++;
        }
        for (int i = 1; i <= n; i++) {
            deg[i] += deg[i - 1];
        }
        for (auto &[u, v] : ver) {
            adj[--deg[u]] = v;
        }

        int ans = 0;
        vector<int> a, p, q(n);
        while (true) {
            a.assign(n, -1), p.assign(n, -1);

            int t = 0;
            for (int i = 0; i < n; i++) {
                if (l[i] == -1) {
                    q[t++] = a[i] = p[i] = i;
                }
            }
```

```
                bool match = false;
                for (int i = 0; i < t; i++) {
                    int x = q[i];
                    if (~l[a[x]]) continue;

                    for (int j = deg[x]; j < deg[x + 1]; j++) {
                        int y = adj[j];
                        if (r[y] == -1) {
                            while (~y) {
                                r[y] = x;
                                swap(l[x], y);
                                x = p[x];
                            }
                            match = true;
                            ++ans;
                            break;
                        }
                        if (p[r[y]] == -1) {
                            q[t++] = y = r[y];
                            p[y] = x;
                            a[y] = a[x];
                        }
                    }
                }
                if (!match) break;
            }
            return ans;
        }
        vector<array<int, 2>> answer() {
            vector<array<int, 2>> ans;
            for (int i = 0; i < n; i++) {
                if (~l[i]) {
                    ans.push_back({i, l[i]});
                }
            }
            return ans;
        }
};

signed main() {
    int n1, n2, m;
    cin >> n1 >> n2 >> m;
    HopcroftKarp flow(n1, n2);
    while (m--) {
        int x, y;
        cin >> x >> y;
        flow.add(x, y);
    }

    cout << flow.work() << "\n";

    auto match = flow.answer();
    for (auto [u, v] : match) {
        cout << u << " " << v << "\n";
    }
}
```

欧拉路径/欧拉回路 Hierholzers

有向图欧拉路径存在判定有向图欧拉路径存在：恰有一个点出度比入度多 1（为起点）恰有一个点入度比出度多 1（为终点) 恰有 N-2 个点入度均等于出度。如果是欧拉回路，则上方起点与终点的条件不存在，全部点均要满足最后一个条件。无向图欧拉路径存在判定无向图欧拉路径存在：恰有两个点度数为奇数（为起点与终点）. 恰有 N-2 个点度数为偶数。有向图欧拉路径求解（字典序最小）：

```
vector<int> ans;
auto dfs = [&](auto self, int x) -> void {
    while (ver[x].size()) {
        int net = *ver[x].begin();
        ver[x].erase(ver[x].begin());
        self(self, net);
```

```
    }
    ans.push_back(x);
};
dfs(dfs, s);
reverse(ans.begin(), ans.end());
for (auto it : ans) {
    cout << it << " ";
}
```

图论常见结论及例题

常见结论

1. 要在有向图上求一个最大点集，使得任意两个点 (i,j) 之间至少存在一条路径（可以是从 i 到 j ，也可以反过来，这两种有一个就行），**即求解最长路**；
2. 要求出连通图上的任意一棵生成树，只需要跑一遍 **bfs** ；
3. 给出一棵树，要求添加尽可能多的边，使得其是二分图：对树进行二分染色，显然，相同颜色的点之间连边不会破坏二分图的性质，故可添加的最多的边数即为 $cnt_{\text{Black}} * cnt_{\text{White}} - (n-1)$ ；
4. 当一棵树可以被黑白染色时，所有染黑节点的度之和等于所有染白节点的度之和；
5. 在竞赛图中，入度小的点，必定能到达出度小（入度大）的点。
6. 在竞赛图中,将所有点按入度从小到大排序,随后依次遍历,若对于某一点 i 满足前 i 个点的入度之和恰好等于 $\left\lfloor \dfrac{n \cdot (n+1)}{2} \right\rfloor$,那么对于上一次满足这一条件的点 p , $p+1$ 到 i 点构成一个新的强连通分量。> 举例说明, 设满足上方条件的点为 p_1, p_2 $(p_1+1 < p_2)$ ，那么点 1 到 p_1 构成一个强连通分量、点 p_1+1 到 p_2 构成一个强连通分量。
7. 选择图中最少数量的边删除，使得图不连通，即求最小割；如果是删除点，那么拆点后求最小割。
8. 如果一张图是**平面图**，那么其边数一定小于等于 $3n-6$ 。
9. 若一张有向完全图存在环，则一定存在三元环。
10. 竞赛图三元环计数：
11. 有向图判是否存在环直接用 topsort；无向图判是否存在环直接用 dsu，也可以使用 topsort，条件变为 `deg[i] <= 1` 时入队。

常见例题

杂

题意：给出一棵节点数为 $2n$ 的树，要求将点分割为 n 个点对，使得点对的点之间的距离和最大。

可以转化为边上问题：对于每一条边，其被利用的次数即为 $\min\{\text{其左边的点的数量}, \text{其右边的点的数量}\}$ ，使用树形 dp 计算一遍即可。如下图样例，答案为 10 。

```
vector<int> val(n + 1, 1);
int ans = 0;
function<void(int, int)> dfs = [&](int x, int fa) {
    for (auto y : ver[x]) {
        if (y == fa) continue;
        dfs(y, x);
        val[x] += val[y];
        ans += min(val[y], k - val[y]);
    }
};
dfs(1, 0);
cout << ans << endl;
```

题意：以哪些点为起点可以无限的在有向图上绕

概括一下这些点可以发现，一类是环上的点，另一类是可以到达环的点。建反图跑一遍 topsort 板子，根据容斥，未被移除的点都是答案

题意：添加最少的边，使得有向图变成一个 SCC

将原图的 SCC 缩点，统计缩点后的新图上入度为 0 和出度为 0 的点的数量 cnt_{in} cnt_{out} ，答案即为 $\max(cnt_{\text{in}}, cnt_{\text{out}})$ 。过程大致是先将一个出度为 0 的点和一个入度为 0 的点相连，剩下的点随便连。

题意：添加最少的边，使得无向图变成一个 E-DCC

将原图的 E-DCC 缩点，统计缩点后的新图上入度为 1 的点（叶子结点）的数量 cnt ，答案即为 $\lceil \frac{cnt}{2} \rceil$ 。过程大致是每次找两个叶子结点（但是还有一些条件限制）相连，若最后余下一个点随便连。

题意：在树上找到一个最大的连通块，使得这个联通内点权和边权之和最大，输出这个值，数据中存在负数的情况。

使用 dfs 即可解决。

```
LL n, point[N];
LL ver[N], head[N], nex[N], tot; bool v[N];
map<pair<LL, LL>, LL> edge;
// void add(LL x, LL y) {}
void dfs(LL x) {
    for (LL i = head[x]; i; i = nex[i]) {
        LL y = ver[i];
        if (v[y]) continue;
        v[y] = true; dfs(y); v[y] = false;
    }
    for (LL i = head[x]; i; i = nex[i]) {
        LL y = ver[i];
        if (v[y]) continue;
        point[x] += max(point[y] + edge[{x, y}], 0LL);
    }
}
void Solve() {
    cin >> n;
    FOR(i, 1, n) cin >> point[i];
    FOR(i, 2, n) {
        LL x, y, w; cin >> x >> y >> w;
        edge[{x, y}] = edge[{y, x}] = w;
        add(x, y), add(y, x);
    }
    v[1] = true; dfs(1); LL ans = -MAX18;
    FOR(i, 1, n) ans = max(ans, point[i]);
    cout << ans << endl;
}
```

Prüfer 序列：凯莱公式

题意：给定 n 个顶点，可以构建出多少棵标记树？

$n \le 4$ 时的样例如上，通项公式为 n^{n-2} 。

Prüfer 序列

一个 n 个点 m 条边的带标号无向图有 k 个连通块。我们希望添加 $k-1$ 条边使得整个图连通，求方案数量。

设 s_i 表示每个连通块的数量，通项公式为 $n^{k-2} \cdot \prod_{i=1}^{k} s_i$ ，当 $k < 2$ 时答案为 1 。

单源最短/次短路计数

