

---

# *Fomalhaut*

Team Reference Material

---

朱玮昊  
王照梓  
唐嘉铭

# Fomalhaut

## Team Reference Material

<b>1 Math</b>	<b>1</b>	<b>4 Graph Theory</b>	<b>36</b>
1.1 素数相关	1	4.1 最短路	36
1.2 gcd 相关	3	4.2 次短路	36
1.3 函数相关	3	4.3 K-短路	37
1.4 筛法	4	4.4 差分约束	38
1.5 同余	5	4.5 生成树	39
1.6 组合数	8	4.6 曼哈顿最小生成树	40
1.7 单纯形	8	4.7 Kruskal 重构树	41
1.8 多项式	12	4.8 生成树计数	42
1.9 pell 方程	23	4.9 最小树形图	42
1.10 一元三次方程求根公式	23	4.10 Tarjan	43
1.11 simpson 积分	23	4.11 支配树	44
1.12 整数划分	23	4.12 二分图匹配	46
1.13 min-max 容斥	24	4.13 一般图匹配	48
1.14 线性基	24	4.14 网络流	51
1.15 Stern-Brocot 树	24	4.15 最大团 / 最大独立集	55
1.16 数相关	24	4.16 2-SAT	55
1.17 定理相关	25	4.17 LCA	56
1.18 博弈论	25	4.18 长链剖分	58
<b>2 String</b>	<b>26</b>	4.19 欧拉回路	58
2.1 KMP	26	4.20 图的绝对中心	60
2.2 ex-KMP	26	4.21 虚树	61
2.3 manacher	27	4.22 点分治	61
2.4 AC 自动机	27	4.23 斯坦纳树	63
2.5 后缀数组	27	4.24 弦图	65
2.6 后缀自动机	28	<b>5 Geometry</b>	<b>66</b>
2.7 ex-后缀自动机	28	5.1 二维几何	66
2.8 回文自动机	29	5.2 平面最近点对	75
2.9 最小表示法	29	5.3 三维几何	75
2.10 回文串分解	29	5.4 三维凸包	76
<b>3 Data Structure</b>	<b>30</b>	5.5 几何公式	78
3.1 KD-tree	30	<b>6 Others</b>	<b>79</b>
3.2 LCT	31	6.1 模拟退火	79
3.3 Treap	32	6.2 莫队	79
3.4 可持久化 Treap	33	6.3 手动开栈	79
3.5 线段树合并	34	6.4 O3	79
3.6 Segment tree beats	35	6.5 STL 释放内存	79
3.7 Hints	36	6.6 Java	79
		6.7 vimrc	80
		6.8 快速读入输出	80

## Ch1. Math

### 1.1. 素数相关

#### 埃氏筛

得到  $[L, R]$  范围内的所有素数

时间复杂度  $O(\sqrt{R} \log \log \sqrt{R} + (R - L) \log \log (R - L))$

1//get\_prime 得到  $[L, R]$  中的所有素数

2//tot 表示素数个数

3//p 数组表示素数

4//数组大小  $\max(\sqrt{R}, R-L)+1$

5#define Maxn 1000007

6int cnt, prime[Maxn];

7bool vis[Maxn];

8int tot, p[Maxn];

```

9int L,R;
10void get_prime(int L,int R)
11{
12    cnt=0;
13    memset(prime,0,sizeof(prime));
14    memset(vis,true,sizeof(vis));
15    int mx=(int)sqrt(R)+1;
16    for (int i=2;i<=mx;i++)
17        if (vis[i])
18        {
19            prime[++cnt]=i;
20            for (int j=2;j<=mx/i;j++)
21                vis[i*j]=false;
22        }
23    tot=0;
24    memset(p,0,sizeof(p));

```

```

25  memset(vis,true,sizeof(vis));
26  if (L==1) vis[0]=false;
27  for (int i=1;i<=cnt;i++)
28  {
29      int val=(L-1)/prime[i];
30      ++val;
31      if (val<=1) val=2;
32      while (val<=R/prime[i])
33      {
34          vis[val*prime[i]-L]=false;
35          ++val;
36      }
37  }
38  for (int i=L;;i++)
39  {
40      if (vis[i-L]) p[++tot]=i;
41      if (i==R) break;
42  }
43 }

```

### 线性筛

得到所有不超过  $N$  的素数

时间复杂度  $O(N)$

```

1 //get_prime 得到所有小于等于 n 的素数
2 //cnt 表示素数个数
3 //prime[i] 表示第 i 个素数的大小
4 //vis[i] 表示 i 是否为素数 (true 表示不是素数, false 表
  ↪ 示是素数)
5 //数组大小 n
6 #define Maxn 40000
7 int n;
8 int prime[Maxn],cnt;
9 bool vis[Maxn];
10 void get_prime(int n)
11 {
12     memset(prime,0,sizeof(prime));
13     cnt=0;
14     memset(vis,false,sizeof(vis));
15     vis[1]=true;
16     for (int i=2;i<=n;i++)
17     {
18         if (!vis[i]) prime[++cnt]=i;
19         for (int j=1;j<=cnt&&prime[j]<=n/i;j++)
20         {
21             vis[i*prime[j]]=true;
22             if (i%prime[j]==0) break;
23         }
24     }
25 }

```

### pollard-rho

将正整数  $N$  分解质因数

时间复杂度  $O(\sqrt[n]{n})$

```

1 //make 函数将 n 分解质因数
2 //cnt 表示素因子个数
3 //pri 数组表示素因子大小
4 //tot 数组表示每个素因子的个数
5 //注意 pri 数组不保证排好序
6 //数组大小为素因子个数 (n<=1e18 只需开到 30)
7 long long n;
8 long long tot[37],pri[37],cnt;
9 map<long long,int> mp;
10 long long multiply(long long x,long long y,long long
  ↪ modp)
11 {
12     return ((x*y-(long long)((long
  ↪ double)x*y+0.5)/modp)*modp)%modp;
13 }
14 long long power(long long a,long long b,long long modp)

```

```

15 {
16     long long res=1,now=a%modp,left=b;
17     while (left>0)
18     {
19         if (left%2==1) res=multiply(res,now,modp);
20         left/=2;
21         now=multiply(now,now,modp);
22     }
23     return res;
24 }
25 long long get_rand()
26 {
27     return (long long)rand()<<30|rand();
28 }
29 long long gcd(long long a,long long b)
30 {
31     return b==0?a:gcd(b,a%b);
32 }
33 long long Polland_Rho(long long n,int c)
34 {
35     long long x,y,cnt=1,k=2;
36     x=y=get_rand()%(n-1)+1;
37     while (true)
38     {
39         ++cnt;
40         x=(multiply(x,x,n)+c)%n;
41         long long g=gcd(n,abs(x-y));
42         if (g>1&&g<n) return g;
43         if (x==y) return n;
44         if (cnt==k)
45         {
46             y=x;
47             k<=1;
48         }
49     }
50 }
51 bool Miller_Robin(long long n)
52 {
53     if (n==1) return false;
54     if (n==2) return true;
55     if (n%2==0) return false;
56     long long a=0,b=n-1;
57     while (b%2==0)
58     {
59         ++a;
60         b/=2;
61     }
62     for (int t=1;t<=100;t++)
63     {
64         long long x=get_rand()%(n-1)+1,y=power(x,b,n);
65         for (long long i=0;i<a;i++)
66         {
67             x=multiply(y,y,n);
68             if (x==1&&y!=1&&y!=n-1) return false;
69             y=x;
70         }
71         if (y!=1) return false;
72     }
73     return true;
74 }
75 void make(long long n)
76 {
77     if (n==1) return;
78     if (Miller_Robin(n))
79     {
80         if (mp[n]) ++tot[mp[n]];
81         else
82         {
83             tot[++cnt]=1;

```

```

84         mp[n]=cnt;
85         pri[cnt]=n;
86     }
87     return;
88 }
89 long long p=n;
90 while (p>=n) p=Polland_Rho(n,get_rand()%(n-1));
91 make(p);
92 make(n/p);
93 }

```

## 1.2. gcd 相关

$\text{ex}_{gcd}$

求出满足  $ax + by = \gcd(a, b)$  的一组整数解  $(x, y)$   
时间复杂度  $O(\log n)$

```

1 //ex_gcd 求出 ax+by=gcd(a,b) 的一组整数解 (x,y)
2 //x,y 可能为负数
3 void ex_gcd(long long a,long long b,long long &x,long
  ↳ long &y)
4 {
5     if (b==0LL)
6     {
7         x=1LL;
8         y=0LL;
9         return;
10    }
11    ex_gcd(b,a%b,x,y);
12    long long t=x;
13    x=y;
14    y=t-1LL*(a/b)*y;
15 }

```

类欧几里得

求出:

$$f(a, b, c, n) = \sum_{i=0}^n \left\lfloor \frac{ai+b}{c} \right\rfloor$$

$$g(a, b, c, n) = \sum_{i=0}^n i \left\lfloor \frac{ai+b}{c} \right\rfloor$$

$$h(a, b, c, n) = \sum_{i=0}^n \left\lfloor \frac{ai+b}{c} \right\rfloor^2$$

当  $a \geq c$  或  $b \geq c$  时,

$$f(a, b, c, n) = f(a\%c, b\%c, c, n) + \frac{n(n+1)}{2} \left\lfloor \frac{a}{c} \right\rfloor + (n+1) \left\lfloor \frac{b}{c} \right\rfloor$$

$$g(a, b, c, n) = g(a\%c, b\%c, c, n) + \frac{n(n+1)(2n+1)}{6} \left\lfloor \frac{a}{c} \right\rfloor + \frac{n(n+1)}{2} \left\lfloor \frac{b}{c} \right\rfloor$$

$$h(a, b, c, n) = h(a\%c, b\%c, c, n) + \frac{n(n+1)(2n+1)}{6} \left\lfloor \frac{a}{c} \right\rfloor^2 + (n+1) \left\lfloor \frac{b}{c} \right\rfloor^2 + 2 \left\lfloor \frac{b}{c} \right\rfloor f(a\%c, b\%c, c, n) + 2 \left\lfloor \frac{a}{c} \right\rfloor g(a\%c, b\%c, c, n) + \left\lfloor \frac{a}{c} \right\rfloor \left\lfloor \frac{b}{c} \right\rfloor n(n+1)$$

否则, 当  $a = 0$  时,

$$f(a, b, c, n) = 0$$

$$g(a, b, c, n) = 0$$

$$h(a, b, c, n) = 0$$

否则,

$$f(a, b, c, n) = n \left\lfloor \frac{an+b}{c} \right\rfloor - f(c, -b+c-1, a, \left\lfloor \frac{an+b}{c} \right\rfloor - 1)$$

$$g(a, b, c, n) = \frac{\left\lfloor \frac{an+b}{c} \right\rfloor n(n+1) - f(c, -b+c-1, a, \left\lfloor \frac{an+b}{c} \right\rfloor - 1) - h(c, -b+c-1, a, \left\lfloor \frac{an+b}{c} \right\rfloor - 1)}{2}$$

$$h(a, b, c, n) = \left\lfloor \frac{an+b}{c} \right\rfloor \left( \left\lfloor \frac{an+b}{c} \right\rfloor + 1 \right) n - 2g(c, -b+c-1, a, \left\lfloor \frac{an+b}{c} \right\rfloor - 1) - f(a, b, c, n) - 2f(c, -b+c-1, a, \left\lfloor \frac{an+b}{c} \right\rfloor - 1) - f(a, b, c, n)$$

时间复杂度  $O(\log N)$

```

1 long long f(long long a,long long b,long long c,long
  ↳ long n)
2 {
3     if (a>=c||b>=c) return f(a%c,b%c,c,n)+1LL*(a/c) *
  ↳ ((1LL*n*(n+1))/2)+1LL*(n+1)*(b/c);
4     else if (a==0LL) return 0LL;
5     else return 1LL*n*((1LL*a*n+b)/c)-f(c,-b+c-1,a,(
  ↳ 1LL*a*n+b)/c-1);
6 }

```

## 1.3. 函数相关

欧拉函数

定义  $\phi(n)$  为  $1..n$  中与  $n$  互质的数的个数

(1) 设  $n = p_1^{q_1} \cdot p_2^{q_2} \dots p_k^{q_k}$ , 我们可得  $\phi(n) = n(1 - \frac{1}{p_1})(1 - \frac{1}{p_2}) \dots (1 - \frac{1}{p_k})$

(2)  $\phi(n)$  为部分积性函数, 即当  $\gcd(n, m) = 1$  时,  $\phi(nm) = \phi(n) \cdot \phi(m)$

(3) 当  $n \neq 2$  时,  $\phi(n)$  为偶数

(4)  $\sum_{d|n} \phi(n) = n$

(5) 小于  $n$  且和  $n$  互素的所有数之和为  $\frac{\phi(n) \cdot n}{2}$

通过线性筛得到  $[1, N]$  所有数的欧拉函数

时间复杂度  $O(N)$

```

1 //get_phi 得到所有小于等于 n 的数的欧拉函数
2 //cnt 表示素数个数
3 //prime[i] 表示第 i 个素数的大小
4 //vis[i] 表示 i 是否为素数 (true 表示不是素数, false 表
  ↳ 示是素数)
5 //phi[i] 表示 i 的欧拉函数
6 //数组大小 n
7 #define Maxn 50007
8 int n;
9 int prime[Maxn], phi[Maxn], cnt=0;
10 bool vis[Maxn];
11 void get_phi(int n)
12 {
13     memset(prime, 0, sizeof(prime));
14     memset(phi, 0, sizeof(phi));
15     phi[1]=1;
16     cnt=0;
17     memset(vis, false, sizeof(vis));
18     vis[1]=true;
19     for (int i=2; i<=n; i++)
20     {
21         if (!vis[i])
22         {
23             prime[++cnt]=i;
24             phi[i]=i-1;
25         }
26         for (int j=1; j<=cnt&&prime[j]<=n/i; j++)
27         {
28             vis[i*prime[j]]=true;
29             if (i%prime[j]==0)
30             {
31                 phi[i*prime[j]]=phi[i]*prime[j];
32                 break;
33             }
34             phi[i*prime[j]]=phi[i]*(prime[j]-1);
35         }
36     }
37 }

```

莫比乌斯函数

莫比乌斯函数:

当  $n = 1$  时,  $\mu(n) = 1$

当  $n = p_1 \cdot p_2 \cdot p_3 \dots p_m$  时,  $\mu(n) = (-1)^m$

当  $n$  不满足以上两种情况时,  $\mu(n) = 0$

莫比乌斯反演:

当  $g(n) = \sum_{d|n} f(d)$  时, 反演可得  $f(n) = \sum_{d|n} \mu(d) \cdot g(\frac{n}{d})$

由上可得以下性质:

(1)  $\sum_{d|x} \mu(d) = [x = 1]$

(2)  $\mu(x)$  为部分积性函数, 即当  $\gcd(n, m) = 1$  时,  $\mu(nm) = \mu(n) \cdot \mu(m)$

(3)  $\sum_{d|n} \mu(d) \cdot \frac{n}{d} = \phi(n)$

通过线性筛得到  $[1, N]$  所有数的莫比乌斯函数

时间复杂度  $O(N)$

```

1 //get_phi 得到所有小于等于 n 的数的欧拉函数
2 //cnt 表示素数个数
3 //prime[i] 表示第 i 个素数的大小
4 //vis[i] 表示 i 是否为素数 (true 表示不是素数, false 表
  ↳ 示是素数)
5 //mu[i] 表示 i 的莫比乌斯函数
6 //数组大小 n
7 #define Maxn 200007

```

```

8 int prime[Maxn], mu[Maxn], cnt=0;
9 bool vis[Maxn];
10 void get_mu(int n)
11 {
12     cnt=0;
13     memset(prime, 0, sizeof(prime));
14     memset(mu, 0, sizeof(mu));
15     memset(vis, false, sizeof(vis));
16     mu[1]=1;
17     vis[1]=true;
18     for (int i=2; i<=n; i++)
19     {
20         if (!vis[i])
21         {
22             prime[++cnt]=i;
23             mu[i]=-1;
24         }
25         for (int j=1; j<=cnt&&prime[j]<=n/i; j++)
26         {
27             vis[i*prime[j]]=true;
28             if (i%prime[j]==0)
29             {
30                 mu[i*prime[j]]=0;
31                 break;
32             }
33             mu[i*prime[j]]=-mu[i];
34         }
35     }
36 }

```

## 1.4. 筛法

### 杜教筛

对于积性函数  $f(i)$ , 构造积性函数  $g(x)$  和  $h(x)$  使得  $h(x) = \sum_{d|x} f(d)g(\frac{x}{d})$

定义  $S(n) = \sum_{i=1}^n f(i)$

则有  $g(1)S(n) = \sum_{i=1}^n h(i) - \sum_{d=2}^n g(d)S(\lfloor \frac{n}{d} \rfloor)$

$S1(n) = \sum_{i=1}^n \mu(i) = 1 - \sum_{d=2}^n S1(\lfloor \frac{n}{d} \rfloor)$

$S2(n) = \sum_{i=1}^n \phi(i) = \sum_{i=1}^n i - \sum_{d=2}^n S2(\lfloor \frac{n}{d} \rfloor)$

$S3(n) = \sum_{i=1}^n i \cdot \phi(i) = \sum_{i=1}^n i^2 - \sum_{d=2}^n d \cdot S3(\lfloor \frac{n}{d} \rfloor)$

时间复杂度  $O(n^{\frac{2}{3}})$

```

1 //calc() 计算 i*phi(i) 的前缀和
2 //B 表示 n^(2/3)
3 //先用线性筛算出 B 以内的答案, 大于 B 的使用杜教筛
4 //一定要用 map 预存已经计算过的部分
5 //inv2 和 inv6 分别表示 2 和 6 的乘法逆元
6 #define modp 1000000007
7 #define B 1000000
8 using namespace std;
9 int s[B+7];
10 int inv2, inv6;
11 unordered_map<int, int> mp;
12 int power(int a, int b)
13 {
14     int res=1, now=a, left=b;
15     while (left>0)
16     {
17         if (left%2==1) res=(1LL*res*now)%modp;
18         left/=2;
19         now=(1LL*now*now)%modp;
20     }
21     return res;
22 }
23 int calc1(int n)
24 {
25     int res=(1LL*n*(n+1))%modp;
26     res=(1LL*res*inv2)%modp;
27     return res;
28 }

```

```

29 int calc2(int n)
30 {
31     int res=(1LL*n*(n+1))%modp;
32     int tmp=(2LL*n+1)%modp;
33     res=(1LL*res*tmp)%modp;
34     res=(1LL*res*inv6)%modp;
35     return res;
36 }
37 int calc(int n)
38 {
39     if (n<=B) return s[n];
40     if (mp.find(n)!=mp.end()) return mp[n];
41     int res=calc2(n);
42     for (int now=2; now<=n; )
43     {
44         int tmp=n/(n/now);
45         int del=(1LL*calc(n/now)*((calc1(tmp)-calc1(
46             ↪ now-1)+modp)%modp))%modp;
47         res=(res+modp-del)%modp;
48         now=tmp+1;
49     }
50     mp[n]=res;
51     return res;
52 }
53 bool vis[B+7];
54 int prime[B+7], phi[B+7], cnt=0;
55 void pre()
56 {
57     phi[1]=1;
58     for (int i=2; i<=B; i++)
59     {
60         if (!vis[i])
61         {
62             prime[++cnt]=i;
63             phi[i]=i-1;
64         }
65         for (int j=1; j<=cnt; j++)
66         {
67             if (1LL*i*prime[j]>B) break;
68             vis[1LL*i*prime[j]]=true;
69             if (i%prime[j]==0)
70             {
71                 phi[1LL*i*prime[j]]=phi[i]*prime[j];
72                 break;
73             } else phi[1LL*i*prime[j]]=phi[i]*(
74                 ↪ prime[j]-1);
75         }
76     }
77     for (int i=1; i<=B; i++)
78     s[i]=(s[i-1]+1LL*i*phi[i])%modp;
79     mp.clear();
80     inv2=power(2, modp-2);
81     inv6=power(6, modp-2);
82 }

```

### $min_{25}$ 筛

设  $f(n)$  是一个积性函数, 且  $f(p)$  是关于  $p$  的简单多项式,  $f(p^c)$  能快速计算

第一部分: 对于  $x = [\frac{n}{i}]$  计算  $\sum_{i=2}^x [i \in prime] f(i)$

第一部分我们假设  $f(n)$  完全积性 (有的时候可以拆成若干个完全积性函数)

线性筛得到  $\sqrt{n}$  范围内的素数, 设  $P_j$  为从小到大第  $j$  个

设  $g(n, j) = \sum_{i=1}^n [i \in P \text{ min}_{prime}(i) > P_j] f(i)$

$g(n, 0) = \sum_{i=2}^n f(i)$

当  $P_j^2 \leq n$  时,  $g(n, j) = g(n, j-1) - f(P_j)[g(\frac{n}{P_j}, j-1) - \sum_{i=1}^{j-1} f(P_i)]$

当  $P_j^2 > n$  时,  $g(n, j) = g(n, j-1)$

第二部分: 计算  $\sum_{i=1}^n f(i)$

设  $S(n, j) = \sum_{i=2}^n [\min_{prime}(i) \geq P_j] f(i)$

$S(n, j) = g(n, |P|) - \sum_{i=1}^{j-1} f(P_i) + \sum_{k=j}^{P_k^2 \leq n} \sum_{e=1}^{P_k^{e+1} \leq n} (S(\frac{n}{P_k^e}, k +$

$1) * f(P_k^e) + f(P_k^{e+1}))$   
 $\sum_{i=1}^n f(i) = S(n, 1) + f(1)$

时间复杂度  $O(\frac{n^{\frac{3}{4}}}{\log n})$

注意两部分的  $f$  函数意义不同!!!

```
1//solve() 计算前缀和
2//f(1)=1
3//f(p^c)=p xor c
4//f(ab)=f(a)f(b) (gcd(a,b)==1)
5//数组大小 2*sqrt(n)+1
6using namespace std;
7#define modp 1000000007
8bool vis[200007];
9int prime[200007], cnt=0;
10long long sum[200007];
11long long val[200007];
12long long g1[200007], g2[200007];
13unordered_map<long long, int> mp;
14int tot=0;
15void get_prime(int n)
16{
17    memset(vis, false, sizeof(vis));
18    vis[1]=true;
19    for (int i=2; i<=n; i++)
20        if (!vis[i])
21        {
22            prime[++cnt]=i;
23            sum[cnt]=(sum[cnt-1]+i)%modp;
24            for (int j=2; j<=n/i; j++)
25                vis[i*j]=true;
26        }
27}
28void get_g()
29{
30    mp.clear();
31    long long now=1;
32    while (now<=n)
33    {
34        val[++tot]=n/now;
35        mp[n/now]=tot;
36        now=n/(n/now)+1;
37    }
38    for (int i=1; i<=tot; i++)
39    {
40        g1[i]=(1LL*(val[i]%modp)*((val[i]+1)%modp))%modp;
41        if (g1[i]%2==1) g1[i]+=modp;
42        g1[i]/=2;
43        g1[i]=(g1[i]+modp-1)%modp;
44        g2[i]=(val[i]+modp-1)%modp;
45    }
46    for (int j=1; j<=cnt; j++)
47        for (int i=1; i<=tot&& 1LL*prime[j]*prime[j]<=val[i]; i++)
48        {
49            int pos=mp[val[i]/prime[j]];
50            g1[i]=(g1[i]-(1LL*prime[j]*(g1[pos]-sum[j-1]+modp)%modp)+modp)%modp;
51            g2[i]=(g2[i]-g2[pos]+j-1)%modp;
52        }
53}
54long long s(long long n, int j)
55{
56    long long res=0LL;
57    if (n<=1LL||prime[j]>n) return res;
```

```
58    int pos=mp[n];
59    res=(g1[pos]-g2[pos]+modp)%modp;
60    res=(res-sum[j-1]+j-1+modp)%modp;
61    for (int k=j; k<=cnt&&1LL*prime[k]*prime[k]<=n; k++)
62    {
63        long long now=1LL*prime[k];
64        for (int e=1; 1LL*prime[k]*now<=n; e++, now=1LL*now*prime[k])
65        {
66            res=(res+1LL*s(n/now, k+1)*(prime[k]^e))%modp;
67            res=(res+(prime[k]^(e+1)))%modp;
68        }
69    }
70    if (j==1) res=(res+2)%modp;
71    return res;
72}
73long long solve(long long n)
74{
75    get_prime((int)sqrt(n)+1);
76    get_g();
77    return (s(n, 1)+1)%modp;
78}
```

## 1.5. 同余

### 高斯消元

时间复杂度  $O(n^3)$

矩阵行列式：通过高斯消元使得主对角线以下位置都是 0，然后主对角线的乘积就是行列式的绝对值

矩阵的逆：将原矩阵和单位矩阵一起进行高斯消元使得只有主对角线不为 0，当原矩阵变成单位矩阵，原单位矩阵就会变成矩阵的逆

解方程：

```
1//n 个方程，n 个变量，第 n+1 个为等式右侧的常数项
2//对于第 i 个变量，a[now][n+1] 为它的值
3//不存在 now 的表示它为任意值
4//true 表示有解，false 表示无解
5int n;
6double f[Maxn], a[Maxn][Maxn];
7bool gauss()
8{
9    int now=1, x;
10    double t;
11    for (int i=1; i<=n; i++)
12    {
13        for (x=now; x<=n; x++)
14            if (fabs(a[x][i])>eps) break;
15        if (x>n) continue;
16        if (x!=now)
17        {
18            for (int j=1; j<=n+1; j++)
19                swap(a[now][j], a[x][j]);
20        }
21        t=a[now][i];
22        for (int j=1; j<=n+1; j++)
23            a[now][j]/=t;
24        for (int j=1; j<=n; j++)
25            if (j!=now)
26            {
27                t=a[j][i];
28                for (int k=1; k<=n+1; k++)
29                    a[j][k]-=t*a[now][k];
30            }
31        ++now;
32    }
33    for (int i=now; i<=n; i++)
34        if (fabs(a[i][n+1])>eps) return false;
```

```

35     return true;
36 }

    求矩阵的行列式:

1 int n, modp;
2 int a[Maxn][Maxn];
3 int det()
4 {
5     int res=1;
6     for (int i=1; i<=n; i++)
7     {
8         for (int j=i+1; j<=n; j++)
9         {
10            while (a[j][i])
11            {
12                int tmp=a[i][i]/a[j][i];
13                for (int k=i; k<=n; k++)
14                {
15                    a[i][k]=(a[i][k]+modp-(
16                        ↪ 1LL*tmp*a[j][k])%modp)%modp;
17                    swap(a[i][k], a[j][k]);
18                }
19                res=(modp-res)%modp;
20            }
21            res=(1LL*res*a[i][i])%modp;
22        }
23    }
24    return res;

```

## BSGS

求最小的  $x$  使得  $A^x \equiv B \pmod{p}$

其中  $p$  是一个素数

时间复杂度  $O(\sqrt{p})$

```

1 //用了 set 做判断卡常数
2 //solve() 计算最小的 x 使得 A^x=B (mod p)
3 unordered_map<int, int> mp;
4 set<int> s;
5 int power(int a, int b, int modp)
6 {
7     int res=1, now=a, left=b;
8     while (left>0)
9     {
10        if (left%2==1) res=(1LL*res*now)%modp;
11        left/=2;
12        now=(1LL*now*now)%modp;
13    }
14    return res;
15 }
16 void solve(int A, int B, int p)
17 {
18     mp.clear();
19     s.clear();
20     int T=(int)ceil(sqrt(p+0.5));
21     int now=1;
22     for (int i=0; i<T; i++)
23     {
24        if (mp.find(now)==mp.end())
25            ↪ mp[now]=i, s.insert(now);
26        now=(1LL*now*A)%p;
27    }
28    int inv=power(now, p-2, p);
29    int res=B;
30    for (int i=0; i<T; i++)
31    {
32        if (s.find(res)!=s.end())
33        {
34            printf("%d\n", i*T+mp[res]);
35            return;
36        }
37    }

```

```

36         res=(1LL*res*inv)%p;
37     }
38     printf("no solution\n");
39 }

```

## ex-BSGS

求最小的  $x$  使得  $A^x \equiv B \pmod{C}$

其中  $C$  不一定是素数

先特判  $x=0$  的情况

设  $d = \gcd(A, C)$ , 则  $B$  是  $d$  的倍数且  $A^{x-1} * \frac{A}{d} \equiv \frac{B}{d} \pmod{\frac{C}{d}}$

直到  $\gcd(A, C) = 1$  为止, 则  $A^{x-k} * \frac{A^k}{\pi_{i=1}^k d_i} \equiv \frac{B}{\pi_{i=1}^k d_i} \pmod{\frac{C}{\pi_{i=1}^k d_i}}$

枚举  $x = 1, 2, \dots, k-1$  的情况,  $x \geq k$  的情况用 BSGS 完成 (求逆元一定要用 ex-gcd 完成)

时间复杂度  $O(\sqrt{n})$

```

1 //solve() 计算最小的 x 使得 A^x=B (mod C)
2 unordered_map<int, int> mp;
3 void ex_gcd(int a, int b, int &x, int &y)
4 {
5     if (b==0)
6     {
7         x=1;
8         y=0;
9         return;
10    }
11    ex_gcd(b, a%b, x, y);
12    int t=x;
13    x=y;
14    y=t-(a/b)*y;
15 }
16 int inv(int a, int p)
17 {
18     int x, y;
19     ex_gcd(a, p, x, y);
20     x=(x%p+p)%p;
21     return x;
22 }
23 void solve(int A, int B, int C)
24 {
25     A%=C;
26     if (B==1)
27     {
28         printf("%d\n", 0);
29         return;
30     }
31     int k=0, tmp=1;
32     int a=A, b=B, c=C;
33     int d=__gcd(A, c);
34     int now=A;
35     while (d!=1)
36     {
37         if (b%d!=0)
38         {
39             printf("no solution\n");
40             return;
41         }
42         b/=d;
43         c/=d;
44         tmp=(1LL*tmp*(A/d))%c;
45         ++k;
46         if (now==B)
47         {
48             printf("%d\n", k);
49             return;
50         }
51         now=(1LL*now*A)%C;
52         d=__gcd(A, c);

```

```

53     }
54     tmp=inv(tmp,c);
55     b=(1LL*b*tmp)%c;
56     a=A;
57     int T=(int)ceil(sqrt(C+0.5))+1;
58     mp.clear();
59     now=1;
60     for (int i=0;i<T;i++)
61     {
62         if (mp.find(now)==mp.end()) mp[now]=i;
63         now=(1LL*now*a)%c;
64     }
65     now=inv(now,c);
66     for (int i=0;i<=T;i++)
67     {
68         if (mp.find(b)!=mp.end())
69         {
70             printf("%d\n",k+i*T+mp[b]);
71             return;
72         }
73         b=(1LL*b*now)%c;
74     }
75     printf("no solution\n");
76 }

```

## 原根

阶:

设  $m \geq 1$  且  $\gcd(a, m) = 1$ , 则使得  $a^t \equiv 1 \pmod{m}$  成立的最小正整数  $t$  称为  $a$  对模  $m$  的阶, 记作  $\delta_m(a)$ .

性质 1: 设  $m \geq 1$  且  $\gcd(a, m) = 1$ , 若  $a^n \equiv 1 \pmod{m}$  且  $n > 0$ , 则  $\delta_m(a) | n$

性质 2:  $\delta_m(a) | \phi(m)$

推论 1: 若  $p$  和  $q$  都是奇素数, 且  $q | (a^p - 1)$ , 则有  $q | (a - 1)$  或  $2p | (q - 1)$

推论 2:  $2^{2^p} - 1$  的任何因子都是  $2kp + 1$  的形式

原根:

若  $\delta_m(a) = \phi(m)$ , 则  $a$  是  $m$  的一个原根

定理 1: 若  $g$  是  $m$  的一个原根, 则  $g, g^2, \dots, g^{\phi(m)}$  恰好是小于  $m$  且与  $m$  互素的数的排列

定理 2: 若  $m$  有原根, 则原根个数为  $\phi(\phi(m))$

定理 3: 一个数有原根当且仅当  $m = 2, 4, p^e, 2p^e$ , 其中  $p$  是奇素数

推论 1: 若  $d | (p - 1)$ , 则  $x^d \equiv 1 \pmod{p}$  恰有  $d$  个解

推论 2: 若  $p$  为素数,  $d | (p - 1)$ , 则阶为  $d$  的最小剩余  $\pmod{p}$  的个数为  $\phi(d)$

拉格朗日定理: 假设  $f(n) = a_n x^n + \dots + a_1 x + a_0$  是一个次数为  $n$ , 首项系数  $a_n$  不能被  $p$  整除的整系数多项式, 且  $n \geq 1$ , 那么  $f(x)$  至多有  $n$  个模  $p$  不同余的根

性质 1: 一个数的最小原根的大小是  $O(n^{\frac{1}{4}})$  的

性质 2: 如果  $g$  为  $n$  的原根, 则  $g^d$  为  $n$  的原根的充要条件是  $(d, \phi(n)) = 1$

时间复杂度  $O(n \log n)$  (?)

```

1 //flag[] 表示是否有原根
2 //get_g() 求出一个原根
3 //solve() 求出所有的原根
4 //get_frac() 求出所有的素因子
5 #define Maxn 1000007
6 int que[Maxn], tmp=0;
7 int ans[Maxn], tot=0;
8 int prime[Maxn], phi[Maxn], cnt=0;
9 bool vis[Maxn], flag[Maxn];
10 void get_phi(int n)
11 {
12     memset(prime, 0, sizeof(prime));
13     memset(phi, 0, sizeof(phi));
14     phi[1]=1;
15     cnt=0;
16     memset(vis, false, sizeof(vis));
17     vis[1]=true;

```

```

18     for (int i=2; i<=n; i++)
19     {
20         if (!vis[i])
21         {
22             prime[++cnt]=i;
23             phi[i]=i-1;
24         }
25         for (int j=1; j<=cnt && prime[j]<=n/i; j++)
26         {
27             vis[i*prime[j]]=true;
28             if (i%prime[j]==0)
29             {
30                 phi[i*prime[j]]=phi[i]*prime[j];
31                 break;
32             }
33             phi[i*prime[j]]=phi[i]*(prime[j]-1);
34         }
35     }
36 }
37 void get_flag(int n)
38 {
39     memset(flag, false, sizeof(flag));
40     flag[2]=true;
41     flag[4]=true;
42     for (int i=2; i<=cnt; i++)
43     {
44         int now=prime[i];
45         flag[now]=true;
46         if (2*now<=n) flag[2*now]=true;
47         while (now<=n/prime[i])
48         {
49             now*=prime[i];
50             flag[now]=true;
51             if (2*now<=n) flag[2*now]=true;
52         }
53     }
54 }
55 void get_frac(int n)
56 {
57     tmp=0;
58     for (int i=1; i<=cnt && prime[i]*prime[i]<=n; i++)
59         if (n%prime[i]==0)
60         {
61             que[++tmp]=prime[i];
62             while (n%prime[i]==0) n/=prime[i];
63         }
64     if (n!=1) que[++tmp]=n;
65 }
66 int power(int a, int b, int modp)
67 {
68     int res=1, now=a, left=b;
69     while (left>0)
70     {
71         if (left%2==1) res=(1LL*res*now)%modp;
72         left/=2;
73         now=(1LL*now*now)%modp;
74     }
75     return res;
76 }
77 int get_g()
78 {
79     get_frac(phi[n]);
80     for (int i=2; i<=n; i++)
81     {
82         if (power(i, phi[n], n)!=1) continue;
83         int j=1;
84         for (; j<=tmp; j++)
85             if (power(i, phi[n]/que[j], n)==1) break;

```



```

86         if (j>tmp) return i;
87     }
88 }
89 void get_all()
90 {
91     tot=1;
92     ans[1]=get_g();
93     int now=(1LL*ans[1]*ans[1])%n;
94     for (int i=2;i<phi[n];i++,now=(1LL*now*ans[1])%n)
95         if (__gcd(i,phi[n])==1) ans[++tot]=now;
96     sort(ans+1,ans+tot+1);
97     for (int i=1;i<tot;i++)
98         printf("%d ",ans[i]);
99     printf("%d\n",ans[tot]);
100 }
101 void solve(int n)
102 {
103     get_phi(1000000);
104     get_flag(1000000);
105     if (n<=3)
106     {
107         printf("%d\n",n-1);
108         return;
109     }
110     if (!flag[n])
111     {
112         printf("%d\n",-1);
113         return;
114     }
115     get_all();
116 }

```

### 中国剩余定理

时间复杂度  $O(n \log n)$

```

1 //flag 表示是否有解
2 //Ans % a[i] = b[i]
3 //Ans = k*A + B
4 #define Maxn 100007
5 int n;
6 long long A,B;
7 long long a[Maxn],b[Maxn];
8 bool flag;
9 long long ex_gcd(long long a,long long b,long long
    ↪ &x,long long &y)
10 {
11     if (b==0LL)
12     {
13         x=1LL;
14         y=0LL;
15         return a;
16     }
17     long long res=ex_gcd(b,a%b,x,y);
18     long long t=x;
19     x=y;
20     y=t-1LL*(a/b)*y;
21     return res;
22 }
23 void solve()
24 {
25     A=1LL,B=0LL;
26     flag=true;
27     for (int i=1;i<=n;i++)
28     {
29         long long x,y;
30         long long d=ex_gcd(A,a[i],x,y);
31         if ((b[i]-B)%d)
32         {
33             flag=false;
34             return;

```

```

35     }
36     x=(1LL*x*((b[i]-B)/d)+a[i]/d)%(a[i]/d);
37     y=1LL*(A/d)*a[i];
38     B=((1LL*x*A+B)%y+y)%y;
39     A=y;
40 }
41 }

```

### 二次剩余

求解  $x^2 \equiv N \pmod{P}$   
其中  $P$  为质数

```

1 //solve() 求解
2 //无解返回-1
3 //有解返回 r, 表示答案为 r 或 p-r
4 long long multiply_mod(long long x,long long y,long
    ↪ long p)
5 {
6     return ((x*y-(long long)((long
    ↪ double)x*y+0.5)/p)*p)%p;p;
7 }
8 long long power_mod(long long a,long long b,long long
    ↪ p)
9 {
10     long long res=1LL,now=a,left=b;
11     while (left>0)
12     {
13         if (left&1) res=multiply_mod(res,now,p);
14         left/=2;
15         now=multiply_mod(now,now,p);
16     }
17     return res;
18 }
19 long long ToneLLi_Shanks(long long n,long long p)
20 {
21     if (n==0) return 0LL;
22     if (p==2LL) return (n&1)?1:-1;
23     if (power_mod(n,p>>1,p)!=1) return -1;
24     if (p&2) return power_mod(n,(p+1)>>2,p);
25     int s=__builtin_ctzll(p^1);
26     long long q=p>>s,z=2LL;
27     for (;power_mod(z,p>>1,p)==1;++z);
28     long long c=power_mod(z,q,p);
29     long long r=power_mod(n,(q+1)>>1,p);
30     long long t=power_mod(n,q,p),tmp;
31     for (int m=s,i;t!=1;)
32     {
33         for (i=0,tmp=t;tmp!=1;++i)
34             tmp=multiply_mod(tmp,tmp,p);
35         for (;i<--m;)
36             c=multiply_mod(c,c,p);
37         r=multiply_mod(r,c,p);
38         c=multiply_mod(c,c,p);
39         t=multiply_mod(t,c,p);
40     }
41     return r;
42 }

```

### 1.6. 组合数

计算  $C(n,m) \bmod p$ , 其中  $p$  是一个素数  
时间复杂度  $O(p + \log n)$

### 1.7. 单纯形

有  $m$  个实数变量和  $n$  个约束条件

$$x_j \geq 0$$

$$\sum_{j=1}^m a_{ij} x_j \leq b_i$$

要使得  $z = \sum_{j=1}^m c_j x_j$  最大

判断是否有解并给出一组合法解

uoj 97pts 代码:

```

1 //op==1 表示需要输出一组合法解
2 //n 个变量
3 //m 个约束条件
4 //sum {i=1,2,...,n} a[i][j] * x[j] <= a[i][n+1]
5 //最大化 z = sum {i=1,2,...,n} a[m+1][i] * x[i]
6 #include <bits/stdc++.h>
7 #define Maxn 107
8 #define inf 1e9
9 #define eps 1e-9
10 using namespace std;
11 double a[Maxn][Maxn];
12 int id[Maxn*2], pos[Maxn];
13 int n, m, op;
14 int sgn(double x)
15 {
16     if (fabs(x) < eps) return 0;
17     else if (x < 0) return -1;
18     else return 1;
19 }
20 void pivot(int x, int y)
21 {
22     swap(id[x], id[y+n]);
23     double t = a[y][x];
24     a[y][x] = -1.0;
25     for (int i = 1; i <= n+1; i++)
26         a[y][i] /= -t;
27     for (int i = 1; i <= m+1; i++)
28         if (i != y)
29         {
30             double t = a[i][x];
31             if (sgn(t) == 0) continue;
32             a[i][x] = 0.0;
33             for (int j = 1; j <= n+1; j++)
34                 a[i][j] += t * a[y][j];
35         }
36 }
37 double simplex()
38 {
39     while (true)
40     {
41         int x = 0;
42         for (int i = 1; i <= n; i++)
43             if (sgn(a[m+1][i]) > 0)
44             {
45                 x = i;
46                 break;
47             }
48         if (x == 0) return a[m+1][n+1];
49         int y = 0;
50         double mn = inf;
51         for (int i = 1; i <= m; i++)
52         {
53             if (sgn(a[i][x]) >= 0) continue;
54             double t = a[i][n+1] / -a[i][x];
55             if (t < mn)
56             {
57                 mn = t;
58                 y = i;
59             }
60         }
61         if (y == 0) return inf;
62         pivot(x, y);
63     }
64 }
65 bool init()
66 {
67     while (true)
68     {
69         int x = 0, y = 0;

```

```

70         for (int i = 1; i <= m; i++)
71             if (sgn(a[i][n+1]) < 0 && (!x || rand() & 1)) x = i;
72         if (!x) return 1;
73         for (int i = 1; i <= n; i++)
74             if (sgn(a[x][i]) > 0 && (!y || rand() & 1)) y = i;
75         if (!y) return 0;
76         pivot(y, x);
77     }
78 }
79 int main()
80 {
81     scanf("%d%d%d", &n, &m, &op);
82     for (int i = 1; i <= n; i++)
83         scanf("%lf", &a[m+1][i]);
84     for (int i = 1; i <= m; i++)
85     {
86         for (int j = 1; j <= n; j++)
87         {
88             scanf("%lf", &a[i][j]);
89             a[i][j] = -a[i][j];
90         }
91         scanf("%lf", &a[i][n+1]);
92     }
93     for (int i = 1; i <= n+m; i++)
94         id[i] = i;
95     if (!init()) printf("Infeasible\n");
96     else
97     {
98         double ans = simplex();
99         if (sgn(ans - inf) == 0) printf("Unbounded\n");
100        else
101        {
102            printf("%.8lf\n", ans);
103            if (op == 1)
104            {
105                for (int i = 1; i <= n+m; i++)
106                    pos[id[i]] = i;
107                for (int i = 1; i <= n; i++)
108                    if (pos[i] <= n) printf("%.8lf\n", 0.0);
109                    else printf("%.8lf\n", a[pos[i]-n][n+1]);
110            }
111        }
112    }
113    return 0;
114 }

```

uoj std:

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 template <typename T> void chmin(T &x, const T &y)
5 {
6     if (x > y) x = y;
7 }
8 template <typename T> void chmax(T &x, const T &y)
9 {
10    if (x < y) x = y;
11 }
12 typedef long long s64;
13 #define rep(i, l, r) for(int i = l; i <= r; ++i)
14 #define per(i, r, l) for(int i = r; i >= l; --i)
15 #define gc (c=getchar())
16 int read()
17 {
18     char c;
19     while(gc < '-');
20     if(c == '-')

```

```

21 {
22     int x=gc-'0';
23     while(gc>='0')x=x*10+c-'0';
24     return -x;
25 }
26 int x=c-'0';
27 while(gc>='0')x=x*10+c-'0';
28 return x;
29 }
30 #undef gc
31
32 const int W=1e8,B=40,T=120;
33 struct db
34 {
35     int a[T],n;
36     bool is_neg;
37     int& operator [] (int x)
38     {
39         return a[x];
40     }
41     int operator [] (int x) const
42     {
43         return a[x];
44     }
45     void left_move(int l)
46     {
47         per(i,n-1,0)a[i+1]=a[i];
48         rep(i,0,l-1)a[i]=0;
49         n+=1;
50     }
51     db (int x=0)
52     {
53         memset(a,0,sizeof(a));n=B;
54         is_neg=0;
55         if(x<0){is_neg=1;x=-x;}
56         while(x){a[n++]=x%W;x/=W;}
57     }
58     void print(char c='\n') const
59     {
60         if(is_neg) putchar('-');
61         if(n>B)
62         {
63             printf("%d",a[n-1]);
64             per(i,n-2,B) printf("%.8d",a[i]);
65         }
66         else printf("0");
67
68         printf(".");
69         per(i,B-1,B-2) printf("%.8d",a[i]);
70         printf("%c",c);
71     }
72     long double evalu(int l) const
73     {
74         long double x=0;
75         per(i,n-1,l)x=x*W+a[i];
76         return x;
77     }
78 };
79 bool operator <(const db &a,const db &b)
80 {
81     if(a.is_neg)
82     {
83         if(!b.is_neg) return 1;
84         per(i,T-1,0)
85         if(a[i]!=b[i]) return a[i]>b[i];
86         return 0;
87     }
88     else
89     {

```

