

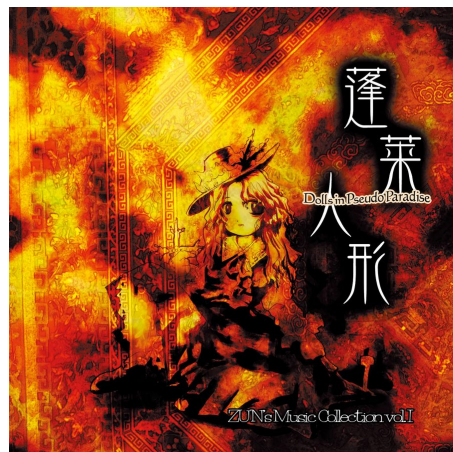
Our village of honest men originally consisted of only eight people.  
We all picked up and moved to a mountain in the east. Two years of honest and boring daily life passed us by.  
One day, one of us found a little hole by a peach tree.  
Yes, after that we wandered into this paradise.  
And right away, I quit being human.

---

— *Dolls in Pseudo Paradise*

## 蓬莱人形算法模板库

REFERENCE DOCUMENT for *Dolls in Pseudo Paradise*



2024-2025

Harbin Institute of Technology

目录	5 网络流	19	8 字符串	34
1 动态规划	5.1 费用流	19	8.1 AC 自动机	34
1.1 多重背包	5.2 最小割树	20	8.2 扩展 KMP	35
1.2 树形背包	5.3 最大流	20	8.3 Manacher	35
1.3 动态动态规划 1	5.4 上下界费用流	21	8.4 回文自动机	35
1.4 插头 dp	5.5 上下界最大流	21	8.5 后缀平衡树	35
1.5 斜率优化	6 数学	22	8.6 后缀数组 (倍增)	35
2 数据结构	6.1 线性代数	22	8.7 后缀数组 (SAIS)	36
2.1 平衡树	6.2 大步小步	25	8.8 广义后缀自动机 (离线)	36
2.2 珂朵莉树	6.3 中国剩余定理	25	8.9 广义后缀自动机 (在线)	37
2.3 可并堆	6.4 狄利克雷前缀和	25	8.10 后缀自动机	37
2.4 线性基	6.5 万能欧几里得	26	9 计算几何	37
2.5 Link Cut 树	6.6 扩展欧几里得	26	9.1 二维凸包	37
2.6 线段树	6.7 快速离散对数	26	9.2 最小圆覆盖	39
2.7 根号数据结构	6.8 快速最大公约数	26	9.3 最左转线	39
3 树论	6.9 原根	27	9.4 二维基础	41
3.1 点分树	6.10 快速乘法逆元 (离线)	27	10 其他	42
3.2 长链剖分	6.11 快速乘法逆元 (在线)	27	10.1 笛卡尔树	42
3.3 重链剖分	6.12 拉格朗日插值	28	10.2 CDQ 分治	42
3.4 树哈希	6.13 min-max 容斥	28	10.3 自适应辛普森	43
3.5 Prufer 序列	6.14 Barrett 取模	28	10.4 模拟退火	43
3.6 虚树	6.15 Pollard's Rho	28	10.5 伪随机生成	43
4 图论	6.16 polya 定理	29	11 header	43
4.1 仙人掌	6.17 min25 筛	29		
4.2 三元环计数	6.18 杜教筛	30	1 动态规划	
4.3 四元环计数	6.19 PN 筛	31	1.1 多重背包	
4.4 基环树	6.20 常用数表	32	1.1.1 用法	
4.5 2-SAT	6.21 二次剩余	32	$n$ 个物品, $m$ 容量背包, 第 $i$ 个物品重量为 $w_i$ 价值为 $v_i$ 共有 $c_i$ 个, 计算不超过容量的情况下最多拿多少价值的物品。	
4.6 割点	6.22 单位根反演	32		
4.7 边双连通分量	7 多项式	32		
4.8 点双连通分量	7.1 NTT 全家桶	32		
4.9 强连通分量	7.2 FWT 全家桶	33		
	7.3 任意模数 NTT	34		

```
1 #include "../header.cpp"
2 int F[MAXN];
3 int main(){
4     int n, m; cin >> n >> m;
5     for(int i = 1; i <= n; ++ i){
```

```

6   int w, v, c; cin >> w >> v >> c;
7   // w: value, v: volume, c: count
8   for(int j = 0; j < v; ++j){
9       deque <tuple<int, int> > Q;
10      for(int k = 0; j + k * v ≤ m; ++k)
11          {
12              int x = j + k * v;
13              int f = F[x] - (x / v) * w;
14              while(!Q.empty() && get<0>(Q.back()) ≤ f)
15                  Q.pop_back();
16              Q.push_back({f, x});
17              while(!Q.empty() && get<1>(Q.front()) < x - c * v)
18                  Q.pop_front();
19              F[x] = get<0>(Q.front()) + (x / v) * w;
20          }
21      }
22      cout << F[m] << endl;
23      return 0;
24  }

```

## 1.2 树形背包

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  typedef long long i64;
4  const int MAXN = 2e3 + 3;
5  vector<int> E[MAXN];
6  int W[MAXN];
7  int F[MAXN][MAXN], S[MAXN];
8  void dfs(int u, int f){
9      F[u][1] = W[u], S[u] = 1;
10     for(auto &v : E[u]) if(v ≠ f){
11         dfs(v, u);
12         for(int i = S[u]; i ≥ 1; --i)
13             for(int j = S[v]; j ≥ 1; --j)
14                 F[u][i + j] = max(F[u][i + j], F[u][i]
15                                     + F[v][j]);
16         S[u] += S[v];
17     }
18 }
19 int main(){
20     int n, m;
21     cin >> n >> m;
22     for(int i = 1; i ≤ n; ++i){
23         int f;
24         cin >> f >> W[i];
25         E[f].push_back(i);

```

```

26     dfs(0, 0);
27     cout << F[0][m + 1] << endl;
28     return 0;
29 }

```

## 1.3 动态动态规划 1

### 1.3.1 例题

给定一棵  $n$  个点的树，点有点权，求最大独立集。  $m$  次修改，每次把  $x$  的权值修改成  $y$ 。

```

1  #include "../header.cpp"
2  int W[MAXN];
3  struct Mat{ int M[2][2]; };
4  struct Vec{ int V[2]; };
5  Mat operator *(const Mat &a, const Mat &b){
6      Mat c;
7      c.M[0][0] = max(a.M[0][0] + b.M[0][0], a.M
8          [0][1] + b.M[1][0]);
9      c.M[0][1] = max(a.M[0][0] + b.M[0][1], a.M
10         [0][1] + b.M[1][1]);
11      c.M[1][0] = max(a.M[1][0] + b.M[0][0], a.M
12         [1][1] + b.M[1][0]);
13      c.M[1][1] = max(a.M[1][0] + b.M[0][1], a.M
14         [1][1] + b.M[1][1]);
15      return c;
16 }
17 Vec operator *(const Mat &a, const Vec &v){
18     Vec r;
19     r.V[0] = max(a.M[0][0] + v.V[0], a.M[0][1] +
20                 v.V[1]);
21     r.V[1] = max(a.M[1][0] + v.V[0], a.M[1][1] +
22                 v.V[1]);
23     return r;
24 }
25 namespace Gra{
26     vector<int> E[MAXN];
27     int G[MAXN], S[MAXN], D[MAXN], T[MAXN], F[
28         MAXN];
29     int X[MAXN], Y[MAXN];
30     int H[MAXN][2];
31     int K[MAXN][2];
32     struct Mat M[MAXN];
33     void dfs1(int u, int f){
34         S[u] = 1;
35         F[u] = f;
36         for(auto &v : E[u]) if(v ≠ f){
37             dfs1(v, u);
38             S[u] += S[v];
39             if(S[v] > S[G[u]]) G[u] = v;

```

```

34     }
35     int o;
36     void dfs2(int u, int f){
37         if(u = G[f])
38             X[u] = X[f];
39         else
40             X[u] = u;
41         H[u][0] = H[u][1] = 0;
42         K[u][0] = K[u][1] = 0;
43         const int &g = G[u];
44         D[u] = ++o;
45         T[o] = u;
46         if(g){
47             dfs2(g, u);
48             Y[u] = Y[g];
49             K[u][0] += max(K[g][0], K[g][1]);
50             K[u][1] += K[g][0];
51         } else {
52             Y[u] = u;
53         }
54         for(auto &v : E[u]) if(v ≠ f && v ≠ g){
55             dfs2(v, u);
56             H[u][0] += max(K[v][0], K[v][1]);
57             H[u][1] += K[v][0];
58         }
59         M[u].M[0][0] = H[u][0];
60         M[u].M[0][1] = H[u][0];
61         M[u].M[1][0] = H[u][1] + W[u];
62         M[u].M[1][1] = -INF;
63         K[u][0] += H[u][0];
64         K[u][1] += H[u][1] + W[u];
65     }
66 }
67 namespace Seg{
68     const int SIZ = 4e5 + 3;
69     struct Mat M[SIZ];
70     #define lc(t) (t << 1)
71     #define rc(t) (t << 1 | 1)
72     void pushup(int t, int a, int b){
73         M[t] = M[lc(t)] * M[rc(t)];
74     }
75     void build(int t, int a, int b){
76         if(a = b){
77             M[t] = Gra :: M[Gra :: T[a]];
78         } else {
79             int c = a + b >> 1;
80             build(lc(t), a, c);
81             build(rc(t), c + 1, b);
82             pushup(t, a, b);
83         }
84     }
85     void modify(int t, int a, int b, int p,

```

```

86  const Mat &w){
87  if(a == b){
88      M[t] = w;
89  } else {
90      int c = a + b >> 1;
91      if(p ≤ c) modify(lc(t), a, c, p, w);
92      else modify(rc(t), c + 1, b, p, w);
93      pushup(t, a, b);
94  }
95  Mat query(int t, int a, int b, int l, int r)
96  {
97      if(l ≤ a && b ≤ r){
98          return M[t];
99      } else {
100          int c = a + b >> 1;
101          if(r ≤ c) return query(lc(t), a, c, l, r); else
102          if(l > c) return query(rc(t), c + 1, b, l, r); else
103          return query(lc(t), a, c, l, r) *
104          query(rc(t), c + 1, b, l, r);
105      }
106  }
107  int qread();
108  int main(){
109      int n = qread(), m = qread();
110      up(1, n, i)
111      W[i] = qread();
112      up(2, n, i){
113          int u = qread(), v = qread();
114          Gra :: E[u].push_back(v);
115          Gra :: E[v].push_back(u);
116      }
117      Gra :: dfs1(1, 0);
118      Gra :: dfs2(1, 0);
119      Seg :: build(1, 1, n);
120      Vec v0;
121      v0.V[0] = v0.V[1] = 0;
122      up(1, m, i){
123          using namespace Gra;
124          int x = qread(), y = qread();
125          W[x] = y;
126          int u = x;
127          while(u ≠ 0){
128              const int &v = X[u];
129              const int &f = F[v];
130              M[u].M[0][0] = H[u][0];
131              M[u].M[0][1] = H[u][0];
132              M[u].M[1][0] = H[u][1] + W[u];
133              M[u].M[1][1] = -INF;

```

```

134  const Vec p = Seg :: query(1, 1, n, D[v
135  ], D[Y[u]]) * v0;
136  Seg :: modify(1, 1, n, D[u], M[u]);
137  const Vec q = Seg :: query(1, 1, n, D[v
138  ], D[Y[u]]) * v0;
139  if(f ≠ 0){
140      H[f][0] = H[f][0] - max(p.V[0], p.V
141      [1]) + max(q.V[0], q.V[1]);
142      H[f][1] = H[f][1] - p.V[0] + q.V[0];
143      u = f;
144  }
145  Vec v1 = Seg :: query(1, 1, n, D[1], D[Y
146  [1]]) * v0;
147  printf("%d\n", max(v1.V[0], v1.V[1]));
148  }
149  return 0;
150  }

```

## 1.4 插头 dp

### 1.4.1 例题

给出  $n \times m$  的方格，有些格子不能铺线，其它格子必须铺，形成一个闭合回路。问有多少种铺法？

```

1  #include "../header.cpp"
2  namespace HashT{
3      const int SIZ = 19999997;
4      int H[SIZ], V[SIZ], N[SIZ], t;
5      bool F[SIZ];
6      i64 W[SIZ];
7      void add(int u, int v, bool f, i64 w){
8          V[++ t] = v, N[t] = H[u], F[t] = f, W[t] =
9          w, H[u] = t;
10     }
11     i64& find(int u, bool f){
12         for(int p = H[u % SIZ]; p; p = N[p])
13             if(V[p] == u && F[p] == f)
14                 return W[p];
15         add(u % SIZ, u, f, 0);
16         return W[t];
17     }
18     char S[MAXN][MAXN];
19     int qread();
20     int n, m;
21     vector <pair<pair<int, bool>, i64> > M[2];
22     int getp(int s, int p){
23         return (s >> (2 * p - 2)) & 3;
24     }
25     int setw(int s, int p, int w){

```

```

26     return (s & ~(3 << (2 * p - 2))) | (w << (2
27     * p - 2));
28 }
29 int findr(int s, int p){
30     int c = 0;
31     for(int q = p; q ≤ m + 1; ++ q){
32         if(((s >> (2 * q - 2)) & 3) == 1) ++ c;
33         if(((s >> (2 * q - 2)) & 3) == 2) -- c;
34         if(c == 0)
35             return q;
36     }
37     return -1;
38 }
39 int findl(int s, int p){
40     int c = 0;
41     for(int q = p; q ≥ 1; -- q){
42         if(((s >> (2 * q - 2)) & 3) == 2) ++ c;
43         if(((s >> (2 * q - 2)) & 3) == 1) -- c;
44         if(c == 0)
45             return q;
46     }
47     return -1;
48 }
49 void state(int s){
50     return ;
51     up(1, m + 1, i){
52         switch(getp(s, i)){
53             case 0 : putchar('#'); break;
54             case 1 : putchar('('); break;
55             case 2 : putchar(')'); break;
56             case 3 : putchar('E');
57         }
58     }
59     puts("");
60 }
61 int main(){
62     n = qread(), m = qread();
63     up(1, n, i)
64     scanf("%s", S[i] + 1);
65     int o = 0;
66     #define X M[ o]
67     #define Y M[!o]
68     vector <pair<int, bool> > T;
69     X.push_back({0, 0}, {1});
70     up(1, n, i){
71         Y.clear();
72         for(auto &u : X){
73             auto [s0, c] = u;
74             auto [s, f] = s0;
75             if(getp(s, m + 1) == 0)
76                 Y.push_back({s << 2, f}, c);
77         }

```

```

77 o ^= 1;
78 up(1, m, j){
79     int x = j, y = j + 1;
80     for(auto &u : X){
81         auto [s0, c] = u;
82         auto [s, f] = s0;
83         int a = getp(s, x);
84         int b = getp(s, y);
85         int t = setw(setw(s, x, 0), y, 0);
86         #define update(t, c) HashT :: find(t,
            f) += c, T.push_back({t, f})
87         if(S[i][j] == '.'){ // 经过该格
88             if(a == 1 && b == 1){
89                 t = setw(t, findr(s, y), 1),
90                 update(t, c);
91             } else
92             if(a == 2 && b == 2){
93                 t = setw(t, findl(s, x), 2),
94                 update(t, c);
95             } else
96             if(a == 1 && b == 2){
97                 if(f == false) // 还没有闭合回路
98                     f = true, update(t, c);
99             } else
100             if(a == 2 && b == 1){
101                 update(t, c);
102             } else
103             if(a == 0 && b == 0){
104                 t = setw(t, x, 1);
105                 t = setw(t, y, 2);
106                 update(t, c);
107             } else { // a == 0 || b == 0
108                 int t1 = setw(t, x, a | b);
109                 int t2 = setw(t, y, a | b);
110                 update(t1, c);
111                 update(t2, c);
112             }
113         }
114         if(S[i][j] == '*'){ // 不经过该格
115             if(a == 0 && b == 0)
116                 update(t, c);
117         }
118     }
119     Y.clear();
120     for(auto &u : T){
121         auto [s, f] = u;
122         if(HashT :: find(s, f) != 0){
123             Y.push_back({{s, f}, HashT :: find(s,
                f)});
124             HashT :: find(s, f) = 0;
125         }
126     }

```

```

127     T.clear(), o ^= 1;
128 }
129 }
130 i64 ans = 0;
131 for(auto &u : X){
132     auto [s0, c] = u;
133     auto [s, f] = s0;
134     bool g = true;
135     up(1, m + 1, i)
136     g &= getp(s, i) == 0;
137     f &= g;
138     if(f)
139         ans = c;
140 }
141 printf("%lld\n", ans);
142 return 0;
143 }

```

## 1.5 斜率优化

### 1.5.1 形式

考虑一个经典的 dp 转移方程如下：

$$f_i = \max_{j < i} \{f(j) + w(j, i)\}$$

我们将式子拆成三个部分：只跟  $i$  有关或者与  $i, j$  均不相关的部分  $a(i)$ ，只跟  $j$  有关的部分  $b(j)$ ，跟  $i, j$  均有关的部分  $c(i, j)$ ：

$$f_i = a(i) + \max_{j < i} \{b(j) + c(i, j)\}$$

斜率优化可被用来解决这样一个情形： $c(i, j) = ic_j$ 。此时  $b(j) + c(i, j)$  可视为关于  $j$  的一次函数。如果  $c_j$  随着  $j$  的增大而单调，那么可用单调栈维护；否则可以考虑 CDQ 分治或者在凸包上二分。在凸包上可以使用二分查询最高/最低点。

### 1.5.2 例题

玩具装箱。原始转移方程为：

$$f_i = \max_{j < i} \{f_j + (s_i - s_j - L')^2\}$$

其中  $s_i = i + \sum_{j \leq i} c_i, L' = L + 1$ 。将其分类得到：

$$f_i = \max_{j < i} \{f_j + s_i^2 + s_j^2 + L'^2 - 2s_i s_j + 2s_j L' - 2s_i L'\}$$

$$= (s_i^2 - 2s_i L' + L'^2) + \max_{j < i} \{(f_j + s_j^2 + 2s_j L') - 2s_i s_j\}$$

在原始的玩具装箱中， $s_j$  单调增加，也就是斜率单调增加。因此可以直接使用单调栈维护凸包。同时  $s_i$  也单调增加，因此可以用指针维护。

```

1 #include "../header.cpp"
2 int n, L, p, e, C[MAXN], Q[MAXN];
3 f80 S[MAXN], F[MAXN];
4 f80 gtx(int x){ return S[x]; }
5 f80 gty(int x){ return F[x] + S[x] * S[x]; }
6 f80 gtw(int x){ return -2.0 * (L - S[x]); }
7 f80 gtk(int x, int y){ return (gty(y) - gty(x))
    / (gtx(y) - gtx(x)); }
8 int main(){
9     cin >> n >> L;
10    for(int i = 1; i <= n; ++ i){
11        cin >> C[i];
12        S[i] = S[i - 1] + C[i];
13    }
14    for(int i = 1; i <= n; ++ i){
15        S[i] += i;
16    }
17    e = p = 1, L ++, Q[p] = 0;
18    for(int i = 1; i <= n; ++ i){
19        while(e < p && gtk(Q[e], Q[e + 1]) < gtw(i
            ))
20            ++ e;
21        int j = Q[e];
22        F[i] = F[j] + pow(S[i] - S[j] - L, 2);
23        while(1 < p && gtk(Q[p - 1], Q[p]) > gtk(Q
            [p], i))
24            e -= (e == p), -- p;
25        Q[++ p] = i;
26    }
27    printf("%.0Lf\n", F[n]);
28    return 0;
29 }

```

## 2 数据结构

### 2.1 平衡树

#### 2.1.1 无旋 Treap



```

1 #include "../header.cpp"
2 mt19937_64 MT(114514);
3 namespace Treap{
4     const int SZ = 1e6 + 1e5 + 3;
5     int F[SZ], C[SZ], S[SZ], W[SZ], X[SZ][2], sz;
6     u64 H[SZ];
7     int newnode(int w){
8         W[++sz] = w, C[sz] = S[sz] = 1; H[sz] = MT();
9         return sz;
10    }
11    void pushup(int x){
12        S[x] = C[x] + S[X[x][0]] + S[X[x][1]];
13    }
14    pair<int, int> split(int u, int x){
15        if(u == 0)
16            return make_pair(0, 0);
17        if(W[u] > x){
18            auto [a, b] = split(X[u][0], x);
19            X[u][0] = b, pushup(u);
20            return make_pair(a, u);
21        } else {
22            auto [a, b] = split(X[u][1], x);
23            X[u][1] = a, pushup(u);
24            return make_pair(u, b);
25        }
26    }
27    int merge(int a, int b){
28        if(a == 0 || b == 0)
29            return a | b;
30        if(H[a] < H[b]){
31            X[a][1] = merge(X[a][1], b), pushup(a);
32            return a;
33        } else {
34            X[b][0] = merge(a, X[b][0]), pushup(b);
35            return b;
36        }
37    }
38    void insert(int &root, int w){
39        auto [p, q] = split(root, w);
40        auto [a, b] = split(p, w - 1);
41        if(b != 0){
42            ++S[b], ++C[b];
43        } else b = newnode(w);
44        p = merge(a, b);
45        root = merge(p, q);
46    }
47    void erase(int &root, int w){
48        auto [p, q] = split(root, w);
49        auto [a, b] = split(p, w - 1);

```

```

50        -- C[b], -- S[b];
51        p = C[b] == 0 ? a : merge(a, b);
52        root = merge(p, q);
53    }
54    int find_rank(int &root, int w){
55        int x = root, o = x, a = 0;
56        for(;;x){
57            if(w < W[x])
58                o = x, x = X[x][0];
59            else {
60                a += S[X[x][0]];
61                if(w == W[x]){
62                    o = x; break;
63                }
64                a += C[x];
65                o = x, x = X[x][1];
66            }
67        }
68        return a + 1;
69    }
70    int find_kth(int &root, int w){
71        int x = root, o = x, a = 0;
72        for(;;x){
73            if(w <= S[X[x][0]])
74                o = x, x = X[x][0];
75            else {
76                w -= S[X[x][0]];
77                if(w <= C[x]){
78                    o = x; break;
79                }
80                w -= C[x];
81                o = x, x = X[x][1];
82            }
83        }
84        return W[x];
85    }
86    int find_pre(int &root, int w){
87        return find_kth(root, find_rank(root, w) - 1);
88    }
89    int find_suc(int &root, int w){
90        return find_kth(root, find_rank(root, w) + 1);
91    }
92 }

```

### 2.1.2 Splay

```

1 #include "../header.cpp"
2 namespace Splay{
3     const int SZ = 1e6 + 1e5 + 3;
4     int F[SZ], C[SZ], S[SZ], X[SZ][2], size;

```

```

5     bool T[SZ];
6     bool is_root(int x){ return F[x] == 0; }
7     bool is_rson(int x){ return X[F[x]][1] == x; }
8     void push_down(int x){
9         if(!T[x]) return;
10        int lc = X[x][0], rc = X[x][1];
11        if(lc) T[lc] ^= 1, swap(X[lc][0], X[lc][1]);
12        if(rc) T[rc] ^= 1, swap(X[rc][0], X[rc][1]);
13        T[x] = 0;
14    }
15    void pushup(int x){
16        S[x] = C[x] + S[X[x][0]] + S[X[x][1]];
17    }
18    void rotate(int x){
19        int y = F[x], z = F[y];
20        bool f = is_rson(x);
21        bool g = is_rson(y);
22        int &t = X[x][!f];
23        if(z){ X[z][g] = t; }
24        if(t){ F[t] = y; }
25        X[y][f] = t, t = y;
26        F[y] = x, pushup(y);
27        F[x] = z, pushup(x);
28    }
29    void splay(int &r, int x, int g = 0){
30        for(int f; f = F[x], f != g; rotate(x))
31            if(F[f] != g) rotate(is_rson(x) == is_rson(f) ? f : x);
32            if(is_root(x)) r = x;
33    }
34    int get_kth(int &r, int w){
35        int x = r, o = x;
36        for(;;x){
37            push_down(x);
38            if(w <= S[X[x][0]]) o = x, x = X[x][0];
39            else {
40                w -= S[X[x][0]];
41                if(C[x] && w <= C[x]){ o = x; break; }
42                w -= C[x], o = x, x = X[x][1];
43            }
44        }
45        splay(r, o); return o;
46    }
47    int build(int l, int r){
48        if(l == r){
49            C[l] = S[l] = 1; return l;
50        }
51        int c = l + r >> 1, a = 0, b = 0;
52        if(l <= c - 1) a = build(l, c - 1), F[a] =

```

```

    c, X[c][0] = a;
    if(c + 1 ≤ r) b = build(c + 1, r), F[b] =
    c, X[c][1] = b;
    C[c] = 1, pushup(c); return c;
}
void output(int n, int &r){
    push_down(r);
    if(X[r][0]) output(n, X[r][0]);
    if(r ≠ 1 && r ≠ n + 2) printf("%d ", r -
    1);
    if(X[r][1]) output(n, X[r][1]);
}
}

```

### 2.1.3 Treap

```

1 #include "../header.cpp"
2 mt19937_64 MT(114514);
3 namespace Treap{
4     const int SIZ = 1e6 + 1e5 + 3;
5     int F[SIZ], C[SIZ], S[SIZ], W[SIZ], X[SIZ
6         ][2], sz;
7     u64 H[SIZ];
8     bool is_root(int x){ return F[x] == 0; }
9     bool is_rson(int x){ return X[F[x]][1] == x
10         ; }
11     int newnode(int w){
12         W[++sz] = w, C[sz] = S[sz] = 1; H[sz] =
13         MT();
14         return sz;
15     }
16     void pushup(int x){
17         S[x] = C[x] + S[X[x][0]] + S[X[x][1]];
18     }
19     void rotate(int &root, int x){
20         int y = F[x], z = F[y];
21         bool f = is_rson(x);
22         bool g = is_rson(y);
23         int &t = X[x][!f];
24         if(z){ X[z][g] = x; } else root = x;
25         if(t){ F[t] = y; }
26         X[y][f] = t, t = y;
27         F[y] = x, pushup(y);
28         F[x] = z, pushup(x);
29     }
30     void insert(int &root, int w){
31         if(root == 0) {root = newnode(w); return;}
32         int x = root, o = x;
33         for(;x;o = x, x = X[x][w > W[x]]){
34             ++S[x]; if(w == W[x]){ ++C[x], o = x;
35             break;}
36         }
37     }
38 }

```

```

33 if(W[o] ≠ w){
34     if(w < W[o]) X[o][0] = newnode(w), F[sz]
35     = o, o = sz;
36     else X[o][1] = newnode(w), F[sz] = o
37     , o = sz;
38 }
39 while(!is_root(o) && H[o] < H[F[o]])
40     rotate(root, o);
41 void erase(int &root, int w){
42     int x = root, o = x;
43     for(;x;o = x, x = X[x][w > W[x]]){
44         --S[x]; if(w == W[x]){ --C[x], o = x;
45         break;}
46     }
47     if(C[o] == 0){
48         while(X[o][0] || X[o][1]){
49             u64 wl = X[o][0] ? H[X[o][0]] :
50             ULLONG_MAX;
51             u64 wr = X[o][1] ? H[X[o][1]] :
52             ULLONG_MAX;
53             if(wl < wr){
54                 int p = X[o][0]; rotate(root, p);
55             } else {
56                 int p = X[o][1]; rotate(root, p);
57             }
58         }
59         if(is_root(o)){
60             root = o;
61         } else {
62             X[F[o]][is_rson(o)] = o;
63         }
64     }
65     int find_rank(int &root, int w){
66         int x = root, o = x, a = 0;
67         for(;x;o = x, x = X[x][w > W[x]]){
68             if(w < W[x])
69                 o = x, x = X[x][0];
70             else {
71                 a += S[X[x][0]];
72                 if(w == W[x]){
73                     o = x; break;
74                 }
75                 a += C[x];
76                 o = x, x = X[x][1];
77             }
78         }
79         return a + 1;
80     }
81     int find_kth(int &root, int w){
82         int x = root, o = x, a = 0;
83     }
84 }

```