```
const int N = 2e5 + 7, M = 1e6 + 7;
int n, m, s, e; int d[N][2], v[N][2]; // 0 代表最短路， 1 代表次短路
Z num[N][2];

void Clear() {
    for (int i = 1; i <= n; ++ i) h[i] = edge[i] = 0;
    tot = 0;
    for (int i = 1; i <= n; ++ i) num[i][0] = num[i][1] = v[i][0] = v[i][1] = 0;
    for (int i = 1; i <= n; ++ i) d[i][0] = d[i][1] = INF;
}

int ver[M], ne[M], h[N], edge[M], tot;
void add(int x, int y, int w) {
```

```
    ver[++ tot] = y, ne[tot] = h[x], h[x] = tot;
    edge[tot] = w;
}

void dji() {
    priority_queue<PIII, vector<PIII>, greater<PIII> > q; q.push({0, s, 0});
    num[s][0] = 1; d[s][0] = 0;
    while (!q.empty()) {
        auto [dis, x, type] = q.top(); q.pop();
        if (v[x][type]) continue; v[x][type] = 1;
        for (int i = h[x]; i; i = ne[i]) {
            int y = ver[i], w = dis + edge[i];
            if (d[y][0] > w) {
                d[y][1] = d[y][0], num[y][1] = num[y][0];
                    // 如果找到新的最短路，将原有的最短路数据转化为次短路
                q.push({d[y][1], y, 1});
                d[y][0] = w, num[y][0] = num[x][type];
                q.push({d[y][0], y, 0});
            }
            else if (d[y][0] == w) num[y][0] += num[x][type];
            else if (d[y][1] > w) {
                d[y][1] = w, num[y][1] = num[x][type];
                q.push({d[y][1], y, 1});
            }
            else if (d[y][1] == w) num[y][1] += num[x][type];
        }
    }
}
void Solve() {
    cin >> n >> m >> s >> e;
    Clear(); //多组样例务必完全清空
    for (int i = 1; i <= m; ++ i) {
        int x, y, w; cin >> x >> y; w = 1;
        add(x, y, w), add(y, x, w);
    }
    dji();
    Z ans = num[e][0];
    if (d[e][1] == d[e][0] + 1) {
        ans += num[e][1]; // 只有在次短路满足条件时才计算（距离恰好比最短路大 1）
    }
    cout << ans.val() << endl;
}
```

判定图中是否存在负环

使用 SPFA ，复杂度为 $\mathcal{O}(KM)$ ，其中常数 K 相较裸的 SPFA 更高。

```
const int N = 1e5 + 7, M = 1e6 + 7;
int n, m;
int ver[M], ne[M], h[N], edge[M], tot;
int d[N], v[N], num[N];

void add(int x, int y, int w)  {
    ver[++ tot] = y, ne[tot] = h[x], h[x] = tot;
    edge[tot] = w;
}
bool spfa() {
    queue<int> q;
    for (int i = 1; i <= n; ++ i) q.push(i), v[i] = 1; //全部入队
    while(!q.empty()) {
        int x = q.front(); q.pop();
        v[x] = 0;
        for (int i = h[x]; i; i = ne[i]) {
            int y = ver[i];
            if(d[y] > d[x] + edge[i]) {
                num[y] = num[x] + 1;
                if (num[y] >= n) return true;
                d[y] = d[x] + edge[i];
                if(!v[y]) q.push(y), v[y] = 1;
            }
        }
```

```
    }
    return false;
}
int main() {
    cin >> n >> m;
    for (int i = 1; i <= m; ++ i) {
        int x, y, w; cin >> x >> y >> w;
        add(x, y, w);
    }
    if(spfa() == true) cout << "Yes" << endl;
    else cout << "No" << endl;
}
```

输出任意一个三元环

原题：给出一张有向完全图，输出任意一个三元环上的全部元素。使用 dfs，复杂度 $\mathcal{O}(N+M)$，可以扩展到非完全图和无向图。

```
int n;
cin >> n;
vector<vector<int>> a(n + 1, vector<int>(n + 1));
for (int i = 1; i <= n; ++i) {
    for (int j = 1; j <= n; ++j) {
        char x;
        cin >> x;
        if (x == '1') a[i][j] = 1;
    }
}

vector<int> vis(n + 1);
function<void(int, int)> dfs = [&](int x, int fa) {
    vis[x] = 1;
    for (int y = 1; y <= n; ++y) {
        if (a[x][y] == 0) continue;
        if (a[y][fa] == 1) {
            cout << fa << " " << x << " " << y;
            exit(0);
        }
        if (!vis[y]) dfs(y, x); // 这一步的 if 判断很关键
    }
};
for (int i = 1; i <= n; ++i) {
    if (!vis[i]) dfs(i, -1);
}
cout << -1;
```

带权最小环大小与计数

原题：给出一张有向带权图，求解图上最小环的长度、有多少个这样的最小环。使用 floyd，复杂度为 $\mathcal{O}(N^3)$，可以扩展到无向图。

```
LL Min = 1e18, ans = 0;
for (int k = 1; k <= n; k++) {
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            if (dis[i][j] > dis[i][k] + dis[k][j]) {
                dis[i][j] = dis[i][k] + dis[k][j];
                cnt[i][j] = cnt[i][k] * cnt[k][j] % mod;
            } else if (dis[i][j] == dis[i][k] + dis[k][j]) {
                cnt[i][j] = (cnt[i][j] + cnt[i][k] * cnt[k][j] % mod) % mod;
            }
        }
    }
    for (int i = 1; i < k; i++) {
        if (a[k][i]) {
            if (a[k][i] + dis[i][k] < Min) {
                Min = a[k][i] + dis[i][k];
                ans = cnt[i][k];
            } else if (a[k][i] + dis[i][k] == Min) {
                ans = (ans + cnt[i][k]) % mod;
            }
        }
```

```
    }
}
```

最小环大小

原题：给出一张无向图，求解图上最小环的长度、有多少个这样的最小环。使用 floyd，可以扩展到有向图。

```
int flody(int n) {
    for (int i = 1; i <= n; ++ i) {
        for (int j = 1; j <= n; ++ j) {
            val[i][j] = dis[i][j]; // 记录最初的边权值
        }
    }
    int ans = 0x3f3f3f3f;
    for (int k = 1; k <= n; ++ k) {
        for (int i = 1; i < k; ++ i) { // 注意这里是没有等于号的
            for (int j = 1; j < i; ++ j) {
                ans = min(ans, dis[i][j] + val[i][k] + val[k][j]);
            }
        }
        for (int i = 1; i <= n; ++ i) { // 往下是标准的 flody
            for (int j = 1; j <= n; ++ j) {
                dis[i][j] = min(dis[i][j], dis[i][k] + dis[k][j]);
            }
        }
    }
    return ans;
}
```

使用 bfs，复杂度为 $\mathcal{O}(N^2)$ 。

```
auto bfs = [&] (int s) {
    queue<int> q; q.push(s);
    dis[s] = 0;
    fa[s] = -1;
    while (q.size()) {
        auto x = q.front(); q.pop();
        for (auto y : ver[x]) {
            if (y == fa[x]) continue;
            if (dis[y] == -1) {
                dis[y] = dis[x] + 1;
                fa[y] = x;
                q.push(y);
            }
            else ans = min(ans, dis[x] + dis[y] + 1);
        }
    }
};
for (int i = 1; i <= n; ++ i) {
    fill(dis + 1, dis + 1 + n, -1);
    bfs(i);
}
cout << ans;
```

本质不同简单环计数

原题：给出一张无向图，输出简单环的数量。注意这里环套环需要分别多次统计，下图答案应当为 7。使用状压 dp，复杂度为 $\mathcal{O}(M \cdot 2^N)$，可以扩展到有向图。

```
int n, m;
cin >> n >> m;
vector<vector<int>> G(n);
for (int i = 0; i < m; i++) {
    int u, v;
    cin >> u >> v;
    u--, v--;
    G[u].push_back(v);
    G[v].push_back(u);
}
vector<vector<LL>> dp(1 << n, vector<LL>(n));
for (int i = 0; i < n; i++) dp[1 << i][i] = 1;
```

```
LL ans = 0;
for (int st = 1; st < (1 << n); st++) {
    for (int u = 0; u < n; u++) {
        if (!dp[st][u]) continue;
        int start = st & -st;
        for (auto v : G[u]) {
            if ((1 << v) < start) continue;
            if ((1 << v) & st) {
                if ((1 << v) == start) {
                    ans += dp[st][u];
                }
            } else {
                dp[st | (1 << v)][v] += dp[st][u];
            }
        }
    }
}
cout << (ans - m) / 2 << "\n";
```

输出任意一个非二元简单环

原题：给出一张无向图，不含自环与重边，输出任意一个简单环的大小以及其上面的全部元素。注意输出的环的大小是随机的，**不等价于最小环**。

由于不含重边与自环，所以环的大小至少为 3，使用 dfs 处理出 dfs 序，复杂度为 $\mathcal{O}(N+M)$，可以扩展到有向图；如果有向图中二元环也允许计入答案，则需要删除下方标注行。

