

# Standard Code Library

stick16

Your Huazhong University of Science and Technology

November 29, 2024

# Contents

一切的开始	3
宏定义	3
数据结构	3
ST 表	3
fhq	4
数学	5
MinMax 容斥	5
exCRT	5
gauss	5
线性筛	6
二元一次方程 $Ax+By=C$	6
求解连续按位异或	7
Miller - Rabin 素数测试	7
Pollard - Rho 因式分解	8
数论常见结论及例题	8
常见结论	8
常见例题	11
图论	12
LCA	12
prim	13
图的连通性:	13
dcc:	14
有向图缩点:	15
二分图	16
欧拉路径/欧拉回路 Hierholzers	17
图论常见结论及例题	18
常见结论	18
常见例题	18
多项式	25
NTT	25
全家桶 (可能不全)	26
常用结论	29
杂	29
普通生成函数 / OGF	29
指数生成函数 / EGF	30
字符串	30
kmp	30
ac	31
z 函数	31
sa	31
sam	32
gsum	32
pam	33
杂项	34
树上莫队	34
带悔贪心	35
模拟费用流	36
约瑟夫问题	38
日期换算 (基姆拉尔森公式)	38
博弈论	38

巴什博弈 . . . . .	38
扩展巴什博弈 . . . . .	38
Nim 博弈 . . . . .	39
Nim 游戏具体取法 . . . . .	39
Moore' s Nim 游戏 (Nim-K 游戏) . . . . .	39
Anti-Nim 游戏 (反 Nim 游戏) . . . . .	39
阶梯 - Nim 博弈 . . . . .	39
SG 游戏 (有向图游戏) . . . . .	39
Anti-SG 游戏 (反 SG 游戏) . . . . .	40
Lasker' s-Nim 游戏 (Multi-SG 游戏) . . . . .	40
Every-SG 游戏 . . . . .	40
威佐夫博弈 . . . . .	41
斐波那契博弈 . . . . .	41
树上删边游戏 . . . . .	41
无向图删边游戏 (Fusion Principle 定理) . . . . .	42
<b>gzy</b> . . . . .	<b>42</b>
缺生源 . . . . .	42
圆方树 . . . . .	42
李超线段树 . . . . .	43
斯坦纳树 . . . . .	44
虚树 . . . . .	44
支配树 . . . . .	45
LCA . . . . .	46
LCT . . . . .	46

## 一切的开始

### 宏定义

- 需要 C++11

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  #define int long long
4  #define mid ((l+r)>>1)
5  // #define mod (998244353)
6  #define mod (1000000007)
7  #define ull unsigned long long
8  #define eps (1e-8)
9  #define mk make_pair
10 #define tim (double)clock()/CLOCKS_PER_SEC
11 #define For(i,a,b) for(int i=(a);i<=(b);++i)
12 #define rep(i,a,b) for(int i=(a);i>=(b);--i)
13 inline namespace IO{
14     inline int read(){
15         int x=0,f=1;char ch;
16         while((ch=getchar())<'0' || x>'9')if(ch=='-')f=-1;
17         while(ch>='0'&&ch<='9'){x=((x<<1)+(x<<3)+(ch^48)),ch=getchar();}
18         return x*f;
19     }
20     void write(char x){putchar(x);}
21     void write(const char *x){for(;;*x;++x)putchar(*x);}
22     void write(char *x){for(;;*x;++x)putchar(*x);}
23     void write(signed x){
24         if(x<0)putchar('-'),x=-x;
25         if(x>9)write(x/10); putchar('0'+x-x/10*10);
26     }
27     void write(long long x){
28         if(x<0)putchar('-'),x=-x;
29         if(x>9)write(x/10); putchar('0'+x-x/10*10);
30     }
31     void write(unsigned long long x){
32         if(x>9)write(x/10);
33         putchar('0'+x-x/10*10);
34     }
35     void write(double x){printf("%.5lf",x);}
36     template<typename type1,typename type2,typename ...typen>
37     void write(type1 a1,type2 a2,typen ...an){
38         write(a1);
39         write(a2,an...);
40     }
41 }using namespace IO;
42 inline int gcd(int x,int y){return y==0?x:gcd(y,x%y);}
43 inline int lcm(int x,int y){return x/gcd(x,y)*y;}
44 inline int lowbit(int x){return x&(-x);}
45 const int N=1900005;
```

## 数据结构

### ST 表

- 二维

```
1  int f[maxn][maxn][10][10];
2  inline int highbit(int x) { return 31 - __builtin_clz(x); }
3  inline int calc(int x, int y, int xx, int yy, int p, int q) {
4      return max(
5          max(f[x][y][p][q], f[xx - (1 << p) + 1][yy - (1 << q) + 1][p][q]),
6          max(f[xx - (1 << p) + 1][y][p][q], f[x][yy - (1 << q) + 1][p][q])
7      );
8  }
9  void init() {
10     FOR (x, 0, highbit(n) + 1)
11     FOR (y, 0, highbit(m) + 1)
12     FOR (i, 0, n - (1 << x) + 1)
```

```

13     FOR (j, 0, m - (1 << y) + 1) {
14         if (!x && !y) { f[i][j][x][y] = a[i][j]; continue; }
15         f[i][j][x][y] = calc(
16             i, j,
17             i + (1 << x) - 1, j + (1 << y) - 1,
18             max(x - 1, 0), max(y - 1, 0)
19         );
20     }
21 }
22 inline int get_max(int x, int y, int xx, int yy) {
23     return calc(x, y, xx, yy, highbit(xx - x + 1), highbit(yy - y + 1));
24 }

```

## fhq

```

1  mt19937 rnd(time(NULL)^clock());
2  int siz[N], pri[N], val[N];
3  int tot;
4  int ls[N], rs[N];
5  #define lc ls[now]
6  #define rc rs[now]
7  int root;
8  inline int New(int k){
9      int z=++tot;
10     val[z]=k, pri[z]=rnd(), siz[z]=1;
11     return z;
12 }
13 inline void push_up(int now){siz[now]=siz[lc]+siz[rc]+1;}
14 void split(int now, int k, int &x, int &y){
15     if(!now){x=y=0; return;}
16     if(val[now]<=k){x=now; split(rc, k, rc, y);}
17     else {y=now; split(lc, k, x, lc);}
18     push_up(now);
19 }
20 int merge(int x, int y){
21     if(!x||!y) return x|y;
22     if(pri[x]<pri[y]){rs[x]=merge(rs[x], y); push_up(x); return x;}
23     else {ls[y]=merge(x, ls[y]); push_up(y); return y;}
24 }
25 inline void insert(int k){
26     int x, y;
27     split(root, k, x, y);
28     root=merge(merge(x, New(k)), y);
29 }
30 inline void del(int k){
31     int x, y, z;
32     split(root, k-1, x, y);
33     split(y, k, y, z);
34     y=merge(ls[y], rs[y]);
35     root=merge(merge(x, y), z);
36 }
37 inline int getrnk(int k){
38     int x, y;
39     split(root, k-1, x, y);
40     int res=siz[x]+1;
41     root=merge(x, y);
42     return res;
43 }
44 inline int getkth(int now, int k){
45     while(1){
46         if(!now) return -1;
47         if(siz[lc]+1==k) return val[now];
48         if(siz[lc]>=k) now=lc;
49         else k-=siz[lc]+1, now=rc;
50     }
51 }
52 inline int getpre(int k){
53     int x, y;
54     split(root, k-1, x, y);
55     int res=getkth(x, siz[x]);
56     root=merge(x, y);

```

```

57     return res;
58 }
59 inline int getnxt(int k){
60     int x,y;
61     split(root,k,x,y);
62     int res=getkth(y,1);
63     root=merge(x,y);
64     return res;
65 }

```

## 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

```

1  int n,m;
2  int a[N],b[N];
3  int exgcd(int a,int b,int &x,int &y){
4      if(b==0){x=1,y=0;return a;}
5      int d=exgcd(b,a%b,x,y);
6      int z=x;x=y,y=z-a/b*y;
7
8      return d;
9  }
10 inline int mul(int a,int b,int mod){
11     int res=0;
12     for(;b>=>=1){
13         if(b&1)res=(res+a)%mod;
14         a=(a<<1)%mod;
15     }
16     return res;
17 }
18 inline int excrt(){
19     int ans=b[1];m=a[1];
20     For(i,2,n){
21         int t,y,c=((b[i]-ans)%a[i]+a[i])%a[i];
22         int d=exgcd(m,a[i],t,y);
23         // t=t*(c/d);
24         t=mul(t,c/d,a[i]);
25         ans+=t*m;
26         m=lcm(m,a[i]);
27         ans=(ans%m+m)%m;
28     }
29     int res=(ans%m+m)%m;
30     return res;
31 }
32 signed main()
33 {
34     n=read();
35     For(i,1,n)a[i]=read(),b[i]=read();
36     write(excrt());
37     return 0;
38 }
39 /*
40 x=b[i](mod a[i])
41 */

```

### gauss

```

1  int n,m;
2  double a[N][N];

```

```

3 inline void gauss(){
4     For(i,1,n){
5         int F=0;
6         For(j,i,n)if(a[j][i]){
7             For(k,1,m)swap(a[i][k],a[j][k]);
8             F=1; break;
9         }
10        if(!F)For(j,1,i-1)if(a[j][i]&&a[i][i]==0){
11            For(k,1,m)swap(a[i][k],a[j][k]);
12            F=1; break;
13        }
14        if(!F)continue;
15        double tt=a[i][i];
16        For(k,i,m)a[i][k]/=tt;
17        For(j,1,n)if(j!=i){
18            tt=a[j][i];
19            For(k,i,m)a[j][k]-=tt*a[i][k];
20        }
21    }
22    For(i,1,n)if(a[i][i]==0&&a[i][m]){write(-1,'\n');return;}
23    For(i,1,n)if(a[i][i]==0&&a[i][m]==0){write(0,'\n');return;}
24    For(i,1,n)printf("x%lld=%0.2lf\n",i,a[i][m]);
25 }
26 signed main()
27 {
28     n=read(),m=n+1;
29     For(i,1,n)For(j,1,m)cin>>a[i][j];
30     gauss();
31     return 0;
32 }
33 /*
34 a[1][1]*x1+a[1][2]*x2+...+a[1][n]*xn=b1
35 a[2][1]*x1+a[2][2]*x2+...+a[2][n]*xn=b2
36 无解-1
37 无穷解 0
38 */

```

## 线性筛

```

1 phi[1]=1,mu[1]=1;
2 For(i,2,N-1){
3     if(!vis[i]){p[++tot]=i,phi[i]=i-1,mu[i]=-1;}
4     for(int j=1;j<=tot&&i*p[j]<N;++j){
5         vis[i*p[j]]=1;
6         if(i%p[j]!=0){
7             phi[i*p[j]]=phi[i]*(p[j]-1);
8             mu[i*p[j]]=-mu[i];
9         }else{
10            phi[i*p[j]]=phi[i]*p[j],mu[i*p[j]]=0;
11            break;
12        }
13    }
14 }

```

## 二元一次方程 $Ax+By=C$