```

80 for(;x;){
81     if(w ≤ S[X[x][0]])
82         o = x, x = X[x][0];
83     else {
84         w -= S[X[x][0]];
85         if(w ≤ C[x]){
86             o = x; break;
87         }
88         w -= C[x];
89         o = x, x = X[x][1];
90     }
91 }
92 return W[x];
93 }
94 int find_pre(int &root, int w){
95     return find_kth(root, find_rank(root, w) -
96     1);
97 }
98 int find_suc(int &root, int w){
99     return find_kth(root, find_rank(root, w +
100     1));
101 }

```

## 2.2 珂朵莉树

```

1 #include "../header.cpp"
2 namespace ODT {
3     // <pos_type, value_type>
4     map<int, long long> M;
5     // 分裂为 [1, p) 和 [p, +inf), 返回后者迭代
6     器
7     auto split(int p) {
8         auto it = prev(M.upper_bound(p));
9         return M.insert(
10             it,
11             make_pair(p, it → second)
12         );
13     }
14     // 区间赋值
15     void assign(int l, int r, int v) {
16         auto it = split(l);
17         split(r + 1);
18         while (it → first ≠ r + 1) {
19             it = M.erase(it);
20         }
21         M[l] = v;
22     }
23     // // 执行操作
24     // void perform(int l, int r) {
25     //     auto it = split(l);
26     //     split(r + 1);
27 }

```

```

26 // while (it → first ≠ r + 1) {
27 //     // Do something...
28 //     it = next(it);
29 // }
30 // }
31 };

```

### 2.3 可并堆

```

1 #include "../header.cpp"
2 namespace LeftHeap{
3     const int SIZ = 1e5 + 3;
4     int W[SIZ], D[SIZ], L[SIZ], R[SIZ], F[SIZ],
5         s;
6     bool E[SIZ];
7     int merge(int u, int v){
8         if(u == 0 || v == 0)
9             return u | v;
10        if(W[u] > W[v] || (W[u] == W[v] && u > v))
11            swap(u, v);
12        int &lc = L[u];
13        int &rc = R[u];
14        rc = merge(rc, v);
15        if(D[lc] < D[rc])
16            swap(lc, rc);
17        D[u] = min(D[lc], D[rc]) + 1;
18        if(lc ≠ 0) F[lc] = u;
19        if(rc ≠ 0) F[rc] = u;
20        return u;
21    }
22    void pop(int &root){
23        int root0 = merge(L[root], R[root]);
24        F[root0] = root0;
25        F[root] = root0;
26        E[root] = true;
27        root = root0;
28    }
29    int top(int &root){
30        return W[root];
31    }
32    int getfa(int u){
33        return u = F[u] ? u : F[u] = getfa(F[u]);
34    }
35    int newnode(int w){
36        ++ s;
37        W[s] = w;
38        F[s] = s;
39        D[s] = 1;
40        return s;
41    }

```

### 2.4 线性基

```

1 #include "../header.cpp"
2 namespace LB{
3     const int SIZ = 60 + 3;
4     i64 W[SIZ], h = 60;
5     void insert(i64 w){
6         for(int i = h; i ≥ 0; -- i){
7             if(w & (1ll << i)){
8                 if(!W[i]){
9                     W[i] = w;
10                    break;
11                } else {
12                    w ^= W[i];
13                }
14            }
15        }
16    }
17    i64 query(i64 x){
18        for(int i = h; i ≥ 0; -- i){
19            if(W[i]){
20                x = max(x, x ^ W[i]);
21            }
22        }
23        return x;
24    }
25 }
26 namespace realLB{
27     const int SIZ = 500 + 3;
28     long double W[SIZ][SIZ];
29     int n = 0;
30     void init(int n0){
31         n = n0;
32     }
33     bool zero(long double w){
34         return fabs(w) < 1e-9;
35     }
36     bool insert(long double X[]){
37         for(int i = 1; i ≤ n; ++ i){
38             if(!zero(X[i])){
39                 if(zero(W[i][i])){
40                     for(int j = 1; j ≤ n; ++ j)
41                         W[i][j] = X[j];
42                     return true;
43                 } else {
44                     long double t = X[i] / W[i][i];
45                     for(int j = 1; j ≤ n; ++ j)
46                         X[j] -= t * W[i][j];
47                 }
48             }
49        }
50        return false;
51    }

```

```

52 }
53 // === TEST ===
54 int qread();
55 const int MAXN = 500 + 3;
56 long double X[MAXN][MAXN], C[MAXN];
57 int I[MAXN];
58 bool cmp(int a, int b){
59     return C[a] < C[b];
60 }
61 int main(){
62     int n, m;
63     cin >> n >> m;
64     realLB :: init(m);
65     for(int i = 1; i ≤ n; ++ i){
66         for(int j = 1; j ≤ m; ++ j){
67             cin >> X[i][j];
68         }
69     }
70     for(int i = 1; i ≤ n; ++ i){
71         cin >> C[i];
72         I[i] = i;
73     }
74     sort(I + 1, I + 1 + n, cmp);
75     int ans = 0, cnt = 0;
76     for(int i = 1; i ≤ n; ++ i){
77         int x = I[i];
78         if(realLB :: insert(X[x]))
79             ans += C[x],
80             cnt += 1;
81     }
82     cout << cnt << " " << ans << endl;
83     return 0;
84 }

```

### 2.5 Link Cut 树

```

1 #include "../header.cpp"
2 namespace LinkCutTree{
3     const int SIZ = 1e5 + 3;
4     int F[SIZ], C[SIZ], S[SIZ], W[SIZ], A[SIZ],
5         X[SIZ][2], size;
6     bool T[SIZ];
7     bool is_root(int x){ return X[F[x]][0] ≠ x
8         && X[F[x]][1] ≠ x; }
9     bool is_rson(int x){ return X[F[x]][1] = x; }
10    int new_node(int w){
11        ++ size;
12        W[size] = w, C[size] = S[size] = 1;
13        A[size] = w, F[size] = 0;
14        X[size][0] = X[size][1] = 0;
15        return size;

```



```

14 }
15 void push_up(int x){
16     S[x] = C[x] + S[X[x][0]] + S[X[x][1]];
17     A[x] = W[x] ^ A[X[x][0]] ^ A[X[x][1]];
18 }
19 void push_down(int x){
20     if(!T[x]) return;
21     int lc = X[x][0], rc = X[x][1];
22     if(lc) T[lc] ^= 1, swap(X[lc][0], X[lc][1]);
23     if(rc) T[rc] ^= 1, swap(X[rc][0], X[rc][1]);
24     T[x] = false;
25 }
26 void update(int x){
27     if(!is_root(x)) update(F[x]); push_down(x);
28 }
29 void rotate(int x){
30     int y = F[x], z = F[y];
31     bool f = is_rson(x);
32     bool g = is_rson(y);
33     if(is_root(y)){
34         F[x] = z, F[y] = x;
35         X[y][f] = X[x][!f], F[X[x][!f]] = y;
36         X[x][!f] = y;
37     } else {
38         F[x] = z, F[y] = x;
39         X[z][g] = x;
40         X[y][f] = X[x][!f], F[X[x][!f]] = y;
41         X[x][!f] = y;
42     }
43     push_up(y), push_up(x);
44 }
45 void splay(int x){
46     update(x);
47     for(int f = F[x]; f = F[x], !is_root(x); rotate(x))
48         if(!is_root(f)) rotate(is_rson(x) == is_rson(f) ? f : x);
49 }
50 int access(int x){
51     int p;
52     for(p = 0; x; p = x, x = F[x]){
53         splay(x), X[x][1] = p, push_up(x);
54     }
55     return p;
56 }
57 void make_root(int x){
58     x = access(x);
59     T[x] ^= 1, swap(X[x][0], X[x][1]);
60 }

```

```

61 int find_root(int x){
62     access(x), splay(x), push_down(x);
63     while(X[x][0]) x = X[x][0], push_down(x);
64     splay(x);
65     return x;
66 }
67 void link(int x, int y){
68     make_root(x), splay(x), F[x] = y;
69 }
70 void cut(int x, int p){
71     make_root(x), access(p), splay(p), X[p][0] = F[x] = 0;
72 }
73 void modify(int x, int w){
74     splay(x), W[x] = w, push_up(x);
75 }
76 }
77 const int MAXN = 1e5 + 3;
78 map<pair<int, int>, bool> M;
79 int n, m;
80 int main(){
81     cin >> n >> m;
82     for(int i = 1; i ≤ n; ++i){
83         int a; cin >> a;
84         LinkCutTree :: new_node(a);
85     }
86     for(int i = 1; i ≤ m; ++i){
87         int o; cin >> o;
88         if(o == 0){
89             int u, v; cin >> u >> v;
90             LinkCutTree :: make_root(u);
91             int p = LinkCutTree :: access(v);
92             printf("%d\n", LinkCutTree :: A[p]);
93         } else if(o == 1){
94             int u, v; cin >> u >> v;
95             int a = LinkCutTree :: find_root(u);
96             int b = LinkCutTree :: find_root(v);
97             if(a ≠ b){
98                 LinkCutTree :: link(u, v);
99                 M[make_pair(min(u, v), max(u, v))] = true;
100             }
101         } else if(o == 2){
102             int u, v; cin >> u >> v;
103             if(M.count(make_pair(min(u, v), max(u, v)))){
104                 M.erase(make_pair(min(u, v), max(u, v)));
105                 LinkCutTree :: cut(u, v);
106             }
107         } else {
108             int u, w; cin >> u >> w;

```

```

109     LinkCutTree :: modify(u, w);
110 }
111 }
112 return 0;
113 }

```

## 2.6 线段树

### 2.6.1 李超树

```

1 #include "../..header.cpp"
2 struct Line{ int id; double k, b; Line() = default; };
3 namespace LCSeg{
4     const int SIZ = 2e5 + 3;
5     struct Line T[SIZ];
6     #define lc(t) (t << 1)
7     #define rc(t) (t << 1 | 1)
8     bool cmp(int p, Line x, Line y){
9         double w1 = x.k * p + x.b;
10        double w2 = y.k * p + y.b;
11        double d = w1 - w2;
12        if(fabs(d) < 1e-8) return x.id > y.id;
13        return d < 0;
14    }
15    void merge(int t, int a, int b, Line x, Line y){
16        int c = a + b >> 1;
17        if(cmp(c, x, y)) swap(x, y);
18        if(cmp(a, y, x)){
19            T[t] = x; if(a ≠ b) merge(rc(t), c + 1, b, T[rc(t)], y);
20        } else {
21            T[t] = x; if(a ≠ b) merge(lc(t), a, c, T[lc(t)], y);
22        }
23    }
24    // 插入线段 (l, f(l)) -- (r, f(r))
25    void modify(int t, int a, int b, int l, int r, Line x){
26        if(l ≤ a && b ≤ r) merge(t, a, b, T[t], x);
27        else {
28            int c = a + b >> 1;
29            if(l ≤ c) modify(lc(t), a, c, l, r, x);
30            if(r > c) modify(rc(t), c + 1, b, l, r, x);
31        }
32    }
33    // 查询 x = p 位置最高的线段 (有多条取编号最小)

```

```

34 void query(int t, int a, int b, int p, Line
    &x){
35     if(cmp(p, x, T[t])) x = T[t];
36     if(a != b){
37         int c = a + b >> 1;
38         if(p ≤ c) query(lc(t), a, c, p, x);
39         if(p > c) query(rc(t), c + 1, b, p, x);
40     }
41 }
42 }

```

### 2.6.2 线段树 3

```

1 #include "../header.cpp"
2 int A[MAXN];
3 struct Node{
4     i64 sum; int len, max1, max2, max_cnt,
        his_mx;
5     Node():
6         sum(0), max1(-INF), max2(-INF), max_cnt(0)
        , his_mx(-INF), len(0) {}
7     Node(int w):
8         sum(w), max1(w), max2(-INF), max_cnt(1)
        , his_mx(w), len(1) {}
9     bool update(int w1, int w2, int h1, int h2){
10         his_mx = max({his_mx, max1 + h1});
11         max1 += w1, max2 += w2;
12         sum += 1ll * w1 * max_cnt + 1ll * w2 * (
            len - max_cnt);
13         return max1 > max2;
14     }
15 };
16 struct Tag{
17     int max_add, max_his_add, umx_add,
        umx_his_add; bool have;
18     void update(int w1, int w2, int h1, int h2){
19         max_his_add = max(max_his_add, max_add +
            h1);
20         umx_his_add = max(umx_his_add, umx_add +
            h2);
21         max_add += w1, umx_add += w2, have = true;
22     }
23     void clear(){
24         max_add = max_his_add = umx_add =
            umx_his_add = have = 0;
25     }
26 };
27 struct Node operator +(Node a, Node b){
28     Node t;
29     t.max1 = max(a.max1, b.max1);
30     if(t.max1 != a.max1){
31         if(a.max1 > t.max2) t.max2 = a.max1;

```

```

32     } else{
33         if(a.max2 > t.max2) t.max2 = a.max2;
34         t.max_cnt += a.max_cnt;
35     }
36     if(t.max1 != b.max1){
37         if(b.max1 > t.max2) t.max2 = b.max1;
38     } else{
39         if(b.max2 > t.max2) t.max2 = b.max2;
40         t.max_cnt += b.max_cnt;
41     }
42     t.sum = a.sum + b.sum, t.len = a.len + b.len
        ;
43     t.his_mx = max(a.his_mx, b.his_mx);
44     return t;
45 }
46 namespace Seg{
47     const int SIZ = 2e6 + 3;
48     struct Node W[SIZ]; struct Tag T[SIZ];
49     #define lc(t) (t << 1)
50     #define rc(t) (t << 1 | 1)
51     void push_up(int t, int a, int b){
52         W[t] = W[lc(t)] + W[rc(t)];
53     }
54     void push_down(int t, int a, int b){
55         if(a == b) T[t].clear();
56         if(T[t].have){
57             int c = a + b >> 1, x = lc(t), y = rc(t)
                ;
58             int w = max(W[x].max1, W[y].max1);
59             int w1 = T[t].max_add, w2 = T[t].umx_add
                , w3 = T[t].max_his_add, w4 = T[t].
                umx_his_add;
60             if(w == W[x].max1)
61                 W[x].update(w1, w2, w3, w4),
62                 T[x].update(w1, w2, w3, w4);
63             else
64                 W[x].update(w2, w2, w4, w4),
65                 T[x].update(w2, w2, w4, w4);
66             if(w == W[y].max1)
67                 W[y].update(w1, w2, w3, w4),
68                 T[y].update(w1, w2, w3, w4);
69             else
70                 W[y].update(w2, w2, w4, w4),
71                 T[y].update(w2, w2, w4, w4);
72             T[t].clear();
73         }
74     }
75     void build(int t, int a, int b){
76         if(a == b){W[t] = Node(A[a]), T[t].clear()
            ;} else {
77             int c = a + b >> 1; T[t].clear();
78             build(lc(t), a, c);

```

```

79         build(rc(t), c + 1, b);
80         push_up(t, a, b);
81     }
82 }
83 void modiadd(int t, int a, int b, int l, int
    r, int w){
84     if(l ≤ a && b ≤ r){
85         T[t].update(w, w, w, w);
86         W[t].update(w, w, w, w);
87     } else {
88         int c = a + b >> 1; push_down(t, a, b);
89         if(l ≤ c) modiadd(lc(t), a, c, l, r,
            w);
90         if(r > c) modiadd(rc(t), c + 1, b, l, r
            , w);
91         push_up(t, a, b);
92     }
93 }
94 void modimin(int t, int a, int b, int l, int
    r, int w){
95     if(l ≤ a && b ≤ r){
96         if(w ≥ W[t].max1) return; else
97         if(w > W[t].max2){
98             int k = w - W[t].max1;
99             T[t].update(k, 0, k, 0);
100             W[t].update(k, 0, k, 0);
101         } else {
102             int c = a + b >> 1;
103             push_down(t, a, b);
104             modimin(lc(t), a, c, l, r, w);
105             modimin(rc(t), c + 1, b, l, r, w);
106             push_up(t, a, b);
107         }
108     } else {
109         int c = a + b >> 1; push_down(t, a, b);
110         if(l ≤ c) modimin(lc(t), a, c, l, r,
            w);
111         if(r > c) modimin(rc(t), c + 1, b, l, r
            , w);
112         push_up(t, a, b);
113     }
114 }
115 Node query(int t, int a, int b, int l, int r
    ){
116     if(l ≤ a && b ≤ r) return W[t];
117     int c = a + b >> 1; Node ret; push_down(t,
        a, b);
118     if(l ≤ c) ret = ret + query(lc(t), a, c
        , l, r);
119     if(r > c) ret = ret + query(rc(t), c + 1,
        b, l, r);
120     return ret;

```

```

121 }
122 }
123 int qread();
124 int main(){
125     int n = qread(), m = qread();
126     for(int i = 1; i ≤ n; ++ i)
127         A[i] = qread();
128     Seg :: build(1, 1, n);
129     for(int i = 1; i ≤ m; ++ i){
130         int op = qread();
131         if(op == 1){
132             int l = qread(), r = qread(), w = qread
                ();
133             Seg :: modiadd(1, 1, n, l, r, w);
134         } else if(op == 2){
135             int l = qread(), r = qread(), w = qread
                ();
136             Seg :: modimin(1, 1, n, l, r, w);
137         } else if(op == 3){
138             int l = qread(), r = qread();
139             auto p = Seg :: query(1, 1, n, l, r);
140             printf("%lld\n", p.sum);
141         } else if(op == 4){
142             int l = qread(), r = qread();
143             auto p = Seg :: query(1, 1, n, l, r);
144             printf("%d\n", p.max1);
145         } else if(op == 5){
146             int l = qread(), r = qread();
147             auto p = Seg :: query(1, 1, n, l, r);
148             printf("%d\n", p.his_mx);
149         }
150     }
151     return 0;
152 }

```

### 2.6.3 扫描线

```

1 #include "../header.cpp"
2 const int MAXN = 1e5 + 3;
3 int X1[MAXN], Y1[MAXN];
4 int X2[MAXN], Y2[MAXN];
5 int n, h, H[MAXN * 2];
6 namespace Seg{
7     #define lc(t) (t << 1)
8     #define rc(t) (t << 1 | 1)
9     const int SZ = 8e5 + 3;
10    int T[SZ], S[SZ], L[SZ];
11    void pushup(int t, int a, int b){
12        S[t] = 0;
13        if(a ≠ b){
14            S[t] = S[lc(t)] + S[rc(t)];
15            L[t] = L[lc(t)] + L[rc(t)];

```

```

16    }
17    if(T[t]) S[t] = L[t];
18 }
19 void modify(int t, int a, int b, int l, int
    r, int w){
20     if(l ≤ a && b ≤ r){
21         T[t] += w, pushup(t, a, b);
22     } else {
23         int c = a + b >> 1;
24         if(l ≤ c) modify(lc(t), a, c, l, r, w);
25         if(r > c) modify(rc(t), c + 1, b, l, r,
            w);
26         pushup(t, a, b);
27     }
28 }
29 void build(int t, int a, int b){
30     if(a == b){
31         L[t] = H[a] - H[a - 1];
32     } else {
33         int c = a + b >> 1;
34         build(lc(t), a, c);
35         build(rc(t), c + 1, b);
36         pushup(t, a, b);
37     }
38 }
39 int query(int t){
40     return S[t];
41 }
42 }
43 tuple <int, int, int> P[MAXN], Q[MAXN];
44 int main(){
45     n = qread();
46     for(int i = 1; i ≤ n; ++ i){
47         X1[i] = qread(), Y1[i] = qread();
48         X2[i] = qread(), Y2[i] = qread();
49         if(X1[i] > X2[i]) swap(X1[i], X2[i]);
50         if(Y1[i] > Y2[i]) swap(Y1[i], Y2[i]);
51         H[++ h] = Y1[i];
52         H[++ h] = Y2[i];
53         P[i] = make_tuple(X1[i], Y1[i], Y2[i]);
54         Q[i] = make_tuple(X2[i], Y1[i], Y2[i]);
55     }
56     sort(H + 1, H + 1 + h);
57     sort(P + 1, P + 1 + n);
58     sort(Q + 1, Q + 1 + n);
59     int o = unique(H + 1, H + 1 + h) - H - 1;
60     Seg :: build(1, 1, o);
61     i64 ans = 0, last = -1;
62     int p = 1, q = 1;
63     while(p ≤ n || q ≤ n){
64         int x = INF;
65         if(p ≤ n) x = min(x, get<0>(P[p]));

```

```

66     if(q ≤ n) x = min(x, get<0>(Q[q]));
67     if(last ≠ -1){
68         ans += 1ll * Seg :: query(1) * (x - last
            );
69     }
70     last = x;
71     while(q ≤ n && get<0>(Q[q]) == x){
72         auto [x, l, r] = Q[q]; ++ q;
73         l = lower_bound(H + 1, H + 1 + o, l) - H
            + 1;
74         r = lower_bound(H + 1, H + 1 + o, r) - H
            ;
75         Seg :: modify(1, 1, o, l, r, 1);
76     }
77     while(p ≤ n && get<0>(P[p]) == x){
78         auto [x, l, r] = P[p]; ++ p;
79         l = lower_bound(H + 1, H + 1 + o, l) - H
            + 1;
80         r = lower_bound(H + 1, H + 1 + o, r) - H
            ;
81         Seg :: modify(1, 1, o, l, r, -1);
82     }
83 }
84 printf("%lld\n", ans);
85 return 0;
86 }

```

## 2.7 根号数据结构

### 2.7.1 块状链表

```

1 #include "../header.cpp"
2 namespace BLOCK{
3     const int SZ = 1e6 + 1e5 + 3;
4     const int BSZ = 2000;
5     list <vector<int>> block;
6     void build(int n, const int A[]){
7         for(int l = 0, r = 0; r ≠ n; ){
8             l = r;
9             r = min(l + BSZ / 2, n);
10            vector <int> V0(A + l, A + r);
11            block.emplace_back(V0);
12        }
13    }
14    int get_kth(int k){
15        for(auto it = block.begin(); it ≠ block.
            end(); ++ it){
16            if(it → size() < k)
17                k -= it → size();
18            else return it → at(k - 1);
19        }
20        return -1;

```

```

21 }
22 int get_rank(int w){
23     int ans = 0;
24     for(auto it = block.begin(); it != block.
25         end(); ++ it){
26         if(it → back() < w)
27             ans += it → size();
28     }
29     else {
30         ans += lower_bound(it → begin(), it
31             → end(), w) - it → begin();
32         break;
33     }
34 }
35 return ans + 1;
36 }
37 // 插入到第 k 个位置
38 void insert(int k, int w){
39     for(auto it = block.begin(); it != block.
40         end(); ++ it){
41         if(it → size() < k)
42             k -= it → size();
43     }
44     else {
45         it → insert(it → begin() + k - 1, w)
46         ;
47         if(it → size() > BSZ){
48             vector<int> V1(it → begin(), it →
49                 begin() + BSZ / 2);
50             vector<int> V2(it → begin() + BSZ
51                 / 2, it → end());
52             *it = V2;
53             block.insert(it, V1);
54         }
55     }
56     return;
57 }
58 // 删除第 k 个数
59 void erase(int k){
60     for(auto it = block.begin(); it != block.
61         end(); ++ it){
62         if(it → size() < k)
63             k -= it → size();
64     }
65     else {
66         it → erase(it → begin() + k - 1);
67         if(it → empty())
68             block.erase(it);
69     }
70     return;
71 }
72 }
73 int A[MAXN];

```

```

66 // == TEST ==
67 int main(){
68     ios :: sync_with_stdio(false);
69     cin.tie(nullptr);
70     int n, m;
71     cin >> n >> m;
72     for(int i = 1; i ≤ n; ++ i)
73         cin >> A[i];
74     sort(A + 1, A + 1 + n);
75     A[n + 1] = INT_MAX;
76     BLOCK :: build(n + 1, A + 1);
77     int last = 0;
78     int ans = 0;
79     // Do some op...
80     cout << ans << endl;
81     return 0;
82 }

```

### 2.7.2 莫队二次离线

```

1 #include " ../header.cpp"
2 int n, m, k, maxt = 16383, X[MAXM], C[MAXM], t
3 ;
4 int A[MAXN], bsize; i64 B[MAXN], R[MAXN];
5 struct Qry1{ int l, r, id; }O[MAXN];
6 struct Qry2{ int id, l, r; };
7 struct Qry3{ int id, l, r; };
8 bool cmp(Qry1 a, Qry1 b){
9     return a.l / bsize == b.l / bsize ? a.r < b.
10         r : a.l < b.l;
11 }
12 vector<Qry2> P[MAXN];
13 vector<Qry3> Q[MAXN];
14 int main(){
15     n = qread(), m = qread(), k = qread(), bsize
16         = sqrt(m + 1);
17     up(1, n, i) A[i] = qread();
18     up(1, m, i){
19         int l = qread(), r = qread(); O[i] = {l, r
20             , i};
21     }
22     sort(O + 1, O + 1 + m, cmp);
23     int l = 1, r = 0;
24     up(1, m, i){
25         int p = O[i].l, q = O[i].r;
26         if(r < q){
27             P[r].push_back({ i, r + 1, q});
28             Q[l - 1].push_back({-i, r + 1, q});
29         }
30         if(r > q){
31             P[q].push_back({-i, q + 1, r});
32             Q[l - 1].push_back({ i, q + 1, r});
33         }
34     }
35 }

```

```

29 }
30 r = q;
31 if(l > p){
32     P[p].push_back({-i, p, l - 1});
33     Q[r].push_back({ i, p, l - 1});
34 }
35 if(l < p){
36     P[l].push_back({ i, l, p - 1});
37     Q[r].push_back({-i, l, p - 1});
38 }
39 l = p;
40 }
41 up(0, maxt, i) if(__builtin_popcount(i) == k
42 ) X[+ t] = i;
43 up(0, n, i){
44     up(1, t, j) += C[A[i] ^ X[j]];
45     for(auto &o : P[i]){
46         if(o.id > 0) R[ o.id] += C[A[o.l]];
47         else R[-o.id] -= C[A[o.l]];
48         if(o.l < o.r)
49             P[i + 1].push_back({o.id, o.l + 1, o.r
50                 });
51     }
52     for(auto &o : Q[i]){
53         up(o.l, o.r, j){
54             if(o.id > 0) R[ o.id] += C[A[j]];
55             else R[-o.id] -= C[A[j]];
56         }
57     }
58     P[i].clear(), Q[i].clear();
59     P[i].shrink_to_fit();
60     Q[i].shrink_to_fit();
61 }
62 i64 ans = 0;
63 up(1, m, i){ ans += R[i], B[O[i].id] = ans;
64 }
65 up(1, m, i) printf("%lld\n", B[i]);
66 return 0;
67 }

```

## 3 树论

### 3.1 点分树

#### 3.1.1 例题

给定  $n$  个点组成的树，点有点权  $v_i$ 。  $m$  个操作，分为两种：

- $0 \times k$  查询距离  $x$  不超过  $k$  的所有点的点权之和；

- $0 \times y$  将点  $x$  的点权修改为  $y$ 。