```

90         if(b.is_neg) return 0;
91         per(i,T-1,0)
92         if(a[i]!=b[i]) return a[i]<b[i];
93         return 0;
94     }
95 }
96 bool operator >=(const db &a,const db &b)
97 {
98     return !(a<b);
99 }
100 bool operator <=(const db &a,const db &b)
101 {
102     return b>=a;
103 }
104 bool operator >(const db &a,const db &b)
105 {
106     return b<a;
107 }
108 bool operator !(const db &a)
109 {
110     int n=a.n;
111     while(n&&!a[n-1])--n;
112     return !n;
113 }
114 db operator -(db a)
115 {
116     a.is_neg^=1;
117     return a;
118 }
119 db operator +(db a,const db &b);
120 db operator -(db a,db b)
121 {
122     b=-b;
123     if(a.is_neg)
124     {
125         if(!b.is_neg)
126         {
127             a=-a;
128             swap(a,b);
129         }
130         else return a+b;
131     }
132     else
133     {
134         if(b.is_neg)
135         {
136             b=-b;
137         }
138         else return a+b;
139     }
140     if(a<b) return -(b-a);
141
142     rep(i,0,a.n-1)
143     if((a[i]-=b[i])<0)
144     {
145         a[i]+=W;
146         --a[i+1];
147     }
148     while(a.n&&!a[a.n-1])--a.n;
149     return a;
150 }
151 db operator +(db a,const db &b)
152 {
153     if(a.is_neg)
154     {
155         if(!b.is_neg) return b-(-a);
156     }
157     else

```

```

158 {
159     if(b.is_neg)return a-(-b);
160 }
161 chmax(a.n,b.n);
162 rep(i,0,a.n-1)
163 if((a[i]+b[i])>=W)
164 {
165     a[i]-=W;
166     ++a[i+1];
167 }
168 if(a[a.n])++a.n;
169 return a;
170 }
171 void operator +=(db &a,const db &b)
172 {
173     a=a+b;
174 }
175 void operator -=(db &a,const db &b)
176 {
177     a=a-b;
178 }
179 db eps1,eps2,eps3;
180 db operator *(const db &a,const db &b)
181 {
182     db ans;
183     ans.is_neg=a.is_neg^b.is_neg;
184     ans.n=max(0,a.n+b.n-B);
185     rep(i,0,ans.n)
186     {
187         int jk=i+B;s64 sum=0;
188         rep(j,max(0,jk-(b.n-1)),min(jk,a.n-1))sum+=(
189             ↪ s64)a[j]*b[jk-j];
189         int x=i;
190         while(sum)
191         {
192             ans[x]+=sum%W;
193             sum/=W;
194             ++x;
195         }
196     }
197     rep(i,0,ans.n)
198     while(ans[i]>=W)
199     {
200         ans[i]-=W;
201         ++ans[i+1];
202     }
203     if(ans.n&&!ans[ans.n-1])--ans.n;
204     return ans;
205 }
206 void operator *=(db &a,const db &b)
207 {
208     a=a*b;
209 }
210 db operator *(db a,int k)
211 {
212     per(i,a.n-1,0)
213     {
214         s64 sum=(s64)a[i]*k;
215         a[i]=sum%W;
216         a[i+1]+=sum/W;
217     }
218     rep(i,0,a.n-1)
219     while(a[i]>=W)
220     {
221         a[i]-=W;
222         ++a[i+1];
223     }
224     if(a[a.n])++a.n;
225     return a;

```

```

226 }
227 db operator *(int k,const db &a)
228 {
229     return a*k;
230 }
231 db operator /(db a,db b)
232 {
233     a.is_neg^=b.is_neg;
234     b.is_neg=0;
235     a.left_move(B);
236     int l=max(0,b.n-20);
237     long double b_e=b.eval(1);
238     db x;x.n=0;
239     per(i,a.n-1,0)
240     {
241         x.left_move(1);x[0]=a[i];
242         if(x>=b)
243         {
244             int k=x.eval(1)/b_e;
245             if(k)--k;
246             a[i]=k;
247             x-=k*b;
248             while(x>=b)
249             {
250                 x-=b;
251                 ++a[i];
252             }
253             }else a[i]=0;
254         }
255     while(a.n&&!a[a.n-1])--a.n;
256     return a;
257 }
258 void operator /=(db &a,const db &b)
259 {
260     a=a/b;
261 }
262 int cmp(const db &x,const db &eps)
263 {
264     return x<-eps?-1:x>eps;
265 }
266 const int N=20+1,M=20+1,inf=1e9;
267 db a[M][N],b[M];
268 int idn[N],idm[M];
269 int n,m;
270 void pivot(int x,int y)
271 {
272     swap(idm[x],idn[y]);
273     db k=a[x][y];
274     rep(j,1,n)a[x][j]/=k;
275     b[x]/=k;
276     a[x][y]=1/k;
277     rep(i,0,m)
278     if(i!=x)
279     {
280         k=a[i][y];
281         b[i]-=k*b[x];
282         a[i][y]=0;
283         rep(j,1,n)a[i][j]-=a[x][j]*k;
284     }
285 }
286 void simplex()
287 {
288     idn[0]=inf;
289     while(1)
290     {
291         int y=0;
292         rep(j,1,n)
293         if(cmp(a[0][j],eps1)>0&&idn[j]<idn[y])y=j;

```

```

294     if(!y)break;
295     int x=0;
296     rep(i,1,m)
297     if(cmp(a[i][y],eps1)>0)
298     {
299         if(!x)x=i;
300         else
301         {
302             int t=cmp(
303                 ↪ b[i]/a[i][y]-b[x]/a[x][y],eps2);
304             if(t<0||(t==0&&idm[i]<idm[x]))x=i;
305         }
306     }
307     if(!x)
308     {
309         puts("Unbounded");
310         exit(0);
311     }
312     pivot(x,y);
313 }
314 void init()
315 {
316     idm[0]=inf;idn[0]=inf;
317     while(1)
318     {
319         int x=0;
320         rep(i,1,m)
321         if(cmp(b[i],eps3)<0&&idm[i]<idm[x])x=i;
322         if(!x)break;
323         int y=0;
324         rep(j,1,n)
325         if(cmp(a[x][j],eps3)<0&&idn[j]<idn[y])y=j;
326         if(!y)
327         {
328             puts("Infeasible");
329             exit(0);
330         }
331         pivot(x,y);
332     }
333 }
334
335 int main()
336 {
337     int t;
338     n=read();m=read();t=read();
339     rep(j,1,n)a[0][j]=read();
340     rep(i,1,m)
341     {
342         rep(j,1,n)a[i][j]=read();
343         b[i]=read();
344     }
345
346     eps1.n=B-20;eps1[eps1.n-1]=1;
347     eps2.n=B;eps2[eps2.n-1]=1;
348     eps3.n=B-20;eps3[eps3.n-1]=1;
349
350     rep(j,1,n)idn[j]=j;
351     rep(i,1,m)idm[i]=n+i;
352     init();
353     simplex();
354     (-b[0]).print('\n');
355     if(t)
356     {
357         rep(j,1,n)
358         {
359             rep(i,1,m)
360             if(idm[i]==j)
361             {

```

```

362                 b[i].print(' ');
363                 goto Find ;
364             }
365             printf("0 ");
366             Find : ;
367         }
368         puts("");
369     }
370 }

```

## 1.8. 多项式

### FFT

时间复杂度  $O(n \log n)$

```

1 //FFT 多项式乘法
2 //n 和 m 表示两个多项式次数
3 //a[i].real,b[i].real 表示 i 次项
4 //结果保存在 a 中
5 #define MAXN 262144
6 typedef struct
7 {
8     double real,imag;
9 } com;
10 com com_add(com a,com
11     ↪ b){return(com){a.real+b.real,a.imag+b.imag};}
12 com com_sub(com a,com
13     ↪ b){return(com){a.real-b.real,a.imag-b.imag};}
14 com com_mul(com a,com b){return(com)
15     ↪ {a.real*b.real-a.imag*b.imag,a.real*b.imag+a.imag*b.real}}
16 int n,m;
17 com a[MAXN+7],b[MAXN+7];
18 void fft(com *a, int n, int flag)
19 {
20     for (int i=n/2,j=1;j<n;++j)
21     {
22         if (i<j) swap(a[i],a[j]);
23         int k=n/2;
24         while (i&k) {i^=k;k/=2;}
25         i^=k;
26     }
27     for (int k=2;k<=n;k*=2)
28     {
29         com root=(com){cos(M_PI/k*flag*2),sin(
30             ↪ M_PI/k*flag*2)};
31         for (int i=0;i<n;i+=k)
32         {
33             com w=(com){1.0, 0.0};
34             for (int j=i;j<i+k/2;++j)
35             {
36                 com u=a[j],v=com_mul(a[j+k/2],w);
37                 a[j]=com_add(u,v);
38                 a[j+k/2]=com_sub(u,v);
39                 w=com_mul(w,root);
40             }
41         }
42     }
43     if (flag== -1)
44     {
45         for (int i=0;i<n;++i)
46             a[i].real=(int)trunc(a[i].real/n+0.5);
47     }
48 }
49 void multiply()
50 {
51     ++n,++m;
52     int len=2;
53     while (len<m+n) len<=1;
54     fft(a,len,1);fft(b,len,1);
55     for (int i=0;i<len;++i) a[i]=com_mul(a[i],b[i]);

```

```

52     fft(a,len,-1);
53 }

NTT
    时间复杂度  $O(n \log n)$ 

1 //NTT 多项式乘法
2 //n 和 m 表示两个多项式次数
3 //a[i],b[i] 表示 i 次项
4 //结果保存在 a 中
5 #define Maxn 262144
6 #define modp 998244353
7 #define g 3
8 int n,m,len;
9 long long a[Maxn+7],b[Maxn+7];
10 void ntt(long long *a, int n, int flag)
11 {
12     for (int i=n/2,j=1;j<n;++j)
13     {
14         if (i<j) swap(a[i],a[j]);
15         int k=n/2;
16         while (i&k) {i^=k;k/=2;}
17         i^=k;
18     }
19     for (int k=2;k<=n;k*=2)
20     {
21         long long rt=power(g,(modp-1)/k);
22         if (flag==-1) rt=power(rt,modp-2);
23         for (int i=0;i<n;i+=k)
24         {
25             long long del=1;
26             for (int j=i;j<i+k/2;++j)
27             {
28                 long long
29                 ↪ u=a[j],v=(1LL*a[j+k/2]*del)%modp;
30                 a[j]=(u+v)%modp;
31                 a[j+k/2]=(u+modp-v)%modp;
32                 del=(1LL*del*rt)%modp;
33             }
34         }
35         if (flag==1)
36         {
37             long long ni=power(n,modp-2);
38             for (int i=0;i<n;i++) a[i]=(1LL*a[i]*ni)%modp;
39         }
40 }
41 void multiply()
42 {
43     ++n,++m;
44     int len=2;
45     while (len<m+n) len<=1;
46     ntt(a,len,1),ntt(b,len,1);
47     for (int i=0;i<len;i++) a[i]=(1LL*a[i]*b[i])%modp;
48     ntt(a,len,-1);
49 }

```

### 任意模数 FFT

时间复杂度  $O(n \log n)$

```

1 typedef complex<double> Complex;
2 #define modp 1000000007
3 #define PI acos(-1.0)
4 const int N=131072,L=16,MASK=(1<<L)-1;
5 Complex w[N+7];
6 void FFT_init()
7 {
8     for (int i=0;i<N;i++)
9         w[i]=Complex(cos(2*i*PI/N),sin(2*i*PI/N));
10 }
11 void FFT(Complex p[],int n)

```

```

12 {
13     for (int i=1,j=0;i<n-1;++i)
14     {
15         for (int s=n;j^=s>>=1,~j&s);
16         if (i<j) swap(p[i],p[j]);
17     }
18     for (int d=0;(1<<d)<n;++d)
19     {
20         int m=1<<d,m2=m*2,rm=n>>(d+1);
21         for (int i=0;i<n;i+=m2)
22             for (int j=0;j<m;j++)
23             {
24                 Complex &p1=p[i+j+m],&p2=p[i+j];
25                 Complex t=w[rm*j]*p1;
26                 p1=p2-t,p2=p2+t;
27             }
28     }
29 }
30 Complex A[N+7],B[N+7],C[N+7],D[N+7];
31 int a[N+7],b[N+7];
32 void mul()
33 {
34     FFT_init();
35     for (int i=0;i<N;i++)
36     {
37         A[i]=Complex(a[i]>>L,a[i]&MASK);
38         B[i]=Complex(b[i]>>L,b[i]&MASK);
39     }
40     FFT(A,N),FFT(B,N);
41     for (int i=0;i<N;++i)
42     {
43         int j=(N-i)%N;
44         Complex da=(A[i]-conj(A[j]))*Complex(0,-0.5),
45                 db=(A[i]+conj(A[j]))*Complex(0.5,0),
46                 dc=(B[i]-conj(B[j]))*Complex(0,-0.5),
47                 dd=(B[i]+conj(B[j]))*Complex(0.5,0);
48         C[j]=da*dd+da*dc*Complex(0,1);
49         D[j]=db*dd+db*dc*Complex(0,1);
50     }
51     FFT(C,N),FFT(D,N);
52     for (int i=0;i<N;++i)
53     {
54         long long da=(long
55         ↪ long)(C[i].imag()/N+0.5)%modp,
56         db=(long
57         ↪ long)(C[i].real()/N+0.5)%modp,
58         dc=(long
59         ↪ long)(D[i].imag()/N+0.5)%modp,
60         dd=(long
61         ↪ long)(D[i].real()/N+0.5)%modp;
62         a[i]=((dd<<(L*2))+((db+dc)<<L)+da)%modp;
63     }
64 }

```

### FWT

时间复杂度  $O(n \log n)$

```

1 #define Maxn 65536
2 int n,m;
3 long long a[Maxn],b[Maxn];
4 void fwt(long long *a,int n,long long f)
5 {
6     for (int i=1;i<n;i<=1)
7         for (int j=0;j<n;j+=(i<<1))
8             for (int k=0;k<i;k++)
9             {
10                 long long x=a[j+k],y=a[j+k+i];
11                 a[j+k]=1LL*f*(x+y)%modp;
12                 a[j+k+i]=1LL*f*(x-y+modp)%modp;
13                 //fwt:

```

```

14         //xor:a[j+k]=x+y,a[j+k+i]=(x-y+mod)
15         ↪ %mod;
16         //and:a[j+k]=x+y;
17         //or:a[j+k+i]=x+y;
18         //dfwt:
19         //xor:a[j+k]=(x+y)/2,a[j+k+i]=(x-y)/2;
20         //and:a[j+k]=x-y;
21         //or:a[j+k+i]=y-x;
22     }
23 void multiply()
24 {
25     ++n,++m;
26     int len=1;
27     while (len<=n||len<=m) len<<=1;
28     fwt(a,len,1),fwt(b,len,1);
29     for (int i=0;i<len;i++)
30         a[i]=(1LL*a[i]*b[i])%modp;
31     fwt(a,len,(modp+1)/2);
32 }

```

### 多项式求逆

给定一个多项式  $F(x)$ ，求出一个多项式  $G(x)$ ，满足  $F(x) * G(x) \equiv 1 \pmod{x^n}$   
 时间复杂度  $O(n \log n)$

```

1#include <bits/stdc++.h>
2using namespace std;
3#define Maxn 524288
4#define modp 998244353
5#define g 3
6int power(int a,int b)
7{
8    int res=1,now=a,left=b;
9    while (left>0)
10    {
11        if (left%2==1) res=(1LL*res*now)%modp;
12        left/=2;
13        now=(1LL*now*now)%modp;
14    }
15    return res;
16}
17#define meminit(A,l,r)
18    ↪ memset(A+(l),0,sizeof(*A)*((r)-(l)))
19#define memcpy(B,A,l,r)
20    ↪ memcpy(B,A+(l),sizeof(*A)*((r)-(l)))
21void FFT(int *a,int n,int f)
22{
23    for (int i=0,j=0;i<n;i++)
24    {
25        if (i>j) swap(a[i],a[j]);
26        for (int t=n>>1;(j^=t)<t;t>>=1);
27    }
28    for (int i=2;i<=n;i<=1)
29    {
30        static int exp[Maxn+7];
31        exp[0]=1;
32        exp[1]=power(g,(modp-1)/i);
33        if (f==1) exp[1]=power(exp[1],modp-2);
34        for (int k=2;k<(i>>1);k++)
35            exp[k]=(1LL*exp[k-1]*exp[1])%modp;
36        for (int j=0;j<n;j+=i)
37            for (int k=0;k<(i>>1);k++)
38            {
39                int &pA=a[j+k],&pB=a[j+k+(i>>1)];
40                int B=(1LL*pB*exp[k])%modp;
41                pB=(pA-B+modp)%modp;
42                pA=(pA+B)%modp;
43            }
44    }
45 }

```

```

43     if (f==1)
44     {
45         int rev=power(n,modp-2);
46         for (int i=0;i<n;i++)
47         {
48             a[i]=(1LL*a[i]*rev)%modp;
49             if (a[i]<0) a[i]+=modp;
50         }
51     }
52 }
53 void getInv(int *a,int *b,int n)
54 {
55     static int tmp[Maxn+7];
56     b[0]=power(a[0],modp-2);
57     for (int c=2,M=1;c<(n<1);c<<=1)
58     {
59         for (;M<=3*(c-1);M<<=1);
60         meminit(b,c,M);
61         meminit(tmp,c,M);
62         memcpy(tmp,a,0,c);
63         FFT(tmp,M,0);
64         FFT(b,M,0);
65         for (int i=0;i<M;i++)
66             b[i]=1LL*b[i]*(
67                 ↪ 2LL-1LL*tmp[i]*b[i]%modp+modp)%modp;
68         FFT(b,M,1);
69         meminit(b,c,M);
70     }
71 }
72 int n,a[Maxn+7],b[Maxn+7];
73 int main()
74 {
75     scanf("%d",&n);
76     for (int i=0;i<n;i++)
77     {
78         scanf("%d",&a[i]);
79         a[i]%=modp;
80     }
81     getInv(a,b,n);
82     for (int i=0;i<n;i++)
83         printf("%d ",b[i]);
84     printf("\n");
85     return 0;
86 }

```

### 多项式取模

给定一个  $n$  次多项式  $F(x)$  和一个  $m$  次多项式  $G(x)$ ，求出多项式  $Q(x)$ ， $R(x)$ ，满足以下条件：

$Q(x)$  次数为  $n-m$ ， $R(x)$  次数小于  $m$ ， $F(x) = Q(x) * G(x) + R(x)$

时间复杂度  $O(n \log n)$

```

1#include <bits/stdc++.h>
2using namespace std;
3#define Maxn 524288
4#define modp 998244353
5#define g 3
6int power(int a,int b)
7{
8    int res=1,now=a,left=b;
9    while (left>0)
10    {
11        if (left%2==1) res=(1LL*res*now)%modp;
12        left/=2;
13        now=(1LL*now*now)%modp;
14    }
15    return res;
16}
17#define meminit(A,l,r)
18    ↪ memset(A+(l),0,sizeof(*A)*((r)-(l)))

```

```

18 #define memcpy(B,A,l,r)
   ↪ memcpy(B,A+(l),sizeof(*A)*((r)-(l)))
19 void FFT(int *a,int n,int f)
20 {
21     for (int i=0,j=0;i<n;i++)
22     {
23         if (i>j) swap(a[i],a[j]);
24         for (int t=n>>1;(j^=t)<t;t>>=1);
25     }
26     for (int i=2;i<=n;i<=1)
27     {
28         static int exp[Maxn+7];
29         exp[0]=1;
30         exp[1]=power(g,(modp-1)/i);
31         if (f==1) exp[1]=power(exp[1],modp-2);
32         for (int k=2;k<(i>>1);k++)
33             exp[k]=(1LL*exp[k-1]*exp[1])%modp;
34         for (int j=0;j<n;j+=i)
35             for (int k=0;k<(i>>1);k++)
36             {
37                 int &pA=a[j+k],&pB=a[j+k+(i>>1)];
38                 int B=(1LL*pB*exp[k])%modp;
39                 pB=(pA-B+modp)%modp;
40                 pA=(pA+B)%modp;
41             }
42     }
43     if (f==1)
44     {
45         int rev=power(n,modp-2);
46         for (int i=0;i<n;i++)
47         {
48             a[i]=(1LL*a[i]*rev)%modp;
49             if (a[i]<0) a[i]+=modp;
50         }
51     }
52 }
53 void getInv(int *a,int *b,int n)
54 {
55     static int tmp[Maxn+7];
56     b[0]=power(a[0],modp-2);
57     for (int c=2,M=1;c<(n<1);c<=1)
58     {
59         for (;M<=3*(c-1);M<=1);
60         meminit(b,c,M);
61         meminit(tmp,c,M);
62         memcpy(tmp,a,0,c);
63         FFT(tmp,M,0);
64         FFT(b,M,0);
65         for (int i=0;i<M;i++)
66             b[i]=1LL*b[i]*((
   ↪ 2LL-1LL*tmp[i]*b[i]%modp+modp)%modp;
67         FFT(b,M,1);
68         meminit(b,c,M);
69     }
70 }
71 void divide(int n,int m,int *a,int *b,int *d,int *r)
72 {
73     static int
   ↪ M,tA[Maxn+7],tB[Maxn+7],inv[Maxn+7],tD[Maxn+7];
74     for (;n>0&&a[n-1]==0;n--);
75     for (;m>0&&b[m-1]==0;m--);
76     for (int i=0;i<n;i++)
77         tA[i]=a[n-i-1];
78     for (int i=0;i<m;i++)
79         tB[i]=b[m-i-1];
80     for (M=1;M<=n-m+1;M<=1);
81     if (m<M) meminit(tB,m,M);
82     getInv(tB,inv,M);
83     for (M=1;M<=2*(n-m+1);M<=1);

```

```

84     meminit(inv,n-m+1,M);
85     meminit(tA,n-m+1,M);
86     FFT(inv,M,0);
87     FFT(tA,M,0);
88     for (int i=0;i<M;i++)
89         d[i]=(1LL*inv[i]*tA[i])%modp;
90     FFT(d,M,1);
91     reverse(d,d+n-m+1);
92     for (M=1;M<=n;M<=1);
93     memcpy(tB,b,0,m);
94     if (m<M) meminit(tB,m,M);
95     memcpy(tD,d,0,n-m+1);
96     meminit(tD,n-m+1,M);
97     FFT(tD,M,0);
98     FFT(tB,M,0);
99     for (int i=0;i<M;i++)
100         r[i]=(1LL*tD[i]*tB[i])%modp;
101     FFT(r,M,1);
102     meminit(r,n,M);
103     for (int i=0;i<n;i++)
104         r[i]=(a[i]-r[i]+modp)%modp;
105 }
106 int n,m;
107 int a[Maxn+7],b[Maxn+7],d[Maxn+7],r[Maxn+7];
108 int main()
109 {
110     scanf("%d%d",&n,&m);
111     for (int i=0;i<=n;i++)
112     {
113         scanf("%d",&a[i]);
114         a[i]%=modp;
115     }
116     ++n;
117     for (int i=0;i<=m;i++)
118     {
119         scanf("%d",&b[i]);
120         b[i]%=modp;
121     }
122     ++m;
123     divide(n,m,a,b,d,r);
124     for (int i=0;i<=n-m;i++)
125         printf("%d ",d[i]);
126     printf("\n");
127     for (int i=0;i<=m-1;i++)
128         printf("%d ",r[i]);
129     printf("\n");
130     return 0;
131 }

```

### 多项式开根

时间复杂度  $O(n \log n)$

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define Maxn 524288
4 #define modp 998244353
5 #define g 3
6 int power(int a,int b)
7 {
8     int res=1,now=a,left=b;
9     while (left>0)
10     {
11         if (left%2==1) res=(1LL*res*now)%modp;
12         left/=2;
13         now=(1LL*now*now)%modp;
14     }
15     return res;
16 }
17 #define meminit(A,l,r)
   ↪ memset(A+(l),0,sizeof(*A)*((r)-(l)))

```



```

18 #define memcpy(B,A,l,r)
   ↪ memcpy(B,A+(l),sizeof(*A)*((r)-(l)))
19 void FFT(int *a,int n,int f)
20 {
21     for (int i=0,j=0;i<n;i++)
22     {
23         if (i>j) swap(a[i],a[j]);
24         for (int t=n>>1;(j^=t)<t;t>>=1);
25     }
26     for (int i=2;i<=n;i<=<=1)
27     {
28         static int exp[Maxn+7];
29         exp[0]=1;
30         exp[1]=power(g,(modp-1)/i);
31         if (f==1) exp[1]=power(exp[1],modp-2);
32         for (int k=2;k<(i>>1);k++)
33             exp[k]=(1LL*exp[k-1]*exp[1])%modp;
34         for (int j=0;j<n;j+=i)
35             for (int k=0;k<(i>>1);k++)
36             {
37                 int &pA=a[j+k],&pB=a[j+k+(i>>1)];
38                 int B=(1LL*pB*exp[k])%modp;
39                 pB=(pA-B+modp)%modp;
40                 pA=(pA+B)%modp;
41             }
42     }
43     if (f==1)
44     {
45         int rev=power(n,modp-2);
46         for (int i=0;i<n;i++)
47         {
48             a[i]=(1LL*a[i]*rev)%modp;
49             if (a[i]<0) a[i]+=modp;
50         }
51     }
52 }
53 void getInv(int *a,int *b,int n)
54 {
55     static int tmp[Maxn+7];
56     b[0]=power(a[0],modp-2);
57     for (int c=2,M=1;c<(n<<1);c<=<=1)
58     {
59         for (;M<=3*(c-1);M<=<=1);
60         meminit(b,c,M);
61         meminit(tmp,c,M);
62         memcpy(tmp,a,0,c);
63         FFT(tmp,M,0);
64         FFT(b,M,0);
65         for (int i=0;i<M;i++)
66             b[i]=1LL*b[i]*(
   ↪ 2LL-1LL*tmp[i]*b[i]%modp+modp)%modp;
67         FFT(b,M,1);
68         meminit(b,c,M);
69     }
70 }
71 void getSqrt(int *a,int *b,int n)
72 {
73     static int tmp[Maxn+7],invb[Maxn+7];
74     b[0]=1;
75     for (int c=2,M=1;c<(n<<1);c<=<=1)
76     {
77         for (;M<=2*(c-1);M<=<=1);
78         meminit(invb,c,M);
79         getInv(b,invb,c);
80         meminit(tmp,c,M);
81         memcpy(tmp,a,0,c);
82         FFT(invb,M,0);
83         FFT(tmp,M,0);
84         int inv2=(modp+1)/2;

```

```

85         for (int i=0;i<M;i++)
86             tmp[i]=(1LL*(1LL*(1LL*tmp[i]*inv2)%modp)
   ↪ *invb[i])%modp;
87         FFT(tmp,M,1);
88         for (int i=0;i<c;i++)
89             b[i]=(1LL*b[i]*inv2+tmp[i])%modp;
90         meminit(b,c,M);
91     }
92 }
93 int n;
94 int a[Maxn+7],b[Maxn+7];
95 int main()
96 {
97     scanf("%d",&n);
98     for (int i=0;i<n;i++)
99     {
100         scanf("%d",&a[i]);
101         a[i]%=modp;
102     }
103     ++n;
104     getSqrt(a,b,n);
105     for (int i=0;i<n-1;i++)
106         printf("%d ",b[i]);
107     printf("\n");
108     return 0;
109 }

```

### 多项式 $\ln$

给出  $n-1$  次多项式  $A(x)$ , 求一个  $\text{mod } x^n$  下的多项式  $B(x)$ , 满足  $B(x) \equiv \ln A(x)$

时间复杂度  $O(n \log n)$

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define Maxn 524288
4 #define modp 998244353
5 #define g 3
6 int power(int a,int b)
7 {
8     int res=1,now=a,left=b;
9     while (left>0)
10     {
11         if (left%2==1) res=(1LL*res*now)%modp;
12         left/=2;
13         now=(1LL*now*now)%modp;
14     }
15     return res;
16 }
17 #define meminit(A,l,r)
   ↪ memset(A+(l),0,sizeof(*A)*((r)-(l)))
18 #define memcpy(B,A,l,r)
   ↪ memcpy(B,A+(l),sizeof(*A)*((r)-(l)))
19 void FFT(int *a,int n,int f)
20 {
21     for (int i=0,j=0;i<n;i++)
22     {
23         if (i>j) swap(a[i],a[j]);
24         for (int t=n>>1;(j^=t)<t;t>>=1);
25     }
26     for (int i=2;i<=n;i<=<=1)
27     {
28         static int exp[Maxn+7];
29         exp[0]=1;
30         exp[1]=power(g,(modp-1)/i);
31         if (f==1) exp[1]=power(exp[1],modp-2);
32         for (int k=2;k<(i>>1);k++)
33             exp[k]=(1LL*exp[k-1]*exp[1])%modp;
34         for (int j=0;j<n;j+=i)
35             for (int k=0;k<(i>>1);k++)

```

```

36         {
37             int &pA=a[j+k],&pB=a[j+k+(i>>1)];
38             int B=(1LL*pB*exp[k])%modp;
39             pB=(pA-B+modp)%modp;
40             pA=(pA+B)%modp;
41         }
42     }
43     if (f==1)
44     {
45         int rev=power(n,modp-2);
46         for (int i=0;i<n;i++)
47         {
48             a[i]=(1LL*a[i]*rev)%modp;
49             if (a[i]<0) a[i]+=modp;
50         }
51     }
52 }
53 void getInv(int *a,int *b,int n)
54 {
55     static int tmp[Maxn+7];
56     b[0]=power(a[0],modp-2);
57     for (int c=2,M=1;c<(n<<1);c<<=1)
58     {
59         for (;M<=3*(c-1);M<<=1);
60         meminit(b,c,M);
61         meminit(tmp,c,M);
62         memcpy(tmp,a,0,c);
63         FFT(tmp,M,0);
64         FFT(b,M,0);
65         for (int i=0;i<M;i++)
66             b[i]=1LL*b[i]*(
67                 ↪ 2LL-1LL*tmp[i]*b[i]%modp+modp)%modp;
68         FFT(b,M,1);
69         meminit(b,c,M);
70     }
71 void getDiff(int *a,int *b,int n)
72 {
73     for (int i=0;i+1<n;i++)
74         b[i]=(1LL*(i+1)*a[i+1])%modp;
75     b[n-1]=0;
76 }
77 void getInt(int *a,int *b,int n)
78 {
79     static int inv[Maxn+7];
80     inv[1]=1;
81     for (int i=2;i<n;i++)
82         inv[i]=(1LL*(modp-modp/i)*inv[modp%i])%modp;
83     b[0]=0;
84     for (int i=1;i<n;i++)
85         b[i]=(1LL*a[i-1]*inv[i])%modp;
86 }
87 void getLn(int *a,int *b,int n)
88 {
89     static int inv[Maxn+7],d[Maxn+7];
90     int M=1;
91     for (;M<=2*(n-1);M<<=1);
92     getInv(a,inv,n);
93     getDiff(a,d,n);
94     meminit(d,n,M);
95     meminit(inv,n,M);
96     FFT(d,M,0);
97     FFT(inv,M,0);
98     for (int i=0;i<M;i++)
99         d[i]=(1LL*d[i]*inv[i])%modp;
100    FFT(d,M,1);
101    getInt(d,b,n);
102 }
103 int n;

```

```

104 int a[Maxn+7],b[Maxn+7];
105 int main()
106 {
107     scanf("%d",&n);
108     for (int i=0;i<n;i++)
109     {
110         scanf("%d",&a[i]);
111         a[i]%=modp;
112     }
113     ++n;
114     getLn(a,b,n);
115     for (int i=0;i<n-1;i++)
116         printf("%d ",b[i]);
117     printf("\n");
118     return 0;
119 }

```

### 多项式 exp

给出  $n-1$  次多项式  $A(x)$ , 求一个  $\text{mod } x^n$  下的多项式  $B(x)$ , 满足  $B(x) \equiv e^{A(x)}$

时间复杂度  $O(n \log n)$

```

1#include <bits/stdc++.h>
2using namespace std;
3#define Maxn 524288
4#define modp 998244353
5#define g 3
6int power(int a,int b)
7{
8    int res=1,now=a,left=b;
9    while (left>0)
10    {
11        if (left%2==1) res=(1LL*res*now)%modp;
12        left/=2;
13        now=(1LL*now*now)%modp;
14    }
15    return res;
16}
17#define meminit(A,l,r)
18    ↪ memset(A+(l),0,sizeof(*A)*((r)-(l)))
19#define memcpy(B,A,l,r)
20    ↪ memcpy(B,A+(l),sizeof(*A)*((r)-(l)))
21void FFT(int *a,int n,int f)
22{
23    for (int i=0,j=0;i<n;i++)
24    {
25        if (i>j) swap(a[i],a[j]);
26        for (int t=n>>1;(j^=t)<t;t>>=1);
27    }
28    for (int i=2;i<=n;i<<=1)
29    {
30        static int exp[Maxn+7];
31        exp[0]=1;
32        exp[1]=power(g,(modp-1)/i);
33        if (f==1) exp[1]=power(exp[1],modp-2);
34        for (int k=2;k<(i>>1);k++)
35            exp[k]=(1LL*exp[k-1]*exp[1])%modp;
36        for (int j=0;j<n;j+=i)
37        {
38            for (int k=0;k<(i>>1);k++)
39            {
40                int &pA=a[j+k],&pB=a[j+k+(i>>1)];
41                int B=(1LL*pB*exp[k])%modp;
42                pB=(pA-B+modp)%modp;
43                pA=(pA+B)%modp;
44            }
45        }
46    }
47    if (f==1)
48    {
49        int rev=power(n,modp-2);

```

```

46     for (int i=0;i<n;i++)
47     {
48         a[i]=(1LL*a[i]*rev)%modp;
49         if (a[i]<0) a[i]+=modp;
50     }
51 }
52 }
53 void getInv(int *a,int *b,int n)
54 {
55     static int tmp[Maxn+7];
56     b[0]=power(a[0],modp-2);
57     for (int c=2,M=1;c<(n<<1);c<=<=1)
58     {
59         for (;M<=3*(c-1);M<=<=1);
60         meminit(b,c,M);
61         meminit(tmp,c,M);
62         memcpy(tmp,a,0,c);
63         FFT(tmp,M,0);
64         FFT(b,M,0);
65         for (int i=0;i<M;i++)
66             b[i]=1LL*b[i]*(
        ↪ 2LL-1LL*tmp[i]*b[i]%modp+modp)%modp;
67         FFT(b,M,1);
68         meminit(b,c,M);
69     }
70 }
71 void getDiff(int *a,int *b,int n)
72 {
73     for (int i=0;i+1<n;i++)
74         b[i]=(1LL*(i+1)*a[i+1])%modp;
75     b[n-1]=0;
76 }
77 void getInt(int *a,int *b,int n)
78 {
79     static int inv[Maxn+7];
80     inv[1]=1;
81     for (int i=2;i<n;i++)
82         inv[i]=(1LL*(modp-modp/i)*inv[modp%i])%modp;
83     b[0]=0;
84     for (int i=1;i<n;i++)
85         b[i]=(1LL*a[i-1]*inv[i])%modp;
86 }
87 void getLn(int *a,int *b,int n)
88 {
89     static int inv[Maxn+7],d[Maxn+7];
90     int M=1;
91     for (;M<=2*(n-1);M<=<=1);
92     getInv(a,inv,n);
93     getDiff(a,d,n);
94     meminit(d,n,M);
95     meminit(inv,n,M);
96     FFT(d,M,0);
97     FFT(inv,M,0);
98     for (int i=0;i<M;i++)
99         d[i]=(1LL*d[i]*inv[i])%modp;
100     FFT(d,M,1);
101     getInt(d,b,n);
102 }
103 void getExp(int *a,int *b,int n)
104 {
105     static int ln[Maxn+7],tmp[Maxn+7];
106     b[0]=1;
107     for (int c=2,M=1;c<(n<<1);c<=<=1)
108     {
109         for (;M<=2*(c-1);M<=<=1);
110         int bound=min(c,n);
111         memcpy(tmp,a,0,bound);
112         meminit(tmp,bound,M);
113         meminit(b,c,M);

```

```

114         getLn(b,ln,c);
115         meminit(ln,c,M);
116         FFT(b,M,0);
117         FFT(tmp,M,0);
118         FFT(ln,M,0);
119         for (int i=0;i<M;i++)
120             b[i]=(1LL*b[i]*(1LL-ln[i]+tmp[i]+modp))
        ↪ %modp;
121         FFT(b,M,1);
122         meminit(b,c,M);
123     }
124 }
125 int n;
126 int a[Maxn+7],b[Maxn+7];
127 int main()
128 {
129     scanf("%d",&n);
130     for (int i=0;i<n;i++)
131     {
132         scanf("%d",&a[i]);
133         a[i]%modp;
134     }
135     ++n;
136     getExp(a,b,n);
137     for (int i=0;i<n-1;i++)
138         printf("%d ",b[i]);
139     printf("\n");
140     return 0;
141 }

```

### 多项式全家桶

```

1 #include <bits/stdc++.h>
2
3 using namespace std;
4 int _w;
5
6 namespace Poly {
7     typedef long long ll;
8
9     const int MOD = 998244353;
10    const int G = 3;
11
12    int fpow( int a, int b ) {
13        int c = 1;
14        while(b) {
15            if( b & 1 ) c = int(1LL * c * a % MOD);
16            a = int(1LL * a * a % MOD);
17            b >>= 1;
18        }
19        return c;
20    }
21
22    int inv( int x ) {
23        return fpow(x, MOD-2);
24    }
25
26    int add_mod( int a, int b ) {
27        a += b;
28        a -= a >= MOD ? MOD : 0;
29        return a;
30    }
31
32    int sub_mod( int a, int b ) {
33        a -= b;
34        a += a < 0 ? MOD : 0;
35        return a;
36    }
37
38    int up( int n ) {

```

```

39     int len = 1;
40     while( len < n ) len <= 1;
41     return len;
42 }
43
44 int sqrt_mod( int n ) {
45     const int p = MOD;
46
47     if( n == 0 ) return 0;
48     if( p == 2 ) return (n & 1) ? 1 : -1;
49     if( fpow(n, p>>1) != 1 )
50         return -1;
51     if( p & 2 ) return fpow(n, (p+1)>>2);
52     int s = __builtin_ctzll(p-1);
53     int q = p>>s, z = 2;
54     for( ; fpow(z, p>>1) == 1; ++z );
55     int c = fpow(z, q);
56     int r = fpow(n, (q+1)>>1);
57     int t = fpow(n, q), tmp;
58     for( int m = s, i; t != 1; ) {
59         for( i = 0, tmp = t; tmp != 1; ++i )
60             tmp = int(1LL * tmp * tmp % p);
61         for( ; i < --m; )
62             c = int(1LL * c * c % p);
63         r = int(1LL * r * c % p);
64         c = int(1LL * c * c % p);
65         t = int(1LL * t * c % p);
66     }
67     return r;
68 }
69
70 namespace Eva {
71     // Evaluation
72
73     int eva( const vector<int> &f, int x ) {
74         int y = 0, sz = (int)f.size();
75         for( int i = sz-1; i >= 0; --i )
76             y = int((1LL * y * x + f[i]) % MOD);
77         return y;
78     }
79 }
80 using Eva::eva;
81
82 namespace Add {
83     vector<int> add( const vector<int> &f, const
84         ↪ vector<int> &g ) {
85         int sz = max( (int)f.size(), (int)g.size()
86             ↪ );
87         vector<int> h(sz);
88         for( int i = 0; i < sz; ++i ) {
89             int fi = i < (int)f.size() ? f[i] : 0;
90             int gi = i < (int)g.size() ? g[i] : 0;
91             h[i] = add_mod(fi, gi);
92         }
93         return h;
94     }
95 }
96 using Add::add;
97
98 namespace Sub {
99     vector<int> sub( const vector<int> &f, const
100         ↪ vector<int> &g ) {
101         int sz = max( (int)f.size(), (int)g.size()
102             ↪ );
103         vector<int> h(sz);
104         for( int i = 0; i < sz; ++i ) {
105             int fi = i < (int)f.size() ? f[i] : 0;
106             int gi = i < (int)g.size() ? g[i] : 0;
107             h[i] = sub_mod(fi, gi);
108         }
109         return h;
110     }
111 }
112
113 namespace NTT {
114     int rb( int x, int n ) {
115         int ans = 0;
116         n >>= 1;
117         while(n) {
118             ans = (ans << 1) | (x & 1);
119             n >>= 1, x >>= 1;
120         }
121         return ans;
122     }
123
124     vector<int> ntt( const vector<int> &a, int n,
125         ↪ int rev ) {
126         vector<int> y(n);
127         for( int i = 0; i < n; ++i ) {
128             int ai = i < (int)a.size() ? a[i] : 0;
129             y[rb(i, n)] = ai;
130         }
131         for( int m = 1; m <= n; m <= 1 ) {
132             int wm = fpow(G, (MOD-1)/m);
133             if( rev ) wm = inv(wm);
134             for( int i = 0; i < n; i += m ) {
135                 int w = 1;
136                 for( int j = 0; j < (m>>1); ++j ) {
137                     int t1 = y[i+j];
138                     int t2 = int(1LL *
139                         ↪ y[i+j+(m>>1)] * w % MOD);
140                     y[i+j] = add_mod(t1, t2);
141                     y[i+j+(m>>1)] = sub_mod(t1,
142                         ↪ t2);
143                     w = int(1LL * w * wm % MOD);
144                 }
145             }
146             if( rev ) {
147                 int invn = inv(n);
148                 for( int i = 0; i < n; ++i )
149                     y[i] = int(1LL * y[i] * invn %
150                         ↪ MOD);
151             }
152             return y;
153         }
154     }
155 }
156 using NTT::ntt;
157
158 namespace Mul {
159     const unsigned LIM = 10000;
160     const unsigned LEN = 20;
161
162     vector<int> mul( vector<int> a, vector<int> b )
163         ↪ {
164         if( a.size() * b.size() <= LIM || a.size()
165             ↪ <= LEN || b.size() <= LEN ) {
166             int n = (int)a.size();
167             int m = (int)b.size();
168             int sz = n+m-1;
169             vector<int> c(sz, 0);
170             for( int i = 0; i < n; ++i )
171                 for( int j = 0; j < m; ++j )
172                     c[i+j] = add_mod( c[i+j],
173                         ↪ int(1LL * a[i] * b[j] %
174                         ↪ MOD) );
175         }
176     }
177 }

```

```

164         return c;
165     } else {
166         int sz = (int)a.size() + (int)b.size()
167         ↪ - 1;
168         int len = up(sz);
169         a = ntt(a, len, 0);
170         b = ntt(b, len, 0);
171         for( int i = 0; i < len; ++i )
172             a[i] = int(1LL * a[i] * b[i] %
173             ↪ MOD);
174         a = ntt(a, len, 1);
175         a.resize(sz);
176         return a;
177     }
178 }
179 using Mul::mul;
180 namespace Inv {
181     vector<int> inv( vector<int> a, int n ) {
182         a.resize(n, 0);
183         if( n == 1 ) {
184             a[0] = Poly::inv( a[0] );
185             return a;
186         } else {
187             vector<int> b = inv(a, n/2);
188             b = ntt(b, n*2, 0);
189             a = ntt(a, n*2, 0);
190             for( int i = 0; i < n*2; ++i ) {
191                 int t1 = add_mod( b[i], b[i] );
192                 int t2 = int(1LL * a[i] * b[i] %
193                 ↪ MOD * b[i] % MOD);
194                 b[i] = sub_mod(t1, t2);
195             }
196             b = ntt(b, n*2, 1);
197             b.resize(n);
198             return b;
199         }
200     }
201 }
202 using Inv::inv;
203 namespace Sqrt {
204     vector<int> sqrt( vector<int> a, int n ) {
205         a.resize(n, 0);
206         if( n == 1 ) {
207             a[0] = sqrt_mod( a[0] );
208             return a;
209         } else {
210             vector<int> b = sqrt(a, n/2);
211             vector<int> invb = inv(b, n);
212             b = ntt(b, n*2, 0);
213             a = ntt(a, n*2, 0);
214             invb = ntt(invb, n*2, 0);
215             for( int i = 0; i < n*2; ++i ) {
216                 int t = add_mod( int(1LL * b[i] *
217                 ↪ b[i] % MOD), a[i] );
218                 b[i] = int(1LL * t * invb[i] %
219                 ↪ MOD);
220             }
221             b = ntt(b, n*2, 1);
222             b.resize(n);
223             int inv2 = (MOD + 1) / 2;
224             for( int i = 0; i < n; ++i )
225                 b[i] = int(1LL * b[i] * inv2 %
226                 ↪ MOD);
227             return b;
228         }
229     }
230 }

```

```

227 }
228 using Sqrt::sqrt;
229 namespace Div {
230     vector<int> div( vector<int> a, vector<int> b )
231     ↪ {
232         int n = (int)a.size();
233         int m = (int)b.size();
234         assert( n >= m );
235         int sz = n-m+1;
236         reverse(a.begin(), a.end());
237         reverse(b.begin(), b.end());
238         a.resize(sz, 0);
239         b.resize(sz, 0);
240         b = inv(b, up(sz));
241         b.resize(sz, 0);
242         a = mul(a, b);
243         a.resize(sz, 0);
244         reverse(a.begin(), a.end());
245         return a;
246     }
247 }
248 using Div::div;
249 namespace Mod {
250     const unsigned LIM = 500000;
251     const int LEN = 100;
252     vector<int> mod( vector<int> a, const
253     ↪ vector<int> &b ) {
254         assert( b.size() - 1 );
255         if( b.size() > a.size() ) {
256             return a;
257         } else {
258             if( a.size() * b.size() <= LIM ||
259             ↪ b.size() <= LEN ) {
260                 int n = (int)a.size();
261                 int m = (int)b.size();
262                 int iv = inv( b[m-1] );
263                 for( int i = n-1; i >= m-1; --i ) {
264                     int k = int(1LL * a[i] * iv %
265                     ↪ MOD);
266                     for( int j = 0; j < m; ++j )
267                         a[i-j] = sub_mod( a[i-j],
268                         ↪ int(1LL * b[m-j-1] * k
269                         ↪ % MOD) );
270                 }
271             } else {
272                 a = sub( a, mul( b, div(a, b) ) );
273             }
274             int sz = (int)a.size() - 1;
275             while( sz && a[sz] == 0 ) --sz;
276             a.resize(sz + 1);
277             assert( a.size() < b.size() );
278             return a;
279         }
280     }
281 }
282 using Mod::mod;
283 namespace Integral {
284     vector<int> rev;
285     void calc_rev( int sz ) {
286         rev.resize(sz+1);
287         rev[0] = 1;
288         for( int i = 1; i <= sz; ++i )
289             rev[i] = int(1LL * rev[i-1] * i % MOD);
290     }
291 }

```