```

1 auto clac = [&](int a, int b, int c) {
2     int u = 1, v = 1;
3     if (a < 0) { // 负数特判, 但是没用经过例题测试
4         a = -a;
5         u = -1;
6     }
7     if (b < 0) {
8         b = -b;
9         v = -1;
10    }
11
12    int x, y, d = exgcd(a, b, x, y), ans;
13    if (c % d != 0) { // 无整数解
14        cout << -1 << "\n";
15    }
16 }

```

```

15     return;
16 }
17 a /= d, b /= d, c /= d;
18 x *= c, y *= c; // 得到可行解
19
20 ans = (x % b + b - 1) % b + 1;
21 auto [A, B] = pair{u * ans, v * (c - ans * a) / b}; // x 最小正整数 特解
22
23 ans = (y % a + a - 1) % a + 1;
24 auto [C, D] = pair{u * (c - ans * b) / a, v * ans}; // y 最小正整数 特解
25
26 int num = (C - A) / b + 1; // xy 均为正整数 的 解的组数
27 };

```

## 求解连续按位异或

$O(1)$   $0^12^{\dots n}$

```

1 unsigned xor_n(unsigned n) {
2     unsigned t = n & 3;
3     if (t & 1) return t / 2u ^ 1;
4     return t / 2u ^ n;
5 }
6
1 i64 xor_n(i64 n) {
2     if (n % 4 == 1) return 1;
3     else if (n % 4 == 2) return n + 1;
4     else if (n % 4 == 3) return 0;
5     else return n;
6 }

```

## Miller - Rabin 素数测试

$O(4\log^3 x)$ , 这里可以看错  $O(1)$

```

1 int mul(int a, int b, int m) {
2     int r = a * b - m * (int)(1.L / m * a * b);
3     return r - m * (r >= m) + m * (r < 0);
4 }
5 int mypow(int a, int b, int m) {
6     int res = 1 % m;
7     for (; b; b >>= 1, a = mul(a, a, m)) {
8         if (b & 1) {
9             res = mul(res, a, m);
10        }
11    }
12    return res;
13 }
14
15 int B[] = {2, 3, 5, 7, 11, 13, 17, 19, 23};
16 bool MR(int n) {
17     if (n <= 1) return 0;
18     for (int p : B) {
19         if (n == p) return 1;
20         if (n % p == 0) return 0;
21     }
22     int m = (n - 1) >> __builtin_ctz(n - 1);
23     for (int p : B) {
24         int t = m, a = mypow(p, m, n);
25         while (t != n - 1 && a != 1 && a != n - 1) {
26             a = mul(a, a, n);
27             t *= 2;
28         }
29         if (a != n - 1 && t % 2 == 0) return 0;
30     }
31     return 1;
32 }

```



## Pollard - Rho 因式分解

以单个因子  $O(\log X)$  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

```
1 int PR(int n) {
2     for (int p : B) {
3         if (n % p == 0) return p;
4     }
5     auto f = [&](int x) -> int {
6         x = mul(x, x, n) + 1;
7         return x >= n ? x - n : x;
8     };
9     int x = 0, y = 0, tot = 0, p = 1, q, g;
10    for (int i = 0; (i & 255) || (g = gcd(p, n)) == 1; i++, x = f(x), y = f(f(y))) {
11        if (x == y) {
12            x = tot++;
13            y = f(x);
14        }
15        q = mul(p, abs(x - y), n);
16        if (q) p = q;
17    }
18    return g;
19 }
20 vector<int> fac(int n) {
21     #define pb emplace_back
22     if (n == 1) return {};
23     if (MR(n)) return {n};
24     int d = PR(n);
25     auto v1 = fac(d), v2 = fac(n / d);
26     auto i1 = v1.begin(), i2 = v2.begin();
27     vector<int> ans;
28     while (i1 != v1.end() || i2 != v2.end()) {
29         if (i1 == v1.end()) {
30             ans.pb(*i2++);
31         } else if (i2 == v2.end()) {
32             ans.pb(*i1++);
33         } else {
34             if (*i1 < *i2) {
35                 ans.pb(*i1++);
36             } else {
37                 ans.pb(*i2++);
38             }
39         }
40     }
41     return ans;
42 }
```

## 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  $\binom{n-1}{m-1}$ 。

- 球同，盒不同、能空

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

- 球同，盒同、能空

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

- 球同，盒同、不能空

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

- 球不同，盒同、不能空

第二类斯特林数  $\text{Stirling2}(n, m)$  - 球不同，盒同、能空

第二类斯特林数之和  $\sum_{i=1}^m \text{Stirling2}(n, m)$ ，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  $m! \cdot \text{Stirling2}(n, m)$ ，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  $[a_1, a_2, \dots, a_n]$ ，有： $\mathbf{X} = (\mathbf{X} \& a_1) \text{or} (\mathbf{X} \& a_2) \text{or} \dots \text{or} (\mathbf{X} \& a_n)$ 。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

1  $\log(n) + 0.5772156649 + 1.0 / (2 * n)$

## 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  $n = \prod_{i=1}^m p_i^{c_i}$ ，那么  $f(n) = \prod_{i=1}^m f(p_i^{c_i})$ 。
- 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  $\varphi(n) = \varphi(n/p) * (p - 1)$ 。
- $\sum_{d \mid n} \varphi(d) = n$ 。如  $n = 10$ ，则  $d = 10/5/2/1$ ，那么  $10 = \varphi(10) + \varphi(5) + \varphi(2) + \varphi(1)$ 。
- $\sum_{i=1}^n \gcd(i, n) = \sum_{d \mid n} \left\lfloor \frac{n}{d} \right\rfloor \varphi(d)$ （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： $\sum_{i=1}^n \sum_{j=i+1}^n (a_i b_j - a_j b_i)^2 = (\sum_{i=1}^n a_i)^2 (\sum_{i=1}^n b_i)^2 - (\sum_{i=1}^n a_i b_i)^2$ 。

## 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】 $1, 2, \dots, n$  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  $a_0 = 1, a_n = a_{n-1} + a_{n-3} + a_{n-5} + \dots (n \geq 1)$  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:  $F_1 + F_3 + F_5 + \dots + F_{2n-1} = F_{2n}$  ;
- 偶数项求和:  $F_2 + F_4 + F_6 + \dots + F_{2n} = F_{2n+1} - 1$  ;
- 平方和:  $F_1^2 + F_2^2 + F_3^2 + \dots + F_n^2 = F_n * F_{n+1}$  ;
- $F_1 + 2F_2 + 3F_3 + \dots + nF_n = nF_{n+2} - F_{n+3} + 2$  ;
- $-F_1 + F_2 - F_3 + \dots + (-1)^n F_n = (-1)^n (F_{n+1} - F_n) + 1$  ;
- $F_{2n-2m-2} (F_{2n} + F_{2n+2}) = F_{2m+2} + F_{4n-2m}$  。

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  $\gcd(x, y) = \gcd(x, y - x)$ ，推广到  $N$  项为  $\gcd(a_1, a_2, \dots, a_N) = \gcd(a_1, a_2 - a_1, \dots, a_N - a_{N-1})$ ，  
- 该推论在“四则运算后 gcd”这类题中有特殊意义，如求解  $\gcd(a_1 + X, a_2 + X, \dots, a_N + X)$  时 See；
- 以下式子成立： $\gcd(a, m) = \gcd(a + x, m) \Leftrightarrow \gcd(a, m) = \gcd(x, m)$ 。求解上式满足条件的  $x$  的数量即为求比  $\frac{m}{\gcd(a, m)}$  小且与其互质的数的个数，即用欧拉函数求解  $\varphi\left(\frac{m}{\gcd(a, m)}\right)$ 。
- 已知序列  $a$ ，定义集合  $S = \{a_i \cdot a_j \mid i < j\}$ ，现在要求解  $\gcd(S)$ ，即为求解  $\gcd(a_j, \gcd(a_i \mid i < j))$ ，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， $\begin{cases} n, n-1, n-3 \text{ 互质} & \gcd(n, n-3) = 1 \text{ 时} \\ n-1, n-2, n-3 \text{ 互质} & \gcd(n, n-3) \neq 1 \text{ 时} \end{cases}$  See；
- 由  $a \bmod b = (b + a) \bmod b = (2 \cdot b + a) \bmod b = \dots = (K \cdot b + a) \bmod b$  可以推广得到  $(a \bmod b) \bmod c = ((K \cdot bc + a) \bmod b) \bmod c$ ，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  $f(p, q) = \sum_{i=1}^n |p_i - q_i|$ ，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  $\frac{X - Y}{2}$  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  $\{\frac{X - Y}{2}, \frac{X - Y}{2}, Y\}$  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  $T(p) = \sum_{i=1}^n T(a_i)$  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： $a + b \equiv a \oplus b \pmod{2}$  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  $\frac{a_i}{2^k}$ ，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， $\{1, 0, 0, 1\} \rightarrow \{0, 2, 0, 1\} \rightarrow \{0, 1, 2, 1\}$ ，即答案为 0 个  $2^3$ 、1 个  $2^2$ 、.....。

```
1 signed main() {
2     int n, k;
3     cin >> n >> k;
4
5     int cnt = __builtin_popcountll(n);
6
7     if (k < cnt || n < k) {
8         cout << "NO\n";
9         return 0;
10    }
11    cout << "YES\n";
12
13    vector<int> num;
14    while (n) {
15        num.push_back(n % 2);
16        n /= 2;
17    }
18
19    for (int i = num.size() - 1; i > 0; i--) {
20        int p = min(k - cnt, num[i]);
21        num[i] -= p;
22        num[i - 1] += 2 * p;
23        cnt += p;
24    }
25
26    for (int i = 0; i < num.size(); i++) {
27        for (int j = 1; j <= num[i]; j++) {
28            cout << (1LL << i) << " ";
29        }
30    }
31 }
```

---

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

```
1 vector<int> edge[N];
2 int fa[N], dep[N], siz[N], son[N];
3 int top[N], dfn[N], tot;
4 void dfs1(int x, int f) {
5     fa[x] = f, dep[x] = dep[f] + 1;
6     siz[x] = 1, son[x] = -1;
7     for (int y: edge[x]) {
8         if (y == f) continue;
9         dfs1(y, x);
10        siz[x] += siz[y];
11        if (son[x] == -1 || siz[y] > siz[son[x]]) son[x] = y;
12    }
```

```

13 }
14 void dfs2(int x,int t){
15     dfn[x]=++tot;
16     top[x]=t;
17     if(son[x]==-1)return;
18     dfs2(son[x],t);
19     for(int y:edge[x]){
20         if(y==fa[x]||y==son[x])continue;
21         dfs2(y,y);
22     }
23 }
24 inline int Lca(int x,int y){
25     int fx=top[x],fy=top[y];
26     while(fx!=fy){
27         if(dep[fx]>=dep[fy])x=fa[fx];
28         else y=fa[fy];
29         fx=top[x],fy=top[y];
30     }
31     if(dfn[x]<dfn[y])return x;
32     else return y;
33 }

```

## prim