```

1 #include "../header.cpp"
2 vector<int> E[MAXN];
3 namespace LCA{
4     const int SIZ = 1e5 + 3;
5     int D[SIZ], F[SIZ];
6     int P[SIZ], Q[SIZ], o;
7     void dfs(int u, int f){
8         P[u] = ++ o;
9         Q[o] = u;
10        F[u] = f;
11        D[u] = D[f] + 1;
12        for(auto &v : E[u]) if(v != f){
13            dfs(v, u);
14        }
15    }
16    const int MAXH = 18 + 3;
17    int h = 18;
18    int ST[SIZ][MAXH];
19    int cmp(int a, int b){
20        return D[a] < D[b] ? a : b;
21    }
22    int T[SIZ], n;
23    void init(int _n){
24        n = _n;
25        dfs(1, 0);
26        for(int i = 1; i <= n; ++ i)
27            ST[i][0] = Q[i];
28        for(int i = 2; i <= n; ++ i)
29            T[i] = T[i >> 1] + 1;
30        for(int i = 1; i <= h; ++ i){
31            for(int j = 1; j <= n; ++ j) if(j + (1 <<
32                i - 1) <= n){
33                ST[j][i] = cmp(ST[j][i - 1], ST[j + (1
34                    << i - 1)][i - 1]);
35            }
36        }
37        int lca(int a, int b){
38            if(a == b)
39                return a;
40            int l = P[a];
41            int r = P[b];
42            if(l > r)
43                swap(l, r);
44            ++ l;
45            int d = T[r - l + 1];
46            return F[cmp(ST[l][d], ST[r - (1 << d) +
47                1][d])];

```

```

48        return D[a] + D[b] - 2 * D[lca(a, b)];
49    }
50 }
51 namespace BIT{
52     void modify(int D[], int n, int p, int w){
53         ++ p;
54         while(p <= n)
55             D[p] += w, p += p & -p;
56     }
57     int query(int D[], int n, int p){
58         if(p < 0) return 0;
59         p = min(n, p + 1);
60         int r = 0;
61         while(p > 0)
62             r += D[p], p -= p & -p;
63         return r;
64     }
65 }
66 namespace PTree{
67     const int SIZ = 1e5 + 3;
68     bool V[SIZ];
69     int S[SIZ], L[SIZ];
70     vector<int> EE[MAXN];
71     int *D1[MAXN];
72     int *D2[MAXN];
73     void dfs1(int s, int &g, int u, int f){
74         S[u] = 1;
75         int maxsize = 0;
76         for(auto &v : E[u]) if(v != f && !V[v]){
77             dfs1(s, g, v, u);
78             if(S[v] > maxsize)
79                 maxsize = S[v];
80             S[u] += S[v];
81         }
82         maxsize = max(maxsize, s - S[u]);
83         if(maxsize <= s / 2)
84             g = u;
85     }
86     int n;
87     void build(int s, int &g, int u, int f){
88         dfs1(s, g, u, f);
89         V[g] = true, L[g] = s;
90         for(auto &u : E[g]) if(!V[u]){
91             int h = 0;
92             if(S[u] < S[g]) build(S[u], h, u, 0);
93             else build(s - S[g], h, u, 0);
94             EE[g].push_back(h);
95             EE[h].push_back(g);
96         }
97     }
98     int F[SIZ];
99     void dfs2(int u, int f){

```

```

100        F[u] = f;
101        for(auto &v : EE[u]) if(v != f){
102            dfs2(v, u);
103        }
104    }
105    void build(int _n){
106        n = _n;
107        int s = n, g = 0;
108        dfs1(s, g, 1, 0);
109        V[g] = true, L[g] = s;
110        for(auto &u : E[g]){
111            int h = 0;
112            if(S[u] < S[g]) build(S[u], h, u, 0);
113            else build(s - S[g], h, u, 0);
114            EE[g].push_back(h);
115            EE[h].push_back(g);
116        }
117        dfs2(g, 0);
118        for(int i = 1; i <= n; ++ i){
119            L[i] += 2;
120            D1[i] = new int[L[i] + 3];
121            D2[i] = new int[L[i] + 3];
122            for(int j = 0; j < L[i] + 3; ++ j)
123                D1[i][j] = D2[i][j] = 0;
124        }
125    }
126    void modify(int x, int w){
127        int u = x;
128        while(1){
129            BIT :: modify(D1[x], L[x], LCA :: dis(u,
130                x), w);
131            int y = F[x];
132            if(y != 0){
133                int e = LCA :: dis(x, y);
134                BIT :: modify(D2[x], L[x], LCA :: dis(
135                    u, y), w);
136                x = y;
137            } else break;
138        }
139    }
140    int query(int x, int d){
141        int ans = 0, u = x;
142        while(1){
143            ans += BIT :: query(D1[x], L[x], d - LCA
144                :: dis(u, x));
145            int y = F[x];
146            if(y != 0){
147                int e = LCA :: dis(x, y);
148                ans -= BIT :: query(D2[x], L[x], d -
149                    LCA :: dis(u, y));
150                x = y;
151            } else break;

```



```

148 }
149 return ans;
150 }
151 }
152 int W[MAXN];
153 int main(){
154     ios :: sync_with_stdio(false);
155     int n, m;
156     cin >> n >> m;
157     for(int i = 1; i ≤ n; ++ i){
158         cin >> W[i];
159     }
160     for(int i = 2; i ≤ n; ++ i){
161         int u, v;
162         cin >> u >> v;
163         E[u].push_back(v);
164         E[v].push_back(u);
165     }
166     LCA :: init(n);
167     PTree :: build(n);
168     for(int i = 1; i ≤ n; ++ i)
169         PTree :: modify(i, W[i]);
170     int lastans = 0;
171     for(int i = 1; i ≤ m; ++ i){
172         int op; cin >> op;
173         if(op == 0){
174             int x, d;
175             cin >> x >> d;
176             x ^= lastans;
177             d ^= lastans;
178             cout << (lastans = PTree :: query(x, d))
179                 << endl;
180         } else {
181             int x, w;
182             cin >> x >> w;
183             x ^= lastans;
184             w ^= lastans;
185             PTree :: modify(x, -W[x]);
186             PTree :: modify(x, W[x] = w);
187         }
188     }
189     return 0;
190 }

```

### 3.2 长链剖分

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 using i64 = long long;
4 const int INF = 1e9;
5 const i64 INFL = 1e18;
6 const int MAXN= 5e5 + 3;

```

```

7 const int MAXM= 19 + 3;
8 vector <int> P[MAXN];
9 vector <int> Q[MAXN];
10 vector <int> E[MAXN];
11 int h = 19;
12 int L[MAXN], F[MAXN], G[MAXN], D[MAXN], S[MAXM
    ][MAXN];
13 void dfs1(int u, int f){
14     L[u] = 1, S[0][u] = f;
15     F[u] = f, D[u] = D[f] + 1;
16     for(int i = 1; i ≤ h; ++ i)
17         S[i][u] = S[i - 1][S[i - 1][u]];
18     for(auto &v : E[u]) if(v ≠ f){
19         dfs1(v, u);
20         if(L[v] > L[G[u]])
21             G[u] = v;
22         L[u] = max(L[u], L[v] + 1);
23     }
24 }
25 int T[MAXN];
26 void dfs2(int u, int f){
27     if(u == G[f]){
28         T[u] = T[f];
29         P[T[u]].push_back(u);
30         Q[T[u]].push_back(F[Q[T[u]].back()]);
31     } else {
32         T[u] = u;
33         P[u].push_back(u);
34         Q[u].push_back(u);
35     }
36     if(G[u]) dfs2(G[u], u);
37     for(auto &v : E[u]) if(v ≠ f && v ≠ G[u
        ])
38         dfs2(v, u);
39 }
40 typedef unsigned int u32;
41 typedef unsigned long long u64;
42 int n, q; u32 s;
43 u32 get(u32 x) {
44     x ^= x << 13;
45     x ^= x >> 17;
46     x ^= x << 5;
47     return s = x;
48 }
49 int qread();
50 int H[MAXN];
51 int main(){
52     scanf("%d%d%u", &n, &q, &s);
53     int root = 0; H[0] = -1;
54     for(int i = 1; i ≤ n; ++ i){
55         int f = qread();
56         if(f == 0)

```

```

57         root = i;
58     } else {
59         E[f].push_back(i);
60         E[i].push_back(f);
61     }
62     H[i] = H[i >> 1] + 1;
63 }
64 dfs1(root, 0);
65 dfs2(root, 0);
66 int lastans = 0;
67 i64 realans = 0;
68 for(int i = 1; i ≤ q; ++ i){
69     int x = (get(s) ^ lastans) % n + 1;
70     int k = (get(s) ^ lastans) % D[x];
71     if(k == 0){
72         lastans = x;
73     } else {
74         int h = H[k];
75         k -= 1 << h;
76         x = S[h][x];
77         int t = T[x];
78         k -= D[x] - D[t];
79         if(k > 0){
80             x = Q[t][k];
81         } else {
82             x = P[t][-k];
83         }
84         lastans = x;
85     }
86     realans ^= 1ll * i * lastans;
87 }
88 printf("%lld\n", realans);
89 return 0;
90 }

```

### 3.3 重链剖分

```

1 #include "../header.cpp"
2 int n, m, root, MOD, A[MAXN];
3 int qread();
4 vector <int> E[MAXN];
5 int S[MAXN], G[MAXN], D[MAXN], F[MAXN];
6 void dfs1(int u, int f){
7     S[u] = 1, G[u] = 0, D[u] = D[f] + 1, F[u] =
    f;
8     for(auto &v : E[u]) if(v ≠ f){
9         dfs1(v, u);
10        S[u] += S[v];
11        if(S[v] > S[G[u]])
12            G[u] = v;
13    }
14 }

```

```

15 int B[MAXN];
16 int P[MAXN], Q[MAXN], T[MAXN], L[MAXN], R[MAXN]
    ], cnt;
17 void dfs2(int u, int f){
18     P[++ cnt] = u, B[cnt] = A[u], Q[u] = cnt;
19     L[u] = cnt;
20     if(u ≠ G[f]) T[u] = u;
21     else T[u] = T[f];
22     if(G[u]) dfs2(G[u], u);
23     for(auto &v : E[u]) if(v ≠ f && v ≠ G[u]){
24         dfs2(v, u);
25     }
26     R[u] = cnt;
27 }
28 namespace Seg{
29     const int SIZ = 4e5 + 3;
30     i64 S[SIZ], T[SIZ];
31     void pushup(int t, int a, int b);
32     void pushdown(int t, int a, int b);
33     void modify(int t, int a, int b, int l, int
        r, int w);
34     i64 query(int t, int a, int b, int l, int r)
        ;
35     void build(int t, int a, int b);
36 }
37 int main(){
38     n = qread(), m = qread(), root = qread(),
        MOD = qread();
39     for(int i = 1; i ≤ n; ++ i)
40         A[i] = qread();
41     for(int i = 2; i ≤ n; ++ i){
42         int u = qread(), v = qread();
43         E[u].push_back(v);
44         E[v].push_back(u);
45     }
46     dfs1(root, 0);
47     dfs2(root, 0);
48     Seg :: build(1, 1, n);
49     for(int i = 1; i ≤ m; ++ i){
50         int op = qread();
51         if(op == 1){
52             int u = qread(), v = qread(), k = qread
                ();
53             while(T[u] ≠ T[v]){
54                 if(D[T[u]] < D[T[v]])
55                     swap(u, v);
56                 Seg :: modify(1, 1, n, Q[T[u]], Q[u],
                    k);
57                 u = F[T[u]];
58             }
59             if(D[u] < D[v]) swap(u, v);
60             Seg :: modify(1, 1, n, Q[v], Q[u], k);

```

```

61     } else if(op == 2){
62         int u = qread(), v = qread();
63         i64 ans = 0;
64         while(T[u] ≠ T[v]){
65             if(D[T[u]] < D[T[v]])
66                 swap(u, v);
67             ans = (ans + Seg :: query(1, 1, n, Q[T
                [u]], Q[u])) % MOD;
68             u = F[T[u]];
69         }
70         if(D[u] < D[v]) swap(u, v);
71         ans = (ans + Seg :: query(1, 1, n, Q[v],
            Q[u])) % MOD;
72         printf("%lld\n", ans);
73     } else if(op == 3){
74         int x = qread(), w = qread();
75         Seg :: modify(1, 1, n, L[x], R[x], w);
76     } else {
77         int x = qread();
78         printf("%lld\n", Seg :: query(1, 1, n, L
            [x], R[x]));
79     }
80 }
81 return 0;
82 }

```

### 3.4 树哈希

#### 3.4.1 用法

给定大小为  $n$  的以 1 为根的树，计算  $h_i$  表示子树  $i$  的哈希值，计算有多少个本质不同的值。

```

1 #include "../header.cpp"
2 u64 xor_shift(u64 x);
3 u64 H[MAXN];
4 vector<int> E[MAXN];
5 void dfs(int u, int f){
6     H[u] = 1;
7     for(auto &v : E[u]) if(v ≠ f){
8         dfs(v, u);
9         H[u] += H[v];
10    }
11    H[u] = xor_shift(H[u]); // !important
12 }
13 int main(){
14     int n;
15     cin >> n;
16     for(int i = 2; i ≤ n; ++ i){
17         int u, v;
18         cin >> u >> v;
19         E[u].push_back(v);

```

```

20     E[v].push_back(u);
21 }
22 dfs(1, 0);
23 sort(H + 1, H + 1 + n);
24 cout << (unique(H + 1, H + 1 + n) - H - 1)
    << endl;
25 return 0;
26 }

```

### 3.5 Prufer 序列

```

1 #include "../header.cpp"
2 int D[MAXN], F[MAXN], P[MAXN];
3 vector<int> tree2prufer(int n){
4     vector<int> P(n);
5     for(int i = 1, j = 1; i ≤ n - 2; ++ i, ++ j){
6         while(D[j]) ++ j;
7         P[i] = F[j];
8         while(i ≤ n - 2 && !--D[P[i]] && P[i] < j
            )
9             P[i + 1] = F[P[i]], i ++;
10    }
11    return P;
12 }
13 vector<int> prufer2tree(int n){
14     vector<int> F(n);
15     for(int i = 1, j = 1; i ≤ n - 1; ++ i, ++ j){
16         while(D[j]) ++ j;
17         F[j] = P[i];
18         while(i ≤ n - 1 && !--D[P[i]] && P[i] < j
            )
19             F[P[i]] = P[i + 1], i ++;
20    }
21    return F;
22 }

```

### 3.6 虚树

```

1 #include "../header.cpp"
2 vector<pair<int, int> > E[MAXN];
3 namespace LCA{
4     const int SIZ = 5e5 + 3;
5     int D[SIZ], H[SIZ], F[SIZ], P[SIZ], Q[SIZ],
        o;
6     void dfs(int u, int f){
7         P[u] = ++ o, Q[o] = u, F[u] = f, D[u] = D[
            f] + 1;
8         for(auto &[v, w] : E[u]) if(v ≠ f){
9             H[v] = H[u] + w, dfs(v, u);
10        }
11    }

```

## 4 图论

## 4.1 仙人掌

## 4.1.1 例题

给定一个仙人掌，多组询问  $u, v$  之间最短路长度。

```

1 #include "../header.cpp"
2 const int MAXD= 18 + 3;
3 struct edge{int u, v, w;};
4 vector <edge> V1[MAXN];
5 vector <edge> V2[MAXN];
6 vector <int> H[MAXN];
7 int n, D[MAXN], W[MAXN], F[MAXD][MAXN];
8 int o, X[MAXN], L[MAXN];
9 bool E[MAXN];
10 void dfs1(int u, int f){
11     D[u] = D[f] + 1, F[0][u] = f;
12     for(auto &e : V1[u]) if(e.v != f){
13         if(D[e.v] && D[e.v] < D[u]){
14             int a = e.u;
15             int b = e.v;
16             int c = ++ o, t = c + n;
17             H[c].push_back(a);
18             L[c] = W[a] - W[b] + e.w;
19             while(a != b)
20                 E[a] = true, a = F[0][a], H[c].
21                     push_back(a);
22             for(auto &x : H[c]){
23                 int w = min(W[x] - W[b], L[c] - W[x] +
24                     W[b]);
25                 V2[x].push_back(edge{x, t, w});
26                 V2[t].push_back(edge{t, x, w});
27             } else if(!D[e.v]){
28                 W[e.v] = W[u] + e.w, dfs1(e.v, u);
29             }
30         }
31     }
32     for(auto &e : V1[u]) if(D[e.v] > D[u]){
33         if(!E[e.v]){
34             V2[e.u].push_back({e.u, e.v, e.w});
35             V2[e.v].push_back({e.v, e.u, e.w});
36         }
37     }
38 }
39 int d = 18;
40 void dfs2(int u, int f){
41     D[u] = D[f] + 1, F[0][u] = f;
42     up(1, d, i) F[i][u] = F[i - 1][F[i - 1][u]];
43     for(auto &e : V2[u]) if(e.v != f){
44         X[e.v] = X[e.u] + e.w;
45         dfs2(e.v, u);
46     }
47 }

```

```

44 }
45 }
46 int lca(int u, int v){
47     if(D[u] < D[v]) swap(u, v);
48     dn(d, 0, i) if(D[F[i][u]] ≥ D[v]) u = F[i][
49         u];
50     if(u == v) return u;
51     dn(d, 0, i) if(F[i][u] ≠ F[i][v]) u = F[i][
52         u], v = F[i][v];
53     return F[0][u];
54 }
55 int jump(int u, int v){
56     dn(d, 0, i) if(D[F[i][v]] > D[u]) v = F[i][
57         v];
58     return v;
59 }
60 int dis(int x, int y){
61     int t = lca(x, y);
62     if(t > n){
63         int u = jump(t, x);
64         int v = jump(t, y);
65         int w = abs(W[u] - W[v]);
66         int l = min(w, L[t - n] - w);
67         return X[x] - X[u] + X[y] - X[v] + l;
68     } else {
69         return X[x] + X[y] - 2 * X[t];
70     }
71 }
72 int m, q;
73 int qread();
74 int main(){
75     n = qread(), m = qread(), q = qread();
76     up(1, m, i){
77         int u = qread(), v = qread(), w = qread();
78         V1[u].push_back(edge{u, v, w});
79         V1[v].push_back(edge{v, u, w});
80     }
81     dfs1(1, 0);
82     dfs2(1, 0);
83     up(1, q, i){
84         int u = qread(), v = qread();
85         printf("%d\n", dis(u, v));
86     }
87     return 0;
88 }

```

## 4.2 三元环计数

### 4.2.1 三元环计数

无向图：考虑将所有点按度数从小往大排序，然后将每条边定向，由排在前面的指向排在后面的，得到一个有向图。然后考虑枚举一个点，再枚举一个点，暴力数，具体见代码。结论是，这样定向后，每个点的出度是  $O(\sqrt{m})$  的。复杂度  $O(m\sqrt{m})$ 。有向图：不难发现，上述方法枚举了三个点，计算有向图三元环也就只需要处理下方向的事，这个由于算法够暴力，随便改改就能做了。

```

34         ] : G[j]) {
35             if (w1 == w2) ans += w1 ? in[k] : out[
36                 k];
37         }
38     for (auto [j, w] : G[i]) in[j] = out[j] =
        0;
    }
    cout << ans << '\n';

```

## 4.3 四元环计数

### 4.3.1 四元环计数

From zpk

- 无向图：类似，由于定向后出度结论过于强大，可以暴力。讨论了三种情况。
- 有向图：缺少题目，但应当类似三元环计数有向形式记录定向边和原边的正反关系。因此此法最强的结论是定向后出度  $O(\sqrt{m})$ ，实际上方法很暴力，应当不难数有向形式的。

```

1 ll n, m; cin >> n >> m;
2 vector<pair<ll, ll>> Edges(m);
3 vector<vector<ll>> G(n + 2);
4 vector<ll> deg(n + 2);
5 for (auto &[i, j] : Edges) cin >> i >> j, ++
    deg[i], ++deg[j];
6 for (auto [i, j] : Edges) {
7     if (deg[i] > deg[j] || (deg[i] == deg[j]
8         && i > j)) swap(i, j);
9     G[i].emplace_back(j), iG[j].emplace_back(i
10 );
11 }
12 ll ans = 0;
13 vector<ll> v1(n + 2), v2(n + 2);
14 for (ll i = 1; i <= n; ++i) {
15     for (auto j : G[i]) for (auto k : G[j]) ++
        v1[k];
16     for (auto j : iG[i]) for (auto k : G[j])
17         ans += v1[k], ++v2[k];
18     for (auto j : G[i]) for (auto k : G[j])
        ans += v1[k] * (v1[k] - 1) / 2, v1[k] =
        0;
19     for (auto j : iG[i]) for (auto k : G[j]) {
20         if (deg[k] > deg[i] || (deg[k] == deg[
21             i] && k > i)) ans += v2[k] * (v2[k]
22                 - 1) / 2;
23         v2[k] = 0;
24     }
25 }

```

```

19     }
20 }
21 cout << ans << '\n';

```

## 4.4 基环树

```

1 #include "../header.cpp"
2 using edge = tuple<int, int, int>;
3 vector<edge> E[MAXN];
4 vector<edge> W;
5 vector<int> C;
6 edge F[MAXN];
7 bool V[MAXN];
8 int I[MAXN], o;
9 void dfs0(int u, int e){
10     V[u] = true;
11     I[u] = ++o;
12     for(auto &[i, v, w] : E[u]) if(i != e){
13         if(V[v]){
14             if(I[v] < I[u]){
15                 for(int p = u; p != v;){
16                     auto &[j, f, x] = F[p];
17                     C.push_back(p);
18                     W.push_back({j, p, x});
19                     p = f;
20                 }
21                 C.push_back(v);
22                 W.push_back({i, v, w});
23             } else {
24                 F[v] = {i, u, w};
25                 dfs0(v, i);
26             }
27         }
28     }
29 }
30 namespace Problem2{
31 // === 删除环上第 i 条边，求直径 ===
32 i64 H[MAXN], A1[MAXN], B1[MAXN], A2[MAXN],
    B2[MAXN], A3[MAXN], B3[MAXN];
33 i64 L[MAXN];
34 i64 dis = 0;
35 void dfs1(int u, int e){
36     for(auto &[i, v, w] : E[u]) if(i != e){
37         if(!V[v]){
38             dfs1(v, i);
39             dis = max(dis, L[u] + w + L[v]);
40             L[u] = max(L[u], L[v] + w);
41         }
42     }
43 }
44 int main(){
45     int n;

```

```

1 // 无向图
2 ll n, m; cin >> n >> m;
3 vector<pair<ll, ll>> Edges(m);
4 vector<vector<ll>> G(n + 2);
5 vector<ll> deg(n + 2);
6 for (auto &[i, j] : Edges) cin >> i >> j, ++
    deg[i], ++deg[j];
7 for (auto [i, j] : Edges) {
8     if (deg[i] > deg[j] || (deg[i] == deg[j]
9         && i > j)) swap(i, j);
10     G[i].emplace_back(j);
11 }
12 vector<ll> val(n + 2);
13 ll ans = 0;
14 for (ll i = 1; i <= n; ++i) {
15     for (auto j : G[i]) ++val[j];
16     for (auto j : G[i]) for (auto k : G[j])
        ans += val[k];
17     for (auto j : G[i]) val[j] = 0;
18 }
19 // 有向图
20 ll n, m; cin >> n >> m;
21 vector<pair<ll, ll>> Edges(m);
22 vector<vector<pll>> G(n + 2);
23 vector<ll> deg(n + 2);
24 for (auto &[i, j] : Edges) cin >> i >> j, ++
    deg[i], ++deg[j];
25 for (auto [i, j] : Edges) {
26     ll flg = 0;
27     if (deg[i] > deg[j] || (deg[i] == deg[j]
28         && i > j)) swap(i, j), flg = 1;
29     G[i].emplace_back(j, flg);
30 }
31 vector<ll> in(n + 2), out(n + 2);
32 ll ans = 0;
33 for (ll i = 1; i <= n; ++i) {
34     for (auto [j, w] : G[i]) w ? (++in[j]) : (
        ++out[j]);
35     for (auto [j, w1] : G[i]) for (auto [k, w2]

```

```

46  cin >> n;
47  for(int i = 1; i ≤ n; ++ i){
48      int u, v, w;
49      cin >> u >> v >> w;
50      E[u].push_back({i, v, w});
51      E[v].push_back({i, u, w});
52  }
53  dfs0(1, 0);
54  memset(V, 0, sizeof(V));
55  for(auto &u : C)
56      V[u] = true;
57  for(auto &u : C){
58      dfs1(u, 0);
59  }
60  int l = 0, r = C.size() - 1;
61  for(int i = l; i ≤ r; ++ i){
62      int x = C[i];
63      if(i > 0)
64          H[i] = H[i - 1] + get<2>(W[i - 1]);
65      A1[i] = L[x] + H[i];
66      B1[i] = L[x] - H[i];
67      A2[i] = L[x] - H[i];
68      B2[i] = L[x] + H[i];
69  }
70  i64 h = H[r] + get<2>(W.back());
71  for(int i = l; i ≤ r; ++ i)
72      A1[i] = max(i == l ? -INFL : A1[i - 1],
73                  L[C[i]] + H[i]),
74      A2[i] = max(i == l ? -INFL : A2[i - 1],
75                  L[C[i]] - H[i]);
76  for(int i = r; i ≥ l; -- i)
77      B1[i] = max(i == r ? -INFL : B1[i + 1],
78                  L[C[i]] - H[i]),
79      B2[i] = max(i == r ? -INFL : B2[i + 1],
80                  L[C[i]] + H[i]);
81  A3[l] = -INFL, B3[r] = -INFL;
82  for(int i = l + 1; i ≤ r; ++ i){
83      int x = C[i];
84      i64 w = A2[i - 1] + L[x] + H[i];
85      A3[i] = max(A3[i - 1], w);
86  }
87  for(int i = r - 1; i ≥ l; -- i){
88      int x = C[i];
89      i64 w = B2[i + 1] + L[x] - H[i];
90      B3[i] = max(B3[i + 1], w);
91  }
92  i64 t = INFL;
93  for(int i = l; i < r; ++ i){
94      i64 d = A1[i] + B1[i + 1] + h;
95      i64 g = A2[i] + B2[i + 1] + 0;
96      d = max({d, dis, A3[i], B3[i + 1]});
97      t = min(t, d);

```

```

94  }
95  t = min(t, max(A3[r], dis));
96  if(t % 2 == 0)
97      cout << t / 2 << ".0" << endl;
98  if(t % 2 == 1)
99      cout << t / 2 << ".5" << endl;
100  return 0;
101 }
102 }
103 namespace Problem3{
104 // === 求最大点权独立集 ===
105 int A[MAXN];
106 i64 X[MAXN], Y[MAXN];
107 i64 P[MAXN][2], Q[MAXN][2];
108 void dfs1(int u, int e){
109     for(auto &[i, v, w] : E[u]) if(i ≠ e){
110         if(!V[v]){
111             dfs1(v, i);
112             Y[u] += max(X[v], Y[v]);
113             X[u] += Y[v];
114         }
115     }
116     X[u] += A[u];
117 }
118 int main(){
119     int n;
120     cin >> n;
121     for(int i = 1; i ≤ n; ++ i){
122         cin >> A[i];
123     }
124     for(int i = 1; i ≤ n; ++ i){
125         int u, v;
126         cin >> u >> v;
127         ++ u, ++ v;
128         E[u].push_back({i, v, 0});
129         E[v].push_back({i, u, 0});
130     }
131     double p;
132     cin >> p;
133     dfs0(1, 0);
134     memset(V, 0, sizeof(V));
135     for(auto &u : C)
136         V[u] = true;
137     for(auto &u : C){
138         dfs1(u, 0);
139     }
140     int l = 0, r = C.size() - 1;
141     P[0][1] = X[C[0]];
142     P[0][0] = -INFL;
143     Q[0][0] = Y[C[0]];
144     Q[0][1] = -INFL;
145     for(int i = l + 1; i ≤ r; ++ i){