```

289         rev[sz] = inv( rev[sz] );
290         for( int i = sz; i >= 1; --i ) {
291             int now = int(1LL * rev[i] * rev[i-1] %
292                 ↪ MOD);
293             rev[i-1] = int(1LL * rev[i] * i % MOD);
294             rev[i] = now;
295         }
296     }
297     vector<int> integral( vector<int> a ) {
298         int sz = (int)a.size();
299         calc_rev(sz);
300         a.resize(sz+1);
301         for( int i = sz; i >= 1; --i )
302             a[i] = a[i-1];
303         a[0] = 0;
304         for( int i = sz; i >= 1; --i )
305             a[i] = int(1LL * a[i] * rev[i] % MOD);
306         return a;
307     }
308     using Integral::integral;
309
310     namespace Derive {
311         vector<int> derive( vector<int> a ) {
312             int sz = (int)a.size();
313             for( int i = 1; i < sz; ++i )
314                 a[i-1] = int(1LL * a[i] * i % MOD);
315             a.resize(sz-1);
316             return a;
317         }
318     }
319     using Derive::derive;
320
321     namespace Ln {
322         vector<int> ln( vector<int> a, int n ) {
323             vector<int> b = inv(a, up(n));
324             b.resize(n);
325             a = integral( mul( derive(a), b ) );
326             a.resize(n);
327             return a;
328         }
329     }
330     using Ln::ln;
331
332     namespace Exp {
333         vector<int> exp( vector<int> a, int n ) {
334             a.resize(n, 0);
335             if( n == 1 ) {
336                 assert( a[0] == 0 );
337                 a[0] = 1;
338                 return a;
339             } else {
340                 vector<int> b = exp(a, n/2);
341                 vector<int> tmp(1, 1);
342                 b = mul( b, sub( add(tmp, a), ln(b, n)
343                     ↪ ) );
344                 b.resize(n);
345                 return b;
346             }
347         }
348     }
349     using Exp::exp;
350
351     namespace MPE {
352         // Multipoint Evaluation
353
354         const int N = 100010;
355
356         int n, num[N];
357
358         vector<int> p[N*4];
359
360         void init( int o, int L, int R ) {
361             if( L == R ) {
362                 p[o].resize(2);
363                 p[o][0] = sub_mod(0, num[L]);
364                 p[o][1] = 1;
365             } else {
366                 int M = (L+R)/2, lc = o<<1, rc = lc|1;
367                 init(lc, L, M);
368                 init(rc, M+1, R);
369                 p[o] = mul( p[lc], p[rc] );
370             }
371         }
372
373         void solve( int o, int L, int R ) {
374             if( L == R ) {
375                 num[L] = eva( p[o], num[L] );
376             } else {
377                 int M = (L+R)/2, lc = o<<1, rc = lc|1;
378                 p[lc] = mod( p[o], p[lc] );
379                 p[rc] = mod( p[o], p[rc] );
380                 solve(lc, L, M);
381                 solve(rc, M+1, R);
382             }
383         }
384
385         vector<int> mpe( const vector<int> &f,
386             ↪ vector<int> a ) {
387             n = (int)a.size();
388             for( int i = 0; i < n; ++i )
389                 num[i] = a[i];
390             init(1, 0, n-1);
391             p[1] = mod(f, p[1]);
392             solve(1, 0, n-1);
393             for( int i = 0; i < n; ++i )
394                 a[i] = num[i];
395             return a;
396         }
397     }
398     using MPE::mpe;
399
400     namespace IPL {
401         // Interpolation
402
403         const int N = 100010;
404
405         vector<int> tmp[N*4];
406
407         void init( const vector<int> &x, int o, int L,
408             ↪ int R ) {
409             if( L == R ) {
410                 vector<int> &ans = tmp[o];
411                 ans.resize(2);
412                 ans[0] = sub_mod(0, x[L]);
413                 ans[1] = 1;
414                 MPE::num[L] = x[L];
415                 MPE::p[o] = tmp[o];
416             } else {
417                 int M = (L+R)/2, lc = o<<1, rc = lc|1;
418                 init(x, lc, L, M);
419                 init(x, rc, M+1, R);
420                 MPE::p[o] = tmp[o] = mul( tmp[lc],
421                     ↪ tmp[rc] );
422             }
423         }
424
425         vector<int> solve( int o, int L, int R ) {

```

```

421     if( L == R ) {
422         return vector<int>(1, MPE::num[L]);
423     } else {
424         int M = (L+R)/2, lc = o<<1, rc = lc|1;
425         return add( mul( solve(lc, L, M),
426             ↪ tmp[rc] ),
427             mul( solve(rc, M+1, R),
428                 ↪ tmp[lc] ) );
429     }
430 }
431
432 void mpe2( const vector<int> &f, int n ) {
433     MPE::p[1] = f;
434     MPE::solve(1, 0, n-1);
435 }
436
437 vector<int> ipl( const vector<int> &x, const
438     ↪ vector<int> &y ) {
439     assert( x.size() == y.size() );
440     int n = (int)x.size();
441     init(x, 1, 0, n-1);
442     mpe2( derive( tmp[1] ), n );
443     for( int i = 0; i < n; ++i )
444         MPE::num[i] = int(1LL * y[i] * inv(
445             ↪ MPE::num[i] ) % MOD);
446     return solve(1, 0, n-1);
447 }
448 }
449 using IPL::ipl;
450 }
451
452 int main() {
453     int n, k;
454     _w = scanf( "%d%d", &n, &k );
455     ++n;
456
457     vector<int> f(k+1), a(k+1);
458     for( int i = 1; i <= k; ++i ) {
459         _w = scanf( "%d", &f[i] );
460         f[i] = (f[i] % Poly::MOD + Poly::MOD) %
461             ↪ Poly::MOD;
462     }
463     for( int i = 1; i <= k; ++i ) {
464         _w = scanf( "%d", &a[i] );
465         a[i] = (a[i] % Poly::MOD + Poly::MOD) %
466             ↪ Poly::MOD;
467     }
468     f[0] = a[0] = 0;
469     reverse(f.begin() + 1, f.end());
470     for( int i = 1; i <= k; ++i )
471         f[i] = (Poly::MOD - f[i]) % Poly::MOD;
472     f.push_back(1);
473
474     vector<int> ans(1, 1), tmp(2);
475     tmp[0] = 0, tmp[1] = 1;
476     while(n) {
477         if( n & 1 ) ans = Poly::mod( Poly::mul(ans,
478             ↪ tmp), f );
479         tmp = Poly::mod( Poly::mul(tmp, tmp), f );
480         n >>= 1;
481     }
482
483     int y = 0;
484     ans.resize(k+1);
485     for( int i = 1; i <= k; ++i )
486         y = (y + int(1LL * ans[i] * a[i] % Poly::MOD))
487             ↪ % Poly::MOD;
488     printf( "%d\n", y );
489     return 0;

```

## BM

给定一个有  $n$  个元素的数列  $a$ ，其中第  $i$  个元素是  $a_i$ 。  
 求一个较短/最短的数列  $b$ ，假设  $b$  有  $m$  个元素，那么要求满足  
 $m < i \leq n, a_i = \sum_{j=1}^m a_{i-j} b_j$   
 时间复杂度  $O(n^2)$

```

1#include <bits/stdc++.h>
2using namespace std;
3#define Maxn 100007
4#define modp 1000000007
5int n,cnt;
6int a[Maxn],fail[Maxn],delta[Maxn];
7vector<int> R[Maxn];
8int power(int a,int b)
9{
10     int res=1,now=a,left=b;
11     while (left>0)
12     {
13         if (left%2==1) res=(1LL*res*now)%modp;
14         left/=2;
15         now=(1LL*now*now)%modp;
16     }
17     return res;
18}
19int main()
20{
21     scanf("%d",&n);
22     for (int i=1;i<=n;i++)
23         scanf("%d",&a[i]);
24     R[0].clear();
25     cnt=0;
26     for (int i=1;i<=n;i++)
27     {
28         if (cnt==0)
29         {
30             if (a[i])
31             {
32                 fail[cnt++]=i;
33                 delta[i]=a[i];
34                 R[cnt].resize(0);
35                 R[cnt].resize(i,0);
36             }
37             continue;
38         }
39         int sum=0,m=(int)R[cnt].size();
40         delta[i]=a[i];
41         fail[cnt]=i;
42         for (int j=0;j<m;j++)
43             sum=(sum+1LL*a[i-j-1]*R[cnt][j])%modp;
44         delta[i]=(delta[i]+modp-sum)%modp;
45         if (!delta[i]) continue;
46         int id=cnt-1,v=i-fail[id]+(int)R[id].size();
47         for (int j=0;j<cnt;j++)
48             if (i-fail[j]+(int)R[j].size()<v)
49             {
50                 id=j;
51                 v=i-fail[j]+(int)R[j].size();
52             }
53         int tmp=(1LL*delta[i]*power(
54             ↪ delta[fail[id]],modp-2))%modp;
55         R[cnt+1]=R[cnt];
56         while ((int)R[cnt+1].size()<v)
57             ↪ R[cnt+1].push_back(0);
58         R[cnt+1][i-fail[id]-1]=
59             ↪ R[cnt+1][i-fail[id]-1]+tmp)%modp;
60         for (int j=0;j<(int)R[id].size();j++)

```

```

58         R[cnt+1][i-fail[id]+j]=(
            ↪ R[cnt+1][i-fail[id]+j]+modp-(
            ↪ 1LL*tmp*R[id][j])%modp)%modp;
59     cnt++;
60 }
61 printf("%d\n", (int)R[cnt].size());
62 for (int i=0; i<(int)R[cnt].size(); i++)
63     printf("%d ", R[cnt][i]);
64 printf("\n");
65 return 0;
66 }

```

## 1.9. pell 方程

对于方程  $x^2 - dy^2 = 1$ , 其中  $d$  是一个非完全平方数的正整数  
设最小解为  $(x_0, y_0)$

则有:

$$x_n = x_0 x_{n-1} + dy_0 y_{n-1}$$

$$y_n = y_0 x_{n-1} + x_0 y_{n-1}$$

```

1 //solve(d,N) 输出最大的 (x,y) 使得 x^2-dy^2=1 且 x<=N
2 #include <bits/stdc++.h>
3 using namespace std;
4 void solve(int d, long long N)
5 {
6     long long x0=0LL, y0=1LL;
7     for (; y0++)
8     {
9         x0=(long long)sqrt(1LL*d*y0*y0+1);
10        if (1LL*x0*x0-1LL*d*y0*y0==1LL) break;
11    }
12    long long x=x0, y=y0;
13    long long resx=-1LL, resy=-1LL;
14    while (x<=N)
15    {
16        if (x%4==3) resx=x/4, resy=y;
17        long long nowx=1LL*x0*x+1LL*d*y0*y;
18        long long nowy=1LL*y0*x+1LL*x0*y;
19        x=nowx;
20        y=nowy;
21    }
22    printf("%lld %lld\n", resx, resy);
23 }
24 long long n;
25 int main()
26 {
27     while (scanf("%lld", &n) != EOF)
28     {
29         if (n==0LL) break;
30         solve(48, 4LL*n+3);
31     }
32     return 0;
33 }

```

## 1.10. 一元三次方程求根公式

```

1 //solve() Ax^3+Bx^2+Cx+D=0 (A!=0)
2 #include <bits/stdc++.h>
3 using namespace std;
4 double a,b,c,d;
5 void solve(double A, double B, double C, double D)
6 {
7     complex<double> i(0,1.0);
8     complex<double> a=A, b=B, c=C, d=D;
9     complex<double> p=(3.0*a*c-b*b)/(3.0*a*a);
10    complex<double> q=(
        ↪ 27.0*a*a*d-9.0*a*b*c+2.0*b*b*b)/(27.0*a*a*a);
11    complex<double> delta=pow(p*p*p/27.0+q*q/4.0, 0.5);
12    complex<double> tmp1=pow(-q/2.0+delta, 1/3.0);
        ↪ , tmp2=pow(-q/2.0-delta, 1/3.0);
13    complex<double> w=-1/2.0+(pow(3.0, 1/2.0)/2.0)*i;
14    complex<double> x1=tmp1+tmp2;

```

```

15    complex<double> x2=w*tmp1+w*w*tmp2;
16    complex<double> x3=w*w*tmp1+w*tmp2;
17    x1-=b/(3.0*a);
18    x2-=b/(3.0*a);
19    x3-=b/(3.0*a);
20    if (x1.real()>x2.real()) swap(x1,x2);
21    if (x2.real()>x3.real()) swap(x2,x3);
22    if (x1.real()>x2.real()) swap(x1,x2);
23    printf("%.2lf %.2lf
        ↪ %.2lf\n", x1.real(), x2.real(), x3.real());
24 }
25 int main()
26 {
27     scanf("%lf%lf%lf%lf", &a, &b, &c, &d);
28     solve(a,b,c,d);
29     return 0;
30 }

```

## 1.11. simpson 积分

```

1 #include <bits/stdc++.h>
2 #define eps 1e-10
3 using namespace std;
4 int T;
5 double a,b,L,R;
6 double f(double x)
7 {
8     return 2.0*sqrt(1.0*b*b-1.0*b*b*x*x/(1.0*a*a));
9 }
10 double calc(double L, double R)
11 {
12     return (R-L)*(f(L)+f(R)+4.0*f(0.5*(L+R)))/6.0;
13 }
14 double simpson(double L, double R)
15 {
16     double tmp=calc(L,R);
17     double mid=0.5*(L+R);
18     double tmp1=calc(L,mid), tmp2=calc(mid,R);
19     if (fabs(tmp-tmp1-tmp2)<15.0*eps) return tmp1+tmp2;
20     return simpson(L,mid)+simpson(mid,R);
21 }
22 void solve()
23 {
24     scanf("%lf%lf%lf%lf", &a, &b, &L, &R);
25     printf("%.3lf\n", simpson(L,R));
26 }
27 int main()
28 {
29     scanf("%d", &T);
30     while (T--) solve();
31     return 0;
32 }

```

## 1.12. 整数划分

把  $n$  拆成几个小于等于  $n$  的正整数相加的形式, 求方案数  
时间复杂度  $O(n\sqrt{n})$

```

1 #define modp 1000000007
2 int ans[100007];
3 void get_factorization(int n)
4 {
5     memset(ans, 0, sizeof(ans));
6     ans[0]=1;
7     for (int i=1; i<=n; i++)
8         for (int j=1, r=1; i>=(3*j*j-j)/2; j++, r*=-1)
9         {
10             ans[i]+=ans[i-(3*j*j-j)/2]*r;
11             ans[i]=(ans[i]%modp+modp)%modp;
12             if (i>=(3*j*j+j)/2)
13             {

```



```

14         ans[i] += ans[i - (3 * j * j + j) / 2] * r;
15         ans[i] = (ans[i] % modp + modp) % modp;
16     }
17 }
18 }

```

把  $n$  拆成几个小于等于  $n$  的正整数相加的形式，相同的数不能超过  $k$  个，求方案数

### 1.13. min-max 容斥

$\max(S)$  表示  $S$  集合最大值的期望， $\min(S)$  表示  $S$  集合最小值的期望

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

$$\max(S, k) = \sum_{T \subseteq S} (-1)^{|T|-k} C(|T| - 1, k - 1) \min(T)$$

$T$  可以取  $S$ ，不可取空集

其中  $\max(S, k)$  表示  $S$  集合中第  $k$  大的元素

### 1.14. 线性基

```

1 //cnt!=n 说明可以构成 0
2 //insert() 线性基插入一个数
3 //process() 处理线性基
4 //query(x) 询问可以构成的第 x 小
5 #include <bits/stdc++.h>
6 using namespace std;
7 long long s[67];
8 int n, q, cnt;
9 void insert(long long x)
10 {
11     for (int i = 60; i >= 0; i--)
12         if (x & (1LL << i))
13         {
14             if (s[i] ^ x ^ s[i];
15             else
16             {
17                 s[i] = x;
18                 break;
19             }
20         }
21 }
22 void process()
23 {
24     for (int i = 60; i >= 0; i--)
25         for (int j = i - 1; j >= 0; j--)
26             if (s[i] & (1LL << j)) s[i] ^= s[j];
27 }
28 long long query(long long x)
29 {
30     if (cnt != n) --x;
31     if (x == 0LL) return 0LL;
32     long long res = 0LL;
33     for (int i = 0; i <= 60; i++)
34         if (s[i])
35         {
36             if (x % 2 == 1) res ^= s[i];
37             x /= 2;
38         }
39     if (x) return -1LL;
40     return res;
41 }
42 void solve()
43 {
44     scanf("%d", &n);
45     memset(s, 0, sizeof(s));
46     for (int i = 1; i <= n; i++)
47     {
48         long long x;
49         scanf("%lld", &x);
50         insert(x);
51     }

```

```

52     process();
53     cnt = 0;
54     for (int i = 0; i <= 60; i++)
55         if (s[i] != 0) ++cnt;
56     scanf("%d", &q);
57     for (int i = 1; i <= q; i++)
58     {
59         long long x;
60         scanf("%lld", &x);
61         printf("%lld\n", query(x));
62     }
63 }
64 int main()
65 {
66     int T;
67     scanf("%d", &T);
68     for (int t = 1; t <= T; t++)
69     {
70         printf("Case #%d:\n", t);
71         solve();
72     }
73     return 0;
74 }

```

### 1.15. Stern-Brocot 树

$$M(L) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

$$M(R) = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$$

$$f(M(S)) = \begin{bmatrix} n & n' \\ m & m' \end{bmatrix} = \frac{m+m'}{n+n'}$$

### 1.16. 数相关

#### Stirling 数

第一类 Stirling 数：将  $p$  个不同的物品排成  $k$  个非空循环排列的方法数

$$S(p, 0) = 0, p \geq 1$$

$$S(p, p) = 1, p \geq 0$$

$$S(p, k) = (p-1)S(p-1, k) + S(p-1, k-1), p-1 \geq k \geq 1$$

第二类 Stirling 数：将  $p$  个不同的物品分成  $k$  个不可辨别的非空集合的方法数

$$S(p, 0) = 0, p \geq 1$$

$$S(p, p) = 1, p \geq 0$$

$$S(p, k) = kS(p-1, k) + S(p-1, k-1), p-1 \geq k \geq 1$$

$$S(m, n) = \frac{1}{m!} \sum_{k=0}^m (-1)^k C(m, k) (m-k)^n = \sum_{k=0}^m \frac{(-1)^k}{k!} * \frac{(m-k)^n}{(m-k)!}$$

|  $k$  个球 |  $m$  个盒子 | 是否允许有空盒子 | 方案数 |

| 各不相同 | 各不相同 | 是 |  $m^k$  |

| 各不相同 | 各不相同 | 否 |  $m! Stirling2(k, m)$  |

| 各不相同 | 完全相同 | 是 |  $\sum_{i=1}^m Stirling2(k, i)$  |

| 各不相同 | 完全相同 | 否 |  $Stirling2(k, m)$  |

| 完全相同 | 各不相同 | 是 |  $C(m+k-1, k)$  |

| 完全相同 | 各不相同 | 否 |  $C(k-1, m-1)$  |

| 完全相同 | 完全相同 | 是 |  $\frac{1}{(1-x)(1-x^2)\dots(1-x^m)}$  的  $x^k$  系数 |

| 完全相同 | 完全相同 | 否 |  $\frac{x^m}{(1-x)(1-x^2)\dots(1-x^m)}$  的  $x^k$  系数 |

#### Catalan 数

$$h_1 = 1$$

$$h_n = \sum_{k=0}^{n-1} h_k * h_{n-1-k}$$

$$h_n = h_{n-1} * \frac{4n-2}{n+1} = \frac{C(2n, n)}{n+1} = C(2n, n) - C(2n, n-1)$$

在一个格点阵列中，从  $(0, 0)$  点走到  $(n, m)$  点且不经过对角线  $x = y$  的方法数  $(x > y) = C(n+m-1, m) - C(n+m-1, m-1)$

在一个格点阵列中，从  $(0, 0)$  点走到  $(n, m)$  点且不穿过对角线  $x = y$  的方法数  $(x \geq y) = C(n+m, m) - C(n+m, m-1)$

#### Fibonacci 数

设  $F_n$  在  $\text{mod } x$  意义下的最短循环节是  $G(x)$

对于素数  $p$ ， $G(p)$  要么是  $p-1$  的因子，要么是  $2(p+1)$  的因子

对于素数  $p$ ， $G(p^k) = G(p) * p^{k-1}$

**Bell 数**

贝尔数表示把  $n$  个带标号的物品划分为若干个不相交的集合的方案数

$$B_0 = 0$$

$$B_{n+1} = \sum_{k=0}^n C(n, k) B_k$$

$$B_n = \sum_{k=1}^n S(n, k), \text{ 其中 } S(n, k) \text{ 表示第二类 Stirling 数}$$

$$B_n = \sum_{m=0}^n S(n, m) = \sum_{m=0}^n \frac{1}{m!} \sum_{k=0}^m (-1)^k C(m, k) (m - k)^n$$

通过维护前缀和可以在  $O(n \log n)$  复杂度内求出单个贝尔数

对于素数  $p$ , 有  $B_{p^m+n} \equiv mB_n + B_{n+1} \pmod{p}$

**伯努利数**

$$B_0 = 1$$

$$\sum_{k=0}^n C(n+1, k) B_k = n+1$$

$$\sum_{i=1}^n i^m = \frac{1}{m+1} \sum_{k=0}^m C(m+1, k) B_k n^{m+1-k}$$

其中  $m=0$  时不成立

(注意:  $B_1 = \frac{1}{2}$ , 和常规意义下的伯努利数不同)

**1.17. 定理相关****Cayley 公式**

设  $T_{n,k}$  为  $n$  个有标号点分成  $k$  棵树的森林数, 其中节点  $1, 2, \dots, k$  属于不同的树, 则

$$T_{n,k} = kn^{n-k-1}$$

**Jacobi's Four Square Theorem**

设  $a^2 + b^2 + c^2 + d^2 = n$  的自然数解个数为  $r_4(n)$ ,  $d(n)$  为  $n$  的约数和

若  $n$  是奇数, 则  $r_4(n) = 8d(n)$ , 否则  $r_4(n) = 24d(n/2)$

**Matrix-Tree 定理**

$G$  的度数矩阵  $D[G]$  是一个  $n \times n$  的矩阵当  $i \neq j$  时,  $d[i][j] = 0$ ; 当  $i = j$  时,  $d[i][j]$  等于  $i$  的度数

$G$  的邻接矩阵  $A[G]$

我们定义  $G$  的 *Kirchhoff* 矩阵 (也称为拉普拉斯算子)  $C[G] = D[G] - A[G]$

则 *Matrix-Tree* 定理可以描述为:

$G$  的所有不同的生成树的个数等于其 *Kirchhoff* 矩阵  $C[G]$  任何一个  $n-1$  阶主子式的行列式的绝对值

无向图生成树计数:  $d[i][i]$  等于  $i$  的度数

有向图外向生成树计数:  $d[i][i]$  等于  $i$  的入度数

有向图内向生成树计数:  $d[i][i]$  等于  $i$  的出度数

**平面图欧拉公式**

对于连通的平面图, 封闭区域  $F =$  边数  $V$  - 点数  $E + 1$

**蔡勒公式**

$$w = ((y + [\frac{y}{4}] + [\frac{c}{4}]) - 2c + [\frac{26(m+1)}{10}]) + d - 1 \pmod{7+7} \pmod{7}$$

如果日期在 1582 年 10 月 4 日或以前, 则

$$w = ((y + [\frac{y}{4}] - c + [\frac{26(m+1)}{10}]) + d + 4) \pmod{7+7} \pmod{7}$$

一二月要看成前一年的 13、14 月

**皮克定理**

给定顶点坐标均是整数 (或正方形格点) 的简单多边形,

$$S (\text{面积}) = n (\text{内部格点数目}) + \frac{1}{2}s (\text{边上格点数目}) - 1$$

**组合数 LCM**

$$(n+1)lcm(C(n,0), C(n,1), \dots, C(n,k)) = lcm(n+1, n, n-1, n-2, \dots, n-k+1)$$

**错排公式**

$$D_1 = 0, D_2 = 1, D_n = (n-1)(D_{n-2} + D_{n-1})$$

**Prufer Sequence**

Prufer sequence 是一个描述一棵树的编号序列。

对于一棵树: 我们每次选择编号最小的叶子节点, 并将它的父亲编号加入序列, 然后删除改点, 直到整棵树只剩下 2 个顶点。

对于一个 prufer 序列: 我们建立初始为  $1 \sim n$  的集合  $S$ , 每一次找出在  $S$  中出现的且不在 prufer 序列中出现的最小值, 将它和序列中的第一项相连, 然后删除序列中的第一项和  $S$  中的这个数, 直到整个序列被删除。

树的形态一共有  $n^{n-2}$  个, 而 prufer 序列也一共有  $n^{n-2}$  个, 它们形成一一对应关系。

**二项式反演**

$$f(n) = \sum_{k=0}^n C(n, k) g(k)$$

$$g(n) = \sum_{k=0}^n (-1)^{n-k} C(n, k) f(k)$$

 **$x^k$  转化**

$$x^k = \sum_{i=1}^k \text{Stirling2}(k, i) * i! * C(x, i)$$

**BEST Theorem**

从  $i$  出发并且回到  $i$  的欧拉回路:

有向图: 以  $i$  为根的外向树 \* (每个点的度数-1)!

无向图: 生成树 \* (每个点的度数-1)!

**拉格朗日四平方和定理**

每个正整数均可表示为 4 个整数的平方和

**第  $k$  小期望**

$f_n(k)$  表示有  $n$  个变量, 和为 1, 第  $k$  小的期望

$$f_n(1) = \frac{1}{n^2}$$

$$f_n(k) = \frac{1}{n^2} + (1 - \frac{1}{n})f_{n-1}(k-1)$$

**Stirling 公式**

$$n! \approx \sqrt{2\pi n} (\frac{n}{e})^n$$

**polya 定理**

记置换  $G = \{a_1, \dots, a_k\}$ , 在  $[1, m]^n$  上, 不同的向量数目为  $L = (\sum_{i=1}^k m^{l(a_i)})/|G|$

其中  $l(a_i)$  表示置换  $a_i$  可以展开为循环的节数, 比如 (1 2)(3 4)(5) 就是 3 节循环

要保证  $G$  中不含重复的置换

**约瑟夫环**

$f(n, m)$  表示初始有  $n$  个人, 第  $m$  个出队的人是谁 (从 0 号开始报数) 则有递推式  $f(n, m) = (f(n-1, m-1) + k) \pmod{n}$ ,  $f(1, 1) = 0$  其中  $k$  表示每报数  $k$  次一个人出队, 注意编号从 0 开始

**泰勒展开**

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2!}f''(x_0)(x - x_0)^2 + \dots + \frac{1}{n!}f^{(n)}(x_0)(x - x_0)^n$$

**生成函数**

| 数列 | OGF |

$$| < 1, 0, 0, \dots > | 1 |$$

$$| < 1, 1, 1, \dots > | \frac{1}{1-x} |$$

$$| < 1, 2, 3, \dots > | \frac{1}{(1-x)^2} |$$

$$| < 1, -1, 1, -1, \dots > | \frac{1}{1+x} |$$

$$| < 1, 2, 1, 0, 0, \dots > | (1+x)^2 |$$

$$| < 1, 4, 6, 4, 1, 0, 0, \dots > | (1+x)^4 |$$

$$| < 1, c, c^2, \dots > | \frac{1}{1-cx} |$$

$$| < C(n, 0), C(n, 1), C(n, 2), \dots > | (1+x)^n |$$

$$| < 1, C(n, 1), C(n+1, 2), C(n+2, 3), \dots > | (1-x)^{-n} |$$

$$| < 0, 1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \dots > | \ln \frac{1}{1-x} |$$

$$| < 0, 1, -\frac{1}{2}, \frac{1}{3}, -\frac{1}{4}, \dots > | \ln(1+x) |$$

$$| < 1, 1, \frac{1}{2}, \frac{1}{6}, \frac{1}{24}, \dots > | e^x |$$

| 数列 | EGF |

$$| < 1, 1, 1, \dots > | e^x |$$

$$| < 0, 1, 2, \dots > | xe^x |$$

$$| < 1, c, c^2, \dots > | e^{cx} |$$

$< f_n >$  的 EGF 就是  $< \frac{f_n}{n!} >$  的 OGF

**1.18. 博弈论****sg 函数**

一个局面的 SG 为 mex(后继局面的 SG), mex 运算为集合中没出现的最小的自然数

几个局面的和的 SG 为单个的 SG 异或, SG 不为 0 时先手必胜, SG 为 0 时后手必胜

**Nim Game**

$n$  堆石子, 每次可以从一堆里面取任意个石子

对于一堆石子, SG 函数就是石子数。整个游戏的 SG 函数是每一堆石子的 SG 函数的异或和

先手必胜: SG 不为 0; 先手必败: SG 为 0

**Bash Game**

每次最多取  $m$  个石子, 其他同 Nim

一堆石子的 SG 函数为石子数  $\pmod{m+1}$

先手必胜: SG 不为 0; 先手必败: SG 为 0

## Nim-k Game

每次最多可以同时从  $k$  堆石子进行操作, 这  $k$  堆可以取不同数量的石子

一堆石子的 SG 函数为石子数, 对每一个二进制位单独算, 求 SG 函数每一个二进制位 1 的个数  $\bmod (k+1)$ , 如果都为 0, 则必败, 否则必胜

## Anti-Nim Game

不能取石子的一方获胜

必胜: SG 不为 0 且至少有一堆石子数大于 1 或 SG 为 0 且每一堆石子数都不超过 1

必败: 其余为必败

## Anti-SG Game

SG 游戏中最先不能行动的一方获胜

必胜: SG 不为 0 且至少有一个游戏的 SG 大于 1 或 SG 为 0 且每一个游戏的 SG 都不超过 1

必败: 其余为必败

## Staircase Nim

阶梯博弈, 每次可以从一个阶梯上拿掉任意数量石子放到下一层阶梯, 不能操作的为输

SG 函数为奇数阶梯上的石子的异或和, 如果移动偶数层的石子到奇数层, 对手一定可以继续移动这些石子到偶数层, 使得其 SG 不变

先手必胜: SG 不为 0 ; 先手必败: SG 为 0

## Lasker's Nim Game

$n$  堆石子, 每次可以从一堆里面取任意个石子, 或者选择某堆至少为 2 的石子, 分成两堆非空石子。

$SG(0) = 0, SG(1) = 1, SG(2) = 2, SG(3) = 4$

对于  $k \geq 1, SG(4k) = 4k-1, SG(4k+1) = 4k+1, SG(4k+2) = 4k+2, SG(4k+3) = 4k+4$

先手必胜: SG 不为 0 ; 先手必败: SG 为 0

## Wythoff Game

有两堆石子, 每次可以从一堆或者两堆里拿走一样数目的石子, 不能取的为输

必败态为 (1, 2), (3, 5), (4, 7), (6, 10)...

差为 1, 2, 3, 4..... 每一对数的第一个数为前面没出现的最小的正整数

$$a_k = \left\lfloor \frac{k(1+\sqrt{5})}{2} \right\rfloor, b_k = a_k + k$$

## 树上删边游戏

给定一棵  $n$  个点的有根树, 每次可以删掉一个子树, 则叶子节点的 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

先手必胜: SG 不为 0 ; 先手必败: SG 为 0

## 无向图删边游戏

把奇环缩成一个点加一条新边, 把偶环缩成一个点, 不影响 SG, 然后套用树上删边游戏

## 翻硬币游戏

$n$  枚硬币排成一排, 有的正面朝上, 有的反面朝上。游戏者根据某些约束翻硬币 (如: 每次只能翻一或两枚, 或者每次只能翻连续的几枚), 但他所翻动的硬币中, 最右边的必须是从正面翻到反面, 谁不能翻谁输

局面的 SG 值等于局面中每个正面朝上的棋子单一存在 (只有它正面朝上, 其余都正面向下) 时的 SG 值的异或和

每一次只能翻转一枚硬币:  $SG(0) = 0, SG(k) = 1(k > 0)$

每一次可以翻转一枚或两枚硬币:  $SG(n) = n$

每次必须翻动两个硬币, 而且这两个硬币的距离要在可行集  $S = 1, 2, 3$  中: Bash Game

每一次必须翻连续的  $n$  个硬币:  $SG(nk) = 1(k > 0)$ , 其他 SG 函数值为 0

每一次可以翻任意长度的连续一段硬币, SG(x) 为 x 中包含的 2 的最高次幂, 即  $SG(x) = \log_2 x + 1$

先手必胜: SG 不为 0 ; 先手必败: SG 为 0

## K 倍动态减法游戏

有一个整数  $S(S \geq 2)$ , 两个人想让它变成 0。首先, 第一个人需要把  $S$  减掉一个正数  $x(0 < x < S)$ 。之后双方轮流把  $S$  减掉一个正整数, 但都不能超过先前一回合对方减掉的数的  $K(1 \leq K \leq 100000)$  倍, 减到 0 的一方获胜。问谁会获得胜利, 若胜利还要求先手第一步至少减去多少。

```
1const int N=750000;
2int i,j;
3ll n,k,a[N],b[N],ans;
4int main()
5{
6    scanf("%lld%lld",&n,&k);
7    a[1]=b[1]=1;
8    for(i=2,j=0;;i++)
9    {
10        a[i]=b[i-1]+1;
11        if(a[i]>n)break;
12        while(a[j+1]*k<a[i])j++;
13        b[i]=a[i]+b[j];
14    }
15    while(a[i]>n)i--;
16    if(a[i]==n)puts("lose");
17    else
18    {
19        while(n)
20        {
21            while(a[i]>n)i--;
22            n-=a[i];
23            ans=a[i];
24        }
25        printf("%lld",ans);
26    }
27}
```

## Ch2. String

### 2.1. KMP

时间复杂度  $O(n)$

```
1for(int i = 1;i <= n;++ i) {
2    int now = nxt[i - 1];
3    while(now != -1 && s[now + 1] != s[i]) now =
4        ⇨ nxt[now];
5    nxt[i] = now + 1;
6}
```

### 2.2. ex-KMP

时间复杂度  $O(n)$

```
1const int N = 1e6 + 5;
2int next[N], ex[N];
```

```
3void getnxt(char *s) {
4    int i = 0, j, po, len = strlen(s);
5    next[0] = len;
6    while(str[i] == str[i + 1] && i + 1 < len) ++ i;
7    next[1] = i;
8    po = 1;
9    for(i = 2; i < len; ++ i) {
10        if(next[i - po] + i < next[po] + po)
11            next[i] = next[i - po];
12        else {
13            j = next[po] + po - i;
14            if(j < 0) j = 0;
15            while(i + j < len && str[j] == str[j + i])
16                j ++;
17            next[i] = j;
18            po = i;
19        }
20    }
```

```

19     }
20 }
21}
22
23void exkmp(char *s1, char *s2) {
24    int i = 0, j, po, len = strlen(s1), l2 =
        ↳ strlen(s2);
25    getnxt(s2);
26    while(s1[i] == s2[i] && i < l2 && i < len)
27        ++ i;
28    ex[0] = i;
29    po = 0;
30    for(i = 1; i < len; ++ i) {
31        if(next[i - po] + i < ex[po] + po)
32            ex[i] = next[i - po];
33        else {
34            j = ex[po] + po - i;
35            if(j < 0)
36                j = 0;
37            while(i + j < len && j < l2 && s1[j + i] ==
        ↳ s2[j]) ++ j;
38            ex[i] = j;
39            po = i;
40        }
41    }
42}

```

## 2.3. manacher

时间复杂度  $O(n)$

```

1 int len;
2 char s[N], str[N];
3
4 void getstr(){
5     int k = 0;
6     str[k++] = '$';
7     for(int i = 0; i < len; ++ i)
8         str[k++] = '#',
9         str[k++] = s[i];
10    str[k++] = '#';
11    len = k;
12}
13
14 void Manacher() {
15     getstr();
16     int mx = 0, id;
17     for(int i = 1; i < len; ++ i)
18     {
19         if(mx > i)
20             Len[i] = min(Len[2 * id - i], mx - i);
21         else Len[i] = 1;
22         while(str[i + Len[i]] == str[i - Len[i]])
23             ++ Len[i];
24         if(Len[i] + i > mx)
25             mx = Len[i] + i, id = i;
26     }
27}

```

## 2.4. AC 自动机

时间复杂度  $O(\sum n)$

```

1 using namespace std;
2 #include <cstdio>
3 #define N 204
4 char s1[N*2], s2[N*2];
5 int fail[N*2], en[N*2], ch[N*2][26], sz, root;
6 bool vis[N*2];
7
8 void init() {
9     sz = root = 1;
10}

```

```

11 void build() {
12     queue <int> q;
13     q.push(1);
14     fail[1] = 0;
15     while(!q.empty()) {
16         int ind = q.front(); q.pop();
17         for(int i = 0; i < 26; ++ i)
18             if(ch[ind][i]) {
19                 int it = fail[ind];
20                 while(!ch[it][i] && it) it = fail[it];
21                 if(!it) it = 1; else it = ch[it][i];
22                 fail[ch[ind][i]] = it;
23                 if(!vis[ch[ind][i]])
24                     q.push(ch[ind][i]);
25             }
26     }
27     for(int i = 1; i <= sz; ++ i) {
28         for(int j = 0; j < 26; ++ j) {
29             if(!ch[i][j]) {
30                 int it = fail[i];
31                 while(it && !ch[it][j]) it = fail[it];
32                 if(!it) ++ it;
33                 else it = ch[it][j];
34                 ch[i][j] = it;
35             }
36         }
37     }
38}

```

## 2.5. 后缀数组

时间复杂度  $O(n \log n)$ , 询问  $O(1)$

```

1 // SA can be fully replaced by SAM
2 // so we get here a  $O(n \log n)$  approach For its
   ↳ thoughts. (Can be used in jcvb)
3 #include <bits/stdc++.h>
4 using namespace std;
5
6 const int N = 2e5 + 5;
7
8 #define REP(i, a, b) for(int i = (a); i <= (b); ++ i)
9 #define PER(i, a, b) for(int i = (a); i >= (b); -- i)
10
11 int n;
12 char s[N];
13
14 namespace SA {
15     int Mem[N * 10], sa[N], H[N], rk[N];
16
17     void build(int m) {
18         int *x = Mem, *y = Mem + (N << 1), *z = y + (N <<
           ↳ 1);
19         REP(i, 1, m) z[i] = 0;
20         REP(i, 1, n) ++ z[x[i] = s[i]];
21         REP(i, 1, m) z[i] += z[i - 1];
22         REP(i, 1, n) sa[z[x[i]] --] = i;
23         for(int k = 1; k <= n; k <= 1) {
24             int p = 0;
25             REP(i, n - k + 1, n) y[++ p] = i;
26             REP(i, 1, n) if(sa[i] > k) y[++ p] = sa[i] - k;
27             REP(i, 1, m) z[i] = 0;
28             REP(i, 1, n) ++ z[x[i]];
29             REP(i, 1, m) z[i] += z[i - 1];
30             PER(i, n, 1) sa[z[x[y[i]]] --] = y[i];
31             swap(x, y); p = 0;
32             REP(i, 1, n) x[sa[i]] = (y[sa[i]] == y[sa[i - 1]]
           ↳ && y[sa[i] + k] == y[sa[i - 1] + k]) ? p : ++
           ↳ p;
33             if(p == n) break;

```

```

34     m = p;
35 }
36 }
37
38 void getheight() {
39     int Now = 0;
40     REP(i, 1, n) rk[sa[i]] = i;
41     REP(i, 1, n) {
42         if(Now) -- Now;
43         if(rk[i] == n) continue;
44         while(s[i + Now] == s[sa[rk[i] + 1] + Now]) ++
            ↪ Now;
45         H[rk[i]] = Now;
46     }
47 }
48 }
49
50 int main() {
51     scanf("%s", s + 1); n = strlen(s + 1);
52     SA :: build('z');
53     SA :: getheight();
54     REP(i, 1, n) printf("%d ", SA :: sa[i]); puts("");
55     REP(i, 1, n - 1) printf("%d ", SA :: H[i]); puts("");
56 }

```

## 2.6. 后缀自动机

时间复杂度  $O(n\sigma)$

```

1 void extend(int c) {
2     if(ch[last][c] && MAX[ch[last][c]] == MAX[last] +
        ↪ 1) {
3         last = ch[last][c]; stk[++ top] = last;
        ↪ tagged[last] = cur; ++ tag[last]; return;
4     }
5     int fl = !!ch[last][c];
6     np = last; p = last = ++ sz; MAX[p] = MAX[np] + 1;
7     for(;; !ch[np][c] && np; np = fa[np]) ch[np][c] = p;
8     if(!np) {
9         stk[++ top] = last; tagged[last] = cur; ++
            ↪ tag[last]; fa[p] = 1; return;
10    }
11    else {
12        q = ch[np][c];
13        if(MAX[q] == MAX[np] + 1) {
14            fa[p] = q; stk[++ top] = last; tagged[last]
                ↪ = cur; ++ tag[last]; return;
15        }
16        nq = (!fl) ? ++ sz : p;
17        for(register int i = 0; i < 26; ++ i) ch[nq][i] =
            ↪ ch[q][i];
18        fa[nq] = fa[q]; MAX[nq] = MAX[np] + 1;
        ↪ if(!fl) fa[p] = nq; fa[q] = nq; S[nq] =
            ↪ S[q];
19        stk[++ top] = last; tagged[last] = cur; ++
            ↪ tag[last];
20        for(;; ch[np][c] == q && np; np = fa[np])
            ↪ ch[np][c] = nq;
21        return;
22    }
23 }
24

```

## 2.7. ex-后缀自动机

时间复杂度  $O(n\sigma)$

```

1 #include <cstdio>
2 #include <iostream>
3 #include <cstring>
4 #include <algorithm>
5 using namespace std;
6

```

```

7 const int N = 6e5 + 5;
8
9 long long ans;
10
11 int S[N] , MAX[N] , ch[N][26] , n , sz , fa[N] , stk[N]
    ↪ , tag[N] , m , last , p , np , q , nq , top ,
    ↪ taged[N];
12
13 int cur;
14
15 char s[N] , t[N];
16
17 void extend(int c) {
18     if(ch[last][c] && MAX[ch[last][c]] == MAX[last] +
        ↪ 1) {
19         last = ch[last][c]; stk[++ top] = last;
        ↪ taged[last] = cur; ++ tag[last]; return;
20     }
21     int fl = !!ch[last][c];
22     np = last; p = last = ++ sz; MAX[p] = MAX[np] + 1;
23     for(;; !ch[np][c] && np; np = fa[np]) ch[np][c] = p;
24     if(!np) {
25         stk[++ top] = last; taged[last] = cur; ++
            ↪ tag[last]; fa[p] = 1; return;
26     }
27     else {
28         q = ch[np][c];
29         if(MAX[q] == MAX[np] + 1) {
30             fa[p] = q; stk[++ top] = last; taged[last]
                ↪ = cur; ++ tag[last]; return;
31         }
32         nq = (!fl) ? ++ sz : p;
33         for(register int i = 0; i < 26; ++ i) ch[nq][i] =
            ↪ ch[q][i];
34         fa[nq] = fa[q]; MAX[nq] = MAX[np] + 1;
        ↪ if(!fl) fa[p] = nq; fa[q] = nq; S[nq] =
            ↪ S[q];
35         stk[++ top] = last; taged[last] = cur; ++
            ↪ tag[last];
36         for(;; ch[np][c] == q && np; np = fa[np])
            ↪ ch[np][c] = nq;
37         return;
38     }
39 }
40
41 long long sqr(int x) {
42     return x * 1ll * x;
43 }
44
45 void chg(int x) {
46     ans -= sqr(S[x]) * 1ll * (MAX[x] - MAX[fa[x]]); ans
        ↪ += sqr((S[x] += tag[x]) * 1ll * (MAX[x] -
            ↪ MAX[fa[x]]));
47     int now = tag[x];
48     tag[x] = 0;
49     x = fa[x];
50     while(taged[x] != cur) {
51         ans += -sqr(S[x]) * 1ll * (MAX[x] -
            ↪ MAX[fa[x]]);
52         S[x] += now;
53         ans += sqr(S[x]) * 1ll * (MAX[x] - MAX[fa[x]]);
54         x = fa[x];
55     }
56     tag[x] += now;
57 }
58
59 main(void) {
60     scanf("%d" , &n); sz = 1;

```

```

61     for(int i = 1; i <= n; ++ i) {
62         scanf("%s" , s + 1);
63         m = strlen(s + 1);
64         last = 1; top = 0; cur = i;
65         for(register int j = 1; j <= m; ++ j) extend(s[j]
        ↪ - 'a'); taged[1] = cur;
66         for(register int j = top; j >= 1; -- j) {
67             chg(stk[j]);
68         }
69         printf("%lld\n" , ans);
70     }
71 }

```

## 2.8. 回文自动机

时间复杂度  $O(n)$

```

1/// the fastest one PAM APIO2014» J®
2#include <cstdio>
3#include <cstring>
4#include <algorithm>
5
6using namespace std;
7typedef long long LL;
8const int N = 300005;
9char S[N];
10int son[N][26], fail[N];
11int cnt[N], len[N];
12int n, st, i, x, cur, nw, last;
13LL ans;
14
15int newnode(int x) {
16    len[st] = x;
17    cnt[st] = 0;
18    return st++;
19}
20
21void init() {
22    gets(S + 1);
23    n = strlen(S + 1);
24    S[0] = '#';
25    newnode(0);
26    newnode(-1);
27    fail[0] = 1;
28    last = 0;
29}
30
31int get_fail(int x, int n) {
32    while (S[n - len[x] - 1] != S[n]) x = fail[x];
33    return x;
34}
35
36int main() {
37    init();
38    for(int i = 1; i <= n; ++ i) {
39        x = S[i] - 'a';
40        cur = get_fail(last, i);
41        if (!son[cur][x]) {
42            nw = newnode(len[cur] + 2);
43            fail[nw] = son[get_fail(fail[cur], i)][x];
44            son[cur][x] = nw;
45        }
46        last = son[cur][x];
47        cnt[last]++;
48    }
49    for (int i = st - 1; i >= 0; -- i)
50        cnt[fail[i]] += cnt[i];
51    for (int i = 2; i < st; ++ i)
52        ans = max(ans, (LL)len[i] * cnt[i]);
53    printf("%lld\n", ans);
54    return 0;

```

