Hourai's Template

哈尔滨工业大学 蓬莱人形

2024 年 11 月 7 日

目录

1 动态规划

1.1 多重背包

1.1.1 用法

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。

```cpp
#include <bits/stdc++.h>
using namespace std;
using i64 = long long;
const int INF  =  1e9;
const i64 INFL = 1e18;
int qread();
const int MAXN = 4e4 + 3;
int F[MAXN];
int main(){
    int n, m;
    cin >> n >> m;
    for(int i = 1;i <= n;++ i){
        int w, v, c;
        cin >> w >> v >> c;
        // w: value, v: volume, c: count
        for(int j = 0;j < v;++ j){
            deque <tuple<int, int> > Q;
            for(int k = 0;j + k * v <= m;++ k){
                int x = j + k * v;
                int f = F[x] - (x / v) * w;
                while(!Q.empty() && get<0>(Q.back ()) <= f)
                    Q.pop_back ();
                Q.push_back({f, x});
                while(!Q.empty() && get<1>(Q.front()) <  x - c * v)
                    Q.pop_front();
                F[x] = get<0>(Q.front()) + (x / v) * w;
            }
        }
    }
    cout << F[m] << endl;
    return 0;
}
```

1.2 树形背包

```cpp
#include<bits/stdc++.h>
using namespace std;
typedef long long i64;
const int MAXN = 2e3 + 3;
vector<int> E[MAXN];
int W[MAXN];
int F[MAXN][MAXN], S[MAXN];
void dfs(int u, int f){
    F[u][1] = W[u];
    S[u]    = 1;
    for(auto &v : E[u]) if(v != f){
        dfs(v, u);
        for(int i = S[u];i >= 1;-- i)
```

```
            for(int j = S[v];j >= 1;-- j)
                F[u][i + j] = max(F[u][i + j], F[u][i] + F[v][j]);
        S[u] += S[v];
    }
}
int main(){
    int n, m;
    cin >> n >> m;
    for(int i = 1;i <= n;++ i){
        int f;
        cin >> f >> W[i];
        E[f].push_back(i);
    }
    dfs(0, 0);
    cout << F[0][m + 1] << endl;
    return 0;
}
```

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。

```
#include<bits/stdc++.h>
#define up(l, r, i) for(int i = l, END##i = r;i <= END##i;++ i)
#define dn(r, l, i) for(int i = r, END##i = l;i >= END##i;-- i)
using namespace std;
typedef long long i64;
const int INF = 1e9;
const int MAXN = 1e5 + 3;
int W[MAXN];
struct Mat{ int M[2][2]; };
struct Vec{ int V[2];    };
Mat operator *(const Mat &a, const Mat &b){
    Mat c;
    c.M[0][0] = max(a.M[0][0] + b.M[0][0], a.M[0][1] + b.M[1][0]);
    c.M[0][1] = max(a.M[0][0] + b.M[0][1], a.M[0][1] + b.M[1][1]);
    c.M[1][0] = max(a.M[1][0] + b.M[0][0], a.M[1][1] + b.M[1][0]);
    c.M[1][1] = max(a.M[1][0] + b.M[0][1], a.M[1][1] + b.M[1][1]);
    return c;
}
Vec operator *(const Mat &a, const Vec &v){
    Vec r;
    r.V[0] = max(a.M[0][0] + v.V[0], a.M[0][1] + v.V[1]);
    r.V[1] = max(a.M[1][0] + v.V[0], a.M[1][1] + v.V[1]);
    return r;
}
namespace Gra{
    vector<int> E[MAXN];
    int G[MAXN], S[MAXN], D[MAXN], T[MAXN], F[MAXN];
    int X[MAXN], Y[MAXN];
    int H[MAXN][2];
    int K[MAXN][2];
    struct Mat M[MAXN];
    void dfs1(int u, int f){
        S[u] = 1;
        F[u] = f;
```

```
        for(auto &v : E[u]) if(v ≠ f){
            dfs1(v, u);
            S[u] += S[v];
            if(S[v] > S[G[u]]) G[u] = v;
        }
    }
    int o;
    void dfs2(int u, int f){
        if(u == G[f])
            X[u] = X[f];
        else
            X[u] = u;
        H[u][0] = H[u][1] = 0;
        K[u][0] = K[u][1] = 0;
        const int &g = G[u];
        D[u] = ++ o;
        T[o] = u;
        if(g){
            dfs2(g, u);
            Y[u] = Y[g];
            K[u][0] += max(K[g][0], K[g][1]);
            K[u][1] += K[g][0];
        } else {
            Y[u] = u;
        }
        for(auto &v : E[u]) if(v ≠ f && v ≠ g){
            dfs2(v, u);
            H[u][0] += max(K[v][0], K[v][1]);
            H[u][1] += K[v][0];
        }
        M[u].M[0][0] = H[u][0];
        M[u].M[0][1] = H[u][0];
        M[u].M[1][0] = H[u][1] + W[u];
        M[u].M[1][1] = -INF;
        K[u][0] += H[u][0];
        K[u][1] += H[u][1] + W[u];
    }
}
namespace Seg{
    const int SIZ = 4e5 + 3;
    struct Mat M[SIZ];
    #define lc(t) (t << 1)
    #define rc(t) (t << 1 | 1)
    void pushup(int t, int a, int b){
        M[t] = M[lc(t)] * M[rc(t)];
    }
    void build(int t, int a, int b){
        if(a == b){
            M[t] = Gra :: M[Gra :: T[a]];
        } else {
            int c = a + b >> 1;
            build(lc(t), a, c);
            build(rc(t), c + 1, b);
            pushup(t, a, b);
        }
    }
    void modify(int t, int a, int b, int p, const Mat &w){
```

```
92          if(a == b){
93              M[t] = w;
94          } else {
95              int c = a + b >> 1;
96              if(p <= c) modify(lc(t), a, c, p, w);
97                  else   modify(rc(t), c + 1, b, p, w);
98              pushup(t, a, b);
99          }
100     }
101     Mat query(int t, int a, int b, int l, int r){
102         if(l <= a && b <= r){
103             return M[t];
104         } else {
105             int c = a + b >> 1;
106             if(r <= c) return query(lc(t), a, c    , l, r); else
107             if(l >  c) return query(rc(t), c + 1, b, l, r); else
108                 return query(lc(t), a, c    , l, r) *
109                        query(rc(t), c + 1, b, l, r);
110         }
111     }
112 }
113 int qread();
114 int main(){
115     int n = qread(), m = qread();
116     up(1, n, i)
117         W[i] = qread();
118     up(2, n, i){
119         int u = qread(), v = qread();
120         Gra :: E[u].push_back(v);
121         Gra :: E[v].push_back(u);
122     }
123     Gra :: dfs1(1, 0);
124     Gra :: dfs2(1, 0);
125     Seg :: build(1, 1, n);
126     Vec v0;
127     v0.V[0] = v0.V[1] = 0;
128     up(1, m, i){
129         using namespace Gra;
130         int x = qread(), y = qread();
131         W[x] = y;
132         int u = x;
133         while(u != 0){
134             const int &v = X[u];
135             const int &f = F[v];
136             M[u].M[0][0] = H[u][0];
137             M[u].M[0][1] = H[u][0];
138             M[u].M[1][0] = H[u][1] + W[u];
139             M[u].M[1][1] = -INF;
140             const Vec p = Seg :: query(1, 1, n, D[v], D[Y[u]]) * v0;
141             Seg :: modify(1, 1, n, D[u], M[u]);
142             const Vec q = Seg :: query(1, 1, n, D[v], D[Y[u]]) * v0;
143             if(f != 0){
144                 H[f][0] = H[f][0] - max(p.V[0], p.V[1]) + max(q.V[0], q
                        .V[1]);
145                 H[f][1] = H[f][1] - p.V[0] + q.V[0];
146             }
147             u = f;
```

```
        }
        Vec v1 = Seg :: query(1, 1, n, D[1], D[Y[1]]) * v0;
        printf("%d\n", max(v1.V[0], v1.V[1]));
    }
    return 0;
}
```

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?

```
#include<bits/stdc++.h>
#define up(l, r, i) for(int i = l, END##i = r;i <= END##i;++ i)
#define dn(r, l, i) for(int i = r, END##i = l;i >= END##i;-- i)
using namespace std;
using i64 = long long;
const int INF  =  1e9;
const i64 INFL = 1e18;
const int MAXN = 20 + 3;
const int MAXM = 67108864 + 3;
namespace HashT{
    const int SIZ = 19999997;
    int H[SIZ], V[SIZ], N[SIZ], t;
    bool F[SIZ];
    i64  W[SIZ];
    void add(int u, int v, bool f, i64 w){
        V[++ t] = v, N[t] = H[u], F[t] = f, W[t] = w, H[u] = t;
    }
    i64& find(int u, bool f){
        for(int p = H[u % SIZ];p;p = N[p])
            if(V[p] == u && F[p] == f)
                return W[p];
        add(u % SIZ, u, f, 0);
        return W[t];
    }
}
char S[MAXN][MAXN];
int qread();
int n, m;
vector <pair<pair<int, bool>, i64> > M[2];
int getp(int s, int p){
    return (s >> (2 * p - 2)) & 3;
}
int setw(int s, int p, int w){
    return (s & ~(3 << (2 * p - 2))) | (w << (2 * p - 2));
}
int findr(int s, int p){
    int c = 0;
    for(int q = p;q <= m + 1;++ q){
        if(((s >> (2 * q - 2)) & 3) == 1) ++ c;
        if(((s >> (2 * q - 2)) & 3) == 2) -- c;
        if(c == 0)
            return q;
    }
```

```
44      return -1;
45  }
46  int findl(int s, int p){
47      int c = 0;
48      for(int q = p;q >= 1;-- q){
49          if(((s >> (2 * q - 2)) & 3) == 2) ++ c;
50          if(((s >> (2 * q - 2)) & 3) == 1) -- c;
51          if(c == 0)
52              return q;
53      }
54      return -1;
55  }
56  void state(int s){
57      return ;
58      up(1, m + 1, i){
59          switch(getp(s, i)){
60              case 0 : putchar('#'); break;
61              case 1 : putchar('('); break;
62              case 2 : putchar(')'); break;
63              case 3 : putchar('E');
64          }
65      }
66      puts("");
67  }
68  int main(){
69      n = qread(), m = qread();
70      up(1, n, i)
71          scanf("%s", S[i] + 1);
72      int o = 0;
73      #define X M[ o]
74      #define Y M[!o]
75      vector <pair<int, bool> > T;
76      X.push_back({{0, 0}, 1});
77      up(1, n, i){
78          Y.clear();
79          for(auto &u : X){
80              auto [s0, c] = u;
81              auto [s, f] = s0;
82              if(getp(s, m + 1) == 0)
83                  Y.push_back({{s << 2, f}, c});
84          }
85          o ^= 1;
86          up(1, m, j){
87              int x = j, y = j + 1;
88              for(auto &u : X){
89                  auto [s0, c] = u;
90                  auto [s, f] = s0;
91                  int a = getp(s, x);
92                  int b = getp(s, y);
93                  int t = setw(setw(s, x, 0), y, 0);
94                  #define update(t, c) HashT :: find(t, f) += c, T.
                        push_back({t, f})
95                  if(S[i][j] == '.'){     // 经过该格
96                      if(a == 1 && b == 1){
97                          t = setw(t, findr(s, y), 1),
98                          update(t, c);
99                      } else
```

```
100                     if(a == 2 && b == 2){
101                         t = setw(t, findl(s, x), 2),
102                         update(t, c);
103                     } else
104                     if(a == 1 && b == 2){
105                         if(f == false)  // 还没有闭合回路
106                             f = true, update(t, c);
107                     } else
108                     if(a == 2 && b == 1){
109                         update(t, c);
110                     } else
111                     if(a == 0 && b == 0){
112                         t = setw(t, x, 1);
113                         t = setw(t, y, 2);
114                         update(t, c);
115                     } else {    // a == 0 || b == 0
116                         int t1 = setw(t, x, a | b);
117                         int t2 = setw(t, y, a | b);
118                         update(t1, c);
119                         update(t2, c);
120                     }
121                 }
122                 if(S[i][j] == '*'){ // 不经过该格
123                     if(a == 0 && b == 0)
124                         update(t, c);
125                 }
126             }
127             Y.clear();
128             for(auto &u : T){
129                 auto [s, f] = u;
130                 if(HashT :: find(s, f) != 0){
131                     Y.push_back({{s, f}, HashT :: find(s, f)});
132                     HashT :: find(s, f) = 0;
133                 }
134             }
135             T.clear(), o ^= 1;
136         }
137     }
138     i64 ans = 0;
139     for(auto &u : X){
140         auto [s0, c] = u;
141         auto [s, f] = s0;
142         bool g = true;
143         up(1, m + 1, i)
144             g &= getp(s, i) == 0;
145         f &= g;
146         if(f)
147             ans = c;
148     }
149     printf("%lld\n", ans);
150     return 0;
151 }
```

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 $b(j) + c(i,j)$ 可视作关于 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 $s_i = i + \sum_{j\le i} c_i, L' = L + 1$。将其分类得到：

$$\begin{aligned} f_i &= \max_{j<i}\{f_j + s_i^2 + s_j^2 + L'^2 - 2s_is_j + 2s_jL' - 2s_iL'\} \\ &= (s_i^2 - 2s_iL' + L'^2) + \max_{j<i}\{(f_j + s_j^2 + 2s_jL') - 2s_is_j\} \end{aligned}$$

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。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long   i64;
typedef long double f80;
const int INF = 2147483647;
const int MAXN= 5e4 + 3;
int n, L, p, e, C[MAXN], Q[MAXN];
f80 S[MAXN], F[MAXN];
f80 gtx(int x){ return S[x]; }
f80 gty(int x){ return F[x] + S[x] * S[x]; }
f80 gtw(int x){ return -2.0 * (L - S[x]); }
f80 gtk(int x,int y){ return (gty(y) - gty(x)) / (gtx(y) - gtx(x)); }
int main(){
    cin >> n >> L;
    for(int i = 1;i <= n;++ i){
        cin >> C[i];
        S[i] = S[i - 1] + C[i];
    }
    for(int i = 1;i <= n;++ i){
        S[i] += i;
    }
    e = p = 1, L ++, Q[p] = 0;
    for(int i = 1;i <= n;++ i){
        while(e < p && gtk(Q[e], Q[e + 1]) < gtw(i))
            ++ e;
```