```

```

146     int x = C[i];
147     P[i][1] = X[x] + P[i - 1][0];
148     P[i][0] = Y[x] + max(P[i - 1][0], P[i -
149     1][1]);
150     Q[i][1] = X[x] + Q[i - 1][0];
151     Q[i][0] = Y[x] + max(Q[i - 1][0], Q[i -
152     1][1]);
153     i64 ans = max({P[r][0], Q[r][0], Q[r][1]});
154     ;
155     cout << fixed << setprecision(1) << ans *
156     p << endl;
157     return 0;
158 }
159 int main(){
160     return Problem3 :: main();
161 }

```

## 4.5 2-SAT

### 4.5.1 例题

$n$  个变量  $m$  个条件, 形如若  $x_i = a$  则  $y_j = b$ , 找到任意一组可行解或者报告无解。

```

1  #include "../header.cpp"
2  namespace SCC{
3      const int MAXN = 2e6 + 3;
4      vector<int> V[MAXN];
5      stack<int> S;
6      int D[MAXN], L[MAXN], C[MAXN], o, s;
7      bool F[MAXN], I[MAXN];
8      void add(int u, int v){ V[u].push_back(v); }
9      void dfs(int u){
10         L[u] = D[u] = ++ o, S.push(u), I[u] = F[u]
11         = true;
12         for(auto &v : V[u]){
13             if(F[v]){
14                 if(I[v]) L[u] = min(L[u], D[v]);
15             } else {
16                 dfs(v), L[u] = min(L[u], L[v]);
17             }
18         }
19         if(L[u] == D[u]){
20             int c = ++ s;
21             while(S.top() ≠ u){
22                 int v = S.top(); S.pop();
23                 I[v] = false;
24                 C[v] = c;
25             }
26             S.pop(), I[u] = false, C[u] = c;

```



```

26 }
27 }
28 }
29 const int MAXN = 1e6 + 3;
30 int X[MAXN][2], o;
31 int main(){
32     ios :: sync_with_stdio(false);
33     int n, m;
34     cin >> n >> m;
35     for(int i = 1; i ≤ n; ++ i)
36         X[i][0] = ++ o;
37     for(int i = 1; i ≤ n; ++ i)
38         X[i][1] = ++ o;
39     for(int i = 1; i ≤ m; ++ i){
40         int a, x, b, y;
41         cin >> a >> x >> b >> y;
42         SCC :: add(X[a][!x], X[b][y]);
43         SCC :: add(X[b][!y], X[a][x]);
44     }
45     for(int i = 1; i ≤ o; ++ i)
46         if(!SCC :: F[i])
47             SCC :: dfs(i);
48     bool ok = true;
49     for(int i = 1; i ≤ n; ++ i){
50         if(SCC :: C[X[i][0]] = SCC :: C[X[i][1]])
51             ok = false;
52     }
53     if(ok){
54         cout << "POSSIBLE" << endl;
55         for(int i = 1; i ≤ n; ++ i){
56             int a = SCC :: C[X[i][0]];
57             int b = SCC :: C[X[i][1]];
58             if(a < b)
59                 cout << 0 << " ";
60             else
61                 cout << 1 << " ";
62         }
63         cout << endl;
64     } else {
65         cout << "IMPOSSIBLE" << endl;
66     }
67     return 0;
68 }

```

#### 4.6 割点

```

1 #include "../header.cpp"
2 vector<int> V[MAXN];
3 int n, m, o, D[MAXN], L[MAXN];
4 bool F[MAXN], C[MAXN];
5 void dfs(int u, int g){
6     L[u] = D[u] = ++ o, F[u] = true; int s = 0;

```

```

7     for(auto &v : V[u]){
8         if(!F[v]){
9             dfs(v, g), ++ s;
10            L[u] = min(L[u], L[v]);
11            if(u ≠ g && L[v] ≥ D[u]) C[u] = true;
12        } else {
13            L[u] = min(L[u], D[v]);
14        }
15    }
16    if(u = g && s > 1) C[u] = true;
17 }
18 int main(){
19     cin >> n >> m;
20     for(int i = 1; i ≤ m; ++ i){
21         int u, v;
22         cin >> u >> v;
23         V[u].push_back(v);
24         V[v].push_back(u);
25     }
26     for(int i = 1; i ≤ n; ++ i)
27         if(!F[i]) dfs(i, i);
28     vector<int> ANS;
29     for(int i = 1; i ≤ n; ++ i)
30         if(C[i]) ANS.push_back(i);
31     cout << ANS.size() << endl;
32     for(auto &u : ANS)
33         cout << u << " ";
34     return 0;
35 }

```

#### 4.7 边双连通分量

```

1 #include "../header.cpp"
2 vector<vector<int>> A;
3 vector<pair<int, int>> V[MAXN];
4 stack<int> S;
5 int D[MAXN], L[MAXN], o;
6 bool I[MAXN];
7 void dfs(int u, int l){
8     D[u] = L[u] = ++ o; I[u] = true, S.push(u);
9     int s = 0;
10    for(auto &p : V[u]) {
11        int v = p.first, id = p.second;
12        if(id ≠ l){
13            if(D[v]){
14                if(I[v]) L[u] = min(L[u], D[v]);
15            } else {
16                dfs(v, id), L[u] = min(L[u], L[v]), ++ s;
17            }
18        }
19    }
20 }

```

```

19     if(D[u] = L[u]){
20         vector<int> T;
21         while(S.top() ≠ u){
22             int v = S.top(); S.pop();
23             T.push_back(v), I[v] = false;
24         }
25         T.push_back(u), S.pop(), I[u] = false;
26         A.push_back(T);
27     }
28 }

```

#### 4.8 点双连通分量

```

1 #include "../header.cpp"
2 vector<vector<int>> A;
3 vector<int> V[MAXN];
4 stack<int> S;
5 int D[MAXN], L[MAXN], o; bool I[MAXN];
6 void dfs(int u, int f){
7     D[u] = L[u] = ++ o; I[u] = true, S.push(u);
8     int s = 0;
9     for(auto &v : V[u]) if(v ≠ f){
10         if(D[v]) L[u] = min(L[u], D[v]);
11     } else {
12         dfs(v, u), L[u] = min(L[u], L[v]), ++ s;
13         if(L[v] ≥ D[u]){
14             vector<int> T;
15             while(S.top() ≠ v){
16                 int t = S.top(); S.pop();
17                 T.push_back(t), I[t] = false;
18             }
19             T.push_back(v), S.pop(), I[v] = false;
20             T.push_back(u);
21             A.push_back(T);
22         }
23     }
24 }
25 if(f = 0 && s = 0){
26     A.push_back({u});
27 }
28 }

```

#### 4.9 强连通分量

```

1 #include "../header.cpp"
2 vector<int> V[MAXN];
3 stack<int> S;
4 int D[MAXN], L[MAXN], C[MAXN], o, s;
5 bool F[MAXN], I[MAXN];
6 void add(int u, int v){ V[u].push_back(v); }

```

```

7 void dfs(int u){
8     L[u] = D[u] = ++ o, S.push(u), I[u] = F[u] =
      true;
9     for(auto &v : V[u]){
10         if(F[v]){
11             if(I[v]) L[u] = min(L[u], D[v]);
12         } else {
13             dfs(v), L[u] = min(L[u], L[v]);
14         }
15     }
16     if(L[u] == D[u]){
17         int c = ++ s;
18         while(S.top() != u){
19             int v = S.top(); S.pop();
20             I[v] = false;
21             C[v] = c;
22         }
23         S.pop(), I[u] = false, C[u] = c;
24     }
25 }
26 vector<int> ANS[MAXN];
27 int main(){
28     int n, m;
29     cin >> n >> m;
30     for(int i = 1; i ≤ m; ++ i){
31         int u, v;
32         cin >> u >> v;
33         V[u].push_back(v);
34     }
35     for(int i = 1; i ≤ n; ++ i)
36         if(!F[i])
37             dfs(i);
38     for(int i = 1; i ≤ n; ++ i){
39         ANS[C[i]].push_back(i);
40     }
41     cout << s << endl;
42     for(int i = 1; i ≤ n; ++ i) if(F[i]){
43         int c = C[i];
44         sort(ANS[c].begin(), ANS[c].end());
45         for(auto &u : ANS[c])
46             cout << u << " ", F[u] = false;
47         cout << endl;
48     }
49     return 0;
50 }

```

```

2 namespace MCMF{
3     int H[MAXN], V[MAXM], N[MAXM], W[MAXM], F[
      MAXM], o = 1, n;
4     void add(int u, int v, int f, int c){
5         V[++ o] = v, N[o] = H[u], H[u] = o, F[o] =
          f, W[o] = c;
6         V[++ o] = u, N[o] = H[v], H[v] = o, F[o] =
          0, W[o] = -c;
7         n = max(n, u);
8         n = max(n, v);
9     }
10    void clear(){
11        for(int i = 1; i ≤ n; ++ i)
12            H[i] = 0;
13        n = 0, o = 1;
14    }
15    bool I[MAXN];
16    i64 D[MAXN];
17    bool spfa(int s, int t){
18        queue<int> Q;
19        Q.push(s), I[s] = true;
20        for(int i = 1; i ≤ n; ++ i)
21            D[i] = INFL;
22        D[s] = 0;
23        while(!Q.empty()){
24            int u = Q.front(); Q.pop(), I[u] = false
          ;
25            for(int i = H[u]; i = N[i]){
26                const int &v = V[i];
27                const int &f = F[i];
28                const int &w = W[i];
29                if(f && D[u] + w < D[v]){
30                    D[v] = D[u] + w;
31                    if(!I[v]) Q.push(v), I[v] = true;
32                }
33            }
34        }
35        return D[t] != INFL;
36    }
37    int C[MAXN]; bool T[MAXN];
38    pair<i64, i64> dfs(int s, int t, int u, i64
      maxf){
39        if(u == t)
40            return make_pair(maxf, 0);
41        i64 totf = 0;
42        i64 totc = 0;
43        T[u] = true;
44        for(int &i = C[u]; i = N[i]){
45            const int &v = V[i];
46            const int &f = F[i];
47            const int &w = W[i];
48            if(f && D[v] == D[u] + w && !T[v]){

```

```

49            auto p = dfs(s, t, v, min(1ll * F[i],
              maxf));
50            i64 f = p.first;
51            i64 c = p.second;
52            F[i] -= f;
53            F[i ^ 1] += f;
54            totf += f;
55            totc += 1ll * f * W[i] + c;
56            maxf -= f;
57            if(maxf == 0){
58                T[u] = false;
59                return make_pair(totf, totc);
60            }
61        }
62        T[u] = false;
63        return make_pair(totf, totc);
64    }
65    pair<i64, i64> mcmf(int s, int t){
66        i64 ans1 = 0;
67        i64 ans2 = 0;
68        pair<i64, i64> r;
69        while(spfa(s, t)){
70            memcpy(C, H, sizeof(int) * (n + 3));
71            r = dfs(s, t, s, INFL);
72            ans1 += r.first;
73            ans2 += r.second;
74        }
75        return make_pair(ans1, ans2);
76    }
77 }
78
79 int qread();
80 int main(){
81     int n = qread(), m = qread(), s = qread(), t
      = qread();
82     for(int i = 1; i ≤ m; ++ i){
83         int u = qread(), v = qread(), f = qread(),
          c = qread();
84         MCMF :: add(u, v, f, c);
85     }
86     pair<long long, long long> ans = MCMF ::
      mcmf(s, t);
87     printf("%lld %lld\n", ans.first, ans.second)
      ;
88     return 0;
89 }

```

## 5 网络流

### 5.1 费用流

```
1 #include " ../header.cpp"
```

## 5.2 最小割树

### 5.2.1 用法

给定无向图求出最小割树, 点  $u$  和  $v$  作为起点终点的  
最小割为树上  $u$  到  $v$  路径上边权的最小值。

```
1 #include "../header.cpp"
2 namespace Dinic{
3     const long long INF = 1e18;
4     const int SIZ = 1e5 + 3;
5     int n, m;
6     int H[SIZ], V[SIZ], N[SIZ], F[SIZ], t = 1;
7     int add(int u, int v, int f){
8         V[++ t] = v, N[t] = H[u], F[t] = f, H[u] =
9             t;
10        V[++ t] = u, N[t] = H[v], F[t] = 0, H[v] =
11            t;
12        n = max(n, u);
13        n = max(n, v);
14        return t - 1;
15    }
16    void clear(){
17        for(int i = 1; i ≤ n; ++ i)
18            H[i] = 0;
19        n = m = 0, t = 1;
20    }
21    int D[SIZ];
22    bool bfs(int s, int t){
23        queue<int> Q;
24        for(int i = 1; i ≤ n; ++ i)
25            D[i] = 0;
26        Q.push(s), D[s] = 1;
27        while(!Q.empty()){
28            int u = Q.front(); Q.pop();
29            for(int i = H[u]; i = N[i]){
30                const int &v = V[i];
31                const int &f = F[i];
32                if(f ≠ 0 && !D[v]){
33                    D[v] = D[u] + 1;
34                    Q.push(v);
35                }
36            }
37        }
38        return D[t] ≠ 0;
39    }
40    int C[SIZ];
41    long long dfs(int s, int t, int u, long long
42        maxf){
43        if(u == t)
44            return maxf;
45        long long totf = 0;
46        for(int &i = C[u]; i = N[i]){
47            const int &v = V[i];
48            const int &f = F[i];
49            if(D[v] == D[u] + 1){
50                long long resf = dfs(s, t, v, min(maxf
51                    , 1ll * f));
52                totf += resf;
53                maxf -= resf;
54                F[i] -= resf;
55                F[i ^ 1] += resf;
56                if(maxf == 0)
57                    return totf;
58            }
59        }
60        return totf;
61    }
62    long long dinic(int s, int t){
63        long long ans = 0;
64        while(bfs(s, t)){
65            memcpy(C, H, sizeof(int) * (n + 3));
66            ans += dfs(s, t, s, INF);
67        }
68        return ans;
69    }
70 }
71 namespace GHTree{
72     const int MAXN = 500 + 5;
73     const int MAXM = 1500 + 5;
74     const int INF = 1e9;
75     int n, m, U[MAXN], V[MAXN], W[MAXN], A[MAXN],
76         B[MAXN];
77     void add(int u, int v, int w){
78         ++ m;
79         U[m] = u;
80         V[m] = v;
81         W[m] = w;
82         A[m] = Dinic :: add(u, v, w);
83         B[m] = Dinic :: add(v, u, w);
84         n = max(n, u);
85         n = max(n, v);
86     }
87     vector<pair<int, int>> E[MAXN];
88     void build(vector<int> N){
89         int s = N.front();
90         int t = N.back();
91         if(s == t) return;
92         for(int i = 1; i ≤ m; ++ i){
93             int a = A[i]; Dinic :: F[a] = W[i],
94                 Dinic :: F[a ^ 1] = 0;
95             int b = B[i]; Dinic :: F[b] = W[i],
96                 Dinic :: F[b ^ 1] = 0;
97         }
98         int w = Dinic :: dinic(s, t);
99     }
100 }
```

```
44     const int &v = V[i];
45     const int &f = F[i];
46     if(D[v] == D[u] + 1){
47         long long resf = dfs(s, t, v, min(maxf
48             , 1ll * f));
49         totf += resf;
50         maxf -= resf;
51         F[i] -= resf;
52         F[i ^ 1] += resf;
53         if(maxf == 0)
54             return totf;
55     }
56 }
57 return totf;
58 }
59 long long dinic(int s, int t){
60     long long ans = 0;
61     while(bfs(s, t)){
62         memcpy(C, H, sizeof(int) * (n + 3));
63         ans += dfs(s, t, s, INF);
64     }
65     return ans;
66 }
67 namespace GHTree{
68     const int MAXN = 500 + 5;
69     const int MAXM = 1500 + 5;
70     const int INF = 1e9;
71     int n, m, U[MAXN], V[MAXN], W[MAXN], A[MAXN],
72         B[MAXN];
73     void add(int u, int v, int w){
74         ++ m;
75         U[m] = u;
76         V[m] = v;
77         W[m] = w;
78         A[m] = Dinic :: add(u, v, w);
79         B[m] = Dinic :: add(v, u, w);
80         n = max(n, u);
81         n = max(n, v);
82     }
83     vector<pair<int, int>> E[MAXN];
84     void build(vector<int> N){
85         int s = N.front();
86         int t = N.back();
87         if(s == t) return;
88         for(int i = 1; i ≤ m; ++ i){
89             int a = A[i]; Dinic :: F[a] = W[i],
90                 Dinic :: F[a ^ 1] = 0;
91             int b = B[i]; Dinic :: F[b] = W[i],
92                 Dinic :: F[b ^ 1] = 0;
93         }
94         int w = Dinic :: dinic(s, t);
95     }
96 }
```

```
92     E[s].push_back(make_pair(t, w));
93     E[t].push_back(make_pair(s, w));
94     vector<int> P;
95     vector<int> Q;
96     for(auto &u : N){
97         if(Dinic :: D[u] ≠ 0)
98             P.push_back(u);
99         else
100             Q.push_back(u);
101     }
102     build(P), build(Q);
103 }
104 int D[MAXN];
105 int cut(int s, int t){
106     queue<int> Q; Q.push(s);
107     for(int i = 1; i ≤ n; ++ i)
108         D[i] = -1;
109     D[s] = INF;
110     while(!Q.empty()){
111         int u = Q.front(); Q.pop();
112         for(auto &e : E[u]){
113             int v = e.first;
114             int w = e.second;
115             if(D[v] == -1){
116                 D[v] = min(D[u], w);
117                 Q.push(v);
118             }
119         }
120     }
121     return D[t];
122 }
123 }
```

### 5.3 最大流

```
1 #include "../header.cpp"
2 namespace Dinic{
3     const i64 INF = 1e18;
4     const int SIZ = 5e5 + 3;
5     int n;
6     int H[MAXN], V[MAXN], N[MAXN], F[MAXN], t =
7         1;
8     void add(int u, int v, int f){
9         V[++ t] = v, N[t] = H[u], F[t] = f, H[u] =
10             t;
11         V[++ t] = u, N[t] = H[v], F[t] = 0, H[v] =
12             t;
13         n = max(n, u);
14         n = max(n, v);
15     }
16    void clear(){
17        for(int i = 1; i ≤ n; ++ i)
18            H[i] = 0;
19        n = m = 0, t = 1;
20    }
21    int D[SIZ];
22    bool bfs(int s, int t){
23        queue<int> Q;
24        for(int i = 1; i ≤ n; ++ i)
25            D[i] = 0;
26        Q.push(s), D[s] = 1;
27        while(!Q.empty()){
28            int u = Q.front(); Q.pop();
29            for(int i = H[u]; i = N[i]){
30                const int &v = V[i];
31                const int &f = F[i];
32                if(f ≠ 0 && !D[v]){
33                    D[v] = D[u] + 1;
34                    Q.push(v);
35                }
36            }
37        }
38        return D[t] ≠ 0;
39    }
40    int C[SIZ];
41    long long dfs(int s, int t, int u, long long
42        maxf){
43        if(u == t)
44            return maxf;
45        long long totf = 0;
46        for(int &i = C[u]; i = N[i]){
47            const int &v = V[i];
48            const int &f = F[i];
49            if(D[v] == D[u] + 1){
50                long long resf = dfs(s, t, v, min(maxf
51                    , 1ll * f));
52                totf += resf;
53                maxf -= resf;
54                F[i] -= resf;
55                F[i ^ 1] += resf;
56                if(maxf == 0)
57                    return totf;
58            }
59        }
60        return totf;
61    }
62    long long dinic(int s, int t){
63        long long ans = 0;
64        while(bfs(s, t)){
65            memcpy(C, H, sizeof(int) * (n + 3));
66            ans += dfs(s, t, s, INF);
67        }
68        return ans;
69    }
70 }
```

```

15     H[i] = 0;
16     n = 0, t = 1;
17 }
18 i64 D[MAXN];
19 bool bfs(int s, int t){
20     queue<int> Q;
21     for(int i = 1; i ≤ n; ++ i)
22         D[i] = 0;
23     Q.push(s), D[s] = 1;
24     while(!Q.empty()){
25         int u = Q.front(); Q.pop();
26         for(int i = H[u]; i ≤ N[i]){
27             const int &v = V[i];
28             const int &f = F[i];
29             if(f ≠ 0 && !D[v]){
30                 D[v] = D[u] + 1;
31                 Q.push(v);
32             }
33         }
34     }
35     return D[t] ≠ 0;
36 }
37 int C[MAXN];
38 i64 dfs(int s, int t, int u, i64 maxf){
39     if(u == t)
40         return maxf;
41     i64 totf = 0;
42     for(int &i = C[u]; i ≤ N[i]){
43         const int &v = V[i];
44         const int &f = F[i];
45         if(f && D[v] == D[u] + 1){
46             i64 f = dfs(s, t, v, min(1ll * f, maxf));
47             F[i] -= f, F[i ^ 1] += f, totf += f,
48             maxf -= f;
49             if(maxf == 0)
50                 return totf;
51         }
52     }
53     return totf;
54 }
55 i64 dinic(int s, int t){
56     i64 ans = 0;
57     while(bfs(s, t)){
58         memcpy(C, H, sizeof(int) * (n + 3));
59         ans += dfs(s, t, s, INFL);
60     }
61     return ans;
62 }

```

## 5.4 上下界费用流

### 5.4.1 用法

- add(u, v, l, r, c): 连一条容量在  $[l, r]$  的从  $u$  到  $v$  的费用为  $c$  的边;
- solve(): 计算无源汇最小费用可行流;
- solve(s, t): 计算有源汇最小费用最大流。

```

1 #define add add0
2 #include "flow-cost.cpp"
3 #undef add
4 namespace MCMF{
5     i64 cost0;
6     int G[MAXN];
7     void add(int u, int v, int l, int r, int c){
8         G[v] += l;
9         G[u] -= l;
10        cost0 += 1ll * l * c;
11        add0(u, v, r - l, c);
12    }
13    i64 solve(){
14        int s = ++ n;
15        int t = ++ n;
16        i64 sum = 0;
17        for(int i = 1; i ≤ n - 2; ++ i){
18            if(G[i] < 0)
19                add0(i, t, -G[i], 0);
20            else
21                add0(s, i, G[i], 0), sum += G[i];
22        }
23        auto res = mcmf(s, t);
24        if(res.first ≠ sum)
25            return -1;
26        return res.second + cost0;
27    }
28    i64 solve(int s0, int t0){
29        add0(t0, s0, INF, 0);
30        int s = ++ n;
31        int t = ++ n;
32        i64 sum = 0;
33        for(int i = 1; i ≤ n - 2; ++ i){
34            if(G[i] < 0)
35                add0(i, t, -G[i], 0);
36            else
37                add0(s, i, G[i], 0), sum += G[i];
38        }
39        auto res = mcmf(s, t);
40        if(res.first ≠ sum)
41            return -1;
42        return res.second + cost0;
43    }

```

44 }

## 5.5 上下界最大流

### 5.5.1 用法

- add(u, v, l, r, c): 连一条容量在  $[l, r]$  的从  $u$  到  $v$  的边;
- solve(): 检查是否存在无源汇可行流;
- solve(s, t): 计算有源汇最大流。

```

1 #define add add0
2 #include "flow-max.cpp"
3 #undef add
4 namespace Dinic{
5     int G[MAXN];
6     void add(int u, int v, int l, int r){
7         G[v] += l;
8         G[u] -= l;
9         add0(u, v, r - l);
10    }
11    void clear(){
12        for(int i = 1; i ≤ t; ++ i){
13            N[i] = F[i] = V[i] = 0;
14        }
15        for(int i = 1; i ≤ n; ++ i){
16            H[i] = G[i] = C[i] = 0;
17        }
18        t = 1, n = 0;
19    }
20    bool solve(){
21        int s = ++ n;
22        int t = ++ n;
23        i64 sum = 0;
24        for(int i = 1; i ≤ n - 2; ++ i){
25            if(G[i] < 0)
26                add0(i, t, -G[i]);
27            else
28                add0(s, i, G[i]), sum += G[i];
29        }
30        auto res = dinic(s, t);
31        if(res ≠ sum)
32            return true;
33        return false;
34    }
35    i64 solve(int s0, int t0){
36        add0(t0, s0, INF);
37        int s = ++ n;
38        int t = ++ n;
39        i64 sum = 0;
40        for(int i = 1; i ≤ n - 2; ++ i){

```

```

41     if(G[i] < 0)
42         add0(i, t, -G[i]);
43     else
44         add0(s, i, G[i]), sum += G[i];
45 }
46 auto res = dinic(s, t);
47 if(res != sum)
48     return -1;
49 return dinic(s0, t0);
50 }
51 }

```

## 6 数学

### 6.1 线性代数

#### 6.1.1 行列式

```

1 #include "../header.cpp"
2 struct Mat{
3     int n, m, W[MAXN][MAXN];
4     Mat(int _n = 0, int _m = 0){
5         n = _n, m = _m;
6         for(int i = 1; i ≤ n; ++ i)
7             for(int j = 1; j ≤ m; ++ j)
8                 W[i][j] = 0;
9     }
10 };
11 int mat_det(Mat a){
12     int ans = 1;
13     const int &n = a.n;
14     for(int i = 1; i ≤ n; ++ i){
15         int f = -1;
16         for(int j = i; j ≤ n; ++ j) if(a.W[j][i] != 0){
17             f = j; break;
18         }
19         if(f == -1) return 0;
20         if(f != i){
21             for(int j = 1; j ≤ n; ++ j)
22                 swap(a.W[i][j], a.W[f][j]);
23             ans = MOD - ans;
24         }
25         for(int j = i + 1; j ≤ n; ++ j) if(a.W[j][i]){
26             while(a.W[j][i]){
27                 int u = a.W[i][i], v = a.W[j][i];
28                 if(u > v){
29                     for(int k = 1; k ≤ n; ++ k)
30                         swap(a.W[i][k], a.W[j][k]);
31                     ans = MOD - ans, swap(u, v);

```

```

32         }
33         int rate = v / u;
34         for(int k = 1; k ≤ n; ++ k){
35             a.W[j][k] = (a.W[j][k] - 1ll * rate
36                 * a.W[i][k] % MOD + MOD) % MOD;
37         }
38     }
39 }
40 for(int i = 1; i ≤ n; ++ i)
41     ans = 1ll * ans * a.W[i][i] % MOD;
42 return ans;
43 }
44 int main(){
45     int n; cin >> n;
46     Mat A(n, n);
47     for(int i = 1; i ≤ n; ++ i)
48         for(int j = 1; j ≤ n; ++ j)
49             cin >> A.W[i][j], A.W[i][j] %= MOD;
50     cout << mat_det(A) << endl;
51     return 0;
52 }

```

#### 6.1.2 高斯消元与求秩 (实数)

```

1 #include "../header.cpp"
2 const double EPS = 1e-9;
3 struct Mat{
4     int n, m;
5     double W[MAXN][MAXN];
6     Mat(int _n = 0, int _m = 0){
7         n = _n;
8         m = _m;
9         for(int i = 1; i ≤ n; ++ i)
10             for(int j = 1; j ≤ m; ++ j)
11                 W[i][j] = 0;
12 }
13 };
14 bool zero(double f){
15     return fabs(f) < EPS;
16 }
17 int mat_rank(Mat &a){
18     const int &n = a.n;
19     const int &m = a.m;
20     int cnt = 0;
21     for(int i = 1; i ≤ m; ++ i){
22         int p = cnt + 1;
23         int f = -1;
24         for(int j = p; j ≤ n; ++ j){
25             if(!zero(a.W[j][i])){
26                 f = j;
27                 break;