```

55}

```

## 2.9. 最小表示法

时间复杂度  $O(n)$

```

1int k = 0, i = 0, j = 1;
2while (k < n && i < n && j < n) {
3    if (sec[(i + k) % n] == sec[(j + k) % n]) {
4        k++;
5    } else {
6        sec[(i + k) % n] > sec[(j + k) % n] ? i = i + k + 1
        ↪ : j = j + k + 1;
7        if (i == j) i++;
8        k = 0;
9    }
10}
11i = min(i, j);
12return i;

```

## 2.10. 回文串分解

```

1if usable
2border's length between  $[2^{i-1}, 2^i]$ 
3should be a 等差数列
4consider a border u.
5 $|u| > 2^{i-1}$ 
6then every other border would be a border of it
7and let two others are s1 and s2 ( $|s1|$  is the longest)
8then
9 $|u| - |s1|$  and  $|u| - |s2|$  is a 周期 of u
10and both longer than the half length
11so  $|u| - |s1| + |u| - |s2| < |u|$ 
12so  $|u| - |s1| = \gcd(|u| - |s1|, |u| - |s2|)$ 
13proved
14

```

时间复杂度  $O(n \log n + n\sigma)$

```

1#include <bits/stdc++.h>
2using namespace std;
3
4const int N = 1000000 + 5;
5
6namespace pam {
7
8    int sz, tot, last;
9    int ch[N][26], len[N], fail[N];
10    int cnt[N], dep[N], dif[N], slink[N];
11    char s[N];
12
13    int node(int l) {
14        ++ sz;
15        memset(ch[sz], 0, sizeof(ch[sz]));
16        len[sz] = l;
17        fail[sz] = fail[sz] = dep[sz] = 0;
18        return sz;
19    }
20
21    void clear() {
22        sz = -1; last = 0;
23        s[tot = 0] = '$';
24        node(0); node(-1);
25        fail[0] = 1;
26    }
27
28    int getfail(int x) {
29        while (s[tot - len[x] - 1] != s[tot]) x = fail[x];
30        return x;
31    }
32
33    void insert(char c) {

```

```

34     s[++ tot] = c;
35     int now = getfail(last);
36     if (!ch[now][c - 'a']) {
37         int x = node(len[now] + 2);
38         fail[x] = ch[getfail(fail[now])][c - 'a'];
39         dep[x] = dep[fail[x]] + 1;
40         ch[now][c - 'a'] = x;
41
42         dif[x] = len[x] - len[fail[x]];
43         if (dif[x] == dif[fail[x]]) slink[x] =
44             ↪ slink[fail[x]];
45         else slink[x] = fail[x];
46     }
47     last = ch[now][c - 'a'];
48     cnt[last]++;
49 }
50 }
51 using pam :: len;
52 using pam :: fail;
53 using pam :: slink;
54 using pam :: dif;
55
56 int n, dp[N], g[N], pre[N], pre2[N];
57 char s[N], t[N], c[N];
58
59 void print(int x) {
60     if(!x) return;
61     if(pre[x] + 2 == x);

```

```

62     else printf("%d %d\n", pre[x] / 2 + 1, x / 2);
63     print(pre[x]);
64     return;
65 }
66
67 int main() {
68     pam :: clear();
69     scanf("%s", s + 1);
70     n = strlen(s + 1);
71     dp[0] = 0;
72     for(int i = 1; i <= n; ++ i) dp[i] = 1e9;
73     for(int i = 1; i <= n; ++ i) {
74         pam :: insert(s[i]);
75         for(int x = pam :: last; x > 1; x = slink[x]) {
76             g[x] = dp[i - len[slink[x]] - dif[x]] + 1,
77             pre2[x] = i - len[slink[x]] - dif[x];
78             if (dif[x] == dif[fail[x]]) {
79                 if(g[fail[x]] < g[x])
80                     g[x] = g[fail[x]], pre2[x] = pre2[fail[x]];
81             }
82             if(g[x] < dp[i]) {
83                 dp[i] = g[x];
84                 pre[i] = pre2[x];
85             }
86         }
87     }
88     cout << dp[n] << endl;
89     return 0;
90 }

```

## Ch3. Data Structure

### 3.1. KD-tree

```

1 // fastest able to extend to more dimensions
2 // n ^ (1 +(d - 1) / d)
3 #include <bits/stdc++.h>
4 using namespace std;
5
6 const int N = 2e5 + 5;
7
8 struct kd {
9     int sm[2], sn[2], v[2], Sum, S, ls, rs;
10
11     kd (int a, int b, int c) {
12         ls = rs = 0;
13         sm[0] = sn[0] = v[0] = a;
14         sm[1] = sn[1] = v[1] = b;
15         s = sum = c;
16     }
17     kd () {}
18 };
19
20 kd t[N], p[N];
21
22 int n, m, A, B, C, D, root, ans;
23
24 bool cmp(kd a, kd b) {
25     return (a.v[D] == b.v[D]) ? (a.v[D ^ 1] < b.v[D ^
26     ↪ 1]) : (a.v[D] < b.v[D]);
27 }
28
29 void up(int x, int y) {
30     for(int i = 0; i < 2; ++ i)
31         t[x].sm[i] = max(t[x].sm[i], t[y].sm[i]),
32         ↪ t[x].sn[i] = min(t[x].sn[i], t[y].sn[i]);

```

```

31     t[x].sum += t[y].sum;
32 }
33
34 int build(int l, int r, int d) {
35     if(l > r) return 0;
36     int mid = l + r >> 1;
37     D = d;
38     nth_element(p + l, p + mid, p + r + 1, cmp);
39     t[mid] = p[mid];
40     t[mid].ls = build(l, mid-1, d ^ 1), t[mid].rs =
41     ↪ build(mid + 1, r, d ^ 1);
42     if(t[mid].ls) up(mid, t[mid].ls);
43     if(t[mid].rs) up(mid, t[mid].rs);
44     return mid;
45 }
46
47 void ins(int y) {
48     int x = root, d = 0;
49     while(x != y) {
50         pushup(x, y);
51         if(t[y].v[0] == t[x].v[0] && t[y].v[1] ==
52         ↪ t[x].v[1]) {
53             t[x].s += t[y].s;
54             -- n;
55             return;
56         }
57         if(t[y].v[d] < t[x].v[d])
58             t[x].ls = (!t[x].ls) ? y : t[x].ls, x = t[x].ls;
59         else
60             t[x].rs = (!t[x].rs) ? y : t[x].rs, x =
61             ↪ t[x].rs;
62         d ^= 1;
63     }
64 }
65
66 int query(int x) {
67     if(!x || t[x].sm[0] < A || t[x].sn[0] > C ||
68     ↪ t[x].sm[1] < B || t[x].sn[1] > D)
69         return 0;

```

```

64 if(t[x].sm[0] <= C && t[x].sn[0] >= A && t[x].sm[1]
    ↪ <= D && t[x].sn[1] >= B)
65     return t[x].sum;
66 int ret=0;
67 if(t[x].v[0] >= A && t[x].v[0] <= C && t[x].v[1] >=
    ↪ B && t[x].v[1] <= D)
68     ret += t[x].s;
69 ret += query(t[x].ls) + query(t[x].rs);
70 return ret;
71 }

```

## 3.2. LCT

LCT-simple

```

1#define lc ch[x][0]
2#define rc ch[x][1]
3//no lian cao zuo,just liantongxing
4// n log n
5struct NODE {
6    int LMAX, RMAX, S, MAX;
7    friend NODE operator + (NODE A, NODE B) {
8        C.S = A.S + B.S;
9        C.LMAX = max(A.S + B.LMAX, A.LMAX);
10       C.RMAX = max(B.RMAX, B.S + A.RMAX);
11       C.MAX = max(A.MAX, max(B.MAX, A.RMAX +
        ↪ B.LMAX));
12       return C;
13    }
14 }C, T[N]; // info, to be changed
15multiset <int> Cl[N], Cm[N];
16multiset <int> ::reverse_iterator IT;
17bool rev[N];int n, m, x, y, rt, ch[N][2], fa[N], val[N]
18void up(int x) {
19    NODE &D = T[x];
20    D.S = D.MAX = D.LMAX = D.RMAX = 0;
21    D.S = val[x];
22    D.MAX = D.LMAX = D.RMAX = max(max(val[x], 0),
    ↪ get(x) + val[x]);
23    D.MAX = max(get2(x) + val[x], max(D.MAX,
    ↪ (Cm[x].size()) ? (*Cm[x].rbegin()) : 0));
24    if(lc) D = T[lc] + D;
25    if(rc) D = D + T[rc];
26 }
27
28void pt(int x) {
29    rev[x] ^= 1;
30    swap(lc, rc);
31    swap(T[x].LMAX, T[x].RMAX);
32 }
33
34void pd(int x) {
35    if(rev[x]) {
36        if(lc)pt(lc); if(rc)pt(rc);
37        rev[x] = 0;
38    }
39 }
40
41bool dir(int x) {
42    return ch[fa[x]][1] == x;
43 }
44
45bool isrt(int x) {
46    return (ch[fa[x]][1] != x && ch[fa[x]][0] != x);
47 }
48
49void dn(int x) {
50    if(!isroot(x))
51        dn(fa[x]);
52    pd(x);
53 }

```

```

54
55void rotate(int x) {
56    int Dx = dir(x), f = fa[x], Df = dir(fa[x]), GF =
    ↪ fa[f];
57    if(!isrt(f)) ch[GF][Df] = x;
58    fa[x] = GF; fa[f] = x;
59    if(ch[x][!Dx]) fa[ch[x][!Dx]] = f;
60    ch[f][Dx] = ch[x][!Dx]; ch[x][!Dx] = f;
61    up(f); up(x);
62 }
63
64void splay(int x) {
65    dn(x);
66    for(; !isrt(x); rotate(x))
67        if(!isrt(fa[x])) rotate(dir(fa[x]) == dir(x) ?
    ↪ fa[x] : x);
68 }
69
70void access(int x) {
71    int t = 0;
72    for(; x; t = x, x = fa[x]) {
73        splay(x);
74        rc = t; up(x);
75    }
76 }
77
78void make_root(int x) {
79    access(x);
80    splay(x);
81    pt(x);
82    rt = x;
83 }
84
85void link(int x, int y) {
86    make_root(x);
87    make_root(y);
88    fa[y] = x;
89    A(x, y, 1);
90    up(x);
91    rt = x;
92 }
93
94void cut(int x, int y) {
95    make_root(x);
96    access(y);
97    splay(y);
98    ch[y][0] = fa[x] = 0;
99    up(y);
100 }

```

LCT-支持链维护

```

1#define lc ch[x][0]
2#define rc ch[x][1]
3// LCT complexed version ji suan zuidaquanzhilian
4// n log^2
5struct NODE {
6    int LMAX, RMAX, S, MAX;
7
8    friend NODE operator + (NODE A, NODE B) {
9        C.S = A.S + B.S;
10       C.LMAX = max(A.S + B.LMAX, A.LMAX);
11       C.RMAX = max(B.RMAX, B.S + A.RMAX);
12       C.MAX = max(A.MAX, max(B.MAX, A.RMAX +
        ↪ B.LMAX));
13       return C;
14    }
15 }C, T[N]; // info, to be changed
16multiset <int> Cl[N], Cm[N];

```



```

17 multiset <int> ::reverse_iterator IT;
18 bool rev[N];
19 int n, m, x, y, rt, ch[N][2], fa[N], val[N];
20 int get(int x) {
21     if(!Cl[x].size())
22         return 0;
23     return *Cl[x].rbegin();
24 } // choose one
25 int get2(int x) {
26     if(Cl[x].size() < 2) return get(x);
27     IT = Cl[x].rbegin();
28     return *IT + *(++ IT);
29 } // choose two
30 void up(int x) {
31     NODE &D = T[x];
32     D.S = D.MAX = D.LMAX = D.RMAX = 0;
33     D.S = val[x];
34     D.MAX = D.LMAX = D.RMAX = max(max(val[x], 0),
35     ↪ get(x) + val[x]);
36     D.MAX = max(get2(x) + val[x], max(D.MAX,
37     ↪ (Cm[x].size()) ? (*Cm[x].rbegin()) : 0));
38     if(lc) D = T[lc] + D;
39     if(rc) D = D + T[rc];
40 }
41 void pt(int x) {
42     rev[x] ^= 1;
43     swap(lc, rc);
44     swap(T[x].LMAX, T[x].RMAX);
45 }
46 void pd(int x) {
47     if(rev[x]) {
48         if(lc) pt(lc); if(rc) pt(rc);
49         rev[x] = 0;
50     }
51 }
52 bool dir(int x) {
53     return ch[fa[x]][1] == x;
54 }
55 #define isroot(x) (ch[fa[x]][1] != x && ch[fa[x]][0] !=
56     ↪ x)
57 void dn(int x) {
58     if(!isroot(x))
59         dn(fa[x]);
60     pd(x);
61 }
62 int Dx, f, Df, GF;
63 void rotate(int x) {
64     Dx = dir(x);
65     f = fa[x];
66     Df = dir(fa[x]);
67     GF = fa[f];
68     if(!isroot(f)) ch[GF][Df] = x;
69     fa[x] = GF;
70     fa[f] = x;
71     if(ch[x][!Dx]) fa[ch[x][!Dx]] = f;
72     ch[f][Dx] = ch[x][!Dx];
73     ch[x][!Dx] = f;
74     up(f);
75     up(x);
76 }
77 void splay(int x) {

```

```

83     dn(x);
84     while(!isroot(x)) {
85         if(isroot(fa[x])) {
86             rotate(x);
87             return;
88         }
89         if(dir(x) == dir(fa[x]))
90             rotate(fa[x]), rotate(x);
91         else rotate(x), rotate(x);
92     }
93 }
94 #define del(K, P) K.erase(K.find(P))
95 void A(int x, int y, int oh) {
96     if(oh == 1) {
97         Cl[x].insert(T[y].LMAX);
98         Cm[x].insert(T[y].MAX);
99     }
100     else {
101         del(Cl[x], T[y].LMAX);
102         del(Cm[x], T[y].MAX);
103     }
104 }
105 void access(int x) {
106     int t = 0;
107     for(; x; t = x, x = fa[x]) {
108         splay(x);
109         if(rc) A(x, rc, 1);
110         if(t) A(x, t, -1);
111         rc = t; up(x);
112     }
113 }
114 void make_root(int x) {
115     access(x);
116     splay(x);
117     pt(x);
118     rt = x;
119 }
120 void link(int x, int y) {
121     make_root(x);
122     make_root(y);
123     fa[y] = x;
124     A(x, y, 1); // of importance
125     up(x);
126     rt = x;
127 }
128 void cut(int x, int y) {
129     make_root(x);
130     access(y);
131     splay(y);
132     ch[y][0] = fa[x] = 0;
133     up(y);
134 }

```

### 3.3. Treap

```

1 // cheating version
2 // nlog n
3 #pragma GCC optimize(2)
4 #include <bits/stdc++.h>
5 using namespace std;
6
7 const int N = 1e6 + 5;
8

```

```

9 int S[N], ch[N][2], n, m, l, r, x, rt, q, D[N], tot =
  ↪ 0;
10 bool rev[N]; char s[N];
11
12 #define pii pair<int, int>
13 #define mp make_pair
14 #define lc ch[x][0]
15 #define rc ch[x][1]
16
17 void up(int x) {
18     if(x)
19         S[x] = S[lc] + S[rc] + 1;
20 }
21
22 void pt(int x) {
23     if(!x) return;
24     rev[x] ^= 1;
25     swap(lc, rc);
26 }
27
28 void dn(int x) {
29     if(rev[x]) {
30         rev[x] = 0;
31         pt(lc); pt(rc);
32     }
33 }
34
35 int Rnd() {
36     ++tot;
37     if(tot > n) tot = 1;
38     return D[tot];
39     // attention
40     // windows
41     return 32768 * rand() + rand();
42     // Linux
43     return rand();
44 }
45
46 int merge(int x, int y) { // Here x is smaller because
  ↪ Xulie
47     if(!x || !y) return x + y;
48     dn(x); dn(y);
49     if(Rnd() % (S[x] + S[y]) >= S[x]) {
50         ch[y][0] = merge(x, ch[y][0]);
51         up(y);
52         return y;
53     }
54     else {
55         ch[x][1] = merge(ch[x][1], y);
56         up(x);
57         return x;
58     }
59 }
60
61 pii split(int x, int len) {
62     dn(x);
63     if(!len) return mp(0, x);
64     if(S[lc] + 1 == len) {
65         int t = ch[x][1];
66         ch[x][1] = 0;
67         up(x);
68         return mp(x, t);
69     }
70     if(len <= S[lc]) {
71         pii now = split(lc, len);
72         ch[x][0] = now.second;
73         up(x);
74         return mp(now.first, x);
75     }

```

```

76     else {
77         pii now = split(rc, len - S[lc] - 1);
78         ch[x][1] = now.first;
79         up(x);
80         return mp(x, now.second);
81     }
82 }
83
84 int All = 0;
85
86 void dfs(int x) {
87     dn(x);
88     if(ch[x][0]) dfs(ch[x][0]);
89     ++All;
90     printf("%d%c", x, (All == n) ? '\n' : ' ');
91     if(ch[x][1]) dfs(ch[x][1]);
92 }
93
94 pii A, B, C;
95
96 int build(int l, int r) {
97     if(l > r) return 0;
98     int x = (l + r) >> 1;
99     lc = build(l, x - 1);
100    rc = build(x + 1, r);
101    up(x);
102    return x;
103 }
104
105 int main() {
106     while(1) {
107         scanf("%d%d", &n, &q);
108         if(n == -1) return 0;
109         for(int i = 1; i <= n; ++i) D[i] = i;
110         random_shuffle(D + 1, D + n + 1);
111         tot = 0;
112         rt = build(1, n);
113         for(int i = 1; i <= q; ++i) {
114             scanf("%s%d%d", s, &l, &r);
115             if(s[0] == 'C') {
116                 scanf("%d", &x);
117                 A = split(rt, l - 1);
118                 B = split(A.second, r - l + 1);
119                 int w = merge(A.first, B.second);
120                 C = split(w, x);
121                 rt = merge(C.first, merge(B.first,
122                 ↪ C.second));
123             }
124             else {
125                 A = split(rt, l - 1);
126                 B = split(A.second, r - l + 1);
127                 pt(B.first);
128                 rt = merge(A.first, merge(B.first,
129                 ↪ B.second));
130             }
131         }
132         All = 0;
133         dfs(rt);
134     }

```

### 3.4. 可持久化 Treap

```

1 // nlog n
2 #include <bits/stdc++.h>
3 using namespace std;
4
5 const int N = 2e5 + 5;
6 const int MAX = N * 128;

```

```

7
8#define LL long long
9
10#define lc ch[x][0]
11#define rc ch[x][1]
12
13namespace Treap{
14    int rt[N], pos[MAX], S[MAX];
15    bool rev[MAX];
16    int val[MAX];
17    LL Sum[MAX];
18    int now, tot, ch[MAX][2];
19    inline int Rand() {static int seed = 703; return
    ↪ seed = int(seed * 48271LL % (~0u>>1));}
20    int newnode(LL x) {
21        val[++ tot] = x;
22        Sum[tot] = x;
23        pos[tot] = Rand();
24        S[tot] = 1;
25        return tot;
26    }
27
28    int copy(int x){
29        int t = newnode(0);
30        ch[t][0] = lc;
31        ch[t][1] = rc;
32        S[t] = S[x];
33        val[t] = val[x];
34        Sum[t] = Sum[x];
35        rev[t] = rev[x];
36        return t;
37    }
38
39    int pt(int x) {
40        x = copy(x);
41        rev[x] ^= 1;
42        return x;
43    }
44
45    void up(int x){
46        S[x] = S[lc] + S[rc] + 1;
47        Sum[x] = Sum[lc] + Sum[rc] + val[x];
48    }
49
50    void dn(int x){
51        if(!rev[x]) return;
52        if(lc) lc = pt(lc);
53        if(rc) rc = pt(rc);
54        swap(lc, rc);
55        rev[x] = 0;
56    }
57
58    int merge(int x,int y){
59        if(!x || !y) return x + y;
60        if(pos[x] < pos[y]) {
61            dn(x);
62            rc = merge(rc ,y);
63            up(x);
64            return x;
65        }
66        dn(y);
67        ch[y][0] = merge(x, ch[y][0]);
68        up(y);
69        return y;
70    }
71
72    void split(int rt, int k, int &x, int &y) {
73        if(!rt) return x = y = 0, void();
74        dn(rt);

```

```

75        if(S[ch[rt][0]] >= k)
76            y = copy(rt), split(ch[y][0], k, x, ch[y][0]),
    ↪ up(y);
77        else x = copy(rt), split(ch[x][1], k -
    ↪ S[ch[x][0]] - 1, ch[x][1], y), up(x);
78    }
79}
80
81using namespace Treap;
82
83int main() {
84    int n, v = 0, op, l, r, a, b, c;
85    LL ans = 0;
86    for(cin >> n; n --;) {
87        scanf("%d%d", &v, &op);
88        if(op == 1) {
89            scanf("%d%d", &l, &r); l ^= ans; r ^= ans;
90            split(rt[v], l, a, b);
91            rt[++ now] = merge(a, merge(newnode(r),
    ↪ b));
92        } else if(op == 2) {
93            scanf("%d", &l); l ^= ans;
94            split(rt[v], l - 1, a, b);
95            split(b, 1, b, c);
96            rt[++ now] = merge(a, c);
97        } else if(op == 3) {
98            scanf("%d%d", &l, &r);
99            l ^= ans, r ^= ans;
100            split(rt[v], r, a, c);
101            split(a, l - 1, a, b);
102            rev[b]^=1;
103            rt[++ now] = merge(a, merge(b, c));
104        } else if(op == 4) {
105            scanf("%d%d", &l, &r);
106            l ^= ans, r ^= ans;
107            split(rt[v], r, a, c);
108            split(a, l - 1, a, b);
109            printf("%lld\n", ans = Sum[b]);
110            rt[++ now] = merge(a, merge(b, c));
111        }
112    }
113    return 0;
114}

```

### 3.5. 线段树合并

```

1// nlog n
2#include <bits/stdc++.h>
3using namespace std;
4
5const int MAX = 1e6 + 5;
6
7#define lc ch[x][0]
8#define rc ch[x][1]
9
10int ch[MAX][2], n, m, x, y, sz, S[MAX];
11
12struct NODE {
13    long long Sum;
14    friend NODE operator + (NODE a, NODE b) {
15        return (NODE){a.Sum + b.Sum};
16    }
17}T[MAX];
18
19int merge(int x, int y) {
20    if(!x || !y) return x + y;
21    T[x] = T[x] + T[y];
22    lc = merge(ch[x][0], ch[y][0], l, mid);
23    rc = merge(ch[x][1], ch[y][1], mid + 1, r);

```

```

24     return x;
25 }
26
27 int main() {
28
29 }

```

### 3.6. Segment tree beats

```

1 // nlog^2
2 // a the min b the second min
3 #pragma GCC optimize(3)
4 #include <bits/stdc++.h>
5 using namespace std;
6
7 const int N = 1.6e6 + 5;
8 const int Segment_Size = N * 4;
9
10 #define int long long
11
12 int n, m, num2[N], cnt2, num[N];
13 map <int, int> Map, C;
14
15 int S[Segment_Size], a[Segment_Size], b[Segment_Size],
16     ↪ L[Segment_Size], R[Segment_Size],
17     ↪ ans[Segment_Size], laz[Segment_Size];
18 int cnt[Segment_Size], ll[Segment_Size],
19     ↪ rr[Segment_Size];
20
21 #define getmid int mid = (L[no] + R[no]) >> 1
22 #define lc (no << 1)
23 #define rc (no << 1 | 1)
24
25 bool one(int no) {
26     return (b[no] == 1e9 + 1);
27 }
28
29 void up(int no) {
30     if(L[no] == R[no]) return;
31     ans[no] = ans[lc] + ans[rc];
32     S[no] = S[lc] + S[rc];
33     b[no] = 1e9 + 1;
34     if(!one(lc)) b[no] = b[lc];
35     if(!one(rc)) b[no] = min(b[no], b[rc]);
36     if(a[lc] <= a[rc]) {
37         a[no] = a[lc];
38         cnt[no] = cnt[lc] + (a[lc] == a[rc]) * cnt[rc];
39     }
40     else a[no] = a[rc], cnt[no] = cnt[rc];
41     if(a[no] != a[lc]) b[no] = min(b[no], a[lc]);
42     if(a[no] != a[rc]) b[no] = min(b[no], a[rc]);
43     ll[no] = ll[lc];
44     rr[no] = rr[rc];
45     ans[no] += abs(rr[lc] - ll[rc]);
46     if(rr[lc] != ll[rc]) {
47         if(rr[lc] == a[no] || ll[rc] == a[no]) {
48             ++ cnt[no];
49         }
50     }
51 }
52
53 void pt(int no, int x) {
54     if(x <= a[no]) return;
55     ans[no] -= 1LL * cnt[no] * (x - a[no]);
56     a[no] = x;
57     ll[no] = max(ll[no], x);
58     rr[no] = max(rr[no], x);
59     laz[no] = max(laz[no], x);
60 }

```

```

59 void pd(int no) {
60     if(!laz[no]) return;
61     pt(lc, laz[no]);
62     pt(rc, laz[no]);
63     laz[no] = 0;
64     up(no);
65 }
66
67 void build(int no, int l, int r) {
68     if(l > r) return;
69     L[no] = l, R[no] = r;
70     laz[no] = 0;
71     getmid;
72     if(l == r) {
73         S[no] = num2[l + 1] - num2[l] - 1;
74         ans[no] = 0;
75         a[no] = 0;
76         b[no] = 1e9 + 1;
77         cnt[no] = 0;
78         ll[no] = rr[no] = 0;
79         return;
80     }
81     build(lc, l, mid);
82     build(rc, mid + 1, r);
83     up(no);
84 }
85
86 void modify(int no, int l, int r, int x) {
87     if(a[no] > x) return;
88     if(L[no] == l && R[no] == r) {
89         if(b[no] > x) {
90             pt(no, x);
91             return;
92         }
93     }
94     pd(no);
95     getmid;
96     if(l > mid) modify(rc, l, r, x);
97     else if(r <= mid) modify(lc, l, r, x);
98     else modify(lc, l, mid, x), modify(rc, mid + 1, r,
99     ↪ x);
100     up(no);
101 }
102 long long cur = 0;
103
104 struct Qu {
105     int l, r, h;
106     void input() {
107         scanf("%lld%lld%lld", &l, &r, &h);
108     }
109 } Q[N];
110
111 set < pair <int, int> > S2;
112
113 bool inter(pair <int, int> a, pair <int, int> b) {
114     if(a.first > b.first) swap(a, b);
115     if(a.second < b.first) return 0;
116     return 1;
117 }
118
119 void song(int l, int r) {
120     while(S2.size()) {
121         set < pair <int, int> > :: iterator it =
122         ↪ S2.upper_bound(make_pair(l, 1e9 + 1));
123         bool flg = 0;
124         if(inter(*it, make_pair(l, r))) {
125             l = min((*it).first, l);

```

```

125         r = max((*it).second, r);
126         flg = 1;
127         cur -= (*it).second - (*it).first;
128         S2.erase(*it);
129         continue;
130     }
131     if(it != S2.begin()) {
132         -- it;
133         if(inter(*it, make_pair(l, r))) {
134             l = min((*it).first, l);
135             r = max((*it).second, r);
136             flg = 1;
137             cur -= (*it).second - (*it).first;
138             S2.erase(*it);
139             continue;
140         }
141     }
142     if(!flg) break;
143 }
144 cur += r - l;
145 S2.insert(make_pair(l, r));
146 }
147
148 void solve() {
149     cin >> n;
150     for(int i = 1; i <= n; ++ i) Q[i].input();
151     Map.clear();

```

```

152     for(int i = 1; i <= n; ++ i) ++ Map[Q[i].l], ++
153         ↪ Map[Q[i].r];
154     cnt2 = 0; C.clear();
155     for(auto x : Map) C[x.first] = ++ cnt2, num2[cnt2]
156         ↪ = x.first;
157     S2.clear(); cur = 0;
158     build(1, 1, cnt2);
159     for(int i = 1; i <= n; ++ i){
160         song(Q[i].l, Q[i].r);
161         Q[i].l = C[Q[i].l];
162         Q[i].r = C[Q[i].r] - 1;
163         modify(1, Q[i].l, Q[i].r, Q[i].h);
164         printf("%lld\n", ans[1] + cur * 2 + ll[1] +
165             ↪ rr[1]);
166     }
167 }
168
169 main() {
170     int t;
171     for(cin >> t; t --> 0;) {
172         solve();
173     }
174 }

```

## 3.7. Hints

- 1 李超树
- 2 优势线段 交点在哪边就把哪边的下放

# Ch4. Graph Theory

## 4.1. 最短路

### dijkstra

```

1 // 最短路 dij
2 // 复杂度  $O(m \log n)$ 
3 #include <iostream>
4 #include <cstring>
5 #include <cstdlib>
6 #include <cstdio>
7 #include <algorithm>
8 #include <queue>
9 using namespace std;
10
11 const int maxn = 500010, inf = 1e9 + 233;
12
13 struct poi{int x, dis;};
14
15 priority_queue<poi>q;
16 bool operator < (poi a, poi b) {return a.dis > b.dis;}
17
18 struct edge{int too, dis, pre;}e[maxn];
19
20 int n, m, s, x, y, z, tot;
21 int a[maxn], last[maxn], dist[maxn];
22
23 template <typename T>
24 inline void read(T &k)
25 {
26     int f = 1; k = 0; char c = getchar();
27     while (c < '0' || c > '9') c == '-' && (f = -1), c
28         ↪ = getchar();
29     while (c <= '9' && c >= '0') k = k * 10 + c - '0',
30         ↪ c = getchar();
31     k *= f;
32 }

```

```

32 inline void add(int x, int y, int z) {e[++tot] =
33     ↪ (edge){y, z, last[x]}; last[x] = tot;}
34
35 inline void dij(int s)
36 {
37     memset(dist, 0x3f, sizeof(dist));
38     dist[s] = 0; q.push((poi){s, 0});
39
40     while (!q.empty())
41     {
42         poi now = q.top(); q.pop();
43         if (now.dis != dist[now.x]) continue;
44
45         for (int i = last[now.x], too; i; i = e[i].pre)
46             if (dist[too = e[i].too] > dist[now.x] +
47                 ↪ e[i].dis)
48             {
49                 dist[too] = dist[now.x] + e[i].dis;
50                 q.push((poi){too, dist[too]});
51             }
52     }
53 }
54
55 int main()
56 {
57     read(n); read(m); read(s);
58     for (int i = 1; i <= m; i++)
59         read(x), read(y), read(z), add(x, y, z);
60
61     dij(s);
62
63     for (int i = 1; i <= n; i++) printf("%d ",
64         ↪ dist[i]);
65 }

```

## 4.2. 次短路

时间复杂度  $O(n \log n)$

- 1 // 严格次短路
- 2 // 复杂度  $O(m \log n)$

```

3#include <iostream>
4#include <cstring>
5#include <cstdlib>
6#include <cstdio>
7#include <algorithm>
8#include <queue>
9using namespace std;
10
11const int maxn = 500010, inf = 1e9 + 233;
12
13struct poi{int x, dis1, dis2;};
14
15priority_queue<poi>q;
16bool operator < (poi a, poi b) {return a.dis1 > b.dis1
    ↪ || (a.dis1 == b.dis1 && a.dis2 > b.dis2);}
17
18struct edge{int too, dis, pre;}e[maxn];
19
20int n, m, x, y, z, tot;
21int a[maxn], last[maxn], dist1[maxn], dist2[maxn];
22
23template <typename T>
24inline void read(T &k)
25{
26    int f = 1; k = 0; char c = getchar();
27    while (c < '0' || c > '9') c == '-' && (f = -1), c
    ↪ = getchar();
28    while (c <= '9' && c >= '0') k = k * 10 + c - '0',
    ↪ c = getchar();
29    k *= f;
30}
31
32inline void add(int x, int y, int z) {e[++tot] =
    ↪ (edge){y, z, last[x]}; last[x] = tot;}
33
34inline void dij(int s)
35{
36    memset(dist1, 0x3f, sizeof(dist1));
37    memset(dist2, 0x3f, sizeof(dist2));
38    dist1[s] = 0; q.push((poi){s, 0, dist2[1]});
39
40    while (!q.empty())
41    {
42        poi now = q.top(); q.pop();
43        if (now.dis1 != dist1[now.x] || now.dis2 !=
    ↪ dist2[now.x]) continue;
44        for (int i = last[now.x], too; i; i = e[i].pre)
45        {
46            int delta = dist1[now.x] + e[i].dis;
47            if (dist1[too = e[i].too] > delta)
48            {
49                dist2[too] = dist1[too];
50                dist1[too] = delta;
51                q.push((poi){too, dist1[too],
    ↪ dist2[too]});
52            }
53            else if (dist1[too] < delta && dist2[too] >
    ↪ delta)
54            {
55                dist2[too] = delta;
56                q.push((poi){too, dist1[too],
    ↪ dist2[too]});
57            }
58
59            delta = dist2[now.x] + e[i].dis;
60            if (dist1[too = e[i].too] > delta)
61            {
62                dist2[too] = dist1[too];
63                dist1[too] = delta;

```

```

64                q.push((poi){too, dist1[too],
    ↪ dist2[too]});
65            }
66            else if (dist1[too] < delta && dist2[too] >
    ↪ delta)
67            {
68                dist2[too] = delta;
69                q.push((poi){too, dist1[too],
    ↪ dist2[too]});
70            }
71        }
72    }
73}
74
75int main()
76{
77    read(n); read(m);
78    for (int i = 1; i <= m; i++)
79        read(x), read(y), read(z), add(x, y, z), add(y,
    ↪ x, z);
80
81    dij(1);
82
83    printf("%d\n", dist2[n]);
84}

```

### 4.3. K-短路

时间复杂度  $O(n \log n + m \log m + k \log k)$

```

1// ans[K] 为 K 短路
2// 复杂度  $O(n \log n + m \log m + k \log k)$ 
3#include <iostream>
4#include <cstring>
5#include <cstdio>
6#include <algorithm>
7#include <cmath>
8#include <queue>
9#define ll long long
10
11using namespace std;
12const int maxn = 500010, inf = 1e9 + 233;
13
14struct Edge{int x, y, z;}p[maxn];
15struct edge{int x, too, dis, pre;}e[maxn << 1];
16struct poi{int x; ll dis;};
17priority_queue<poi>q, Q;
18bool operator < (poi a, poi b) {return a.dis > b.dis;}
19int n, m, tott, s, t, K, N, TT, x, y, z, tot;
20ll dist[maxn], w[maxn], ans[maxn];
21int rt[maxn], seq[maxn], last[maxn], from[maxn],
    ↪ dep[maxn];
22
23inline void add(int x, int y, int z) {e[++tott] =
    ↪ (edge){x, y, z, last[x]}; last[x] = tot;}
24
25inline void dij(int s)
26{
27    memset(dist, 0x3f, sizeof(dist));
28    memset(dep, 0x3f, sizeof(dep));
29    memset(from, 0, sizeof(from));
30    q.push((poi){s, dist[s] = 0}); dep[s] = 1;
31
32    while (!q.empty())
33    {
34        poi now = q.top(); q.pop();
35        if (dist[now.x] != now.dis) continue;
36        for (int i = last[now.x], too; i; i = e[i].pre)
37            if (dist[too = e[i].too] > dist[now.x] +
    ↪ e[i].dis)

```

```

38     {
39         dist[too] = dist[now.x] + e[i].dis;
40         dep[too] = dep[now.x] + 1;
41         from[too] = i;
42         q.push((poi){too, dist[too]});
43     }
44 }
45
46 memset(w, 0, sizeof(w));
47 for (int i = 1; i <= N; i++)
48     if (from[i]) w[from[i]] = -1;
49 for (int i = 1; i <= m; i++)
50 {
51     if (~w[i] && dist[p[i].x] < inf && dist[p[i].y]
52         ↪ < inf)
53         w[i] = -dist[p[i].x] + dist[p[i].y] +
54         ↪ p[i].z;
55     else w[i] = -1;
56 }
57
58 inline bool cmp_dep(int p, int q){return dep[p] <
59     ↪ dep[q];}
60
61 struct Heap
62 {
63     ll key;
64     int pos, lc, rc, dist;
65 }hp[maxn];
66
67 inline int merge_simple(int x, int y)
68 {
69     if (!x) return y; if (!y) return x;
70     if (hp[x].key > hp[y].key) swap(x, y);
71     hp[x].rc = merge_simple(hp[x].rc, y);
72     if (hp[hp[x].lc].dist < hp[hp[x].rc].dist)
73         swap(hp[x].lc, hp[x].rc);
74     hp[x].dist = hp[hp[x].rc].dist + 1;
75     return x;
76 }
77
78 inline int merge_full(int x, int y)
79 {
80     if (!x) return y; if (!y) return x;
81     if (hp[x].key > hp[y].key) swap(x, y);
82     int now = ++tott;
83     hp[now] = hp[x];
84     hp[now].rc = merge_full(hp[now].rc, y);
85     if (hp[hp[now].lc].dist < hp[hp[now].rc].dist)
86         swap(hp[now].lc, hp[now].rc);
87     hp[now].dist = hp[hp[now].rc].dist + 1;
88     return now;
89 }
90
91 inline void clear()
92 {
93     tot = 0;
94     memset(last, 0, sizeof(last));
95     memset(rt, 0, sizeof(rt));
96 }
97
98 int main()
99 {
100     while (scanf("%d%d", &n, &m) != EOF)
101     {
102         clear();
103         scanf("%d%d%d", &s, &t, &K, &TT);

```

```

104     for (int i = 1; i <= m; i++)
105         scanf("%d%d%d", &p[i].x, &p[i].y,
106             ↪ &p[i].z), add(p[i].y, p[i].x, p[i].z);
107     N = n; tott = 0;
108     dij(t);
109     if (dist[s] > TT)
110     {
111         puts("Whitesnake!");
112         continue;
113     }
114     for (int i = 1; i <= N; i++) seq[i] = i;
115     sort(seq + 1, seq + 1 + N, cmp_dep);
116     tot = 0; memset(last, 0, sizeof(last));
117     for (int i = 1; i <= m; i++)
118         add(p[i].x, p[i].y, p[i].z);
119     rt[t] = 0; hp[0].dist = -1;
120     for (int i = 1; i <= N; i++)
121     {
122         int x = seq[i], y = e[from[x]].too;
123         rt[x] = 0;
124         for (int j = last[x]; j; j = e[j].pre)
125             if (~w[j])
126             {
127                 hp[++tott] = (Heap){w[j], j, 0, 0, 0};
128                 rt[x] = merge_simple(rt[x], tott);
129             }
130             if (i == 1) continue;
131             rt[x] = merge_full(rt[x], rt[y]);
132         }
133     }
134     while (!Q.empty()) Q.pop();
135     Q.push((poi){0, dist[s]});
136     p[0].y = s;
137     for (int k = 1; k <= K; k++)
138     {
139         if (Q.empty())
140         {
141             ans[k] = -1;
142             continue;
143         }
144         poi now = Q.top(); Q.pop();
145         ans[k] = now.dis;
146         int P = now.x;
147         if (hp[P].lc)
148             Q.push((poi){hp[P].lc, hp[hp[P].lc].key
149                 ↪ + now.dis - hp[P].key});
150         if (hp[P].rc)
151             Q.push((poi){hp[P].rc, hp[hp[P].rc].key
152                 ↪ + now.dis - hp[P].key});
153         if (rt[p[hp[P].pos].y])
154             Q.push((poi){rt[p[hp[P].pos].y],
155                 ↪ hp[rt[p[hp[P].pos].y]].key +
156                 ↪ now.dis});
157     }
158     }
159
160     if (ans[K] == -1 || ans[K] > TT)
161         ↪ puts("Whitesnake!");
162     else puts("yareyaredawa");
163 }

```

#### 4.4. 差分约束

时间复杂度  $O(n \log n)$

```

1 // 差分约束
2 // 求最大值为最短路 最小值为最长路
3 // 本质上是不等式和 spfa 的松弛过程相似
4 // 复杂度 spfa 复杂度  $O(kE)$ 

```

```

5#include<iostream>
6#include<cstring>
7#include<cstdlib>
8#include<cstdio>
9#define ll long long
10using namespace std;
11const int maxn = 500010, inf = 1e9 + 233;
12struct edge{int too, dis, pre;}e[maxn];
13int n, m, x, y, z, front, rear, tot, mx, mn;
14ll dist[maxn], ans;
15int h[maxn], v[maxn], last[maxn], tim[maxn];
16bool flag;
17
18inline void add(int x, int y, int z) {e[++tot] =
    ↪ (edge){y, z, last[x]}; last[x] = tot;}
19
20inline bool spfa()
21{
22    for (int i = mn; i <= mx; i++) v[i] = 0, dist[i] =
    ↪ -inf;
23    dist[mn] = 0; v[mn] = 1; front = rear = 0;
    ↪ h[++rear] = mn;
24    while (front != rear)
25    {
26        int now = h[++front];
27        for (int i = last[now], too; i; i = e[i].pre)
28            if (dist[too = e[i].too] < dist[now] +
    ↪ e[i].dis)
29            {
30                dist[too] = dist[now] + e[i].dis;
31                if(!v[too])
32                {
33                    v[too] = 1; h[++rear] = too;
34                    if (++tim[too] > n) return 0;
35                }
36            }
37        v[now] = 0;
38    }
39
40    return 1;
41}
42
43int main()
44{
45    while (scanf("%d", &n) != EOF)
46    {
47        memset(last, 0, sizeof(last));
48        memset(dist, 0, sizeof(dist));
49        memset(tim, 0, sizeof(tim));
50        mx = -inf; mn = inf; tot = 0;
51        for (int i = 1; i <= n; i++)
52        {
53            scanf("%d%d%d", &x, &y, &z);
54            mx = max(mx, y); mn = min(mn, x);
55            add(x - 1, y, z);
56        }
57
58        mn--;
59        for (int i = mn; i < mx; i++) add(i, i + 1,
    ↪ 0), add(i + 1, i, -1);
60        if (!spfa()) puts("-1");
61        else printf("%lld\n", dist[mx]);
62    }
63}

```

## 4.5. 生成树

### 最小生成树

时间复杂度  $O(m \log n)$

```

1// 最小生成树
2#include <iostream>
3#include <cstring>
4#include <cstdlib>
5#include <cstdio>
6#include <cmath>
7#include <algorithm>
8#define ll long long
9using namespace std;
10const int maxn = 500010, inf = 1e9 + 233;
11
12struct edge{int x, y, dis;}e[maxn];
13
14int n, m;
15int fa[maxn];
16
17template <typename T>
18inline void read(T &k)
19{
20    int f = 1; k = 0; char c = getchar();
21    while (c < '0' || c > '9') c == '-' && (f = -1), c
    ↪ = getchar();
22    while (c <= '9' && c >= '0') k = k * 10 + c - '0',
    ↪ c = getchar();
23    k *= f;
24}
25
26bool operator < (edge a, edge b) {return a.dis <
    ↪ b.dis;}
27
28int gf(int x) {return fa[x] == x ? x : fa[x] =
    ↪ gf(fa[x]);}
29
30int main()
31{
32    read(n); read(m);
33    for (int i = 1; i <= m; i++)
34        read(e[i].x), read(e[i].y), read(e[i].dis);
35
36    sort(e + 1, e + 1 + m);
37    for (int i = 1; i <= n; i++) fa[i] = i;
38
39    int cnt = 0, ans = 0;
40    for (int i = 1; i <= m; i++)
41    {
42        int x = gf(e[i].x), y = gf(e[i].y);
43        if (x == y) continue;
44        cnt++; ans += e[i].dis;
45        fa[x] = y;
46    }
47
48    if (cnt != n - 1) puts("orz");
49    else printf("%d\n", ans);
50}

```

### 次小生成树

时间复杂度  $O(m \log n)$

```

1// 严格次小生成树需要维护树上路径的最大值和次大值
2// 最大值为 mx[] 次大值为 mn[]
3// 复杂度  $O(n \log n)$ 
4#include <iostream>
5#include <cstring>
6#include <cstdio>
7#include <algorithm>
8#include <cmath>
9#define ll long long
10
11using namespace std;
12const int maxn = 500010, inf = 1e9 + 233;

```



```

13
14 struct edge{int too, dis, pre;}e[maxn << 1];
15 struct poi{int x, y, dis;}p[maxn];
16 int n, m, x, y, z, tot;
17 ll mst;
18 int gfa[maxn], fa[20][maxn], last[maxn], mx[20][maxn],
    ↪ mn[20][maxn], dep[maxn];
19 bool flag[maxn];
20
21 int gf(int x) {return gfa[x] == x ? x : gfa[x] =
    ↪ gf(gfa[x]);}
22
23 inline void add(int x, int y, int z) {e[++tot] =
    ↪ (edge){y, z, last[x]}; last[x] = tot;}
24
25 inline bool cmp(poi a, poi b) {return a.dis < b.dis;}
26
27 void dfs(int x)
28 {
29     for (int i = last[x], too; i; i = e[i].pre)
30         if ((too = e[i].too) != fa[0][x])
31             {
32                 fa[0][too] = x;
33                 dep[too] = dep[x] + 1;
34                 mx[0][too] = e[i].dis;
35                 mn[0][too] = -inf;
36                 dfs(too);
37             }
38 }
39
40 int lca(int x, int y)
41 {
42     if (dep[x] < dep[y]) swap(x, y);
43     for (int i = 18; ~i; i--)
44         if (dep[fa[i][x]] >= dep[y]) x = fa[i][x];
45     for (int i = 18; ~i; i--)
46         if (fa[i][x] != fa[i][y]) x = fa[i][x], y =
            ↪ fa[i][y];
47     return x == y ? x : fa[0][x];
48 }
49
50 int query(int x, int y, int w)
51 {
52     int ans = -inf;
53     for (int i = 18; ~i; i--)
54         if (dep[fa[i][x]] >= dep[y])
55             {
56                 if (w != mx[i][x]) ans = max(ans,
                    ↪ mx[i][x]);
57                 else ans = max(ans, mn[i][x]);
58                 x = fa[i][x];
59             }
60     return ans;
61 }
62
63 int main()
64 {
65     scanf("%d%d", &n, &m);
66     for (int i = 1; i <= m; i++)
67         scanf("%d%d%d", &x, &y, &z), p[i] = (poi){x, y,
            ↪ z};
68     sort(p + 1, p + 1 + m, cmp);
69     for (int i = 1; i <= n; i++) gfa[i] = i;
70     for (int i = 1; i <= m; i++)
71     {
72         int x = gf(p[i].x), y = gf(p[i].y);
73         if (x == y) continue;
74         gfa[x] = y;
75         flag[i] = 1;

```