```
vector<int> dis(n + 1, -1), fa(n + 1);
auto dfs = [&](auto self, int x) -> void {
    for (auto y : ver[x]) {
        if (y == fa[x]) continue; // 二元环需删去该行
        if (dis[y] == -1) {
            dis[y] = dis[x] + 1;
            fa[y] = x;
            self(self, y);
        } else if (dis[y] < dis[x]) {
            cout << dis[x] - dis[y] + 1 << endl;
            int pre = x;
            cout << pre << " ";
            while (pre != y) {
                pre = fa[pre];
                cout << pre << " ";
            }
            cout << endl;
            exit(0);
        }
    }
};
for (int i = 1; i <= n; i++) {
    if (dis[i] == -1) {
        dis[i] = 0;
        dfs(dfs, 1);
    }
}
```

有向图环计数

原题：给出一张有向图，输出环的数量。注意这里环套环仅需要计算一次，数据包括二元环和自环，下图例应当输出 3 个环。使用 dfs 染色法，复杂度为 $\mathcal{O}(N+M)$。

```
int ans = 0;
vector<int> vis(n + 1);
auto dfs = [&](auto self, int x) -> void {
    vis[x] = 1;
    for (auto y : ver[x]) {
        if (vis[y] == 0) {
            self(self, y);
        } else if (vis[y] == 1) {
            ans++;
        }
```

```
    }
    vis[x] = 2;
};
for (int i = 1; i <= n; i++) {
    if (!vis[i]) {
        dfs(dfs, i);
    }
}
cout << ans << endl;
```

输出有向图任意一个环

原题：给出一张有向图，输出任意一个环，数据包括二元环和自环。使用 dfs 染色法。

```
vector<int> dis(n + 1), vis(n + 1), fa(n + 1);
auto dfs = [&](auto self, int x) -> void {
    vis[x] = 1;
    for (auto y : ver[x]) {
        if (vis[y] == 0) {
            dis[y] = dis[x] + 1;
            fa[y] = x;
            self(self, y);
        } else if (vis[y] == 1) {
            cout << dis[x] - dis[y] + 1 << endl;
            int pre = x;
            cout << pre << " ";
            while (pre != y) {
                pre = fa[pre];
                cout << pre << " ";
            }
            cout << endl;
            exit(0);
        }
    }
    vis[x] = 2;
};
for (int i = 1; i <= n; i++) {
    if (!vis[i]) {
        dfs(dfs, i);
    }
}
```

判定带环图是否是平面图

原题：给定一个环以一些额外边，对于每一条额外边判定其位于环外还是环内，使得任意两条无重合顶点的额外边都不相交（即这张图构成平面图）使用 2-sat。考虑全部边都位于环内，那么“一条边完全包含另一条边”、“两条边完全没有交集”这两种情况都不会相交，可以直接跳过这两种情况的讨论。

```
signed main() {
    int n, m;
    cin >> n >> m;
    vector<pair<int, int>> in(m);
    for (int i = 0, x, y; i < m; i++) {
        cin >> x >> y;
        in[i] = minmax(x, y);
    }
    TwoSat sat(m);
    for (int i = 0; i < m; i++) {
        auto [s, e] = in[i];
        for (int j = i + 1; j < m; j++) {
            auto [S, E] = in[j];
            if (s < S && S < e && e < E || S < s && s < E && E < e) {
                sat.add(i, 0, j, 0);
                sat.add(i, 1, j, 1);
            }
        }
    }
    if (!sat.work()) {
        cout << "Impossible\n";
        return 0;
```

```cpp
    }
    auto ans = sat.answer();
    for (auto it : ans) {
        cout << (it ? "out" : "in") << " ";
    }
}
```

多项式

常用结论

杂

- 求 $B_i = \sum_{k=i}^{n} C_k^i A_k$，即 $B_i = \frac{1}{i!}\sum_{k=i}^{n}\frac{1}{(k-i)!}\cdot k!A_k$，反转后卷积。
- NTT 中，$\omega_n = $ `qpow(G,(mod-1)/n))`。
- 遇到 $\sum_{i=0}^{n}[i\%k=0]f(i)$ 可以转换为 $\sum_{i=0}^{n}\frac{1}{k}\sum_{j=0}^{k-1}(\omega_k^i)^j f(i)$ 。(单位根卷积)
- 广义二项式定理 $(1+x)^\alpha = \sum_{i=0}^{\infty}\binom{n}{\alpha}x^i$ 。

普通生成函数 / OGF

- 普通生成函数：$A(x) = a_0 + a_1x + a_2x^2 + ... = \langle a_0, a_1, a_2, ...\rangle$;
- $1 + x^k + x^{2k} + ... = \frac{1}{1-x^k}$;
- 取对数后 $= -\ln(1-x^k) = \sum_{i=1}^{\infty}\frac{1}{i}x^{ki}$ 即 $\sum_{i=1}^{\infty}\frac{1}{i}x^i \otimes x^k$（polymul_special）;
- $x + \frac{x^2}{2} + \frac{x^3}{3} + ... = -\ln(1-x)$;
- $1 + x + x^2 + ... + x^{m-1} = \frac{1-x^m}{1-x}$;
- $1 + 2x + 3x^2 + ... = \frac{1}{(1-x)^2}$（借用导数，$nx^{n-1} = (x^n)'$）;
- $C_m^0 + C_m^1x + C_m^2x^2 + ... + C_m^mx^m = (1+x)^m$（二项式定理）;
- $C_m^0 + C_{m+1}^1x^1 + C_{m+2}^2x^2 + ... = \frac{1}{(1-x)^{m+1}}$（归纳法证明）;
- $\sum_{n=0}^{\infty}F_nx^n = \frac{(F_1-F_0)x+F_0}{1-x-x^2}$（F 为斐波那契数列，列方程 $G(x) = xG(x) + x^2G(x) + (F_1-F_0)x + F_0$）;
- $\sum_{n=0}^{\infty}H_nx^n = \frac{1-\sqrt{n-4x}}{2x}$（H 为卡特兰数）;
- 前缀和 $\sum_{n=0}^{\infty}s_nx^n = \frac{1}{1-x}f(x)$;
- 五边形数定理：$\prod_{i=1}^{\infty}(1-x^i) = \sum_{k=0}^{\infty}(-1)^kx^{\frac{1}{2}k(3k\pm1)}$ 。

指数生成函数 / EGF

- 指数生成函数：$A(x) = a_0 + a_1x + a_2\frac{x^2}{2!} + a_3\frac{x^3}{3!} + ... = \langle a_0, a_1, a_2, a_3, ...\rangle$;
- 普通生成函数转换为指数生成函数：系数乘以 $n!$;
- $1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + ... = \exp x$;
- 长度为 n 的循环置换数为 $P(x) = -\ln(1-x)$，长度为 n 的置换数为 $\exp P(x) = \frac{1}{1-x}$（注意是**指数**生成函数）

- n 个点的生成树个数是 $P(x)=\sum_{n=1}^{\infty} n^{n-2}\frac{x^n}{n!}$，n 个点的生成森林个数是 $\exp P(x)$；
- n 个点的无向连通图个数是 $P(x)$，n 个点的无向图个数是 $\exp P(x)=\sum_{n=0}^{\infty} 2^{\frac{1}{2}n(n-1)}\frac{x^n}{n!}$；
- 长度为 $n(n\geq 2)$ 的循环置换数是 $P(x)=-\ln(1-x)-x$，长度为 n 的错排数是 $\exp P(x)$。

字符串

kmp

```
string s,t;cin>>s>>t;
    int n=s.size(),m=t.size();
    s=' '+s,t=' '+t;
    vi nxt(t.size());
    for(int i=2,j=0;i<=m;i++)
    {
        while(j&&t[j+1]!=t[i])j=nxt[j];
        if(t[j+1]==t[i])j++;
        nxt[i]=j;
    }
    for(int i=1,j=0;i<=n;i++)
    {
        while(j&&t[j+1]!=s[i])j=nxt[j];
        if(t[j+1]==s[i])j++;
        if(j==m)
        {
            cout<<i-m+1<<'\n';
            j=nxt[j];
        }
    }
    for(int i=1;i<=m;i++)cout<<nxt[i]<<" ";
```

ac