```

```

28     }
29 }
30 if(f == -1)
31     continue;
32 if(f != p){
33     for(int j = 1; j ≤ m; ++ j)
34         swap(a.W[p][j], a.W[f][j]);
35 }
36 ++ cnt;
37 for(int j = p + 1; j ≤ n; ++ j){
38     double rate = a.W[j][i] / a.W[p][i];
39     for(int k = 1; k ≤ m; ++ k){
40         a.W[j][k] -= rate * a.W[p][k];
41     }
42 }
43 }
44 return cnt;
45 }
46 double X[MAXN];
47 int main(){
48     int n;
49     cin >> n;
50     Mat A(n, n);
51     Mat T(n, n + 1);
52     for(int i = 1; i ≤ n; ++ i){
53         for(int j = 1; j ≤ n; ++ j)
54             cin >> A.W[i][j];
55         for(int j = 1; j ≤ n; ++ j)
56             T.W[i][j] = A.W[i][j];
57         cin >> T.W[i][n + 1];
58     }
59     int res1 = mat_rank(A);
60     int res2 = mat_rank(T);
61     if(res1 != res2)
62         cout << -1 << endl;
63     else
64         if(res2 < n)
65             cout << 0 << endl;
66     else {
67         for(int i = n; i ≥ 1; -- i){
68             X[i] = T.W[i][n + 1] / T.W[i][i];
69             for(int j = i - 1; j ≥ 1; -- j){
70                 double rate = T.W[j][i] / T.W[i][i];
71                 T.W[j][i] -= rate * T.W[i][i];
72                 T.W[j][n + 1] -= rate * T.W[i][n + 1];
73             }
74         }
75         for(int i = 1; i ≤ n; ++ i)
76             cout << "x" << i << "=" << fixed <<
77                 setprecision(2) << X[i] << endl;
78     }
79     return 0;

```



```
79 }
```

### 6.1.3 高斯消元与求秩 (整数)

```
1 #include "../header.cpp"
2 struct Mat{
3     int n, m;
4     int W[MAXN][MAXN];
5     Mat(int _n = 0, int _m = 0){
6         n = _n;
7         m = _m;
8         for(int i = 1; i ≤ n; ++ i)
9             for(int j = 1; j ≤ m; ++ j)
10                 W[i][j] = 0;
11     }
12 };
13 int power(int a, int b){
14     int r = 1;
15     while(b){
16         if(b & 1) r = 1ll * r * a % MOD;
17         b >>= 1, a = 1ll * a * a % MOD;
18     }
19     return r;
20 }
21 int inv(int x){
22     return power(x, MOD - 2);
23 }
24 int mat_rank(Mat &a){
25     const int &n = a.n;
26     const int &m = a.m;
27     int cnt = 0;
28     for(int i = 1; i ≤ m; ++ i){
29         int p = cnt + 1;
30         int f = -1;
31         for(int j = p; j ≤ n; ++ j){
32             if(a.W[j][i] ≠ 0){
33                 f = j;
34                 break;
35             }
36         }
37         if(f == -1) continue;
38         if(f ≠ p){
39             for(int j = 1; j ≤ m; ++ j)
40                 swap(a.W[p][j], a.W[f][j]);
41         }
42         ++ cnt;
43         int invp = inv(a.W[p][i]);
44         for(int j = p + 1; j ≤ n; ++ j){
45             int rate = 1ll * a.W[j][i] * invp % MOD;
46             for(int k = 1; k ≤ m; ++ k){
47
```

```
48         a.W[j][k] = (a.W[j][k] - 1ll * rate *
49             a.W[p][k] % MOD + MOD) % MOD;
50     }
51 }
52 return cnt;
53 }
54 int X[MAXN];
55 int main(){
56     int n;
57     cin >> n;
58     Mat A(n, n);
59     Mat T(n, n + 1);
60     for(int i = 1; i ≤ n; ++ i){
61         for(int j = 1; j ≤ n; ++ j)
62             cin >> A.W[i][j];
63         for(int j = 1; j ≤ n; ++ j)
64             T.W[i][j] = A.W[i][j];
65         cin >> T.W[i][n + 1];
66     }
67     int res1 = mat_rank(A);
68     int res2 = mat_rank(T);
69     if(res1 ≠ res2)
70         cout << -1 << endl;
71     else
72         if(res2 < n)
73             cout << 0 << endl;
74     else {
75         for(int i = n; i ≥ 1; -- i){
76             int invp = inv(T.W[i][i]);
77             X[i] = 1ll * T.W[i][n + 1] * invp % MOD;
78             for(int j = i - 1; j ≥ 1; -- j){
79                 int rate = 1ll * T.W[j][i] * invp %
80                     MOD;
81                 T.W[j][i] = (T.W[j][i] - 1ll *
82                     rate * T.W[i][i] % MOD + MOD) %
83                     MOD;
84                 T.W[j][n + 1] = (T.W[j][n + 1] - 1ll *
85                     rate * T.W[i][n + 1] % MOD + MOD) %
86                     MOD;
87             }
88             for(int i = 1; i ≤ n; ++ i)
89                 cout << "x" << i << "=" << X[i] << endl;
90         }
91     }
92     return 0;
93 }
```

### 6.1.4 矩阵求逆

```
1 #include<bits/stdc++.h>
2 using namespace std;
```

```
3 using i64 = long long;
4 const int INF = 1e9;
5 const i64 INFL = 1e18;
6 const int MAXN = 400 + 3;
7 const int MOD = 1e9 + 7;
8 struct Mat{
9     int n, m;
10    int W[MAXN][MAXN];
11    Mat(int _n = 0, int _m = 0){
12        n = _n, m = _m;
13        for(int i = 1; i ≤ n; ++ i)
14            for(int j = 1; j ≤ m; ++ j)
15                W[i][j] = 0;
16    }
17 };
18 int power(int a, int b){
19     int r = 1;
20     while(b){
21         if(b & 1) r = 1ll * r * a % MOD;
22         b >>= 1, a = 1ll * a * a % MOD;
23     }
24     return r;
25 }
26 int inv(int x){
27     return power(x, MOD - 2);
28 }
29 bool mat_inv(Mat &a){
30     const int &n = a.n;
31     Mat b(n, n);
32     for(int i = 1; i ≤ n; ++ i)
33         b.W[i][i] = 1;
34     for(int i = 1; i ≤ n; ++ i){
35         int f = -1;
36         for(int j = i; j ≤ n; ++ j) if(a.W[j][i]
37             ≠ 0){
38             f = j;
39             break;
40         }
41         if(f == -1){
42             return false;
43         }
44         if(f ≠ i){
45             for(int j = 1; j ≤ n; ++ j)
46                 swap(a.W[i][j], a.W[f][j]),
47                 swap(b.W[i][j], b.W[f][j]);
48         }
49         int invp = inv(a.W[i][i]);
50         for(int j = i + 1; j ≤ n; ++ j){
51             int rate = 1ll * a.W[j][i] * invp
52                 % MOD;
53             for(int k = 1; k ≤ n; ++ k){
54                 a.W[j][k] = (a.W[j][k] - 1ll *
55
```

```

        rate * a.W[i][k] % MOD +
        MOD) % MOD;
        b.W[j][k] = (b.W[j][k] - 1ll *
        rate * b.W[i][k] % MOD +
        MOD) % MOD;
    }
}
for(int i = n; i ≥ 1; -- i){
    int invp = inv(a.W[i][i]);
    for(int j = 1; j ≤ n; ++ j){
        a.W[i][j] = 1ll * a.W[i][j] * invp
        % MOD;
        b.W[i][j] = 1ll * b.W[i][j] * invp
        % MOD;
    }
    for(int j = i - 1; j ≥ 1; -- j){
        int rate = 1ll * a.W[j][i] % MOD;
        for(int k = 1; k ≤ n; ++ k){
            a.W[j][k] = (a.W[j][k] - 1ll *
            rate * a.W[i][k] % MOD +
            MOD) % MOD;
            b.W[j][k] = (b.W[j][k] - 1ll *
            rate * b.W[i][k] % MOD +
            MOD) % MOD;
        }
    }
}
for(int i = 1; i ≤ n; ++ i)
    for(int j = 1; j ≤ n; ++ j)
        a.W[i][j] = b.W[i][j];
return true;
}
int X[MAXN];
int main(){
    int n;
    cin >> n;
    Mat A(n, n);
    for(int i = 1; i ≤ n; ++ i)
        for(int j = 1; j ≤ n; ++ j)
            cin >> A.W[i][j];
    bool res = mat_inv(A);
    if(res == false){
        cout << "No Solution" << endl;
    } else {
        for(int i = 1; i ≤ n; ++ i)
            for(int j = 1; j ≤ n; ++ j)
                cout << A.W[i][j] << " \n"[j
                == n];
    }
    return 0;
}

```

### 6.1.5 矩阵树

**LGV 定理叙述** 设  $G$  是一张有向无环图, 边带权, 每个点的度数有限。给定起点集合  $A = \{a_1, a_2, \dots, a_n\}$ , 终点集合  $B = \{b_1, b_2, \dots, b_n\}$ 。

- 一段路径  $p: v_0 \xrightarrow{w_1} v_1 \xrightarrow{w_2} v_2 \rightarrow \dots \rightarrow^{w_k} v_k$  的边权被定义为  $\omega(p) = \prod w_i$ 。
- 一对顶点  $(a, b)$  的权值定义为  $e(a, b) = \sum_{p: a \rightarrow b} \omega(p)$ 。

设矩阵  $M$  如下:

$$M = \begin{pmatrix} e(a_1, b_1) & e(a_1, b_2) & \cdots & e(a_1, b_n) \\ e(a_2, b_1) & e(a_2, b_2) & \cdots & e(a_2, b_n) \\ \vdots & \vdots & \ddots & \vdots \\ e(a_n, b_1) & e(a_n, b_2) & \cdots & e(a_n, b_n) \end{pmatrix}$$

从  $A$  到  $B$  得到一个不交的路径组  $p = (p_1, p_2, \dots, p_n)$ , 其中从  $a_i$  到达  $b_{\pi_i}$ ,  $\pi$  是一个排列。定义  $\sigma(\pi)$  是  $\pi$  逆序对的数量。

给出 LGV 的叙述如下:

$$\det(M) = \sum_{p: A \rightarrow B} (-1)^{\sigma(\pi)} \prod_{i=1}^n \omega(p_i)$$

可以将边权视作边的重数, 那么  $e(a, b)$  就可以视为从  $a$  到  $b$  的不同路径方案数。

**矩阵树定理** 对于无向图,

- 定义度数矩阵  $D_{i,j} = [i = j] \deg(i)$ ;
- 定义邻接矩阵  $E_{i,j} = E_{j,i}$  是从  $i$  到  $j$  的边数个数;
- 定义拉普拉斯矩阵  $L = D - E$ 。

对于无向图的矩阵树定理叙述如下:

$$t(G) = \det(L_i) = \frac{1}{n} \lambda_1 \lambda_2 \cdots \lambda_{n-1}$$

其中  $L_i$  是将  $L$  删去第  $i$  行和第  $i$  列得到的子式。

对于有向图, 类似于无向图定义入度矩阵、出度矩阵、邻接矩阵  $D^{\text{in}}, D^{\text{out}}, E$ , 同时定义拉普拉斯矩阵  $L^{\text{in}} = D^{\text{in}} - E, L^{\text{out}} = E$ 。

$$t^{\text{leaf}}(G, k) = \det(L_k^{\text{in}})$$

$$t^{\text{root}}(G, k) = \det(L_k^{\text{out}})$$

其中  $t^{\text{leaf}}(G, k)$  表示以  $k$  为根的叶向树,  $t^{\text{root}}(G, k)$  表示以  $k$  为根的根向树。

**BEST 定理** 对于一个有向欧拉图  $G$ , 记点  $i$  的出度为  $\text{out}_i$ , 同时  $G$  的根向生成树个数为  $T$ 。  $T$  可以任意选取根。则  $G$  的本质不同的欧拉回路个数为:

$$T \prod_i (\text{out}_i - 1)!$$

```

1 #include "../header.cpp"
2 struct Mat{
3     int n, m;
4     int W[MAXN][MAXN];
5     Mat(int _n = 0, int _m = 0){
6         n = _n;
7         m = _m;
8         for(int i = 1; i ≤ n; ++ i)
9             for(int j = 1; j ≤ m; ++ j)
10                 W[i][j] = 0;
11     }
12 };
13 int mat_det(Mat a){
14     int ans = 1;
15     const int &n = a.n;
16     for(int i = 1; i ≤ n; ++ i){
17         int f = -1;
18         for(int j = i; j ≤ n; ++ j) if(a.W[j][i] ≠
19             0){
20             f = j;
21             break;
22         }
23         if(f == -1){
24             return 0;
25         }
26         if(f ≠ i){
27             for(int j = 1; j ≤ n; ++ j)
28                 swap(a.W[i][j], a.W[f][j]);
29             ans = MOD - ans;
30         }
31     }
32 }

```

```

30 for(int j = i + 1; j ≤ n; ++ j) if(a.W[j][i]
31   ){
32   while(a.W[j][i]){
33     int u = a.W[i][i];
34     int v = a.W[j][i];
35     if(u > v){
36       for(int k = 1; k ≤ n; ++ k)
37         swap(a.W[i][k], a.W[j][k]);
38       ans = MOD - ans;
39       swap(u, v);
40     }
41     int rate = v / u;
42     for(int k = 1; k ≤ n; ++ k){
43       a.W[j][k] = (a.W[j][k] - 1ll * rate
44         * a.W[i][k] % MOD + MOD) % MOD;
45     }
46   }
47   for(int i = 1; i ≤ n; ++ i)
48     ans = 1ll * ans * a.W[i][i] % MOD;
49   return ans;
50 }
51 int D[MAXN];
52 int W[MAXN][MAXN];
53 int main(){
54   int n, m, t;
55   cin >> n >> m >> t;
56   for(int i = 1; i ≤ m; ++ i){
57     int u, v, w;
58     cin >> u >> v >> w;
59     if(u ≠ v){
60       if(t == 0){ // 无向图
61         D[u] = (D[u] + w) % MOD;
62         D[v] = (D[v] + w) % MOD;
63         W[u][v] = (W[u][v] + w) % MOD;
64         W[v][u] = (W[v][u] + w) % MOD;
65       } else { // 叶向树
66         D[v] = (D[v] + w) % MOD;
67         W[u][v] = (W[u][v] + w) % MOD;
68       }
69     }
70   }
71   Mat A(n - 1, n - 1);
72   for(int i = 2; i ≤ n; ++ i)
73     for(int j = 2; j ≤ n; ++ j) // 以 1 为根的
74       // 叶向树
75       A.W[i - 1][j - 1] = MOD - W[i][j];
76   for(int i = 2; i ≤ n; ++ i)
77     A.W[i - 1][i - 1] = (D[i] + A.W[i - 1][i - 1] % MOD);
78   cout << mat_det(A) << endl;

```

```

78 return 0;
79 }

```

## 6.2 大步小步

### 6.2.1 用法

给定  $a, p$  求出  $x$  使得  $a^x = y \pmod{p}$ , 其中  $p$  为质数。

```

1 #include "../header.cpp"
2 namespace BSGS {
3   unordered_map <int, int> M;
4   int solve(int a, int y, int p){ // a ^ x =
5     // y (mod p)
6     M.clear();
7     int B = sqrt(p);
8     int w1 = y, u1 = power(a, p - 2, p);
9     int w2 = 1, u2 = power(a, B, p);
10    for(int i = 0; i < B; ++ i){
11      M[w1] = i;
12      w1 = 1ll * w1 * u1 % p;
13    }
14    for(int i = 0; i < p / B; ++ i){
15      if(M.count(w2)){
16        return i * B + M[w2];
17      }
18      w2 = 1ll * w2 * u2 % p;
19    }
20    return -1;
21  }

```

## 6.3 中国剩余定理

### 6.3.1 定理

对于线性方程:

$$\begin{cases} x \equiv a_1 \pmod{m_1} \\ x \equiv a_2 \pmod{m_2} \\ \dots \\ x \equiv a_n \pmod{m_n} \end{cases}$$

如果  $a_i$  两两互质, 可以得到  $x$  的解  $x \equiv L \pmod{M}$ ,

其中  $M = \prod m_i$ , 而  $L$  由下式给出:

$$L = \left( \sum a_i m_i \times ((M/m_i)^{-1} \pmod{m_i}) \right) \pmod{M}$$

```

1 #include "../header.cpp"
2 i64 A[MAXN], B[MAXN], M = 1;
3 i64 exgcd(i64 a, i64 b, i64 &x, i64 &y);
4 int main(){
5   int n; cin >> n;
6   for(int i = 1; i ≤ n; ++ i){
7     cin >> B[i] >> A[i];
8     M = M * B[i];
9   }
10  i64 L = 0;
11  for(int i = 1; i ≤ n; ++ i){
12    i64 m = M / B[i], b, k;
13    exgcd(m, B[i], b, k);
14    L = (L + (__int128)A[i] * m * b) % M;
15  }
16  L = (L % M + M) % M;
17  cout << L << endl;
18  return 0;
19 }

```

## 6.4 狄利克雷前缀和

### 6.4.1 用法

计算:

$$s(i) = \sum_{d|i} f_d$$

```

1 #include "../header.cpp"
2 unsigned A[MAXN];
3 int p, P[MAXN]; bool V[MAXN];
4 void solve(int n){
5   for(int i = 2; i ≤ n; ++ i){
6     if(!V[i]){
7       P[++ p] = i;
8       for(int j = 1; j ≤ n / i; ++ j){ // 前缀
9         // 和
10        A[j * i] += A[j];
11      }
12      for(int j = 1; j ≤ p && P[j] ≤ n / i; ++ j){
13        V[i * P[j]] = true;
14        if(i % P[j] == 0) break;
15      }
16    }
17  }

```

## 6.5 万能欧几里得

### 6.5.1 类欧几里得 (万能欧几里得)

From *zpk*

一种神奇递归, 对  $y = \left\lfloor \frac{Ax+B}{C} \right\rfloor$  向右和向上走的每一步进行压缩, 做到  $O(\log V)$  复杂度。其中  $A \geq C$  就是直接压缩, 向右之后必有至少  $\lfloor A/C \rfloor$  步向上。  $A < C$  实际上切换  $x, y$  轴后, 相当于压缩了一个上取整折线, 而上取整下取整可以互化, 便又可以递归。

代码中从  $(0, 0)$  走到  $(n, \lfloor (An+B)/C \rfloor)$ , 假设了  $A, B, C \geq 0, C \neq 0$  (类欧基本都作此假设),  $U, R$  矩阵是从右往左乘的, 对列向量进行优化, 和实际操作顺序恰好相反。快速幂的  $\log$  据说可以被递归过程均摊掉, 实际上并不会导致变成两个  $\log$ 。

```
1 Matrix solve(ll n, ll A, ll B, ll C, Matrix R,
2   Matrix U) { // (0, 0) 走到 (n, (An+B)/C)
3   if (A ≥ C) return solve(n, A % C, B, C, U
4     .qpow(A / C) * R, U);
5   ll l = B / C, r = (A * n + B) / C;
6   if (l == r) return R.qpow(n) * U.qpow(l);
7   // l = r → l = r or A = 0 or n = 0.
8   ll p = (C * r - B - 1) / A + 1;
9   return R.qpow(n - p) * U * solve(r - l -
10     1, C, C - B % C + A - 1, A, U, R) * U.
11     qpow(l);
12 }
```

## 6.6 扩展欧几里得

### 6.6.1 内容

给定  $a, b$ , 求出  $ax + by = \gcd(a, b)$  的一组  $x, y$ 。

```
1 int exgcd(int a, int b, int &x, int &y){
2   if(a == 0){
3     x = 0, y = 1; return b;
4   } else {
5     int x0 = 0, y0 = 0;
6     int d = exgcd(b % a, a, x0, y0);
7     x = y0 - (b / a) * x0;
8     y = x0;
9     return d;
10  }
11 }
```

## 6.7 快速离散对数

### 6.7.1 用法

给定原根  $g$  以及模数  $\text{mod}$ ,  $T$  次询问  $x$  的离散对数。  
复杂度  $O(\text{mod}^{2/3} + T \log \text{mod})$ 。

```
1 #include "../header.cpp"
2 namespace BSGS {
3   unordered_map<int, int> M;
4   int B, U, P, g;
5   void init(int g, int P0, int B0);
6   int solve(int y);
7 }
8 const int MAXN = 1e5 + 3;
9 int H[MAXN], P[MAXN], H0, p, h, g, mod;
10 bool V[MAXN];
11 int solve(int x){
12   if(x ≤ h) return H[x];
13   int v = mod / x, r = mod % x;
14   if(r < x - r) return ((H0 + solve(r)) % (mod
15     - 1) - H[v] + mod - 1) % (mod - 1);
16   else return (solve(x - r) - H[v +
17     1] + mod - 1) % (mod - 1);
18 }
19 int main(){
20   ios :: sync_with_stdio(false);
21   cin.tie(nullptr);
22   cin >> g >> mod;
23   h = sqrt(mod) + 1;
24   BSGS :: init(g, mod, sqrt(1ll * mod * sqrt(
25     mod) / log10(mod)));
26   H0 = BSGS :: solve(mod - 1);
27   H[1] = 0;
28   for(int i = 2; i ≤ h; ++ i){
29     if(!V[i]){
30       P[++ p] = i;
31       H[i] = BSGS :: solve(i);
32     }
33   }
34   for(int j = 1; j ≤ p && P[j] ≤ h / i; ++ j)
35     {
36       int &p = P[j];
37       H[i * p] = (H[i] + H[p]) % (mod - 1);
38       V[i * p] = true;
39       if(i % p == 0) break;
40     }
41   int T; cin >> T;
42   while(T --){
43     int x; cin >> x;
44     cout << solve(x) << "\n";
45   }
46   return 0;
47 }
```

43

}

## 6.8 快速最大公约数

### 6.8.1 用法

已知小值域  $m$  以及  $n$  次询问,  $O(m)$  预处理,  $O(1)$  单次查询  $x, y$  的最大公约数。

```
1 #include "../header.cpp"
2 const int MAXT = 1e6 + 3;
3 int G[MAXM][MAXM], T[MAXT][3];
4 int A[MAXN], B[MAXN], o = 1e6, h = 1e3, V[MAXT
5   ];
6 int tgcd(int a, int b){
7   if(a ≤ h && b ≤ h) return G[a][b];
8   return a == b ? a : 1;
9 }
10 int qgcd(int a, int b){
11   int ans = 1;
12   up(0, 2, i){
13     if(T[b][i] > h){
14       if(a % T[b][i] == 0) a /= T[b][i], ans
15         *= T[b][i];
16     } else {
17       int d = G[a % T[b][i]][T[b][i]];
18       a /= d, ans *= d;
19     }
20   }
21   return ans;
22 }
23 int main(){
24   ios :: sync_with_stdio(false);
25   cin.tie(nullptr);
26   up(1, h, i) G[0][i] = G[i][0] = i;
27   up(1, h, i) up(1, h, j){
28     if(i ≥ j) G[i][j] = G[i - j][j];
29     else G[i][j] = G[i][j - i];
30   }
31   up(2, o, i) if(!V[i]){
32     V[i] = i;
33     for(int j = 2; i * j ≤ o; ++ j)
34       if(!V[i * j]) V[i * j] = i;
35   }
36   T[1][0] = T[1][1] = T[1][2] = 1;
37   up(2, o, i){
38     int p = V[i];
39     int a = T[i / p][0];
40     int b = T[i / p][1];
41     int c = T[i / p][2];
42     int x, y, z;
43     if(p ≥ h){
44       int
```

```

42     x = 1, y = i / p, z = p;
43 } else {
44     if(c * p ≤ h){
45         x = a, y = b, z = c * p;
46     }
47     else if(b * p ≤ h){
48         x = a, y = b * p, z = c;
49         if(y > z) swap(y, z);
50     }
51     else if(a * p ≤ h){
52         x = a * p, y = b, z = c;
53         if(x > y) swap(x, y);
54         if(y > z) swap(y, z);
55     } else {
56         x = a * b, y = c, z = p;
57         if(x > y) swap(x, y);
58         if(y > z) swap(y, z);
59         if(x > z) swap(x, z);
60     }
61 }
62 T[i][0] = x;
63 T[i][1] = y;
64 T[i][2] = z;
65 }
66 int n;
67 cin >> n;
68 up(1, n, i) cin >> A[i];
69 up(1, n, i) cin >> B[i];
70 up(1, n, i){
71     int s = 0, u = 1;
72     up(1, n, j){
73         int d = gcd(A[i], B[j]);
74         u = 1ll * u * i % MOD;
75         s = (s + 1ll * d * u) % MOD;
76     }
77     printf("%d\n", s);
78 }
79 return 0;
80 }

```

## 6.9 原根

### 6.9.1 用法

计算  $P$  的最小原根。

原根表, 其中  $P = r \times 2^k$ , 对应原根为  $g$ 。

Prime	$g$	Prime	$g$
104857601	3	7881299347898369	6
167772161	3	31525197391593473	3
469762049	3	180143985094819841	6
998244353	3	1945555039024054273	5
1004535809	3	4179340454199820289	3

```

1 #include "../header.cpp"
2 int getphi(int x){
3     int t = x, r = x;
4     for(int i = 2; i ≤ x / i; ++ i){
5         if(t % i == 0){
6             r = r / i * (i - 1);
7             while(t % i == 0)
8                 t /= i;
9         }
10    }
11    if(t ≠ 1){
12        r = r / t * (t - 1);
13    }
14    return r;
15 }
16 vector<int> getprime(int x){
17     vector<int> p;
18     int t = x;
19     for(int i = 2; i ≤ x / i; ++ i){
20         if(t % i == 0){
21             p.push_back(i);
22             while(t % i == 0)
23                 t /= i;
24         }
25    }
26    if(t ≠ 1)
27        p.push_back(x);
28    return p;
29 }
30 bool test(int g, int m, int mm, vector<int> &p)
31 ){
32     for(auto &p: P){
33         if(power(g, mm / p, m) == 1)
34             return false;
35     }
36     return true;
37 }
38 int get_genshin(int m){
39     int mm = getphi(m);
40     vector<int> P = getprime(mm);

```

```

40 for(int i = 1; ++ i){
41     if(test(i, m, mm, P))
42         return i;
43 }
44 }

```

## 6.10 快速乘法逆元 (离线)

### 6.10.1 用法

离线计算  $x = [x_1, x_2, \dots, x_n]$  在模  $p$  意义下的逆元。

```

1 #include "../header.cpp"
2 int A[MAXN], B[MAXN];
3 int P[MAXN], Q[MAXN];
4 int main(){
5     ios :: sync_with_stdio(false);
6     cin.tie(nullptr);
7     int n, p, K, S = 1;
8     cin >> n >> p >> K;
9     P[0] = 1;
10    for(int i = 1; i ≤ n; ++ i){
11        cin >> A[i];
12        P[i] = 1ll * P[i - 1] * A[i] % p;
13    }
14    Q[n] = power(P[n], p - 2, p);
15    for(int i = n; i ≥ 1; -- i){
16        Q[i - 1] = 1ll * Q[i] * A[i] % p;
17        B[i] = 1ll * Q[i] * P[i - 1] % p;
18    }
19    int ans = 0;
20    for(int i = 1; i ≤ n; ++ i){
21        S = 1ll * S * K % p;
22        ans = (ans + 1ll * S * B[i]) % p;
23    }
24    cout << ans << "\n";
25    return 0;
26 }

```

## 6.11 快速乘法逆元 (在线)

### 6.11.1 用法

在线计算  $x = [x_1, x_2, \dots, x_n]$  在模  $p$  意义下的逆元。

```

1 #include "../header.cpp"
2 pair<int, int> F[MAXN], G[MAXN];
3 int I[MAXN];
4 using u32 = uint32_t;
5 u32 read(u32 &seed);
6 int main(){
7     ios :: sync_with_stdio(false);
8     cin.tie(nullptr);