```

76         add(p[i].x, p[i].y, p[i].dis);
77         add(p[i].y, p[i].x, p[i].dis);
78         mst += p[i].dis;
79     }
80
81     dfs(1);
82
83     mx[0][1] = mn[0][1] = -inf; dep[1] = 1; dfs(1);
84     for (int i = 1; i < 19; i++)
85     {
86         for (int j = 1; j <= n; j++)
87             {
88                 fa[i][j] = fa[i - 1][fa[i - 1][j]];
89                 mx[i][j] = max(mx[i - 1][j], mx[i - 1][fa[i
                    ↪ - 1][j]]);
90                 mn[i][j] = max(mn[i - 1][j], mn[i - 1][fa[i
                    ↪ - 1][j]]);
91                 if (mx[i - 1][j] != mx[i - 1][fa[i -
                    ↪ 1][j]])
92                     mn[i][j] = max(mn[i][j], min(mx[i -
                    ↪ 1][j], mx[i - 1][fa[i - 1][j]]));
93             }
94     }
95
96     ll ans = 1e18;
97     for (int i = 1; i <= m; i++)
98         if (!flag[i])
99             {
100                 int x = p[i].x, y = p[i].y, f = lca(x, y);
101                 ans = min(ans, mst - max(query(x, f,
                    ↪ p[i].dis), query(y, f, p[i].dis)) +
                    ↪ p[i].dis);
102             }
103
104     printf("%lld\n", ans);
105 }

```

## 4.6. 曼哈顿最小生成树

时间复杂度  $O(n \log n)$

```

1 // 曼哈顿距离最小生成树
2 // 复杂度  $O(n \log n)$ 
3 #include <iostream>
4 #include <cstring>
5 #include <cstdio>
6 #include <algorithm>
7 #include <cmath>
8
9 using namespace std;
10 const int maxn = 500010, inf = 1e9 + 233;
11
12 struct poi{int x, y, pos;}p[maxn];
13 bool operator < (poi a, poi b) {return a.x < b.x ||
    ↪ (a.x == b.x && a.y < b.y);}
14
15 struct edge{int x, y, dis;}e[maxn << 2];
16 bool operator < (edge a, edge b) {return a.dis <
    ↪ b.dis;}
17
18 struct bit{int w, pos;}tr[maxn];
19 bit operator + (bit a, bit b) {return a.w > b.w ? b :
    ↪ a;}
20
21 int n, k, tot;
22 int fa[maxn], a[maxn], b[maxn];
23
24 int gf(int x) {return fa[x] == x ? x : fa[x] =
    ↪ gf(fa[x]);}
25

```

```

26 inline void add(int x, int y, int z) {e[++tot] =
    ↪ (edge){x, y, z};}
27
28 inline void update(int x, bit delta) {for (; x; x -= x
    ↪ & -x) tr[x] = tr[x] + delta;}
29
30 inline int query(int x, int n) {bit ans = (bit){inf,
    ↪ -1}; for (; x <= n; x += x & -x) ans = ans + tr[x];
    ↪ return ans.pos;}
31
32 inline int dis (poi a, poi b) {return abs(a.x - b.x) +
    ↪ abs(a.y - b.y);}
33
34 void solve(int n, poi *p)
35 {
36     for (int j = 0; j < 4; j++)
37     {
38         if (j == 1 || j == 3)
39         {
40             for (int i = 1; i <= n; i++)
41                 swap(p[i].x, p[i].y);
42         }
43         else if (j == 2)
44         {
45             for (int i = 1; i <= n; i++)
46                 p[i].x = -p[i].x;
47         }
48         sort(p + 1, p + 1 + n);
49         for (int i = 1; i <= n; i++)
50             a[i] = b[i] = p[i].y - p[i].x;
51         sort(b + 1, b + 1 + n);
52         int N = unique(b + 1, b + 1 + n) - b - 1;
53         for (int i = 1; i <= N; i++) tr[i] = (bit){inf,
            ↪ -1};
54         for (int i = n; i; i--)
55         {
56             a[i] = lower_bound(b + 1, b + 1 + N, a[i])
                ↪ - b;
57             int ans = query(a[i], N);
58             if (~ans) add(p[i].pos, p[ans].pos,
                ↪ dis(p[i], p[ans]));
59             update(a[i], (bit){p[i].x + p[i].y, i});
60         }
61     }
62 }
63
64 int main()
65 {
66     scanf("%d%d", &n, &k); k = n - k;
67     for (int i = 1; i <= n; i++)
68         scanf("%d%d", &p[i].x, &p[i].y), p[i].pos = i;
69     solve(n, p);
70     for (int i = 1; i <= n; i++) fa[i] = i;
71     sort(e + 1, e + 1 + tot);
72     for (int i = 1; i <= tot; i++)
73     {
74         int x = gf(e[i].x), y = gf(e[i].y);
75         if (x == y) continue;
76         fa[x] = y;
77         k--;
78         if (k == 0) return printf("%d\n", e[i].dis), 0;
79     }
80 }

```

```

4 // 以下为建树过程，两点 lca 的权值为两点之间所有简单路径
    ↪ 中最大边权的最小值
5 // sort(q + 1, q + 1 + m);
6 // for (int i = 1; i <= n; i++) fa[i] = i;
7 // for (int i = 1; i <= m; i++)
8 // {
9 //     int fx = gf(q[i].x), fy = gf(q[i].y);
10 //     if (fx == fy) continue;
11 //     add2(++tott, fx); add2(tott, fy);
12 //     fa[tott] = tott; fa[fx] = fa[fy] = tott;
13 //     w[tott] = q[i].l;
14 // }
15 #include<iostream>
16 #include<cstring>
17 #include<cstdlib>
18 #include<cstdio>
19 #include<algorithm>
20 #include<queue>
21 #define ll long long
22 using namespace std;
23 const int maxn = 1000010, inf = 1e9 + 233;
24 struct poi {int x, dis;};
25 priority_queue<poi>Q;
26 bool operator < (poi a, poi b) {return a.dis > b.dis;};
27 struct que {int x, y, l;}; q[maxn];
28 struct edge {int too, dis, pre;}; e[maxn << 1], e2[maxn <<
    ↪ 1];
29 int n, m, tot, tot2, totd, T, x, y, z, l;
30 int last[maxn], dist[maxn], f[20][maxn], last2[maxn],
    ↪ fa[maxn], mn[maxn], w[maxn];
31
32 bool operator < (que a, que b) {return a.l > b.l;};
33
34 int gf(int x){return fa[x] == x ? x : fa[x] =
    ↪ gf(fa[x]);}
35
36 inline void add(int x, int y, int z){e[++tot]=(edge){y,
    ↪ z, last[x]}; last[x]=tot;}
37
38 inline void add2(int x, int y){e2[++tot2]=(edge){y, 0,
    ↪ last2[x]}; last2[x]=tot2;}
39
40 inline void dij(int s)
41 {
42     for (int i = 1; i <= n; i++) dist[i] = inf;
43     dist[s] = 0; Q.push((poi){s, 0});
44     while (!Q.empty())
45     {
46         poi now = Q.top(); Q.pop();
47         if (now.dis != dist[now.x]) continue;
48         for (int i = last[now.x], too; i; i = e[i].pre)
49             if (dist[too = e[i].too] > dist[now.x] +
                ↪ e[i].dis)
50             {
51                 dist[too] = dist[now.x] + e[i].dis;
52                 Q.push((poi){too, dist[too]});
53             }
54     }
55 }
56
57 void dfs(int x)
58 {
59     if (x <= n) mn[x] = dist[x]; else mn[x] = inf;
60     for (int i = last2[x], too; i; i = e2[i].pre)
61         f[0][too = e2[i].too] = x, dfs(too), mn[x] =
            ↪ min(mn[x], mn[too]);
62 }
63

```

## 4.7. Kruskal 重构树

时间复杂度  $O(n \log n)$

1 // kruskal 重构树  
2 // 复杂度  $O(m \log m)$   
3 // NOI2018 归程

```

64 inline int jump(int x, int y)
65 {
66     for(int i = 19; ~i; i--)
67         if(f[i][x] && w[f[i][x]] > y) x = f[i][x];
68     return x;
69 }
70
71 inline void clear()
72 {
73     tot = tot2 = tott = 0;
74     memset(last, 0, sizeof(last));
75     memset(last2, 0, sizeof(last2));
76     memset(f, 0, sizeof(f));
77     memset(w, 0, sizeof(w));
78     memset(mn, 0, sizeof(mn));
79 }
80
81 int main()
82 {
83     scanf("%d", &T);
84     while(T--)
85     {
86         clear();
87         scanf("%d%d", &n, &m); tott = n;
88         for (int i = 1; i <= m; i++)
89         {
90             scanf("%d%d%d", &x, &y, &z, &l);
91             add(x, y, z); add(y, x, z);
92             q[i] = (que){x, y, l};
93         }
94         dij(1); sort(q + 1, q + 1 + m);
95         for (int i = 1; i <= n; i++) fa[i] = i;
96         for (int i = 1; i <= m; i++)
97         {
98             int fx = gf(q[i].x), fy = gf(q[i].y);
99             if (fx == fy) continue;
100             add2(++tott, fx); add2(tott, fy);
101             fa[tott] = tott; fa[fx] = fa[fy] = tott;
102             w[tott] = q[i].l;
103         }
104         dfs(tott);
105         for (int j = 1; j <= 19; j++)
106             for (int i = 1; i <= tott; i++)
107                 f[j][i] = f[j - 1][f[j - 1][i]];
108         int Q, K, S, v, p;
109         scanf("%d%d%d", &Q, &K, &S);
110         int ans=0;
111         while (Q--)
112         {
113             scanf("%d", &v, &p);
114             v = (v + K * ans - 1) % n + 1;
115             p = (p + K * ans) % (S + 1);
116             int x = jump(v, p);
117             printf("%d\n", ans = mn[x]);
118         }
119     }
120 }

```

## 4.8. 生成树计数

时间复杂度  $O(n^3)$

```

1 // 生成树计数
2 // 复杂度  $O(n^3)$ 
3 #include <iostream>
4 #include <cstring>
5 #include <cstdio>
6 #include <algorithm>
7 #include <cmath>
8
9 using namespace std;

```

```

10 const int maxn = 110, inf = 1e9 + 233;
11 const double eps = 1e-9;
12 int T, n, m, x, y;
13 double b[maxn][maxn], a[maxn][maxn];
14 int mp[maxn][maxn];
15
16 double det(double a[][maxn], int n)
17 {
18     int cnt = 0;
19     double ans = 1;
20     for (int i = 1; i <= n; i++)
21         for (int j = 1; j <= n; j++)
22             b[i][j] = a[i][j];
23
24     for (int i = 1; i <= n; i++)
25     {
26         if (fabs(b[i][i]) < eps)
27         {
28             int j;
29             for (j = i + 1; j <= n; j++)
30                 if (fabs(b[j][i]) > eps) break;
31             if (j == n + 1) return 0;
32             for (int k = i; k <= n; k++) swap(b[i][k],
33                 ↪ b[j][k]);
34             cnt++;
35         }
36
37         ans *= b[i][i];
38         for (int k = i + 1; k <= n; k++) b[i][k] /=
39             ↪ b[i][i];
40
41         for (int j = i + 1; j <= n; j++)
42             for (int k = i + 1; k <= n; k++)
43                 b[j][k] -= b[j][i] * b[i][k];
44     }
45
46     if (cnt & 1) ans = -ans;
47     return ans;
48 }
49
50 int main()
51 {
52     scanf("%d", &T);
53     while (T--)
54     {
55         scanf("%d%d", &n, &m);
56         memset(mp, 0, sizeof(mp));
57         for (int i = 1; i <= m; i++)
58             scanf("%d%d", &x, &y), mp[x][y] = mp[y][x]
59             ↪ = 1;
60
61         memset(a, 0, sizeof(a));
62         for (int i = 1; i <= n; i++)
63             for (int j = 1; j <= n; j++)
64                 if (i != j && mp[i][j])
65                     a[i][i]++, a[i][j] = -1;
66
67         printf("%.01f\n", det(a, n - 1));
68     }
69 }

```

## 4.9. 最小树形图

时间复杂度  $O(nm)$

```

1 //最小树形图 朱刘算法
2 // 复杂度  $O(nm)$ 
3 //1. 求最短弧集合 E;
4 //2. 判断集合 E 中有没有有向环, 如果有转步骤 3, 否则转 4;

```

```

5//3. 收缩点, 把有向环收缩成一个点, 并且对图重新构建, 包括
    ↪ 边权值的改变和点的处理, 之后再转步骤 1;
6//4. 展开收缩点, 求得最小树形图;
7#include <iostream>
8#include <cstring>
9#include <cstdio>
10#include <algorithm>
11#include <cmath>
12
13using namespace std;
14const int maxn = 500010, inf = 1e9 + 233;
15struct edge{int x, y, dis;}e[maxn];
16
17int n, m, x, y, z, root;
18int inw[maxn], col[maxn], nxt[maxn], vis[maxn];
19
20int solve()
21{
22    int ans = 0;
23    while (1)
24    {
25        for (int i = 1; i <= n; i++) inw[i] = inf;
26        for (int i = 1; i <= m; i++)
27        {
28            int x = e[i].x, y = e[i].y;
29            if (x != y && e[i].dis < inw[y]) // 对每个
                ↪ 点找最小入边
                inw[y] = e[i].dis, nxt[y] = x;
30        }
31        for (int i = 1; i <= n; i++)
32            if (i != root && inw[i] == inf) return -1;
33        int tott = 0;
34        for (int i = 1; i <= n; i++)
35            vis[i] = col[i] = 0;
36        for (int i = 1; i <= n; i++)
37        {
38            if (i == root) continue;
39            ans += inw[i];
40            int x = i;
41            while (vis[x] != i && !col[x] && x != root)
42                ↪ //找环
                vis[x] = i, x = nxt[x];
43            if (!col[x] && x != root)
44            {
45                col[x] = ++tott; //把环上点缩为同一点
46                for (int y = nxt[x]; x != y; y =
47                    ↪ nxt[y])
48                    col[y] = tott;
49            }
50        }
51        if (!tott) break;
52        for (int i = 1; i <= n; i++)
53            if (!col[i]) col[i] = ++tott;
54        // 收缩有向环
55        for (int i = 1; i <= m; i++)
56        {
57            int x = e[i].x, y = e[i].y;
58            e[i].x = col[x]; e[i].y = col[y];
59            if (col[x] != col[y]) e[i].dis -= inw[y];
60        }
61        root = col[root];
62        n = tott;
63    }
64    return ans;
65}
66
67int main()

```

```

70{
71    scanf("%d%d%d", &n, &m, &root);
72    for (int i = 1; i <= m; i++)
73        scanf("%d%d%d", &x, &y, &z), e[i] = (edge){x,
74            ↪ y, z};
75    printf("%d\n", solve());
76}

```

## 4.10. Tarjan

### 有向图强联通分量

```

1// tarjan 求强连通分量
2#include <iostream>
3#include <cstring>
4#include <cstdlib>
5#include <cstdio>
6#include <algorithm>
7using namespace std;
8
9const int maxn = 500010, inf = 1e9 + 233;
10
11struct edge{int x, too, pre;}e[maxn << 1], e2[maxn <<
12    ↪ 1];
13
14int n, m, tot, tot2, x, y, tott, top, color;
15int w[maxn], sum[maxn], mx[maxn], col[maxn], dfn[maxn],
16    ↪ low[maxn];
17int ru[maxn], st[maxn], lack[maxn], last[maxn],
18    ↪ last2[maxn];
19
20template <typename T>
21inline void read(T &k)
22{
23    int f = 1; k = 0; char c = getchar();
24    while (c < '0' || c > '9') c == '-' && (f = -1), c
25        ↪ = getchar();
26    while (c <= '9' && c >= '0') k = k * 10 + c - '0',
27        ↪ c = getchar();
28    k *= f;
29}
30
31inline void add(int x, int y) {e[++tot] = (edge){x, y,
32    ↪ last[x]}; last[x] = tot;}
33
34inline void add2(int x, int y) {e2[++tot2] = (edge){x,
35    ↪ y, last2[x]}; last2[x] = tot2;}
36
37void tarjan(int x)
38{
39    dfn[x] = low[x] = ++tott; st[++top] = x; lack[x] =
40    ↪ top;
41
42    for (int i = last[x], too; i; i = e[i].pre)
43        if (!dfn[too = e[i].too]) tarjan(too), low[x] =
44            ↪ min(low[x], low[too]);
45        else if (!col[too]) low[x] = min(low[x],
46            ↪ dfn[too]);
47
48    if (dfn[x] == low[x])
49        for (color++; top >= lack[x]; top--)
50            col[st[top]] = color;
51}
52
53inline void topsort()
54{
55    int top = 0;
56    for (int i = 1; i <= color; i++)
57        if (!ru[i]) st[++top] = i;
58}

```

```

49 while (top)
50 {
51     int now = st[top--];
52     for (int i = last2[now], too; i; i = e2[i].pre)
53     {
54         ru[too = e2[i].too]--;
55         mx[too] = max(mx[too], sum[too] + mx[now]);
56         if (!ru[too]) st[++top] = too;
57     }
58 }
59 }
60
61 int main()
62 {
63     read(n); read(m);
64     for (int i = 1; i <= n; i++) read(w[i]);
65     for (int i = 1; i <= m; i++)
66         read(x), read(y), add(x, y);
67
68     for (int i = 1; i <= n; i++)
69         if (!dfn[i]) tarjan(i);
70
71     for (int i = 1; i <= m; i++)
72         if (col[e[i].x] != col[e[i].too])
73             add2(col[e[i].x], col[e[i].too]),
74             ↪ ru[col[e[i].too]]++;
75
76     for (int i = 1; i <= n; i++) mx[col[i]] += w[i],
77     ↪ sum[col[i]] += w[i];
78
79     topsort();
80
81     int ans = 0;
82     for (int i = 1; i <= color; i++) ans = max(ans,
83     ↪ mx[i]);
84     printf("%d\n", ans);
85 }

```

### 无向图割点和桥

```

1 // tarjan 求割点
2 // 割边为 对于节点 u，若邻接点中存在一点 v 满足 low[v]
3   ↪ > dfn[u]，则 (u, v) 为割边
4 #include <iostream>
5 #include <cstring>
6 #include <cstdlib>
7 #include <cstdio>
8 #include <cmath>
9 #include <algorithm>
10 #define ll long long
11 using namespace std;
12 const int maxn = 500010, inf = 1e9 + 233;
13 struct edge{int too, pre;}e[maxn << 1];
14
15 int n, m, x, y, tot, tott;
16 int dfn[maxn], low[maxn], last[maxn];
17 bool cut[maxn];
18
19 template <typename T>
20 inline void read(T &k)
21 {
22     int f = 1; k = 0; char c = getchar();
23     while (c < '0' || c > '9') c == '-' && (f = -1), c
24     ↪ = getchar();
25     while (c <= '9' && c >= '0') k = k * 10 + c - '0',
26     ↪ c = getchar();
27     k *= f;
28 }

```

```

28 inline void add(int x, int y) {e[++tot] = (edge){y,
29   ↪ last[x]}; last[x] = tot;}
30 void tarjan(int x, int fa)
31 {
32     dfn[x] = low[x] = ++tott;
33
34     int son = 0;
35     for (int i = last[x], too; i; i = e[i].pre)
36     {
37         if (!dfn[too = e[i].too])
38         {
39             tarjan(too, fa);
40             low[x] = min(low[x], low[too]);
41             cut[x] |= (low[too] >= dfn[x] && x != fa);
42             son += x == fa;
43         }
44         low[x] = min(low[x], dfn[too]);
45     }
46
47     cut[x] |= (x == fa && son >= 2);
48 }
49
50 int main()
51 {
52     read(n); read(m);
53     for (int i = 1; i <= m; i++)
54         read(x), read(y), add(x, y), add(y, x);
55
56     for (int i = 1; i <= n; i++)
57         if (!dfn[i]) tarjan(i, i);
58
59     int ans = 0;
60     for (int i = 1; i <= n; i++) ans += cut[i];
61     printf("%d\n", ans);
62     for (int i = 1; i <= n; i++)
63         if (cut[i]) printf("%d ", i);
64 }

```

## 4.11. 支配树

### DAG 支配树

时间复杂度  $O(n \log n)$

```

1 // DAG 支配树
2 // 复杂度  $O(n \log n)$ 
3 // g2 是支配树
4 // 本题为 [ZJOI2012] 灾难：求每一个点是从起点到几个点的必
5   ↪ 经点
6 #include <iostream>
7 #include <cstring>
8 #include <cstdio>
9 #include <algorithm>
10 #include <cmath>
11
12 using namespace std;
13 const int maxn = 500010, inf = 1e9 + 233;
14 struct edge{int too, pre;}e[maxn << 4];
15 int n, m, tot, x;
16 int size[maxn], h[maxn], d[maxn], fa[maxn][17],
17   ↪ etr[maxn], bin[maxn];
18
19 int lca(int x, int y)
20 {
21     if (x < 0) return y;
22     if (d[x] < d[y]) swap(x, y);
23     int tmp = d[x] - d[y];
24     for (int i = 0; i < 17; i++)
25         if (tmp & bin[i]) x = fa[x][i];
26     for (int i = 16; ~i; i--)

```

```

25     if (fa[x][i] != fa[y][i])
26         x = fa[x][i], y = fa[y][i];
27     return x == y ? x : fa[x][0];
28 }
29
30 struct node
31 {
32     int last[maxn];
33     void add(int x, int y) {e[++tot] = (edge){y,
        ↪ last[x]}; last[x] = tot; etr[y]++;}
34     void topo()
35     {
36         int front = 0, rear = 0;
37         for (int i = 1; i <= n; i++)
38             if (!etr[i]) h[++rear] = i;
39         while (front < rear)
40         {
41             int now = h[++front];
42             for (int i = last[now], too; i; i =
        ↪ e[i].pre)
43             {
44                 etr[too = e[i].too]--;
45                 if (!etr[too]) h[++rear] = too;
46             }
47         }
48     }
49
50     void addedge(int x, int y)
51     {
52         add(x, y);
53         d[y] = d[x] + 1;
54         fa[y][0] = x;
55         for (int i = 1; i < 17; i++)
56             fa[y][i] = fa[fa[y][i - 1]][i - 1];
57     }
58
59     void dfs(int x)
60     {
61         size[x] = 1;
62         for (int i = last[x], too; i; i = e[i].pre)
63             dfs(too = e[i].too), size[x] += size[too];
64     }
65 }g1, g2;
66
67 void build()
68 {
69     bin[0] = 1;
70     for (int i = 1; i < 17; i++)
71         bin[i] = bin[i - 1] << 1;
72     for (int i = n; i; i --)
73     {
74         int x = h[i], tmp = -1;
75         for (int j = g1.last[x]; j; j = e[j].pre)
76             tmp = lca(tmp, e[j].too);
77         g2.addedge(max(tmp, 0), x);
78     }
79 }
80
81 int main()
82 {
83     scanf("%d", &n);
84     for (int i = 1; i <= n; i++)
85     {
86         scanf("%d", &x);
87         while (x) g1.add(i, x), scanf("%d", &x);
88     }
89
90     g1.topo();
91     build();

```

```

92     g2.dfs(0);
93     for (int i = 1; i <= n; i++) printf("%d\n", size[i]
        ↪ - 1);
94 }

```

### 有向图支配树

时间复杂度  $O((n + m) \log n)$

```

1 // 支配树
2 // 复杂度  $O(n \log n)$ 
3 // semi[] 为半支配点
4 // idom[x] 是 x 在支配树上的祖先
5 #include <iostream>
6 #include <cstdio>
7 #include <cstdlib>
8 #include <algorithm>
9 #include <cstring>
10 #include <vector>
11 #include <cmath>
12 #include <map>
13 #include <set>
14 #define LL long long
15 using namespace std;
16 const int maxn = 500010, inf = 1e9 + 233;
17 struct edge{int too, pre;}e[maxn << 1], e2[maxn << 1],
    ↪ e3[maxn << 1];
18 int n, m, tott, tot, tot2, tot3, x, y;
19 int last[maxn], last2[maxn], last3[maxn];
20 int dfn[maxn], semi[maxn], idom[maxn], ans[maxn],
    ↪ f[maxn], rev[maxn], fa[maxn], Mi[maxn];
21
22 inline void add1(int x, int y){e[++tot] = (edge){y,
    ↪ last[x]}; last[x] = tot;}
23
24 inline void add2(int x, int y){e2[++tot2] = (edge){y,
    ↪ last2[x]}; last2[x] = tot2;}
25
26 inline void add3(int x, int y){e3[++tot3] = (edge){y,
    ↪ last3[x]}; last3[x] = tot3;}
27
28 int gf(int x)
29 {
30     if (fa[x] == x) return x;
31     int fx = fa[x], y = gf(fa[x]);
32     if (dfn[semi[Mi[fx]]] < dfn[semi[Mi[x]]]) Mi[x] =
        ↪ Mi[fx];
33     return fa[x] = y;
34 }
35
36 void tarjan(int x)
37 {
38     dfn[x] = ++tott; rev[tott] = x;
39     for (int i = last[x], too; i; i = e[i].pre)
40         if (!dfn[too = e[i].too])
41             f[too] = x, tarjan(too);
42 }
43
44 void dfs(int x)
45 {
46     ans[x] = 1;
47     for (int i = last3[x], too; i; i = e3[i].pre)
48         dfs(too = e3[i].too), ans[x] += ans[too];
49 }
50
51 inline void calc()
52 {
53     for (int j = n; j >= 2; j--)
54     {
55         int x = rev[j], tmp = n;
56         for (int i = last2[x], too; i; i = e2[i].pre)

```

```

57     if (dfn[too] = e2[i].too)
58     {
59         if (dfn[too] < dfn[x]) tmp = min(tmp,
        ↪ dfn[too]);
60         else gf(too), tmp = min(tmp,
        ↪ dfn[semi[Mi[too]]]);
61     }
62     semi[x] = rev[tmp]; fa[x] = f[x];
63     add3(semi[x], x);
64
65     x = rev[j - 1];
66     for (int i = last3[x], too; i; i = e3[i].pre)
67     {
68         gf(too = e3[i].too);
69         if (semi[Mi[too]] == x) idom[too] = x;
70         else idom[too] = Mi[too];
71     }
72 }
73
74 for (int i = 2; i <= n; i++)
75 {
76     int x = rev[i];
77     if (idom[x] != semi[x])
78         idom[x] = idom[idom[x]];
79 }
80
81 memset(last3, 0, sizeof(last3));
82 for (int i = 2; i <= n; i++)
83     add3(idom[i], i);
84 dfs(1);
85 }
86
87 inline void init()
88 {
89     for (int i = 1; i <= n; i++)
90         fa[i] = semi[i] = Mi[i] = i;
91 }
92
93 int main()
94 {
95     scanf("%d%d", &n, &m);
96     for (int i = 1; i <= m; i++)
97     {
98         scanf("%d%d", &x, &y);
99         add1(x, y);
100        add2(y, x);
101    }
102    tarjan(1);
103    init();
104    calc();
105    for (int i = 1; i <= n; i++)
106        printf("%d ", ans[i]);
107 }

```

## 4.12. 二分图匹配

### 匈牙利

时间复杂度  $O(nm)$

```

1 // 二分图最大匹配
2 // 复杂度  $O(nm)$ 
3 #include <iostream>
4 #include <cstring>
5 #include <cstdlib>
6 #include <cstdio>
7 #include <cmath>
8 #include <algorithm>
9 #define ll long long
10 using namespace std;
11 const int maxn = 500010, inf = 1e9 + 233;
12

```

```

13 struct edge{int too, pre;}e[maxn];
14 int n1, n2, m, tot, ans, x, y, T;
15 int last[maxn], lik[maxn], v[maxn];
16
17 template <typename T>
18 inline void read(T &k)
19 {
20     int f = 1; k = 0; char c = getchar();
21     while (c < '0' || c > '9') c == '-' && (f = -1), c
        ↪ = getchar();
22     while (c <= '9' && c >= '0') k = k * 10 + c - '0',
        ↪ c = getchar();
23     k *= f;
24 }
25
26 inline void add(int x, int y)
27 {
28     if (x > n1 || y > n2) return;
29     e[++tot] = (edge){y, last[x]}; last[x] = tot;
30 }
31
32 int dfs(int x)
33 {
34     for (int i = last[x], too; i; i = e[i].pre)
35         if (v[too = e[i].too] != T)
36         {
37             v[too] = T;
38             if (!lik[too] || dfs(lik[too]))
39             {
40                 lik[too] = x;
41                 return 1;
42             }
43         }
44     return 0;
45 }
46
47 int main()
48 {
49     read(n1); read(n2); read(m);
50     for (int i = 1; i <= m; i++)
51         read(x), read(y), add(x, y);
52
53     for (int i = 1; i <= n1; i++) ++T, ans += dfs(i);
54
55     printf("%d\n", ans);
56 }
57 }

```

### HK

时间复杂度  $O(\sqrt{n} * m)$

```

1 // HK
2 // 复杂度  $O(m\sqrt{n})$ 
3 #include <iostream>
4 #include <cstring>
5 #include <cstdio>
6 #include <algorithm>
7 #include <cmath>
8
9 using namespace std;
10 namespace Hopcroft
11 {
12     const int N = 5010, M = 1000010; // 最大的单侧点个
        ↪ 数
13     int cnt, pos[N], neg[N]; // pos[] 为左侧点所匹配到
        ↪ 的右侧点编号, 从 0 开始
14     // neg[] 反之, 没有匹配到对应的点则为 -1
15     // 传入左侧点个数 n 和左侧点至右侧点的边表 e[], 返回
        ↪ 匹配点的数量 cnt

```

```

16 int lx[N], ly[N], q[N], n, g[N], v[M], nxt[M], ed;
17 void init(int _n) {n = _n; for (int i = ed = 0; i <
    ↪ n; i++) g[i] = 0;}
18 void add(int x, int y) {v[++ed] = y; nxt[ed] =
    ↪ g[x]; g[x] = ed;}
19 bool dfs(int x)
20 {
21     int c = lx[x] + 1, y = lx[x] = -1;
22     for (int i = g[x]; i; i = nxt[i])
23         if (ly[y = v[i]] == c)
24         {
25             ly[y] = -1;
26             if (~neg[y] && !dfs(neg[y])) continue;
27             pos[neg[y] = x] = y;
28             return 1;
29         }
30     return 0;
31 }
32
33 int work()
34 {
35     int i, x, y;
36     fill(pos, pos + n, -1); fill(neg, neg + n, -1);
37     for (x = cnt = 0; x < n; x++)
38         for (i = g[x]; i; i = nxt[i])
39         {
40             if (~neg[y = v[i]]) continue;
41             pos[neg[y] = x] = y;
42             cnt++; break;
43         }
44     while (1)
45     {
46         int h = 0, t = 0, ok = 0;
47         fill(lx, lx + n, -1); fill(ly, ly + n, -1);
48         for (x = 0; x < n; x++)
49             if (pos[x] < 0) lx[q[t++] = x] = 0;
50         while (h != t)
51         {
52             for (i = g[x = q[h++]]; i; i = nxt[i])
53             {
54                 if (~ly[y = v[i]]) continue;
55                 ly[y] = 1 + lx[x];
56                 if (!neg[y] && !lx[neg[y]])
57                     ↪ continue;
58                 if (~neg[y])
59                     lx[q[t++] = neg[y]] = 1 +
60                     ↪ ly[y];
61                 else ok = 1;
62             }
63             if (!ok) return cnt;
64             for (x = 0; x < n; x++)
65                 if (pos[x] < 0 && dfs(x)) cnt++;
66         }
67     }
68
69 int n, m, e, x, y;
70 int main()
71 {
72     scanf("%d%d%d", &n, &m, &e);
73     Hopcroft::init(n + m + 1);
74     for (int i = 1; i <= e; i++)
75     {
76         scanf("%d%d", &x, &y);
77         if (x <= n && y <= m) Hopcroft::add(x, n + y);
78     }
79     printf("%d\n", Hopcroft::work());
80 }

```

## 二分图最大权匹配 Kuhn-Munkres

时间复杂度  $O(n^3)$ 

```

1 // 复杂度  $O(n^3)$ 
2 #include <iostream>
3 #include <cstring>
4 #include <cstdio>
5 #include <algorithm>
6 #include <cmath>
7 using namespace std;
8 typedef long long ll;
9 const int maxn = 510, inf = 1e9 + 233;
10 int n, nl, nr, m, x, y, z;
11 int g[maxn][maxn];
12
13 namespace KM
14 {
15     int left[maxn], right[maxn];
16     int visl[maxn], visr[maxn];
17     int lx[maxn], ly[maxn], slack[maxn];
18
19     bool augment (int x)
20     {
21         visl[x] = 1;
22         for (int y = 1; y <= n; y++)
23         {
24             if (visr[y]) continue;
25             int slk = lx[x] + ly[y] - g[x][y];
26             if (!slk)
27             {
28                 visr[y] = 1;
29                 if (!right[y] || augment(right[y]))
30                 {
31                     right[y] = x; left[x] = y;
32                     return 1;
33                 }
34             }
35             else slack[y] = min(slack[y], slk);
36         }
37         return 0;
38     }
39
40     void solve()
41     {
42         for (int i = 1; i <= n; i++)
43             for (int j = 1; j <= n; j++)
44                 ly[j] = max(ly[j], g[i][j]);
45
46         for (int i = 1; i <= n; i++)
47         {
48             for (int j = 1; j <= n; j++)
49                 visl[j] = visr[j] = 0, slack[j] = inf;
50             if (augment(i)) continue;
51             while (1)
52             {
53                 int d = inf, x;
54                 for (int j = 1; j <= n; j++)
55                     if (!visr[j])
56                         d = min(d, slack[j]);
57                 for (int j = 1; j <= n; j++)
58                 {
59                     if (!visr[j])
60                     {
61                         ly[j] -= d;
62                         slack[j] -= d;
63                         if (!slack[j]) x = j;
64                     }
65                     if (!visl[j])

```



```

66         lx[j] += d;
67     }
68
69     if (!right[x]) break;
70     visr[x] = 1; visl[right[x]] = 1;
71     x = right[x];
72     for (int y = 1; y <= n; y++)
73         slack[y] = min(slack[y], lx[x] + ly[y]
74             ↪ - g[x][y]);
75
76     for (int j = 1; j <= n; j++)
77         visl[j] = visr[j] = 0;
78     augment(i);
79 }
80 }
81
82 void answer()
83 {
84     ll ans = 0;
85     for (int i = 1; i <= n; i++)
86         ans += lx[i] + ly[i];
87     printf("%lld\n", ans);
88     for (int i = 1; i <= n1; i++)
89         printf("%d ", g[i][left[i]] ? left[i] : 0);
90     puts("");
91 }
92 }
93
94 int main()
95 {
96     scanf("%d%d%d", &n1, &nr, &m);
97     n = max(n1, nr);
98
99     while (m--)
100         scanf("%d%d%d", &x, &y, &z), g[x][y] =
101             ↪ max(g[x][y], z);
102
103     KM::solve();
104     KM::answer();
105     return 0;
106 }

```

### 二分图关键点 / 边判定

1 // 二分图关键边 / 点的求法为依次删除每个边每个点重新跑最  
 ↪ 大匹配检查最大匹配是否减少

```

2 #include <iostream>
3 #include <cstring>
4 #include <cstdlib>
5 #include <cstdio>
6 #include <cmath>
7 #include <algorithm>
8 #define ll long long
9 using namespace std;
10 const int maxn = 500010, inf = 1e9 + 233;
11
12 struct edge{int too, pre;}e[maxn];
13 int n1, n2, m, tot, ans, x, y, T;
14 int last[maxn], lik[maxn], v[maxn];
15 bool vis[maxn];
16
17 inline void add(int x, int y)
18 {
19     if (x > n1 || y > n2) return;
20     e[++tot] = (edge){y, last[x]}; last[x] = tot;
21 }
22
23 int dfs(int x)
24 {

```

```

25     for (int i = last[x], too; i; i = e[i].pre)
26         if (!vis[i] && v[too = e[i].too] != T)
27         {
28             v[too] = T;
29             if (!lik[too] || dfs(lik[too]))
30             {
31                 lik[too] = x;
32                 return 1;
33             }
34         }
35
36     return 0;
37 }
38
39 int hungary()
40 {
41     memset(lik, 0, sizeof(lik));
42     int ans = 0;
43     for (int i = 1; i <= n1; i++)
44         ++T, ans += dfs(i);
45     return ans;
46 }
47
48 inline void clear()
49 {
50     tot = 0;
51     memset(last, 0, sizeof(last));
52 }
53
54 int main()
55 {
56     int TT = 0;
57     while (scanf("%d%d%d", &n1, &n2, &m) != EOF)
58     {
59         clear();
60         TT++;
61         for (int i = 1; i <= m; i++)
62         {
63             scanf("%d%d", &x, &y);
64             add(x, y);
65         }
66
67         int ans = hungary(), ans2 = 0;
68
69         for (int i = 1; i <= tot; i++)
70         {
71             vis[i] = 1;
72             int now = hungary();
73             if (now < ans) ans2++;
74             vis[i] = 0;
75         }
76         printf("Board %d have %d important blanks for
77             ↪ %d chessmen.\n", TT, ans2, ans);
78     }
79 }

```

## 4.13. 一般图匹配

### 带花树

时间复杂度  $O(nm)$

```

1 // 一般图最大匹配 带花树 uoj
2 // 复杂度  $O(nm)$ 
3 #include <iostream>
4 #include <cstring>
5 #include <cstdio>
6 #include <algorithm>
7 #include <cmath>
8
9 using namespace std;

```

```

10 const int maxn = 510, inf = 1e9 + 233;
11 struct edge{int too, pre;}e[500010];
12 int n, m, x, y, tot, front, rear, tim;
13 int pre[maxn], lik[maxn], fa[maxn], last[maxn],
    ↪ st[maxn], vis[maxn], h[1000010];
14
15 inline void add(int x, int y) {e[++tot] = (edge){y,
    ↪ last[x]}; last[x] = tot;}
16
17 int gf(int x) {return fa[x] == x ? x : fa[x] =
    ↪ gf(fa[x]);}
18
19 int lca(int x, int y)
20 {
21     tim++; x = gf(x); y = gf(y);
22     while (1)
23     {
24         if (x)
25         {
26             if (vis[x] == tim) return x;
27             vis[x] = tim; x = gf(pre[lik[x]]);
28         }
29         swap(x, y);
30     }
31 }
32
33 void blossom(int x, int y, int g)
34 {
35     while (gf(x) != g)
36     {
37         pre[x] = y;
38         if (st[lik[x]] == 1) st[h[++rear] = lik[x]] =
            ↪ 0;
39         if (gf(x) == x) fa[x] = g;
40         if (gf(lik[x]) == lik[x]) fa[lik[x]] = g;
41         x = pre[y = lik[x]];
42     }
43 }
44
45 int find(int s)
46 {
47     memset(st, -1, sizeof(st));
48     memset(pre, 0, sizeof(pre));
49     for (int i = 1; i <= n; i++) fa[i] = i;
50     front = 0; rear = 1; h[1] = s; st[s] = 0;
51     int las;
52     while (front < rear)
53     {
54         int now = h[++front];
55         for (int i = last[now], too; i; i = e[i].pre)
56             if (st[too = e[i].too] == -1)
57             {
58                 pre[too] = now; st[too] = 1;
59                 if (!lik[too])
60                 {
61                     while (now)
62                     {
63                         las = lik[now];
64                         lik[now] = too;
65                         lik[too] = now;
66                         now = pre[too = las];
67                     }
68                     return 1;
69                 }
70                 st[h[++rear] = lik[too]] = 0;
71             }
72         else if (fa[too] != fa[now] && !st[too])
73             las = lca(too, now), blossom(too, now,
            ↪ las), blossom(now, too, las);

```

```

74     }
75     return 0;
76 }
77
78
79
80 int main()
81 {
82     scanf("%d%d", &n, &m);
83     int ans = 0;
84     for (int i = 1; i <= m; i++)
85     {
86         scanf("%d%d", &x, &y);
87         add(x, y); add(y, x);
88         if (!lik[x] && !lik[y]) lik[x] = y, lik[y] = x,
            ↪ ans++;
89     }
90     for (int i = 1; i <= n; i++)
91         if (!lik[i]) ans += find(i);
92     printf("%d\n", ans);
93     for (int i = 1; i <= n; i++)
94         printf("%d ", lik[i]);
95 }

```

## 一般图最大权匹配

```

1 // 一般图最大权匹配
2 // Claris
3 #include<bits/stdc++.h>
4 #define DIST(e) (lab[e.u]+lab[e.v]-g[e.u][e.v].w*2)
5 using namespace std;
6 typedef long long ll;
7 const int N=1023,INF=1e9;
8 struct Edge{
9     int u,v,w;
10 } g[N][N];
11 int
    ↪ n,m,n_x,lab[N],match[N],slack[N],st[N],pa[N],flower_from[N];
12 vector<int> flower[N];
13 deque<int> q;
14 void update_slack(int u,int x){
15     if(!slack[x]||DIST(g[u][x])<DIST(g[slack[x]][x]))
        ↪ slack[x]=u;
16 }
17 void set_slack(int x){
18     slack[x]=0;
19     for(int u=1; u<=n; ++u)
20         if(g[u][x].w>0&&st[u]!=x&&S[st[u]]==0)
            ↪ update_slack(u,x);
21 }
22 void q_push(int x){
23     if(x<=n)return q.push_back(x);
24     for(int i=0; i<flower[x].size();
        ↪ ++i)q_push(flower[x][i]);
25 }
26 void set_st(int x,int b){
27     st[x]=b;
28     if(x<=n)return;
29     for(int i=0; i<flower[x].size();
        ↪ ++i)set_st(flower[x][i],b);
30 }
31 int get_pr(int b,int xr){
32     int pr=find(flower[b].begin(),flower[b].end(),xr)
        ↪ -flower[b].begin();
33     if(pr%2==1){
34         reverse(flower[b].begin()+1,flower[b].end());
35         return (int)flower[b].size()-pr;
36     }
37     else return pr;
38 }

```

```

39 void set_match(int u,int v){
40     match[u]=g[u][v].v;
41     if(u<=n)return;
42     Edge e=g[u][v];
43     int xr=flower_from[u][e.u],pr=get_pr(u,xr);
44     for(int i=0; i<pr;
45         ↪ ++i)set_match(flower[u][i],flower[u][i^1]);
46     set_match(xr,v);
47     rotate(flower[u].begin(),flower[u].begin()
48         ↪ +pr,flower[u].end());
49 }
50 void augment(int u,int v){
51     int xnv=st[match[u]];
52     set_match(u,v);
53     if(!xnv)return;
54     set_match(xnv,st[pa[xnv]]);
55     augment(st[pa[xnv]],xnv);
56 }
57 int get_lca(int u,int v){
58     static int t=0;
59     for(++t; u||v; swap(u,v)){
60         if(u==0)continue;
61         if(vis[u]==t)return u;
62         vis[u]=t;
63         u=st[match[u]];
64         if(u)u=st[pa[u]];
65     }
66     return 0;
67 }
68 void add_blossom(int u,int lca,int v){
69     int b=n+1;
70     while(b<=n_x&&st[b])++b;
71     if(b>n_x)++n_x;
72     lab[b]=0,S[b]=0;
73     match[b]=match[lca];
74     flower[b].clear();
75     flower[b].push_back(lca);
76     for(int x=u,y; x!=lca; x=st[pa[y]])
77         flower[b].push_back(x),flower[b].push_back(
78             ↪ y=st[match[x]]),q_push(y);
79     reverse(flower[b].begin()+1,flower[b].end());
80     for(int x=v,y; x!=lca; x=st[pa[y]])
81         flower[b].push_back(x),flower[b].push_back(
82             ↪ y=st[match[x]]),q_push(y);
83     set_st(b,b);
84     for(int x=1; x<=n_x; ++x)g[b][x].w=g[x][b].w=0;
85     for(int x=1; x<=n; ++x)flower_from[b][x]=0;
86     for(int i=0; i<flower[b].size(); ++i){
87         int xs=flower[b][i];
88         for(int x=1; x<=n_x; ++x)
89             if(g[b][x].w==0||DIST(g[xs][x])<DIST(g[b][x]))
90                 g[b][x]=g[xs][x],g[x][b]=g[x][xs];
91         for(int x=1; x<=n; ++x)
92             if(flower_from[xs][x])flower_from[b][x]=xs;
93     }
94     set_slack(b);
95 }
96 void expand_blossom(int b){
97     for(int i=0; i<flower[b].size(); ++i)
98         set_st(flower[b][i],flower[b][i]);
99     int xr=flower_from[b][g[b][pa[b]].u],pr=get_pr(b,xr);
100     for(int i=0; i<pr; i+=2){
101         int xs=flower[b][i],xns=flower[b][i+1];
102         pa[xs]=g[xns][xs].u;
103         S[xs]=1,S[xns]=0;
104         slack[xs]=0,set_slack(xns);
105         q_push(xns);
106     }
107     S[xr]=1,pa[xr]=pa[b];
108     for(int i=pr+1; i<flower[b].size(); ++i){
109         int xs=flower[b][i];
110         S[xs]=-1,set_slack(xs);
111     }
112     st[b]=0;
113 }
114 bool on_found_Edge(const Edge &e){
115     int u=st[e.u],v=st[e.v];
116     if(S[v]==-1){
117         pa[v]=e.u,S[v]=1;
118         int nu=st[match[v]];
119         slack[v]=slack[nu]=0;
120         S[nu]=0,q_push(nu);
121     }
122     else if(S[v]==0){
123         int lca=get_lca(u,v);
124         if(!lca)return augment(u,v),augment(v,u),1;
125         else add_blossom(u,lca,v);
126     }
127     return 0;
128 }
129 bool matching(){
130     fill(S,S+n_x+1,-1),fill(slack,slack+n_x+1,0);
131     q.clear();
132     for(int x=1; x<=n_x; ++x)
133         if(st[x]==x&&!match[x])pa[x]=0,S[x]=0,q_push(x);
134     if(q.empty())return 0;
135     for(;;){
136         while(q.size()){
137             int u=q.front();
138             q.pop_front();
139             if(S[st[u]]==1)continue;
140             for(int v=1; v<=n; ++v)
141                 if(g[u][v].w>0&&st[u]!=st[v]){
142                     if(DIST(g[u][v])==0){
143                         if(on_found_Edge(g[u][v]))return 1;
144                     }
145                     else update_slack(u,st[v]);
146                 }
147         }
148         int d=INF;
149         for(int b=n+1; b<=n_x; ++b)
150             if(st[b]==b&&S[b]==1)d=min(d,lab[b]/2);
151         for(int x=1; x<=n_x; ++x)
152             if(st[x]==x&&slack[x]){
153                 if(S[x]==-1)d=min(d,DIST(g[slack[x]][x]));
154                 else if(S[x]==0)d=min(d,DIST(g[slack[x]][x])
155                     ↪ /2);
156             }
157         for(int u=1; u<=n; ++u){
158             if(S[st[u]]==0){
159                 if(lab[u]<=d)return 0;
160                 lab[u]-=d;
161             }
162             else if(S[st[u]]==1)lab[u]+=d;
163         }
164         for(int b=n+1; b<=n_x; ++b)
165             if(st[b]==b){
166                 if(S[st[b]]==0)lab[b]+=d*2;
167                 else if(S[st[b]]==1)lab[b]-=d*2;
168             }
169         q.clear();
170         for(int x=1; x<=n_x; ++x)
171             if(st[x]==x&&slack[x]&&st[slack[x]]!=x&&DIST(
172                 ↪ g[slack[x]][x])==0)
173                 if(on_found_Edge(g[slack[x]][x]))return 1;
174         for(int b=n+1; b<=n_x; ++b)

```