```
constexpr int MAXN=3e5+10;
int son[MAXN][26],nxt[MAXN],mmax[MAXN],tt;
inline void insert(char s[])
{
    int i=0,p=0;
    for(;s[i];i++)
    {
        int c=s[i]-'a';
        if(!son[p][c])son[p][c]=++tt;
        p=son[p][c];
    }
    mmax[p]=i;
}
inline void built()
{
    queue<int>q;
    for(int i=0;i<26;i++)if(son[0][i])q.push(son[0][i]);
    while(!q.empty())
    {
        int u=q.front();q.pop();
        for(int i=0;i<26;i++)
        {
            int&v=son[u][i];
            if(v){q.push(v),nxt[v]=son[nxt[u]][i];}
            else v=son[nxt[u]][i];
        }cmax(mmax[u],mmax[nxt[u]]);
    }
}
```

z 函数

```
int n=strlen(s+1);
    z[1]=n;
```

```
    for(int i=2,l=0,r=0;i<=n;i++)
    {
        int& p=z[i];
        if(i<=r)p=min(z[i-l+1],r-i+1);
        while(i+p<=n&&s[p+i]==s[p+1])p++;
        if(i+p-1>r)l=i,r=i+p-1;
    }

    int m=strlen(t+1);
    for(int i=1,l=0,r=0;i<=m;i++)
    {
        int& p=nxt[i];
        if(i<=r)p=min(z[i-l+1],r-i+1);
        while(i+p<=m&&t[p+i]==s[p+1])p++;
        if(i+p-1>r)l=i,r=i+p-1;
    }
```

sa

```
void get_sa(int n,int m)
{
    for(int i=1;i<=n;i++)cnt[x[i]=a[i]]++;
    for(int i=1;i<=m;i++)cnt[i]+=cnt[i-1];
    for(int i=n;i;i--)sa[cnt[x[i]]--]=i;
    for(int k=1;k<=n;k<<=1)
    {
        int p=0;
        for(int i=n-k+1;i<=n;i++)y[++p]=i;
        for(int i=1;i<=n;i++)if(sa[i]>k)y[++p]=sa[i]-k;
        for(int i=1;i<=m;i++)cnt[i]=0;
        for(int i=1;i<=n;i++)cnt[x[i]]++;
        for(int i=1;i<=m;i++)cnt[i]+=cnt[i-1];
        for(int i=n;i;i--)sa[cnt[x[y[i]]]--]=y[i],y[i]=0;
        swap(x,y);
         p=1;x[sa[1]]=1;
        for(int i=2;i<=n;i++)
            x[sa[i]]=(y[sa[i]]==y[sa[i-1]]&&y[sa[i]+k]==y[sa[i-1]+k])?p:++p;
        if(p==n)break;
        m=p;
    }
    for(int i=1;i<=n;i++)rk[sa[i]]=i;
    //for(int i=1;i<=n;i++)cerr<<rk[i]<<" ";
    for(int i=1,k=0;i<=n;i++)
    {
        if(rk[i]==1)continue;
        if(k)k--;
        int j=sa[rk[i]-1];
        while(i+k<=n&&j+k<=n&&a[i+k]==a[j+k])k++;
        hi[rk[i]]=k;
    }
}
```

sam

```
array<int,26> son[MAXN];
int fa[MAXN],len[MAXN];
int la=1,tt=1;
inline void insert(int c)
{
    int p=la,np=la=++tt;
    len[la]=len[p]+1;
    for(;p&&!son[p][c];p=fa[p])son[p][c]=np;
    if(!p)fa[np]=1;
    else
    {
        int q=son[p][c];
        if(len[q]==len[p]+1)fa[np]=q;
```

```
        else
        {
            int nq=++tt;
            son[nq]=son[q];
            fa[nq]=fa[q];
            len[nq]=len[p]+1;
            fa[np]=fa[q]=nq;
            for(;p&&son[p][c]==q;p=fa[p])son[p][c]=nq;
        }
    }
}
```

gsum

```
#include<bits/stdc++.h>
//#pragma GCC optimize("Ofast")
using namespace std;
template<typename T>
inline bool cmax(T&x,const T& y){return x<y?x=y,1:0;}
template<typename T>
inline bool cmin(T&x,const T& y){return y<x?x=y,1:0;}
typedef long long ll;
typedef pair<int,int> pii;
typedef pair<ll,ll> pll;
typedef vector<int> vi;
typedef vector<vector<int> > vii;
const int mod=998244353;
inline void MOD(int&x){x-=mod,x+=x>>31&mod;}
inline void MOD(ll& x){x-=mod,x+=x>>63&mod;}
inline int add(int x,int y){MOD(x+=y);return x;}
inline int mul(int x,int y){return 1ll*x*y%mod;}
template<typename ... A>inline int mul(const int& x,const A&... p){return 1ll*x*mul(p...)%mod;}
inline ll ksm(ll a,ll p=mod-2){ll ans=1;for(;p;p>>=1,a=a*a%mod)if(p&1)ans=ans*a%mod;return ans;}
typedef long double LD;
const int MAXN=2e6+10;
array<int,26>son[MAXN];
int fa[MAXN],len[MAXN],la=1,tt=1;
int insert(int c,int la)
{
    if(son[la][c]&&len[son[la][c]]==len[la]+1)return son[la][c];
    int p=la,np=++tt,flag=0;
    len[np]=len[p]+1;
    for(;p&&!son[p][c];p=fa[p])son[p][c]=np;
    if(!p){fa[np]=1;return np;}
    int q=son[p][c];
    if(len[q]==len[p]+1){fa[np]=q;return np;}
    if(p==la)flag=1,np=0,tt--;
    int nq=++tt;son[nq]=son[q],fa[nq]=fa[q],len[nq]=len[p]+1;
    fa[q]=fa[np]=nq;
    for(;p&&son[p][c]==q;p=fa[p])son[p][c]=nq;
    return flag?nq:np;
}
int main()
{
    ios::sync_with_stdio(0),cin.tie(0),cout.tie(0);cout<<fixed<<setprecision(10);
    int n;cin>>n;
    for(int i=1;i<=n;i++)
    {
        string s;cin>>s;
        la=1;
        for(auto p:s)la=insert(p-'a',la);
    }
    ll ans=0;
    for(int i=2;i<=tt;i++)ans+=len[i]-len[fa[i]];
    cout<<ans<<'\n';
    cout<<tt<<'\n';
    return 0;
}
```

pam

```
string s;
int son[MAXN][26],nxt[MAXN],len[MAXN],la=1,tt=1;
int cnt[MAXN];
inline int getfail(int p,int n)
{
    while(n-len[p]-1<=0||s[n-len[p]-1]!=s[n])p=nxt[p];
    return p;
}
inline int insert(char c,int n)
{
    c-='a';
    int p=getfail(la,n);
    if(!son[p][c])
    {
        nxt[++tt]=son[getfail(nxt[p],n)][c];
        son[p][c]=tt;
        len[tt]=len[p]+2;
        cnt[tt]=cnt[nxt[tt]]+1;
    }
    return cnt[la=son[p][c]];
}
int main()
{
    ios::sync_with_stdio(0),cin.tie(0),cout.tie(0);cout<<fixed<<setprecision(10);
    cin>>s;
    s=' '+s;
    nxt[0]=1,len[1]=-1;
    int la=0;
    for(int i=1;i<s.size();i++)
    {
        if(i>1)s[i]=(s[i]-'a'+la)%26+'a';
        la=insert(s[i],i);
        cout<<la<<' ';
    }

    return 0;
}
```

杂项

树上莫队

st[i] 为 i 加入到欧拉序的时间，ed[i] 为回溯时 i 加入欧拉序的时间设 st[x]<st[y]

若 lca(x,y)=x，则 x,y 在一条链上，在 st[x]~st[y] 这段区间中，有的点出现了两次，有的店没有出现过，这些点都是对答案无贡献的，我们只需要统计出现 1 次的点就好。

若 lca(x,y)!=x，此时 x,y 位于不同子树内，我们只需要按照上面的方法统计 ed[x]~st[y] 这段区间内的点。

同时，注意特判 lca。

```
const int N=300005;
int n,m;
vector<int> edge[N];
int fa[N],dep[N],siz[N],son[N];
int top[N],dfn[N],tot;
int rnk[N],Fir[N],Sec[N],cnt;
void dfs1(int x,int f){
```