```



```

9  u32 seed;
10 int n, p;
11 cin >> n >> p >> seed;
12 int m = pow(p, 1.0 / 3.0);
13 I[1] = 1;
14 for(int i = 2; i ≤ p / m; ++ i){
15     I[i] = 1ll * (p / i) * (p - I[p % i]) % p;
16 }
17 for(int i = 1; i < m; ++ i){
18     for(int j = i + 1; j ≤ m; ++ j){
19         if(!F[i * m * m / j].second){
20             F[i * m * m / j] = { i, j };
21             G[i * m * m / j] = { i, j };
22         }
23     }
24 }
25 F[0] = G[0] = { 0, 1 };
26 F[m * m] = G[m * m] = { 1, 1 };
27 for(int i = 1; i < m * m; ++ i) if(!F[i].second)
28     F[i] = F[i - 1];
29 for(int i = m * m - 1; i ≥ 1; -- i) if(!G[i].second)
30     G[i] = G[i + 1];
31 int lastans = 0;
32 for(int i = 1; i ≤ n; ++ i){
33     int a, inv;
34     a = (read(seed) ^ lastans) % (p - 1) + 1;
35     int w = 1ll * a * m * m / p;
36     auto &yy1 = F[w].second; // *avoid y1 in <cmath>
37     if(1ll * a * yy1 % p ≤ p / m){
38         inv = 1ll * I[1ll * a * yy1 % p] * yy1 % p;
39     } else {
40         auto &yy2 = G[w].second;
41         inv = 1ll * I[1ll * a * (p - yy2) % p] * (p - yy2) % p;
42     }
43     lastans = inv;
44 }
45 cout << lastans << "\n";
46 return 0;
47 }

```

## 6.12 拉格朗日插值

### 6.12.1 定理

给定  $n$  个横坐标不同的点  $(x_i, y_i)$ , 可以唯一确定一个  $n-1$  阶多项式如下:

$$f(x) = \sum_{i=1}^n \frac{\prod_{j \neq i} (x - x_j)}{\prod_{j \neq i} (x_i - x_j)} \cdot y_i$$

## 6.13 min-max 容斥

### 6.13.1 定理

$$\max_{i \in S} \{x_i\} = \sum_{T \subseteq S} (-1)^{|T|-1} \min_{j \in T} \{x_j\}$$

$$\min_{i \in S} \{x_i\} = \sum_{T \subseteq S} (-1)^{|T|-1} \max_{j \in T} \{x_j\}$$

期望意义下上式依然成立。

另外设  $\max^k$  表示第  $k$  大的元素, 可以推广为如下式

子:

$$\max_{i \in S}^k \{x_i\} = \sum_{T \subseteq S} (-1)^{|T|-k} \binom{|T|-1}{k-1} \min_{j \in T} \{x_j\}$$

此外在数论上可以得到:

$$\text{lcm}_{i \in S} \{x_i\} = \prod_{T \subseteq S} \left( \gcd_{j \in T} \{x_j\} \right)^{(-1)^{|T|-1}}$$

### 6.13.2 应用

对于计算 “ $n$  个属性都出现的期望时间” 问题, 设第  $i$  个属性第一次出现的时间是  $t_i$ , 所求即为  $\max(t_i)$ , 使用 min-max 容斥转为计算  $\min(t_i)$ 。

比如  $n$  个独立物品, 每次抽中物品  $i$  的概率是  $p_i$ , 问期望抽多少次抽中所有物品。那么就可以计算  $\min_S$  表示第一次抽中物品集合  $S$  内物品的时间, 可以得到:

$$\max_U = \sum_{S \subseteq U} (-1)^{|S|-1} \min_S = \sum_{S \subseteq U} (-1)^{|S|-1} \cdot \frac{1}{\sum_{x \in S} p_x}$$

## 6.14 Barrett 取模

### 6.14.1 用法

调用 init 计算出  $S$  和  $X$ , 得到计算  $\lfloor x/P \rfloor = (x \times X)/2^{60+S}$ 。从而计算  $x \bmod P = x - P \times \lfloor x/P \rfloor$ 。

```

1 #include "../header.cpp"
2 i64 S = 0, X = 0;
3 void init(int MOD){
4     while((1 << (S + 1)) < MOD) S ++;
5     X = ((__int128)1 << 60 + S) / MOD + !(((__int128)1 << 60 + S) % MOD);
6     cerr << S << " " << X << endl;
7 }
8 int power(i64 x, int y, int MOD){
9     i64 r = 1;
10    while(y){
11        if(y & 1){
12            r = r * x;
13            r = r - MOD * ((__int128)r * X >> 60 + S);
14        }
15        x = x * x;
16        x = x - MOD * ((__int128)x * X >> 60 + S);
17        y >>= 1;
18    }
19    return r;
20 }

```

## 6.15 Pollard's Rho

### 6.15.1 用法

- 调用 test(n) 判断  $n$  是否是质数;
- 调用 rho(n) 计算  $n$  分解质因数后的结果, 不保证结果有序。

```

1 #include "../header.cpp"
2 i64 step(i64 a, i64 c, i64 m){
3     return ((__int128)a * a + c) % m;
4 }
5 i64 multi(i64 a, i64 b, i64 m){
6     return ((__int128) a * b % m;
7 }
8 i64 power(i64 a, i64 b, i64 m){
9     i64 r = 1;
10    while(b){
11        if(b & 1) r = multi(r, a, m);
12        b >>= 1, a = multi(a, a, m);
13    }

```

```

14 return r;
15 }
16 mt19937_64 MT;
17 bool test(i64 n){
18     if(n < 3 || n % 2 == 0) return n == 2;
19     i64 u = n - 1, t = 0;
20     while(u % 2 == 0) u /= 2, t += 1;
21     int test_time = 20;
22     for(int i = 1; i ≤ test_time; ++ i){
23         i64 a = MT() % (n - 2) + 2;
24         i64 v = power(a, u, n);
25         if(v == 1) continue;
26         int s;
27         for(s = 0; s < t; ++ s){
28             if(v == n - 1) break;
29             v = multi(v, v, n);
30         }
31         if(s == t) return false;
32     }
33     return true;
34 }
35 basic_string<i64> rho(i64 n){
36     if(n == 1) return { };
37     if(test(n)) return {n};
38     i64 a = MT() % (n - 1) + 1;
39     i64 x1 = MT() % (n - 1), x2 = x1;
40     for(int i = 1; i <= 1){
41         i64 tot = 1;
42         for(int j = 1; j ≤ i; ++ j){
43             x2 = step(x2, a, n);
44             tot = multi(tot, labs(x1 - x2), n);
45             if(j % 127 == 0){
46                 i64 d = __gcd(tot, n);
47                 if(d > 1)
48                     return rho(d) + rho(n / d);
49             }
50         }
51         i64 d = __gcd(tot, n);
52         if(d > 1)
53             return rho(d) + rho(n / d);
54         x1 = x2;
55     }
56 }

```

## 6.16 polya 定理

### 6.16.1 Burnside 引理

记所有染色方案的集合为  $X$ , 其中单个染色方案为  $x_0$ . 一种对称操作  $g \in X$  作用于染色方案  $x \in X$  上可以得到另外一种染色  $x'$ .

将所有对称操作作为集合  $G$ , 那么  $Gx = \{gx \mid g \in G\}$  是与  $x$  本质相同的染色方案的集合, 形式化地称为  $x$  的轨道. 统计本质不同染色方案数, 就是统计不同轨道个数.

Burnside 引理说明如下:

$$|X/G| = \frac{1}{|G|} \sum_{g \in G} |X^g|$$

其中  $X^g$  表示在  $g \in G$  的作用下, 不动点的集合. 不动点被定义为  $x = gx$  的  $x$ .

### 6.16.2 Polya 定理

对于通常的染色问题,  $X$  可以看作一个长度为  $n$  的序列, 每个元素是 1 到  $m$  的整数. 可以将  $n$  看作面数、 $m$  看作颜色数. Polya 定理叙述如下:

$$|X/G| = \frac{1}{|G|} \sum_{g \in G} \sum_{m \in G} m^{c(g)}$$

其中  $c(g)$  表示对一个序列做轮换操作  $g$  可以分解成多少个置换环。

然而, 增加了限制 (比如要求某种颜色必须要多少个), 就无法直接应用 Polya 定理, 需要利用 Burnside 引理进行具体问题具体分析。

### 6.16.3 应用

给定  $n$  个点  $n$  条边的环, 现在有  $n$  种颜色, 给每个顶点染色, 询问有多少种本质不同的染色方案。

显然  $X$  是全体元素在 1 到  $n$  之间长度为  $n$  的序列,  $G$  是所有可能的单次旋转方案, 共有  $n$  种, 第  $i$  种方案会把 1 置换到  $i$ . 于是:

$$\begin{aligned}
 \text{ans} &= \frac{1}{|G|} \sum_{i=1}^n m^{c(g_i)} \\
 &= \frac{1}{n} \sum_{i=1}^n n^{\text{gcd}(i, n)} \\
 &= \frac{1}{n} \sum_{d|n} n^d \sum_{i=1}^n [\text{gcd}(i, n) = d] \\
 &= \frac{1}{n} \sum_{d|n} n^d \varphi(n/d)
 \end{aligned}$$

```

1 #include "../header.cpp"
2 vector <tuple<int, int> > P;
3 void solve(int step, int n, int d, int f, int
    &ans){
4     if(step == P.size()){
5         ans = (ans + 1ll * power(n, n / d) * f) %
            MOD;
6     } else {
7         auto [w, c] = P[step];
8         int dd = 1, ff = 1;
9         for(int i = 0; i ≤ c; ++ i){
10             solve(step + 1, n, d * dd, f * ff, ans);
11             ff = ff * (w - (i == 0));
12             dd = dd * w;
13         }
14     }
15 }
16 int main(){
17     int T; cin >> T;
18     while(T --){
19         int n, t;
20         cin >> n;
21         t = n;
22         for(int i = 2; i * i ≤ n; ++ i) if(n % i ==
            0){
23             int w = i, c = 0;
24             while(t % i == 0){
25                 t /= i, c ++;
26             }
27             P.push_back({ w, c });
28         }
29         if(t != 1){
30             P.push_back({ t, 1 });
31         }
32         int ans = 0;
33         solve(0, n, 1, 1, ans);
34         ans = 1ll * ans * power(n, MOD - 2) % MOD;
35         cout << ans << endl;
36         P.clear();
37     }
38     return 0;
39 }

```

## 6.17 min25 筛

设有一个积性函数  $f(n)$ , 满足  $f(p^k)$  可以快速求, 考虑搞一个在质数位置和  $f(n)$  相等的  $g(n)$ , 满足它有完全积性, 并且单点和前缀和都可以快速求, 然后通过第一部分筛出  $g$  在质数位置的前缀和, 从而相当于得到  $f$  在质数

位置的前缀和, 然后利用它, 做第二部分, 求出  $f$  的前缀和。

第一部分:  $G_k(n) = \sum_{i=1}^n [\text{mindiv}(i) > p_k \text{ or isprime}(i)]g(i)$  ( $p_0 = 1$ ), 则有  $G_k(n) = G_{k-1}(n) - g(p_k)(G_{k-1}(n/p_k) - G_{k-1}(p_{k-1}))$ , 复杂度  $O(n^{3/4}/\log n)$ 。

第二部分:  $F_k(n) = \sum_{i=1}^n [\text{mindiv}(i) \geq p_k]f(i)$ ,  $F_k(n) = \sum_{\substack{h \geq k \\ p_h^2 \leq n}} \sum_{\substack{c \geq 1 \\ p_h^{c+1} \leq n}} (f(p_h^c)F_{h+1}(n/p_h^c) + f(p_h^{c+1})) + F_{\text{prime}}(n) - F_{\text{prime}}(p_{k-1})$ , 在  $n \leq 10^{13}$  可以证明复杂度  $O(n^{3/4}/\log n)$ 。

常见细节问题:

- 由于  $n$  通常是  $10^{10}$  到  $10^{11}$  的数, 导致  $n$  会爆 int,  $n^2$  会爆 long long, 而且往往会用自然数幂和, 更容易爆, 所以要小心。
- 记  $s = \lfloor \sqrt{n} \rfloor$ , 由于  $F$  递归时会去找  $F_{h+1}$ , 会访问到  $s$  以内最大的质数往后的一个质数, 而已经证明对于所有  $n \in \mathbb{N}^+$ ,  $[n+1, 2n]$  中有至少一个质数, 所以只需要筛到  $2s$  即可。
- 注意补回  $f(1)$ 。

```

22         v = n / l, r = n / v;
23         if (v ≤ sqrt_n) id1[v] = ++cnt;
24         else id2[init_n / v] = ++cnt;
25         val.emplace_back(v);
26     }
27 }
28 ll id(ll n) {
29     if (n ≤ sqrt_n) return id1[n];
30     else return id2[init_n / n];
31 }
32 }
33 using namespace init;
34 // 计算 $G_k$, 两个参数分别是 $g$ 从 $2$ 开始
   的前缀和和 $g$
35 auto calcG = [&] (auto&& sum, auto&& g) →
   vector<ll> {
36     vector<ll> G(cnt + 1);
37     for (int i = 1; i ≤ cnt; ++i) G[i] = sum(
   val[i]);
38     ll pre = 0;
39     for (int i = 1; p[i] * p[i] ≤ n; ++i) {
40         for (int j = 1; j ≤ cnt; ++j) {
41             if (p[i] * p[j] > val[j]) break;
42             ll tmp = id(val[j] / p[i]);
43             G[j] = (G[j] - g(p[i]) * (G[tmp] -
   pre)) % MD;
44         }
45         pre = (pre + g(p[i])) % MD;
46     }
47     for (int i = 1; i ≤ cnt; ++i) G[i] = (G[i]
   % MD + MD) % MD;
48     return G;
49 };
50 // 计算 $F_k$, 直接搜, 不用记忆化。`fp` 是 $F_{\text{prime}}$, `pc` 是 $p^c$, 其中 `f(p[h]^c)` 要替换掉。
51 function<ll(ll, int)> calcF = [&] (ll m, int k) {
52     if (p[k] > m) return 0;
53     ll ans = (fp[id(m)] - fp[id(p[k - 1])]) %
   MD;
54     for (int h = k; p[h] * p[h] ≤ m; ++h) {
55         ll pc = p[h], c = 1;
56         while (pc * p[h] ≤ m) {
57             ans = (ans + calcF(m / pc, h + 1)
   * f(p[h]^c)) % MD;
58             ++c, pc = pc * p[h], ans = (ans +
   f(p[h]^c)) % MD;
59         }
60     }
61     return ans;
62 };

```

## 6.18 杜教筛

### 6.18.1 用法

对于积性函数  $f$ , 找到易求前缀和的积性函数  $g, h$  使得  $h = f * g$ , 根据递推式计算  $S(n) = \sum_{i=1}^n f(i)$ :

$$S(n) = H(n) - \sum_{d=1}^n g(d) \times S\left(\left\lfloor \frac{n}{d} \right\rfloor\right)$$

### 6.18.2 例题

- 对于  $f = \varphi$ , 寻找  $g = 1, h = \text{id}$ ;
- 对于  $f = \mu$ , 寻找  $g = 1, h = \varepsilon$ 。

```

1 #include "../header.cpp"
2 const int H = 1e7;
3 int P[MAXN], p; bool V[MAXN];
4 i64 ph[MAXN], sph[MAXN];
5 i64 mu[MAXN], smu[MAXN];
6 i64 tp[MAXN];
7 i64 solve_ph(i64 N){
8     for(int d = N / H; d ≥ 1; -- d){
9         i64 n = N / d;
10        i64 wh = 1ll * n * (n + 1) / 2;
11        tp[d] = wh;
12        for(i64 l = 2, r; l ≤ n; l = r + 1){
13            r = n / (n / l);
14            i64 wg = r - l + 1;
15            i64 ws = n / l ≤ H ? sph[n / l] : tp[N
   / (n / l)];
16            tp[d] -= wg * ws;
17        }
18    }
19    return N ≤ H ? sph[N] : tp[1];
20 }
21 i64 solve_mu(i64 N){
22     for(int d = N / H; d ≥ 1; -- d){
23         i64 n = N / d;
24         i64 wh = 1;
25         tp[d] = wh;
26         for(i64 l = 2, r; l ≤ n; l = r + 1){
27             r = n / (n / l);
28             i64 wg = r - l + 1;
29             i64 ws = n / l ≤ H ? smu[n / l] : tp[N
   / (n / l)];
30             tp[d] -= wg * ws;
31         }
32     }
33     return N ≤ H ? smu[N] : tp[1];
34 }
35 int main(){

```

```

1 // 预处理, $1$ 所在的块也算进去了
2 namespace init {
3     ll init_n, sqrt_n;
4     vector<ll> np, p, id1, id2, val;
5     ll cnt;
6     void main(ll n) {
7         init_n = n, sqrt_n = sqrt(n);
8         ll M = sqrt_n * 2; // 筛出一个 > floor
   (sqrt(n)) 的质数, 避免后续讨论边界
9         np.resize(M + 1), p.resize(M + 1);
10        for (ll i = 2; i ≤ M; ++i) {
11            if (!np[i]) p[++p[0]] = i;
12            for (ll j = 1; j ≤ p[0]; ++j) {
13                if (i * p[j] > M) break;
14                np[i * p[j]] = 1;
15                if (i % p[j] == 0) break;
16            }
17        }
18        p[0] = 1;
19        id1.resize(sqrt_n + 1), id2.resize(
   sqrt_n + 1);
20        val.resize(1);
21        for (ll l = 1, r, v; l ≤ n; l = r +
   1) {

```

```

36 ios :: sync_with_stdio(false);
37 cin.tie(nullptr);
38 ph[1] = 1;
39 mu[1] = 1;
40 for(int i = 2; i ≤ H; ++ i){
41     if(!V[i]){
42         P[++ p] = i;
43         ph[i] = i - 1;
44         mu[i] = -1;
45     }
46     for(int j = 1; j ≤ p && P[j] ≤ H / i; ++ j)
47     {
48         int &p = P[j];
49         V[i * p] = true;
50         if(i % p == 0){
51             ph[i * p] = ph[i] * p;
52             mu[i * p] = 0;
53             break;
54         } else {
55             ph[i * p] = ph[i] * (p - 1);
56             mu[i * p] = -mu[i];
57         }
58     }
59     for(int i = 1; i ≤ H; ++ i){
60         sph[i] = sph[i - 1] + ph[i];
61         smu[i] = smu[i - 1] + mu[i];
62     }
63     int T; cin >> T;
64     while(T --> 0){
65         int n; cin >> n;
66         cout << solve_ph(n) << " " << solve_mu(n)
67         << "\n";
68     }
69     return 0;

```

## 6.19 PN 筛

### 6.19.1 用法

对于积性函数  $f(x)$ , 寻找积性函数  $g(x)$  使得  $g(p) = f(p)$ , 且  $g$  易求前缀和  $G$ .

令  $h = f * g^{-1}$ , 可以证明只有 PN 处  $h$  的函数值非 0, PN 指每个素因子幂次都不小于 2 的数。同时可以证明  $n$  以内的 PN 只有  $\mathcal{O}(\sqrt{n})$  个, 且可以暴力枚举质因子幂次得到所有 PN。

可利用下面公式计算  $h(p^c)$ :

$$h(p^c) = f(p^c) - \sum_{i=1}^c g(p^i) \times h(p^{c-i})$$

### 6.19.2 例题

定义积性函数  $f(x)$  满足  $f(p^k) = p^k(p^k - 1)$ , 计算  $\sum f(i)$ 。

取  $g(p) = \text{id}(p)\varphi(p) = f(p)$ , 根据  $g * \text{id} = \text{id}_2$  利用杜教筛求解。  $h(p^c)$  的值利用递推式进行计算。

```

1 #include "../header.cpp"
2 const int H = 1e7;
3 const int MOD = 1e9 + 7;
4 const int DIV2 = 500000004;
5 const int DIV6 = 166666666;
6 int P[MAXN], p; bool V[MAXN];
7 int g[MAXN], le[MAXN], ge[MAXN];
8 int s1(i64 n){ // 1^1 + 2^1 + ... + n^1
9     n %= MOD;
10    return 1ll * n * (n + 1) % MOD * DIV2 % MOD;
11 }
12 int s2(i64 n){ // 1^2 + 2^2 + ... + n^2
13     n %= MOD;
14    return 1ll * n * (n + 1) % MOD * (2 * n + 1)
15    % MOD * DIV6 % MOD;
16 }
17 int sg(i64 n, i64 N){
18     return n ≤ H ? le[n] : ge[N / n];
19 }
20 int sieve_du(i64 N){
21     for(int d = N / H; d ≥ 1; -- d){
22         i64 n = N / d;
23         int wh = s2(n);
24         for(i64 l = 2, r; l ≤ n; l = r + 1){
25             r = n / (n / l);
26             int wg = (s1(r) - s1(l - 1) + MOD) % MOD;
27             int ws = sg(n / l, N);
28             ge[d] = (ge[d] + 1ll * wg * ws) % MOD;
29         }
30         ge[d] = (wh - ge[d] + MOD) % MOD;
31     }
32     return N ≤ H ? le[N] : ge[1];
33 }
34 vector<int> hc[MAXM], gc[MAXM];
35 int ANS;
36 void sieve_pn(int last, i64 x, int h, i64 N){
37     ANS = (ANS + 1ll * h * sg(N / x, N)) % MOD;

```

```

37 for(i64 i = last + 1; x ≤ N / P[i] / P[i]; ++ i){
38     int c = 2;
39     for(i64 t = x * P[i] * P[i]; t ≤ N; t *= P[i], c++){
40         int hh = 1ll * h * hc[i][c] % MOD;
41         sieve_pn(i, t, hh, N);
42     }
43 }
44 }
45 int main(){
46     ios :: sync_with_stdio(false);
47     cin.tie(nullptr);
48     g[1] = 1;
49     for(int i = 2; i ≤ H; ++ i){
50         if(!V[i]){
51             P[++ p] = i, g[i] = 1ll * i * (i - 1) % MOD;
52         }
53         for(int j = 1; j ≤ p && P[j] ≤ H / i; ++ j)
54         {
55             int &p = P[j];
56             V[i * p] = true;
57             if(i % p == 0){
58                 g[i * p] = 1ll * g[i] * p % MOD * p % MOD;
59                 break;
60             } else {
61                 g[i * p] = 1ll * g[i] * p % MOD * (p - 1) % MOD;
62             }
63         }
64     }
65     for(int i = 1; i ≤ H; ++ i){
66         le[i] = (le[i - 1] + g[i]) % MOD;
67     }
68     i64 N;
69     cin >> N;
70     for(int i = 1; i ≤ p && 1ll * P[i] * P[i] ≤ N; i++){
71         int &p = P[i];
72         hc[i].push_back(1);
73         gc[i].push_back(1);
74         for(i64 c = 1, t = p; t ≤ N; t = t * p, ++ c){
75             if(c == 1){
76                 gc[i].push_back(1ll * p * (p - 1) % MOD);
77             } else {
78                 gc[i].push_back(1ll * gc[i].back() * p % MOD * p % MOD);

```

```

1 #include "../header.cpp"
2 int inv(int x);
3 const int MAX_ = (1 << 19) + 3;
4 using cplx = complex<double>;
5 const long double pi = acos(-1);
6 namespace Poly{
7     void FFT(int n, cplx Z[]){
8         static int W[MAX_];
9         int l = 1; W[0] = 0;
10        while (n >= 1)
11            up(0, l - 1, i)
12            W[l++] = W[i] << 1 | 1, W[i] <= 1;
13        up(0, l - 1, i)
14        if(W[i] > i) swap(Z[i], Z[W[i]]);
15        for (n = l >> 1, l = 1; n >= 1, l <= 1)
16        {
17            cplx* S = Z, o(cos(pi / l), sin(pi / l))
18            ;
19            up(0, n - 1, i){
20                cplx s(1, 0);
21                up(0, l - 1, j){
22                    cplx x = S[j] + s * S[j + l];

```



```

21     cplx y = S[j] - s * S[j + l];
22     S[j] = x, S[j + l] = y, s = s * o;
23 }
24 S += l << 1;
25 }
26 }
27 }
28 void IFFT(int n, cplx Z[]){
29     FFT(n, Z); reverse(Z + 1, Z + n);
30     up(0, n - 1, i) Z[i] /= n;
31 }
32 void NTT(int n, int Z[]){
33     static int W[MAX_];
34     int g = 3, l = 1; W[0] = 0;
35     while (n >= 1)
36         up(0, l - 1, i)
37             W[l++] = W[i] << 1 | 1, W[i] <= 1;
38     up(0, l - 1, i)
39         if (W[i] > i) swap(Z[i], Z[W[i]]);
40     for (n = l >> 1, l = 1; n >= 1, l <= 1)
41     {
42         int* S = Z, o = power(g, (MOD - 1) / l / 2);
43         up(0, n - 1, i){
44             int s = 1;
45             up(0, l - 1, j){
46                 int x = (S[j] + 1ll * s * S[j + l] % MOD) % MOD;
47                 int y = (S[j] - 1ll * s * S[j + l] % MOD + MOD) % MOD;
48                 S[j] = x, S[j + l] = y;
49                 s = 1ll * s * o % MOD;
50             }
51             S += l << 1;
52         }
53     }
54 void INTT(int n, int Z[]){
55     NTT(n, Z); reverse(Z + 1, Z + n);
56     int o = inv(n);
57     up(0, n - 1, i)
58         Z[i] = 1ll * Z[i] * o % MOD;
59 }
60 void MUL(int n, int A[], int B[]){ // 乘法
61     NTT(n, A), NTT(n, B);
62     up(0, n - 1, i)
63         A[i] = 1ll * A[i] * B[i] % MOD;
64     INTT(n, A);
65 }
66 void INV(int n, int Z[], int T[]){ // 乘法逆

```

```

67     static int A[MAX_];
68     up(0, n - 1, i)
69         T[i] = 0;
70     T[0] = power(Z[0], MOD - 2);
71     for (int l = 1; l < n; l <= 1){
72         up(0, 2 * l - 1, i) A[i] = Z[i];
73         up(2 * l, 4 * l - 1, i) A[i] = 0;
74         NTT(4 * l, A), NTT(4 * l, T);
75         up(0, 4 * l - 1, i)
76             T[i] = (2ll * T[i] - 1ll * A[i] * T[i] % MOD * T[i] % MOD + MOD) % MOD;
77         INTT(4 * l, T);
78         up(2 * l, 4 * l - 1, i)
79             T[i] = 0;
80     }
81 }
82 void DIF(int n, int Z[], int T[]){ // 微分
83     up(0, n - 2, i)
84         T[i] = 1ll * Z[i + 1] * (i + 1) % MOD;
85     T[n - 1] = 0;
86 }
87 void INT(int n, int c, int Z[], int T[]){ // 积分
88     up(1, n - 1, i)
89         T[i] = 1ll * Z[i - 1] * inv(i) % MOD;
90     T[0] = c;
91 }
92 void LN(int n, int* Z, int* T){ // 求对数
93     static int A[MAX_], B[MAX_];
94     up(0, 2 * n - 1, i)
95         A[i] = B[i] = 0;
96     DIF(n, Z, A), INV(n, Z, B), MUL(2 * n, A, B), INT(n, 0, A, T);
97 }
98 void EXP(int n, int* Z, int* T){ // 求指数
99     static int A[MAX_], B[MAX_];
100     up(1, 2 * n - 1, i) T[i] = 0;
101     T[0] = 1;
102     for (int l = 1; l < n; l <= 1){
103         LN(2 * l, T, A);
104         up(0, 2 * l - 1, i)
105             B[i] = (-A[i] + Z[i] + MOD) % MOD;
106         B[0] = (B[0] + 1) % MOD;
107         up(2 * l, 4 * l - 1, i)
108             T[i] = B[i] = 0;
109         MUL(4 * l, T, B);
110     }
111 }
112 void SQT(int n, int* Z, int* T){ // 开