```

169     if(st[b]==b&&S[b]==1&&lab[b]==0)expand_blossom(
    ↪ b);
170 }
171 return 0;
172 }
173 pair<ll,int> weight_blossom(){
174     fill(match,match+n+1,0);
175     n_x=n;
176     int n_matches=0;
177     ll tot_weight=0;
178     for(int u=0; u<=n; ++u)st[u]=u,flower[u].clear();
179     int w_max=0;
180     for(int u=1; u<=n; ++u)
181         for(int v=1; v<=n; ++v){
182             flower_from[u][v]=(u==v?u:0);
183             w_max=max(w_max,g[u][v].w);
184         }
185     for(int u=1; u<=n; ++u)lab[u]=w_max;
186     while(matching())++n_matches;
187     for(int u=1; u<=n; ++u)
188         if(match[u]&&match[u]<u)
189             tot_weight+=g[u][match[u]].w;
190     return make_pair(tot_weight,n_matches);
191 }
192 int main(){
193     cin>>n>>m;
194     for(int u=1; u<=n; ++u)
195         for(int v=1; v<=n; ++v)
196             g[u][v]=Edge {u,v,0};
197     for(int i=0,u,v,w; i<m; ++i){
198         cin>>u>>v>>w;
199         g[u][v].w=g[v][u].w=w;
200     }
201     cout<<weight_blossom().first<<'\n';
202     for(int u=1; u<=n; ++u)cout<<match[u]<<' ';
203 }

```

## 4.14. 网络流

### 最大流

时间复杂度  $O(n^2 * m)$

```

1 // 二分图最大匹配
2 // 复杂度  $O(n\sqrt{n})$ 
3 #include <iostream>
4 #include <cstring>
5 #include <cstdlib>
6 #include <cstdio>
7 #include <cmath>
8 #include <algorithm>
9 #define ll long long
10 using namespace std;
11 const int maxn = 500010, inf = 1e9 + 233;
12
13 struct edge{int too, cf, pre;}e[maxn << 1];
14
15 int tot = 1, n, m, s, t, x, y, z, n1, n2;
16 int last[maxn], h[maxn], dist[maxn], cur[maxn];
17
18 template <typename T>
19 inline void read(T &k)
20 {
21     int f = 1; k = 0; char c = getchar();
22     while (c < '0' || c > '9') c == '-' && (f = -1), c
    ↪ = getchar();
23     while (c <= '9' && c >= '0') k = k * 10 + c - '0',
    ↪ c = getchar();
24     k *= f;
25 }
26

```

```

27 inline void add(int x, int y, int z) {e[++tot] =
    ↪ (edge){y, z, last[x]}; last[x] = tot;}
28
29 inline void ins(int x, int y, int z) {add(x, y, z);
    ↪ add(y, x, 0);}
30
31 bool bfs()
32 {
33     int front = 0, rear = 1;
34     memset(dist, -1, sizeof(dist)); h[1] = s; dist[s] =
    ↪ 0;
35     while (front != rear)
36     {
37         int now = h[++front];
38         for (int i = last[now], too; i; i = e[i].pre)
39             if (dist[too = e[i].too] == -1 && e[i].cf)
40             {
41                 dist[too] = dist[now] + 1;
42                 if (too == t) return 1;
43                 h[++rear] = too;
44             }
45     }
46     return 0;
47 }
48
49 int dfs(int x, int f)
50 {
51     if (x == t || !f) return f;
52
53     int flow = 0;
54     for (int &i = last[x], too; i; i = e[i].pre)
55         if (dist[too = e[i].too] == dist[x] + 1 &&
    ↪ e[i].cf)
56         {
57             int tmp = dfs(too, min(f - flow, e[i].cf));
58             e[i].cf -= tmp; e[i ^ 1].cf += tmp; flow +=
    ↪ tmp;
59             if (flow == f) break;
60         }
61
62     return flow;
63 }
64
65 inline int dinic()
66 {
67     int ans = 0;
68     memcpy(cur, last, sizeof(last));
69     while (bfs())
70     {
71         ans += dfs(s, inf);
72         memcpy(last, cur, sizeof(cur));
73     }
74     return ans;
75 }
76
77 int main()
78 {
79     read(n1); read(n2); read(m);
80     s = n1 + n2 + 1; t = n1 + n2 + 2;
81     for (int i = 1; i <= n1; i++)
82         ins(s, i, 1);
83     for (int i = 1; i <= n2; i++)
84         ins(i + n1, t, 1);
85     for (int i = 1; i <= m; i++)
86         read(x), read(y), ins(x, y + n1, 1);
87
88     printf("%d\n", dinic());
89 }

```

## 费用流 spfa

```

1 // 最小费用最大流 spfa
2 // 复杂度  $O(\text{Flow} * kE)$ 
3 #include <iostream>
4 #include <cstring>
5 #include <cstdlib>
6 #include <cstdio>
7 #include <cmath>
8 #include <algorithm>
9 #define ll long long
10 using namespace std;
11 const int maxn = 5010, maxm = 50010, inf = 1e9 + 233;
12 struct edge{int too, cf, dis, pre;}e[maxn << 1];
13 int n, m, s, t, x, y, z, l, tot = 1;
14 ll cost;
15 int h[maxn], dist[maxn], last[maxn], cur[maxn];
16 bool vis[maxn], vis2[maxn];
17
18 template <typename T>
19 inline void read(T &k)
20 {
21     int f = 1; k = 0; char c = getchar();
22     while (c < '0' || c > '9') c == '-' && (f = -1), c
        ↪ = getchar();
23     while (c <= '9' && c >= '0') k = k * 10 + c - '0',
        ↪ c = getchar();
24     k *= f;
25 }
26
27 inline void add(int x, int y, int z, int l) {e[++tot] =
    ↪ (edge){y, z, l, last[x]}; last[x] = tot;}
28
29 inline void ins(int x, int y, int z, int l) {add(x, y,
    ↪ z, l); add(y, x, 0, -l);}
30
31 inline bool spfa()
32 {
33     memset(dist, 0x3f, sizeof(dist));
34     memset(vis, 0, sizeof(vis));
35     vis[s] = 1; dist[s] = 0;
36
37     int front = 0, rear = 1; h[1] = s;
38     while (front != rear)
39     {
40         int now = h[++front];
41         for (int i = last[now], too; i; i = e[i].pre)
42             if (dist[too = e[i].too] > dist[now] +
                ↪ e[i].dis && e[i].cf)
43             {
44                 dist[too] = dist[now] + e[i].dis;
45                 if (!vis[too]) vis[too] = 1, h[++rear]
                    ↪ = too;
46             }
47         vis[now] = 0;
48     }
49
50     return dist[t] < inf;
51 }
52
53 int dfs(int x, int f)
54 {
55     if (x == t || !f) return f;
56
57     int flow = 0; vis2[x] = 1;
58     for (int &i = cur[x], too; i; i = e[i].pre)
59         if (dist[too = e[i].too] == dist[x] + e[i].dis
            ↪ && e[i].cf && !vis2[too])
60         {
61             int tmp = dfs(too, min(f - flow, e[i].cf));

```

```

62             e[i].cf -= tmp; e[i ^ 1].cf += tmp; flow +=
                ↪ tmp;
63             cost += 1ll * tmp * e[i].dis;
64             if (f == flow) break;
65         }
66     vis2[x] = 0;
67
68     return flow;
69 }
70
71 inline void dinic()
72 {
73     int ans = 0;
74     while (spfa())
75     {
76         memcpy(cur, last, sizeof(last));
77         ans += dfs(s, inf);
78     }
79
80     printf("%d %lld\n", ans, cost);
81 }
82
83 int main()
84 {
85     read(n); read(m); read(s); read(t);
86     for (int i = 1; i <= m; i++)
87     {
88         read(x); read(y); read(z); read(l);
89         ins(x, y, z, l);
90     }
91
92     dinic();
93 }

```

## 费用流 dijkstra

```

1 // 最小费用最大流 dij
2 // 复杂度  $O(\text{Flow} \times E \log V)$ 
3 #include <iostream>
4 #include <cstring>
5 #include <cstdlib>
6 #include <cstdio>
7 #include <cmath>
8 #include <queue>
9 #include <algorithm>
10 #define ll long long
11 using namespace std;
12 const int maxn = 5010, maxm = 50010, inf = 1e9 + 233;
13 struct edge{int too, cf, dis, pre;}e[maxn << 1];
14 struct poi{int x, dis;};
15 priority_queue<poi>q;
16 bool operator < (poi a, poi b) {return a.dis > b.dis;}
17 int n, m, s, t, x, y, z, l, tot = 1;
18 ll cost;
19 int dist[maxn], last[maxn], cur[maxn];
20 bool vis2[maxn];
21
22 template <typename T>
23 inline void read(T &k)
24 {
25     int f = 1; k = 0; char c = getchar();
26     while (c < '0' || c > '9') c == '-' && (f = -1), c
        ↪ = getchar();
27     while (c <= '9' && c >= '0') k = k * 10 + c - '0',
        ↪ c = getchar();
28     k *= f;
29 }
30
31 inline void add(int x, int y, int z, int l) {e[++tot] =
    ↪ (edge){y, z, l, last[x]}; last[x] = tot;}

```

```

32
33 inline void ins(int x, int y, int z, int l) {add(x, y,
    ↪ z, l); add(y, x, 0, -l);}
34
35 inline bool dij()
36 {
37     memset(dist, 0x3f, sizeof(dist));
38     dist[s] = 0; q.push((poi){s, 0});
39
40     while (!q.empty())
41     {
42         poi now = q.top(); q.pop();
43         if (dist[now.x] != now.dis) continue;
44         for (int i = last[now.x], too; i; i = e[i].pre)
45             if (e[i].cf && dist[too = e[i].too] >
    ↪ now.dis + e[i].dis)
46             {
47                 dist[too] = now.dis + e[i].dis;
48                 q.push((poi){too, dist[too]});
49             }
50     }
51     return dist[t] < inf;
52 }
53
54 int dfs(int x, int f)
55 {
56     if (x == t || !f) return f;
57
58     int flow = 0; vis2[x] = 1;
59     for (int &i = cur[x], too; i; i = e[i].pre)
60         if (dist[too = e[i].too] == dist[x] + e[i].dis
    ↪ && e[i].cf && !vis2[too])
61         {
62             int tmp = dfs(too, min(f - flow, e[i].cf));
63             e[i].cf -= tmp; e[i ^ 1].cf += tmp; flow +=
    ↪ tmp;
64             cost += 1ll * tmp * e[i].dis;
65             if (f == flow) break;
66         }
67     vis2[x] = 0;
68     return flow;
69 }
70 }
71
72 inline void dinic()
73 {
74     int ans = 0;
75     while (dij())
76     {
77         memcpy(cur, last, sizeof(last));
78         ans += dfs(s, inf);
79     }
80
81     printf("%d %lld\n", ans, cost);
82 }
83
84 int main()
85 {
86     read(n); read(m); read(s); read(t);
87     for (int i = 1; i <= m; i++)
88     {
89         read(x); read(y); read(z); read(l);
90         ins(x, y, z, l);
91     }
92
93     dinic();
94 }

```

## 全局最小割

时间复杂度  $O(n^3)$ 

```

1 // 全局最小割
2 // 至少去掉多少条边可以把图分成两个不相连的子图
3 // 复杂度  $O(n^3)$ 
4 #include <iostream>
5 #include <cstring>
6 #include <cstdio>
7 #include <algorithm>
8 #include <cmath>
9
10 using namespace std;
11 const int maxn = 110, inf = 1e9 + 233;
12 int n, m, x, y, z;
13 int mp[maxn][maxn], dis[maxn], vis[maxn], p[maxn];
14
15
16 int s_w()
17 {
18     int ans = inf;
19     for (int i = 1; i <= n; i++) p[i] = i;
20     while (n > 1)
21     {
22         int k, pre = 1;
23         memset(vis, 0, sizeof(vis));
24         memset(dis, 0, sizeof(dis));
25         vis[p[pre]] = 1;
26         for (int i = 2; i <= n; i++)
27         {
28             k = -1;
29             for (int j = 2; j <= n; j++)
30                 if (!vis[p[j]])
31                 {
32                     dis[p[j]] += mp[p[pre]][p[j]];
33                     if (k == -1 || dis[p[k]] <
    ↪ dis[p[j]]) k = j;
34                 }
35             vis[p[k]] = 1;
36             if (i == n)
37             {
38                 ans = min(ans, dis[p[k]]);
39                 for (int j = 1; j <= n; j++)
40                 {
41                     mp[p[pre]][p[j]] += mp[p[j]][p[k]];
42                     mp[p[j]][p[pre]] += mp[p[j]][p[k]];
43                 }
44                 p[k] = p[n--];
45             }
46             pre = k;
47         }
48     }
49     return ans;
50 }
51
52 int main()
53 {
54     while (scanf("%d%d", &n, &m) != EOF)
55     {
56         memset(mp, 0, sizeof(mp));
57         for (int i = 1; i <= m; i++)
58         {
59             scanf("%d%d%d", &x, &y, &z);
60             x++; y++;
61             mp[x][y] += z; mp[y][x] += z;
62         }
63         printf("%d\n", s_w());
64     }
65 }

```

66 }

## 最小割树

时间复杂度  $O(n^3 * m)$ 

```

1 // 最小割树
2 // 复杂度  $O(n^3 \times m)$ 
3 // build() 为建树
4 // query(x, y) 查询 x 到 y 的最小割
5 #include <iostream>
6 #include <cstring>
7 #include <cstdlib>
8 #include <cstdio>
9 #include <cmath>
10 #include <algorithm>
11 #define ll long long
12 using namespace std;
13 const int maxn = 510, inf = 1e9 + 233;
14
15 struct edge {int too, cf, pre, c;} e[500010], e2[500010];
16
17 int tot = 1, n, m, s, t, x, y, z, tot2, Q, now;
18 int last[maxn], h[maxn], dist[maxn], cur[maxn],
    ↪ last2[maxn], dep[maxn];
19 int fa[10][maxn], mn[10][maxn], pdx[maxn], tdx[maxn],
    ↪ col[maxn];
20
21 template <typename T>
22 inline void read(T &k)
23 {
24     int f = 1; k = 0; char c = getchar();
25     while (c < '0' || c > '9') c == '-' && (f = -1), c
    ↪ = getchar();
26     while (c <= '9' && c >= '0') k = k * 10 + c - '0',
    ↪ c = getchar();
27     k *= f;
28 }
29
30 inline void add(int x, int y, int z) {e[++tot] =
    ↪ (edge){y, z, last[x], z}; last[x] = tot;}
31
32 inline void add2(int x, int y, int z) {e2[++tot2] =
    ↪ (edge){y, z, last2[x], 0}; last2[x] = tot2;}
33
34 bool bfs()
35 {
36     int front = 0, rear = 1;
37     memset(dist, -1, sizeof(dist)); h[1] = s; dist[s] =
    ↪ 0;
38     while (front != rear)
39     {
40         int now = h[++front];
41         for (int i = last[now], too; i; i = e[i].pre)
42             if (dist[too = e[i].too] == -1 && e[i].cf)
43             {
44                 dist[too] = dist[now] + 1;
45                 if (too == t) return 1;
46                 h[++rear] = too;
47             }
48     }
49     return 0;
50 }
51
52 int dfs(int x, int f)
53 {
54     if (x == t || !f) return f;
55
56     int flow = 0;
57     for (int &i = last[x], too; i; i = e[i].pre)

```

```

58     if (dist[too = e[i].too] == dist[x] + 1 &&
    ↪ e[i].cf)
59     {
60         int tmp = dfs(too, min(f - flow, e[i].cf));
61         e[i].cf -= tmp; e[i ^ 1].cf += tmp; flow +=
    ↪ tmp;
62         if (flow == f) break;
63     }
64
65     return flow;
66 }
67
68 inline int dinic()
69 {
70     int ans = 0;
71     for (int i = 1; i <= tot; i++) e[i].cf = e[i].c;
72     memcpy(cur, last, sizeof(last));
73     while (bfs())
74     {
75         ans += dfs(s, inf);
76         memcpy(last, cur, sizeof(cur));
77     }
78     return ans;
79 }
80
81 void get_col(int x)
82 {
83     col[x] = now;
84     for (int i = last[x], too; i; i = e[i].pre)
85         if (e[i].cf && col[too = e[i].too] != now)
86             get_col(too);
87 }
88
89 void build(int l, int r)
90 {
91     if (l == r) return;
92     int x = pdx[l], y = pdx[l + 1];
93     s = x; t = y;
94     int cut = dinic();
95     now++; get_col(x);
96     int p = l, q = r;
97     for (int i = 1; i <= r; i++)
98         if (col[pdx[i]] == now) tdx[p++] = pdx[i];
99         else tdx[q--] = pdx[i];
100     for (int i = 1; i <= r; i++)
101         pdx[i] = tdx[i];
102     add2(x, y, cut);
103     add2(y, x, cut);
104     build(l, p - 1); build(q + 1, r);
105 }
106
107 void dfs(int x)
108 {
109     for (int i = last2[x], too; i; i = e2[i].pre)
110         if ((too = e2[i].too) != fa[0][x])
111             fa[0][too] = x, mn[0][too] = e2[i].cf,
    ↪ dep[too] = dep[x] + 1, dfs(too);
112 }
113
114 int query(int x, int y)
115 {
116     int ans = inf;
117     if (dep[x] < dep[y]) swap(x, y);
118     for (int i = 8; ~i; i--)
119         if (dep[fa[i][x]] >= dep[y])
120             ans = min(ans, mn[i][x]), x = fa[i][x];
121     for (int i = 8; ~i; i--)
122         if (fa[i][x] != fa[i][y])

```

```

123         ans = min(ans, min(mn[i][x], mn[i][y])), x
        ↪ = fa[i][x], y = fa[i][y];
124     if (x != y) ans = min(ans, min(mn[0][x],
        ↪ mn[0][y]));
125     return ans;
126 }
127
128 int main()
129 {
130     read(n); read(m);
131     for (int i = 1; i <= m; i++)
132         read(x), read(y), read(z), add(x, y, z), add(y,
        ↪ x, z);
133
134     for (int i = 1; i <= n; i++) pdx[i] = i;
135     memset(mn, 0x3f, sizeof(mn));
136     build(1, n); dep[1] = 1; dfs(1);
137     for (int i = 1; i < 9; i++)
138         for (int j = 1; j <= n; j++)
139         {
140             fa[i][j] = fa[i - 1][fa[i - 1][j]];
141             mn[i][j] = min(mn[i - 1][j], mn[i - 1][fa[i
        ↪ - 1][j]]);
142         }
143     scanf("%d", &Q);
144     while (Q--)
145     {
146         scanf("%d%d", &x, &y);
147         printf("%d\n", query(x, y));
148     }
149 }

```

## 4.15. 最大团 / 最大独立集

```

1 //最大团 / 最大独立集
2 //最大独立集 = 补图的最大团
3 // 复杂度  $O(3^{\frac{n}{3}})$ 
4 #include <iostream>
5 #include <cstring>
6 #include <cstdio>
7 #include <algorithm>
8 #include <cmath>
9
10 using namespace std;
11 const int maxn = 110, inf = 1e9 + 233;
12 int n, m, ans, x, y;
13 bool mp[maxn][maxn];
14 int cnt[maxn], group[maxn], p[maxn];
15
16 bool dfs(int x, int num)
17 {
18     for (int i = x + 1; i <= n; i++)
19     {
20         if (cnt[i] + num <= ans) return 0;
21         if (mp[x][i])
22         {
23             int now = 1;
24             while (now <= num)
25             {
26                 if (!mp[i][p[now]]) break;
27                 now++;
28             }
29             if (now == num + 1)
30             {
31                 p[num + 1] = i;
32                 if (dfs(i, num + 1)) return 1;
33             }
34         }
35     }
36 }

```

```

37     if (num > ans)
38     {
39         ans = num;
40         for (int i = 1; i <= ans; i++)
41             group[i] = p[i];
42         return 1;
43     }
44     return 0;
45 }
46
47 void maxClique()
48 {
49     ans = 0;
50     for (int i = n; i; i--)
51         p[1] = i, dfs(i, 1), cnt[i] = ans;
52 }
53
54 int main()
55 {
56     int T;
57     scanf("%d", &T);
58     while (T--)
59     {
60         memset(mp, 0, sizeof(mp));
61
62         scanf("%d%d", &n, &m);
63         for (int i = 1; i <= m; i++)
64             scanf("%d%d", &x, &y), mp[x][y] = mp[y][x]
        ↪ = 1;
65
66         for (int i = 1; i <= n; i++)
67             for (int j = 1; j <= n; j++)
68                 if (i == j) mp[i][j] = 0;
69                 else mp[i][j] ^= 1;
70
71         maxClique();
72
73         printf("%d\n", ans);
74         for (int i = 1; i <= ans; i++)
75             printf("%d ", group[i]);
76         puts("");
77     }
78 }

```

## 4.16. 2-SAT

```

1 // 复杂度  $O(m)$ 
2 #include <iostream>
3 #include <cstdlib>
4 #include <cstring>
5 #include <cstdio>
6 #include <algorithm>
7 #include <queue>
8 #include <cmath>
9 #include <map>
10 #define ll long long
11 using namespace std;
12 const int maxn = 16010, inf = 1e9;
13 struct poi
14 {
15     int too, pre, x;
16 } e[maxn * 10], e2[maxn * 10];
17 int last[maxn], last2[maxn], dfn[maxn], low[maxn],
    ↪ col[maxn], lack[maxn], ru[maxn], rs[maxn],
    ↪ st[maxn],
18     op[maxn];
19 int n, m, x, y, z, tot, tot2, tott, top, color, flag;
20 void read(int &k)
21 {

```



```

22     int f = 1; k = 0;
23     char c = getchar();
24     while (c < '0' || c > '9') c == '-' && (f = -1), c
        ↪ = getchar();
25     while (c <= '9' && c >= '0') k = k * 10 + c - '0',
        ↪ c = getchar();
26     k *= f;
27 }
28 void add(int x, int y)
29 {
30     e[++tot].too = y;
31     e[tot].x = x;
32     e[tot].pre = last[x];
33     last[x] = tot;
34 }
35 void add2(int x, int y)
36 {
37     e2[++tot2].too = y;
38     e2[tot2].x = x;
39     e2[tot2].pre = last2[x];
40     last2[x] = tot2;
41 }
42 int next(int x)
43 { return x & 1 ? x + 1 : x - 1; }
44 void tarjan(int x)
45 {
46     dfn[x] = low[x] = ++tott;
47     st[++top] = x;
48     lack[x] = top;
49     for (int i = last[x]; i; i = e[i].pre)
50         if (!dfn[e[i].too])
51             tarjan(e[i].too), low[x] = min(low[x],
                ↪ low[e[i].too]);
52         else if (!col[e[i].too])
53             low[x] = min(low[x], dfn[e[i].too]);
54     if (dfn[x] == low[x])
55         for (color++; top >= lack[x]; top--)
            ↪ col[st[top]] = color;
56 }
57 void topsort()
58 {
59     top = 0;
60     for (int i = 1; i <= color; i++)
61         if (!ru[i])
62             st[++top] = i;
63     while (top)
64     {
65         int now = st[top--];
66         if (!rs[now])
67             rs[now] = 1, rs[op[now]] = 2;
68         for (int i = last2[now]; i; i = e2[i].pre)
69             if (!ru[e2[i].too])
70                 st[++top] = e2[i].too;
71     }
72 }
73 void clr()
74 {
75     color = top = tot = tot2 = flag = 0;
76     memset(col, 0, sizeof(col));
77     memset(dfn, 0, sizeof(dfn));
78     memset(ru, 0, sizeof(ru));
79     memset(rs, 0, sizeof(rs));
80     memset(lack, 0, sizeof(lack));
81     memset(last, 0, sizeof(last));
82     memset(last2, 0, sizeof(last2));
83 }
84 int main()
85 {
86     while (scanf("%d%d", &n, &m) != EOF)

```

```

87     {
88         clr();
89         for (int i = 1; i <= m; i++) read(x), read(y),
            ↪ add(x, next(y)), add(y, next(x));
90         for (int i = 1; i <= 2 * n; i++)
91             if (!dfn[i])
92                 tarjan(i);
93         for (int i = 1; i <= n; i++)
94             if (col[i << 1] == col[next(i << 1)])
95             {
96                 printf("NIE\n");
97                 flag = 1;
98                 break;
99             } else
100                 op[col[i << 1]] = col[next(i << 1)],
                    ↪ op[col[next(i << 1)]] = col[i <<
                    ↪ 1];
101         if (flag)
102             continue;
103         for (int i = 1; i <= tot; i++)
104             if (col[e[i].x] != col[e[i].too])
105                 add2(col[e[i].too], col[e[i].x]),
                    ↪ ru[col[e[i].x]]++;
106         topsort();
107         for (int i = 1; i <= n; i++)
108             if (rs[col[(i << 1)]] == 1)
109                 printf("%d\n", i << 1);
110             else
111                 printf("%d\n", next(i << 1));
112     }
113     return 0;
114 }

```

## 4.17. LCA

### 倍增

```

1 // 倍增 LCA
2 // 复杂度  $O(n \log n)$ 
3 #include <iostream>
4 #include <cstring>
5 #include <cstdio>
6 #include <algorithm>
7 #include <cmath>
8
9 using namespace std;
10 const int maxn = 500010, inf = 1e9 + 233;
11 struct edge {int too, pre;} e[maxn << 1];
12 int n, m, s, x, y, tot;
13 int fa[20][maxn], dep[maxn], last[maxn];
14
15 inline void add(int x, int y) {e[++tot] = (edge){y,
    ↪ last[x]}; last[x] = tot;}
16
17 void dfs(int x)
18 {
19     for (int i = last[x], too; i; i = e[i].pre)
20         if ((too = e[i].too) != fa[0][x])
21             dep[too] = dep[x] + 1, fa[0][too] = x,
                ↪ dfs(too);
22 }
23
24 int lca(int x, int y)
25 {
26     if (dep[x] < dep[y]) swap(x, y);
27     for (int i = 19; ~i; i--)
28         if (dep[fa[i][x]] >= dep[y]) x = fa[i][x];
29     for (int i = 19; ~i; i--)
30         if (fa[i][x] != fa[i][y]) x = fa[i][x], y =
            ↪ fa[i][y];

```

```

31     return x == y ? x : fa[0][x];
32 }
33
34 int main()
35 {
36     scanf("%d%d%d", &n, &m, &s);
37     for (int i = 1; i < n; i++)
38         scanf("%d%d", &x, &y), add(x, y), add(y, x);
39
40     dep[s] = 1; dfs(s);
41
42     for (int i = 1; i < 20; i++)
43         for (int j = 1; j <= n; j++)
44             fa[i][j] = fa[i - 1][fa[i - 1][j]];
45
46     for (int i = 1; i <= m; i++)
47         scanf("%d%d", &x, &y), printf("%d\n", lca(x,
48             ↪ y));

```

## ST 表

```

1 // 复杂度 预处理  $O(n \log n)$  单次询问  $O(1)$ 
2 #include <iostream>
3 #include <cstring>
4 #include <cstdio>
5 #include <algorithm>
6 #include <cmath>
7
8 using namespace std;
9 const int maxn = 500010, inf = 1e9 + 233;
10 struct edge{int too, pre;}e[maxn << 1];
11 int n, m, s, x, y, tot, tott;
12 int last[maxn], f[20][maxn << 1], pos[maxn], dep[maxn];
13
14 inline void add(int x, int y) {e[++tot] = (edge){y,
15     ↪ last[x]}; last[x] = tot;}
16
17 void dfs(int x, int fa)
18 {
19     f[0][++tott] = x; pos[x] = tott;
20     for (int i = last[x], too; i; i = e[i].pre)
21         if ((too = e[i].too) != fa)
22         {
23             dep[too] = dep[x] + 1;
24             dfs(too, x);
25             f[0][++tott] = x;
26         }
27
28 inline void rmq()
29 {
30     for (int i = 1; i < 20; i++)
31         for (int j = 1; j <= tott; j++)
32             if (dep[f[i - 1][j]] < dep[f[i - 1][j] + (1
33                 ↪ << (i - 1))])
34                 f[i][j] = f[i - 1][j];
35             else f[i][j] = f[i - 1][j] + (1 << (i -
36                 ↪ 1));
37
38 int lca(int x, int y)
39 {
40     int l = pos[x], r = pos[y];
41     if (l > r) swap(l, r);
42     int k = log2(r - l + 1);
43     if (dep[f[k][l]] < dep[f[k][r - (1 << k) + 1]])
44         ↪ return f[k][l];
45     return f[k][r - (1 << k) + 1];

```

```

45
46 int main()
47 {
48     scanf("%d%d%d", &n, &m, &s);
49     for (int i = 1; i < n; i++)
50         scanf("%d%d", &x, &y), add(x, y), add(y, x);
51
52     dep[0] = inf;
53     dfs(s, 0);
54
55     rmq();
56
57     for (int i = 1; i <= m; i++)
58         scanf("%d%d", &x, &y), printf("%d\n", lca(x,
59             ↪ y));

```

## 树剖

```

1 // 树剖 lca
2 // 复杂度  $O(n \log n)$ 
3 #include <iostream>
4 #include <cstring>
5 #include <cstdio>
6 #include <algorithm>
7 #include <cmath>
8
9 using namespace std;
10 const int maxn = 500010, inf = 1e9 + 233;
11
12 struct edge{int too, pre;}e[maxn << 1];
13
14 int n, m, s, tot, x, y;
15 int last[maxn], size[maxn], fa[maxn], son[maxn],
16     ↪ top[maxn], dep[maxn];
17
18 template <typename T>
19 inline void read(T &k)
20 {
21     int f = 1; k = 0; char c = getchar();
22     while (c < '0' || c > '9') c == '-' && (f = -1), c
23         ↪ = getchar();
24     while (c <= '9' && c >= '0') k = k * 10 + c - '0',
25         ↪ c = getchar();
26     k *= f;
27
28 inline void add(int x, int y){e[++tot] = (edge){y,
29     ↪ last[x]}; last[x] = tot;}
30
31 void dfs1(int x)
32 {
33     size[x] = 1;
34     for (int i = last[x], too; i; i = e[i].pre)
35         if ((too = e[i].too) != fa[x])
36         {
37             fa[too] = x; dep[too] = dep[x] + 1;
38             dfs1(too);
39             size[x] += size[too];
40             if (size[too] > size[son[x]]) son[x] = too;
41         }
42
43 void dfs2(int x, int tp)
44 {
45     top[x] = tp;
46     if (son[x]) dfs2(son[x], tp);
47
48     for (int i = last[x], too; i; i = e[i].pre)
49         if ((too = e[i].too) != fa[x] && too != son[x])

```

```

48         dfs2(too, too);
49 }
50
51 inline int lca(int x, int y)
52 {
53     int f1 = top[x], f2 = top[y];
54     while (f1 != f2)
55     {
56         if (dep[f1] < dep[f2]) swap(x, y), swap(f1,
57             ↪ f2);
58         x = fa[f1]; f1 = top[x];
59     }
60     return dep[x] < dep[y] ? x : y;
61 }
62
63 int main()
64 {
65     read(n); read(m); read(s);
66     for (int i = 1; i < n; i++)
67         read(x), read(y), add(x, y), add(y, x);
68     dfs1(s); dfs2(s, s);
69
70     for (int i = 1; i <= m; i++)
71         read(x), read(y), printf("%d\n", lca(x, y));
72 }

```

## 4.18. 长链剖分

```

1 // 长链剖分求 k 祖先
2 // 预处理  $\mathcal{O}(n \log n)$  单次查询  $\mathcal{O}(1)$ 
3 #include <iostream>
4 #include <cstring>
5 #include <cstdlib>
6 #include <cstdio>
7 #include <cmath>
8 #include <vector>
9 #include <algorithm>
10 #define ll long long
11 using namespace std;
12 const int maxn = 500010, inf = 1e9;
13 struct edge {int too, pre;} e[maxn << 1];
14 int n, m, x, y, tot, K, now;
15 int last[maxn], mxdep[maxn], dep[maxn], fa[21][maxn],
16     ↪ son[maxn], top[maxn], hbt[maxn];
17 vector<int> dv[maxn], uv[maxn];
18
19 template <typename T>
20 inline void read(T &k)
21 {
22     int f = 1;
23     k = 0;
24     char c = getchar();
25     while (c < '0' || c > '9') c == '-' && (f = -1), c
26     ↪ = getchar();
27     while (c <= '9' && c >= '0') k = k * 10 + c - '0',
28     ↪ c = getchar();
29     k *= f;
30 }
31
32 inline void add(int x, int y) {e[++tot] = (edge){y,
33     ↪ last[x]}; last[x] = tot;}
34
35 void dfs1(int x)
36 {
37     for (int i = last[x]; i; i = e[i].pre)
38         if (e[i].too != fa[0][x])
39         {
40             mxdep[e[i].too] = dep[e[i].too] = dep[x] +
41             ↪ 1;

```

```

37         fa[0][e[i].too] = x;
38         dfs1(e[i].too);
39         mxdep[x] = max(mxdep[x], mxdep[e[i].too]);
40         if (mxdep[e[i].too] > mxdep[son[x]])
41             son[x] = e[i].too;
42     }
43 }
44
45 void dfs2(int x, int tp)
46 {
47     top[x] = tp;
48     if (x == tp)
49     {
50         dv[x].push_back(x);
51         uv[x].push_back(x);
52         now = son[x];
53         while (now) dv[x].push_back(now), now =
54             ↪ son[now];
55         now = fa[0][x];
56         for (int i = 1; i < dv[x].size() && now; i++)
57             ↪ uv[x].push_back(now), now = fa[0][now];
58     }
59     if (son[x])
60         dfs2(son[x], tp);
61     for (int i = last[x]; i; i = e[i].pre)
62         if (e[i].too != fa[0][x] && e[i].too != son[x])
63             ↪ dfs2(e[i].too, e[i].too);
64 }
65
66 inline int query(int x, int K)
67 {
68     if (!K) return x;
69     if (K > dep[x])
70         return 0;
71     x = fa[hbt[K]][x];
72     K ^= (1 << hbt[K]);
73     int fx = top[x];
74     if (dep[x] - dep[fx] >= K)
75         return dv[fx][dep[x] - dep[fx] - K];
76     return uv[fx][K - (dep[x] - dep[fx])];
77 }
78
79 int main()
80 {
81     read(n);
82     for (int i = 1; i < n; i++)
83         read(x), read(y), add(x, y), add(y, x);
84     dfs1(1); dfs2(1, 1);
85     for (int j = 1; j <= 19; j++)
86         for (int i = 1; i <= n; i++)
87             fa[j][i] = fa[j - 1][fa[j - 1][i]];
88     for (int i = 0; (1 << i) <= n; i++)
89         hbt[1 << i] = i;
90     for (int i = 1; i <= n; i++)
91         hbt[i] = max(hbt[i], hbt[i - 1]);
92     read(m);
93     int ans = 0;
94     for (int i = 1; i <= m; i++)
95     {
96         read(x); read(K);
97         printf("%d\n", ans = query(x, K));
98     }
99 }

```

## 4.19. 欧拉回路

### 有向图 / 无向图

```

1 // uoj117 模板题库 欧拉回路
2 // 复杂度  $\mathcal{O}(m)$ 

```

```

3 // T = 1 表示无向图
4 // T = 2 表示有向图
5 #include <iostream>
6 #include <cstring>
7 #include <cstdlib>
8 #include <cstdio>
9 #include <cmath>
10 #include <algorithm>
11 #define ll long long
12 using namespace std;
13 const int maxn = 500010, inf = 1e9 + 233;
14 struct edge{int too, pre;}e[maxn << 1];
15 int n, m, T, x, y, tot = 1;
16 int last[maxn], deg_in[maxn], deg_out[maxn], ans[maxn];
17 bool vis[maxn];
18
19 template <typename T>
20 inline void read(T &k)
21 {
22     int f = 1; k = 0; char c = getchar();
23     while (c < '0' || c > '9') c == '-' && (f = -1), c
        ↪ = getchar();
24     while (c <= '9' && c >= '0') k = k * 10 + c - '0',
        ↪ c = getchar();
25     k *= f;
26 }
27
28 inline void add(int x, int y) {e[++tot] = (edge){y,
        ↪ last[x]}; last[x] = tot;}
29
30 void dfs(int x)
31 {
32     for (int &i = last[x]; i; i = e[i].pre)
33         if (!vis[i >> 1])
34             {
35                 vis[i >> 1] = 1;
36                 int tmp = i;
37                 dfs(e[i].too);
38                 ans[++ans[0]] = tmp;
39             }
40 }
41
42 int main()
43 {
44     read(T);
45
46     read(n); read(m);
47     for (int i = 1; i <= m; i++)
48     {
49         read(x); read(y);
50         add(x, y);
51         if (T == 1)
52             add(y, x), deg_out[y]++, deg_out[x]++;
53         else
54             tot++, deg_out[x]++, deg_in[y]++;
55     }
56
57     if (T == 1)
58     {
59         for (int i = 1; i <= n; i++)
60             if (deg_out[i] & 1) return puts("NO"), 0;
61     }
62     else
63     {
64         for (int i = 1; i <= n; i++)
65             if (deg_in[i] != deg_out[i]) return
                ↪ puts("NO"), 0;
66     }
67

```

```

68     dfs(x);
69
70     if (ans[0] != m) puts("NO");
71     else
72     {
73         puts("YES");
74         for (int i = m; i; i--)
75             printf("%d ", (ans[i] & 1) ? -(ans[i] >> 1)
                ↪ : (ans[i] >> 1));
76     }
77 }

```

### 混合图

```

1 // 混合图欧拉回路
2 // 复杂度  $O(n^2 \times m)$ 
3 #include <iostream>
4 #include <cstring>
5 #include <cstdlib>
6 #include <cstdio>
7 #include <cmath>
8 #include <algorithm>
9 #define ll long long
10 using namespace std;
11 const int maxn = 1010, inf = 1e9 + 233;
12
13 struct edge{int too, cf, pre;}e[maxn << 1];
14
15 int tot = 1, n, m, s, t, x, y, z, T;
16 int last[maxn], h[maxn], dist[maxn], cur[maxn],
        ↪ in[maxn], out[maxn];
17
18 template <typename T>
19 inline void read(T &k)
20 {
21     int f = 1; k = 0; char c = getchar();
22     while (c < '0' || c > '9') c == '-' && (f = -1), c
        ↪ = getchar();
23     while (c <= '9' && c >= '0') k = k * 10 + c - '0',
        ↪ c = getchar();
24     k *= f;
25 }
26
27 inline void add(int x, int y, int z) {e[++tot] =
        ↪ (edge){y, z, last[x]}; last[x] = tot;}
28
29 inline void ins(int x, int y, int z) {add(x, y, z);
        ↪ add(y, x, 0);}
30
31 bool bfs()
32 {
33     int front = 0, rear = 1;
34     memset(dist, -1, sizeof(dist)); h[1] = s; dist[s] =
        ↪ 0;
35     while (front != rear)
36     {
37         int now = h[++front];
38         for (int i = last[now], too; i; i = e[i].pre)
39             if (dist[too = e[i].too] == -1 && e[i].cf)
40             {
41                 dist[too] = dist[now] + 1;
42                 if (too == t) return 1;
43                 h[++rear] = too;
44             }
45     }
46     return 0;
47 }
48
49 int dfs(int x, int f)

```

```

50 {
51     if (x == t || !f) return f;
52
53     int flow = 0;
54     for (int &i = last[x], too; i; i = e[i].pre)
55         if (dist[too = e[i].too] == dist[x] + 1 &&
56             ↪ e[i].cf)
57         {
58             int tmp = dfs(too, min(f - flow, e[i].cf));
59             e[i].cf -= tmp; e[i ^ 1].cf += tmp; flow +=
60                 ↪ tmp;
61             if (flow == f) break;
62         }
63     return flow;
64 }
65 inline int dinic()
66 {
67     int ans = 0;
68     memcpy(cur, last, sizeof(last));
69     while (bfs())
70     {
71         ans += dfs(s, inf);
72         memcpy(last, cur, sizeof(cur));
73     }
74     return ans;
75 }
76
77 inline void init()
78 {
79     tot = 1;
80     memset(last, 0, sizeof(last));
81     memset(in, 0, sizeof(in));
82     memset(out, 0, sizeof(out));
83 }
84
85 int main()
86 {
87     scanf("%d", &T);
88     while (T--)
89     {
90         init();
91         scanf("%d%d", &n, &m);
92         for (int i = 1; i <= m; i++)
93         {
94             scanf("%d%d%d", &x, &y, &z);
95             out[x]++; in[y]++;
96             if (!z) ins(x, y, 1);
97         }
98
99         int flag = 0, sum = 0, d;
100         for (int i = 1; i <= n; i++)
101         {
102             d = in[i] - out[i];
103             if (d & 1) flag = 1;
104             else if (d < 0) ins(0, i, (-d) >> 1);
105             else if (d > 0)
106             {
107                 ins(i, n + 1, d >> 1);
108                 sum += d >> 1;
109             }
110         }
111         s = 0, t = n + 1;
112         if (sum != dinic()) flag = 1;
113         puts(flag ? "impossible" : "possible");
114     }
115 }

```

## 4.20. 图的绝对中心

```

1 // 图的绝对中心
2 // 复杂度  $O(n^3)$  floyd
3 #include <iostream>
4 #include <cstring>
5 #include <cstdio>
6 #include <algorithm>
7 #include <cmath>
8
9 using namespace std;
10 const int maxn = 210, inf = 1e9 + 233;
11 struct edge{int x, y, dis;}e[maxn * maxn];
12 int n, m;
13 int rk[maxn][maxn], val[maxn], d[maxn][maxn];
14
15 inline bool cmp(int a, int b) {return val[a] < val[b];}
16
17 void floyd()
18 {
19     for (int k = 1; k <= n; k++)
20         for (int i = 1; i <= n; i++)
21             for (int j = 1; j <= n; j++)
22                 d[i][j] = min(d[i][j], d[i][k] +
23                     ↪ d[k][j]);
24 }
25
26 void solve()
27 {
28     floyd();
29     for (int i = 1; i <= n; i++)
30     {
31         for (int j = 1; j <= n; j++)
32             rk[i][j] = j, val[j] = d[i][j];
33         sort(rk[i] + 1, rk[i] + 1 + n, cmp);
34     }
35
36     int ans = inf;
37     for (int i = 1; i <= n; i++) ans = min(ans,
38         ↪ d[i][rk[i][n]] << 1);
39     for (int i = 1; i <= m; i++)
40     {
41         int x = e[i].x, y = e[i].y, z = e[i].dis;
42         for (int j = n, i = n - 1; i; i--)
43             if (d[y][rk[x][i]] > d[y][rk[x][j]])
44             {
45                 ans = min(ans, d[x][rk[x][i]] +
46                     ↪ d[y][rk[x][j]] + z);
47                 j = i;
48             }
49     }
50
51     printf("%lf\n", ans / 2.0);
52 }
53
54 int main()
55 {
56     scanf("%d%d", &n, &m);
57     memset(d, 0x3f, sizeof(d));
58     for (int i = 1; i <= n; i++) d[i][i] = 0;
59     for (int i = 1; i <= m; i++)
60     {
61         scanf("%d%d%d", &e[i].x, &e[i].y, &e[i].dis);
62         d[e[i].x][e[i].y] = d[e[i].y][e[i].x] =
63             ↪ e[i].dis;
64     }
65     solve();
66 }

```

## 4.21. 虚树