```
        int j = Q[e];
        F[i] = F[j] + pow(S[i] - S[j] - L, 2);
        while(1 < p && gtk(Q[p - 1], Q[p]) > gtk(Q[p], i))
            e -= (e == p), -- p;
        Q[++ p] = i;
    }
    printf("%.0Lf\n", F[n]);
    return 0;
}
```

2 数据结构

2.1 平衡树

2.1.1 无旋 Treap

```
#include<bits/stdc++.h>
using namespace std;
using i64 = long long;
const int INF  =  1e9;
const i64 INFL = 1e18;
typedef unsigned int       u32;
typedef unsigned long long u64;
mt19937_64 MT(114514);
namespace Treap{
    const int SIZ = 1e6 + 1e5 + 3;
    int F[SIZ], C[SIZ], S[SIZ], W[SIZ], X[SIZ][2], sz;
    u64 H[SIZ];
    int  newnode(int w){
        W[++ sz] = w, C[sz] = S[sz] = 1; H[sz] = MT();
        return sz;
    }
    void pushup(int x){
        S[x] = C[x] + S[X[x][0]] + S[X[x][1]];
    }
    pair<int, int> split(int u, int x){
        if(u == 0)
            return make_pair(0, 0);
        if(W[u] > x){
            auto [a, b] = split(X[u][0], x);
            X[u][0] = b, pushup(u);
            return make_pair(a, u);
        } else {
            auto [a, b] = split(X[u][1], x);
            X[u][1] = a, pushup(u);
            return make_pair(u, b);
        }
    }
    int merge(int a, int b){
        if(a == 0 || b == 0)
            return a | b;
        if(H[a] < H[b]){
            X[a][1] = merge(X[a][1], b), pushup(a);
            return a;
        } else {
            X[b][0] = merge(a, X[b][0]), pushup(b);
```

```
41              return b;
42          }
43      }
44      void insert(int &root, int w){
45          auto [p, q] = split(root, w    );
46          auto [a, b] = split(   p, w - 1);
47          if(b ≠ 0){
48              ++ S[b], ++ C[b];
49          } else b = newnode(w);
50          p    = merge(a, b);
51          root = merge(p, q);
52      }
53      void erase(int &root, int w){
54          auto [p, q] = split(root, w    );
55          auto [a, b] = split(   p, w - 1);
56          -- C[b], -- S[b];
57          p    = C[b] == 0 ? a : merge(a, b);
58          root = merge(p, q);
59      }
60      int  find_rank(int &root, int w){
61          int x = root, o = x, a = 0;
62          for(;x;){
63              if(w <  W[x])
64                  o = x, x = X[x][0];
65              else {
66                  a += S[X[x][0]];
67                  if(w == W[x]){
68                      o = x; break;
69                  }
70                  a += C[x];
71                  o = x, x = X[x][1];
72              }
73          }
74          return a + 1;
75      }
76      int  find_kth(int &root, int w){
77          int x = root, o = x, a = 0;
78          for(;x;){
79              if(w ⩽ S[X[x][0]])
80                  o = x, x = X[x][0];
81              else {
82                  w -= S[X[x][0]];
83                  if(w ⩽ C[x]){
84                      o = x; break;
85                  }
86                  w -= C[x];
87                  o = x, x = X[x][1];
88              }
89          }
90          return W[x];
91      }
92      int  find_pre(int &root, int w){
93          return find_kth(root, find_rank(root, w) - 1);
94      }
95      int  find_suc(int &root, int w){
96          return find_kth(root, find_rank(root, w + 1));
97      }
```

```
}
// ==== TEST ====
int qread();
int main(){
    using namespace Treap;
    int n = qread(), m = qread(), root = 0;
    for(int i = 1;i ⩽ n;++ i){
        int a = qread(); insert(root, a);
    }
    int last_ans = 0, ans = 0;
    for(int i = 1;i ⩽ m;++ i){
        int op = qread(), x = qread() ^ last_ans;
        switch(op){
            case 1 : insert(root, x); break;
            case 2 : erase (root, x); break;
            case 3 : ans ^= (last_ans = find_rank(root, x)); break;
            case 4 : ans ^= (last_ans = find_kth (root, x)); break;
            case 5 : ans ^= (last_ans = find_pre (root, x)); break;
            case 6 : ans ^= (last_ans = find_suc (root, x)); break;
        }
    }
    printf("%d\n", ans);
    return 0;
}
```

2.1.2 Splay

```
#include<bits/stdc++.h>
using namespace std;
using i64 = long long;
const int INF  =  1e9;
const i64 INFL = 1e18;
namespace Splay{
    const int SIZ = 1e6 + 1e5 + 3;
    int F[SIZ], C[SIZ], S[SIZ], X[SIZ][2], size;
    bool T[SIZ];
    bool is_root(int x){ return   F[x]     == 0;}
    bool is_rson(int x){ return X[F[x]][1] == x;}
    void push_down(int x){
        if(!T[x]) return;
        int lc = X[x][0], rc = X[x][1];
        if(lc) T[lc] ^= 1, swap(X[lc][0], X[lc][1]);
        if(rc) T[rc] ^= 1, swap(X[rc][0], X[rc][1]);
        T[x] = 0;
    }
    void pushup(int x){
        S[x] = C[x] + S[X[x][0]] + S[X[x][1]];
    }
    void rotate(int x){
        int y = F[x], z = F[y];
        bool f = is_rson(x);
        bool g = is_rson(y);
        int &t = X[x][!f];
        if(z){ X[z][g] = x; }
        if(t){ F[t]    = y; }
        X[y][f] = t, t = y;
        F[y] = x, pushup(y);
```

```
        F[x] = z, pushup(x);
    }
    void splay(int &r, int x, int g = 0){
        for(int f;f = F[x], f != g;rotate(x))
            if(F[f] != g) rotate(is_rson(x) == is_rson(f) ? f : x);
        if(is_root(x)) r = x;
    }
    int  get_kth(int &r, int w){
        int x = r, o = x;
        for(;x;){
            push_down(x);
            if(w <= S[X[x][0]]) o = x, x = X[x][0]; else {
                w -= S[X[x][0]];
                if(C[x] && w <= C[x]){o = x; break;}
                w -= C[x], o = x, x = X[x][1];
            }
        }
        splay(r, o); return o;
    }
    int  build(int l, int r){
        if(l == r){
            C[l] = S[l] = 1; return l;
        }
        int c = l + r >> 1, a = 0, b = 0;
        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]);
    }
}
int qread();
int main(){
    using namespace Splay;
    int n = qread(), m = qread();
    int root = build(1, n + 2);
    for(int i = 1;i <= m;++ i){
        int l = qread() + 1, r = qread() + 1;
        int u = get_kth(root, r + 1);
        int v = get_kth(root, l - 1);
        splay(root, v, 0), splay(root, u, v);
        int t = X[u][0];
        T[t] ^= 1, swap(X[t][0], X[t][1]);
    }
    output(n, root);
    return 0;
}
```

2.1.3 Treap

```
#include<bits/stdc++.h>
#define up(l, r, i) for(int i = l, END##i = r;i <= END##i;++ i)
#define dn(r, l, i) for(int i = r, END##i = l;i >= END##i;-- i)
```

```
4  using namespace std;
5  typedef long long i64;
6  const int INF = 2147483647;
7  typedef unsigned int       u32;
8  typedef unsigned long long u64;
9  mt19937_64 MT(114514);
10 namespace Treap{
11     const int SIZ = 1e6 + 1e5 + 3;
12     int F[SIZ], C[SIZ], S[SIZ], W[SIZ], X[SIZ][2], sz;
13     u64 H[SIZ];
14     bool is_root(int x){ return   F[x]     == 0;}
15     bool is_rson(int x){ return X[F[x]][1] == x;}
16     int  newnode(int w){
17         W[++ sz] = w, C[sz] = S[sz] = 1; H[sz] = MT();
18         return sz;
19     }
20     void pushup(int x){
21         S[x] = C[x] + S[X[x][0]] + S[X[x][1]];
22     }
23     void rotate(int &root, int x){
24         int y = F[x], z = F[y];
25         bool f = is_rson(x);
26         bool g = is_rson(y);
27         int &t = X[x][!f];
28         if(z){ X[z][g] = x; } else root = x;
29         if(t){ F[t]    = y; }
30         X[y][f] = t, t = y;
31         F[y] = x, pushup(y);
32         F[x] = z, pushup(x);
33     }
34     void insert(int &root, int w){
35         if(root == 0) {root = newnode(w); return;}
36         int x = root, o = x;
37         for(;x;o = x, x = X[x][w > W[x]]){
38             ++ S[x]; if(w == W[x]){ ++ C[x], o = x; break;}
39         }
40         if(W[o] != w){
41             if(w < W[o]) X[o][0] = newnode(w), F[sz] = o, o = sz;
42             else         X[o][1] = newnode(w), F[sz] = o, o = sz;
43         }
44         while(!is_root(o) && H[o] < H[F[o]])
45             rotate(root, o);
46     }
47     void erase(int &root, int w){
48         int x = root, o = x;
49         for(;x;o = x, x = X[x][w > W[x]]){
50             -- S[x]; if(w == W[x]){ -- C[x], o = x; break;}
51         }
52         if(C[o] == 0){
53             while(X[o][0] || X[o][1]){
54                 u64 wl = X[o][0] ? H[X[o][0]] : ULLONG_MAX;
55                 u64 wr = X[o][1] ? H[X[o][1]] : ULLONG_MAX;
56                 if(wl < wr){
57                     int p = X[o][0]; rotate(root, p);
58                 } else {
59                     int p = X[o][1]; rotate(root, p);
60                 }
```

```
            }
            if(is_root(o)){
                root = 0;
            } else {
                X[F[o]][is_rson(o)] = 0;
            }
        }
    }
    int  find_rank(int &root, int w){
        int x = root, o = x, a = 0;
        for(;x;){
            if(w <  W[x])
                o = x, x = X[x][0];
            else {
                a += S[X[x][0]];
                if(w == W[x]){
                    o = x; break;
                }
                a += C[x];
                o = x, x = X[x][1];
            }
        }
        return a + 1;
    }
    int  find_kth(int &root, int w){
        int x = root, o = x, a = 0;
        for(;x;){
            if(w <= S[X[x][0]])
                o = x, x = X[x][0];
            else {
                w -= S[X[x][0]];
                if(w <= C[x]){
                    o = x; break;
                }
                w -= C[x];
                o = x, x = X[x][1];
            }
        }
        return W[x];
    }
    int  find_pre(int &root, int w){
        return find_kth(root, find_rank(root, w) - 1);
    }
    int  find_suc(int &root, int w){
        return find_kth(root, find_rank(root, w + 1));
    }
}
int qread();
int main(){
    using namespace Treap;
    int n = qread(), m = qread(), root = 0;
    up(1, n, i){
        int a = qread(); insert(root, a);
    }
    int last_ans = 0, ans = 0;
    up(1, m, i){
        int op = qread(), x = qread() ^ last_ans;
```

```
        switch(op){
            case 1 : insert(root, x); break;
            case 2 : erase (root, x); break;
            case 3 : ans ^= (last_ans = find_rank(root, x)); break;
            case 4 : ans ^= (last_ans = find_kth (root, x)); break;
            case 5 : ans ^= (last_ans = find_pre (root, x)); break;
            case 6 : ans ^= (last_ans = find_suc (root, x)); break;
        }
    }
    printf("%d\n", ans);
    return 0;
}
```

2.2 珂朵莉树

```
#include<bits/stdc++.h>
using namespace std;
using u64 = unsigned long long;
const int MAXN = 1e6 + 3;
int power(int a, int b, int p){
    int r = 1;
    while(b){
        if(b & 1) r = 1ll * r * a % p;
        b >>= 1,  a = 1ll * a * a % p;
    }
    return r;
}
namespace ODT {
    // <pos_type, value_type>
    map <int, long long> M;
    // 分裂为 [1, p) 和 [p, +inf), 返回后者迭代器
    auto split(int p) {
        auto it = prev(M.upper_bound(p));
        return M.insert(
            it,
            make_pair(p, it -> second)
        );
    }
    // 区间赋值
    void assign(int l, int r, int v) {
        auto it = split(l);
        split(r + 1);
        while (it -> first != r + 1) {
            it = M.erase(it);
        }
        M[l] = v;
    }
    // // 执行操作
    // void perform(int l, int r) {
    //     auto it = split(l);
    //     split(r + 1);
    //     while (it -> first != r + 1) {
    //         // Do something...
    //         it = next(it);
    //     }
    // }
    void modify1(int l, int r, int w) {
```

```
        auto it = split(l);
        split(r + 1);
        while(it -> first != r + 1) {
            it -> second += w;
            it = next(it);
        }
    }
    void modify2(int l, int r, int w) {
        assign(l, r, w);
    }
    long long query1(int l, int r, int k) {
        auto it = split(l);
        split(r + 1);
        map <long long, int> T;
        while(it -> first != r + 1) {
            T[it -> second] += next(it) -> first - it -> first;
            it = next(it);
        }
        for(auto &[w, c]: T){
            if(c >= k)
                return w;
            k -= c;
        }
        return -1;
    }
    long long query2(int l, int r, int x, int y) {
        auto it = split(l);
        split(r + 1);
        int ans = 0;
        while(it -> first != r + 1) {
            int c = next(it) -> first - it -> first;
            long long a = it -> second;
            ans = (ans + 1ll * c * power(a % y, x, y)) % y;
            it = next(it);
        }
        return ans;
    }
};
const int MOD = 1e9 + 7;
int read(int &seed){
    int ret = seed;
    seed = (seed * 7ll + 13) % MOD;
    return ret;
}
int main(){
    ios :: sync_with_stdio(false);
    cin.tie(nullptr);
    int n, m, seed, vmax;
    cin >> n >> m >> seed >> vmax;
    ODT :: M[n + 1] = 0;
    for(int i = 1;i <= n;++ i){
        int a = read(seed) % vmax + 1;
        ODT :: M[i] = a;
    }
    for(int i = 1;i <= m;++ i){
        int op = read(seed) % 4 + 1;
        int l = read(seed) % n + 1;
```