```

1  const int N = 550, INF = 0x3f3f3f3f;
2  int n, m, g[N][N];
3  int d[N], v[N];
4  int prim() {
5      ms(d, 0x3f); //这里的 d 表示到“最小生成树集合”的距离
6      int ans = 0;
7      for (int i = 0; i < n; ++ i) { //遍历 n 轮
8          int t = -1;
9          for (int j = 1; j <= n; ++ j)
10             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
11                 t = j;
12             v[t] = 1; //将 t 加入“最小生成树集合”
13             if (i && d[t] == INF) return INF; //如果发现不连通，直接返回
14             if (i) ans += d[t];
15             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
16         }
17         return ans;
18     }
19     int main() {
20         ms(g, 0x3f); cin >> n >> m;
21         while (m -- ) {
22             int x, y, w; cin >> x >> y >> w;
23             g[x][y] = g[y][x] = min(g[x][y], w);
24         }
25         int t = prim();
26         if (t == INF) cout << "impossible" << endl;
27         else cout << t << endl;
28     } //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, 在不超过边  $(x, y)$  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, 在不超过  $(x, y)$  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:**

```

1  int n,m;
2  vector<pair<int,int>> edge[N];
3  int dfn[N],low[N],tot;
4  int vis[N],dcc[N],cnt;
5  vector<int> vec[N];
6  void dfs(int x,int ID){
7      dfn[x]=low[x]=++tot;
8      for(auto yy:edge[x]){
9          int y=yy.first,id=yy.second;
10         if(id==ID)continue;
11         if(!dfn[y]){
12             dfs(y,id);
13             low[x]=min(low[x],low[y]);
14         }else low[x]=min(low[x],dfn[y]);
15         if(low[y]>dfn[x])vis[id]=1;

```

```

16     }
17 }
18 void dfsc(int x,int ID){
19     dcc[x]=cnt; vec[cnt].push_back(x);
20     for(auto yy:edge[x]){
21         int y=yy.first,id=yy.second;
22         if(id==ID||vis[id])continue;
23         if(!dcc[y])dfsc(y,id);
24     }
25 }
26 signed main()
27 {
28     n=read(),m=read();
29     For(i,1,m){
30         int u=read(),v=read();
31         edge[u].push_back(mk(v,i));
32         edge[v].push_back(mk(u,i));
33     }
34     For(i,1,n)if(!dfn[i])dfs(i,0);
35     For(i,1,n)if(!dcc[i])cnt++,dfsc(i,0);
36     write(cnt,'\n');
37     For(i,1,cnt){
38         write((signed)vec[i].size(),' ');
39         for(int x:vec[i])write(x,' ');
40         write('\n');
41     }
42     return 0;
43 }

```

有向图缩点:

```

1  int n,m;
2  int a[N];
3  int dfn[N],low[N],tot;
4  int stk[N],vis[N],top;
5  int col[N],V[N],cnt;
6  vector<int> edge[N],e[N];
7  void dfs(int x){
8      dfn[x]=low[x]=++tot;
9      stk[++top]=x,vis[x]=1;
10     for(int y:edge[x]){
11         if(!dfn[y]){
12             dfs(y);
13             low[x]=min(low[x],low[y]);
14         }else if(vis[y])low[x]=min(low[x],dfn[y]);
15     }
16     if(dfn[x]==low[x]){
17         ++cnt;
18         while(stk[top]!=x){
19             int p=stk[top--];
20             V[cnt]+=a[p];
21             col[p]=cnt,vis[p]=0;
22         }int p=stk[top--];
23         V[cnt]+=a[p];
24         col[p]=cnt,vis[p]=0;
25     }
26 }
27 int deg[N];
28 int dp[N];
29 signed main()
30 {
31     n=read(),m=read();
32     For(i,1,n)a[i]=read();
33     For(i,1,m){
34         int u=read(),v=read();
35         edge[u].push_back(v);
36         // edge[v].push_back(u);
37     }
38     For(i,1,n)if(!dfn[i])dfs(i);
39     For(i,1,n)for(int y:edge[i]){
40         if(col[i]==col[y])continue;

```



```

41         e[col[i]].push_back(col[y]);
42         deg[col[y]]++;
43     }
44     queue<int> q;
45     For(i,1,cnt)if(!deg[i])q.push(i),dp[i]=V[i];
46     while(!q.empty()){
47         int x=q.front();q.pop();
48         for(int y:e[x]){
49             deg[y]--;dp[y]=max(dp[y],dp[x]+V[y]);
50             if(!deg[y])q.push(y);
51         }
52     }
53     int res=0;
54     For(i,1,cnt)res=max(res,dp[i]);
55     write(res,'\n');
56     return 0;
57 }
58 /*
59
60 */

```

## 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

```

1  struct HopcroftKarp {
2      int n, m;
3      vector<array<int, 2>> ver;
4      vector<int> l, r;
5
6      HopcroftKarp(int n, int m) : n(n), m(m) { // 左右半部
7          l.assign(n, -1);
8          r.assign(m, -1);
9      }
10     void add(int x, int y) {
11         x--, y--; // 这个板子是 0-idx 的
12         ver.push_back({x, y});
13     }
14     int work() {
15         vector<int> adj(ver.size());
16
17         mt19937 rgen(chrono::steady_clock::now().time_since_epoch().count());
18         shuffle(ver.begin(), ver.end(), rgen); // 随机化防卡
19
20         vector<int> deg(n + 1);
21         for (auto &[u, v] : ver) {
22             deg[u]++;
23         }
24         for (int i = 1; i <= n; i++) {
25             deg[i] += deg[i - 1];
26         }
27         for (auto &[u, v] : ver) {
28             adj[--deg[u]] = v;
29         }
30
31         int ans = 0;
32         vector<int> a, p, q(n);
33         while (true) {
34             a.assign(n, -1), p.assign(n, -1);
35
36             int t = 0;
37             for (int i = 0; i < n; i++) {
38                 if (l[i] == -1) {
39                     q[t++] = a[i] = p[i] = i;
40                 }
41             }

```

```

42
43     bool match = false;
44     for (int i = 0; i < t; i++) {
45         int x = q[i];
46         if (~l[a[x]]) continue;
47
48         for (int j = deg[x]; j < deg[x + 1]; j++) {
49             int y = adj[j];
50             if (r[y] == -1) {
51                 while (~y) {
52                     r[y] = x;
53                     swap(l[x], y);
54                     x = p[x];
55                 }
56                 match = true;
57                 ++ans;
58                 break;
59             }
60             if (p[r[y]] == -1) {
61                 q[t++] = y = r[y];
62                 p[y] = x;
63                 a[y] = a[x];
64             }
65         }
66         if (!match) break;
67     }
68     return ans;
69 }
70
71 vector<array<int, 2>> answer() {
72     vector<array<int, 2>> ans;
73     for (int i = 0; i < n; i++) {
74         if (~l[i]) {
75             ans.push_back({i, l[i]});
76         }
77     }
78     return ans;
79 }
80 };
81
82 signed main() {
83     int n1, n2, m;
84     cin >> n1 >> n2 >> m;
85     HopcroftKarp flow(n1, n2);
86     while (m--) {
87         int x, y;
88         cin >> x >> y;
89         flow.add(x, y);
90     }
91
92     cout << flow.work() << "\n";
93
94     auto match = flow.answer();
95     for (auto [u, v] : match) {
96         cout << u << " " << v << "\n";
97     }
98 }

```

## 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）：

```

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

```

```

7     }
8     ans.push_back(x);
9 };
10 dfs(dfs, s);
11 reverse(ans.begin(), ans.end());
12 for (auto it : ans) {
13     cout << it << " ";
14 }

```

## 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  $\left\lfloor \frac{n \cdot (n + 1)}{2} \right\rfloor$ ，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  $\deg[i] \leq 1$  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。

```

1 vector<int> val(n + 1, 1);
2 int ans = 0;
3 function<void(int, int)> dfs = [&](int x, int fa) {
4     for (auto y : ver[x]) {
5         if (y == fa) continue;
6         dfs(y, x);
7         val[x] += val[y];
8         ans += min(val[y], k - val[y]);
9     }
10 };
11 dfs(1, 0);
12 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  $cnt_{\text{in}}, cnt_{\text{out}}$ ，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。

```
1 LL n, point[N];
2 LL ver[N], head[N], nex[N], tot; bool v[N];
3 map<pair<LL, LL>, LL> edge;
4 // void add(LL x, LL y) {}
5 void dfs(LL x) {
6     for (LL i = head[x]; i; i = nex[i]) {
7         LL y = ver[i];
8         if (v[y]) continue;
9         v[y] = true; dfs(y); v[y] = false;
10    }
11    for (LL i = head[x]; i; i = nex[i]) {
12        LL y = ver[i];
13        if (v[y]) continue;
14        point[x] += max(point[y] + edge[{x, y}], 0LL);
15    }
16 }
17 void Solve() {
18     cin >> n;
19     FOR(i, 1, n) cin >> point[i];
20     FOR(i, 2, n) {
21         LL x, y, w; cin >> x >> y >> w;
22         edge[{x, y}] = edge[{y, x}] = w;
23         add(x, y), add(y, x);
24     }
25     v[1] = true; dfs(1); LL ans = -MAX18;
26     FOR(i, 1, n) ans = max(ans, point[i]);
27     cout << ans << endl;
28 }
```

## 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？

$n \leq 4$  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

```
1 const int N = 2e5 + 7, M = 1e6 + 7;
2 int n, m, s, e; int d[N][2], v[N][2]; // 0 代表最短路, 1 代表次短路
3 Z num[N][2];
4
5 void Clear() {
6     for (int i = 1; i <= n; ++ i) h[i] = edge[i] = 0;
7     tot = 0;
8     for (int i = 1; i <= n; ++ i) num[i][0] = num[i][1] = v[i][0] = v[i][1] = 0;
9     for (int i = 1; i <= n; ++ i) d[i][0] = d[i][1] = INF;
10 }
11
12 int ver[M], ne[M], h[N], edge[M], tot;
13 void add(int x, int y, int w) {
```

```

14     ver[++ tot] = y, ne[tot] = h[x], h[x] = tot;
15     edge[tot] = w;
16 }
17
18 void dji() {
19     priority_queue<PIII, vector<PIII>, greater<PIII> > q; q.push({0, s, 0});
20     num[s][0] = 1; d[s][0] = 0;
21     while (!q.empty()) {
22         auto [dis, x, type] = q.top(); q.pop();
23         if (v[x][type]) continue; v[x][type] = 1;
24         for (int i = h[x]; i; i = ne[i]) {
25             int y = ver[i], w = dis + edge[i];
26             if (d[y][0] > w) {
27                 d[y][1] = d[y][0], num[y][1] = num[y][0];
28                 // 如果找到新的最短路, 将原有的最短路数据转化为次短路
29                 q.push({d[y][1], y, 1});
30                 d[y][0] = w, num[y][0] = num[x][type];
31                 q.push({d[y][0], y, 0});
32             }
33             else if (d[y][0] == w) num[y][0] += num[x][type];
34             else if (d[y][1] > w) {
35                 d[y][1] = w, num[y][1] = num[x][type];
36                 q.push({d[y][1], y, 1});
37             }
38             else if (d[y][1] == w) num[y][1] += num[x][type];
39         }
40     }
41 }
42 void Solve() {
43     cin >> n >> m >> s >> e;
44     Clear(); //多组样例务必完全清空
45     for (int i = 1; i <= m; ++ i) {
46         int x, y, w; cin >> x >> y; w = 1;
47         add(x, y, w), add(y, x, w);
48     }
49     dji();
50     Z ans = num[e][0];
51     if (d[e][1] == d[e][0] + 1) {
52         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)
53     }
54     cout << ans.val() << endl;
55 }

```

### 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。