```

1 // 虚树
2 // 虚树建后存在 last2 中
3 #include<iostream>
4 #include<cstring>
5 #include<cstdlib>
6 #include<cstdio>
7 #include<algorithm>
8 #define ll long long
9 using namespace std;
10 const int maxn = 500010, inf = 1e9 + 233;
11 struct poi{int too, dis, pre;}e[maxn], e2[maxn];
12 int n, m, Q, x, y, z, tot, tot2, tott;
13 int size[maxn], dep[maxn], son[maxn], dfn[maxn],
    ↪ las[maxn], last[maxn], last2[maxn], fa[maxn],
    ↪ top[maxn], cost[maxn], a[maxn], st[maxn];
14 ll f[maxn];
15
16 template <typename T>
17 inline void read(T &k)
18 {
19     int f = 1; k = 0; char c = getchar();
20     while (c < '0' || c > '9') c == '-' && (f = -1), c
    ↪ = getchar();
21     while (c <= '9' && c >= '0') k = k * 10 + c - '0',
    ↪ c = getchar();
22     k *= f;
23 }
24
25 inline void add1(int x, int y, int z) {e[++tot] =
    ↪ (poi){y, z, last[x]}; last[x] = tot;}
26
27 inline void add2(int x, int y, int z) {if (x == y)
    ↪ return; e2[++tot2] = (poi){y, z, last2[x]};
    ↪ last2[x] = tot2;}
28
29 inline bool cmp(int a, int b) {return dfn[a] < dfn[b];}
30
31 void dfs1(int x)
32 {
33     size[x] = 1; dep[x] = dep[fa[x]] + 1;
34     for (int i = last[x], too; i; i = e[i].pre)
35         if ((too = e[i].too) != fa[x])
36             {
37                 cost[too] = min(cost[x], e[i].dis);
38                 fa[too] = x; dfs1(too);
39                 size[x] += size[too];
40                 if (size[too] > size[son[x]]) son[x] = too;
41             }
42 }
43
44 void dfs2(int x, int tp)
45 {
46     top[x] = tp; dfn[x] = ++tott;
47     if (son[x]) dfs2(son[x], tp);
48     for (int i = last[x], too; i; i = e[i].pre)
49         if ((too = e[i].too) != fa[x] && too != son[x])
    ↪ dfs2(too, too);
50     las[x] = tott;
51 }
52
53 inline int lca(int x, int y)
54 {
55     int f1 = top[x], f2 = top[y];
56     while (f1 != f2)
57     {
58         if (dep[f1] < dep[f2]) swap(x, y), swap(f1,
    ↪ f2);
59         x = fa[f1]; f1 = top[x];

```

```

60     }
61     return dep[x] < dep[y] ? x : y;
62 }
63
64 void dp(int x)
65 {
66     ll sum = 0;
67     for (int i = last2[x], too; i; i = e2[i].pre)
68         dp(too = e2[i].too), sum += f[too];
69     f[x] = last2[x] ? min(sum, (ll)cost[x]) : cost[x];
70     last2[x] = 0;
71     if (x == 1) f[x] = sum;
72 }
73
74 int main()
75 {
76     read(n);
77     for (int i = 1; i < n; i++)
78         read(x), read(y), read(z), add1(x, y, z),
    ↪ add1(y, x, z);
79     cost[1] = inf; dfs1(1); dfs2(1, 1); read(Q);
80     for (int i = 1; i <= Q; i++)
81     {
82         read(m);
83         for (int j = 1; j <= m; j++) read(a[j]);
84         sort(a + 1, a + 1 + m, cmp);
85         int cnt = 1, top = 1; st[1] = 1; tot2 = 0;
86         for (int j = 2; j <= m; j++)
87             if (dfn[a[cnt]] > dfn[a[j]] || las[a[cnt]]
    ↪ < dfn[a[j]]) a[++cnt] = a[j];
88         for (int j = 1; j <= cnt; j++)
89         {
90             int fa = lca(a[j], st[top]);
91             while (1)
92             {
93                 if (dfn[st[top - 1]] <= dfn[fa])
94                 {
95                     add2(fa, st[top--], 0);
96                     if (dfn[st[top - 1]] < dfn[fa])
    ↪ st[++top] = fa;
97                     break;
98                 }
99                 add2(st[top - 1], st[top], 0); top--;
100             }
101             st[++top] = a[j];
102         }
103         while (top > 1) add2(st[top - 1], st[top], 0),
    ↪ top--;
104         dp(1); printf("%lld\n", f[1]);
105     }
106 }

```

## 4.22. 点分治

## 点分治

时间复杂度  $O(n \log n)$ 

```

1 // 点分治
2 // 复杂度  $O(n \log n)$ 
3 #include <iostream>
4 #include <cstring>
5 #include <cstdlib>
6 #include <cstdio>
7 #include <cmath>
8 #include <algorithm>
9 #define ll long long
10 using namespace std;
11 const int maxn = 500010, inf = 1e9;
12 struct edge {int too, dis, pre;} e[maxn << 1];
13 int n, m, x, y, z, tot, sum, root;

```

```

14 int last[maxn], mx[maxn], size[maxn], cnt[10000010],
   ↪ d[maxn], q[maxn];
15 ll ans[maxn];
16 bool v[maxn];
17
18 template <typename T>
19 inline void read(T &k)
20 {
21     int f = 1; k = 0; char c = getchar();
22     while (c < '0' || c > '9') c == '-' && (f = -1), c
   ↪ = getchar();
23     while (c <= '9' && c >= '0') k = k * 10 + c - '0',
   ↪ c = getchar();
24     k *= f;
25 }
26
27 inline void add(int x, int y, int z) {e[++tot] =
   ↪ (edge){y, z, last[x]}; last[x] = tot;}
28
29 void getroot(int x, int fa)
30 {
31     size[x] = 1; mx[x] = 0;
32     for (int i = last[x], too; i; i = e[i].pre)
33         if ((too = e[i].too) != fa && !v[too])
34             {
35                 getroot(too, x);
36                 size[x] += size[too];
37                 mx[x] = max(mx[x], size[too]);
38             }
39     mx[x] = max(mx[x], sum - size[x]);
40     if (mx[x] < mx[root])
41         root = x;
42 }
43
44 void update(int x, int fa, int delta)
45 {
46     if (delta == 1)
47     {
48         for (int i = 1; i <= m; i++)
49             if (d[x] <= q[i])
50                 ans[i] += cnt[q[i] - d[x]];
51     } else
52         cnt[d[x]] += delta;
53     for (int i = last[x], too; i; i = e[i].pre)
54         if ((too = e[i].too) != fa && !v[too])
55             d[too] = d[x] + e[i].dis, update(too, x,
   ↪ delta);
56 }
57
58 inline void calc(int x)
59 {
60     d[x] = 0; cnt[0] = 1;
61     for (int i = last[x], too; i; i = e[i].pre)
62         if (!v[too = e[i].too])
63             d[too] = d[x] + e[i].dis, update(too, x,
   ↪ 1), update(too, x, 2);
64     for (int i = last[x], too; i; i = e[i].pre)
65         if (!v[too = e[i].too])
66             update(too, x, -2);
67 }
68
69 void dfs(int x)
70 {
71     v[x] = 1; calc(x);
72     for (int i = last[x], too; i; i = e[i].pre)
73         if (!v[too = e[i].too])
74             {
75                 root = 0;
76                 sum = size[too];

```

```

77                 getroot(too, 0);
78                 dfs(root);
79             }
80 }
81
82 int main()
83 {
84     read(n); read(m);
85     for (int i = 1; i < n; i++)
86         read(x), read(y), read(z), add(x, y, z), add(y,
   ↪ x, z);
87     for (int i = 1; i <= m; i++) read(q[i]);
88     root = 0;
89     mx[0] = inf;
90     sum = n;
91     getroot(1, 0);
92     dfs(root);
93     for (int i = 1; i <= m; i++) puts(ans[i] ? "AYE" :
   ↪ "NAY");
94 }

```

### 点分树

时间复杂度  $O(n \log^2 n)$

```

1 // 点分树
2 // init() 为建点分树
3 // 一般维护 tr1 tr2 分别用于维护子树贡献以及去除子树内对
   ↪ 父亲的贡献
4 // faz[] 为在点分树上的父亲
5 // 本题为询问距离不超过 k 的点的权值和
6 // 复杂度  $O(n \log^2 n)$ 
7 #include <iostream>
8 #include <cstring>
9 #include <cstdio>
10 #include <algorithm>
11 #include <cmath>
12 #include <vector>
13
14 using namespace std;
15 const int maxn = 500010, inf = 1e9 + 233;
16
17 int n, m, tott, ty, x, y, tot;
18 int a[maxn], last[maxn], faz[maxn], f[maxn],
   ↪ size[maxn], fa[20][maxn], dep[maxn], p[maxn];
19 bool vis[maxn];
20
21 struct bit
22 {
23     vector <int> tr;
24     void build(int n)
25     {
26         for (int i = 0; i <= n + 3; i++)
27             ↪ tr.push_back(0);
28     }
29     void add(int x, int delta)
30     {
31         for (; x < tr.size(); x += x & -x) tr[x] +=
   ↪ delta;
32     }
33     int query(int x)
34     {
35         x = min(x, (int)tr.size() - 1);
36         int ans = 0;
37         for (; x; x -= x & -x) ans += tr[x];
38         return ans;
39     }
40 }tr1[maxn], tr2[maxn];
41

```

```

42 struct edge{int too, pre;}e[maxn << 1];
43
44 inline void add(int x, int y){e[++tot] = (edge){y,
    ↪ last[x]}; last[x] = tot;}
45
46 int lca (int x, int y)
47 {
48     if (dep[x] < dep[y]) swap(x, y);
49     for (int i = 19; ~i; i--)
50         if (dep[fa[i][x]] >= dep[y]) x = fa[i][x];
51     for (int i = 19; ~i; i--)
52         if (fa[i][x] != fa[i][y]) x = fa[i][x], y =
    ↪ fa[i][y];
53     return x == y ? x : fa[0][x];
54 }
55
56 int get_dis(int x, int y) {return dep[x] + dep[y] -
    ↪ (dep[lca(x, y)] << 1);}
57
58 void dfs1(int x)
59 {
60     for (int i = last[x], too; i; i = e[i].pre)
61         if ((too = e[i].too) != fa[0][x])
62             dep[too] = dep[x] + 1, fa[0][too] = x,
    ↪ dfs1(too);
63 }
64
65 void dfs2(int x, int fa)
66 {
67     p[++tott] = x;
68     size[x] = 1;
69     for (int i = last[x], too; i; i = e[i].pre)
70         if ((too = e[i].too) != fa && !vis[too])
71             dfs2(too, x), size[x] += size[too],
    ↪ f[x] = max(f[x], size[too]);
72 }
73
74 void dfs3(int rt, int x, int fa, int len)
75 {
76     tr1[rt].add(len, a[x]);
77     for (int i = last[x], too; i; i = e[i].pre)
78         if ((too = e[i].too) != fa && !vis[too])
79             dfs3(rt, too, x, len + 1);
80 }
81
82 void init(int x, int fa)
83 {
84     tott = 0;
85     dfs2(x, 0);
86     for (int i = 1; i <= tott; i++)
87         f[p[i]] = max(f[p[i]], tott - size[p[i]]);
88     int rt = 0; f[0] = inf;
89     for (int i = 1; i <= tott; i++)
90         if (f[rt] > f[p[i]]) rt = p[i];
91     for (int i = 1; i <= tott; i++) f[p[i]] = 0;
92     vis[rt] = 1; x = rt; faz[x] = fa;
93     tr2[x].build(tott);
94     if (fa)
95     {
96         for (int i = 1; i <= tott; i++)
97             tr2[x].add(get_dis(p[i], fa), a[p[i]]);
98     }
99     tr1[x].build(tott);
100
101     for (int i = last[x], too; i; i = e[i].pre)
102         if (!vis[too = e[i].too]) dfs3(x, too, x, 1);
103
104     for (int i = last[x], too; i; i = e[i].pre)
105         if (!vis[too = e[i].too]) init(too, x);

```

```

106 }
107
108 void update(int x, int k)
109 {
110     int now = x;
111     while (faz[now])
112     {
113         int d = get_dis(x, faz[now]);
114         tr2[now].add(d, -a[x]);
115         tr2[now].add(d, k);
116         now = faz[now];
117         tr1[now].add(d, -a[x]);
118         tr1[now].add(d, k);
119     }
120     a[x] = k;
121 }
122
123 int query(int x, int k)
124 {
125     int now = x, ans = 0;
126     ans = a[x] + tr1[x].query(k);
127     while (faz[now])
128     {
129         int d = get_dis(faz[now], x);
130         if (d <= k) ans += a[faz[now]] +
    ↪ tr1[faz[now]].query(k - d) -
    ↪ tr2[now].query(k - d);
131         now = faz[now];
132     }
133     return ans;
134 }
135
136 int main()
137 {
138     scanf("%d%d", &n, &m);
139     for (int i = 1; i <= n; i++) scanf("%d", &a[i]);
140     for (int i = 1; i < n; i++)
141         scanf("%d%d", &x, &y), add(x, y), add(y, x);
142     dep[1] = 1; dfs1(1);
143     for (int i = 1; i < 20; i++)
144         for (int j = 1; j <= n; j++)
145             fa[i][j] = fa[i - 1][fa[i - 1][j]];
146     init(1, 0);
147     int pre = 0;
148     for (int i = 1; i <= m; i++)
149     {
150         scanf("%d%d%d", &ty, &x, &y);
151         x ^= pre; y ^= pre;
152         if (!ty) printf("%d\n", pre = query(x, y));
153         else update(x, y);
154     }
155 }

```

## 4.23. 斯坦纳树

时间复杂度  $O(n * 3^k)$

```

1 // 斯坦纳树
2 // 复杂度  $O(n \times 3^k)$ 
3 // WC 2018 游览计划
4 // 带边权无向图上有几个 (一般约 10 个) 点是【关键点】，要求
    ↪ 选择一些边使这些点在同一个联通块内，同时要求所选的边
    ↪ 的边权和最小
5 // dp[i][j] 表示以 i 号节点为根，当前状态为 j (j 的二进制
    ↪ 中已经与 i 连通的点对应位置为 1)
6 // 当根 i 不改变时可以直接转移
7 // 当根 i 改变时 (根变为新加入的节点 j, 且 i 与 j 相邻),
    ↪ 用 SPFA 转移
8 #include <iostream>
9 #include <cstring>

```



```

10#include <cstdlib>
11#include <cstdio>
12#include <cmath>
13#include <algorithm>
14using namespace std;
15const int maxn = 50, inf = 1e9 + 233;
16const int dx[4] = {1, 0, -1, 0}, dy[4] = {0, 1, 0, -1};
17struct poi{int x, y;}h[maxn * maxn];
18struct trans{int i, j, k;}pre[maxn][maxn][1 << 10];
19int n, m, front, rear, cnt;
20int mp[maxn][maxn], f[11][11][1 << 10];
21bool v[11][11];
22
23template <typename T>
24inline void read(T &k)
25{
26    int f = 1; k = 0; char c = getchar();
27    while (c < '0' || c > '9') c == '-' && (f = -1), c
        ↪ = getchar();
28    while (c <= '9' && c >= '0') k = k * 10 + c - '0',
        ↪ c = getchar();
29    k *= f;
30}
31
32void dfs(int x, int y, int st)
33{
34    if (!st) return;
35
36    v[x][y] = 1;
37
38    int i = pre[x][y][st].i, j = pre[x][y][st].j, k =
        ↪ pre[x][y][st].k;
39    dfs(i, j, k);
40    if (i == x && j == y) dfs(i, j, st ^ k);
41}
42
43inline void solve(int x, int y)
44{
45    printf("%d\n", f[x][y][(1 << cnt) - 1]);
46
47    memset(v, 0, sizeof(v));
48    dfs(x, y, (1 << cnt) - 1);
49
50    for (int i = 1; i <= n; i++)
51    {
52        for (int j = 1; j <= m; j++)
53            putchar(mp[i][j] ? (v[i][j] ? 'o' : '_') :
                ↪ 'x');
54
55        puts("");
56    }
57}
58
59inline void spfa(int st)
60{
61    while (front != rear)
62    {
63        poi now = h[++front];
64        if (front == maxn * maxn) front = -1;
65        for (int i = 0; i < 4; i++)
66        {
67            poi nxt = (poi) {now.x + dx[i], now.y +
                ↪ dy[i]};
68
69            if (nxt.x < 1 || nxt.y < 1 || nxt.x > n ||
                ↪ nxt.y > m) continue;
70
71            int delta = f[now.x][now.y][st] +
                ↪ mp[nxt.x][nxt.y];
72
73            if (f[nxt.x][nxt.y][st] > delta)
74            {
75                f[nxt.x][nxt.y][st] = delta;
76                pre[nxt.x][nxt.y][st] = (trans) {now.x,
                    ↪ now.y, st};
77
78                if (!v[nxt.x][nxt.y])
79                {
80                    h[++rear] = (poi) {nxt.x, nxt.y};
81                    if (rear == maxn * maxn) rear = -1;
82
83                    v[now.x][now.y] = 1;
84                }
85            }
86        }
87
88        v[now.x][now.y] = 0;
89    }
90}
91
92int main()
93{
94    memset(f, 0x3f, sizeof(f));
95
96    read(n); read(m); cnt = 0;
97    for (int i = 1; i <= n; i++)
98    {
99        for (int j = 1; j <= m; j++)
100        {
101            read(mp[i][j]);
102            if (!mp[i][j]) f[i][j][1 << cnt] = 0,
                ↪ cnt++;
103        }
104    }
105
106    for (int i = 1; i < (1 << cnt); i++)
107    {
108        front = rear = 0;
109        memset(v, 0, sizeof(v));
110
111        for (int j = 1; j <= n; j++)
112        {
113            for (int k = 1; k <= m; k++)
114            {
115                for (int st = i; st; st = (st - 1) & i)
116                {
117                    int delta = f[j][k][st] + f[j][k][i
                        ↪ ^ st] - mp[j][k];
118                    if (f[j][k][i] > delta)
119                    {
120                        f[j][k][i] = delta;
121                        pre[j][k][i] = (trans) {j, k,
                            ↪ st};
122                    }
123                }
124
125                if (f[j][k][i] < inf) h[++rear] = (poi)
                    ↪ {j, k}, v[j][k] = 1;
126            }
127        }
128
129        spfa(i);
130    }
131
132    for (int i = 1; i <= n; i++)
133    {
134        for (int j = 1; j <= m; j++)

```

```

135         if (!mp[i][j]) return solve(i, j), 0;
136     }
137 }

```

## 4.24. 弦图

弦：连接环中两个不相邻的点的边

弦图：一个无向图中，任意一个大小超过 3 的环都至少有一个弦

弦图的诱导子图一定是弦图

从完美消除序列逆序来

每次给一个点染色可以染的最小的颜色

从中得出结论：团数 = 颜色数

完美消除序列中，能选就选

结论：最大独立集 = 最小团覆盖时间复杂度  $O(n + m)$

### 判断是否弦图

```

1 // 判断一个图是否为弦图
2 // 复杂度  $O(n + m)$ 
3 #include <iostream>
4 #include <cstring>
5 #include <cstdio>
6 #include <algorithm>
7 #include <cmath>
8
9 using namespace std;
10 const int maxn = 500010, inf = 1e9 + 233;
11
12 struct edge{int too, pre;}e[maxn << 1];
13 int n, m, x, y, tot;
14 int last[maxn], deg[maxn], pos[maxn];
15 bool vis[maxn];
16
17 inline void add(int x, int y) {e[++tot] = (edge){y,
    ↪ last[x]}; last[x] = tot;}
18
19 inline void init()
20 {
21     memset(pos, 0, sizeof(pos));
22     memset(last, 0, sizeof(last));
23     memset(deg, 0, sizeof(deg));
24     deg[0] = -1; pos[0] = maxn - 1;
25 }
26
27 inline bool check()
28 {
29     for (int i = n; i; i--)
30     {
31         int x = 0, y = 0;
32         for (int j = 1; j <= n; j++)
33             if (!pos[j] && deg[j] > deg[x])
34                 x = j;
35         pos[x] = i;
36         for (int j = last[x], too; j; j = e[j].pre)
37         {
38             ++deg[too = e[j].too];
39             if (pos[too] > pos[x] && pos[too] < pos[y])
40                 y = too;
41         }
42         if (!y) continue;
43
44         vis[y] = 1;
45         for (int j = last[y]; j; j = e[j].pre)
46             vis[e[j].too] = 1;
47         for (int j = last[x]; j; j = e[j].pre)
48             if (pos[e[j].too] > pos[x] &&
49                 ↪ !vis[e[j].too])
50                 return 0;
51     }
52     return 1;

```

```

53 }
54
55 int main()
56 {
57     while (scanf("%d%d", &n, &m))
58     {
59         if (!n && !m) return 0;
60         init();
61         for (int i = 1; i <= m; i++)
62             scanf("%d%d", &x, &y), add(x, y), add(y,
63                 ↪ x);
64
65         if (check()) puts("Perfect");
66         else puts("Imperfect");
67     }

```

### 弦图最小色数

```

1 // 弦图最大团 / 最小着色
2 // 复杂度  $O(n + m)$ 
3 // 完美消除序列，然后从后往前依次给每个点染上可以染的最小
4 // 颜色个数为 max(label) + 1
5 #include <iostream>
6 #include <cstring>
7 #include <cstdio>
8 #include <algorithm>
9 #include <cmath>
10 #include <vector>
11
12 using namespace std;
13 const int maxn = 1000010, inf = 1e9 + 233;
14
15 struct edge{int too, pre;}e[maxn << 1];
16 int n, m, x, y, tot;
17 int label[maxn], last[maxn], seq[maxn];
18 bool vis[maxn];
19 vector <int> v[maxn];
20
21 inline void add(int x, int y) {e[++tot] = (edge){y,
    ↪ last[x]}; last[x] = tot;}
22
23 inline void MCS()
24 {
25     for (int i = 1; i <= n; i++) v[0].push_back(i);
26     for (int t = 1, now, best = 0; t <= n; t++)
27     {
28         bool flag = 0;
29         while (!flag)
30         {
31             for (int i = v[best].size() - 1; ~i; i--)
32                 if (!vis[v[best][i]])
33                 {
34                     now = v[best][i];
35                     flag = 1;
36                     break;
37                 }
38             else v[best].pop_back();
39             if (!flag) --best;
40         }
41
42         seq[t] = now; vis[now] = 1;
43         for (int i = last[now], too; i; i = e[i].pre)
44             if (!vis[too = e[i].too])
45             {
46                 v[++label[too]].push_back(too);
47                 best = max(best, label[too]);
48             }

```

```

49     }
50 }
51
52 int main()
53 {
54     scanf("%d%d", &n, &m);
55     for (int i = 1; i <= m; i++)
56         scanf("%d%d", &x, &y), add(x, y), add(y, x);
57
58     MCS();
59
60     int ans = 0;
61     for (int i = 1; i <= n; i++) ans = max(ans,
        ↪ label[i]);
62     printf("%d\n", ans + 1);
63 }

```

### 弦图最大独立集

```

1 // 弦图最大独立集 / 最小团覆盖
2 // 完美消除序列中倒序贪心地能选就选
3 // 复杂度  $O(n + m)$ 
4 #include <iostream>
5 #include <cstring>
6 #include <cstdio>
7 #include <algorithm>
8 #include <cmath>
9 #include <vector>
10
11 using namespace std;
12 const int maxn = 500010, inf = 1e9 + 233;
13
14 struct edge{int too, pre;}e[maxn << 1];
15 int n, m, x, y, tot;
16 int label[maxn], last[maxn], seq[maxn];
17 bool vis[maxn];
18 vector <int> v[maxn];
19
20 inline void add(int x, int y) {e[++tot] = (edge){y,
    ↪ last[x]}; last[x] = tot;}
21
22 inline void MCS()
23 {
24     for (int i = 1; i <= n; i++) v[0].push_back(i);
25     for (int t = 1, now, best = 0; t <= n; t++)

```

```

26     {
27         bool flag = 0;
28         while (!flag)
29         {
30             for (int i = v[best].size() - 1; ~i; i--)
31                 if (!vis[v[best][i]])
32                 {
33                     now = v[best][i];
34                     flag = 1;
35                     break;
36                 }
37             else v[best].pop_back();
38             if (!flag) --best;
39         }
40
41         seq[t] = now; vis[now] = 1;
42         for (int i = last[now], too; i; i = e[i].pre)
43             if (!vis[too = e[i].too])
44             {
45                 v[++label[too]].push_back(too);
46                 best = max(best, label[too]);
47             }
48         }
49     }
50
51 int main()
52 {
53     scanf("%d%d", &n, &m);
54     for (int i = 1; i <= m; i++)
55         scanf("%d%d", &x, &y), add(x, y), add(y, x);
56
57     MCS();
58
59     memset(vis, 0, sizeof(vis));
60     int ans = 0;
61     for (int i = n; i; i--)
62         if (!vis[seq[i]])
63         {
64             ans++; vis[seq[i]] = 1;
65             for (int j = last[seq[i]]; j; j = e[j].pre)
66                 vis[e[j].too] = 1;
67         }
68     printf("%d\n", ans);
69 }

```

## Ch5. Geometry

### 5.1. 二维几何

```

1 // 计算几何模板
2 const double eps = 1e-8;
3 const double inf = 1e20;
4 const double pi = acos(-1.0);
5 const int maxp = 1010;
6 // Compares a double to zero
7 int sgn(double x){
8     if(fabs(x) < eps)return 0;
9     if(x < 0)return -1;
10    else return 1;
11 }
12 //square of a double
13 inline double sqr(double x){return x*x;}
14 /*
15  * Point
16  * Point() - Empty constructor
17  * Point(double _x,double _y) - constructor

```

```

18  * input() - double input
19  * output() - %.2f output
20  * operator == - compares x and y
21  * operator < - compares first by x, then by y
22  * operator - - return new Point after
    ↪ subtracting corresponding x and y
23  * operator ^ - cross product of 2d points
24  * operator * - dot product
25  * len() - gives length from origin
26  * len2() - gives square of length from
    ↪ origin
27  * distance(Point p) - gives distance from p
28  * operator + Point b - returns new Point after adding
    ↪ corresponding x and y
29  * operator * double k - returns new Point after
    ↪ multiplieing x and y by k
30  * operator / double k - returns new Point after
    ↪ divideing x and y by k
31  * rad(Point a,Point b)- returns the angle of Point a
    ↪ and Point b from this Point
32  * trunc(double r) - return Point that if truncated
    ↪ the distance from center to r

```

```

33 * rotright()          - returns 90 degree ccw rotated
   ↪ point
34 * rotright()          - returns 90 degree cw rotated
   ↪ point
35 * rotate(Point p,double angle) - returns Point after
   ↪ rotateing the Point centering at p by angle radian
   ↪ ccw
36 */
37 struct Point{
38     double x,y;
39     Point(){
40     Point(double _x,double _y){
41         x = _x;
42         y = _y;
43     }
44     void input(){
45         scanf("%lf%lf",&x,&y);
46     }
47     void output(){
48         printf("%.2f %.2f\n",x,y);
49     }
50     bool operator == (Point b)const{
51         return sgn(x-b.x) == 0 && sgn(y-b.y) == 0;
52     }
53     bool operator < (Point b)const{
54         return sgn(x-b.x)== 0?sgn(y-b.y)<0:x<b.x;
55     }
56     Point operator -(const Point &b)const{
57         return Point(x-b.x,y-b.y);
58     }
59     //叉积
60     double operator ^(const Point &b)const{
61         return x*b.y - y*b.x;
62     }
63     //点积
64     double operator *(const Point &b)const{
65         return x*b.x + y*b.y;
66     }
67     //返回长度
68     double len(){
69         return hypot(x,y); //库函数
70     }
71     //返回长度的平方
72     double len2(){
73         return x*x + y*y;
74     }
75     //返回两点的距离
76     double distance(Point p){
77         return hypot(x-p.x,y-p.y);
78     }
79     Point operator +(const Point &b)const{
80         return Point(x+b.x,y+b.y);
81     }
82     Point operator *(const double &k)const{
83         return Point(x*k,y*k);
84     }
85     Point operator /(const double &k)const{
86         return Point(x/k,y/k);
87     }
88     //计算 pa 和 pb 的夹角
89     //就是求这个点看 a,b 所成的夹角
90     //测试 LightOJ1203
91     double rad(Point a,Point b){
92         Point p = *this;
93         return fabs(atan2(
94             ↪ fabs((a-p)^(b-p)),(a-p)*(b-p) ));
95     }
96     //化为长度为 r 的向量
97     Point trunc(double r){
98         double l = len();
99         if(!sgn(l))return *this;
100        r /= l;
101        return Point(x*r,y*r);
102    }
103    //逆时针旋转 90 度
104    Point rotright(){
105        return Point(-y,x);
106    }
107    //顺时针旋转 90 度
108    Point rotright(){
109        return Point(y,-x);
110    }
111    //绕着 p 点逆时针旋转 angle
112    Point rotate(Point p,double angle){
113        Point v = (*this) - p;
114        double c = cos(angle), s = sin(angle);
115        return Point(p.x + v.x*c - v.y*s,p.y + v.x*s +
116            ↪ v.y*c);
117    }
118    /*
119    * Stores two points
120    * Line() - Empty constructor
121    * Line(Point _s,Point _e) - Line through _s and
122        ↪ _e
123    * operator == - checks if two
124        ↪ points are same
125    * Line(Point p,double angle) - one end p , another
126        ↪ end at angle degree
127    * Line(double a,double b,double c) - Line of equation
128        ↪ ax + by + c = 0
129    * input() - inputs s and e
130    * adjust() - orders in such a
131        ↪ way that s < e
132    * length() - distance of se
133    * angle() - return 0 <= angle <
134        ↪ pi
135    * relation(Point p) - 3 if point is on
136        ↪ line
137    * 1 if point on the
138    * 2 if point on the
139    * left of line
140    * right of line
141    * pointonseg(double p) - return true if
142        ↪ point on segment
143    * parallel(Line v) - return true if they
144        ↪ are parallel
145    * segcrossseg(Line v) - returns 0 if does
146        ↪ not intersect
147    * returns 1 if
148    * non-standard intersection
149    * returns 2 if
150    * intersects
151    * linecrossseg(Line v) - line and seg
152    * linecrossline(Line v) - 0 if parallel
153    * 1 if coincides
154    * 2 if intersects
155    * crosspoint(Line v) - returns
156        ↪ intersection point
157    * dispointtoline(Point p) - distance from point
158        ↪ p to the line
159    * dispointtoseg(Point p) - distance from p to
160        ↪ the segment
161    * dissegtoseg(Line v) - distance of two
162        ↪ segment
163    * lineprog(Point p) - returns projected
164        ↪ point p on se line

```

```

145 * symmetrypoint(Point p)          - returns reflection
    ↪ point of p over se
146 *
147 */
148 struct Line{
149     Point s,e;
150     Line(){ }
151     Line(Point _s,Point _e){
152         s = _s;
153         e = _e;
154     }
155     bool operator ==(Line v){
156         return (s == v.s)&&(e == v.e);
157     }
158     //根据一个点和倾斜角 angle 确定直线,0<=angle<pi`
159     Line(Point p,double angle){
160         s = p;
161         if(sgn(angle-pi/2) == 0){
162             e = (s + Point(0,1));
163         }
164         else{
165             e = (s + Point(1,tan(angle)));
166         }
167     }
168     //ax+by+c=0
169     Line(double a,double b,double c){
170         if(sgn(a) == 0){
171             s = Point(0,-c/b);
172             e = Point(1,-c/b);
173         }
174         else if(sgn(b) == 0){
175             s = Point(-c/a,0);
176             e = Point(-c/a,1);
177         }
178         else{
179             s = Point(0,-c/b);
180             e = Point(1,(-c-a)/b);
181         }
182     }
183     void input(){
184         s.input();
185         e.input();
186     }
187     void adjust(){
188         if(e < s)swap(s,e);
189     }
190     //求线段长度
191     double length(){
192         return s.distance(e);
193     }
194     //返回直线倾斜角 0<=angle<pi`
195     double angle(){
196         double k = atan2(e.y-s.y,e.x-s.x);
197         if(sgn(k) < 0)k += pi;
198         if(sgn(k-pi) == 0)k -= pi;
199         return k;
200     }
201     //点和直线关系`
202     //1 在左侧`
203     //2 在右侧`
204     //3 在直线上`
205     int relation(Point p){
206         int c = sgn((p-s)^(e-s));
207         if(c < 0)return 1;
208         else if(c > 0)return 2;
209         else return 3;
210     }
211     //点在线段上的判断
212     bool pointonseg(Point p){
213         return sgn((p-s)^(e-s)) == 0 &&
            ↪ sgn((p-s)*(p-e)) <= 0;
214     }
215     //两向量平行 (对应直线平行或重合)`
216     bool parallel(Line v){
217         return sgn((e-s)^(v.e-v.s)) == 0;
218     }
219     //两线段相交判断`
220     //2 规范相交`
221     //1 非规范相交`
222     //0 不相交`
223     int segcrossseg(Line v){
224         int d1 = sgn((e-s)^(v.s-s));
225         int d2 = sgn((e-s)^(v.e-s));
226         int d3 = sgn((v.e-v.s)^(s-v.s));
227         int d4 = sgn((v.e-v.s)^(e-v.s));
228         if((d1^d2)==-2 && (d3^d4)==-2 )return 2;
229         return (d1==0 && sgn((v.s-s)*(v.s-e))<=0) ||
            ↪ (d2==0 && sgn((v.e-s)*(v.e-e))<=0) ||
            ↪ (d3==0 && sgn((s-v.s)*(s-v.e))<=0) ||
            ↪ (d4==0 && sgn((e-v.s)*(e-v.e))<=0);
230     }
231     //直线和线段相交判断`
232     //~*this line -v seg`
233     //2 规范相交`
234     //1 非规范相交`
235     //0 不相交`
236     int linecrossseg(Line v){
237         int d1 = sgn((e-s)^(v.s-s));
238         int d2 = sgn((e-s)^(v.e-s));
239         if((d1^d2)==-2) return 2;
240         return (d1==0 || d2==0);
241     }
242     //两直线关系`
243     //0 平行`
244     //1 重合`
245     //2 相交`
246     int linecrossline(Line v){
247         if((*this).parallel(v))
248             return v.relation(s)==3;
249         return 2;
250     }
251     //求两直线的交点`
252     //要保证两直线不平行或重合`
253     Point crosspoint(Line v){
254         double a1 = (v.e-v.s)^(s-v.s);
255         double a2 = (v.e-v.s)^(e-v.s);
256         return Point((s.x*a2-e.x*a1)/(a2-a1), (
            ↪ s.y*a2-e.y*a1)/(a2-a1));
257     }
258     //点到直线的距离
259     double dispointtoline(Point p){
260         return fabs((p-s)^(e-s))/length();
261     }
262     //点到线段的距离
263     double dispointtoseg(Point p){
264         if(sgn((p-s)*(e-s))<0 || sgn((p-e)*(s-e))<0)
265             return min(p.distance(s),p.distance(e));
266         return dispointtoline(p);
267     }
268     //返回线段到线段的距离`
269     //前提是两线段不相交,相交距离就是 0 了`
270     double dissegtoseg(Line v){
271         return min(min(dispointtoseg(v.s),
            ↪ dispointtoseg(v.e)),min(v.dispointtoseg(
            ↪ s),v.dispointtoseg(e)));
272     }
273     //返回点 p 在直线上的投影`
274

```

```

277 Point lineprog(Point p){
278     return s + (
        ↪ ((e-s)*((e-s)*(p-s)))/((e-s).len2()) );
279 }
280 //返回点 p 关于直线的对称点`
281 Point symmetrpoint(Point p){
282     Point q = lineprog(p);
283     return Point(2*q.x-p.x,2*q.y-p.y);
284 }
285 };
286 //圆
287 struct circle{
288     Point p;//圆心
289     double r;//半径
290     circle(){}
291     circle(Point _p,double _r){
292         p = _p;
293         r = _r;
294     }
295     circle(double x,double y,double _r){
296         p = Point(x,y);
297         r = _r;
298     }
299     //`三角形的外接圆`
300     //`需要 Point 的 + / rotate() 以及 Line 的
        ↪ crosspoint()`
301     //`利用两条边的中垂线得到圆心`
302     //`测试: UVA12304`
303     circle(Point a,Point b,Point c){
304         Line u =
            ↪ Line((a+b)/2,((a+b)/2)+((b-a).rotleft()));
305         Line v =
            ↪ Line((b+c)/2,((b+c)/2)+((c-b).rotleft()));
306         p = u.crosspoint(v);
307         r = p.distance(a);
308     }
309     //`三角形的内切圆`
310     //`参数 bool t 没有作用,只是为了和上面外接圆函数区
        ↪ 别`
311     //`测试: UVA12304`
312     circle(Point a,Point b,Point c,bool t){
313         Line u,v;
314         double m = atan2(b.y-a.y,b.x-a.x), n =
            ↪ atan2(c.y-a.y,c.x-a.x);
315         u.s = a;
316         u.e = u.s + Point(cos((n+m)/2),sin((n+m)/2));
317         v.s = b;
318         m = atan2(a.y-b.y,a.x-b.x), n =
            ↪ atan2(c.y-b.y,c.x-b.x);
319         v.e = v.s + Point(cos((n+m)/2),sin((n+m)/2));
320         p = u.crosspoint(v);
321         r = Line(a,b).dispointtoseg(p);
322     }
323     //输入
324     void input(){
325         p.input();
326         scanf("%lf",&r);
327     }
328     //输出
329     void output(){
330         printf("%.2lf %.2lf %.2lf\n",p.x,p.y,r);
331     }
332     bool operator == (circle v){
333         return (p==v.p) && sgn(r-v.r)==0;
334     }
335     bool operator < (circle v) const{
336         return ((p<v.p)||((p==v.p)&&sgn(r-v.r)<0));
337     }
338     //面积

```

```

339 double area(){
340     return pi*r*r;
341 }
342 //周长
343 double circumference(){
344     return 2*pi*r;
345 }
346 //`点和圆的关系`
347 //`0 圆外`
348 //`1 圆上`
349 //`2 圆内`
350 int relation(Point b){
351     double dst = b.distance(p);
352     if(sgn(dst-r) < 0)return 2;
353     else if(sgn(dst-r)==0)return 1;
354     return 0;
355 }
356 //`线段和圆的关系`
357 //`比较的是圆心到线段的距离和半径的关系`
358 int relationseg(Line v){
359     double dst = v.dispointtoseg(p);
360     if(sgn(dst-r) < 0)return 2;
361     else if(sgn(dst-r) == 0)return 1;
362     return 0;
363 }
364 //`直线和圆的关系`
365 //`比较的是圆心到直线的距离和半径的关系`
366 int relationline(Line v){
367     double dst = v.dispointtoline(p);
368     if(sgn(dst-r) < 0)return 2;
369     else if(sgn(dst-r) == 0)return 1;
370     return 0;
371 }
372 //`两圆的关系`
373 //`5 相离`
374 //`4 外切`
375 //`3 相交`
376 //`2 内切`
377 //`1 内含`
378 //`需要 Point 的 distance`
379 //`测试: UVA12304`
380 int relationcircle(circle v){
381     double d = p.distance(v.p);
382     if(sgn(d-r-v.r) > 0)return 5;
383     if(sgn(d-r-v.r) == 0)return 4;
384     double l = fabs(r-v.r);
385     if(sgn(d-r-v.r)<0 && sgn(d-l)>0)return 3;
386     if(sgn(d-l)==0)return 2;
387     if(sgn(d-l)<0)return 1;
388 }
389 //`求两个圆的交点,返回 0 表示没有交点,返回 1 是一个
    ↪ 交点,2 是两个交点`
390 //`需要 relationcircle`
391 //`测试: UVA12304`
392 int pointcrosscircle(circle v,Point &p1,Point &p2){
393     int rel = relationcircle(v);
394     if(rel == 1 || rel == 5)return 0;
395     double d = p.distance(v.p);
396     double l = (d*d+r*r-v.r*v.r)/(2*d);
397     double h = sqrt(r*r-l*l);
398     Point tmp = p + (v.p-p).trunc(l);
399     p1 = tmp + ((v.p-p).rotleft().trunc(h));
400     p2 = tmp + ((v.p-p).rotright().trunc(h));
401     if(rel == 2 || rel == 4)
402         return 1;
403     return 2;
404 }
405 //`求直线和圆的交点,返回交点个数`

```

```

406 int pointcrossline(Line v,Point &p1,Point &p2){
407     if(!(*this).relationline(v))return 0;
408     Point a = v.lineprog(p);
409     double d = v.dispointtoline(p);
410     d = sqrt(r*r-d*d);
411     if(sgn(d) == 0){
412         p1 = a;
413         p2 = a;
414         return 1;
415     }
416     p1 = a + (v.e-v.s).trunc(d);
417     p2 = a - (v.e-v.s).trunc(d);
418     return 2;
419 }
420 //得到过 a,b 两点, 半径为 r1 的两个圆`
421 int gercircle(Point a,Point b,double r1,circle
    ↪ &c1,circle &c2){
422     circle x(a,r1),y(b,r1);
423     int t = x.pointcrosscircle(y,c1.p,c2.p);
424     if(!t)return 0;
425     c1.r = c2.r = r;
426     return t;
427 }
428 //得到与直线 u 相切, 过点 q, 半径为 r1 的圆`
429 //测试: UVA12304`
430 int getcircle(Line u,Point q,double r1,circle
    ↪ &c1,circle &c2){
431     double dis = u.dispointtoline(q);
432     if(sgn(dis-r1*2)>0)return 0;
433     if(sgn(dis) == 0){
434         c1.p = q + ((u.e-u.s).rotleft().trunc(r1));
435         c2.p = q +
            ↪ ((u.e-u.s).rotright().trunc(r1));
436         c1.r = c2.r = r1;
437         return 2;
438     }
439     Line u1 = Line((u.s +
        ↪ (u.e-u.s).rotleft().trunc(r1)),u.e +
        ↪ (u.e-u.s).rotleft().trunc(r1));
440     Line u2 = Line((u.s +
        ↪ (u.e-u.s).rotright().trunc(r1)),u.e +
        ↪ (u.e-u.s).rotright().trunc(r1));
441     circle cc = circle(q,r1);
442     Point p1,p2;
443     if(!cc.pointcrossline(u1,p1,p2))
        ↪ cc.pointcrossline(u2,p1,p2);
444     c1 = circle(p1,r1);
445     if(p1 == p2){
446         c2 = c1;
447         return 1;
448     }
449     c2 = circle(p2,r1);
450     return 2;
451 }
452 //同时与直线 u,v 相切, 半径为 r1 的圆`
453 //测试: UVA12304`
454 int getcircle(Line u,Line v,double r1,circle
    ↪ &c1,circle &c2,circle &c3,circle &c4){
455     if(u.parallel(v))return 0;//两直线平行
456     Line u1 = Line(u.s +
        ↪ (u.e-u.s).rotleft().trunc(r1),u.e +
        ↪ (u.e-u.s).rotleft().trunc(r1));
457     Line u2 = Line(u.s +
        ↪ (u.e-u.s).rotright().trunc(r1),u.e +
        ↪ (u.e-u.s).rotright().trunc(r1));
458     Line v1 = Line(v.s +
        ↪ (v.e-v.s).rotleft().trunc(r1),v.e +
        ↪ (v.e-v.s).rotleft().trunc(r1));
459     Line v2 = Line(v.s +
        ↪ (v.e-v.s).rotright().trunc(r1),v.e +
        ↪ (v.e-v.s).rotright().trunc(r1));
460     c1.r = c2.r = c3.r = c4.r = r1;
461     c1.p = u1.crosspoint(v1);
462     c2.p = u1.crosspoint(v2);
463     c3.p = u2.crosspoint(v1);
464     c4.p = u2.crosspoint(v2);
465     return 4;
466 }
467 //同时与不相交圆 cx,cy 相切, 半径为 r1 的圆`
468 //测试: UVA12304`
469 int getcircle(circle cx,circle cy,double r1,circle
    ↪ &c1,circle &c2){
470     circle x(cx.p,r1+cx.r),y(cy.p,r1+cy.r);
471     int t = x.pointcrosscircle(y,c1.p,c2.p);
472     if(!t)return 0;
473     c1.r = c2.r = r1;
474     return t;
475 }
476 //过一点作圆的切线 (先判断点和圆的关系)`
477 //测试: UVA12304`
478 int tangentline(Point q,Line &u,Line &v){
479     int x = relation(q);
480     if(x == 2)return 0;
481     if(x == 1){
482         u = Line(q,q + (q-p).rotleft());
483         v = u;
484         return 1;
485     }
486     double d = p.distance(q);
487     double l = r*r/d;
488     double h = sqrt(r*r-l*l);
489     u = Line(q,p + ((q-p).trunc(l) +
        ↪ (q-p).rotleft().trunc(h)));
490     v = Line(q,p + ((q-p).trunc(l) +
        ↪ (q-p).rotright().trunc(h)));
491     return 2;
492 }
493 //求两圆相交的面积`
494 double areacircle(circle v){
495     int rel = relationcircle(v);
496     if(rel >= 4)return 0.0;
497     if(rel <= 2)return min(area(),v.area());
498     double d = p.distance(v.p);
499     double hf = (r+v.r+d)/2.0;
500     double ss = 2*sqrt(hf*(hf-r)*(hf-v.r)*(hf-d));
501     double a1 = acos((r*r+d*d-v.r*v.r)/(2.0*r*d));
502     a1 = a1*r*r;
503     double a2 =
        ↪ acos((v.r*v.r+d*d-r*r)/(2.0*v.r*d));
504     a2 = a2*v.r*v.r;
505     return a1+a2-ss;
506 }
507 //求圆和三角形 pab 的相交面积`
508 //测试: POJ3675 HDU3982 HDU2892`
509 double areatriangle(Point a,Point b){
510     if(sgn((p-a)^(p-b)) == 0)return 0.0;
511     Point q[5];
512     int len = 0;
513     q[len++] = a;
514     Line l(a,b);
515     Point p1,p2;
516     if(pointcrossline(l,q[1],q[2])==2){
517         if(sgn((a-q[1])*(b-q[1]))<0)q[len++] =
            ↪ q[1];
518         if(sgn((a-q[2])*(b-q[2]))<0)q[len++] =
            ↪ q[2];
519     }

```