```
        int r = read(seed) % n + 1;
        int x, y;
        if(l > r)
            swap(l, r);
        if(op == 3){
            x = (read(seed) % (r - l + 1)) + 1;
        } else
            x = read(seed) % vmax + 1;
        if(op == 4)
            y = read(seed) % vmax + 1;
        if(op == 1){
            ODT :: modify1(l, r, x);
        } else
        if(op == 2){
            ODT :: modify2(l, r, x);
        } else
        if(op == 3){
            cout << ODT :: query1(l, r, x) << "\n";
        } else
        if(op == 4){
            cout << ODT :: query2(l, r, x, y) << "\n";
        }
    }
    return 0;
}
```

2.3 可并堆

```
#include <bits/stdc++.h>
using namespace std;
using i64 = long long;
const int INF  =  1e9;
const i64 INFL = 1e18;
namespace LeftHeap{
    const int SIZ = 1e5 + 3;
    int W[SIZ], D[SIZ];
    int L[SIZ], R[SIZ];
    int F[SIZ], s;
    bool E[SIZ];
    int merge(int u, int v){
        if(u == 0 || v == 0)
            return u | v;
        if(W[u] > W[v] || (W[u] == W[v] && u > v))
            swap(u, v);
        int &lc = L[u];
        int &rc = R[u];
        rc = merge(rc, v);
        if(D[lc] < D[rc])
            swap(lc, rc);
        D[u] = min(D[lc], D[rc]) + 1;
        if(lc != 0) F[lc] = u;
        if(rc != 0) F[rc] = u;
        return u;
    }
    void pop(int &root){
        int root0 = merge(L[root], R[root]);
        F[root0] = root0;
```

```
30          F[root ] = root0;
31          E[root ] = true;
32          root = root0;
33      }
34      int top(int &root){
35          return W[root];
36      }
37      int getfa(int u){
38          return u == F[u] ? u : F[u] = getfa(F[u]);
39      }
40      int newnode(int w){
41          ++ s;
42          W[s] = w;
43          F[s] = s;
44          D[s] = 1;
45          return s;
46      }
47  }
48  // ==== TEST ====
49  int qread();
50  const int MAXN = 1e5 + 3;
51  int A[MAXN], O[MAXN];
52  int main(){
53      int n, m;
54      cin >> n >> m;
55      for(int i = 1;i <= n;++ i){
56          cin >> A[i];
57          O[i] = LeftHeap :: newnode(A[i]);
58      }
59      for(int i = 1;i <= m;++ i){
60          int op;
61          cin >> op;
62          if(op == 1){
63              int x, y;
64              cin >> x >> y;
65              if(LeftHeap :: E[O[x]])
66                  continue;
67              if(LeftHeap :: E[O[y]])
68                  continue;
69              int fx = LeftHeap :: getfa(O[x]);
70              int fy = LeftHeap :: getfa(O[y]);
71              if(fx != fy){
72                  LeftHeap :: merge(fx, fy);
73              }
74          } else {
75              int x;
76              cin >> x;
77              if(LeftHeap :: E[O[x]]){
78                  cout << -1 << endl;
79                  continue;
80              }
81              int fx = LeftHeap :: getfa(O[x]);
82              cout << LeftHeap :: top(fx) << endl;
83              LeftHeap :: pop(fx);
84          }
85      }
86      return 0;
```

```
}
```

2.4 线性基

```cpp
#include <bits/stdc++.h>
using namespace std;
using i64 = long long;
const int INF  =  1e9;
const i64 INFL = 1e18;
namespace LB{
    const int SIZ = 60 + 3;
    i64 W[SIZ], h = 60;
    void insert(i64 w){
        for(int i = h;i >= 0;-- i){
            if(w & (1ll << i)){
                if(!W[i]){
                    W[i] = w;
                    break;
                } else {
                    w ^= W[i];
                }
            }
        }
    }
    i64 query(i64 x){
        for(int i = h;i >= 0;-- i){
            if(W[i]){
                x = max(x, x ^ W[i]);
            }
        }
        return x;
    }
}
namespace realLB{
    const int SIZ = 500 + 3;
    long double W[SIZ][SIZ];
    int n = 0;
    void init(int n0){
        n = n0;
    }
    bool zero(long double w){
        return fabs(w) < 1e-9;
    }
    bool insert(long double X[]){
        for(int i = 1; i <= n;++ i){
            if(!zero(X[i])){
                if(zero(W[i][i])){
                    for(int j = 1;j <= n;++ j)
                        W[i][j] = X[j];
                    return true;
                } else {
                    long double t = X[i] / W[i][i];
                    for(int j = 1;j <= n;++ j)
                        X[j] -= t * W[i][j];
                }
            }
        }
```

```
        return false;
    }
}
// ==== TEST ====
int qread();
const int MAXN = 500 + 3;
long double X[MAXN][MAXN], C[MAXN];
int I[MAXN];
bool cmp(int a, int b){
    return C[a] < C[b];
}
int main(){
    int n, m;
    cin >> n >> m;
    realLB :: init(m);
    for(int i = 1;i <= n;++ i){
        for(int j = 1;j <= m;++ j){
            cin >> X[i][j];
        }
    }
    for(int i = 1;i <= n;++ i){
        cin >> C[i];
        I[i] = i;
    }
    sort(I + 1, I + 1 + n, cmp);
    int ans = 0, cnt = 0;
    for(int i = 1;i <= n;++ i){
        int x = I[i];
        if(realLB :: insert(X[x]))
            ans += C[x],
            cnt += 1;
    }
    cout << cnt << " " << ans << endl;
    return 0;
}
```

2.5 Link Cut 树

```
#include<bits/stdc++.h>
using namespace std;
typedef long long i64;
namespace LinkCutTree{
    const int SIZ = 1e5 + 3;
    int F[SIZ], C[SIZ], S[SIZ], W[SIZ], A[SIZ], X[SIZ][2], size;
    bool T[SIZ];
    bool is_root(int x){ return X[F[x]][0] != x && X[F[x]][1] != x;}
    bool is_rson(int x){ return X[F[x]][1] == x;}
    int  new_node(int w){
        ++ size;
        W[size] = w, C[size] = S[size] = 1;
        A[size] = w, F[size] = 0;
        X[size][0] = X[size][1] = 0;
        return size;
    }
    void push_up(int x){
        S[x] = C[x] + S[X[x][0]] + S[X[x][1]];
        A[x] = W[x] ^ A[X[x][0]] ^ A[X[x][1]];
```

```
    }
    void push_down(int x){
        if(!T[x]) return;
        int lc = X[x][0], rc = X[x][1];
        if(lc) T[lc] ^= 1, swap(X[lc][0], X[lc][1]);
        if(rc) T[rc] ^= 1, swap(X[rc][0], X[rc][1]);
        T[x] = false;
    }
    void update(int x){
        if(!is_root(x)) update(F[x]); push_down(x);
    }
    void rotate(int x){
        int y = F[x], z = F[y];
        bool f = is_rson(x);
        bool g = is_rson(y);
        if(is_root(y)){
            F[x] = z;
            F[y] = x;
            X[y][ f] = X[x][!f], F[X[x][!f]] = y;
            X[x][!f] = y;
        } else {
            F[x] = z;
            F[y] = x;
            X[z][ g] = x;
            X[y][ f] = X[x][!f], F[X[x][!f]] = y;
            X[x][!f] = y;
        }
        push_up(y), push_up(x);
    }
    void splay(int x){
        update(x);
        for(int f = F[x];f = F[x], !is_root(x);rotate(x))
            if(!is_root(f)) rotate(is_rson(x) == is_rson(f) ? f : x);
    }
    int  access(int x){
        int p;
        for(p = 0;x;p = x, x = F[x]){
            splay(x), X[x][1] = p, push_up(x);
        }
        return p;
    }
    void make_root(int x){
        x = access(x);
        T[x] ^= 1, swap(X[x][0], X[x][1]);
    }
    int find_root(int x){
        access(x), splay(x), push_down(x);
        while(X[x][0]) x = X[x][0], push_down(x);
        splay(x);
        return x;
    }
    void link(int x, int y){
        make_root(x), splay(x), F[x] = y;
    }
    void cut(int x, int p){
        make_root(x), access(p), splay(p), X[p][0] = F[x] = 0;
    }
```

```
    void modify(int x, int w){
        splay(x), W[x] = w, push_up(x);
    }
}
const int MAXN = 1e5 + 3;
map<pair<int, int>, bool> M;
int n, m;
int main(){
    cin >> n >> m;
    for(int i = 1;i <= n;++ i){
        int a; cin >> a;
        LinkCutTree :: new_node(a);
    }
    for(int i = 1;i <= m;++ i){
        int o; cin >> o;
        if(o == 0){
            int u, v; cin >> u >> v;
            LinkCutTree :: make_root(u);
            int p = LinkCutTree :: access(v);
            printf("%d\n", LinkCutTree :: A[p]);
        } else if(o == 1){
            int u, v; cin >> u >> v;
            int a = LinkCutTree :: find_root(u);
            int b = LinkCutTree :: find_root(v);
            if(a != b){
                LinkCutTree :: link(u, v);
                M[make_pair(min(u, v), max(u, v))] = true;
            }
        } else if(o == 2){
            int u, v; cin >> u >> v;
            if(M.count(make_pair(min(u, v), max(u, v)))){
                M.erase(make_pair(min(u, v), max(u, v)));
                LinkCutTree :: cut(u, v);
            }
        } else {
            int u, w; cin >> u >> w;
            LinkCutTree :: modify(u, w);
        }
    }
    return 0;
}
```

2.6 线段树

2.6.1 李超树

```
#include<bits/stdc++.h>
using namespace std;
typedef long long i64;
struct Line{ int id; double k, b; Line() = default;};
namespace LCSeg{
    const int SIZ = 2e5 + 3;
    struct Line T[SIZ];
    #define lc(t) (t << 1)
    #define rc(t) (t << 1 | 1)
    bool cmp(int p, Line x, Line y){
        double w1 = x.k * p + x.b;
```

```
        double w2 = y.k * p + y.b;
        double d = w1 - w2;
        if(fabs(d) < 1e-8) return x.id > y.id;
        return d < 0;
    }
    void merge(int t, int a, int b, Line x, Line y){
        int c = a + b >> 1;
        if(cmp(c, x, y)) swap(x, y);
        if(cmp(a, y, x)){
            T[t] = x; if(a ≠ b) merge(rc(t), c + 1, b, T[rc(t)], y);
        } else {
            T[t] = x; if(a ≠ b) merge(lc(t), a,     c, T[lc(t)], y);
        }
    }
    void modify(int t, int a, int b, int l, int r, Line x){
        if(l ⩽ a && b ⩽ r) merge(t, a, b, T[t], x);
        else {
            int c = a + b >> 1;
            if(l ⩽ c) modify(lc(t), a,     c, l, r, x);
            if(r >  c) modify(rc(t), c + 1, b, l, r, x);
        }
    }
    void query(int t, int a, int b, int p, Line &x){
        if(cmp(p, x, T[t])) x = T[t];
        if(a ≠ b){
            int c = a + b >> 1;
            if(p ⩽ c) query(lc(t), a,     c, p, x);
            if(p >  c) query(rc(t), c + 1, b, p, x);
        }
    }
}
const int MOD1 = 39989;
const int MOD2 = 1e9;
int qread();
int m = 39989, o;
int main(){
    int n = qread(), last_ans = 0;
    for(int i = 1;i ⩽ n;++ i){
        int op = qread(); if(op == 0){
            int k = (qread() + last_ans - 1) % MOD1 + 1;
            Line x = {0, 0, 0}; LCSeg :: query(1, 1, m, k, x);
            printf("%d\n", last_ans = x.id);
        } else {
            int _x1 = (qread() + last_ans - 1) % MOD1 + 1;
            int _y1 = (qread() + last_ans - 1) % MOD2 + 1;
            int _x2 = (qread() + last_ans - 1) % MOD1 + 1;
            int _y2 = (qread() + last_ans - 1) % MOD2 + 1;
            if(_x1 > _x2) swap(_x1, _x2), swap(_y1, _y2);
            double k, b; int d = ++ o;
            if(_x1 == _x2) k = 0, b = max(_y1, _y2);
                else k = 1.0 * (_y2 - _y1) / (_x2 - _x1), b = _y1 - k *
                    _x1;
            Line x = {d, k, b}; LCSeg :: modify(1, 1, m, _x1, _x2, x);
        }
    }
    return 0;
}
```

2.6.2 线段树 3

```cpp
#include<bits/stdc++.h>
using namespace std;
typedef long long i64;
const int INF = 2e9;
const int MAXN= 5e5 + 3;
int A[MAXN];
struct Node{
    i64 sum; int len, max1, max2, max_cnt, his_mx;
    Node():
        sum(0), max1(-INF), max2(-INF), max_cnt(0), his_mx(-INF), len(0) {}
    Node(int w):
        sum(w), max1(   w), max2(-INF), max_cnt(1), his_mx(   w), len(1) {}
    bool update(int w1, int w2, int h1, int h2){
        his_mx = max({his_mx, max1 + h1});
        max1 += w1, max2 += w2;
        sum += 1ll * w1 * max_cnt + 1ll * w2 * (len - max_cnt);
        return max1 > max2;
    }
};
struct Tag{
    int max_add, max_his_add, umx_add, umx_his_add; bool have;
    void update(int w1, int w2, int h1, int h2){
        max_his_add = max(max_his_add, max_add + h1);
        umx_his_add = max(umx_his_add, umx_add + h2);
        max_add += w1, umx_add += w2, have = true;
    }
    void clear(){
        max_add = max_his_add = umx_add = umx_his_add = have = 0;
    }
};
struct Node operator +(Node a, Node b){
    Node t;
    t.max1 = max(a.max1, b.max1);
    if(t.max1 != a.max1){
        if(a.max1 > t.max2) t.max2 = a.max1;
    } else{
        if(a.max2 > t.max2) t.max2 = a.max2;
        t.max_cnt += a.max_cnt;
    }
    if(t.max1 != b.max1){
        if(b.max1 > t.max2) t.max2 = b.max1;
    } else{
        if(b.max2 > t.max2) t.max2 = b.max2;
        t.max_cnt += b.max_cnt;
    }
    t.sum = a.sum + b.sum, t.len = a.len + b.len;
    t.his_mx = max(a.his_mx, b.his_mx);
    return t;
}
namespace Seg{
    const int SIZ = 2e6 + 3;
    struct Node W[SIZ]; struct Tag T[SIZ];
    #define lc(t) (t << 1)
    #define rc(t) (t << 1 | 1)
```