```
    fa[x]=f,dep[x]=dep[f]+1;
    siz[x]=1,son[x]=-1;
    rnk[++cnt]=x,Fir[x]=cnt;
    for(int y:edge[x]){
        if(y==f)continue;
        dfs1(y,x);
        siz[x]+=siz[y];
        if(son[x]==-1||siz[y]>siz[son[x]])son[x]=y;
    }rnk[++cnt]=x,Sec[x]=cnt;
}
void dfs2(int x,int t){
    top[x]=t,dfn[x]=++tot;
    if(son[x]==-1)return;
    dfs2(son[x],t);
    for(int y:edge[x]){
        if(y==fa[x]||y==son[x])continue;
        dfs2(y,y);
    }
}
inline int Lca(int x,int y){
    int fx=top[x],fy=top[y];
    while(fx!=fy){
        if(dep[fx]>=dep[fy])x=fa[fx];
        else y=fa[fy];
        fx=top[x],fy=top[y];
    }
    if(dfn[x]<dfn[y])return x;
    else return y;
}
int bel[N];
int a[N],b[N],lsh;
int ans[N];
struct node{int l,r,lca,id;}q[N];
inline bool cmp(node a,node b){
    return (bel[a.l]^bel[b.l])?(bel[a.l]<bel[b.l]):((bel[a.l]&1)?a.r<b.r:a.r>b.r);
}
int vis[N],Cnt[N];
int now;
inline void cal(int pos){
    vis[pos]?(now-=!--Cnt[a[pos]]):(now+=!Cnt[a[pos]]++);
    vis[pos]^=1;
}
signed main()
{
    n=read(),m=read();
    For(i,1,n)a[i]=read(),b[i]=a[i];
    sort(b+1,b+n+1);
    lsh=unique(b+1,b+n+1)-(b+1);
    For(i,1,n)a[i]=lower_bound(b+1,b+lsh+1,a[i])-b;
    For(i,1,n-1){
        int u=read(),v=read();
        edge[u].push_back(v);
        edge[v].push_back(u);
    }dfs1(1,1);dfs2(1,1);
    int SZ=sqrt(2*n);For(i,1,2*n)bel[i]=i/SZ;
    For(i,1,m){
        int l=read(),r=read(),lca=Lca(l,r);
        if(Fir[l]>Fir[r])swap(l,r);
        if(l==lca)q[i]=node{Fir[l],Fir[r],0,i};
        else q[i]=node{Sec[l],Fir[r],lca,i};
    }sort(q+1,q+m+1,cmp);
    int l=1,r=0;
    For(i,1,m){
        int L=q[i].l,R=q[i].r,lca=q[i].lca;
        while(r<R)cal(rnk[++r]);
        while(r>R)cal(rnk[r--]);
        while(l>L)cal(rnk[--l]);
        while(l<L)cal(rnk[l++]);
        if(lca)cal(lca);
        ans[q[i].id]=now;
        if(lca)cal(lca);
```

```
    }
    For(i,1,m)write(ans[i],'\n');

    return 0;
}
```

带悔贪心

cyrcyr 今天在种树，他在一条直线上挖了 n 个坑。这 n 个坑都可以种树，但为了保证每一棵树都有充足的养料，cyrcyr 不会在相邻的两个坑中种树。而且由于 cyrcyr 的树种不够，他至多会种 k 棵树。假设 cyrcyr 有某种神能力，能预知自己在某个坑种树的获利会是多少（可能为负），请你帮助他计算出他的最大获利。

```
int n,k;
struct node{
    int x,id;
}a[N];
bool operator<(node A,node B){return A.x<B.x;}
priority_queue<node> q;
int L[N],R[N];
int vis[N];
inline void del(int x){
    L[x]=L[L[x]];
    R[x]=R[R[x]];
    R[L[x]]=x;
    L[R[x]]=x;
}
inline void work(){
    n=read(),k=read();
    For(i,1,n){
        a[i].x=read(),a[i].id=i;
        q.push(a[i]);
        L[i]=i-1; if(i!=n)R[i]=i+1;
    }
    int ans=0,res=0;
    For(i,1,k){
        while(vis[q.top().id])q.pop();
        node tt=q.top(); q.pop();
        res+=tt.x; ans=max(ans,res);
        int _L=L[tt.id],_R=R[tt.id];
        a[tt.id].x=a[_L].x+a[_R].x-tt.x;
        vis[_L]=vis[_R]=1;
        q.push(a[tt.id]);
        del(tt.id);
    }
    write(ans,'\n');
}
```

模拟费用流

有 $x+y+z$ 个人，第 i 个人有 A_i 个金币，B_i 个银币，C_i 个铜币。

要选出 x 个人获得其金币，选出 y 个人获得其银币，选出 z 个人获得其铜币。在不重复选某个人的情况下，最大化获得的币的总数。

$x+y+z \le 10^5$。

```
struct node{int x,id;};
inline bool operator<(const node &A,const node &B){
    if(A.x!=B.x)return A.x<B.x;
    return A.id<B.id;
}
int n;
int X,Y,Z;
int a[N],b[N],c[N];
priority_queue<node> q1,q2,q3;
int vis[N];
signed main()
{
    X=read(),Y=read(),Z=read();n=X+Y+Z;
    For(i,1,n)a[i]=read(),b[i]=read(),c[i]=read();
    int ans=0;
```

```
    For(i,1,n)ans+=a[i],b[i]-=a[i],c[i]-=a[i];
    For(i,1,n)q1.push(node{b[i],i});
    For(i,1,Y){
        node tt=q1.top();q1.pop();
        ans+=tt.x;vis[tt.id]=1;
    }
    // write(ans,'\n');
    For(i,1,n){
        if(vis[i])q3.push(node{c[i]-b[i],i});
        else q2.push(node{c[i],i});
    }
    For(i,1,Z){
        while(!q1.empty()&&vis[q1.top().id])q1.pop();
        while(!q2.empty()&&vis[q2.top().id])q2.pop();
        int v1,v2,v3;
        if(q1.empty())v1=-1e9; else v1=q1.top().x;
        if(q2.empty())v2=-1e9; else v2=q2.top().x;
        if(q3.empty())v3=-1e9; else v3=q3.top().x;
        // write(v1,' ',v2,' ',v3,'\n');
        if(v2>=v1+v3){
            ans+=v2;node tt=q2.top();q2.pop();
            vis[tt.id]=2;
        }else{
            ans+=v1+v3;node tt1=q1.top(),tt3=q3.top();
            q1.pop();q3.pop();vis[tt1.id]=vis[tt3.id]=1;
            q3.push(node{c[tt1.id]-b[tt1.id],tt1.id});
        }
    }
    // For(i,1,n)write(vis[i],' '); write('\n');
    write(ans,'\n');
    return 0;
}
```

把 N 个人派遣到 K 个城市，每个城市需要的人数是固定的。

把不同的人派遣到不同城市，代价都是不同的，求最小代价。n1e5 k10

```
int S,T;
vector<pair<int,int>> edge[21];
int dis[21],vis[21],pre[21];
inline void spfa(){
    queue<int> q;
    memset(dis,-0x3f,sizeof(dis));
    memset(vis,0,sizeof(vis));
    q.push(S);dis[S]=0,vis[S]=1;
    while(!q.empty()){
        int x=q.front();q.pop();
        vis[x]=0;
        for(auto yy:edge[x]){
            int y=yy.first,w=yy.second;
            if(dis[y]<dis[x]+w){
                dis[y]=dis[x]+w,pre[y]=x;
                if(!vis[y]){vis[y]=1;q.push(y);}
            }
        }
    }
}
int n,k;
int a[N];
int c[N][11];
priority_queue<pair<int,int>> q[11][11];
int id[11][11],bel[N];
inline void cl(int x,int y){
    while(!q[x][y].empty()&&bel[q[x][y].top().second]!=id[x][y])q[x][y].pop();
}
inline void update(int x,int p){
    bel[x]=p;
    For(i,1,k)if(i!=p)q[i][p].push(mk(c[x][i]-c[x][p],x));
}
int ans;
inline void work(){
    For(i,1,T)edge[i].clear();
```