```

520     }
521     q[len++] = b;
522     if(len == 4 && sgn((q[0]-q[1])*(q[2]-q[1]))
    ↪ >0)swap(q[1],q[2]);
523     double res = 0;
524     for(int i = 0;i < len-1;i++){
525         if(relation(q[i])==0||relation(q[i+1])==0){
526             double arg = p.rad(q[i],q[i+1]);
527             res += r*r*arg/2.0;
528         }
529         else{
530             res += fabs((q[i]-p)^(q[i+1]-p))/2.0;
531         }
532     }
533     return res;
534 }
535 };
536
537 /*
538 * n,p Line l for each side
539 * input(int _n) - inputs _n
    ↪ size polygon
540 * add(Point q) - adds a point
    ↪ at end of the list
541 * getline() - populates
    ↪ line array
542 * cmp - comparision
    ↪ in convex_hull order
543 * norm() - sorting in
    ↪ convex_hull order
544 * getconvex(polygon &convex) - returns
    ↪ convex hull in convex
545 * Graham(polygon &convex) - returns
    ↪ convex hull in convex
546 * isconvex() - checks if
    ↪ convex
547 * relationpoint(Point q) - returns 3 if
    ↪ q is a vertex
548 * - 2 if
    ↪ on a side
549 * - 1 if
    ↪ inside
550 * - 0 if
    ↪ outside
551 * convexcute(Line u,polygon &po) - left side of
    ↪ u in po
552 * gercircumference() - returns side
    ↪ length
553 * getarea() - returns area
554 * getdir() - returns 0 for
    ↪ cw, 1 for ccw
555 * getbarycentre() - returns
    ↪ barycenter
556 *
557 */
558 struct polygon{
559     int n;
560     Point p[maxp];
561     Line l[maxp];
562     void input(int _n){
563         n = _n;
564         for(int i = 0;i < n;i++){
565             p[i].input();
566         }
567     }
568     void add(Point q){
569         p[n++] = q;
570     }
571     void getline(){
572         for(int i = 0;i < n;i++){
573             l[i] = Line(p[i],p[(i+1)%n]);
574         }
575     }
576     struct cmp{
577         Point p;
578         cmp(const Point &p0){p = p0;}
579         bool operator()(const Point &aa,const Point
    ↪ &bb){
580             Point a = aa, b = bb;
581             int d = sgn((a-p)^(b-p));
582             if(d == 0){
583                 return sgn(a.distance(p)-b.distance(p))
    ↪ < 0;
584             }
585             return d > 0;
586         }
587     };
588     //进行极角排序`
589     //首先需要找到最左下角的点`
590     //需要重载号好 Point 的 < 操作符 (min 函数要用) `
591     void norm(){
592         Point mi = p[0];
593         for(int i = 1;i < n;i++)mi = min(mi,p[i]);
594         sort(p,p+n,cmp(mi));
595     }
596     //得到凸包`
597     //得到的凸包里面的点编号是 0~n-1 的`
598     //两种凸包的方法`
599     //注意如果有影响,要特判下所有点共点,或者共线的特殊
    ↪ 情况`
600     //测试 LightOJ1203 LightOJ1239`
601     void getconvex(polygon &convex){
602         sort(p,p+n);
603         convex.n = n;
604         for(int i = 0;i < min(n,2);i++){
605             convex.p[i] = p[i];
606         }
607         if(convex.n == 2 && (convex.p[0] ==
    ↪ convex.p[1]))convex.n--;//特判
608         if(n <= 2)return;
609         int &top = convex.n;
610         top = 1;
611         for(int i = 2;i < n;i++){
612             while(top && sgn((convex.p[top]-p[i])^(
    ↪ convex.p[top-1]-p[i])) <=
    ↪ 0)
613                 top--;
614             convex.p[++top] = p[i];
615         }
616         int temp = top;
617         convex.p[++top] = p[n-2];
618         for(int i = n-3;i >= 0;i--){
619             while(top != temp &&
    ↪ sgn((convex.p[top]-p[i])^(
    ↪ convex.p[top-1]-p[i])) <=
    ↪ 0)
620                 top--;
621             convex.p[++top] = p[i];
622         }
623         if(convex.n == 2 && (convex.p[0] ==
    ↪ convex.p[1]))convex.n--;//特判
624         convex.norm();//原来得到的是顺时针的点,排序后
    ↪ 逆时针`
625     }
626     //得到凸包的另外一种方法`
627     //测试 LightOJ1203 LightOJ1239`
628     void Graham(polygon &convex){
629         norm();

```



```

629     int &top = convex.n;
630     top = 0;
631     if(n == 1){
632         top = 1;
633         convex.p[0] = p[0];
634         return;
635     }
636     if(n == 2){
637         top = 2;
638         convex.p[0] = p[0];
639         convex.p[1] = p[1];
640         if(convex.p[0] == convex.p[1])top--;
641         return;
642     }
643     convex.p[0] = p[0];
644     convex.p[1] = p[1];
645     top = 2;
646     for(int i = 2; i < n; i++){
647         while( top > 1 &&
            ↪ sgn((convex.p[top-1]-convex.p[top-2])
            ↪ ^ (p[i]-convex.p[top-2])) <= 0
            ↪ )
648             top--;
649         convex.p[top++] = p[i];
650     }
651     if(convex.n == 2 && (convex.p[0] ==
            ↪ convex.p[1]))convex.n--; //特判
652 }
653 //判断是不是凸的
654 bool isconvex(){
655     bool s[2];
656     memset(s, false, sizeof(s));
657     for(int i = 0; i < n; i++){
658         int j = (i+1)%n;
659         int k = (j+1)%n;
660         s[sgn((p[j]-p[i])^(p[k]-p[i]))+1] = true;
661         if(s[0] && s[2])return false;
662     }
663     return true;
664 }
665 //判断点和任意多边形的关系
666 // 3 点上
667 // 2 边上
668 // 1 内部
669 // 0 外部
670 int relationpoint(Point q){
671     for(int i = 0; i < n; i++){
672         if(p[i] == q)return 3;
673     }
674     getline();
675     for(int i = 0; i < n; i++){
676         if(l[i].pointonseg(q))return 2;
677     }
678     int cnt = 0;
679     for(int i = 0; i < n; i++){
680         int j = (i+1)%n;
681         int k = sgn((q-p[j])^(p[i]-p[j]));
682         int u = sgn(p[i].y-q.y);
683         int v = sgn(p[j].y-q.y);
684         if(k > 0 && u < 0 && v >= 0)cnt++;
685         if(k < 0 && v < 0 && u >= 0)cnt--;
686     }
687     return cnt != 0;
688 }
689 //直线 u 切割凸多边形左侧
690 //注意直线方向
691 //测试: HDU3982
692 void convexcut(Line u, polygon &po){
693     int &top = po.n; //注意引用
694     top = 0;
695     for(int i = 0; i < n; i++){
696         int d1 = sgn((u.e-u.s)^(p[i]-u.s));
697         int d2 = sgn((u.e-u.s)^(p[(i+1)%n]-u.s));
698         if(d1 >= 0)po.p[top++] = p[i];
699         if(d1*d2 < 0)po.p[top++] =
            ↪ u.crosspoint(Line(p[i], p[(i+1)%n]));
700     }
701 }
702 //得到周长
703 //测试 LightOJ1239
704 double getcircumference(){
705     double sum = 0;
706     for(int i = 0; i < n; i++){
707         sum += p[i].distance(p[(i+1)%n]);
708     }
709     return sum;
710 }
711 //得到面积
712 double getarea(){
713     double sum = 0;
714     for(int i = 0; i < n; i++){
715         sum += (p[i]^p[(i+1)%n]);
716     }
717     return fabs(sum)/2;
718 }
719 //得到方向
720 // 1 表示逆时针, 0 表示顺时针
721 bool getdir(){
722     double sum = 0;
723     for(int i = 0; i < n; i++){
724         sum += (p[i]^p[(i+1)%n]);
725     }
726     if(sgn(sum) > 0)return 1;
727     return 0;
728 }
729 //得到重心
730 Point getbarycentre(){
731     Point ret(0,0);
732     double area = 0;
733     for(int i = 1; i < n-1; i++){
734         double tmp = (p[i]-p[0])^(p[i+1]-p[0]);
735         if(sgn(tmp) == 0)continue;
736         area += tmp;
737         ret.x += (p[0].x+p[i].x+p[i+1].x)/3*tmp;
738         ret.y += (p[0].y+p[i].y+p[i+1].y)/3*tmp;
739     }
740     if(sgn(area)) ret = ret/area;
741     return ret;
742 }
743 //多边形和圆交的面积
744 //测试: POJ3675 HDU3982 HDU2892
745 double areacircle(circle c){
746     double ans = 0;
747     for(int i = 0; i < n; i++){
748         int j = (i+1)%n;
749         if(sgn( (p[j]-c.p)^(p[i]-c.p) ) >= 0)
750             ans += c.areatriangle(p[i], p[j]);
751         else ans -= c.areatriangle(p[i], p[j]);
752     }
753     return fabs(ans);
754 }
755 //多边形和圆关系
756 // 2 圆完全在多边形内
757 // 1 圆在多边形里面, 碰到了多边形边界
758 // 0 其它
759 int relationcircle(circle c){
760     getline();
761     int x = 2;

```

```

761     if(relationpoint(c.p) != 1)return 0;//圆心不在
    ↪ 内部
762     for(int i = 0;i < n;i++){
763         if(c.relationseg(l[i])==2)return 0;
764         if(c.relationseg(l[i])==1)x = 1;
765     }
766     return x;
767 }
768 };
769 //`AB X AC`
770 double cross(Point A,Point B,Point C){
771     return (B-A)^(C-A);
772 }
773 //`AB*AC`
774 double dot(Point A,Point B,Point C){
775     return (B-A)*(C-A);
776 }
777 //`最小矩形面积覆盖`
778 //` A 必须是凸包 (而且是逆时针顺序)`
779 //` 测试 UVA 10173`
780 double minRectangleCover(polygon A){
781     //`要特判 A.n < 3 的情况`
782     if(A.n < 3)return 0.0;
783     A.p[A.n] = A.p[0];
784     double ans = -1;
785     int r = 1, p = 1, q;
786     for(int i = 0;i < A.n;i++){
787         //`卡出离边 A.p[i] - A.p[i+1] 最远的点`
788         while( sgn( cross(A.p[i],A.p[i+1],A.p[r+1]) -
    ↪ cross(A.p[i],A.p[i+1],A.p[r]) ) >= 0 )
789             r = (r+1)%A.n;
790         //`卡出 A.p[i] - A.p[i+1] 方向上正向 n 最远的点`
791         while(sgn( dot(A.p[i],A.p[i+1],A.p[p+1]) -
    ↪ dot(A.p[i],A.p[i+1],A.p[p]) ) >= 0 )
792             p = (p+1)%A.n;
793         if(i == 0)q = p;
794         //`卡出 A.p[i] - A.p[i+1] 方向上负向最远的点`
795         while(sgn(dot(A.p[i],A.p[i+1],A.p[q+1]) -
    ↪ dot(A.p[i],A.p[i+1],A.p[q])) <= 0)
796             q = (q+1)%A.n;
797         double d = (A.p[i] - A.p[i+1]).len2();
798         double tmp = cross(A.p[i],A.p[i+1],A.p[r]) *
799             (dot(A.p[i],A.p[i+1],A.p[p]) -
    ↪ dot(A.p[i],A.p[i+1],A.p[q]))/d;
800         if(ans < 0 || ans > tmp)ans = tmp;
801     }
802     return ans;
803 }
804
805 //`直线切凸多边形`
806 //`多边形是逆时针的, 在 q1q2 的左侧`
807 //`测试:HDU3982`
808 vector<Point> convexCut(const vector<Point> &ps,Point
    ↪ q1,Point q2){
809     vector<Point>qs;
810     int n = ps.size();
811     for(int i = 0;i < n;i++){
812         Point p1 = ps[i], p2 = ps[(i+1)%n];
813         int d1 = sgn((q2-q1)^(p1-q1)), d2 =
    ↪ sgn((q2-q1)^(p2-q1));
814         if(d1 >= 0)
815             qs.push_back(p1);
816         if(d1 * d2 < 0)
817             qs.push_back(Line(p1,p2).crosspoint(Line(
    ↪ q1,q2)));
818     }
819     return qs;
820 }
821 //`半平面交`
822 //`测试 POJ3335 POJ1474 POJ1279`
823 //`*****`
824 struct halfplane:public Line{
825     double angle;
826     halfplane(){}
827     //`表示向量 s->e 逆时针 (左侧) 的半平面`
828     halfplane(Point _s,Point _e){
829         s = _s;
830         e = _e;
831     }
832     halfplane(Line v){
833         s = v.s;
834         e = v.e;
835     }
836     void calcangle(){
837         angle = atan2(e.y-s.y,e.x-s.x);
838     }
839     bool operator <(const halfplane &b)const{
840         return angle < b.angle;
841     }
842 };
843 struct halfplanes{
844     int n;
845     halfplane hp[maxp];
846     Point p[maxp];
847     int que[maxp];
848     int st,ed;
849     void push(halfplane tmp){
850         hp[n++] = tmp;
851     }
852     //去重
853     void unique(){
854         int m = 1;
855         for(int i = 1;i < n;i++){
856             if(sgn(hp[i].angle-hp[i-1].angle) != 0)
857                 hp[m++] = hp[i];
858             else if(sgn( (hp[m-1].e-hp[m-1].s)^(
    ↪ hp[i].s-hp[m-1].s) ) >
    ↪ 0)
859                 hp[m-1] = hp[i];
860         }
861         n = m;
862     }
863     bool halfplaneinsert(){
864         for(int i = 0;i < n;i++)hp[i].calcangle();
865         sort(hp,hp+n);
866         unique();
867         que[st=0] = 0;
868         que[ed=1] = 1;
869         p[1] = hp[0].crosspoint(hp[1]);
870         for(int i = 2;i < n;i++){
871             while(st<ed && sgn((hp[i].e-hp[i].s)^(
    ↪ p[ed]-hp[i].s))<0)ed--;
872             while(st<ed && sgn((hp[i].e-hp[i].s)^(
    ↪ p[st+1]-hp[i].s))<0)st++;
873             que[++ed] = i;
874             if(hp[i].parallel(hp[que[ed-1]]))return
    ↪ false;
875             p[ed]=hp[i].crosspoint(hp[que[ed-1]]);
876         }
877         while(st<ed &&
    ↪ sgn((hp[que[st]].e-hp[que[st]].s)^(
    ↪ p[ed]-hp[que[st]].s))<0)ed--;
878         while(st<ed &&
    ↪ sgn((hp[que[ed]].e-hp[que[ed]].s)^(
    ↪ p[st+1]-hp[que[ed]].s))<0)st++;
879         if(st+1==ed)return false;
880         return true;

```

```

881 }
882 //得到最后半平面交得到的凸多边形`
883 //需要先调用 halfplaneinsert() 且返回 true`
884 void getconvex(polygon &con){
885     p[st] = hp[que[st]].crosspoint(hp[que[ed]]);
886     con.n = ed-st+1;
887     for(int j = st,i = 0;j <= ed;i++,j++)
888         con.p[i] = p[j];
889 }
890 };
891 //*****
892
893 const int maxn = 1010;
894 struct circles{
895     circle c[maxn];
896     double ans[maxn];//ans[i] 表示被覆盖了 i 次的面积`
897     double pre[maxn];
898     int n;
899     circles(){
900     void add(circle cc){
901         c[n++] = cc;
902     }
903     //x 包含在 y 中`
904     bool inner(circle x,circle y){
905         if(x.relationcircle(y) != 1)return 0;
906         return sgn(x.r-y.r)<=0?1:0;
907     }
908     //圆的面积并去掉内含的圆
909     void init_or(){
910         bool mark[maxn] = {0};
911         int i,j,k=0;
912         for(i = 0;i < n;i++){
913             for(j = 0;j < n;j++){
914                 if(i != j && !mark[j]){
915                     if( (c[i]==c[j])||inner(c[i],c[j])
916                         ↪ )break;
917                 }
918                 if(j < n)mark[i] = 1;
919             }
920             for(i = 0;i < n;i++)
921                 if(!mark[i])
922                     c[k++] = c[i];
923             n = k;
924         }
925     //圆的面积交去掉内含的圆`
926     void init_add(){
927         int i,j,k;
928         bool mark[maxn] = {0};
929         for(i = 0;i < n;i++){
930             for(j = 0;j < n;j++){
931                 if(i != j && !mark[j]){
932                     if( (c[i]==c[j])||inner(c[j],c[i])
933                         ↪ )break;
934                 }
935                 if(j < n)mark[i] = 1;
936             }
937             for(i = 0;i < n;i++)
938                 if(!mark[i])
939                     c[k++] = c[i];
940             n = k;
941         }
942     //半径为 r 的圆，弧度为 th 对应的弓形的面积`
943     double areaarc(double th,double r){
944         return 0.5*r*r*(th-sin(th));
945     }
946     //测试 SPOJVCIRCLES SPOJCIRUT`
947     //SPOJVCIRCLES 求 n 个圆并的面积，需要加上
948     ↪ init_or()去掉重复圆 (否则 WA)`

```

```

946 //SPOJCIRUT 是求被覆盖 k 次的面积，不能加
947 ↪ init_or()
948 //对于求覆盖多少次面积的问题，不能解决相同圆，而且不
949 ↪ 能 init_or()
950 //求多圆面积并，需要 init_or，其中一个目的就是去掉
951 ↪ 相同圆`
952 void getarea(){
953     memset(ans,0,sizeof(ans));
954     vector<pair<double,int>>v;
955     for(int i = 0;i < n;i++){
956         v.clear();
957         v.push_back(make_pair(-pi,1));
958         v.push_back(make_pair(pi,-1));
959         for(int j = 0;j < n;j++){
960             if(i != j){
961                 Point q = (c[j].p - c[i].p);
962                 double ab = q.len(),ac = c[i].r, bc
963                     ↪ = c[j].r;
964                 if(sgn(ab+ac-bc)<=0){
965                     v.push_back(make_pair(-pi,1));
966                     v.push_back(make_pair(pi,-1));
967                     continue;
968                 }
969                 if(sgn(ab+bc-ac)<=0)continue;
970                 if(sgn(ab-ac-bc)>0)continue;
971                 double th = atan2(q.y,q.x), fai =
972                     ↪ acos((ac*ac+ab*ab-bc*bc)/(
973                     ↪ 2.0*ac*ab));
974                 double a0 = th-fai;
975                 if(sgn(a0+pi)<0)a0+=2*pi;
976                 double a1 = th+fai;
977                 if(sgn(a1-pi)>0)a1-=2*pi;
978                 if(sgn(a0-a1)>0){
979                     v.push_back(make_pair(a0,1));
980                     v.push_back(make_pair(pi,-1));
981                     v.push_back(make_pair(-pi,1));
982                     v.push_back(make_pair(a1,-1));
983                 }
984                 else{
985                     v.push_back(make_pair(a0,1));
986                     v.push_back(make_pair(a1,-1));
987                 }
988             }
989         }
990     sort(v.begin(),v.end());
991     int cur = 0;
992     for(int j = 0;j < v.size();j++){
993         if(cur && sgn(v[j].first-pre[cur])){
994             ans[cur] += areaarc(
995                 ↪ v[j].first-pre[cur],c[i].r);
996             ans[cur] += 0.5*(Point(
997                 ↪ c[i].p.x+c[i].r*cos(pre[cur])
998                 ↪ ,c[i].p.y+c[i].r*sin(
999                 ↪ pre[cur]))^Point(
1000                 ↪ c[i].p.x+c[i].r*cos(
1001                 ↪ v[j].first)
1002                 ↪ ,c[i].p.y+c[i].r*sin(
1003                 ↪ v[j].first)));
1004         }
1005         cur += v[j].second;
1006         pre[cur] = v[j].first;
1007     }
1008 }
1009 for(int i = 1;i < n;i++){
1010     ans[i] -= ans[i+1];
1011 }
1012 }

```

## 5.2. 平面最近点对

```

1 const int MAXN = 100010;
2 const double eps = 1e-8;
3 const double INF = 1e20;
4 struct Point{
5     double x,y;
6     void input(){
7         scanf("%lf%lf",&x,&y);
8     }
9 };
10 double dist(Point a,Point b){
11     return sqrt((a.x-b.x)*(a.x-b.x) +
12         ↪ (a.y-b.y)*(a.y-b.y));
13 }
14 Point p[MAXN];
15 Point tmp[100010];
16 bool cmpx(Point a,Point b){
17     return a.x < b.x || (a.x == b.x && a.y < b.y);
18 }
19 bool cmpy(Point a,Point b){
20     return a.y < b.y || (a.y == b.y && a.x < b.x);
21 }
22 double Closest_Pair(int left,int right){
23     double d = INF;
24     if(left == right)return d;
25     if(left+1 == right)return dist(p[left],p[right]);
26     int mid = (left+right)/2;
27     double d1 = Closest_Pair(left,mid);
28     double d2 = Closest_Pair(mid+1,right);
29     d = min(d1,d2);
30     int cnt = 0;
31     for(int i = left;i <= right;i++){
32         if(fabs(p[mid].x - p[i].x) <= d)
33             tmp[cnt++] = p[i];
34     }
35     sort(tmp,tmp+cnt,cmpy);
36     for(int i = 0;i < cnt;i++){
37         for(int j = i+1;j < cnt && tmp[j].y -
38             ↪ tmp[i].y < d;j++){
39             d = min(d,dist(tmp[i],tmp[j]));
40         }
41     }
42     return d;
43 }
44 int main(){
45     int n;
46     while(scanf("%d",&n) == 1 && n){
47         for(int i = 0;i < n;i++)p[i].input();
48         sort(p,p+n,cmpx);
49         printf("%.2lf\n",Closest_Pair(0,n-1));
50     }
51     return 0;
52 }

```

## 5.3. 三维几何

```

1 const double eps = 1e-8;
2 int sgn(double x){
3     if(fabs(x) < eps)return 0;
4     if(x < 0)return -1;
5     else return 1;
6 }
7 struct Point3{
8     double x,y,z;
9     Point3(double _x = 0,double _y = 0,double _z = 0){
10         x = _x;
11         y = _y;
12         z = _z;
13     }
14     void input(){
15         scanf("%lf%lf%lf",&x,&y,&z);

```

```

16     }
17     void output(){
18         scanf("%.2lf %.2lf %.2lf\n",x,y,z);
19     }
20     bool operator ==(const Point3 &b)const{
21         return sgn(x-b.x) == 0 && sgn(y-b.y) == 0 &&
22             ↪ sgn(z-b.z) == 0;
23     }
24     bool operator <(const Point3 &b)const{
25         return sgn(x-b.x)==0?(sgn(y-b.y)==0?sgn(
26             ↪ z-b.z)<0:y<b.y):x<b.x;
27     }
28     double len(){
29         return sqrt(x*x+y*y+z*z);
30     }
31     double len2(){
32         return x*x+y*y+z*z;
33     }
34     double distance(const Point3 &b)const{
35         return sqrt((x-b.x)*(x-b.x)+(y-b.y)*(y-b.y)+(
36             ↪ z-b.z)*(z-b.z));
37     }
38     Point3 operator -(const Point3 &b)const{
39         return Point3(x-b.x,y-b.y,z-b.z);
40     }
41     Point3 operator +(const Point3 &b)const{
42         return Point3(x+b.x,y+b.y,z+b.z);
43     }
44     Point3 operator *(const double &k)const{
45         return Point3(x*k,y*k,z*k);
46     }
47     Point3 operator /(const double &k)const{
48         return Point3(x/k,y/k,z/k);
49     }
50     //点乘
51     double operator *(const Point3 &b)const{
52         return x*b.x+y*b.y+z*b.z;
53     }
54     //叉乘
55     Point3 operator ^(const Point3 &b)const{
56         return Point3(
57             ↪ y*b.z-z*b.y,z*b.x-x*b.z,x*b.y-y*b.x);
58     }
59     double rad(Point3 a,Point3 b){
60         Point3 p = (*this);
61         return acos( ( (a-p)*(b-p) )/
62             ↪ (a.distance(p)*b.distance(p)) );
63     }
64     //变换长度
65     Point3 trunc(double r){
66         double l = len();
67         if(!sgn(l))return *this;
68         r /= l;
69         return Point3(x*r,y*r,z*r);
70     }
71 };
72 struct Line3{
73     Point3 s,e;
74     Line3(){}
75     Line3(Point3 _s,Point3 _e)
76     {
77         s = _s;
78         e = _e;
79     }
80     bool operator ==(const Line3 v)
81     {
82         return (s==v.s)&&(e==v.e);
83     }
84 }

```

```

79     }
80     void input()
81     {
82         s.input();
83         e.input();
84     }
85     double length()
86     {
87         return s.distance(e);
88     }
89     //点到直线距离
90     double dispointtoline(Point3 p)
91     {
92         return ((e-s)^(p-s)).len()/s.distance(e);
93     }
94     //点到线段距离
95     double dispointtoseg(Point3 p)
96     {
97         if(sgn((p-s)*(e-s)) < 0 || sgn((p-e)*(s-e)) <
            ↪ 0)
98             return min(p.distance(s),e.distance(p));
99         return dispointtoline(p);
100     }
101     //返回点 p 在直线上的投影
102     Point3 lineprog(Point3 p)
103     {
104         return s + (
            ↪ ((e-s)*((e-s)*(p-s)))/((e-s).len2()) );
105     }
106     //p 绕此向量逆时针 arg 角度
107     Point3 rotate(Point3 p,double ang)
108     {
109         if(sgn(((s-p)^(e-p)).len()) == 0)return p;
110         Point3 f1 = (e-s)^(p-s);
111         Point3 f2 = (e-s)^(f1);
112         double len = ((s-p)^(e-p)).len()/s.distance(e);
113         f1 = f1.trunc(len); f2 = f2.trunc(len);
114         Point3 h = p+f2;
115         Point3 pp = h+f1;
116         return h + ((p-h)*cos(ang)) +
            ↪ ((pp-h)*sin(ang));
117     }
118     //点在直线上
119     bool pointonseg(Point3 p)
120     {
121         return sgn( ((s-p)^(e-p)).len() ) == 0 &&
            ↪ sgn((s-p)*(e-p)) == 0;
122     }
123 };
124 struct Plane
125 {
126     Point3 a,b,c,o;//平面上的三个点,以及法向量
127     Plane(){}
128     Plane(Point3 _a,Point3 _b,Point3 _c)
129     {
130         a = _a;
131         b = _b;
132         c = _c;
133         o = pvec();
134     }
135     Point3 pvec()
136     {
137         return (b-a)^(c-a);
138     }
139     //ax+by+cz+d = 0
140     Plane(double _a,double _b,double _c,double _d)
141     {
142         o = Point3(_a,_b,_c);
143         if(sgn(_a) != 0)

```

```

144         a = Point3((-_d-_c-_b)/_a,1,1);
145     else if(sgn(_b) != 0)
146         a = Point3(1,(-_d-_c-_a)/_b,1);
147     else if(sgn(_c) != 0)
148         a = Point3(1,1,(-_d-_a-_b)/_c);
149     }
150     //点在平面上的判断
151     bool pointonplane(Point3 p)
152     {
153         return sgn((p-a)*o) == 0;
154     }
155     //两平面夹角
156     double angleplane(Plane f)
157     {
158         return acos(o*f.o)/(o.len()*f.o.len());
159     }
160     //平面和直线的交点,返回值是交点个数
161     int crossline(Line3 u,Point3 &p)
162     {
163         double x = o*(u.e-a);
164         double y = o*(u.s-a);
165         double d = x-y;
166         if(sgn(d) == 0)return 0;
167         p = ((u.s*x)-(u.e*y))/d;
168         return 1;
169     }
170     //点到平面最近点 (也就是投影)
171     Point3 pointtoplane(Point3 p)
172     {
173         Line3 u = Line3(p,p+o);
174         crossline(u,p);
175         return p;
176     }
177     //平面和平面的交线
178     int crossplane(Plane f,Line3 &u)
179     {
180         Point3 oo = o^f.o;
181         Point3 v = o^oo;
182         double d = fabs(f.o*v);
183         if(sgn(d) == 0)return 0;
184         Point3 q = a + (v*(f.o*(f.a-a))/d);
185         u = Line3(q,q+oo);
186         return 1;
187     }
188 };

```

## 5.4. 三维凸包

```

1 const double eps = 1e-8;
2 const int MAXN = 550;
3 int sgn(double x){
4     if(fabs(x) < eps)return 0;
5     if(x < 0)return -1;
6     else return 1;
7 }
8 struct Point3{
9     double x,y,z;
10    Point3(double _x = 0, double _y = 0, double _z =
        ↪ 0){
11        x = _x;
12        y = _y;
13        z = _z;
14    }
15    void input(){
16        scanf("%lf%lf%lf",&x,&y,&z);
17    }
18    bool operator ==(const Point3 &b)const{
19        return sgn(x-b.x) == 0 && sgn(y-b.y) == 0 &&
            ↪ sgn(z-b.z) == 0;

```

```

20     }
21     double len(){
22         return sqrt(x*x+y*y+z*z);
23     }
24     double len2(){
25         return x*x+y*y+z*z;
26     }
27     double distance(const Point3 &b)const{
28         return sqrt((x-b.x)*(x-b.x)+(y-b.y)*(y-b.y)+(
29             ↪ z-b.z)*(z-b.z));
30     }
31     Point3 operator -(const Point3 &b)const{
32         return Point3(x-b.x,y-b.y,z-b.z);
33     }
34     Point3 operator +(const Point3 &b)const{
35         return Point3(x+b.x,y+b.y,z+b.z);
36     }
37     Point3 operator *(const double &k)const{
38         return Point3(x*k,y*k,z*k);
39     }
40     Point3 operator /(const double &k)const{
41         return Point3(x/k,y/k,z/k);
42     }
43     //点乘
44     double operator *(const Point3 &b)const{
45         return x*b.x + y*b.y + z*b.z;
46     }
47     //叉乘
48     Point3 operator ^(const Point3 &b)const{
49         return Point3(
50             ↪ y*b.z-z*b.y,z*b.x-x*b.z,x*b.y-y*b.x);
51     }
52 };
53 struct CH3D{
54     struct face{
55         //表示凸包一个面上的三个点的编号
56         int a,b,c;
57         //表示该面是否属于最终的凸包上的面
58         bool ok;
59     };
60     //初始顶点数
61     int n;
62     Point3 P[MAXN];
63     //凸包表面的三角形数
64     int num;
65     //凸包表面的三角形
66     face F[8*MAXN];
67     int g[MAXN][MAXN];
68     //叉乘
69     Point3 cross(const Point3 &a,const Point3 &b,const
70         ↪ Point3 &c){
71         return (b-a)^(c-a);
72     }
73     //^ 三角形面积 *2^
74     double area(Point3 a,Point3 b,Point3 c){
75         return ((b-a)^(c-a)).len();
76     }
77     //^ 四面体有向面积 *6^
78     double volume(Point3 a,Point3 b,Point3 c,Point3 d){
79         return ((b-a)^(c-a))*(d-a);
80     }
81     //^ 正: 点在面同向^
82     double dblcmp(Point3 &p,face &f){
83         Point3 p1 = P[f.b] - P[f.a];
84         Point3 p2 = P[f.c] - P[f.a];
85         Point3 p3 = p - P[f.a];
86         return (p1^p2)*p3;
87     }
88     void deal(int p,int a,int b){
89         int f = g[a][b];
90         face add;
91         if(F[f].ok){
92             if(dblcmp(P[p],F[f]) > eps)
93                 dfs(p,f);
94             else {
95                 add.a = b;
96                 add.b = a;
97                 add.c = p;
98                 add.ok = true;
99                 g[p][b] = g[a][p] = g[b][a] = num;
100                 F[num++] = add;
101             }
102         }
103     }
104     //递归搜索所有应该从凸包内删除的面
105     void dfs(int p,int now){
106         F[now].ok = false;
107         deal(p,F[now].b,F[now].a);
108         deal(p,F[now].c,F[now].b);
109         deal(p,F[now].a,F[now].c);
110     }
111     bool same(int s,int t){
112         Point3 &a = P[F[s].a];
113         Point3 &b = P[F[s].b];
114         Point3 &c = P[F[s].c];
115         return fabs(volume(a,b,c,P[F[t].a])) < eps &&
116             fabs(volume(a,b,c,P[F[t].b])) < eps &&
117             fabs(volume(a,b,c,P[F[t].c])) < eps;
118     }
119     //构建三维凸包
120     void create(){
121         num = 0;
122         face add;
123
124         //*****
125         //此段是为了保证前四个点不共面
126         bool flag = true;
127         for(int i = 1;i < n;i++){
128             if(!(P[0] == P[i])){
129                 swap(P[1],P[i]);
130                 flag = false;
131                 break;
132             }
133         }
134         if(flag)return;
135         flag = true;
136         for(int i = 2;i < n;i++){
137             if( ((P[i]-P[0])^(P[1]-P[0])).len() > eps
138                 ↪ ){
139                 swap(P[2],P[i]);
140                 flag = false;
141                 break;
142             }
143         }
144         if(flag)return;
145         flag = true;
146         for(int i = 3;i < n;i++){
147             if(fabs(
148                 ↪ ((P[i]-P[0])^(P[2]-P[0]))*(P[1]-P[0]) )
149                 ↪ > eps){
150                 swap(P[3],P[i]);
151                 flag = false;
152                 break;
153             }
154         }
155         if(flag)return;
156         //*****

```

```

151
152     for(int i = 0;i < 4;i++){
153         add.a = (i+1)%4;
154         add.b = (i+2)%4;
155         add.c = (i+3)%4;
156         add.ok = true;
157         if(dblcmp(P[i],add) > 0)swap(add.b,add.c);
158         g[add.a][add.b] = g[add.b][add.c] =
            ↳ g[add.c][add.a] = num;
159         F[num++] = add;
160     }
161     for(int i = 4;i < n;i++){
162         for(int j = 0;j < num;j++){
163             if(F[j].ok && dblcmp(P[i],F[j]) > eps){
164                 dfs(i,j);
165                 break;
166             }
167             int tmp = num;
168             num = 0;
169             for(int i = 0;i < tmp;i++){
170                 if(F[i].ok)
171                     F[num++] = F[i];
172             }
173             //表面积
174             //^测试: HDU3528^
175             double area(){
176                 double res = 0;
177                 if(n == 3){
178                     Point3 p = cross(P[0],P[1],P[2]);
179                     return p.len()/2;
180                 }
181                 for(int i = 0;i < num;i++){
182                     res += area(P[F[i].a],P[F[i].b],P[F[i].c]);
183                 }
184                 return res/2.0;
185             }
186             double volume(){
187                 double res = 0;
188                 Point3 tmp = Point3(0,0,0);
189                 for(int i = 0;i < num;i++){
190                     res += volume(
191                         ↳ tmp,P[F[i].a],P[F[i].b],P[F[i].c]);
192                 }
193             //表面三角形个数
194             int triangle(){
195                 return num;
196             }
197             //表面多边形个数
198             //^测试: HDU3662^
199             int polygon(){
200                 int res = 0;
201                 for(int i = 0;i < num;i++){
202                     bool flag = true;
203                     for(int j = 0;j < i;j++){
204                         if(same(i,j)){
205                             flag = 0;
206                             break;
207                         }
208                     }
209                     res += flag;
210                 }
211                 return res;
212             }
213             //重心
214             //^测试: HDU4273^
215             Point3 barycenter(){
216                 Point3 ans = Point3(0,0,0);
217                 Point3 o = Point3(0,0,0);
218                 double all = 0;
219                 for(int i = 0;i < num;i++){

```

```

218         double vol = volume(
219             ↳ o,P[F[i].a],P[F[i].b],P[F[i].c]);
220         ans = ans +
221             ↳ ((o+P[F[i].a]+P[F[i].b]+P[F[i].c])
222             ↳ /4.0)*vol);
223         all += vol;
224     }
225     ans = ans/all;
226     return ans;
227 }
228 //点到面的距离
229 //^测试: HDU4273^
230 double ptoface(Point3 p,int i){
231     double tmp1 = fabs(volume(
232         ↳ P[F[i].a],P[F[i].b],P[F[i].c],p));
233     double tmp2 = ((P[F[i].b]-P[F[i].a])^(
234         ↳ P[F[i].c]-P[F[i].a])).len();
235     return tmp1/tmp2;
236 }
237 }
238 CH3D hull;
239 int main()
240 {
241     while(scanf("%d",&hull.n) == 1){
242         for(int i = 0;i < hull.n;i++)hull.P[i].input();
243         hull.create();
244         Point3 p = hull.barycenter();
245         double ans = 1e20;
246         for(int i = 0;i < hull.num;i++){
247             ans = min(ans,hull.ptoface(p,i));
248         }
249         printf("%.3lf\n",ans);
250     }
251     return 0;
252 }

```

## 5.5. 几何公式

### • 三角形

半周长  $p = \frac{a+b+c}{2}$

面积  $S = \frac{1}{2}aH_a = \frac{1}{2}ab \cdot \sin C = \sqrt{p(p-a)(p-b)(p-c)} = pr = \frac{abc}{4R}$

中线长  $M_a = \frac{1}{2}\sqrt{2(b^2+c^2)-a^2} = \frac{1}{2}\sqrt{b^2+c^2+2bc \cdot \cos A}$

角平分线长  $T_a = \frac{\sqrt{bc((b+c)^2-a^2)}}{b+c} = \frac{2bc}{b+c} \cos \frac{A}{2}$

高  $H_a = b \sin C = \sqrt{b^2 - (\frac{a^2+b^2-c^2}{2a})^2}$

内切圆半径  $r = \frac{S}{p} = 4R \sin \frac{A}{2} \sin \frac{B}{2} \sin \frac{C}{2} = \sqrt{\frac{(p-a)(p-b)(p-c)}{p}}$

$p \tan \frac{A}{2} \tan \frac{B}{2} \tan \frac{C}{2} = \frac{abc}{4S} = \frac{a}{2 \sin A}$

外接圆半径  $R = \frac{abc}{4S} = \frac{a}{2 \sin A}$

旁切圆半径  $r_A = \frac{-a+b+c}{2 \sin A}$

重心  $(\frac{x_1+x_2+x_3}{3}, \frac{y_1+y_2+y_3}{3})$

外心  $(\frac{\begin{vmatrix} x_1^2+y_1^2 & y_1 & 1 \\ x_2^2+y_2^2 & y_2 & 1 \\ x_3^2+y_3^2 & y_3 & 1 \end{vmatrix}}{\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}}, \frac{\begin{vmatrix} x_1 & x_1^2+y_1^2 & 1 \\ x_2 & x_2^2+y_2^2 & 1 \\ x_3 & x_3^2+y_3^2 & 1 \end{vmatrix}}{\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}})$

内心  $(\frac{ax_1+bx_2+cx_3}{a+b+c}, \frac{ay_1+by_2+cy_3}{a+b+c})$

垂心  $(\frac{\begin{vmatrix} x_2x_3+y_2y_3 & 1 & y_1 \\ x_3x_1+y_3y_1 & 1 & y_2 \\ x_1x_2+y_1y_2 & 1 & y_3 \end{vmatrix}}{\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}}, \frac{\begin{vmatrix} x_2x_3+y_2y_3 & x_1 & 1 \\ x_3x_1+y_3y_1 & x_2 & 1 \\ x_1x_2+y_1y_2 & x_3 & 1 \end{vmatrix}}{\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}})$

旁心  $(\frac{-ax_1+bx_2+cx_3}{-a+b+c}, \frac{-ay_1+by_2+cy_3}{-a+b+c})$

Trilinear coordinates:  $\frac{ax}{ax+by+cz}A + \frac{by}{ax+by+cz}B + \frac{cz}{ax+by+cz}C$

$x, y, z$  分别代表点  $P$  到边的距离

Fermat point:  $x : y : z = \csc(A + \frac{\pi}{3}) : \csc(B + \frac{\pi}{3}) : \csc(C + \frac{\pi}{3})$

### • 圆

弧长  $l = rA$

弦长  $a = 2\sqrt{2hr - h^2} = 2r \cdot \sin \frac{A}{2}$

弓形高  $h = r - \sqrt{r^2 - \frac{a^2}{4}} = r(1 - \cos \frac{A}{2})$

扇形面积  $S_1 = \frac{1}{2}lr = \frac{1}{2}Ar^2$

弓形面积  $S_2 = \frac{1}{2}r^2(A - \sin A)$

### • Circles of Apollonius

已知三个两两相切的圆，半径为  $r_1, r_2, r_3$

与它们外切的圆半径为  $\frac{r_1 r_2 r_3}{r_1 r_2 + r_2 r_3 + r_3 r_1 - 2\sqrt{r_1 r_2 r_3 (r_1 + r_2 + r_3)}}$

与它们内切的圆半径为  $\frac{r_1 r_2 r_3}{r_1 r_2 + r_2 r_3 + r_3 r_1 + 2\sqrt{r_1 r_2 r_3 (r_1 + r_2 + r_3)}}$

### • 棱台

体积  $V = \frac{1}{3}h(A_1 + A_2 + \sqrt{A_1 A_2})$

正棱台侧面积  $S = \frac{1}{2}(p_1 + p_2)l$ ,  $l$  为侧高

### • 球

体积  $V = \frac{4}{3}\pi r^3$

表面积  $S = 4\pi r^2$

### • 球台

侧面积  $S = 2\pi rh$

体积  $V = \frac{1}{6}\pi h(3(r_1^2 + r_2^2) + h^2)$

### • 球扇形

球面面积  $S = 2\pi rh$

体积  $V = \frac{2}{3}\pi r^2 h = \frac{2}{3}\pi r^3 h(1 - \cos \varphi)$

### • 球面三角形

考虑单位球上的球面三角形， $a, b, c$  表示三边长（弧所对球心角）， $A, B, C$  表示三角大小（切线夹角）

余弦定理  $\cos a = \cos b \cdot \cos c + \sin b \cdot \sin c \cdot \cos A$

正弦定理  $\frac{\sin A}{\sin a} = \frac{\sin B}{\sin b} = \frac{\sin C}{\sin c}$

球面面积  $S = A + B + C - \pi$

### • 四面体

体积  $V = \frac{1}{6} |\vec{AB} \cdot (\vec{AC} \times \vec{AD})|$

## Ch6. Others

### 6.1. 模拟退火

```
1 inline db rand01(){return rand()/2147483647.0;}
2 inline db randp(){return (rand()&1?1:-1)*rand01();}
3 inline db f(db x){
4     if (maxx<fans){fans=maxx;ans_x=x;}
5     return maxx;
6 }
7 int main(){
8     srand(233);
9     db x=0,fnow=f(x);
10    fans=1e30;
11    for (db T=1e4;T>1e-4;T*=0.997){
12        db nx=x+randp()*T,fnext=f(nx),delta=fnext-fnow;
13        if (delta<1e-9||exp(-delta/T)>rand01()){
14            x=nx;
15            fnow=fnext;
16        }
17    }
18    return 0;
19 }
```

### 6.2. 莫队

#### normal

把整个区间  $[1, n]$  分成若干块，以询问的左端点所在块为第一关键字，以询问的右端点大小为第二关键字，对询问进行排序。

时间复杂度  $O(n\sqrt{m})$

#### with modify

对询问进行排序，每个询问除了左端点和右端点还要记录这次询问是在第几次修改之后（时间），以左端点所在块为第一关键字，以右端点所在块为第二关键字，以时间为第三关键字进行排序。

时间复杂度  $O(n^{\frac{5}{3}})$

#### on tree

这里我们设  $st[i]$  表示访问到  $i$  时加入欧拉序的时间， $ed[i]$  表示回溯经过  $i$  时加入欧拉序的时间

不妨设  $st[x] < st[y]$ （也就是先访问  $x$ ，再访问  $y$ ）分情况讨论  
若  $lca(x, y) = x$ ，这时  $x, y$  在一条链上，那么  $st[x]$  到  $st[y]$  这段区间中，有的点出现了两次，有的点没有出现，这些点都是对答案没有贡献的，我们只需要统计出现过 1 次的点就好

若  $lca(x, y) \neq x$ ，此时  $x, y$  位于不同的子树内，我们只需要按照上面的方法统计  $ed[x]$  到  $st[y]$  这段区间内的点。我们没有统计  $lca$ ，因此我们需要特判  $lca$

时间复杂度  $O(n\sqrt{m})$

### 6.3. 手动开栈

```
1 register char *_sp __asm__("rsp");
2 int main(){
```

```
3     const int size=400<<20; //400 MB
4     static char *sys, *mine(new char[size]+size-4096);
5     sys = _sp;
6     _sp=mine;
7     _main();
8     _sp=sys;
9     return 0;
10 }
```

### 6.4. O3

```
1 __attribute__((optimize("-O3"))) void f(){}
```

### 6.5. STL 释放内存

```
1 template<typename T>
2 __inline void clear(T &container){
3     container.clear();
4     T(container).swap(container);
5 }
```

### 6.6. Java

```
1 import java.util.*;
2 import java.math.*;
3 import java.io.*;
4 public class Main{
5     static Scanner sc = new Scanner(System.in);
6     public static void main(String[] args){
7         BigInteger A = read();    //高精度小数:
8         ↪ BigDecimal
9
10        //转 double:
11        ↪ A.doubleValue()
12
13        //BigDecimal 转 BigInteger:
14        ↪ A.toBigInteger()
15
16        //BigInteger 转 BigDecimal: A = new
17        ↪ BigDecimal(B)
18
19        BigInteger B = read();
20        BigInteger C = A.subtract(B);
21        //add,subtract,multiply,divide,mod
22        //gcd(BigInteger),pow(int)
23        //A.compareTo(B) 返回 A-B 的符号 (-1、0、
24        ↪ 1)
25        System.out.println(C);
26    }
27    public static BigInteger read(){
28        String a = sc.next();    //读一行:
29        ↪ nextLine()
30        BigInteger A = new BigInteger(a);
31        return A;
32    }
33 }
```



```

25 /*
26 数组
27 int a[] = new int[100];
28 长度: 数组 a.length 字符串 a.length()
29
30 保留小数位数
31 double a=1.0;
32 System.out.println(String.format("%.2f",a));
33
34 文件 IO
35 InputStream in = new FileInputStream("test.in");
36 Scanner cin = new Scanner(in);
37 PrintStream out = new PrintStream("test.out");
38
39 重定向
40 System.setIn(in);
41 System.setOut(out);
42 */

```

## 6.7. vimrc

```

1 imap [ []<LEFT>
2 imap ( ()<LEFT>
3 imap { {}<LEFT>
4 inoremap " ""<LEFT>
5 inoremap ' ''<LEFT>

```

```

6 set mouse=a
7 set number
8 set ts=4
9 set autoindent
10 imap <F1> <ESC>:w %<cr>i
11 imap <F3> <ESC>:w %<cr>:!g++ % -o %<
   ↪ -std=c++14<cr>:!./%< < in > out<cr>:!gedit out<cr>i
12 imap <F5> <ESC>:!./%< < in > out<cr>:!gedit out<cr>i

```

## 6.8. 快速读入输出

```

1 ll in()
2 {
3     ll re=0,f=1;char x=getchar();
4     while(x<'0' || x>'9'){if(x=='-')f=-1;x=getchar();}
5     while(x>='0' && x<='9')re=re*10+x-48,x=getchar();
6     return re*f;
7 }
8 void out(ll x)
9 {
10     if(x<0)putchar('-'),x=-x;
11     if(!x){putchar('0');return;}
12     char z[21],ct=0;
13     while(x){z[++ct]=x%10+48,x/=10;}
14     while(ct){putchar(z[ct]);--ct;}
15 }

```