```

1  const int N = 1e5 + 7, M = 1e6 + 7;
2  int n, m;
3  int ver[M], ne[M], h[N], edge[M], tot;
4  int d[N], v[N], num[N];
5
6  void add(int x, int y, int w) {
7      ver[++ tot] = y, ne[tot] = h[x], h[x] = tot;
8      edge[tot] = w;
9  }
10 bool spfa() {
11     queue<int> q;
12     for (int i = 1; i <= n; ++ i) q.push(i), v[i] = 1; //全部入队
13     while(!q.empty()) {
14         int x = q.front(); q.pop();
15         v[x] = 0;
16         for (int i = h[x]; i; i = ne[i]) {
17             int y = ver[i];
18             if(d[y] > d[x] + edge[i]) {
19                 num[y] = num[x] + 1;
20                 if (num[y] >= n) return true;
21                 d[y] = d[x] + edge[i];
22                 if(!v[y]) q.push(y), v[y] = 1;
23             }
24         }
25     }
26 }

```

```

25     }
26     return false;
27 }
28 int main() {
29     cin >> n >> m;
30     for (int i = 1; i <= m; ++i) {
31         int x, y, w; cin >> x >> y >> w;
32         add(x, y, w);
33     }
34     if(spfa() == true) cout << "Yes" << endl;
35     else cout << "No" << endl;
36 }

```

### 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。

```

1  int n;
2  cin >> n;
3  vector<vector<int>> a(n + 1, vector<int>(n + 1));
4  for (int i = 1; i <= n; ++i) {
5      for (int j = 1; j <= n; ++j) {
6          char x;
7          cin >> x;
8          if (x == '1') a[i][j] = 1;
9      }
10 }
11
12 vector<int> vis(n + 1);
13 function<void(int, int)> dfs = [&](int x, int fa) {
14     vis[x] = 1;
15     for (int y = 1; y <= n; ++y) {
16         if (a[x][y] == 0) continue;
17         if (a[y][fa] == 1) {
18             cout << fa << " " << x << " " << y;
19             exit(0);
20         }
21         if (!vis[y]) dfs(y, x); // 这一步的 if 判断很关键
22     }
23 };
24 for (int i = 1; i <= n; ++i) {
25     if (!vis[i]) dfs(i, -1);
26 }
27 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。

```

1  LL Min = 1e18, ans = 0;
2  for (int k = 1; k <= n; k++) {
3      for (int i = 1; i <= n; i++) {
4          for (int j = 1; j <= n; j++) {
5              if (dis[i][j] > dis[i][k] + dis[k][j]) {
6                  dis[i][j] = dis[i][k] + dis[k][j];
7                  cnt[i][j] = cnt[i][k] * cnt[k][j] % mod;
8              } else if (dis[i][j] == dis[i][k] + dis[k][j]) {
9                  cnt[i][j] = (cnt[i][j] + cnt[i][k] * cnt[k][j] % mod) % mod;
10             }
11         }
12     }
13     for (int i = 1; i < k; i++) {
14         if (a[k][i]) {
15             if (a[k][i] + dis[i][k] < Min) {
16                 Min = a[k][i] + dis[i][k];
17                 ans = cnt[i][k];
18             } else if (a[k][i] + dis[i][k] == Min) {
19                 ans = (ans + cnt[i][k]) % mod;
20             }
21         }
22     }
23 }

```

```

22     }
23 }

```

### 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。

```

1  int floyd(int n) {
2      for (int i = 1; i <= n; ++ i) {
3          for (int j = 1; j <= n; ++ j) {
4              val[i][j] = dis[i][j]; // 记录最初的边权值
5          }
6      }
7      int ans = 0x3f3f3f3f;
8      for (int k = 1; k <= n; ++ k) {
9          for (int i = 1; i < k; ++ i) { // 注意这里是没有等于号的
10             for (int j = 1; j < i; ++ j) {
11                 ans = min(ans, dis[i][j] + val[i][k] + val[k][j]);
12             }
13         }
14         for (int i = 1; i <= n; ++ i) { // 往下是标准的 floyd
15             for (int j = 1; j <= n; ++ j) {
16                 dis[i][j] = min(dis[i][j], dis[i][k] + dis[k][j]);
17             }
18         }
19     }
20     return ans;
21 }

```

使用 bfs，复杂度为  $\mathcal{O}(N^2)$ 。

```

1  auto bfs = [&] (int s) {
2      queue<int> q; q.push(s);
3      dis[s] = 0;
4      fa[s] = -1;
5      while (q.size()) {
6          auto x = q.front(); q.pop();
7          for (auto y : ver[x]) {
8              if (y == fa[x]) continue;
9              if (dis[y] == -1) {
10                 dis[y] = dis[x] + 1;
11                 fa[y] = x;
12                 q.push(y);
13             }
14             else ans = min(ans, dis[x] + dis[y] + 1);
15         }
16     }
17 };
18 for (int i = 1; i <= n; ++ i) {
19     fill(dis + 1, dis + 1 + n, -1);
20     bfs(i);
21 }
22 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。

```

1  int n, m;
2  cin >> n >> m;
3  vector<vector<int>> G(n);
4  for (int i = 0; i < m; i++) {
5      int u, v;
6      cin >> u >> v;
7      u--, v--;
8      G[u].push_back(v);
9      G[v].push_back(u);
10 }
11 vector<vector<LL>> dp(1 << n, vector<LL>(n));
12 for (int i = 0; i < n; i++) dp[1 << i][i] = 1;

```

```

13 LL ans = 0;
14 for (int st = 1; st < (1 << n); st++) {
15     for (int u = 0; u < n; u++) {
16         if (!dp[st][u]) continue;
17         int start = st & -st;
18         for (auto v : G[u]) {
19             if ((1 << v) < start) continue;
20             if ((1 << v) & st) {
21                 if ((1 << v) == start) {
22                     ans += dp[st][u];
23                 }
24             } else {
25                 dp[st | (1 << v)][v] += dp[st][u];
26             }
27         }
28     }
29 }
30 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。

```

1 vector<int> dis(n + 1, -1), fa(n + 1);
2 auto dfs = [&](auto self, int x) -> void {
3     for (auto y : ver[x]) {
4         if (y == fa[x]) continue; // 二元环需删去该行
5         if (dis[y] == -1) {
6             dis[y] = dis[x] + 1;
7             fa[y] = x;
8             self(self, y);
9         } else if (dis[y] < dis[x]) {
10            cout << dis[x] - dis[y] + 1 << endl;
11            int pre = x;
12            cout << pre << " ";
13            while (pre != y) {
14                pre = fa[pre];
15                cout << pre << " ";
16            }
17            cout << endl;
18            exit(0);
19        }
20    }
21 };
22 for (int i = 1; i <= n; i++) {
23     if (dis[i] == -1) {
24         dis[i] = 0;
25         dfs(dfs, i);
26     }
27 }

```

### 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  $\mathcal{O}(N + M)$ 。

```

1 int ans = 0;
2 vector<int> vis(n + 1);
3 auto dfs = [&](auto self, int x) -> void {
4     vis[x] = 1;
5     for (auto y : ver[x]) {
6         if (vis[y] == 0) {
7             self(self, y);
8         } else if (vis[y] == 1) {
9             ans++;
10        }
11    }
12 }

```



```

11     }
12     vis[x] = 2;
13 };
14 for (int i = 1; i <= n; i++) {
15     if (!vis[i]) {
16         dfs(dfs, i);
17     }
18 }
19 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。

```

1 vector<int> dis(n + 1), vis(n + 1), fa(n + 1);
2 auto dfs = [&](auto self, int x) -> void {
3     vis[x] = 1;
4     for (auto y : ver[x]) {
5         if (vis[y] == 0) {
6             dis[y] = dis[x] + 1;
7             fa[y] = x;
8             self(self, y);
9         } else if (vis[y] == 1) {
10            cout << dis[x] - dis[y] + 1 << endl;
11            int pre = x;
12            cout << pre << " ";
13            while (pre != y) {
14                pre = fa[pre];
15                cout << pre << " ";
16            }
17            cout << endl;
18            exit(0);
19        }
20    }
21    vis[x] = 2;
22 };
23 for (int i = 1; i <= n; i++) {
24     if (!vis[i]) {
25         dfs(dfs, i);
26     }
27 }

```

### 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。

```

1 signed main() {
2     int n, m;
3     cin >> n >> m;
4     vector<pair<int, int>> in(m);
5     for (int i = 0, x, y; i < m; i++) {
6         cin >> x >> y;
7         in[i] = minmax(x, y);
8     }
9     TwoSat sat(m);
10    for (int i = 0; i < m; i++) {
11        auto [s, e] = in[i];
12        for (int j = i + 1; j < m; j++) {
13            auto [S, E] = in[j];
14            if (s < S && S < e && e < E || S < s && s < E && E < e) {
15                sat.add(i, 0, j, 0);
16                sat.add(i, 1, j, 1);
17            }
18        }
19    }
20    if (!sat.work()) {
21        cout << "Impossible\n";
22        return 0;
23    }
24 }

```

```

23     }
24     auto ans = sat.answer();
25     for (auto it : ans) {
26         cout << (it ? "out" : "in") << " ";
27     }
28 }

```

## 多项式

### NTT

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  using ll = long long;
4  const ll MOD = 998244353;
5  const int N = 4e6+10;
6  int rev[N];
7  int qpow(int a, int b){
8      int ret = 1;
9      for(; b ; b >>= 1){
10         if(b & 1) ret = 1ll * ret * a % MOD;
11         a = 1ll * a * a % MOD;
12     }
13     return ret;
14 }
15 int ginv(int x){
16     return qpow(x, MOD - 2);
17 }
18 int add(int x, int y){
19     x += y;
20     if(x >= MOD) return x - MOD;
21     return x;
22 }
23 int sub(int x, int y){
24     if(x < y) return x + MOD - y;
25     return x - y;
26 }
27 int preNTT(int len){
28     int deg = 1;
29     while(deg < len) deg *= 2;
30     for(int i = 0; i < deg; ++i)
31         rev[i] = (rev[i >> 1] >> 1) | (i & 1 ? deg / 2 : 0);
32     return deg;
33 }
34 struct poly : vector<int>{
35     using vector::vector;
36     using vector::operator [];
37     friend void NTT(poly &f, int deg, int opt){
38         f.resize(deg);
39         for(int i = 0; i < deg; ++i){
40             if(i < rev[i]) std::swap(f[i], f[rev[i]]);
41         }
42         for(int h = 2, m = 1; h <= deg; h *= 2, m *= 2){
43             int w1 = qpow(3, (MOD - 1) / h);
44             for(int l = 0; l < deg; l += h){
45                 int w0 = 1;
46                 for(int i = l, j = 0; i < l + m; ++j, ++i){
47                     int x = f[i], y = 1ll * w0 * f[i + m] % MOD;
48                     f[i] = x + y > MOD ? x + y - MOD : x + y;
49                     f[i + m] = x < y ? MOD + x - y : x - y;
50                     w0 = 1ll * w0 * w1 % MOD;
51                 }
52             }
53         }
54         if(opt == -1){
55             reverse(f.begin() + 1, f.end());
56             ll iv = ginv(deg);
57             for(auto &it : f){
58                 it = it * iv % MOD;
59             }