```
55      void push_up(int t, int a, int b){
56          W[t] = W[lc(t)] + W[rc(t)];
57      }
58      void push_down(int t, int a, int b){
59          if(a == b) T[t].clear();
60          if(T[t].have){
61              int c = a + b >> 1, x = lc(t), y = rc(t);
62              int w = max(W[x].max1, W[y].max1);
63              int w1 = T[t].max_add, w2 = T[t].umx_add, w3 = T[t].
                    max_his_add, w4 = T[t].umx_his_add;
64              if(w == W[x].max1)
65                  W[x].update(w1, w2, w3, w4),
66                  T[x].update(w1, w2, w3, w4);
67              else
68                  W[x].update(w2, w2, w4, w4),
69                  T[x].update(w2, w2, w4, w4);
70              if(w == W[y].max1)
71                  W[y].update(w1, w2, w3, w4),
72                  T[y].update(w1, w2, w3, w4);
73              else
74                  W[y].update(w2, w2, w4, w4),
75                  T[y].update(w2, w2, w4, w4);
76              T[t].clear();
77          }
78      }
79      void build(int t, int a, int b){
80          if(a == b){W[t] = Node(A[a]), T[t].clear();} else {
81              int c = a + b >> 1; T[t].clear();
82              build(lc(t), a,     c);
83              build(rc(t), c + 1, b);
84              push_up(t, a, b);
85          }
86      }
87      void modiadd(int t, int a, int b, int l, int r, int w){
88          if(l <= a && b <= r){
89              T[t].update(w, w, w, w);
90              W[t].update(w, w, w, w);
91          } else {
92              int c = a + b >> 1; push_down(t, a, b);
93              if(l <= c) modiadd(lc(t), a,     c, l, r, w);
94              if(r >  c) modiadd(rc(t), c + 1, b, l, r, w);
95              push_up(t, a, b);
96          }
97      }
98      void modimin(int t, int a, int b, int l, int r, int w){
99          if(l <= a && b <= r){
100             if(w >= W[t].max1) return; else
101             if(w >  W[t].max2){
102                 int k = w - W[t].max1;
103                 T[t].update(k, 0, k, 0);
104                 W[t].update(k, 0, k, 0);
105             } else {
106                 int c = a + b >> 1;
107                 push_down(t, a, b);
108                 modimin(lc(t), a,     c, l, r, w);
109                 modimin(rc(t), c + 1, b, l, r, w);
110                 push_up(t, a, b);
```

```
            }
        } else {
            int c = a + b >> 1; push_down(t, a, b);
            if(l ⩽ c) modimin(lc(t), a,     c, l, r, w);
            if(r >  c) modimin(rc(t), c + 1, b, l, r, w);
            push_up(t, a, b);
        }
    }
    Node query(int t, int a, int b, int l, int r){
        if(l ⩽ a && b ⩽ r) return W[t];
        int c = a + b >> 1; Node ret; push_down(t, a, b);
        if(l ⩽ c) ret = ret + query(lc(t), a,     c, l, r);
        if(r >  c) ret = ret + query(rc(t), c + 1, b, l, r);
        return ret;
    }
}
int qread();
int main(){
    int n = qread(), m = qread();
    for(int i = 1;i ⩽ n;++ i)
        A[i] = qread();
    Seg :: build(1, 1, n);
    for(int i = 1;i ⩽ m;++ i){
        int op = qread();
        if(op == 1){
            int l = qread(), r = qread(), w = qread();
            Seg :: modiadd(1, 1, n, l, r, w);
        } else if(op == 2){
            int l = qread(), r = qread(), w = qread();
            Seg :: modimin(1, 1, n, l, r, w);
        } else if(op == 3){
            int l = qread(), r = qread();
            auto p = Seg :: query(1, 1, n, l, r);
            printf("%lld\n", p.sum);
        } else if(op == 4){
            int l = qread(), r = qread();
            auto p = Seg :: query(1, 1, n, l, r);
            printf("%d\n", p.max1);
        } else if(op == 5){
            int l = qread(), r = qread();
            auto p = Seg :: query(1, 1, n, l, r);
            printf("%d\n", p.his_mx);
        }
    }
    return 0;
}
```

2.6.3 扫描线

```
#include<bits/stdc++.h>
using namespace std;
using i64 = long long;
const int INF  =  1e9;
const i64 INFL = 1e18;
int qread(){
    int w = 1, c, ret;
```

```
    while((c = getchar()) > '9' || c < '0') w = (c == '-' ? -1 : 1);
       ret = c - '0';
    while((c = getchar()) >= '0' && c <= '9') ret = ret * 10 + c - '0';
    return ret * w;
}
const int MAXN = 1e5 + 3;
int X1[MAXN], Y1[MAXN];
int X2[MAXN], Y2[MAXN];
int n, h, H[MAXN * 2];
namespace Seg{
    #define lc(t) (t << 1)
    #define rc(t) (t << 1 | 1)
    const int SIZ = 8e5 + 3;
    int T[SIZ], S[SIZ], L[SIZ];
    void pushup(int t, int a, int b){
        S[t] = 0;
        if(a != b){
            S[t] = S[lc(t)] + S[rc(t)];
            L[t] = L[lc(t)] + L[rc(t)];
        }
        if(T[t]) S[t] = L[t];
    }
    void modify(int t, int a, int b, int l, int r, int w){
        if(l <= a && b <= r){
            T[t] += w, pushup(t, a, b);
        } else {
            int c = a + b >> 1;
            if(l <= c) modify(lc(t), a, c, l, r, w);
            if(r > c) modify(rc(t), c + 1, b, l, r, w);
            pushup(t, a, b);
        }
    }
    void build(int t, int a, int b){
        if(a == b){
            L[t] = H[a] - H[a - 1];
        } else {
            int c = a + b >> 1;
            build(lc(t), a, c);
            build(rc(t), c + 1, b);
            pushup(t, a, b);
        }
    }
    int query(int t){
        return S[t];
    }
}
tuple <int, int, int> P[MAXN];
tuple <int, int, int> Q[MAXN];
int main(){
    n = qread();
    for(int i = 1;i <= n;++ i){
        X1[i] = qread(), Y1[i] = qread();
        X2[i] = qread(), Y2[i] = qread();
        if(X1[i] > X2[i]) swap(X1[i], X2[i]);
        if(Y1[i] > Y2[i]) swap(Y1[i], Y2[i]);
        H[++ h] = Y1[i];
        H[++ h] = Y2[i];
```

```
        P[i] = make_tuple(X1[i], Y1[i], Y2[i]);
        Q[i] = make_tuple(X2[i], Y1[i], Y2[i]);
    }
    sort(H + 1, H + 1 + h);
    sort(P + 1, P + 1 + n);
    sort(Q + 1, Q + 1 + n);
    int o = unique(H + 1, H + 1 + h) - H - 1;
    Seg :: build(1, 1, o);
    i64 ans = 0, last = -1;
    int p = 1, q = 1;
    while(p ⩽ n || q ⩽ n){
        int x = INF;
        if(p ⩽ n) x = min(x, get<0>(P[p]));
        if(q ⩽ n) x = min(x, get<0>(Q[q]));
        if(last ≠ -1){
            ans += 1ll * Seg :: query(1) * (x - last);
        }
        last = x;
        while(q ⩽ n && get<0>(Q[q]) == x){
            auto [x, l, r] = Q[q]; ++ q;
            l = lower_bound(H + 1, H + 1 + o, l) - H + 1;
            r = lower_bound(H + 1, H + 1 + o, r) - H;
            Seg :: modify(1, 1, o, l, r, 1);
        }
        while(p ⩽ n && get<0>(P[p]) == x){
            auto [x, l, r] = P[p]; ++ p;
            l = lower_bound(H + 1, H + 1 + o, l) - H + 1;
            r = lower_bound(H + 1, H + 1 + o, r) - H;
            Seg :: modify(1, 1, o, l, r, -1);
        }
    }
    printf("%lld\n", ans);
    return 0;
}
```

2.7 根号数据结构

2.7.1 块状链表

```
#include <bits/stdc++.h>
using namespace std;
using i64 = long long;
const int INF  =  1e9;
const i64 INFL = 1e18;
namespace BLOCK{
    const int SIZ = 1e6 + 1e5 + 3;
    const int BSZ = 2000;
    list <vector<int> > block;
    void build(int n, const int A[]){
        for(int l = 0, r = 0;r ≠ n;){
            l = r;
            r = min(l + BSZ / 2, n);
            vector <int> V0(A + l, A + r);
            block.emplace_back(V0);
        }
    }
    int get_kth(int k){
```

```
19          for(auto it = block.begin();it != block.end();++ it){
20              if(it -> size() < k)
21                  k -= it -> size();
22              else return it -> at(k - 1);
23          }
24          return -1;
25      }
26      int get_rank(int w){
27          int ans = 0;
28          for(auto it = block.begin();it != block.end();++ it){
29              if(it -> back() < w)
30                  ans += it -> size();
31              else {
32                  ans += lower_bound(it -> begin(), it -> end(), w) - it
                        -> begin();
33                  break;
34              }
35          }
36          return ans + 1;
37      }
38      // 插入到第 k 个位置
39      void insert(int k, int w){
40          for(auto it = block.begin();it != block.end();++ it){
41              if(it -> size() < k)
42                  k -= it -> size();
43              else{
44                  it -> insert(it -> begin() + k - 1, w);
45                  if(it -> size() > BSZ){
46                      vector <int> V1(it -> begin(), it -> begin() + BSZ
                            / 2);
47                      vector <int> V2(it -> begin() + BSZ / 2, it -> end
                            ());
48                      *it = V2;
49                      block.insert(it, V1);
50                  }
51                  return;
52              }
53          }
54      }
55      // 删除第 k 个数
56      void erase(int k){
57          for(auto it = block.begin();it != block.end();++ it){
58              if(it -> size() < k)
59                  k -= it -> size();
60              else{
61                  it -> erase(it -> begin() + k - 1);
62                  if(it -> empty())
63                      block.erase(it);
64                  return;
65              }
66          }
67      }
68  }
69  int qread();
70  const int MAXN = 1e5 + 3;
71  int A[MAXN];
72  // ==== TEST ====
```

```cpp
int main(){
    ios :: sync_with_stdio(false);
    cin.tie(nullptr);
    int n, m;
    cin >> n >> m;
    for(int i = 1;i ⩽ n;++ i)
        cin >> A[i];
    sort(A + 1, A + 1 + n);
    A[n + 1] = INT_MAX;
    BLOCK :: build(n + 1, A + 1);
    int last = 0;
    int ans = 0;
    for(int i = 1;i ⩽ m;++ i){
        int op;
        cin >> op;
        if(op == 1){
            int x; cin >> x; x ^= last;
            int k = BLOCK :: get_rank(x);
            BLOCK :: insert(k, x);
        } else
        if(op == 2){
            int x; cin >> x; x ^= last;
            int k = BLOCK :: get_rank(x);
            BLOCK :: erase(k);
        } else
        if(op == 3){
            int x; cin >> x; x ^= last;
            int k = BLOCK :: get_rank(x);
            last = k, ans ^= last;
        } else
        if(op == 4){
            int x; cin >> x; x ^= last;
            int k = BLOCK :: get_kth (x);
            last = k, ans ^= last;
        } else
        if(op == 5){
            int x; cin >> x; x ^= last;
            int k = BLOCK :: get_rank(x);
            last = BLOCK :: get_kth (k - 1), ans ^= last;
        } else
        if(op == 6){
            int x; cin >> x; x ^= last;
            int k = BLOCK :: get_rank(x + 1);
            last = BLOCK :: get_kth (k), ans ^= last;
        }
    }
    cout << ans << endl;
    return 0;
}
```

2.7.2 莫队二次离线

```cpp
#include<bits/stdc++.h>
#define up(l, r, i) for(int i = l, END##i = r;i ⩽ END##i;++ i)
#define dn(r, l, i) for(int i = r, END##i = l;i ⩾ END##i;-- i)
using namespace std;
typedef long long i64;
```

```
const int INF = 2147483647;
const int MAXN= 1e5 + 3;
const int MAXM= (1 << 14) + 3;
int n, m, k, maxt = 16383, X[MAXM], C[MAXM], t;
const int BUF_SIZE = 1e6;
char *p1, *p2, BUF[BUF_SIZE];
inline char readc();
inline int qread();
int A[MAXN], bsize; i64 B[MAXN], R[MAXN];
struct Qry1{ int l, r, id; }O[MAXN];
struct Qry2{ int id, l, r; };
struct Qry3{ int id, l, r; };
bool cmp(Qry1 a, Qry1 b){
    return a.l / bsize == b.l / bsize ? a.r < b.r : a.l < b.l;
}
vector <Qry2> P[MAXN];
vector <Qry3> Q[MAXN];
int main(){
    n = qread(), m = qread(), k = qread(), bsize = sqrt(m + 1);
    up(1, n, i) A[i] = qread();
    up(1, m, i){
        int l = qread(), r = qread(); O[i] = {l, r, i};
    }
    sort(O + 1, O + 1 + m, cmp);
    int l = 1, r = 0;
    up(1, m, i){
        int p = O[i].l, q = O[i].r;
        if(r < q){
            P[r    ].push_back({ i, r + 1, q});
            Q[l - 1].push_back({-i, r + 1, q});
        }
        if(r > q){
            P[q    ].push_back({-i, q + 1, r});
            Q[l - 1].push_back({ i, q + 1, r});
        }
        r = q;
        if(l > p){
            P[p].push_back({-i, p, l - 1});
            Q[r].push_back({ i, p, l - 1});
        }
        if(l < p){
            P[l].push_back({ i, l, p - 1});
            Q[r].push_back({-i, l, p - 1});
        }
        l = p;
    }
    up(0, maxt, i) if(__builtin_popcount(i) == k) X[++ t] = i;
    up(0, n, i){
        up(1, t, j) ++ C[A[i] ^ X[j]];
        for(auto &o : P[i]){
            if(o.id > 0) R[ o.id] += C[A[o.l]];
            else         R[-o.id] -= C[A[o.l]];
            if(o.l < o.r)
                P[i + 1].push_back({o.id, o.l + 1, o.r});
        }
        for(auto &o : Q[i]){
            up(o.l, o.r, j){
```