```
    For(i,1,k)if(a[i])edge[S].push_back(mk(i,0));
    For(i,1,k){
        cl(i,i);if(!q[i][i].empty())edge[i].push_back(mk(T,q[i][i].top().first));
    }
    For(i,1,k)For(j,1,k)if(i!=j){
        cl(i,j);if(!q[i][j].empty())edge[i].push_back(mk(j,q[i][j].top().first));
    }spfa();ans+=dis[T];
    // exit(0);
    vector<int> vec;
    int x=T; while(x)vec.push_back(x),x=pre[x];
    // for(int pp:vec)write(pp,' ');
        // exit(0);
    for(int i=0;i+1<(signed)vec.size();++i){
        // write(i,'\n');
        if(vec[i]==T){
            int p=vec[i+1];
            cl(p,p);update(q[p][p].top().second,p);
        }else if(vec[i+1]==S){
            a[vec[i]]--;
        }else{
            cl(vec[i+1],vec[i]);
            update(q[vec[i+1]][vec[i]].top().second,vec[i+1]);
        }
    }
}
signed main()
{
    n=read(),k=read(),S=k+1,T=S+1;
    For(i,1,k)a[i]=read();
    For(i,1,k)For(j,1,k)if(i!=j)id[i][j]=j;
    For(i,1,n)For(j,1,k){
        c[i][j]=-read();
        q[j][j].push(mk(c[i][j],i));
    }
    For(i,1,n)work();
    write(-ans,'\n');
    return 0;;
}
```

约瑟夫问题

n 个人编号 $0, 1, 2..., n-1$，每次数到 k 出局，求最后剩下的人的编号。

$\mathcal{O}(N)$。

```
int jos(int n,int k){
    int res=0;
    repeat(i,1,n+1)res=(res+k)%i;
    return res; // res+1，如果编号从 1 开始
}
```

$\mathcal{O}(K\log N)$，适用于 K 较小的情况。

```
int jos(int n,int k){
    if(n==1 || k==1)return n-1;
    if(k>n)return (jos(n-1,k)+k)%n; // 线性算法
    int res=jos(n-n/k,k)-n%k;
    if(res<0)res+=n; // mod n
    else res+=res/(k-1); // 还原位置
    return res; // res+1，如果编号从 1 开始
}
```

日期换算（基姆拉尔森公式）

已知年月日，求星期数。

```
int week(int y,int m,int d){
    if(m<=2)m+=12,y--;
    return (d+2*m+3*(m+1)/5+y+y/4-y/100+y/400)%7+1;
}
```

博弈论

巴什博奕

有 N 个石子，两名玩家轮流行动，按以下规则取石子：

规定：每人每次可以取走 $X(1 \le X \le M)$ 个石子，拿到最后一颗石子的一方获胜。

双方均采用最优策略，询问谁会获胜。

两名玩家轮流报数。

规定：第一个报数的人可以报 $X(1 \le X \le M)$，后报数的人需要比前者所报数大 $Y(1 \le Y \le M)$，率先报到 N 的人获胜。

双方均采用最优策略，询问谁会获胜。

- $N = K \cdot (M+1)$（其中 $K \in \mathbb{N}^+$），后手必胜（后手可以控制每一回合结束时双方恰好取走 $M+1$ 个，重复 K 轮后即胜利）；
- $N = K \cdot (M+1) + R$（其中 $K \in \mathbb{N}^+, 0 < R < M+1$），先手必胜（先手先取走 R 个，之后控制每一回合结束时双方恰好取走 $M+1$ 个，重复 K 轮后即胜利）。

扩展巴什博弈

有 N 颗石子，两名玩家轮流行动，按以下规则取石子：。

规定：每人每次可以取走 $X(a \le X \le b)$ 个石子，如果最后剩余物品的数量小于 a 个，则不能再取，拿到最后一颗石子的一方获胜。

双方均采用最优策略，询问谁会获胜。

- $N = K \cdot (a+b)$ 时，后手必胜；
- $N = K \cdot (a+b) + R_1$（其中 $K \in \mathbb{N}^+, 0 < R_1 < a$）时，后手必胜（这些数量不够再取一次，先手无法逆转局面）；
- $N = K \cdot (a+b) + R_2$（其中 $K \in \mathbb{N}^+, a \le R_2 \le b$）时，先手必胜；
- $N = K \cdot (a+b) + R_3$（其中 $K \in \mathbb{N}^+, b < R_3 < a+b$）时，先手必胜（这些数量不够再取一次，后手无法逆转局面）；

Nim 博弈

有 N 堆石子，给出每一堆的石子数量，两名玩家轮流行动，按以下规则取石子：

规定：每人每次任选一堆，取走正整数颗石子，拿到最后一颗石子的一方获胜（注：几个特点是**不能跨堆**、**不能不拿**）。

双方均采用最优策略，询问谁会获胜。

记初始时各堆石子的数量 $(A_1, A_2, \dots, A_n)$，定义尼姆和 $Sum_N = A_1 \oplus A_2 \oplus \dots \oplus A_n$。

当 $Sum_N = 0$ 时先手必败，反之先手必胜。

Nim 游戏具体取法

先计算出尼姆和，再对每一堆石子计算 $A_i \oplus Sum_N$，记为 X_i。

若得到的值 $X_i < A_i$，X_i 即为一个可行解，即**剩下 X_i 颗石头，取走 $A_i - X_i$ 颗石头**（这里取小于号是因为至少要取走 1 颗石子）。

Moore’s Nim 游戏（Nim-K 游戏）

有 N 堆石子，给出每一堆的石子数量，两名玩家轮流行动，按以下规则取石子：

规定：每人每次任选不超过 K 堆，对每堆都取走不同的正整数颗石子，拿到最后一颗石子的一方获胜。

双方均采用最优策略，询问谁会获胜。

把每一堆石子的石子数用二进制表示，定义 One_i 为二进制第 i 位上 1 的个数。

以下局面先手必胜：

对于每一位，$One_1, One_2, \dots, One_N$ 均不为 $K+1$ 的倍数。

Anti-Nim 游戏（反 Nim 游戏）

有 N 堆石子，给出每一堆的石子数量，两名玩家轮流行动，按以下规则取石子：

规定：每人每次任选一堆，取走正整数颗石子，拿到最后一颗石子的一方**出局**。

双方均采用最优策略，询问谁会获胜。

- 所有堆的石头数量均不超过 1 ，且 $\boldsymbol{Sum_N = 0}$ （也可看作 “且有偶数堆”）；
- 至少有一堆的石头数量大于 1 ，且 $\boldsymbol{Sum_N \neq 0}$ 。

阶梯 - Nim 博弈

有 N 级台阶，每一级台阶上均有一定数量的石子，给出每一级石子的数量，两名玩家轮流行动，按以下规则操作石子：

规定：每人每次任选一级台阶，拿走正整数颗石子放到下一级台阶中，已经拿到地面上的石子不能再拿，拿到最后一颗石子的一方获胜。

双方均采用最优策略，询问谁会获胜。

对奇数台阶做传统 `Nim` 博弈，当 $\boldsymbol{Sum_N = 0}$** 时先手必败，反之先手必胜。**

SG 游戏（有向图游戏）

我们使用以下几条规则来定义暴力求解的过程：

- 使用数字来表示输赢情况，0 代表局面必败，非 0 代表**存在必胜可能**，我们称这个数字为这个局面的 SG 值；
- 找到最终态，根据题意人为定义最终态的输赢情况；
- 对于非最终态的某个节点，其 SG 值为所有子节点的 SG 值取 `mex` ；
- 单个游戏的输赢态即对应根节点的 SG 值是否为 0 ，为 0 代表先手必败，非 0 代表先手必胜；
- 多个游戏的总 SG 值为单个游戏 SG 值的异或和。

使用哈希表，以 $\mathcal{O}(N+M)$ 的复杂度计算。