```

```

60     }
61 }
62 friend void dmul(poly& f, const poly& g){
63     for(int i = 0 ; i < g.size(); ++i){
64         f[i] = 1ll * f[i] * g[i] % MOD;
65     }
66 }
67 friend poly operator *(poly f, poly g){
68     if(f.empty() || g.empty()) return {};
69     int len = f.size() + g.size() - 1;
70     int deg = preNTT(len);
71     NTT(f, deg, 1);
72     NTT(g, deg, 1);
73     dmul(f, g);
74     NTT(f, deg, -1);
75     f.resize(len);
76     return f;
77 }
78 friend void operator *=(poly& f, poly g){
79     if(f.empty() || g.empty()) {
80         f = {};
81         return;
82     }
83     int len = f.size() + g.size() - 1;
84     int deg = preNTT(len);
85     NTT(f, deg, 1);
86     NTT(g, deg, 1);
87     dmul(f, g);
88     NTT(f, deg, -1);
89     f.resize(len);
90 }
91 }f;
92 int read(){
93     int x = 0;
94     char ch = getchar();
95     while(!isdigit(ch)) ch = getchar();
96     while(isdigit(ch)) x = x * 10 + ch - '0', ch = getchar();
97     return x;
98 }
99
100 int main(){
101     int n, m;
102     n = read(), m = read();
103     ++n, ++m;
104     poly f, g;
105     f.resize(n), g.resize(m);
106     for(int i = 0; i < n; ++i) f[i] = read();
107     for(int i = 0; i < m; ++i) g[i] = read();
108     f = f * g;
109     for(auto it : f){
110         printf("%d ", it);
111     }
112     printf("\n");
113     return 0;
114 }
115

```

## 全家桶（可能不全）

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  using ll = long long;
4  const ll MOD = 998244353;
5  const int N = 4e5+10;
6  int rev[N];
7  ll qpow(ll a, ll b){
8      ll ret = 1;
9      for(; b ; b >>= 1){
10         if(b & 1) ret = ret * a % MOD;
11         a = a * a % MOD;
12     }

```

```

13     return ret;
14 }
15 ll ginv(ll x){
16     return qpow(x, MOD - 2);
17 }
18 ll add(ll x, ll y){
19     x += y;
20     if(x >= MOD) return x - MOD;
21     return x;
22 }
23 ll sub(ll x, ll y){
24     x -= y;
25     if(x < 0) return x + MOD;
26     return x;
27 }
28 ll w[N];
29 int preNTT(int len){
30     int deg = 1;
31     while(deg < len) deg *= 2;
32     for(int i = 0; i < deg; ++i)
33         rev[i] = (rev[i >> 1] >> 1) | (i & 1 ? deg / 2 : 0);
34     w[0] = 1;
35     w[1] = qpow(3, (MOD - 1) / deg);
36     for(int i = 2; i < deg; ++i) w[i] = w[i - 1] * w[1] % MOD;
37     return deg;
38 }
39 struct poly : vector<ll>{
40     using vector::vector;
41     using vector::operator [];
42     void cksize(int x){
43         if(size() < x) resize(x);
44     }
45     ll& operator()(int x){
46         cksize(x + 1);
47         return this->operator[](x);
48     }
49     friend poly diff(poly f){
50         for(int i = 0; i + 1 < f.size(); ++i){
51             f[i] = f[i + 1] * (i + 1) % MOD;
52         }
53         f.pop_back();
54         return f;
55     }
56     friend poly inte(poly f){
57         if(f.empty()) return {};
58         for(int i = int(f.size()) - 1; i >= 1; --i){
59             f[i] = f[i - 1] * ginv(i) % MOD;
60         }
61         f[0] = 0;
62         return f;
63     }
64     void makemod(){
65         for(auto & it : (*this)){
66             it = (it % MOD + MOD) % MOD;
67         }
68     }
69     friend void NTT(poly &f, int deg, int opt){
70         f.resize(deg);
71         // f.ckmod();
72         for(int i = 0; i < deg; ++i){
73             if(i < rev[i]) std::swap(f[i], f[rev[i]]);
74         }
75         for(int h = 2, m = 1, t = deg / 2; h <= deg; h *= 2, m *= 2, t /= 2){
76             for(int l = 0; l < deg; l += h){
77                 for(int i = l, j = 0; i < l + m; ++j, ++i){
78                     ll x = f[i], y = w[t * j] * f[i + m] % MOD;
79                     f[i] = add(x, y);
80                     f[i + m] = sub(x, y);
81                 }
82             }
83         }

```

```

84     f.makemod();
85     if(opt == -1){
86         reverse(f.begin() + 1, f.end());
87         ll iv = ginv(deg);
88         for(auto &it : f){
89             it = it * iv % MOD;
90         }
91     }
92 }
93 friend poly dmul(poly f, const poly& g){
94     f.cksize(g.size());
95     for(int i = 0; i < g.size(); ++i){
96         f[i] = f[i] * g[i] % MOD;
97     }
98     return f;
99 }
100 friend poly operator - (poly f, const poly& g){
101     f.cksize(g.size());
102     for(int i = 0; i < g.size(); ++i){
103         f[i] = (f[i] - g[i] + MOD) % MOD;
104     }
105     return f;
106 }
107 friend poly operator *(poly f, poly g){
108     if(f.empty() || g.empty()) return {};
109     int len = f.size() + g.size() - 1;
110     int deg = preNTT(len);
111     NTT(f, deg, 1);
112     NTT(g, deg, 1);
113     f = dmul(std::move(f), g);
114     NTT(f, deg, -1);
115     f.resize(len);
116     return f;
117 }
118 friend poly pinv(const poly& f){
119     if(f.empty()) return {};
120     poly ret;
121     ret(0) = ginv(f[0]);
122     poly a;
123     for(int len = 2; len < (2 * f.size()); len *= 2){
124         a.assign(f.begin(), f.begin() + min(len, (int)f.size()));
125         int deg = preNTT(a.size() + 2 * ret.size() - 2);
126         NTT(ret, deg, 1);
127         NTT(a, deg, 1);
128         for(int i = 0; i < deg; ++i)
129             ret[i] = (2 - a[i] * ret[i] % MOD) * ret[i] % MOD;
130         ret.makemod();
131         NTT(ret, deg, -1);
132         ret.resize(len); // to mod
133     }
134     ret.resize(f.size());
135     return ret;
136 }
137 friend poly sqrt(const poly& f){
138     poly res;
139     res(0) = 1;
140     poly a;
141     ll iv2 = ginv(2);
142     for(int len = 2; len < 2 * f.size(); len *= 2){
143         res.resize(len); //to ensure enough space & inv's mod is len
144         a.assign(f.begin(), f.begin() + min(len, (int)f.size()));
145         a = a * pinv(res);
146         for(int i = 0; i < len; ++i) res[i] = (res[i] + a[i]) * iv2 % MOD;
147     }
148     res.resize(f.size());
149     return res;
150 }
151 void prt(const{
152     for(auto it : (*this)){
153         cerr << it << " ";
154     }

```

```

155     cerr<<endl;
156 }
157 friend poly ln(const poly& f){
158     poly res = inte(diff(f) * pinv(f));
159     res.resize(f.size());
160     return res;
161 }
162 friend poly exp(const poly& f){
163     poly ret;
164     ret(0) = 1;
165     poly a, b;
166     for(int len = 2; len < f.size() * 2; len *= 2){
167         ret.resize(len); // to ensure INV's mod is len
168         a = ln(ret);
169         b.assign(f.begin(), f.begin() + min(len, (int)f.size()));
170         b = b - a;
171         b[0] ++ ;
172         ret = ret * b;
173         ret.resize(len); // to mod len
174     }
175     ret.resize(f.size());
176     return ret;
177 }
178 }f;
179 int read(){
180     int x = 0;
181     char ch = getchar();
182     while(!isdigit(ch)) ch = getchar();
183     while(isdigit(ch)) x = x * 10 + ch - '0', ch = getchar();
184     return x;
185 }
186 int main(){
187     ios::sync_with_stdio(false), cin.tie(0), cout.tie(0);
188     int n;
189     cin >> n;
190     f.resize(n);
191     for(int i = 0; i < n; ++i){
192         cin >> f[i];
193     }
194     f = exp(f);
195     for(int i = 0; i < n; ++i){
196         cout << f[i] <<" ";
197     }
198     cout <<endl;
199     return 0;
200 }
201

```

## 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,  $\omega_n = \text{qpow}(G, (\text{mod}-1)/n)$ 。
- 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  $(1+x)^\alpha = \sum_{i=0}^{\infty} \binom{\alpha}{i} x^i$ 。

### 普通生成函数 / OGF

- 普通生成函数:  $A(x) = a_0 + a_1 x + a_2 x^2 + \dots = \langle a_0, a_1, a_2, \dots \rangle$ ;
- $1 + x^k + x^{2k} + \dots = \frac{1}{1 - x^k}$ ;

- 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);
- $x + \frac{x^2}{2} + \frac{x^3}{3} + \dots = -\ln(1-x)$ ;
- $1 + x + x^2 + \dots + x^{m-1} = \frac{1-x^m}{1-x}$ ;
- $1 + 2x + 3x^2 + \dots = \frac{1}{(1-x)^2}$  (借用导数,  $nx^{n-1} = (x^n)'$ );
- $C_m^0 + C_m^1 x + C_m^2 x^2 + \dots + C_m^m x^m = (1+x)^m$  (二项式定理);
- $C_m^0 + C_{m+1}^1 x + C_{m+2}^2 x^2 + \dots = \frac{1}{(1-x)^{m+1}}$  (归纳法证明);
- $\sum_{n=0}^{\infty} F_n x^n = \frac{(F_1 - F_0)x + F_0}{1 - x - x^2}$  (F 为斐波那契数列, 列方程  $G(x) = xG(x) + x^2G(x) + (F_1 - F_0)x + F_0$ );
- $\sum_{n=0}^{\infty} H_n x^n = \frac{1 - \sqrt{1-4x}}{2x}$  (H 为卡特兰数);
- 前缀和  $\sum_{n=0}^{\infty} s_n x^n = \frac{1}{1-x} f(x)$ ;
- 五边形数定理:  $\prod_{i=1}^{\infty} (1 - x^i) = \sum_{k=0}^{\infty} (-1)^k x^{\frac{1}{2}k(3k \pm 1)}$ 。

## 指数生成函数 / EGF

- 指数生成函数:  $A(x) = a_0 + a_1 x + a_2 \frac{x^2}{2!} + a_3 \frac{x^3}{3!} + \dots = \langle a_0, a_1, a_2, a_3, \dots \rangle$ ;
- 普通生成函数转换为指数生成函数: 系数乘以  $n!$ ;
- $1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots = \exp x$ ;
- 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