```

```

113     根
114     static int A[MAX_], B[MAX_];
115     up(1, 2 * n - 1, i) T[i] = 0;
116     T[0] = 1;
117     int o = inv(2);
118     for (int l = 1; l < n; l <= 1){
119         INV(2 * l, T, A);
120         up(0, 2 * l - 1, i)
121             B[i] = Z[i];
122         up(2 * l, 4 * l - 1, i)
123             A[i] = B[i] = 0;
124         MUL(4 * l, A, B);
125         up(0, 2 * l - 1, i)
126             T[i] = 1ll * (T[i] + A[i]) * o % MOD;
127     }
128 void SHF(int n, int c, int* Z, int* T){ // 平移
129     static int A[MAX_], B[MAX_], F[MAX_], G[
130         MAX_];
131     int o = 1;
132     up(1, n - 1, i)
133         F[i] = 1ll * F[i - 1] * i % MOD,
134         G[i] = 1ll * G[i - 1] * inv(i) % MOD;
135     up(0, n - 1, i)
136         A[i] = 1ll * Z[n - 1 - i] * F[n - 1 - i] % MOD;
137     up(0, n - 1, i){
138         B[i] = 1ll * G[i] * o % MOD;
139         o = 1ll * o * c % MOD;
140     }
141     int l = 1; while (l < 2 * n - 1) l <= 1;
142     up(n, l - 1, i)
143         A[i] = B[i] = 0;
144     MUL(l, A, B);
145     up(0, n - 1, i)
146         T[n - 1 - i] = 1ll * G[n - 1 - i] * A[i] % MOD;
147 }

```

## 7.2 FWT 全家桶

### 7.2.1 用法

沃尔什全家桶。

包含与卷积、或卷积、异或卷积，定义分别为二进制与、或、异或带下式：

$$b_k = \sum_{i \otimes j = k} a_i \times b_j$$

```

1 #include "../header.cpp"
2 namespace Solve1{ // or 卷积
3     void FWT(int n, int *A){
4         for(int l = 1 << n, u = 2, v = 1; u ≤ l; u
5             <=<= 1, v <=<= 1)
6             for(int j = 0; j < l; j += u)
7                 for(int k = 0; k < v; ++ k)
8                     A[j + v + k] = (A[j + v + k] + A[j +
9                         k]) % MOD;
10    }
11    void IFWT(int n, int *A){
12        for(int l = 1 << n, u = l, v = l / 2; u >
13            1; u >>= 1, v >>= 1)
14            for(int j = 0; j < l; j += u)
15                for(int k = 0; k < v; ++ k)
16                    A[j + v + k] = (A[j + v + k] - A[j +
17                        k] + MOD) % MOD;
18    }
19 }
20 namespace Solve2{ // and 卷积
21     void FWT(int n, int *A){
22         for(int l = 1 << n, u = 2, v = 1; u ≤ l; u
23             <=<= 1, v <=<= 1)
24             for(int j = 0; j < l; j += u)
25                 for(int k = 0; k < v; ++ k)
26                     A[j + k] = (A[j + k] + A[j + v + k])
27                         % MOD;
28    }
29    void IFWT(int n, int *A){
30        for(int l = 1 << n, u = l, v = l / 2; u >
31            1; u >>= 1, v >>= 1)
32            for(int j = 0; j < l; j += u)
33                for(int k = 0; k < v; ++ k){
34                    int a = A[j + k];
35                    int b = A[j + v + k];
36                    A[j + k] = (a + b + MOD) % MOD;
37                    A[j + v + k] = (a - b + MOD) % MOD;
38                }
39    }
40 }
41 namespace Solve3{ // xor 卷积
42     void FWT(int n, int *A){
43         for(int l = 1 << n, u = 2, v = 1; u ≤ l; u
44             <=<= 1, v <=<= 1)
45             for(int j = 0; j < l; j += u)
46                 for(int k = 0; k < v; ++ k){
47                     int a = A[j + k];
48                     int b = A[j + v + k];
49                     A[j + k] = (a + b + MOD) % MOD;
50                     A[j + v + k] = (a - b + MOD) % MOD;
51                 }
52    }
53    void IFWT(int n, int *A){
54        int div2 = (MOD + 1) / 2;
55        for(int l = 1 << n, u = l, v = l / 2; u >
56            1; u >>= 1, v >>= 1)
57            for(int j = 0; j < l; j += u)
58                for(int k = 0; k < v; ++ k)
59                    A[j + v + k] = (A[j + v + k] + A[j +
60                        k] * div2 % MOD + MOD) % MOD;
61    }
62 }

```

```

43     for(int l = 1 << n, u = l, v = l / 2; u >
44         1; u >>= 1, v >>= 1)
45         for(int j = 0; j < l; j += u)
46             for(int k = 0; k < v; ++ k){
47                 int a = A[j + k];
48                 int b = A[j + v + k];
49                 A[j + k] = 1ll * (a + b + MOD) *
50                     div2 % MOD;
51                 A[j + v + k] = 1ll * (a - b + MOD) *
52                     div2 % MOD;
53             }
54    }
55 }

```

### 7.3 任意模数 NTT

```

1 #include "poly-family.cpp"
2 const int BLOCK = 32768;
3 using cplx = complex<double>;
4 cplx A1[MAXN], A2[MAXN], B1[MAXN], B2[MAXN];
5 int n, m, L, mod;
6 cplx P[MAXN], Q[MAXN];
7 void FFTFFT(int L, cplx X[], cplx Y[]){
8     for(int i = 0; i < L; ++ i){
9         P[i] = { X[i].real(), Y[i].imag() };
10    }
11    Poly :: FFT(L, P);
12    for(int i = 0; i < L; ++ i){
13        Q[i] = (i == 0 ? P[0] : P[L - i]);
14        Q[i].imag(-Q[i].imag());
15    }
16    for(int i = 0; i < L; ++ i){
17        X[i] = (P[i] + Q[i]);
18        Y[i] = (Q[i] - P[i]) * cplx(0, 1);
19        X[i] /= 2, Y[i] /= 2;
20    }
21 }
22 int main(){
23     ios :: sync_with_stdio(false);
24     cin.tie(nullptr);
25     cin >> n >> m >> mod;
26     for(int i = 0; i ≤ n; ++ i){
27         int a; cin >> a; a %= mod;
28         A1[i].real(a / BLOCK);
29         A2[i].imag(a % BLOCK);
30     }
31     for(int i = 0; i ≤ m; ++ i){
32         int a; cin >> a; a %= mod;
33         B1[i].real(a / BLOCK);
34         B2[i].imag(a % BLOCK);
35     }
36     for(L = 1; L ≤ n + m; L <=<= 1);

```

```

37     FFTFFT(L, A1, A2), FFTFFT(L, B1, B2);
38     for(int i = 0; i < L; ++ i){
39         P[i] = A1[i] * B1[i] + cplx(0, 1) * A2[i]
40             * B1[i];
41         Q[i] = A1[i] * B2[i] + cplx(0, 1) * A2[i]
42             * B2[i];
43     }
44     Poly :: IFFT(L, P);
45     Poly :: IFFT(L, Q);
46     for(int i = 0; i < L; ++ i){
47         long long a1b1 = P[i].real() + 0.5;
48         long long a2b1 = P[i].imag() + 0.5;
49         long long a1b2 = Q[i].real() + 0.5;
50         long long a2b2 = Q[i].imag() + 0.5;
51         long long w = ((a1b1 % mod * (BLOCK *
52             BLOCK % mod)) + ((a2b1 + a1b2) % mod) *
53             BLOCK + a2b2) % mod;
54         if(i ≤ n + m) cout << w << " ";
55     }
56     return 0;
57 }

```

## 8 字符串

### 8.1 AC 自动机

```

1 #include "../header.cpp"
2 namespace ACAM{
3     int C[MAXN][MAXM], F[MAXN], o;
4     void insert(char *S){
5         int p = 0, len = 0;
6         for(int i = 0; S[i]; ++ i){
7             int e = S[i] - 'a';
8             if(C[p][e]) p = C[p][e];
9             else p = C[p][e] = ++ o;
10            ++ len;
11        }
12    }
13    void build(){
14        queue<int> Q; Q.push(0);
15        while(!Q.empty()){
16            int u = Q.front(); Q.pop();
17            for(int i = 0; i < 26; ++ i){
18                int v = C[u][i];
19                if(v == 0) continue;
20                int p = F[u];
21                while(!C[p][i] && p ≠ 0) p = F[p];
22                if(C[p][i] && C[p][i] ≠ v)
23                    F[v] = C[p][i];
24                Q.push(v);
25            }
26        }
27    }

```

```

26 }
27 }
28 }

```

## 8.2 扩展 KMP

### 8.2.1 定义

$$z_i^{(1)} = |\text{lcp}(b, \text{suffix}(b, i))|$$

$$z_i^{(2)} = |\text{lcp}(b, \text{suffix}(a, i))|$$

```

1 #include "../header.cpp"
2 char A[MAXN], B[MAXN * 2];
3 int n, m, l, r, Z[MAXN * 2];
4 i64 ans1, ans2;
5 int main(){
6     scanf("%s%s", A + 1, B + 1);
7     n = strlen(A + 1);
8     m = strlen(B + 1);
9     l = 0, r = 0; Z[1] = 0, ans1 = m + 1;
10    for(int i = 2; i ≤ m; ++ i){
11        if(i ≤ r) Z[i] = min(r - i + 1, Z[i - l + 1]);
12        else Z[i] = 0;
13        while(B[Z[i] + 1] == B[i + Z[i]])
14            ++ Z[i];
15        if(i + Z[i] - 1 > r)
16            r = i + Z[i] - 1, l = i;
17        ans1 ^= 1ll * i * (Z[i] + 1);
18    }
19    l = 0, r = 0;
20    Z[1] = 0, B[m + 1] = '#', strcat(B + 1, A + 1);
21    for(int i = 2; i ≤ n + m + 1; ++ i){
22        if(i ≤ r) Z[i] = min(r - i + 1, Z[i - l + 1]);
23        else Z[i] = 0;
24        while(B[Z[i] + 1] == B[i + Z[i]])
25            ++ Z[i];
26        if(i + Z[i] - 1 > r)
27            r = i + Z[i] - 1, l = i;
28    }
29    for(int i = m + 2; i ≤ n + m + 1; ++ i){
30        ans2 ^= 1ll * (i - m - 1) * (Z[i] + 1);
31    }
32    printf("%lld\n%lld\n", ans1, ans2);
33    return 0;
34 }

```

## 8.3 Manacher

```

1 #include "../header.cpp"
2 const int MAXN = 2.2e7 + 11;
3 char S[MAXN], T[MAXN]; int n, R[MAXN];
4 int main(){
5     scanf("%s", S + 1);
6     n = strlen(S + 1);
7     for(int i = 1; i ≤ n; ++ i){
8         T[2 * i - 1] = S[i], T[2 * i] = '#';
9     }
10    T[0] = '#', n = 2 * n;
11    int p = 0, x = 0, ans = 0;
12    for(int i = 1; i ≤ n; ++ i){
13        if(i ≤ p) R[i] = min(R[2 * x - i], p - i);
14        while(i - R[i] - 1 ≥ 0 && T[i + R[i] + 1] == T[i - R[i] - 1])
15            ++ R[i];
16        if(i + R[i] > p){
17            p = i + R[i];
18            x = i;
19        }
20        ans = max(ans, R[i]);
21    }
22    printf("%d\n", ans);
23    return 0;
24 }

```

## 8.4 回文自动机

```

1 #include "../header.cpp"
2 namespace PAM{
3     const int SIZ = 5e5 + 3;
4     int n, s, F[SIZ], L[SIZ], D[SIZ];
5     int M[SIZ][MAXM];
6     char S[SIZ];
7     void init(){
8         S[0] = '$', n = 1;
9         F[s = 0] = -1, L[0] = -1, D[0] = 0;
10        F[s = 1] = 0, L[1] = 0, D[1] = 0;
11    }
12    void extend(int &last, char c){
13        S[++ n] = c;
14        int e = c - 'a', a = last;
15        while(c ≠ S[n - 1 - L[a]]) a = F[a];
16        if(M[a][e]){
17            last = M[a][e];
18        } else {
19            int cur = M[a][e] = ++ s;
20            L[cur] = L[a] + 2;
21            if(a == 0){

```

```

22        F[cur] = 1;
23        } else {
24            int b = F[a];
25            while(c ≠ S[n - 1 - L[b]])
26                b = F[b];
27            F[cur] = M[b][e];
28        }
29        D[cur] = D[F[cur]] + 1;
30        last = cur;
31    }
32 }
33 }

```

## 8.5 后缀平衡树

### 8.5.1 本代码尚未完成

## 8.6 后缀数组 (倍增)

```

1 #include "../header.cpp"
2 int n, m, A[MAXN], B[MAXN];
3 int C[MAXN], R[MAXN], P[MAXN], Q[MAXN];
4 char S[MAXN];
5 int main(){
6     scanf("%s", S), n = strlen(S), m = 256;
7     for(int i = 0; i < n; ++ i) R[i] = S[i];
8     for (int k = 1; k ≤ n; k <= 1){
9         for(int i = 0; i < n; ++ i){
10            Q[i] = ((i + k > n - 1) ? 0 : R[i + k]);
11            P[i] = R[i];
12            m = max(m, R[i]);
13        }
14        #define fun(a, b, c) \
15            memset(C, 0, sizeof(int) * (m + 1));
16        for(int i = 0; i < n; ++ i) C[a] += 1;
17        for(int i = 1; i ≤ m; ++ i) C[i] += C[i - 1];
18        for(int i = n - 1; i ≥ 0; -- i) c[-- C[a]] = b;
19        fun(Q[i], i, B)
20        fun(P[B[i]], B[i], A)
21        #undef fun
22        int p = 1; R[A[0]] = 1;
23        for(int i = 1; i ≤ n - 1; ++ i){
24            bool f1 = P[A[i]] == P[A[i - 1]];
25            bool f2 = Q[A[i]] == Q[A[i - 1]];
26            R[A[i]] = f1 && f2 ? R[A[i - 1]] : ++ p;
27        }
28        if (m == n) break;
29    }

```

```

30 for(int i = 0; i < n; ++ i)
31     printf("%u ", A[i] + 1);
32 return 0;
33 }

```

## 8.7 后缀数组 (SAIS)

```

1 #include "../header.cpp"
2 #define LTYPE 0
3 #define STYPE 1
4 void induce_sort(int n, int S[], int T[], int
5 m, int LM[], int SA[], int C[]){
6     vector<int> BL(n), BS(n), BM(n);
7     fill(SA, SA + n, -1);
8     for(int i = 0; i < n; ++ i){ // 预处理
9         BM[i] = BS[i] = C[i] - 1;
10        BL[i] = i == 0 ? 0 : C[i - 1];
11    }
12    for(int i = m - 1; i ≥ 0; -- i) // 放置
13        LMS 后缀
14        SA[BM[S[LM[i]]] --] = LM[i];
15    for(int i = 0, p; i < n; ++ i) // 计算 L
16        类型后缀的位置
17        if(SA[i] > 0 && T[p = SA[i] - 1] == LTYPE)
18            SA[BL[S[p]] ++] = p;
19    for(int i = n - 1, p; i ≥ 0; -- i) // 计算 S
20        类型后缀的位置
21        if(SA[i] > 0 && T[p = SA[i] - 1] == STYPE)
22            SA[BS[S[p]] --] = p;
23    // 长度 n, 字符集 [0, n), 要求最后一个元素为 0
24    // 例如输入 ababa 传入 n = 6, S = [1 2 1 2 1
25    // 0]
26    void sais(int n, int S[], int SA[]){
27        vector<int> T(n), C(n), I(n, -1);
28        T[n - 1] = STYPE;
29        for(int i = n - 2; i ≥ 0; -- i){ // 递推类
30            型
31            T[i] = S[i] == S[i + 1] ? T[i + 1] : (S[i]
32            < S[i + 1] ? STYPE : LTYPE);
33        }
34        for(int i = 0; i < n; ++ i) // 统计个数
35            C[S[i]] ++;
36        for(int i = 1; i < n; ++ i) // 前缀累加
37            C[i] += C[i - 1];
38        vector<int> P;
39        for(int i = 0; i < n; ++ i){ // 统计 LMS 后
40            缀
41            if(T[i] == STYPE && (i == 0 || T[i - 1] ==
42            LTYPE)){
43                I[i] = P.size(), P.push_back(i);

```

```

36 }
37 }
38 int m = P.size(), tot = 0, cnt = 0;
39 induce_sort(n, S, T.data(), m, P.data(), SA,
40 C.data());
41 vector<int> S0(m), SA0(m);
42 for(int i = 0, x, y = -1; i < n; ++ i){
43     if((x = I[SA[i]]) ≠ -1){
44         if(tot == 0 || P[x + 1] - P[x] ≠ P[y +
45         1] - P[y])
46             tot ++;
47         else for(int p1 = P[x], p2 = P[y]; p2 ≤
48             P[y + 1]; ++ p1, ++ p2){
49             if((S[p1] << 1 | T[p1]) ≠ (S[p2] << 1
50             | T[p2])){
51                 tot ++; break;
52             }
53         }
54         S0[y = x] = tot - 1;
55     }
56 }
57 if(tot == m){
58     for(int i = 0; i < m; ++ i)
59         SA0[S0[i]] = i;
60 } else {
61     sais(m, S0.data(), SA0.data());
62 }
63 for(int i = 0; i < m; ++ i)
64     S0[i] = P[SA0[i]];
65 induce_sort(n, S, T.data(), m, S0.data(), SA
66 , C.data());
67 }
68 int S[MAXN], SA[MAXN], H[MAXM], G[MAXM];
69 int main(){
70     int n = 0, t = 0, m = 256;
71     for(char c = cin.get(); isgraph(c); c = cin.
72     get()){
73         S[n ++] = c;
74         H[c] ++;
75     }
76     for(int i = 0; i < m; ++ i){
77         t += !!H[i], G[i] = t;
78     }
79     for(int i = 0; i < n; ++ i){
80         S[i] = G[S[i]];
81     }
82     sais(n + 1, S, SA);
83     for(int i = 1; i ≤ n; ++ i){
84         cout << SA[i] + 1 << " ";
85     }
86     return 0;
87 }

```

## 8.8 广义后缀自动机 (离线)

```

1 #include "../header.cpp"
2 namespace SAM{
3     const int SIZ = 2e6 + 3;
4     int M[SIZ][MAXM];
5     int L[SIZ], F[SIZ], S[SIZ];
6     int s = 0, h = 25;
7     void init(){
8         F[0] = -1, s = 0;
9     }
10    void extend(int &last, char c){
11        int e = c - 'a';
12        int cur = ++ s;
13        L[cur] = L[last] + 1;
14        int p = last;
15        while(p ≠ -1 && !M[p][e])
16            M[p][e] = cur, p = F[p];
17        if(p == -1){
18            F[cur] = 0;
19        } else {
20            int q = M[p][e];
21            if(L[p] + 1 == L[q]){
22                F[cur] = q;
23            } else {
24                int clone = ++ s;
25                L[clone] = L[p] + 1;
26                F[clone] = F[q];
27                for(int i = 0; i ≤ h; ++ i)
28                    M[clone][i] = M[q][i];
29                while(p ≠ -1 && M[p][e] == q)
30                    M[p][e] = clone, p = F[p];
31                F[cur] = F[q] = clone;
32            }
33        }
34        last = cur;
35    }
36    void solve(){
37        i64 ans = 0;
38        for(int i = 1; i ≤ s; ++ i)
39            ans += L[i] - L[F[i]];
40        cout << ans << endl;
41    }
42 }
43 namespace Trie{
44     const int SIZ = 1e6 + 3;
45     int M[SIZ][MAXM], s, h = 25;
46     void insert(char *S){
47         int p = 0;
48         for(int i = 0; S[i]; ++ i){
49             int e = S[i] - 'a';
50             if(M[p][e]){
51                 p = M[p][e];

```

```

52     } else
53     {
54         p = M[p][e] = ++ s;
55     }
56 int O[SIZ];
57 void build_sam(){
58     queue <int> Q;
59     Q.push(0);
60     while(!Q.empty()){
61         int u = Q.front(); Q.pop();
62         for(int i = 0; i ≤ h; ++ i){
63             char c = i + 'a';
64             if(M[u][i]){
65                 int v = M[u][i];
66                 O[v] = O[u];
67                 SAM :: extend(O[v], c);
68                 Q.push(v);
69             }
70         }
71     }
72 }
73 }

```

## 8.9 广义后缀自动机 (在线)

```

1 #include "../header.cpp"
2 namespace SAM{
3     const int SIZ = 2e6 + 3;
4     int M[SIZ][MAXM];
5     int L[SIZ], F[SIZ], S[SIZ];
6     int s = 0, h = 25;
7     void init(){
8         F[0] = -1, s = 0;
9     }
10    void extend(int &last, char c){
11        int e = c - 'a';
12        if(M[last][e]){
13            int p = last;
14            int q = M[last][e];
15            if(L[q] == L[last] + 1){
16                last = q;
17            } else {
18                int clone = ++ s;
19                L[clone] = L[p] + 1;
20                F[clone] = F[q];
21                for(int i = 0; i ≤ h; ++ i)
22                    M[clone][i] = M[q][i];
23                while(p ≠ -1 && M[p][e] == q)
24                    M[p][e] = clone, p = F[p];
25                F[q] = clone;
26                last = clone;
27            }

```

```

28    } else {
29        int cur = ++ s;
30        L[cur] = L[last] + 1;
31        int p = last;
32        while(p ≠ -1 && !M[p][e])
33            M[p][e] = cur, p = F[p];
34        if(p == -1){
35            F[cur] = 0;
36        } else {
37            int q = M[p][e];
38            if(L[p] + 1 == L[q]){
39                F[cur] = q;
40            } else {
41                int clone = ++ s;
42                L[clone] = L[p] + 1;
43                F[clone] = F[q];
44                for(int i = 0; i ≤ h; ++ i)
45                    M[clone][i] = M[q][i];
46                while(p ≠ -1 && M[p][e] == q)
47                    M[p][e] = clone, p = F[p];
48                F[cur] = F[q] = clone;
49            }
50        }
51        last = cur;
52    }
53 }
54 void solve(){
55     i64 ans = 0;
56     for(int i = 1; i ≤ s; ++ i)
57         ans += L[i] - L[F[i]];
58     cout << ans << endl;
59 }
60 }
61 // 每次插入新字符串前将 last 清零

```

## 8.10 后缀自动机

```

1 #include "../header.cpp"
2 namespace SAM{
3     const int SIZ = 2e6 + 3;
4     int M[SIZ][MAXM];
5     int L[SIZ], F[SIZ], S[SIZ];
6     int last = 0, s = 0, h = 25;
7     void init(){
8         F[0] = -1, last = s = 0;
9     }
10    void extend(char c){
11        int cur = ++ s, e = c - 'a';
12        L[cur] = L[last] + 1;
13        S[cur] = 1;
14        int p = last;
15        while(p ≠ -1 && !M[p][e])

```

```

16        M[p][e] = cur, p = F[p];
17    if(p == -1){
18        F[cur] = 0;
19    } else {
20        int q = M[p][e];
21        if(L[p] + 1 == L[q]){
22            F[cur] = q;
23        } else {
24            int clone = ++ s;
25            L[clone] = L[p] + 1;
26            F[clone] = F[q];
27            S[clone] = 0;
28            for(int i = 0; i ≤ h; ++ i)
29                M[clone][i] = M[q][i];
30            while(p ≠ -1 && M[p][e] == q)
31                M[p][e] = clone, p = F[p];
32            F[cur] = F[q] = clone;
33        }
34    }
35    last = cur;
36 }
37 vector <int> E[SIZ];
38 void build(){
39     for(int i = 1; i ≤ s; ++ i){
40         E[F[i]].push_back(i);
41     }
42 }
43 i64 ans = 0;
44 void dfs(int u){
45     for(auto &v : E[u]){
46         dfs(v), S[u] += S[v];
47     }
48     if(S[u] > 1)
49         ans = max(ans, 1ll * S[u] * L[u]);
50 }
51 }

```

# 9 计算几何

## 9.1 二维凸包

### 9.1.1 例题

给定  $n$  个点, 保证每三点不共线。要求找到一个简单多边形满足它不是凸包, 使得该多边形面积最大。

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 using i64 = long long;
4 const int MAXN = 2e5 + 3;
5 int X[MAXN], Y[MAXN];

```



```

6 struct Frac {
7     int a, b;
8     Frac (int _a, int _b){
9         if(_b < 0){
10             a = -_a, b = -_b;
11         } else {
12             a = _a, b = _b;
13         }
14     }
15 };
16 struct Node {
17     int x, y;
18 }P[MAXN];
19 bool operator < (const Frac A, const Frac B){
20     return 1ll * A.a * B.b - 1ll * A.b * B.a < 0;
21 }
22 bool operator < (const Node A, const Node B){
23     return A.x == B.x ? A.y > B.y : A.x < B.x;
24 }
25 const Frac intersect(Node A, Node B){
26     int a = B.y - A.y;
27     int b = A.x - B.x;
28     assert(b != 0);
29     if(b < 0){
30         a = -a, b = -b;
31     }
32     return Frac(a, b);
33 }
34 bool F[MAXN];
35 int main(){
36     int TT;
37     cin >> TT;
38     while(TT -- ){
39         int n;
40         cin >> n;
41         int maxx = -1e9, minx = 1e9;
42         for(int i = 1; i ≤ n; ++ i){
43             auto &[x, y] = P[i];
44             cin >> x >> y;
45             F[i] = false;
46         }
47         sort(P + 1, P + 1 + n);
48         vector<int> Q1, Q2, Q;
49         // Q1 计算上凸壳, Q2 计算下凸壳
50         for(int i = 1; i ≤ n; ++ i){
51             auto &[x, y] = P[i];
52             if(Q1.size() ≤ 1){
53                 Q1.push_back(i);
54             } else {
55                 while(Q1.size() ≥ 2){
56                     auto &[x1, y1] = P[Q1[Q1.

```

```

57                     size() - 2]];
58                     long long cmp = 1ll * (y -
59                     y1) * (x1 - x2) - 1ll *
60                     (x - x1) * (y1 - y2);
61                     if(cmp > 0){
62                         Q1.pop_back();
63                     } else break;
64                 }
65                 Q1.push_back(i);
66             }
67             if(Q2.size() ≤ 1){
68                 Q2.push_back(i);
69             } else {
70                 while(Q2.size() ≥ 2){
71                     auto &[x1, y1] = P[Q2[Q2.
72                     size() - 1]];
73                     auto &[x2, y2] = P[Q2[Q2.
74                     size() - 2]];
75                     long long cmp = 1ll * (y -
76                     y1) * (x1 - x2) - 1ll *
77                     (x - x1) * (y1 - y2);
78                     if(cmp < 0){
79                         Q2.pop_back();
80                     } else break;
81                 }
82                 Q2.push_back(i);
83             }
84         }
85         Q = Q1;
86         for(int i = Q2.size(); i ≠ 0; i --){
87             if(i ≠ Q2.size())
88                 Q.push_back(Q2[i - 1]);
89         }
90         long long area = 0;
91         int x0 = P[Q[0]].x;
92         int y0 = P[Q[0]].y;
93         for(int i = 1; i + 1 < Q.size(); ++ i){
94             auto &[x1, y1] = P[Q[i]];
95             auto &[x2, y2] = P[Q[i + 1]];
96             area += 1ll * (x1 - x0) * (y2 - y0)
97             - 1ll * (x2 - x0) * (y1 - y0);
98         }
99         area = -area;
100         for(auto &i: Q1) F[i] = true;
101         for(auto &i: Q2) F[i] = true;
102         bool ok = false;
103         for(int i = 1; i ≤ n; ++ i) if(!F[i]){
104             ok = true;
105             maxx = max(maxx, P[i].x);
106             minx = min(minx, P[i].x);
107         }

```