```cpp
int n, m, a[N], num[N];
int sg(int x) {
    if (num[x] != -1) return num[x];

    unordered_set<int> S;
    for (int i = 1; i <= m; ++ i)
        if(x >= a[i])
            S.insert(sg(x - a[i]));

    for (int i = 0; ; ++ i)
        if (S.count(i) == 0)
            return num[x] = i;
}
void Solve() {
    cin >> m;
    for (int i = 1; i <= m; ++ i) cin >> a[i];
    cin >> n;

    int ans = 0; memset(num, -1, sizeof num);
    for (int i = 1; i <= n; ++ i) {
        int x; cin >> x;
        ans ^= sg(x);
    }

    if (ans == 0) no;
    else yes;
}
```

Anti-SG 游戏（反 SG 游戏）

SG 游戏中最先不能行动的一方获胜。

以下局面先手必胜：

- **单局游戏的 SG 值均不超过 1 ，且总 SG 值为 0；**

- **至少有一局单局游戏的 SG 值大于 1，且总 SG 值不为 0。**

在本质上，这与 Anti-Nim 游戏的结论一致。

Lasker’s-Nim 游戏（Multi-SG 游戏）

有 N 堆石子，给出每一堆的石子数量，两名玩家轮流行动，每人每次任选以下规定的一种操作石子：

- 任选一堆，取走正整数颗石子；
- 任选数量大于 2 的一堆，分成两堆非空石子。

拿到最后一颗石子的一方获胜。双方均采用最优策略，询问谁会获胜。

本题使用 SG 函数求解，SG 值定义为：

$$SG(x)=\begin{cases}x-1 & ,x \mod 4=0\\ x & ,x \mod 4=1\\ x & ,x \mod 4=2\\ x+1 & ,x \mod 4=3\end{cases}$$

Every-SG 游戏

给出一个有向无环图，其中 K 个顶点上放置了石子，两名玩家轮流行动，按以下规则操作石子：

移动图上所有还能够移动的石子；

无法移动石子的一方出局。双方均采用最优策略，询问谁会获胜。

定义 $step$ 为某一局游戏至多需要经过的回合数。

以下局面先手必胜：$step$ 为奇数。

威佐夫博弈

有两堆石子，给出每一堆的石子数量，两名玩家轮流行动，每人每次任选以下规定的一种操作石子：

- 任选一堆，取走正整数颗石子；
- 从两队中同时取走正整数颗石子。

拿到最后一颗石子的一方获胜。双方均采用最优策略，询问谁会获胜。

以下局面先手必败：

$(1,2),(3,5),(4,7),(6,10),\ldots$ 具体而言，每一对的第一个数为此前没出现过的最小整数，第二个数为第一个数加上 $1,2,3,4,\ldots$。

更一般地，对于第 k 对数，第一个数为 $First_k=\left\lfloor\frac{k*(1+\sqrt{5})}{2}\right\rfloor$，第二个数为 $Second_k=First_k+k$。

其中，在两堆石子的数量均大于 10^9 次时，由于需要使用高精度计算，我们需要人为定义 $\frac{1+\sqrt{5}}{2}$ 的取值为 $lorry=1.618033988749894848204586834$。

```cpp
const double lorry = (sqrt(5.0) + 1.0) / 2.0;
//const double lorry = 1.618033988749894848204586834;
void Solve() {
    int n, m; cin >> n >> m;
    if (n < m) swap(n, m);
    double x = n - m;
    if ((int)(lorry * x) == m) cout << "lose\n";
    else cout << "win\n";
}
```

斐波那契博弈

有一堆石子，数量为 N，两名玩家轮流行动，按以下规则取石子：

先手第 1 次可以取任意多颗，但不能全部取完，此后每人取的石子数不能超过上个人的两倍，拿到最后一颗石子的一方获胜。

双方均采用最优策略，询问谁会获胜。

当且仅当 N 为斐波那契数时先手必败。

```
int fib[100] = {1, 2};
map<int, bool> mp;
void Force() {
  for (int i = 2; i <= 86; ++ i) fib[i] = fib[i - 1] + fib[i - 2];
    for (int i = 0; i <= 86; ++ i) mp[fib[i]] = 1;
}
void Solve() {
    int n; cin >> n;
    if (mp[n] == 1) cout << "lose\n";
    else cout << "win\n";
}
```

树上删边游戏

给出一棵 N 个节点的有根树，两名玩家轮流行动，按以下规则操作：

选择任意一棵子树并删除（即删去任意一条边，不与根相连的部分会同步被删去）；

删掉最后一棵子树的一方获胜（换句话说，删去根节点的一方失败）。双方均采用最优策略，询问谁会获胜。

结论：当根节点 SG 值非 1 时先手必胜。

相较于传统 SG 值的定义，本题的 SG 函数值定义为：

- 叶子节点的 SG 值为 0 。
- 非叶子节点的 SG 值为其所有孩子节点 SG 值 $+1$ 的异或和。

```
auto dfs = [&](auto self, int x, int fa) -> int {
    int x = 0;
    for (auto y : ver[x]) {
        if (y == fa) continue;
        x ^= self(self, y, x);
    }
    return x + 1;
};
cout << (dfs(dfs, 1, 0) == 1 ? "Bob\n" : "Alice\n");
```

无向图删边游戏（Fusion Principle 定理）

给出一张 N 个节点的无向联通图，有一个点作为图的根，两名玩家轮流行动，按以下规则操作：

选择任意一条边删除，不与根相连的部分会同步被删去；

删掉最后一条边的一方获胜。双方均采用最优策略，询问谁会获胜。

- **对于奇环，我们将其缩成一个新点 + 一条新边；**
- **对于偶环，我们将其缩成一个新点；**
- **所有连接到原来环上的边全部与新点相连。**

此时，本模型转化为“树上删边游戏”。

gzy

缺生源

```
#include<bits/stdc++.h>
//#pragma GCC optimize("Ofast")
using namespace std;
template<typename T>
inline bool cmax(T&x,const T& y){return x<y?x=y,1:0;}
template<typename T>
inline bool cmin(T&x,const T& y){return y<x?x=y,1:0;}
typedef long long ll;
typedef pair<int,int> pii;
typedef pair<ll,ll> pll;
```

```
typedef vector<int> vi;
typedef vector<vector<int> > vii;
const int mod=998244353;
inline void MOD(int&x){x-=mod,x+=x>>31&mod;}
inline void MOD(ll& x){x-=mod,x+=x>>63&mod;}
inline int add(int x,int y){MOD(x+=y);return x;}
inline int mul(int x,int y){return 1ll*x*y%mod;}
template<typename ... A>inline int mul(const int& x,const A&... p){return 1ll*x*mul(p...)%mod;}
inline ll ksm(ll a,ll p=mod-2){ll ans=1;for(;p;p>>=1,a=a*a%mod)if(p&1)ans=ans*a%mod;return ans;}
typedef long double LD;
const int MAXN=2e5+10;
```

圆方树

```
const intMAXN=2e6+10;
struct edge{int from,v;}e[MAXN*2];
int head[MAXN],cnt,n,m;
inline void addd(int u,int v)
{e[++cnt]=(edge){head[u],v},head[u]=cnt;}
#define tree(u) for(int i=head[u],v=e[i].v;i;i=e[i].from,v=e[i].v)
vector<int>g[MAXN];
int tt,tot,S;
int dfn[MAXN],low[MAXN],w[MAXN],s[MAXN],top;
inline void add(int u,int v){g[u].push_back(v),g[v].push_back(u);}
void tarjan(int u,int fa)
{
    dfn[u]=low[u]=++tt,s[++top]=u,S++;w[u]=-1;
    tree(u)
    {
        if(!dfn[v])
        {
            tarjan(v,u);
            if(low[v]>=dfn[u])
            {
                tot++;
                add(u,tot);w[tot]=1;
                int x=0;
                do
                {
                    x=s[top--];w[tot]++;
                    add(x,tot);
                }while(x!=v);
            }
            cmin(low[u],low[v]);
        }
        else if(v!=fa)cmin(low[u],dfn[v]);
    }
```

李超线段树

```
 struct node{LD k,b;inline LD val(ll x){return k*x+b;}};
node f[MAXN];
int id[MAXN];
const LD eps=1e-6;
inline bool better(int x,int y,int p)
{
    auto a=f[x].val(p),b=f[y].val(p);
    if(a-b>eps)return 1;
    else if(fabs(a-b)<eps&&x<y)return 1;
    else return 0;
}
struct xds
{
    xds *ls,*rs;
    int l,r;
    int id;
    xds(int L,int R):l(L),r(R),id(0)
    {
        if(L==R)return;
        int mid=(L+R)>>1;
```

```
        ls=new xds(L,mid),rs=new xds(mid+1,R);
    }
    void update(int ql,int qr,int u)
    {
        int mid=(l+r)>>1;
        if(ql<=l&&r<=qr)
        {
            if(!id)return id=u,void();
            if(l==r)
            {
                if(better(u,id,l))id=u;
                return;
            }
            if(f[u].val(mid)>f[id].val(mid))swap(u,id);
            if(better(u,id,l)) ls->update(ql,qr,u);
            else if(better(u,id,r)) rs->update(ql,qr,u);
            return;
        }
        if(ql<=mid) ls->update(ql,qr,u);
        if(qr >mid) rs->update(ql,qr,u);