```

1 string s,t;cin>>s>>t;
2     int n=s.size(),m=t.size();
3     s=' '+s,t=' '+t;
4     vi nxt(t.size());
5     for(int i=2,j=0;i<=m;i++)
6     {
7         while(j&&t[j+1]!=t[i])j=nxt[j];
8         if(t[j+1]==t[i])j++;
9         nxt[i]=j;
10    }
11    for(int i=1,j=0;i<=n;i++)
12    {
13        while(j&&t[j+1]!=s[i])j=nxt[j];
14        if(t[j+1]==s[i])j++;
15        if(j==m)
16        {
17            cout<<i-m+1<<'\\n';
18            j=nxt[j];
19        }
20    }

```

```
21     for(int i=1;i<=m;i++)cout<<nxt[i]<<" ";
```

ac

```
1  constexpr int MAXN=3e5+10;
2  int son[MAXN][26],nxt[MAXN],mmax[MAXN],tt;
3  inline void insert(char s[])
4  {
5      int i=0,p=0;
6      for(;s[i];i++)
7      {
8          int c=s[i]-'a';
9          if(!son[p][c])son[p][c]=++tt;
10         p=son[p][c];
11     }
12     mmax[p]=i;
13 }
14 inline void built()
15 {
16     queue<int>q;
17     for(int i=0;i<26;i++)if(son[0][i])q.push(son[0][i]);
18     while(!q.empty())
19     {
20         int u=q.front();q.pop();
21         for(int i=0;i<26;i++)
22         {
23             int&v=son[u][i];
24             if(v){q.push(v),nxt[v]=son[nxt[u]][i];}
25             else v=son[nxt[u]][i];
26         }cmax(mmax[u],mmax[nxt[u]]);
27     }
28 }
```

z函数

```
1  int n=strlen(s+1);
2  z[1]=n;
3  for(int i=2,l=0,r=0;i<=n;i++)
4  {
5      int& p=z[i];
6      if(i<=r)p=min(z[i-l+1],r-i+1);
7      while(i+p<=n&&s[p+i]==s[p+1])p++;
8      if(i+p-1>r)l=i,r=i+p-1;
9  }
10
11  int m=strlen(t+1);
12  for(int i=1,l=0,r=0;i<=m;i++)
13  {
14      int& p=nxt[i];
15      if(i<=r)p=min(z[i-l+1],r-i+1);
16      while(i+p<=m&&t[p+i]==s[p+1])p++;
17      if(i+p-1>r)l=i,r=i+p-1;
18  }
```

sa

```
1  void get_sa(int n,int m)
2  {
3      for(int i=1;i<=n;i++)cnt[x[i]=a[i]]++;
4      for(int i=1;i<=m;i++)cnt[i]+=cnt[i-1];
5      for(int i=n;i;i--)sa[cnt[x[i]]--]=i;
6      for(int k=1;k<=n;k<=1)
7      {
8          int p=0;
9          for(int i=n-k+1;i<=n;i++)y[++p]=i;
10         for(int i=1;i<=n;i++)if(sa[i]>k)y[++p]=sa[i]-k;
11         for(int i=1;i<=m;i++)cnt[i]=0;
12         for(int i=1;i<=n;i++)cnt[x[i]]++;
13         for(int i=1;i<=m;i++)cnt[i]+=cnt[i-1];
```



```

14     for(int i=n;i;i--)sa[cnt[x[y[i]]]--]=y[i],y[i]=0;
15     swap(x,y);
16     p=1;x[sa[1]]=1;
17     for(int i=2;i<=n;i++)
18         x[sa[i]]=(y[sa[i]]==y[sa[i-1]]&&y[sa[i]+k]==y[sa[i-1]+k])?p:++p;
19     if(p==n)break;
20     m=p;
21 }
22 for(int i=1;i<=n;i++)rk[sa[i]]=i;
23 //for(int i=1;i<=n;i++)cerr<<rk[i]<<" ";
24 for(int i=1,k=0;i<=n;i++)
25 {
26     if(rk[i]==1)continue;
27     if(k)k--;
28     int j=sa[rk[i]-1];
29     while(i+k<=n&&j+k<=n&&a[i+k]==a[j+k])k++;
30     hi[rk[i]]=k;
31 }
32 }

```

## 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

```

1  #include<bits/stdc++.h>
2  // #pragma GCC optimize("Ofast")
3  using namespace std;
4  template<typename T>
5  inline bool cmax(T&x,const T& y){return x<y?x=y,1:0;}
6  template<typename T>
7  inline bool cmin(T&x,const T& y){return y<x?x=y,1:0;}
8  typedef long long ll;
9  typedef pair<int,int> pii;
10 typedef pair<ll,ll> pll;
11 typedef vector<int> vi;
12 typedef vector<vector<int>> > vii;
13 const int mod=998244353;
14 inline void MOD(int&x){x-=mod,x+=x>>31&mod;}
15 inline void MOD(ll& x){x-=mod,x+=x>>63&mod;}

```

```

16 inline int add(int x,int y){MOD(x+=y);return x;}
17 inline int mul(int x,int y){return 1ll*x*y%mod;}
18 template<typename ... A>inline int mul(const int& x,const A&... p){return 1ll*x*mul(p...)%mod;}
19 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;}
20 typedef long double LD;
21 const int MAXN=2e6+10;
22 array<int,26>son[MAXN];
23 int fa[MAXN],len[MAXN],la=1,tt=1;
24 int insert(int c,int la)
25 {
26     if(son[la][c]&&len[son[la][c]]==len[la]+1)return son[la][c];
27     int p=la,np=++tt,flag=0;
28     len[np]=len[p]+1;
29     for(;p&&!son[p][c];p=fa[p])son[p][c]=np;
30     if(!p){fa[np]=1;return np;}
31     int q=son[p][c];
32     if(len[q]==len[p]+1){fa[np]=q;return np;}
33     if(p==la)flag=1,np=0,tt--;
34     int nq=++tt;son[nq]=son[q],fa[nq]=fa[q],len[nq]=len[p]+1;
35     fa[q]=fa[np]=nq;
36     for(;p&&son[p][c]==q;p=fa[p])son[p][c]=nq;
37     return flag?q:nq;
38 }
39 int main()
40 {
41     ios::sync_with_stdio(0),cin.tie(0),cout.tie(0);cout<<fixed<<setprecision(10);
42     int n;cin>>n;
43     for(int i=1;i<=n;i++)
44     {
45         string s;cin>>s;
46         la=1;
47         for(auto p:s)la=insert(p-'a',la);
48     }
49     ll ans=0;
50     for(int i=2;i<=tt;i++)ans+=len[i]-len[fa[i]];
51     cout<<ans<<'\n';
52     cout<<tt<<'\n';
53     return 0;
54 }

```

## 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, 有的点没有出现, 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。

```

1  const int N=300005;
2  int n,m;
3  vector<int> edge[N];
4  int fa[N],dep[N],siz[N],son[N];
5  int top[N],dfn[N],tot;
6  int rnk[N],Fir[N],Sec[N],cnt;
7  void dfs1(int x,int f){
8      fa[x]=f,dep[x]=dep[f]+1;
9      siz[x]=1,son[x]=-1;
10     rnk[++cnt]=x,Fir[x]=cnt;
11     for(int y:edge[x]){
12         if(y==f)continue;
13         dfs1(y,x);
14         siz[x]+=siz[y];
15         if(son[x]==-1||siz[y]>siz[son[x]])son[x]=y;
16     }rnk[++cnt]=x,Sec[x]=cnt;
17 }
18 void dfs2(int x,int t){
19     top[x]=t,dfn[x]=++tot;
20     if(son[x]==-1)return;
21     dfs2(son[x],t);
22     for(int y:edge[x]){
23         if(y==fa[x]||y==son[x])continue;
24         dfs2(y,y);
25     }
26 }
27 inline int Lca(int x,int y){
28     int fx=top[x],fy=top[y];
29     while(fx!=fy){
30         if(dep[fx]>=dep[fy])x=fa[fx];
31         else y=fa[fy];
32         fx=top[x],fy=top[y];
33     }
34     if(dfn[x]<dfn[y])return x;
35     else return y;
36 }
37 int bel[N];
38 int a[N],b[N],lsh;

```

```

39 int ans[N];
40 struct node{int l,r,lca,id;}q[N];
41 inline bool cmp(node a,node b){
42     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);
43 }
44 int vis[N],Cnt[N];
45 int now;
46 inline void cal(int pos){
47     vis[pos]?(now-=!--Cnt[a[pos]]):(now+=!Cnt[a[pos]]++);
48     vis[pos]^=1;
49 }
50 signed main()
51 {
52     n=read(),m=read();
53     For(i,1,n)a[i]=read(),b[i]=a[i];
54     sort(b+1,b+n+1);
55     lsh=unique(b+1,b+n+1)-(b+1);
56     For(i,1,n)a[i]=lower_bound(b+1,b+lsh+1,a[i])-b;
57     For(i,1,n-1){
58         int u=read(),v=read();
59         edge[u].push_back(v);
60         edge[v].push_back(u);
61     }dfs1(1,1);dfs2(1,1);
62     int SZ=sqrt(2*n);For(i,1,2*n)bel[i]=i/SZ;
63     For(i,1,m){
64         int l=read(),r=read(),lca=Lca(l,r);
65         if(Fir[l]>Fir[r])swap(l,r);
66         if(l==lca)q[i]=node{Fir[l],Fir[r],0,i};
67         else q[i]=node{Sec[l],Fir[r],lca,i};
68     }sort(q+1,q+m+1,cmp);
69     int l=1,r=0;
70     For(i,1,m){
71         int L=q[i].l,R=q[i].r,lca=q[i].lca;
72         while(r<R)cal(rnk[++r]);
73         while(r>R)cal(rnk[r--]);
74         while(l>L)cal(rnk[--l]);
75         while(l<L)cal(rnk[l++]);
76         if(lca)cal(lca);
77         ans[q[i].id]=now;
78         if(lca)cal(lca);
79     }
80     For(i,1,m)write(ans[i],'\n');
81
82     return 0;
83 }

```

## 带悔贪心

cycrcy 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，cycrcy 不会在相邻的两个坑中种树。而且由于 cycrcy 的树种不够，他至多会种  $k$  棵树。假设 cycrcy 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。

```

1 int n,k;
2 struct node{
3     int x,id;
4 }a[N];
5 bool operator<(node A,node B){return A.x<B.x;}
6 priority_queue<node> q;
7 int L[N],R[N];
8 int vis[N];
9 inline void del(int x){
10     L[x]=L[L[x]];
11     R[x]=R[R[x]];
12     R[L[x]]=x;
13     L[R[x]]=x;
14 }
15 inline void work(){
16     n=read(),k=read();
17     For(i,1,n){
18         a[i].x=read(),a[i].id=i;

```

```

19     q.push(a[i]);
20     L[i]=i-1; if(i!=n)R[i]=i+1;
21 }
22 int ans=0,res=0;
23 For(i,1,k){
24     while(vis[q.top().id])q.pop();
25     node tt=q.top(); q.pop();
26     res+=tt.x; ans=max(ans,res);
27     int _L=L[tt.id],_R=R[tt.id];
28     a[tt.id].x=a[_L].x+a[_R].x-tt.x;
29     vis[_L]=vis[_R]=1;
30     q.push(a[tt.id]);
31     del(tt.id);
32 }
33 write(ans,'\n');
34 }

```

## 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。

$x + y + z \leq 10^5$ 。