```

101         if(!ok){
102             cout << -1 << "\n";
103             continue;
104         }
105         vector<int> L1;
106         vector<int> L2;
107         // L1 插入 kx + b 维护下凸壳
108         for(int i = 1; i ≤ n; ++ i) if(!F[i]){
109             auto &[k, b] = P[i];
110             if(!L1.empty() && k == P[L1.back()
111             ].x)
112                 continue;
113             while(L1.size() ≥ 2){
114                 auto &P1 = P[L1[L1.size() -
115                 1]];
116                 auto &P2 = P[L1[L1.size() -
117                 2]];
118                 Frac i1 = intersect(P1, P[i]);
119                 Frac i2 = intersect(P2, P[i]);
120                 if(i1 < i2){
121                     L1.pop_back();
122                 } else break;
123             }
124             L1.push_back(i);
125         }
126         // L2 插入 kx + b 维护上凸壳
127         for(int i = n; i ≥ 1; -- i) if(!F[i]){
128             auto &[k, b] = P[i];
129             if(!L2.empty() && k == P[L2.back()
130             ].x)
131                 continue;
132             while(L2.size() ≥ 2){
133                 auto &P1 = P[L2[L2.size() -
134                 1]];
135                 auto &P2 = P[L2[L2.size() -
136                 2]];
137                 Frac i1 = intersect(P1, P[i]);
138                 Frac i2 = intersect(P2, P[i]);
139                 if(i1 < i2){
140                     L2.pop_back();
141                 } else break;
142             }
143             L2.push_back(i);
144         }
145         vector<Frac> E1;
146         E1.push_back(Frac(-2e9, 1));
147         for(int i = 0; i + 1 < L1.size(); ++ i){
148             auto &P1 = P[L1[i]];
149             auto &P2 = P[L1[i + 1]];
150             E1.push_back(intersect(P1, P2));
151         }
152         vector<Frac> E2;

```

```

147 E2.push_back(Frac( -2e9, 1 ));
148 for(int i = 0; i + 1 < L2.size(); ++ i){
149     auto &P1 = P[L2[i]];
150     auto &P2 = P[L2[i + 1]];
151     E2.push_back(intersect(P1, P2));
152 }
153 long long ans = 0;
154 for(int i = 0; i + 1 < Q.size(); ++ i){
155     auto &x1, y1 = P[Q[i]];
156     auto &x2, y2 = P[Q[i + 1]];
157     long long w = 1ll * x2 * y1 - 1ll
158         * x1 * y2;
159     int A = y2 - y1;
160     int B = x1 - x2;
161     int x = 0, y = 0;
162     if(B == 0){
163         if(A > 0){
164             x = minx, y = 0;
165         } else {
166             x = maxx, y = 0;
167         }
168     } else
169     if(B < 0){
170         Frac K = Frac(-A, -B);
171         int p = 0;
172         for(int k = 20; k ≥ 0; -- k){
173             int pp = p | 1 << k;
174             if(pp < E1.size() && E1[pp]
175                 < K){
176                 p = pp;
177             }
178         }
179         x = P[L1[p]].x;
180         y = P[L1[p]].y;
181     } else {
182         Frac K = Frac( A, B);
183         int p = 0;
184         for(int k = 20; k ≥ 0; -- k){
185             int pp = p | 1 << k;
186             if(pp < E2.size() && E2[pp]
187                 < K){
188                 p = pp;
189             }
190         }
191         x = P[L2[p]].x;
192         y = P[L2[p]].y;
193     }
194     ans = max(ans, area - (w + 1ll * A
195         * x + 1ll * B * y));
196 }
197 // cerr << "ans = " << ans << endl;
198 cout << ans << "\n";
199 }

```

```

196 return 0;
197 }

9.2 最小圆覆盖

1 #include "2d.cpp"
2 point geto(point a, point b, point c) {
3     double a1, a2, b1, b2, c1, c2;
4     point ans(0, 0);
5     a1 = 2 * (b.x - a.x), b1 = 2 * (b.y - a.y)
6     ,
7     c1 = sqr(b.x) - sqr(a.x) + sqr(b.y) - sqr(
8         a.y);
9     a2 = 2 * (c.x - a.x), b2 = 2 * (c.y - a.y)
10    ,
11    c2 = sqr(c.x) - sqr(a.x) + sqr(c.y) - sqr(
12        a.y);
13    if (equal(a1, 0)) {
14        ans.y = c1 / b1;
15        ans.x = (c2 - ans.y * b2) / a2;
16    } else if (equal(b1, 0)) {
17        ans.x = c1 / a1;
18        ans.y = (c2 - ans.x * a2) / b2;
19    } else {
20        ans.x = (c2 * b1 - c1 * b2) / (a2 * b1
21            - a1 * b2);
22        ans.y = (c2 * a1 - c1 * a2) / (b2 * a1
23            - b1 * a2);
24    }
25    return ans;
26 }
27 mt19937 MT;
28 circ minimal(vector <point> V){
29     shuffle(V.begin(), V.end(), MT);
30     point o = V[0];
31     double r = 0;
32     for(int i = 0; i < V.size(); ++ i) {
33         if (sign(dis(o, V[i]) - r) ≠ 1)
34             continue;
35         o.x = (V[i].x + V[0].x) / 2;
36         o.y = (V[i].y + V[0].y) / 2;
37         r = dis(V[i], V[0]) / 2;
38         for(int j = 0; j < i; ++ j) {
39             if (sign(dis(o, V[j]) - r) ≠ 1)
40                 continue;
41             o.x = (V[i].x + V[j].x) / 2;
42             o.y = (V[i].y + V[j].y) / 2;
43             r = dis(V[i], V[j]) / 2;
44             for(int k = 0; k < j; ++ k) {
45                 if (sign(dis(o, V[k]) - r) ≠
46                     1) continue;
47                 o = geto(V[i], V[j], V[k]);

```

```

39         r = dis(o, V[i]);
40     }
41 }
42 }
43 circ res;
44 res.o = o;
45 res.r = r;
46 return res;
47 }

```

### 9.3 最左转线

```

1 #include "2d.cpp"
2 namespace DSU{
3     const int MAXN = 1e5 + 3;
4     int F[MAXN];
5     int getfa(int u){
6         return u == F[u] ? u : F[u] = getfa(F[
7             u]);
8     }
9 }
10 namespace Dual{
11     const int MAXN = 1e5 + 3;
12     const int MAXM = 1e5 + 3;
13     int A[MAXN], B[MAXN], W[MAXN], I[MAXN], n,
14         m;
15     int outer;
16     bool cmp(int a, int b){
17         return W[a] < W[b];
18     }
19     vector <pair<int, int>> E[MAXN];
20     const int MAXT = 20 + 3;
21     int F[MAXN][MAXT], G[MAXN][MAXT], D[MAXN],
22         h = 20;
23     void dfs(int u, int f){
24         D[u] = D[f] + 1;
25         for(int i = 1; i ≤ h; ++ i)
26             F[u][i] = F[F[u][i - 1]][i - 1],
27             G[u][i] = max(G[u][i - 1], G[F[u][
28                 i - 1]][i - 1]);
29         for(auto &[v, w] : E[u]) if(v ≠ f){
30             G[v][0] = w;
31             F[v][0] = u;
32             dfs(v, u);
33         }
34     }
35     void build(){
36         for(int i = 1; i ≤ n; ++ i)
37             DSU :: F[i] = i;
38         for(int i = 1; i ≤ m; ++ i)
39             I[i] = i;
40         sort(I + 1, I + 1 + m, cmp);

```

```

37     for(int i = 1; i ≤ m; ++ i){
38         int a = A[I[i]];
39         int b = B[I[i]];
40         int w = W[I[i]];
41         int fa = DSU :: getfa(a);
42         int fb = DSU :: getfa(b);
43         if(fa ≠ fb){
44             DSU :: F[fa] = fb;
45             E[a].push_back({b, w});
46             E[b].push_back({a, w});
47         }
48     }
49     dfs(1, 0);
50 }
51 int solve(int u, int v){
52     if(u = outer || v = outer)
53         return -1;
54     int ans = 0;
55     if(D[u] < D[v]) swap(u, v);
56     for(int i = h; i ≥ 0; -- i)
57         if(D[F[u][i]] ≥ D[v]){
58             ans = max(ans, G[u][i]);
59             u = F[u][i];
60         }
61     if(u = v) return ans;
62     for(int i = h; i ≥ 0; -- i)
63         if(F[u][i] ≠ F[v][i]){
64             ans = max(ans, G[u][i]);
65             ans = max(ans, G[v][i]);
66             u = F[u][i];
67             v = F[v][i];
68         }
69     ans = max(ans, G[u][0]);
70     ans = max(ans, G[v][0]);
71     return ans;
72 }
73 }
74 namespace Planer{
75     const int MAXN = 1e5 + 3 + 3;
76     const int MAXE = 2e5 + 3;
77     const int MAXG = 1e5 + 3;
78     const int MAXQ = 2e5 + 3;
79     point P[MAXN];
80     using edge = tuple<int, int>;
81     double gety(int a, int b, double x){
82         return P[a].y + (x - P[a].x) / (P[b].x
83             - P[a].x) * (P[b].y - P[a].y);
84     }
85     double scanx;
86     struct Cmp1{
87         bool operator()(const pair<edge, int>
88             l1, const pair<edge, int> l2) const

```

```

87     {
88         const edge &e1 = l1.first;
89         const edge &e2 = l2.first;
90         double h1 = gety(get<0>(e1), get
91             <1>(e1), scanx);
92         double h2 = gety(get<0>(e2), get
93             <1>(e2), scanx);
94         return h1 < h2;
95     };
96     struct Cmp2{
97         bool operator()(const pair<edge, int>
98             l1, const pair<edge, int> l2) const
99     {
100         if(l1.second = l2.second)
101             return false;
102         const edge &e1 = l1.first;
103         const edge &e2 = l2.first;
104         vec v1 = P[get<1>(e1)] - P[get<0>(
105             e1)];
106         vec v2 = P[get<1>(e2)] - P[get<0>(
107             e2)];
108         if(sign(v1.y) ≠ sign(v2.y)){
109             return v1.y > 0;
110         } else {
111             return sign(mulx(v1, v2)) =
112                 1;
113         }
114     };
115     vector <pair<edge, int> > E[MAXN];
116     vector <int> G[MAXG];
117     int L[MAXE], R[MAXE], W[MAXE], n, m, q, o;
118     double theta;
119     int outer;
120     void rotate(){
121         srand(time(0));
122         theta = PI * rand() / RAND_MAX;
123     }
124     int add(double x, double y){
125         srand(time(0));
126         P[++ n] = rotate(vec(x, y), theta);
127         return n;
128     }
129     int link(int u, int v, int w){
130         ++ m;
131         E[u].push_back({{u, v}, ++ o});
132         L[o] = u, R[o] = v, W[o] = w;
133         E[v].push_back({{v, u}, ++ o});
134         L[o] = v, R[o] = u, W[o] = w;
135         return m;
136     }
137     int I[MAXE];

```

```

132 int polys;
133 pair<edge, int> findleft(int l, int r){
134     auto it = lower_bound(E[r].begin(), E[
135         r].end(), make_pair(edge(r, l), 0),
136         Cmp2());
137     if(it = E[r].begin())
138         return E[r].back();
139     else
140         return *(it - 1);
141 }
142 void leftmost(){
143     for(int i = 1; i ≤ n; ++ i){
144         sort(E[i].begin(), E[i].end(),
145             Cmp2());
146     }
147     for(int p = 1; p ≤ n; ++ p){
148         for(auto &[e1, id1] : E[p]){
149             auto &[x, y] = e1;
150             if(!I[id1]){
151                 int l = x;
152                 int r = y;
153                 I[id1] = ++ polys;
154                 G[polys].push_back(id1);
155                 while(r ≠ p){
156                     auto [e2, id2] =
157                         findleft(l, r);
158                     auto [a, b] = e2;
159                     I[id2] = polys;
160                     G[polys].push_back(id2
161                         );
162                     l = r;
163                     r = b;
164                 }
165             }
166         }
167     }
168     for(int i = 1; i ≤ polys; ++ i){
169         double area = 0;
170         for(int j = 0; j < G[i].size(); ++ j)
171             area += mulx(P[L[G[i][j]]], P[
172                 R[G[i][j]]]);
173         if(area < 0)
174             outer = i;
175     }
176 }
177 void dual(){
178     Dual :: n = polys;
179     Dual :: m = 0;
180     for(int i = 1; i ≤ m; ++ i){
181         int u = I[2 * i - 1], v = I[2 * i

```

```

    ], w = W[2 * i];
    if(u == outer || v == outer)
        w = 1e9L + 1;
    ++ Dual :: m;
    Dual :: A[ Dual :: m ] = u;
    Dual :: B[ Dual :: m ] = v;
    Dual :: W[ Dual :: m ] = w;
}
Dual :: build();
Dual :: outer = outer;
}
set <pair<edge, int>, Cmp1> S;
vector <pair<double, int>> T;
vector <pair<double, int>> Q;
double X[MAXQ], Y[MAXQ];
int Z[MAXQ];
int ask(double x, double y){
    ++ q;
    point p = rotate(vec(x, y), theta);
    X[q] = p.x;
    Y[q] = p.y;
    return q;
}
void locate(){
    T.clear(), Q.clear(), S.clear();
    for(int i = 1; i ≤ q; ++ i){
        Q.push_back(make_pair(X[i], i));
    }
    for(int i = 1; i ≤ polys; ++ i){
        for(auto &e : G[i]){
            int u = L[e];
            int v = R[e];
            if(P[u].x > P[v].x){
                T.push_back(make_pair(P[v].x + 1e-5, e));
                T.push_back(make_pair(P[u].x - 1e-5, -e));
            }
        }
    }
    sort(T.begin(), T.end());
    sort(Q.begin(), Q.end());
    int p1 = 0, p2 = 0;
    scanx = -1e9;
    Cmp1 CMP;
    while(p1 < Q.size() || p2 < T.size()){
        // for(auto it1 = S.begin(), it2 =
        // next(S.begin()); it2 ≠ S.end()
        // ++ it1, ++ it2)
        // assert(CMP(*it1, *it2));
        double x1 = p1 < Q.size() ? Q[p1].
        first : 1e9;

```

```

223 double x2 = p2 < T.size() ? T[p2].
224 first : 1e9;
225 scanx = min(x1, x2);
226 if(equal(scanx, x1)){
227     auto &x = X[Q[p1].second];
228     auto &y = Y[Q[p1].second];
229     auto &z = Z[Q[p1].second];
230     P[n + 1] = point(-1e9, y);
231     P[n + 2] = point(1e9, y);
232     auto it = S.lower_bound({{n +
233         1, n + 2}, 0});
234     if(it == S.end())
235         z = outer;
236     else
237         z = it → second;
238     ++ p1;
239 }
240 if(equal(scanx, x2)){
241     int g = T[p2].second;
242     if(g > 0){
243         assert(!S.count({{L[g], R[
244             g]}, I[g]}));
245         S.insert({{L[g], R[g]}, I[
246             g]});
247     } else {
248         g = -g;
249         assert(S.count({{L[g], R[
250             g]}, I[g]}));
251         S.erase({{L[g], R[g]}, I[
252             g]});
253     }
254     ++ p2;
255 }
256 }
257 }
258 const int MAXN = 1e5 + 3;
259 int A[MAXN], B[MAXN];
260 int main(){
261 #ifndef ONLINE_JUDGE
262     freopen("test.in", "r", stdin);
263     freopen("test.out", "w", stdout);
264 #endif
265 int n, m, q;
266 Planer :: rotate();
267 cin >> n >> m;
268 for(int i = 1; i ≤ n; ++ i){
    double x, y;
    cin >> x >> y;
    Planer :: add(x, y);
}
for(int i = 1; i ≤ m; ++ i){

```

```

269 int u, v, w;
270 cin >> u >> v >> w;
271 Planer :: link(u, v, w);
272 }
273 Planer :: leftmost();
274 Planer :: dual();
275 cin >> q;
276 for(int i = 1; i ≤ q; ++ i){
277     double a1, b1, a2, b2;
278     cin >> a1 >> b1;
279     A[i] = Planer :: ask(a1, b1);
280     cin >> a2 >> b2;
281     B[i] = Planer :: ask(a2, b2);
282 }
283 Planer :: locate();
284 for(int i = 1; i ≤ q; ++ i)
285     A[i] = Planer :: Z[A[i]],
286     B[i] = Planer :: Z[B[i]];
287 for(int i = 1; i ≤ q; ++ i){
288     int ans = Dual :: solve(A[i], B[i]);
289     cout << ans << endl;
290 }
291 return 0;
292 }

```

## 9.4 二维基础

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 using i64 = long long;
4 const int INF = 1e9;
5 const i64 INFL = 1e18;
6 int qread();
7 const double EPS = 1e-9;
8 const double PI = acos(-1);
9 bool equal(double a, double b){
10     return fabs(a - b) < EPS;
11 }
12 int sign(double a){
13     if(equal(a, 0))
14         return 0;
15     return a > 0 ? 1 : -1;
16 }
17 double sqr(double x){
18     return x * x;
19 }
20 struct vec{ // 二维向量
21     double x;
22     double y;
23     vec(){ }
24     vec(double _x, double _y) : x(_x), y(_y){ }
25 };

```

```

26 vec operator +(const vec &a, const vec &b){
27     return vec(a.x + b.x, a.y + b.y);
28 }
29 vec operator -(const vec &a, const vec &b){
30     return vec(a.x - b.x, a.y - b.y);
31 }
32 double mulp(const vec &a, const vec &b){
33     return a.x * b.x + a.y * b.y;
34 }
35 double mulx(const vec &a, const vec &b){
36     return a.x * b.y - a.y * b.x;
37 }
38 vec mul(const double &r, const vec &a){
39     return vec(r * a.x, r * a.y);
40 }
41 bool equal(vec a, vec b){
42     return equal(a.x, b.x) && equal(a.y, b.y);
43 }
44 using point = vec;
45 point rotate(point a, double t){
46     double c = cos(t);
47     double s = sin(t);
48     return point(a.x * c - a.y * s, a.y * c +
49                 a.x * s);
50 }
51 bool cmpx(point a, point b){
52     return sign(a.x - b.x) == -1;
53 }
54 bool cmpy(point a, point b){
55     return sign(a.y - b.y) == -1;
56 }
57 struct line{    // 有向直线
58     point o;
59     vec p;
60     line(point _o, vec _p) : o(_o), p(_p){}
61 };
62 struct segm{    // 有向线段
63     point a, b;
64     segm(point _a, point _b) : a(_a), b(_b){}
65 };
66 int side(line l, point p){
67     return sign(mulx(l.p, p - l.o));
68 }
69 int side(segm s, point p){
70     return sign(mulx(s.b - s.a, p - s.a));
71 }
72 bool parallel(line a, line b){
73     return equal(0, mulx(a.p, b.p));
74 }
75 double abs(vec a){
76     return sqrt(a.x * a.x + a.y * a.y);

```

```

77 double dis(point a, point b){
78     return sqrt(sqr(a.x - b.x) + sqr(a.y - b.y));
79 }
80 double abs(segm s){
81     return dis(s.a, s.b);
82 }
83 double dis(line a, point p){
84     return abs(mulx(p - a.o, a.p)) / abs(a.p);
85 }
86 point intersection(line a, line b){
87     return b.o + mul(mulx(b.o - a.o, a.p) /
88                     mulx(a.p, b.p), b.p);
89 }
90 bool intersect(double l1, double r1, double l2
91               , double r2){
92     if(l1 > r1) swap(l1, r1);
93     if(l2 > r2) swap(l2, r2);
94     if(equal(r1, l2) || equal(r2, l1))
95         return true;
96     return !equal(max(r1, r2) - min(l1, l2),
97                 r1 - l1 + r2 - l2);
98 }
99 bool intersect(segm s1, segm s2){
100     bool fx = intersect(s1.a.x, s1.b.x, s2.a.x
101                       , s2.b.x);
102     if(!fx) return false;
103     bool fy = intersect(s1.a.y, s1.b.y, s2.a.y
104                       , s2.b.y);
105     if(!fy) return false;
106     bool g1 = side(s1, s2.a) * side(s1, s2.b)
107             = 1;
108     if(g1) return false;
109     bool g2 = side(s2, s1.a) * side(s2, s1.b)
110             = 1;
111     if(g2) return false;
112     return true;
113 }
114 struct circ{    // 二维圆形
115     point o;
116     double r;
117 };
118 struct poly{    // 二维多边形
119     vector<point> P;
120 };
121 double area(point a, point b, point c){
122     return abs(mulx(b - a, c - a)) / 2;
123 }
124 double area(const poly &p){
125     double ans = 0;
126     for(int i = 0; i < P.size(); ++ i){
127         const point &l = P[i];

```

```

121     const point &r = P.P[i + 1 == P.P.size
122         () ? 0 : i + 1];
123     ans += mulx(l, r);
124 }
125 return ans / 2;

```

## 10 其他

### 10.1 笛卡尔树

```

1 #include "../header.cpp"
2 // Li: 左儿子; Ri: 右儿子
3 int n, L[MAXN], R[MAXN], A[MAXN];
4 void build(){
5     stack<int> S;
6     A[n + 1] = -1e9;
7     for(int i = 1; i ≤ n + 1; ++ i){
8         int v = 0;
9         while(!S.empty() && A[S.top()] > A[i]){
10             auto u = S.top();
11             R[u] = v, v = u, S.pop();
12         }
13         L[i] = v, S.push(i);
14     }
15 }

```

### 10.2 CDQ 分治

#### 10.2.1 例题

给定三元组序列  $(a_i, b_i, c_i)$ , 求解  $f(i) = \sum_j [a_j \leq a_i \wedge b_j \leq b_i \wedge c_j \leq c_i]$ .

```

1 #include "../header.cpp"
2 struct Node{
3     int id, a, b, c;
4 }A[MAXN], B[MAXN];
5 bool cmp(Node a, Node b){
6     if(a.a ≠ b.a) return a.a < b.a;
7     if(a.b ≠ b.b) return a.b < b.b;
8     if(a.c ≠ b.c) return a.c < b.c;
9     return a.id < b.id;
10 }
11 int K[MAXN], H[MAXN];
12 int qread();
13 int n, m, D[MAXM];
14 namespace BIT{
15     void increase(int x, int w){
16         while(x ≤ m) D[x] += w, x += x & -x;

```



```

17 }
18 void decrease(int x, int w){
19     while(x ≤ m) D[x] -= w, x += x & -x;
20 }
21 void query(int x, int &r){
22     while(x) r += D[x], x -= x & -x;
23 }
24 }
25 void cdq(int l, int r){
26     if(l ≠ r){
27         int t = l + r >> 1; cdq(l, t), cdq(t + 1,
28             r);
29         int p = l, q = t + 1, u = l;
30         while(p ≤ t && q ≤ r){
31             if(A[p].b ≤ A[q].b)
32                 BIT :: increase(A[p].c, 1), B[u ++] =
33                     A[p ++];
34             else
35                 BIT :: query(A[q].c, K[A[q].id]), B[u
36                     ++] = A[q ++];
37             up(l, t, i) BIT :: decrease(A[i].c, 1);
38             up(l, r, i) A[i] = B[i];
39         }
40     }
41     int main(){
42         n = qread(), m = qread();
43         up(1, n, i) A[i].id = i, A[i].a = qread(), A
44             [i].b = qread(), A[i].c = qread();
45         sort(A + 1, A + 1 + n, cmp), cdq(1, n);
46         sort(A + 1, A + 1 + n, cmp);
47         dn(n, 1, i){
48             if(A[i].a = A[i + 1].a && A[i].b = A[i +
49                 1].b && A[i].c = A[i + 1].c)
50                 K[A[i].id] = K[A[i + 1].id];
51                 H[K[A[i].id]] ++;
52         }
53         up(0, n - 1, i) printf("%d\n", H[i]);
54         return 0;
55     }

```

## 10.3 自适应辛普森

### 10.3.1 例题

计算

$$\int_0^{+\infty} x^{(a/x)-x}$$

```

1 #include "../header.cpp"
2 double simpson(double (*f)(double), double l,
3     double r){
4     double mid = (l + r) / 2;
5     return (r - l) * (f(l) + 4 * f(mid) + f(r))
6         / 6.0;
7 }
8 double adapt_simpson(double (*f)(double),
9     double l, double r, double EPS, int step){
10     double mid = (l + r) / 2;
11     double w0 = simpson(f, l, r);
12     double w1 = simpson(f, l, mid);
13     double w2 = simpson(f, mid, r);
14     if(fabs(w0 - w1 - w2) < EPS && step < 0)
15         return w1 + w2;
16     else
17         return adapt_simpson(f, l, mid, EPS, step
18             - 1) +
19             adapt_simpson(f, mid, r, EPS, step
20                 - 1);
21 }
22 double a, l, r;
23 double fun(double x){
24     return pow(x, a / x - x);
25 }
26 int main(){
27     cin >> a;
28     if(a < 0)
29         cout << "orz" << endl;
30     else {
31         l = 1e-9, r = 150;
32         cout << fixed << setprecision(5) <<
33             adapt_simpson(fun, l, r, 1e-9, 15);
34     }
35 }

```

## 10.4 模拟退火

### 10.4.1 例题

给定  $n$  个物品挂在洞下，第  $i$  个物品坐标  $(x_i, y_i)$  重量为  $w_i$ 。询问平衡点。

```

1 #include "../header.cpp"
2 const double T0 = 2e3, Tk = 1e-14, delta =
3     0.993, R = 1e-3;
4 mt19937 MT(114514);
5 double distance(double x, double y, double a,
6     double b){
7     return sqrt(pow(a - x, 2) + pow(b - y, 2));
8 }

```

```

7 const int MAXN = 1e3 + 3;
8 double X[MAXN], Y[MAXN], W[MAXN]; int n;
9 double calculate(double x, double y){
10     double gx, gy, a;
11     for(int i = 0; i < n; ++i){
12         a = atan2(y - Y[i], x - X[i]);
13         gx += cos(a) * W[i];
14         gy += sin(a) * W[i];
15     }
16     return pow(gx, 2) + pow(gy, 2);
17 }
18 double ex, ey, eans = 1e18;
19 void SA(){
20     double T = T0, x = 0, y = 0, ans = calculate
21         (x, y);
22     double ansx, ansy;
23     uniform_real_distribution<double> U;
24     while(T > Tk){
25         double nx, ny, nans;
26         nx = x + 2 * (U(MT) - .5) * T;
27         ny = y + 2 * (U(MT) - .5) * T;
28         if((nans = calculate(nx, ny)) < ans){
29             ans = nans;
30             ansx = x = nx;
31             ansy = y = ny;
32         } else if(exp(-distance(nx, ny, x, y) / T
33             / R) > U(MT)){
34             x = nx, y = ny;
35         }
36         T *= delta;
37     }
38     if(ans < eans) eans = ans, ex = ansx, ey =
39         ansy;
40 }

```

## 10.5 伪随机生成

```

1 #include "../header.cpp"
2 u32 xorshift32(u32 &x){
3     x ^= x << 13, x ^= x >> 17, x ^= x << 5;
4     return x;
5 }
6 u64 xorshift64(u64 &x){
7     x ^= x << 13, x ^= x >> 17, x ^= x << 17;
8     return x;
9 }

```

## 11 header

```

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

```

<pre>2 using namespace std; 3 #define up(l, r, i) for(int i = l, END##i = r;   i ≤ END##i; ++ i) 4 #define dn(r, l, i) for(int i = r, END##i = l;   i ≥ END##i; -- i) 5 using i64 = long long;</pre>	<pre>6 using f80 = long double; 7 using u32 = unsigned; 8 using u64 = unsigned long long; 9 const int INF = 1e9; 10 const i64 INFL = 1e18; 11 int qread();</pre>	<pre>12 int power(int a, int b); 13 int power(int a, int b, int p); 14 const int MAXN = 10 + 3, MAXM = 10 + 3; 15 const int MOD = 998244353;</pre>
--	--	--