    }
    int ask(int qq)
    {
        if(l==r)return id;
        int mid=(l+r)>>1;
        int ans=qq<=mid?ls->ask(qq):rs->ask(qq);
        if(better(id,ans,qq))ans=id;

        return ans;
    }
};
```

斯坦纳树

```
vector<pii>e[MAXN];
int F[1<<10][MAXN];
bool is[MAXN];
signed main()
{
    ios::sync_with_stdio(0),cin.tie(0),cout.tie(0);
    int n,m,p;
    cin>>n>>m>>p;
    int u,v,w;
    for(int i=1;i<=m;i++)
    {
        cin>>u>>v>>w;
        e[u].emplace_back(v,w);
        e[v].emplace_back(u,w);
    }
    memset(F,0x3f,sizeof(F));
    for(int i=0;i<p;i++)
    {
        int x;cin>>x;
        F[1<<i][x]=0;
    }
    for(int s=1;s<(1<<p);s++)
    {
        for(int x=s&(s-1);x;x=s&(x-1))
            for(int i=1;i<=n;i++)
                cmin(F[s][i],F[x][i]+F[s^x][i]);
        queue<int>q;
        for(int i=1;i<=n;i++) q.push(i),is[i]=1;
        while(!q.empty())
        {
            int u=q.front();q.pop();is[u]=0;
            for(auto&p:e[u])if(cmin(F[s][p.X],F[s][u]+p.Y))
            {
                if(!is[p.X])q.push(p.X),is[p.X]=1;
            }
        }
```

```cpp
    }
    int ans=1e9;
    for(int i=1;i<=n;i++)cmin(ans,F[(1<<p)-1][i]);
    cout<<ans;
    return 0;
}
```

虚树

```cpp
const int MAXN=3e5+10;
vector<pii> e[MAXN];
vi g[MAXN];
const int K=20;
pii F[K+1][MAXN*2];
int tt;
int dfn[MAXN],d[MAXN];
int mmin[MAXN];
int h[MAXN*2];
void dfs(int u,int fa)
{
    dfn[u]=++tt;
    F[0][tt]={d[u],u};
    for(const pii&p:e[u])
    {
        int v=p.X;if(v!=fa)d[v]=d[u]+1,mmin[v]=min(mmin[u],p.Y),dfs(v,u),F[0][++tt]={d[u],u};
    }

}
inline int LCA(int x,int y)
{
    x=dfn[x],y=dfn[y];
    if(x>y)swap(x,y);int k=31-__builtin_clz(y-x+1);
    return min(F[k][x],F[k][y-(1<<k)+1]).Y;
}
int s[MAXN],top;
bool is[MAXN];
ll dp(int u)
{
    ll sum=0;
    for(const int&v:g[u])sum+=dp(v);
    if(u==1)
    {
        if(is[u])sum=mmin[u];
        g[u].clear(),is[u]=0;
        return sum;
    }
    if(is[u])sum=mmin[u];
    else cmin(sum,(ll)mmin[u]);
    g[u].clear(),is[u]=0;
    return sum;
}
signed main()
{
    ios::sync_with_stdio(0),cin.tie(0),cout.tie(0);
    int n,m;
    cin>>n;
    int u,v,w;
    for(int i=1;i<n;i++)cin>>u>>v>>w,e[u].emplace_back(v,w),e[v].emplace_back(u,w);
    mmin[1]=1e9;
    dfs(1,0);
    for(int i=1;i<=K;i++)
        for(int j=1;j+(1<<i)-1<=tt;j++)
            F[i][j]=min(F[i-1][j],F[i-1][j+(1<<(i-1))]);
    cin>>m;
    while(m--)
    {
        int k;cin>>k;
        for(int i=1;i<=k;i++)cin>>h[i],is[h[i]]=1;
        sort(h+1,h+k+1,[&](const int&x,const int& y){return dfn[x]<dfn[y];});
        s[top=1]=h[1];
        for(int i=2;i<=k;i++)
```

```
        {
            int u=h[i];
            int l=LCA(u,s[top]);
            while(1)
            {
                if(d[l]>=d[s[top-1]])
                {
                    if(l==s[top])break;
                    g[l].push_back(s[top]);
                    if(l!=s[top-1])s[top]=l;
                    else top--;
                    break;
                }
                else g[s[top-1]].push_back(s[top]),top--;
            }
            s[++top]=u;
        }
        while(top>1)g[s[top-1]].push_back(s[top]),top--;
        cout<<dp(s[1])<<'\n';
    }
    return 0;
}
```

支配树

```
const int MAXN=65534+10;
const int K=16;
int F[K+1][MAXN],d[MAXN];
vi e[MAXN];
inline int LCA(int u,int v)
{
    if(!u||!v)return u|v;
    if(d[u]<d[v])swap(u,v);
    for(int i=K;~i;i--)if(d[F[i][u]]>=d[v])u=F[i][u];
    if(u==v)return u;
    for(int i=K;~i;i--)if(F[i][u]!=F[i][v])u=F[i][u],v=F[i][v];
    return F[0][u];
}
int in[MAXN],siz[MAXN];
int main()
{
    ios::sync_with_stdio(0),cin.tie(0),cout.tie(0);cout<<fixed<<setprecision(10);
    int n;cin>>n;
    for(int i=1,x;i<=n;i++)
    {
        cin>>x;
        while(x)
        {
            e[x].push_back(i);
            in[i]++;
            cin>>x;
        }
    }
    for(int i=1;i<=n;i++)if(!in[i])in[i]=1,e[n+1].push_back(i);
    vi q;q.push_back(n+1);int h=0;
    while(h<q.size())
    {
        int u=q[h++];
        d[u]=d[F[0][u]]+1;
        for(int i=1;i<=K;i++)F[i][u]=F[i-1][F[i-1][u]];
        for(auto&v:e[u])
        {
            F[0][v]=LCA(F[0][v],u);
            if(!--in[v])q.push_back(v);
        }
    }
    for(int i=q.size()-1;~i;i--)
    {
        siz[F[0][q[i]]]+=siz[q[i]]+1;
    }
    for(int i=1;i<=n;i++)cout<<siz[i]<<'\n';
```

```
    return 0;
}
```

LCA

```
 vi e[MAXN];
int dfn[MAXN],tt,F[K+1][MAXN];
void dfs(int u,int fa)
{
    F[0][dfn[u]=++tt]=fa;
    for(auto&v:e[u])if(v!=fa)dfs(v,u);
}
inline int Min(int u,int v){return dfn[u]<dfn[v]?u:v;}
inline int LCA(int u,int v)
{
    if(u==v)return u;
    u=dfn[u],v=dfn[v];
    if(u>v)swap(u,v);
    int k=__lg(v-u++);
    return Min(F[k][u],F[k][v-(1<<k)+1]);
}
```

LCT

```
const int MAXN=3e5+10;
struct node
{
    int son[2],fa;
    bool rev;
    int w,mmin;
}t[MAXN];
#define ls t[p].son[0]
#define rs t[p].son[1]
inline bool isroot(int p){return p!=t[t[p].fa].son[0]&&p!=t[t[p].fa].son[1];}
const int INF=2e9;
int n,m,T;
inline void push_up(int p)
{
    t[p].mmin=p>n?t[p].w:INF;
    if(ls)cmin(t[p].mmin,t[ls].mmin);
    if(rs)cmin(t[p].mmin,t[rs].mmin);
}
inline void rev(int p){swap(ls,rs),t[p].rev^=1;}
inline void push_down(int p)
{
    if(t[p].rev)
    {
        if(ls)rev(ls);if(rs)rev(rs);
        t[p].rev=0;
    }
}
inline void rotate(int x)
{
    int y=t[x].fa,z=t[y].fa;
    if(!isroot(y)) t[z].son[y==t[z].son[1]]=x;t[x].fa=z;
    int r=x==t[y].son[1];z=t[x].son[r^1],t[x].son[r^1]=y;
    t[y].fa=x,t[y].son[r]=z,t[z].fa=y;push_up(y);
}
inline void splay(int p)
{
    static int s[MAXN],top;
    s[top=1]=p;int x=p;
    while(!isroot(p))s[++top]=p=t[p].fa;
    while(top)push_down(s[top--]);
    while(!isroot(x))
    {
        int y=t[x].fa;if(!isroot(y))rotate(((x==t[y].son[0])!=(y==t[t[y].fa].son[0]))?x:y);
        rotate(x);
    }push_up(x);
}
```

```
inline void access(int p)
{
    for(int i=0;p;p=t[i=p].fa)splay(p),rs=i,push_up(p);
}
inline void makeroot(int p){access(p),splay(p),rev(p);}
inline void link(int x,int y)
{
    makeroot(x),t[x].fa=y;
}
inline void cut(int x,int y)
{
    makeroot(x),access(y),splay(y);
    t[x].fa=0,t[y].son[0]=0;
}
```