```
                if(o.id > 0) R[ o.id] += C[A[j]];
                else         R[-o.id] -= C[A[j]];
            }
        }
        P[i].clear(), Q[i].clear();
        P[i].shrink_to_fit();
        Q[i].shrink_to_fit();
    }
    i64 ans = 0;
    up(1, m, i){ ans += R[i], B[O[i].id] = ans; }
    up(1, m, i) printf("%lld\n", B[i]);
    return 0;
}
```

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 x k` 查询距离 x 不超过 k 的所有点的点权之和;
- `0 x y` 将点 x 的点权修改为 y。

```
#include<bits/stdc++.h>
#define endl "\n"
using namespace std;
const int MAXN = 1e5 + 3;
vector<int> E[MAXN];
namespace LCA{
    const int SIZ = 1e5 + 3;
    int D[SIZ], F[SIZ];
    int P[SIZ], Q[SIZ], o;
    void dfs(int u, int f){
        P[u] = ++ o;
        Q[o] = u;
        F[u] = f;
        D[u] = D[f] + 1;
        for(auto &v : E[u]) if(v != f){
            dfs(v, u);
        }
    }
    const int MAXH = 18 + 3;
    int h = 18;
    int ST[SIZ][MAXH];
    int cmp(int a, int b){
        return D[a] < D[b] ? a : b;
    }
    int T[SIZ], n;
    void init(int _n){
        n = _n;
        dfs(1, 0);
        for(int i = 1;i <= n;++ i)
            ST[i][0] = Q[i];
        for(int i = 2;i <= n;++ i)
```

```
32              T[i] = T[i >> 1] + 1;
33          for(int i = 1;i <= h;++ i){
34              for(int j = 1;j <= n;++ j) if(j + (1 << i - 1) <= n){
35                  ST[j][i] = cmp(ST[j][i - 1], ST[j + (1 << i - 1)][i -
                        1]);
36              }
37          }
38      }
39      int lca(int a, int b){
40          if(a == b)
41              return a;
42          int l = P[a];
43          int r = P[b];
44          if(l > r)
45              swap(l, r);
46          ++ l;
47          int d = T[r - l + 1];
48          return F[cmp(ST[l][d], ST[r - (1 << d) + 1][d])];
49      }
50      int dis(int a, int b){
51          return D[a] + D[b] - 2 * D[lca(a, b)];
52      }
53  }
54  namespace BIT{
55      void modify(int D[], int n, int p, int w){
56          ++ p;
57          while(p <= n)
58              D[p] += w, p += p & -p;
59      }
60      int query(int D[], int n, int p){
61          if(p < 0) return 0;
62          p = min(n, p + 1);
63          int r = 0;
64          while(p > 0)
65              r += D[p], p -= p & -p;
66          return r;
67      }
68  }
69  namespace PTree{
70      const int SIZ = 1e5 + 3;
71      bool V[SIZ];
72      int  S[SIZ], L[SIZ];
73      vector<int> EE[MAXN];
74      int *D1[MAXN];
75      int *D2[MAXN];
76      void dfs1(int s, int &g, int u, int f){
77          S[u] = 1;
78          int maxsize = 0;
79          for(auto &v : E[u]) if(v != f && !V[v]){
80              dfs1(s, g, v, u);
81              if(S[v] > maxsize)
82                  maxsize = S[v];
83              S[u] += S[v];
84          }
85          maxsize = max(maxsize, s - S[u]);
86          if(maxsize <= s / 2)
87              g = u;
```

```
88      }
89      int n;
90      void build(int s, int &g, int u, int f){
91          dfs1(s, g, u, f);
92          V[g] = true, L[g] = s;
93          for(auto &u : E[g]) if(!V[u]){
94              int h = 0;
95              if(S[u] < S[g]) build(S[u], h, u, 0);
96              else            build(s - S[g], h, u, 0);
97              EE[g].push_back(h);
98              EE[h].push_back(g);
99          }
100     }
101     int F[SIZ];
102     void dfs2(int u, int f){
103         F[u] = f;
104         for(auto &v : EE[u]) if(v ≠ f){
105             dfs2(v, u);
106         }
107     }
108     void build(int _n){
109         n = _n;
110         int s = n, g = 0;
111         dfs1(s, g, 1, 0);
112         V[g] = true, L[g] = s;
113         for(auto &u : E[g]){
114             int h = 0;
115             if(S[u] < S[g]) build(S[u], h, u, 0);
116             else            build(s - S[g], h, u, 0);
117             EE[g].push_back(h);
118             EE[h].push_back(g);
119         }
120         dfs2(g, 0);
121         for(int i = 1;i ⩽ n;++ i){
122             L[i] += 2;
123             D1[i] = new int[L[i] + 3];
124             D2[i] = new int[L[i] + 3];
125             for(int j = 0;j < L[i] + 3;++ j)
126                 D1[i][j] = D2[i][j] = 0;
127         }
128     }
129     void modify(int x, int w){
130         int u = x;
131         while(1){
132             BIT :: modify(D1[x], L[x], LCA :: dis(u, x), w);
133             int y = F[x];
134             if(y ≠ 0){
135                 int e = LCA :: dis(x, y);
136                 BIT :: modify(D2[x], L[x], LCA :: dis(u, y), w);
137                 x = y;
138             } else break;
139         }
140     }
141     int query(int x, int d){
142         int ans = 0, u = x;
143         while(1){
144             ans += BIT :: query(D1[x], L[x], d - LCA :: dis(u, x));
```

```
            int y = F[x];
            if(y ≠ 0){
                int e = LCA :: dis(x, y);
                ans -= BIT :: query(D2[x], L[x], d - LCA :: dis(u, y));
                x = y;
            } else break;
        }
        return ans;
    }
}
int W[MAXN];
int main(){
    ios :: sync_with_stdio(false);
    int n, m;
    cin >> n >> m;
    for(int i = 1;i ⩽ n;++ i){
        cin >> W[i];
    }
    for(int i = 2;i ⩽ n;++ i){
        int u, v;
        cin >> u >> v;
        E[u].push_back(v);
        E[v].push_back(u);
    }
    LCA :: init(n);
    PTree :: build(n);
    for(int i = 1;i ⩽ n;++ i)
        PTree :: modify(i, W[i]);
    int lastans = 0;
    for(int i = 1;i ⩽ m;++ i){
        int op; cin >> op;
        if(op == 0){
            int x, d;
            cin >> x >> d;
            x ^= lastans;
            d ^= lastans;
            cout << (lastans = PTree :: query(x, d)) << endl;
        } else {
            int x, w;
            cin >> x >> w;
            x ^= lastans;
            w ^= lastans;
            PTree :: modify(x, -W[x]    );
            PTree :: modify(x,  W[x] = w);
        }
    }
    return 0;
}
```

3.2 长链剖分

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

```
const int MAXM=  19 + 3;
vector <int> P[MAXN];
vector <int> Q[MAXN];
vector <int> E[MAXN];
int h = 19;
int L[MAXN], F[MAXN], G[MAXN], D[MAXN], S[MAXM][MAXN];
void dfs1(int u, int f){
    L[u] = 1, S[0][u] = f;
    F[u] = f, D[u] = D[f] + 1;
    for(int i = 1;i <= h;++ i)
        S[i][u] = S[i - 1][S[i - 1][u]];
    for(auto &v : E[u]) if(v != f){
        dfs1(v, u);
        if(L[v] > L[G[u]])
            G[u] = v;
        L[u] = max(L[u], L[v] + 1);
    }
}
int T[MAXN];
void dfs2(int u, int f){
    if(u == G[f]){
        T[u] = T[f];
        P[T[u]].push_back(u);
        Q[T[u]].push_back(F[Q[T[u]].back()]);
    } else {
        T[u] = u;
        P[u].push_back(u);
        Q[u].push_back(u);
    }
    if(G[u]) dfs2(G[u], u);
    for(auto &v : E[u]) if(v != f && v != G[u])
        dfs2(v, u);
}
typedef unsigned int       u32;
typedef unsigned long long u64;
int n, q; u32 s;
u32 get(u32 x) {
    x ^= x << 13;
    x ^= x >> 17;
    x ^= x << 5;
    return s = x;
}
int qread();
int H[MAXN];
int main(){
    scanf("%d%d%u", &n, &q, &s);
    int root = 0; H[0] = -1;
    for(int i = 1;i <= n;++ i){
        int f = qread();
        if(f == 0)
            root = i;
        else {
            E[f].push_back(i);
            E[i].push_back(f);
        }
        H[i] = H[i >> 1] + 1;
    }
```

```
    dfs1(root, 0);
    dfs2(root, 0);
    int lastans = 0;
    i64 realans = 0;
    for(int i = 1;i ⩽ q;++ i){
        int x = (get(s) ^ lastans) % n + 1;
        int k = (get(s) ^ lastans) % D[x];
        if(k == 0){
            lastans = x;
        } else {
            int h = H[k];
            k -= 1 << h;
            x = S[h][x];
            int t = T[x];
            k -= D[x] - D[t];
            if(k > 0){
                x = Q[t][k];
            } else {
                x = P[t][-k];
            }
            lastans = x;
        }
        realans ^= 1ll * i * lastans;
    }
    printf("%lld\n", realans);
    return 0;
}
```

3.3 重链剖分

```
#include<bits/stdc++.h>
using namespace std;
using i64 = long long;
const int INF  =  1e9;
const i64 INFL = 1e18;
const int MAXN= 1e5 + 3;
int MOD;
int n, m, root;
int A[MAXN];
int qread();
vector <int> E[MAXN];
int S[MAXN], G[MAXN], D[MAXN], F[MAXN];
void dfs1(int u, int f){
    S[u] = 1, G[u] = 0, D[u] = D[f] + 1, F[u] = f;
    for(auto &v : E[u]) if(v ≠ f){
        dfs1(v, u);
        S[u] += S[v];
        if(S[v] > S[G[u]])
            G[u] = v;
    }
}
int B[MAXN];
int P[MAXN], Q[MAXN], T[MAXN], L[MAXN], R[MAXN], cnt;
void dfs2(int u, int f){
    P[++ cnt] = u, B[cnt] = A[u], Q[u] = cnt;
    L[u] = cnt;
    if(u ≠ G[f]) T[u] = u;
```

```
28          else      T[u] = T[f];
29      if(G[u]) dfs2(G[u], u);
30      for(auto &v : E[u]) if(v != f && v != G[u]){
31          dfs2(v, u);
32      }
33      R[u] = cnt;
34  }
35  namespace Seg{
36      #define lc(t) (t << 1)
37      #define rc(t) (t << 1 | 1)
38      const int SIZ = 4e5 + 3;
39      i64 S[SIZ], T[SIZ];
40      void pushup(int t, int a, int b){
41          S[t] = (S[lc(t)] + S[rc(t)]) % MOD;
42      }
43      void pushdown(int t, int a, int b){
44          if(T[t]){
45              int c = a + b >> 1;
46              T[lc(t)] = (T[lc(t)] + T[t]) % MOD;
47              T[rc(t)] = (T[rc(t)] + T[t]) % MOD;
48              S[lc(t)] = (S[lc(t)] + 1ull * (c - a + 1) * T[t]) % MOD;
49              S[rc(t)] = (S[rc(t)] + 1ull * (b - c    ) * T[t]) % MOD;
50              T[t] = 0;
51          }
52      }
53      void modify(int t, int a, int b, int l, int r, int w){
54          if(l <= a && b <= r){
55              S[t] = (S[t] + 1ll * w * (b - a + 1)) % MOD;
56              T[t] = (T[t] + w) % MOD;
57          } else {
58              int c = a + b >> 1;
59              pushdown(t, a, b);
60              if(l <= c) modify(lc(t), a, c, l, r, w);
61              if(r >  c) modify(rc(t), c + 1, b, l, r, w);
62              pushup(t, a, b);
63          }
64      }
65      i64 query(int t, int a, int b, int l, int r){
66          if(l <= a && b <= r)
67              return S[t];
68          int c = a + b >> 1;
69          i64 ans = 0;
70          pushdown(t, a, b);
71          if(l <= c) ans = (ans + query(lc(t), a, c, l, r)) % MOD;
72          if(r >  c) ans = (ans + query(rc(t), c + 1, b, l, r)) % MOD;
73          return ans;
74      }
75      void build(int t, int a, int b){
76          if(a == b){
77              S[t] = B[a] % MOD;
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      }
```