```

1  struct node{int x,id;};
2  inline bool operator<(const node &A,const node &B){
3      if(A.x!=B.x)return A.x<B.x;
4      return A.id<B.id;
5  }
6  int n;
7  int X,Y,Z;
8  int a[N],b[N],c[N];
9  priority_queue<node> q1,q2,q3;
10 int vis[N];
11 signed main()
12 {
13     X=read(),Y=read(),Z=read();n=X+Y+Z;
14     For(i,1,n)a[i]=read(),b[i]=read(),c[i]=read();
15     int ans=0;
16     For(i,1,n)ans+=a[i],b[i]-=a[i],c[i]-=a[i];
17     For(i,1,n)q1.push(node{b[i],i});
18     For(i,1,Y){
19         node tt=q1.top();q1.pop();
20         ans+=tt.x;vis[tt.id]=1;
21     }
22     // write(ans,'\n');
23     For(i,1,n){
24         if(vis[i])q3.push(node{c[i]-b[i],i});
25         else q2.push(node{c[i],i});
26     }
27     For(i,1,Z){
28         while(!q1.empty()&&vis[q1.top().id])q1.pop();
29         while(!q2.empty()&&vis[q2.top().id])q2.pop();
30         int v1,v2,v3;
31         if(q1.empty())v1=-1e9; else v1=q1.top().x;
32         if(q2.empty())v2=-1e9; else v2=q2.top().x;
33         if(q3.empty())v3=-1e9; else v3=q3.top().x;
34         // write(v1,' ',v2,' ',v3,'\n');
35         if(v2>=v1+v3){
36             ans+=v2;node tt=q2.top();q2.pop();
37             vis[tt.id]=2;
38         }else{
39             ans+=v1+v3;node tt1=q1.top(),tt3=q3.top();
40             q1.pop();q3.pop();vis[tt1.id]=vis[tt3.id]=1;
41             q3.push(node{c[tt1.id]-b[tt1.id],tt1.id});
42         }
43     }
44     // For(i,1,n)write(vis[i],' '); write('\n');
45     write(ans,'\n');

```

```

46     return 0;
47 }

```

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

```

1  int S,T;
2  vector<pair<int,int>> edge[21];
3  int dis[21],vis[21],pre[21];
4  inline void spfa(){
5      queue<int> q;
6      memset(dis,-0x3f,sizeof(dis));
7      memset(vis,0,sizeof(vis));
8      q.push(S);dis[S]=0,vis[S]=1;
9      while(!q.empty()){
10         int x=q.front();q.pop();
11         vis[x]=0;
12         for(auto yy:edge[x]){
13             int y=yy.first,w=yy.second;
14             if(dis[y]<dis[x]+w){
15                 dis[y]=dis[x]+w,pre[y]=x;
16                 if(!vis[y]){vis[y]=1;q.push(y);}
17             }
18         }
19     }
20 }
21 int n,k;
22 int a[N];
23 int c[N][11];
24 priority_queue<pair<int,int>> q[11][11];
25 int id[11][11],bel[N];
26 inline void cl(int x,int y){
27     while(!q[x][y].empty()&&bel[q[x][y].top().second]!=id[x][y])q[x][y].pop();
28 }
29 inline void update(int x,int p){
30     bel[x]=p;
31     For(i,1,k)if(i!=p)q[i][p].push(mk(c[x][i]-c[x][p],x));
32 }
33 int ans;
34 inline void work(){
35     For(i,1,T)edge[i].clear();
36     For(i,1,k)if(a[i])edge[S].push_back(mk(i,0));
37     For(i,1,k){
38         cl(i,i);if(!q[i][i].empty())edge[i].push_back(mk(T,q[i][i].top().first));
39     }
40     For(i,1,k)For(j,1,k)if(i!=j){
41         cl(i,j);if(!q[i][j].empty())edge[i].push_back(mk(j,q[i][j].top().first));
42     }spfa();ans+=dis[T];
43     // exit(0);
44     vector<int> vec;
45     int x=T; while(x)vec.push_back(x),x=pre[x];
46     // for(int pp:vec)write(pp,' ');
47     // exit(0);
48     for(int i=0;i+1<(signed)vec.size();++i){
49         // write(i,'\n');
50         if(vec[i]==T){
51             int p=vec[i+1];
52             cl(p,p);update(q[p][p].top().second,p);
53         }else if(vec[i+1]==S){
54             a[vec[i]]--;
55         }else{
56             cl(vec[i+1],vec[i]);
57             update(q[vec[i+1]][vec[i]].top().second,vec[i+1]);
58         }
59     }
60 }
61 signed main()
62 {
63     n=read(),k=read(),S=k+1,T=S+1;
64     For(i,1,k)a[i]=read();
65     For(i,1,k)For(j,1,k)if(i!=j)id[i][j]=j;

```

```

66     For(i,1,n)For(j,1,k){
67         c[i][j]=read();
68         q[j][j].push(mk(c[i][j],i));
69     }
70     For(i,1,n)work();
71     write(ans,'\n');
72     return 0;;
73 }

```

## 约瑟夫问题

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

$\mathcal{O}(N)$ 。

```

1  int jos(int n,int k){
2      int res=0;
3      repeat(i,1,n+1)res=(res+k)%i;
4      return res; // res+1, 如果编号从 1 开始
5  }

```

$\mathcal{O}(K \log N)$ ，适用于  $K$  较小的情况。

```

1  int jos(int n,int k){
2      if(n==1 || k==1)return n-1;
3      if(k>n)return (jos(n-1,k)+k)%n; // 线性算法
4      int res=jos(n-n/k,k)-n%k;
5      if(res<0)res+=n; // mod n
6      else res+=res/(k-1); // 还原位置
7      return res; // res+1, 如果编号从 1 开始
8  }

```

## 日期换算（基姆拉尔森公式）

已知年月日，求星期数。

```

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

```

## 博弈论

### 巴什博弈

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  $X(1 \leq X \leq M)$  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  $X(1 \leq X \leq M)$ ，后报数的人需要比前者所报数大  $Y(1 \leq Y \leq M)$ ，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  $X(a \leq X \leq b)$  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  $K \in \mathbb{N}^+, a \leq R_2 \leq b$ ）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  $Sum_N = 0$ （也可看作“且有偶数堆”）；
- 至少有一堆的石头数量大于 1，且  $Sum_N \neq 0$ 。

## 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  $Sum_N = 0^{**}$  时先手必败，反之先手必胜。<sup>\*\*</sup>

## 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。

```

1  int n, m, a[N], num[N];
2  int sg(int x) {
3      if (num[x] != -1) return num[x];
4
5      unordered_set<int> S;
6      for (int i = 1; i <= m; ++ i)
7          if (x >= a[i])
8              S.insert(sg(x - a[i]));
9
10     for (int i = 0; ; ++ i)
11         if (S.count(i) == 0)
12             return num[x] = i;
13 }
14 void Solve() {
15     cin >> m;
16     for (int i = 1; i <= m; ++ i) cin >> a[i];
17     cin >> n;
18
19     int ans = 0; memset(num, -1, sizeof num);
20     for (int i = 1; i <= n; ++ i) {
21         int x; cin >> x;
22         ans ^= sg(x);
23     }
24
25     if (ans == 0) no;
26     else yes;
27 }

```

### 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：

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

### 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：

$(1, 2), (3, 5), (4, 7), (6, 10), \dots$  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  $1, 2, 3, 4, \dots$ 。

更一般地，对于第  $k$  对数，第一个数为  $First_k = \left\lfloor \frac{k \cdot (1 + \sqrt{5})}{2} \right\rfloor$ ，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  $lorry = 1.618033988749894848204586834$ 。

```
1 const double lorry = (sqrt(5.0) + 1.0) / 2.0;
2 //const double lorry = 1.618033988749894848204586834;
3 void Solve() {
4     int n, m; cin >> n >> m;
5     if (n < m) swap(n, m);
6     double x = n - m;
7     if ((int)(lorry * x) == m) cout << "lose\n";
8     else cout << "win\n";
9 }
```

## 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。

```
1 int fib[100] = {1, 2};
2 map<int, bool> mp;
3 void Force() {
4     for (int i = 2; i <= 86; ++i) fib[i] = fib[i - 1] + fib[i - 2];
5     for (int i = 0; i <= 86; ++i) mp[fib[i]] = 1;
6 }
7 void Solve() {
8     int n; cin >> n;
9     if (mp[n] == 1) cout << "lose\n";
10    else cout << "win\n";
11 }
```

## 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。

```

1 auto dfs = [&](auto self, int x, int fa) -> int {
2     int x = 0;
3     for (auto y : ver[x]) {
4         if (y == fa) continue;
5         x ^= self(self, y, x);
6     }
7     return x + 1;
8 };
9 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

```

1 #include<bits/stdc++.h>
2 // #pragma GCC optimize("Ofast")
3 using namespace std;
4 template<typename T>
5 inline bool cmax(T&x,const T& y){return x<y?x=y,1:0;}
6 template<typename T>
7 inline bool cmin(T&x,const T& y){return y<x?x=y,1:0;}
8 typedef long long ll;
9 typedef pair<int,int> pii;
10 typedef pair<ll,ll> pll;
11 typedef vector<int> vi;
12 typedef vector<vector<int>> > vii;
13 const int mod=998244353;
14 inline void MOD(int&x){x-=mod,x+=x>>31&mod;}
15 inline void MOD(ll& x){x-=mod,x+=x>>63&mod;}
16 inline int add(int x,int y){MOD(x+=y);return x;}
17 inline int mul(int x,int y){return 1ll*x*y%mod;}
18 template<typename ... A>inline int mul(const int& x,const A&... p){return 1ll*x*mul(p...)%mod;}
19 inline ll ksm(ll a,ll p=mod-2){ll ans=1;for(;p>>=1;a=a%mod)if(p&1)ans=ans*a%mod;return ans;}
20 typedef long double LD;
21 const int MAXN=2e5+10;

```

## 圆方树

```

1 const intMAXN=2e6+10;
2 struct edge{int from,v;}e[MAXN*2];
3 int head[MAXN],cnt,n,m;
4 inline void addd(int u,int v)
5 {e[++cnt]=(edge){head[u],v},head[u]=cnt;}
6 #define tree(u) for(int i=head[u],v=e[i].v;i;i=e[i].from,v=e[i].v)
7 vector<int>g[MAXN];
8 int tt,tot,S;
9 int dfn[MAXN],low[MAXN],w[MAXN],s[MAXN],top;
10 inline void add(int u,int v){g[u].push_back(v),g[v].push_back(u);}
11 void tarjan(int u,int fa)
12 {
13     dfn[u]=low[u]=++tt,s[++top]=u,S++;w[u]=-1;
14     tree(u)
15     {
16         if(!dfn[v])