```
}
int main(){
    n = qread(), m = qread(), root = qread(), MOD = qread();
    for(int i = 1;i ⩽ n;++ i)
        A[i] = qread();
    for(int i = 2;i ⩽ n;++ i){
        int u = qread(), v = qread();
        E[u].push_back(v);
        E[v].push_back(u);
    }
    dfs1(root, 0);
    dfs2(root, 0);
    Seg :: build(1, 1, n);
    for(int i = 1;i ⩽ m;++ i){
        int op = qread();
        if(op == 1){
            int u = qread(), v = qread(), k = qread();
            while(T[u] ≠ T[v]){
                if(D[T[u]] < D[T[v]])
                    swap(u, v);
                Seg :: modify(1, 1, n, Q[T[u]], Q[u], k);
                u = F[T[u]];
            }
            if(D[u] < D[v]) swap(u, v);
            Seg :: modify(1, 1, n, Q[v], Q[u], k);
        } else if(op == 2){
            int u = qread(), v = qread();
            i64 ans = 0;
            while(T[u] ≠ T[v]){
                if(D[T[u]] < D[T[v]])
                    swap(u, v);
                ans = (ans + Seg :: query(1, 1, n, Q[T[u]], Q[u])) %
                   MOD;
                u = F[T[u]];
            }
            if(D[u] < D[v]) swap(u, v);
            ans = (ans + Seg :: query(1, 1, n, Q[v], Q[u])) % MOD;
            printf("%lld\n", ans);
        } else if(op == 3){
            int x = qread(), w = qread();
            Seg :: modify(1, 1, n, L[x], R[x], w);
        } else {
            int x = qread();
            printf("%lld\n", Seg :: query(1, 1, n, L[x], R[x]));
        }
    }
    return 0;
}
```

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。

```
#include<bits/stdc++.h>
using namespace std;
using u64 = unsigned long long;
```

```cpp
const int MAXN = 1e6 + 3;
u64 xor_shift(u64 x){
    x ^= x << 13;
    x ^= x >> 7;
    x ^= x << 17;
    return x;
}
u64 H[MAXN];
vector <int> E[MAXN];
void dfs(int u, int f){
    H[u] = 1;
    for(auto &v: E[u]) if(v != f){
        dfs(v, u);
        H[u] += H[v];
    }
    H[u] = xor_shift(H[u]); // !important
}
int main(){
    int n;
    cin >> n;
    for(int i = 2;i <= n;++ i){
        int u, v;
        cin >> u >> v;
        E[u].push_back(v);
        E[v].push_back(u);
    }
    dfs(1, 0);
    sort(H + 1, H + 1 + n);
    cout << (unique(H + 1, H + 1 + n) - H - 1) << endl;
    return 0;
}
```

3.5 Prufer 序列

3.5.1 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。

```cpp
#include<bits/stdc++.h>
using namespace std;
const int MAXN = 5e6 + 3;
int D[MAXN], F[MAXN], P[MAXN];
vector<int> tree2prufer(int n){
    vector <int> P(n);
    for(int i = 1, j = 1;i <= n - 2;++ i, ++ j){
        while(D[j]) ++ j;
        P[i] = F[j];
        while(i <= n - 2 && !--D[P[i]] && P[i] < j)
            P[i + 1] = F[P[i]], i ++;
    }
    return P;
}
vector<int> prufer2tree(int n){
    vector <int> F(n);
    for(int i = 1, j = 1;i <= n - 1;++ i, ++ j){
        while(D[j]) ++ j;
        F[j] = P[i];
        while(i <= n - 1 && !--D[P[i]] && P[i] < j)
```

```
            F[P[i]] = P[i + 1], i ++;
    }
    return F;
}
int main(){
    ios :: sync_with_stdio(false);
    cin.tie(nullptr);
    int n, m;
    cin >> n >> m;
    vector <int> ANS;
    if(m == 1){     // tree → prufer
        for(int i = 1;i <= n - 1;++ i){
            cin >> F[i], D[F[i]] ++;
        }
        ANS = tree2prufer(n);
    } else {        // prufer → tree
        for(int i = 1;i <= n - 2;++ i){
            cin >> P[i], D[P[i]] ++;
        }
        P[n - 1] = n;
        ANS = prufer2tree(n);
    }
    long long ans = 0, cnt = 0;
    for(int i = 1;i <= n - (m == 1 ? 2 : 1);++ i)
        ans ^= 1ll * ANS[i] * i;
    cout << ans << "\n";
    return 0;
}
```

3.6 虚树

```
#include<bits/stdc++.h>
using namespace std;
const int MAXN = 5e5 + 3;
vector<pair<int, int> > E[MAXN];
namespace LCA{
    const int SIZ = 5e5 + 3;
    int D[SIZ], H[SIZ], F[SIZ];
    int P[SIZ], Q[SIZ], o;
    void dfs(int u, int f){
        P[u] = ++ o;
        Q[o] = u;
        F[u] = f;
        D[u] = D[f] + 1;
        for(auto &[v, w] : E[u]) if(v != f){
            H[v] = H[u] + w, dfs(v, u);
        }
    }
    const int MAXH = 18 + 3;
    int h = 18;
    int ST[SIZ][MAXH];
    int cmp(int a, int b){
        return D[a] < D[b] ? a : b;
    }
    int T[SIZ], n;
    void init(int _n, int root){
        n = _n;
```

```
27          dfs(root, 0);
28          for(int i = 1;i ⩽ n;++ i)
29              ST[i][0] = Q[i];
30          for(int i = 2;i ⩽ n;++ i)
31              T[i] = T[i >> 1] + 1;
32          for(int i = 1;i ⩽ h;++ i){
33              for(int j = 1;j ⩽ n;++ j) if(j + (1 << i - 1) ⩽ n){
34                  ST[j][i] = cmp(ST[j][i - 1], ST[j + (1 << i - 1)][i -
                        1]);
35              }
36          }
37      }
38      int lca(int a, int b){
39          if(a == b)
40              return a;
41          int l = P[a];
42          int r = P[b];
43          if(l > r)
44              swap(l, r);
45          ++ l;
46          int d = T[r - l + 1];
47          return F[cmp(ST[l][d], ST[r - (1 << d) + 1][d])];
48      }
49      int dis(int a, int b){
50          return H[a] + H[b] - 2 * H[lca(a, b)];
51      }
52  }
53  bool cmp(int a, int b){
54      return LCA :: P[a] < LCA :: P[b];
55  }
56  bool I[MAXN];
57  vector <int> E1[MAXN];
58  vector <int> V1;
59  void solve(vector <int> &V){
60      using LCA :: lca;
61      using LCA :: D;
62      stack <int> S;
63      sort(V.begin(), V.end(), cmp);
64      S.push(1);
65      int v, l;
66      for(auto &u : V) I[u] = true;
67      for(auto &u : V) if(u ≠ 1){
68          int f = lca(u, S.top());
69          l = -1;
70          while(D[v = S.top()] > D[f]){
71              if(l ≠ -1)
72                  E1[v].push_back(l);
73              V1.push_back(l = v), S.pop();
74          }
75          if(l ≠ -1)
76              E1[f].push_back(l);
77          if(f ≠ S.top())
78              S.push(f);
79          S.push(u);
80      }
81      l = -1;
82      while(!S.empty()){
```

```
        v = S.top();
        if(l ≠ -1)
            E1[v].push_back(l);
        V1.push_back(l = v), S.pop();
    }
    // dfs(1, 0); // SOLVE HERE !!!
    for(auto &u : V1)
        E1[u].clear(), I[u] = false;
    V1.clear();
}
```

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。

```
#include<bits/stdc++.h>
#define up(l, r, i) for(int i = l, END##i = r;i ≤ END##i;++ i)
#define dn(r, l, i) for(int i = r, END##i = l;i ≥ END##i;-- i)
using namespace std;
typedef long long i64;
const int INF = 2147483647;
const int MAXN= 2e5 + 3;
const int MAXM= 2e5 + 3;
const int MAXD=  18 + 3;
struct edge{int u, v, w;};
vector <edge> V1[MAXN];
vector <edge> V2[MAXN];
vector <int> H[MAXN];
int n, D[MAXN], W[MAXN], F[MAXD][MAXN];
int o, X[MAXN], L[MAXN];
bool E[MAXN];
void dfs1(int u, int f){
    D[u] = D[f] + 1, F[0][u] = f;
    for(auto &e : V1[u]) if(e.v ≠ f){
        if(D[e.v] && D[e.v] < D[u]){
            int a = e.u;
            int b = e.v;
            int c = ++ o, t = c + n;
            H[c].push_back(a);
            L[c] = W[a] - W[b] + e.w;
            while(a ≠ b)
                E[a] = true, a = F[0][a], H[c].push_back(a);
            for(auto &x : H[c]){
                int w = min(W[x] - W[b], L[c] - W[x] + W[b]);
                V2[x].push_back(edge{x, t, w});
                V2[t].push_back(edge{t, x, w});
            }
        } else if(!D[e.v]){
            W[e.v] = W[u] + e.w, dfs1(e.v, u);
        }
    }
    for(auto &e : V1[u]) if(D[e.v] > D[u]){
        if(!E[e.v]){
```

```
            V2[e.u].push_back({e.u, e.v, e.w});
            V2[e.v].push_back({e.v, e.u, e.w});
        }
    }
}
int d = 18;
void dfs2(int u, int f){
    D[u] = D[f] + 1, F[0][u] = f;
    up(1, d, i) F[i][u] = F[i - 1][F[i - 1][u]];
    for(auto &e : V2[u]) if(e.v != f){
        X[e.v] = X[e.u] + e.w;
        dfs2(e.v, u);
    }
}
int lca(int u, int v){
    if(D[u] < D[v]) swap(u, v);
    dn(d, 0, i) if(D[F[i][u]] >= D[v]) u = F[i][u];
    if(u == v) return u;
    dn(d, 0, i) if(F[i][u] != F[i][v]) u = F[i][u], v = F[i][v];
    return F[0][u];
}
int jump(int u, int v){
    dn(d, 0, i) if(D[F[i][v]] >  D[u]) v = F[i][v];
    return v;
}
int dis(int x, int y){
    int t = lca(x, y);
    if(t > n){
        int u = jump(t, x);
        int v = jump(t, y);
        int w = abs(W[u] - W[v]);
        int l = min(w, L[t - n] - w);
        return X[x] - X[u] + X[y] - X[v] + l;
    } else {
        return X[x] + X[y] - 2 * X[t];
    }
}
int m, q;
int qread();
int main(){
    n = qread(), m = qread(), q = qread();
    up(1, m, i){
        int u = qread(), v = qread(), w = qread();
        V1[u].push_back(edge{u, v, w});
        V1[v].push_back(edge{v, u, w});
    }
    dfs1(1, 0);
    dfs2(1, 0);
    up(1, q, i){
        int u = qread(), v = qread();
        printf("%d\n", dis(u, v));
    }
    return 0;
}
```

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。

```cpp
// 无向图
ll n, m; cin >> n >> m;
vector<pair<ll, ll>> Edges(m);
vector<vector<ll>> G(n + 2);
vector<ll> deg(n + 2);
for (auto &[i, j] : Edges) cin >> i >> j, ++deg[i], ++deg[j];
for (auto [i, j] : Edges) {
    if (deg[i] > deg[j] || (deg[i] == deg[j] && i > j)) swap(i, j);
    G[i].emplace_back(j);
}
vector<ll> val(n + 2);
ll ans = 0;
for (ll i = 1; i <= n; ++i) {
    for (auto j : G[i]) ++val[j];
    for (auto j : G[i]) for (auto k : G[j]) ans += val[k];
    for (auto j : G[i]) val[j] = 0;
}
// 有向图
ll n, m; cin >> n >> m;
vector<pair<ll, ll>> Edges(m);
vector<vector<pll>> G(n + 2);
vector<ll> deg(n + 2);
for (auto &[i, j] : Edges) cin >> i >> j, ++deg[i], ++deg[j];
for (auto [i, j] : Edges) {
    ll flg = 0;
    if (deg[i] > deg[j] || (deg[i] == deg[j] && i > j)) swap(i, j), flg
        = 1;
    G[i].emplace_back(j, flg);
}
vector<ll> in(n + 2), out(n + 2);
ll ans = 0;
for (ll i = 1; i <= n; ++i) {
    for (auto [j, w] : G[i]) w ? (++in[j]) : (++out[j]);
    for (auto [j, w1] : G[i]) for (auto [k, w2] : G[j]) {
        if (w1 == w2) ans += w1 ? in[k] : out[k];
    }
    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。因为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。

```cpp
ll n, m; cin >> n >> m;
vector<pair<ll, ll>> Edges(m);
vector<vector<ll>> G(n + 2), iG(n + 2);
vector<ll> deg(n + 2);
for (auto &[i, j] : Edges) cin >> i >> j, ++deg[i], ++deg[j];
for (auto [i, j] : Edges) {
    if (deg[i] > deg[j] || (deg[i] == deg[j] && i > j)) swap(i, j);
    G[i].emplace_back(j), iG[j].emplace_back(i);
}
ll ans = 0;
vector<ll> v1(n + 2), v2(n + 2);
for (ll i = 1; i <= n; ++i) {
    for (auto j : G[i]) for (auto k : G[j]) ++v1[k];
    for (auto j : iG[i]) for (auto k : G[j]) ans += v1[k], ++v2[k];
    for (auto j : G[i]) for (auto k : G[j]) ans += v1[k] * (v1[k] - 1)
       / 2, v1[k] = 0;
    for (auto j : iG[i]) for (auto k : G[j]) {
        if (deg[k] > deg[i] || (deg[k] == deg[i] && k > i)) ans += v2[k
           ] * (v2[k] - 1) / 2;
        v2[k] = 0;
    }
}
cout << ans << '\n';
```

4.4 基环树

```cpp
#include<bits/stdc++.h>
using namespace std;
typedef long long i64;
const int INF  =  1e9;
const i64 INFL = 1e18;
const int MAXN = 1e5 + 3;
using edge = tuple<int, int, int>;
vector <edge> E[MAXN];
vector <edge> W;
vector <int>  C;
edge F[MAXN];
bool V[MAXN];
int  I[MAXN], o;
void dfs0(int u, int e){
    V[u] = true;
    I[u] = ++ o;
    for(auto &[i, v, w] : E[u]) if(i != e){
        if(V[v]){
            if(I[v] < I[u]){
                for(int p = u;p != v;){
                    auto &[j, f, x] = F[p];
                    C.push_back(p);
                    W.push_back({j, p, x});
                    p = f;
                }
                C.push_back(v);
                W.push_back({i, v, w});
```

```
            }
        } else {
            F[v] = {i, u, w};
            dfs0(v, i);
        }
    }
}
namespace Problem2{
// ==== 删除环上第 i 条边，求直径 ====
    i64 H[MAXN], A1[MAXN], B1[MAXN], A2[MAXN], B2[MAXN], A3[MAXN], B3[
       MAXN];
    i64 L[MAXN];
    i64 dis = 0;
    void dfs1(int u, int e){
        for(auto &[i, v, w] : E[u]) if(i != e){
            if(!V[v]){
                dfs1(v, i);
                dis = max(dis, L[u] + w + L[v]);
                L[u] = max(L[u], L[v] + w);
            }
        }
    }
    int main(){
        int n;
        cin >> n;
        for(int i = 1;i <= n;++ i){
            int u, v, w;
            cin >> u >> v >> w;
            E[u].push_back({i, v, w});
            E[v].push_back({i, u, w});
        }
        dfs0(1, 0);
        memset(V, 0, sizeof(V));
        for(auto &u : C)
            V[u] = true;
        for(auto &u : C){
            dfs1(u, 0);
        }
        int l = 0, r = C.size() - 1;
        for(int i = l;i <= r;++ i){
            int x = C[i];
            if(i > 0)
                H[i] = H[i - 1] + get<2>(W[i - 1]);
            A1[i] = L[x] + H[i];
            B1[i] = L[x] - H[i];
            A2[i] = L[x] - H[i];
            B2[i] = L[x] + H[i];
        }
        i64 h = H[r] + get<2>(W.back());
        for(int i = l;i <= r;++ i)
            A1[i] = max(i == l ? -INFL : A1[i - 1], L[C[i]] + H[i]),
            A2[i] = max(i == l ? -INFL : A2[i - 1], L[C[i]] - H[i]);
        for(int i = r;i >= l;-- i)
            B1[i] = max(i == r ? -INFL : B1[i + 1], L[C[i]] - H[i]),
            B2[i] = max(i == r ? -INFL : B2[i + 1], L[C[i]] + H[i]);
        A3[l] = -INFL, B3[r] = -INFL;
        for(int i = l + 1;i <= r;++ i){
```

```
            int x = C[i];
            i64 w = A2[i - 1] + L[x] + H[i];
            A3[i] = max(A3[i - 1], w);
        }
        for(int i = r - 1;i >= l;-- i){
            int x = C[i];
            i64 w = B2[i + 1] + L[x] - H[i];
            B3[i] = max(B3[i + 1], w);
        }
        i64 t =  INFL;
        for(int i = l;i < r;++ i){
            i64 d = A1[i] + B1[i + 1] + h;
            i64 g = A2[i] + B2[i + 1] + 0;
            d = max({d, dis, A3[i], B3[i + 1]});
            t = min(t, d);
        }
        t = min(t, max(A3[r], dis));
        if(t % 2 == 0)
            cout << t / 2 << ".0" << endl;
        if(t % 2 == 1)
            cout << t / 2 << ".5" << endl;
        return 0;
    }
}
namespace Problem3{
// ==== 求最大点权独立集 ====
    int A[MAXN];
    i64 X[MAXN], Y[MAXN];
    i64 P[MAXN][2], Q[MAXN][2];
    void dfs1(int u, int e){
        for(auto &[i, v, w] : E[u]) if(i != e){
            if(!V[v]){
                dfs1(v, i);
                Y[u] += max(X[v], Y[v]);
                X[u] += Y[v];
            }
        }
        X[u] += A[u];
    }
    int main(){
        int n;
        cin >> n;
        for(int i = 1;i <= n;++ i){
            cin >> A[i];
        }
        for(int i = 1;i <= n;++ i){
            int u, v;
            cin >> u >> v;
            ++ u, ++ v;
            E[u].push_back({i, v, 0});
            E[v].push_back({i, u, 0});
        }
        double p;
        cin >> p;
        dfs0(1, 0);
        memset(V, 0, sizeof(V));
        for(auto &u : C)
```

```
            V[u] = true;
        for(auto &u : C){
            dfs1(u, 0);
        }
        int l = 0, r = C.size() - 1;
        P[0][1] = X[C[0]];
        P[0][0] = -INFL;
        Q[0][0] = Y[C[0]];
        Q[0][1] = -INFL;
        for(int i = l + 1;i <= r;++ i){
            int x = C[i];
            P[i][1] = X[x] + P[i - 1][0];
            P[i][0] = Y[x] + max(P[i - 1][0], P[i - 1][1]);
            Q[i][1] = X[x] + Q[i - 1][0];
            Q[i][0] = Y[x] + max(Q[i - 1][0], Q[i - 1][1]);
        }
        i64 ans = max({P[r][0], Q[r][0], Q[r][1]});
        cout << fixed << setprecision(1) << ans * p << endl;
        return 0;
    }
}
int main(){
    return Problem3 :: main();
}
```

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。

```
#include<bits/stdc++.h>
using namespace std;
using i64 = long long;
const int INF  =  1e9;
const i64 INFL = 1e18;
namespace SCC{
    const int MAXN= 2e6 + 3;
    vector <int> V[MAXN];
    stack  <int> S;
    int D[MAXN], L[MAXN], C[MAXN], o, s;
    bool F[MAXN], I[MAXN];
    void add(int u, int v){ V[u].push_back(v); }
    void dfs(int u){
        L[u] = D[u] = ++ o, S.push(u), I[u] = F[u] = true;
        for(auto &v : V[u]){
            if(F[v]){
                if(I[v]) L[u] = min(L[u], D[v]);
            } else {
                dfs(v),  L[u] = min(L[u], L[v]);
            }
        }
        if(L[u] == D[u]){
            int c = ++ s;
            while(S.top() != u){
                int v = S.top(); S.pop();
                I[v] = false;
                C[v] = c;
```

```
            }
            S.pop(), I[u] = false, C[u] = c;
        }
    }
}
const int MAXN = 1e6 + 3;
int X[MAXN][2], o;
int main(){
    ios :: sync_with_stdio(false);
    int n, m;
    cin >> n >> m;
    for(int i = 1;i <= n;++ i)
        X[i][0] = ++ o;
    for(int i = 1;i <= n;++ i)
        X[i][1] = ++ o;
    for(int i = 1;i <= m;++ i){
        int a, x, b, y;
        cin >> a >> x >> b >> y;
        SCC :: add(X[a][!x], X[b][y]);
        SCC :: add(X[b][!y], X[a][x]);
    }
    for(int i = 1;i <= o;++ i)
        if(!SCC :: F[i])
            SCC :: dfs(i);
    bool ok = true;
    for(int i = 1;i <= n;++ i){
        if(SCC :: C[X[i][0]] == SCC :: C[X[i][1]])
            ok = false;
    }
    if(ok){
        cout << "POSSIBLE" << endl;
        for(int i = 1;i <= n;++ i){
            int a = SCC :: C[X[i][0]];
            int b = SCC :: C[X[i][1]];
            if(a < b)
                cout << 0 << " ";
            else
                cout << 1 << " ";
        }
        cout << endl;
    } else {
        cout << "IMPOSSIBLE" << endl;
    }
    return 0;
}
```

4.6 割点

```
#include<bits/stdc++.h>
using namespace std;
const int MAXN= 2e4 + 3;
const int MAXM= 1e5 + 3;
vector<int> V[MAXN];
int n, m, o, D[MAXN], L[MAXN];
bool F[MAXN], C[MAXN];
void dfs(int u, int g){
    L[u] = D[u] = ++ o, F[u] = true; int s = 0;
```

```
    for(auto &v : V[u]){
        if(!F[v]){
            dfs(v, g), ++ s;
            L[u] = min(L[u], L[v]);
            if(u != g && L[v] >= D[u]) C[u] = true;
        } else {
            L[u] = min(L[u], D[v]);
        }
    }
    if(u == g && s > 1) C[u] = true;
}
int main(){
    cin >> n >> m;
    for(int i = 1;i <= m;++ i){
        int u, v;
        cin >> u >> v;
        V[u].push_back(v);
        V[v].push_back(u);
    }
    for(int i = 1;i <= n;++ i)
        if(!F[i]) dfs(i, i);
    vector <int> ANS;
    for(int i = 1;i <= n;++ i)
        if(C[i]) ANS.push_back(i);
    cout << ANS.size() << endl;
    for(auto &u : ANS)
        cout << u << " ";
    return 0;
}
```

4.7 边双连通分量

```
#include<bits/stdc++.h>
using namespace std;
using i64 = long long;
const int INF  =  1e9;
const i64 INFL = 1e18;
const int MAXN= 5e5 + 3;
vector <vector<int>> A;
vector <pair<int, int>> V[MAXN];
stack  <int> S;
int D[MAXN], L[MAXN], o;
bool I[MAXN];
void dfs(int u, int l){
    D[u] = L[u] = ++ o; I[u] = true, S.push(u); int s = 0;
    for(auto &p : V[u]) {
        int v = p.first, id = p.second;
        if(id != l){
            if(D[v]){
                if(I[v])    L[u] = min(L[u], D[v]);
            } else {
                dfs(v, id), L[u] = min(L[u], L[v]), ++ s;
            }
        }
    }
    if(D[u] == L[u]){
        vector <int> T;
```

```
        while(S.top() ≠ u){
            int v = S.top(); S.pop();
            T.push_back(v), I[v] = false;
        }
        T.push_back(u), S.pop(), I[u] = false;
        A.push_back(T);
    }
}
```

4.8 点双连通分量

```
#include<bits/stdc++.h>
using namespace std;
using i64 = long long;
const int INF  =  1e9;
const i64 INFL = 1e18;
const int MAXN= 5e5 + 3;
vector <vector<int>> A;
vector <int> V[MAXN];
stack  <int> S;
int D[MAXN], L[MAXN], o; bool I[MAXN];
void dfs(int u, int f){
    D[u] = L[u] = ++ o; I[u] = true, S.push(u); int s = 0;
    for(auto &v : V[u]) if(v ≠ f){
        if(D[v]){
            if(I[v])   L[u] = min(L[u], D[v]);
        } else {
            dfs(v, u), L[u] = min(L[u], L[v]), ++ s;
            if(L[v] ≥ D[u]){
                vector <int> T;
                while(S.top() ≠ v){
                    int t = S.top(); S.pop();
                    T.push_back(t), I[t] = false;
                }
                T.push_back(v), S.pop(), I[v] = false;
                T.push_back(u);
                A.push_back(T);
            }
        }
    }
    if(f == 0 && s == 0){
        A.push_back({u});
    }
}
```

4.9 强连通分量

```
#include<bits/stdc++.h>
using namespace std;
using i64 = long long;
const int INF  =  1e9;
const i64 INFL = 1e18;
const int MAXN= 5e5 + 3;
vector <int> V[MAXN];
stack  <int> S;
```

```cpp
int D[MAXN], L[MAXN], C[MAXN], o, s;
bool F[MAXN], I[MAXN];
void add(int u, int v){ V[u].push_back(v); }
void dfs(int u){
    L[u] = D[u] = ++ o, S.push(u), I[u] = F[u] = true;
    for(auto &v : V[u]){
        if(F[v]){
            if(I[v]) L[u] = min(L[u], D[v]);
        } else {
            dfs(v),  L[u] = min(L[u], L[v]);
        }
    }
    if(L[u] == D[u]){
        int c = ++ s;
        while(S.top() != u){
            int v = S.top(); S.pop();
            I[v] = false;
            C[v] = c;
        }
        S.pop(), I[u] = false, C[u] = c;
    }
}
vector <int> ANS[MAXN];
int main(){
    int n, m;
    cin >> n >> m;
    for(int i = 1;i <= m;++ i){
        int u, v;
        cin >> u >> v;
        V[u].push_back(v);
    }
    for(int i = 1;i <= n;++ i)
        if(!F[i])
            dfs(i);
    for(int i = 1;i <= n;++ i){
        ANS[C[i]].push_back(i);
    }
    cout << s << endl;
    for(int i = 1;i <= n;++ i) if(F[i]){
        int c = C[i];
        sort(ANS[c].begin(), ANS[c].end());
        for(auto &u : ANS[c])
            cout << u << " ", F[u] = false;
        cout << endl;
    }
    return 0;
}
```

5 网络流

5.1 费用流

```cpp
#include<bits/stdc++.h>
using namespace std;
namespace MCMF{
    using i64 = long long;
```

```
5     const i64 INF  = 1e18;
6     const int MAXN = 1e5 + 3;
7     const int MAXM = 2e5 + 3;
8     int H[MAXN], V[MAXM], N[MAXM], W[MAXM], F[MAXM], o = 1, n;
9     void add(int u, int v, int f, int c){
10        V[++ o] = v, N[o] = H[u], H[u] = o, F[o] = f, W[o] =  c;
11        V[++ o] = u, N[o] = H[v], H[v] = o, F[o] = 0, W[o] = -c;
12        n = max(n, u);
13        n = max(n, v);
14    }
15    void clear(){
16        for(int i = 1;i ⩽ n;++ i)
17            H[i] = 0;
18        n = 0, o = 1;
19    }
20    bool I[MAXN];
21    i64 D[MAXN];
22    bool spfa(int s, int t){
23        queue <int> Q;
24        Q.push(s), I[s] = true;
25        for(int i = 1;i ⩽ n;++ i)
26            D[i] = INF;
27        D[s] = 0;
28        while(!Q.empty()){
29            int u = Q.front(); Q.pop(), I[u] = false;
30            for(int i = H[u];i;i = N[i]){
31                const int &v = V[i];
32                const int &f = F[i];
33                const int &w = W[i];
34                if(f && D[u] + w < D[v]){
35                    D[v] = D[u] + w;
36                    if(!I[v]) Q.push(v), I[v] = true;
37                }
38            }
39        }
40        return D[t] ≠ INF;
41    }
42    int C[MAXN]; bool T[MAXN];
43    pair<i64, i64> dfs(int s, int t, int u, i64 maxf){
44        if(u == t)
45            return make_pair(maxf, 0);
46        i64 totf = 0;
47        i64 totc = 0;
48        T[u] = true;
49        for(int &i = C[u];i;i = N[i]){
50            const int &v = V[i];
51            const int &f = F[i];
52            const int &w = W[i];
53            if(f && D[v] == D[u] + w && !T[v]){
54                auto p = dfs(s, t, v, min(1ll * F[i], maxf));
55                i64 f = p.first;
56                i64 c = p.second;
57                F[i    ] -= f;
58                F[i ^ 1] += f;
59                totf += f;
60                totc += 1ll * f * W[i] + c;
61                maxf -= f;
```

```
                if(maxf == 0){
                    T[u] = false;
                    return make_pair(totf, totc);
                }
            }
        }
        T[u] = false;
        return make_pair(totf, totc);
    }
    pair<i64, i64> mcmf(int s, int t){
        i64 ans1 = 0;
        i64 ans2 = 0;
        pair<i64, i64> r;
        while(spfa(s, t)){
            memcpy(C, H, sizeof(H));
            r = dfs(s, t, s, INF);
            ans1 += r.first;
            ans2 += r.second;
        }
        return make_pair(ans1, ans2);
    }
}
int qread();
int main(){
    int n = qread(), m = qread(), s = qread(), t = qread();
    for(int i = 1;i ⩽ m;++ i){
        int u = qread(), v = qread(), f = qread(), c = qread();
        MCMF :: add(u, v, f, c);
    }
    pair<long long, long long> ans = MCMF :: mcmf(s, t);
    printf("%lld %lld\n", ans.first, ans.second);
    return 0;
}
```