```

```

17     {
18         tarjan(v,u);
19         if(low[v]>=dfn[u])
20         {
21             tot++;
22             add(u,tot);w[tot]=1;
23             int x=0;
24             do
25             {
26                 x=s[top--];w[tot]++;
27                 add(x,tot);
28             }while(x!=v);
29         }
30         cmin(low[u],low[v]);
31     }
32     else if(v!=fa)cmin(low[u],dfn[v]);
33 }

```

## 李超线段树

```

1  struct node{LD k,b;inline LD val(LL x){return k*x+b;}};
2  node f[MAXN];
3  int id[MAXN];
4  const LD eps=1e-6;
5  inline bool better(int x,int y,int p)
6  {
7      auto a=f[x].val(p),b=f[y].val(p);
8      if(a-b>eps)return 1;
9      else if(fabs(a-b)<eps&& x<y)return 1;
10     else return 0;
11 }
12 struct xds
13 {
14     xds *ls,*rs;
15     int l,r;
16     int id;
17     xds(int L,int R):l(L),r(R),id(0)
18     {
19         if(L==R)return;
20         int mid=(L+R)>>1;
21         ls=new xds(L,mid),rs=new xds(mid+1,R);
22     }
23     void update(int ql,int qr,int u)
24     {
25         int mid=(l+r)>>1;
26         if(ql<=l&&r<=qr)
27         {
28             if(!id)return id=u,void();
29             if(l==r)
30             {
31                 if(better(u,id,l))id=u;
32                 return;
33             }
34             if(f[u].val(mid)>f[id].val(mid))swap(u,id);
35             if(better(u,id,l)) ls->update(ql,qr,u);
36             else if(better(u,id,r)) rs->update(ql,qr,u);
37             return;
38         }
39         if(ql<=mid) ls->update(ql,qr,u);
40         if(qr >mid) rs->update(ql,qr,u);
41     }
42 }
43 int ask(int qq)
44 {
45     if(l==r)return id;
46     int mid=(l+r)>>1;
47     int ans=qq<=mid?ls->ask(qq):rs->ask(qq);
48     if(better(id,ans,qq))ans=id;
49     return ans;
50 }
51 }

```

```
52 };
```

## 斯坦纳树

```
1 vector<pii>e[MAXN];
2 int F[1<<10][MAXN];
3 bool is[MAXN];
4 signed main()
5 {
6     ios::sync_with_stdio(0),cin.tie(0),cout.tie(0);
7     int n,m,p;
8     cin>>n>>m>>p;
9     int u,v,w;
10    for(int i=1;i<=m;i++)
11    {
12        cin>>u>>v>>w;
13        e[u].emplace_back(v,w);
14        e[v].emplace_back(u,w);
15    }
16    memset(F,0x3f,sizeof(F));
17    for(int i=0;i<p;i++)
18    {
19        int x;cin>>x;
20        F[1<<i][x]=0;
21    }
22    for(int s=1;s<(1<<p);s++)
23    {
24        for(int x=s&(s-1);x=s&(x-1))
25            for(int i=1;i<=n;i++)
26                cmin(F[s][i],F[x][i]+F[s^x][i]);
27        queue<int>q;
28        for(int i=1;i<=n;i++) q.push(i),is[i]=1;
29        while(!q.empty())
30        {
31            int u=q.front();q.pop();is[u]=0;
32            for(auto&p:e[u])if(cmin(F[s][p.X],F[s][u]+p.Y))
33            {
34                if(!is[p.X])q.push(p.X),is[p.X]=1;
35            }
36        }
37    }
38    int ans=1e9;
39    for(int i=1;i<=n;i++) cmin(ans,F[(1<<p)-1][i]);
40    cout<<ans;
41    return 0;
42 }
```

## 虚树

```
1 const int MAXN=3e5+10;
2 vector<pii> e[MAXN];
3 vi g[MAXN];
4 const int K=20;
5 pii F[K+1][MAXN*2];
6 int tt;
7 int dfn[MAXN],d[MAXN];
8 int mmin[MAXN];
9 int h[MAXN*2];
10 void dfs(int u,int fa)
11 {
12     dfn[u]=++tt;
13     F[0][tt]={d[u],u};
14     for(const pii&p:e[u])
15     {
16         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};
17     }
18 }
19
20 inline int LCA(int x,int y)
21 {
```

```

22     x=dfn[x],y=dfn[y];
23     if(x>y)swap(x,y);int k=31-__builtin_clz(y-x+1);
24     return min(F[k][x],F[k][y-(1<<k)+1]).Y;
25 }
26 int s[MAXN],top;
27 bool is[MAXN];
28 ll dp(int u)
29 {
30     ll sum=0;
31     for(const int&v:g[u])sum+=dp(v);
32     if(u==1)
33     {
34         if(is[u])sum=mmin[u];
35         g[u].clear(),is[u]=0;
36         return sum;
37     }
38     if(is[u])sum=mmin[u];
39     else cmin(sum,(ll)mmin[u]);
40     g[u].clear(),is[u]=0;
41     return sum;
42 }
43 signed main()
44 {
45     ios::sync_with_stdio(0),cin.tie(0),cout.tie(0);
46     int n,m;
47     cin>>n;
48     int u,v,w;
49     for(int i=1;i<n;i++)cin>>u>>v>>w,e[u].emplace_back(v,w),e[v].emplace_back(u,w);
50     mmin[1]=1e9;
51     dfs(1,0);
52     for(int i=1;i<=K;i++)
53         for(int j=1;j+(1<<i)-1<=tt;j++)
54             F[i][j]=min(F[i-1][j],F[i-1][j+(1<<(i-1))]);
55     cin>>m;
56     while(m--)
57     {
58         int k;cin>>k;
59         for(int i=1;i<=k;i++)cin>>h[i],is[h[i]]=1;
60         sort(h+1,h+k+1,[&](const int&x,const int&y){return dfn[x]<dfn[y];});
61         s[top=1]=h[1];
62         for(int i=2;i<=k;i++)
63         {
64             int u=h[i];
65             int l=LCA(u,s[top]);
66             while(1)
67             {
68                 if(d[l]>=d[s[top-1]])
69                 {
70                     if(l==s[top])break;
71                     g[l].push_back(s[top]);
72                     if(l!=s[top-1])s[top]=l;
73                     else top--;
74                     break;
75                 }
76                 else g[s[top-1]].push_back(s[top]),top--;
77             }
78             s[++top]=u;
79         }
80         while(top>1)g[s[top-1]].push_back(s[top]),top--;
81         cout<<dp(s[1])<<'\n';
82     }
83     return 0;
84 }

```

## 支配树

```

1  const int MAXN=65534+10;
2  const int K=16;
3  int F[K+1][MAXN],d[MAXN];
4  vi e[MAXN];
5  inline int LCA(int u,int v)

```

```

6 {
7     if(!u||!v)return u|v;
8     if(d[u]<d[v])swap(u,v);
9     for(int i=K;~i;i--)if(d[F[i][u]]>=d[v])u=F[i][u];
10    if(u==v)return u;
11    for(int i=K;~i;i--)if(F[i][u]!=F[i][v])u=F[i][u],v=F[i][v];
12    return F[0][u];
13 }
14 int in[MAXN],siz[MAXN];
15 int main()
16 {
17     ios::sync_with_stdio(0),cin.tie(0),cout.tie(0);cout<<fixed<<setprecision(10);
18     int n;cin>>n;
19     for(int i=1,x;i<=n;i++)
20     {
21         cin>>x;
22         while(x)
23         {
24             e[x].push_back(i);
25             in[i]++;
26             cin>>x;
27         }
28     }
29     for(int i=1;i<=n;i++)if(!in[i])in[i]=1,e[n+1].push_back(i);
30     vi q;q.push_back(n+1);int h=0;
31     while(h<q.size())
32     {
33         int u=q[h++];
34         d[u]=d[F[0][u]]+1;
35         for(int i=1;i<=K;i++)F[i][u]=F[i-1][F[i-1][u]];
36         for(auto&v:e[u])
37         {
38             F[0][v]=LCA(F[0][v],u);
39             if(--in[v])q.push_back(v);
40         }
41     }
42     for(int i=q.size()-1;~i;i--)
43     {
44         siz[F[0][q[i]]]+=siz[q[i]]+1;
45     }
46     for(int i=1;i<=n;i++)cout<<siz[i]<<'\\n';
47     return 0;
48 }

```

## LCA

```

1     vi e[MAXN];
2     int dfn[MAXN],tt,F[K+1][MAXN];
3     void dfs(int u,int fa)
4     {
5         F[0][dfn[u]=++tt]=fa;
6         for(auto&v:e[u])if(v!=fa)dfs(v,u);
7     }
8     inline int Min(int u,int v){return dfn[u]<dfn[v]?u:v;}
9     inline int LCA(int u,int v)
10    {
11        if(u==v)return u;
12        u=dfn[u],v=dfn[v];
13        if(u>v)swap(u,v);
14        int k=__lg(v-u++);
15        return Min(F[k][u],F[k][v-(1<<k)+1]);
16    }

```

## LCT

```

1     const int MAXN=3e5+10;
2     struct node
3     {
4         int son[2],fa;
5         bool rev;

```

```

6     int w,mmin;
7 }t[MAXN];
8 #define ls t[p].son[0]
9 #define rs t[p].son[1]
10 inline bool isroot(int p){return p!=t[t[p].fa].son[0]&&p!=t[t[p].fa].son[1];}
11 const int INF=2e9;
12 int n,m,T;
13 inline void push_up(int p)
14 {
15     t[p].mmin=p>n?t[p].w:INF;
16     if(ls)cmin(t[p].mmin,t[ls].mmin);
17     if(rs)cmin(t[p].mmin,t[rs].mmin);
18 }
19 inline void rev(int p){swap(ls,rs),t[p].rev^=1;}
20 inline void push_down(int p)
21 {
22     if(t[p].rev)
23     {
24         if(ls)rev(ls);if(rs)rev(rs);
25         t[p].rev=0;
26     }
27 }
28 inline void rotate(int x)
29 {
30     int y=t[x].fa,z=t[y].fa;
31     if(!isroot(y)) t[z].son[y==t[z].son[1]]=x;t[x].fa=z;
32     int r=x==t[y].son[1];z=t[x].son[r^1],t[x].son[r^1]=y;
33     t[y].fa=x,t[y].son[r]=z,t[z].fa=y;push_up(y);
34 }
35 inline void splay(int p)
36 {
37     static int s[MAXN],top;
38     s[top]=p;int x=p;
39     while(!isroot(p))s[++top]=p=t[p].fa;
40     while(top)push_down(s[top--]);
41     while(!isroot(x))
42     {
43         int y=t[x].fa;if(!isroot(y))rotate(((x==t[y].son[0])!=(y==t[t[y].fa].son[0]))?x:y);
44         rotate(x);
45     }push_up(x);
46 }
47 inline void access(int p)
48 {
49     for(int i=0;p;p=t[i=p].fa)splay(p),rs=i,push_up(p);
50 }
51 inline void makeroot(int p){access(p),splay(p),rev(p);}
52 inline void link(int x,int y)
53 {
54     makeroot(x),t[x].fa=y;
55 }
56 inline void cut(int x,int y)
57 {
58     makeroot(x),access(y),splay(y);
59     t[x].fa=0,t[y].son[0]=0;
60 }

```