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。

```
#include <bits/stdc++.h>
using namespace std;
int qread();
namespace Dinic{
    const long long INF = 1e18;
    const int SIZ = 1e5 + 3;
    int n, m;
    int H[SIZ], V[SIZ], N[SIZ], F[SIZ], t = 1;
    int add(int u, int v, int f){
        V[++ t] = v, N[t] = H[u], F[t] = f, H[u] = t;
        V[++ t] = u, N[t] = H[v], F[t] = 0, H[v] = t;
        n = max(n, u);
        n = max(n, v);
        return t - 1;
    }
    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;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 maxf){
42          if(u == t)
43              return maxf;
44          long long totf = 0;
45          for(int &i = C[u];i;i = N[i]){
46              const int &v = V[i];
47              const int &f = F[i];
48              if(D[v] == D[u] + 1){
49                  long long resf = dfs(s, t, v, min(maxf, 1ll * f));
50                  totf += resf;
51                  maxf -= resf;
52                  F[i    ] -= resf;
53                  F[i ^ 1] += resf;
54                  if(maxf == 0)
55                      return totf;
56              }
57          }
58          return totf;
59      }
60      long long dinic(int s, int t){
61          long long ans = 0;
62          while(bfs(s, t)){
63              memcpy(C, H, sizeof(H));
64              ans += dfs(s, t, s, INF);
65          }
66          return ans;
67      }
68  }
69  namespace GHTree{
70      const int MAXN =  500 + 5;
71      const int MAXM = 1500 + 5;
72      const int INF  = 1e9;
73      int n, m, U[MAXM], V[MAXM], W[MAXM], A[MAXM], B[MAXM];
```

```
    void add(int u, int v, int w){
        ++ m;
        U[m] = u;
        V[m] = v;
        W[m] = w;
        A[m] = Dinic :: add(u, v, w);
        B[m] = Dinic :: add(v, u, w);
        n = max(n, u);
        n = max(n, v);
    }
    vector <pair<int, int> > E[MAXN];
    void build(vector <int> N){
        int s = N.front();
        int t = N.back();
        if(s == t) return;
        for(int i = 1;i <= m;++ i){
            int a = A[i]; Dinic :: F[a] = W[i], Dinic :: F[a ^ 1] = 0;
            int b = B[i]; Dinic :: F[b] = W[i], Dinic :: F[b ^ 1] = 0;
        }
        int w = Dinic :: dinic(s, t);
        E[s].push_back(make_pair(t, w));
        E[t].push_back(make_pair(s, w));
        vector <int> P;
        vector <int> Q;
        for(auto &u : N){
            if(Dinic :: D[u] != 0)
                P.push_back(u);
            else
                Q.push_back(u);
        }
        build(P), build(Q);
    }
    int D[MAXN];
    int cut(int s, int t){
        queue <int> Q; Q.push(s);
        for(int i = 1;i <= n;++ i)
            D[i] = -1;
        D[s] = INF;
        while(!Q.empty()){
            int u = Q.front(); Q.pop();
            for(auto &e : E[u]){
                int v = e.first;
                int w = e.second;
                if(D[v] == -1){
                    D[v] = min(D[u], w);
                    Q.push(v);
                }
            }
        }
        return D[t];
    }
}
```

5.3 最大流

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

```cpp
using i64 = long long;
const int INF  =  1e9;
const i64 INFL = 1e18;
namespace Dinic{
    const i64 INF = 1e18;
    const int SIZ = 5e5 + 3;
    int n, m;
    int H[SIZ], V[SIZ], N[SIZ], F[SIZ], t = 1;
    void add(int u, int v, int f){
        V[++ t] = v, N[t] = H[u], F[t] = f, H[u] = t;
        V[++ t] = u, N[t] = H[v], F[t] = 0, H[v] = t;
        n = max(n, u);
        n = max(n, v);
    }
    void clear(){
        for(int i = 1;i ⩽ n;++ i)
            H[i] = 0;
        n = m = 0, t = 1;
    }
    int D[SIZ];
    bool bfs(int s, int t){
        queue <int> Q;
        for(int i = 1;i ⩽ n;++ i)
            D[i] = 0;
        Q.push(s), D[s] = 1;
        while(!Q.empty()){
            int u = Q.front(); Q.pop();
            for(int i = H[u];i;i = N[i]){
                const int &v = V[i];
                const int &f = F[i];
                if(f ≠ 0 && !D[v]){
                    D[v] = D[u] + 1;
                    Q.push(v);
                }
            }
        }
        return D[t] ≠ 0;
    }
    int C[SIZ];
    i64 dfs(int s, int t, int u, i64 maxf){
        if(u == t)
            return maxf;
        i64 totf = 0;
        for(int &i = C[u];i;i = N[i]){
            const int &v = V[i];
            const int &f = F[i];
            if(D[v] == D[u] + 1){
                i64 resf = dfs(s, t, v, min(maxf, 1ll * f));
                totf += resf;
                maxf -= resf;
                F[i    ] -= resf;
                F[i ^ 1] += resf;
                if(maxf == 0)
                    return totf;
            }
        }
        return totf;
```

```
    }
    i64 dinic(int s, int t){
        i64 ans = 0;
        while(bfs(s, t)){
            memcpy(C, H, sizeof(H));
            ans += dfs(s, t, s, INF);
        }
        return ans;
    }
}
// ==== TEST ====
int qread();
int main(){
    int n = qread(), m = qread(), s = qread(), t = qread();
    for(int i = 1;i ⩽ m;++ i){
        int u = qread(), v = qread(), f = qread();
        Dinic :: add(u, v, f);
    }
    printf("%lld\n", Dinic :: dinic(s, t));
    return 0;
}
```

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。

```
#include<bits/stdc++.h>
using namespace std;
using i64 = long long;
const int INF  = 1e9;
const i64 INFL = 1e18;
namespace MCMF{
    const int MAXN = 1e5 + 3;
    const int MAXM = 2e5 + 3;
    int H[MAXN], V[MAXM], N[MAXM], W[MAXM], F[MAXM], o = 1, n;
    void add0(int u, int v, int f, int c){
        V[++ o] = v, N[o] = H[u], H[u] = o, F[o] = f, W[o] =  c;
        V[++ o] = u, N[o] = H[v], H[v] = o, F[o] = 0, W[o] = -c;
        n = max(n, u);
        n = max(n, v);
    }
    bool I[MAXN];
    i64 D[MAXN];
    bool spfa(int s, int t){
        queue <int> Q;
        Q.push(s), I[s] = true;
        for(int i = 1;i ⩽ n;++ i)
            D[i] = INFL;
        D[s] = 0;
        while(!Q.empty()){
            int u = Q.front(); Q.pop(), I[u] = false;
            for(int i = H[u];i;i = N[i]){
```

```
27                  const int &v = V[i];
28                  const int &f = F[i];
29                  const int &w = W[i];
30                  if(f && D[u] + w < D[v]){
31                      D[v] = D[u] + w;
32                      if(!I[v]) Q.push(v), I[v] = true;
33                  }
34              }
35          }
36          return D[t] =/= INFL;
37      }
38      int C[MAXN]; bool T[MAXN];
39      pair<i64, i64> dfs(int s, int t, int u, i64 maxf){
40          if(u == t)
41              return make_pair(maxf, 0);
42          i64 totf = 0;
43          i64 totc = 0;
44          T[u] = true;
45          for(int &i = C[u];i;i = N[i]){
46              const int &v = V[i];
47              const int &f = F[i];
48              const int &w = W[i];
49              if(f && D[v] == D[u] + w && !T[v]){
50                  auto p = dfs(s, t, v, min(1ll * F[i], maxf));
51                  i64 f = p.first;
52                  i64 c = p.second;
53                  F[i    ] -= f;
54                  F[i ^ 1] += f;
55                  totf += f;
56                  totc += 1ll * f * W[i] + c;
57                  maxf -= f;
58                  if(maxf == 0){
59                      T[u] = false;
60                      return make_pair(totf, totc);
61                  }
62              }
63          }
64          T[u] = false;
65          return make_pair(totf, totc);
66      }
67      pair<i64, i64> mcmf(int s, int t){
68          i64 ans1 = 0;
69          i64 ans2 = 0;
70          pair<i64, i64> r;
71          while(spfa(s, t)){
72              memcpy(C, H, sizeof(H));
73              r = dfs(s, t, s, INFL);
74              ans1 += r.first;
75              ans2 += r.second;
76          }
77          return make_pair(ans1, ans2);
78      }
79      i64 cost0;
80      int G[MAXN];
81      void add(int u, int v, int l, int r, int c){
82          G[v] += l;
83          G[u] -= l;
```

```
        cost0 += 1ll * l * c;
        add0(u, v, r - l, c);
    }
    i64 solve(){
        int s = ++ n;
        int t = ++ n;
        i64 sum = 0;
        for(int i = 1;i <= n - 2;++ i){
            if(G[i] < 0)
                add0(i, t, -G[i], 0);
            else
                add0(s, i,  G[i], 0), sum += G[i];
        }
        auto res = mcmf(s, t);
        if(res.first != sum)
            return -1;
        return res.second + cost0;
    }
    i64 solve(int s0, int t0){
        add0(t0, s0, INF, 0);
        int s = ++ n;
        int t = ++ n;
        i64 sum = 0;
        for(int i = 1;i <= n - 2;++ i){
            if(G[i] < 0)
                add0(i, t, -G[i], 0);
            else
                add0(s, i,  G[i], 0), sum += G[i];
        }
        auto res = mcmf(s, t);
        if(res.first != sum)
            return -1;
        return res.second + cost0;
    }
}
// ==== TEST ====
int qread();
int main(){
    return 0;
}
```

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。

```
#include <bits/stdc++.h>
using namespace std;
int qread();
using i64 = long long;
const int INF  =  1e9;
const i64 INFL = 1e18;
namespace MCMF{
```

```cpp
const int MAXN = 1e5 + 3;
const int MAXM = 2e5 + 3;
int H[MAXN], V[MAXM], N[MAXM], F[MAXM], o = 1, n;
void add0(int u, int v, int f){
    V[++ o] = v, N[o] = H[u], H[u] = o, F[o] = f;
    V[++ o] = u, N[o] = H[v], H[v] = o, F[o] = 0;
    n = max(n, u);
    n = max(n, v);
}
i64 D[MAXN];
bool bfs(int s, int t){
    queue <int> Q;
    for(int i = 1;i ⩽ n;++ i)
        D[i] = 0;
    Q.push(s), D[s] = 1;
    while(!Q.empty()){
        int u = Q.front(); Q.pop();
        for(int i = H[u];i;i = N[i]){
            const int &v = V[i];
            const int &f = F[i];
            if(f ≠ 0 && !D[v]){
                D[v] = D[u] + 1;
                Q.push(v);
            }
        }
    }
    return D[t] ≠ 0;
}
int C[MAXN];
i64 dfs(int s, int t, int u, i64 maxf){
    if(u == t)
        return maxf;
    i64 totf = 0;
    for(int &i = C[u];i;i = N[i]){
        const int &v = V[i];
        const int &f = F[i];
        if(f && D[v] == D[u] + 1){
            i64 f = dfs(s, t, v, min(1ll * F[i], maxf));
            F[i    ] -= f;
            F[i ^ 1] += f;
            totf += f;
            maxf -= f;
            if(maxf == 0){
                return totf;
            }
        }
    }
    return totf;
}
i64 mcmf(int s, int t){
    i64 ans = 0;
    while(bfs(s, t)){
        memcpy(C, H, sizeof(H));
        ans += dfs(s, t, s, INFL);
    }
    return ans;
}
```

```
65      int G[MAXN];
66      void add(int u, int v, int l, int r){
67          G[v] += l;
68          G[u] -= l;
69          add0(u, v, r - l);
70      }
71      void clear(){
72          for(int i = 1;i ⩽ o;++ i){
73              N[i] = F[i] = V[i] = 0;
74          }
75          for(int i = 1;i ⩽ n;++ i){
76              H[i] = G[i] = C[i] = 0;
77          }
78          o = 1, n = 0;
79      }
80      bool solve(){
81          int s = ++ n;
82          int t = ++ n;
83          i64 sum = 0;
84          for(int i = 1;i ⩽ n - 2;++ i){
85              if(G[i] < 0)
86                  add0(i, t, -G[i]);
87              else
88                  add0(s, i,  G[i]), sum += G[i];
89          }
90          auto res = mcmf(s, t);
91          if(res ≠ sum)
92              return true;
93          return false;
94      }
95      i64 solve(int s0, int t0){
96          add0(t0, s0, INF);
97          int s = ++ n;
98          int t = ++ n;
99          i64 sum = 0;
100         for(int i = 1;i ⩽ n - 2;++ i){
101             if(G[i] < 0)
102                 add0(i, t, -G[i]);
103             else
104                 add0(s, i,  G[i]), sum += G[i];
105         }
106         auto res = mcmf(s, t);
107         if(res ≠ sum)
108             return -1;
109         return mcmf(s0, t0);
110     }
111 }
112 const int MAXN = 1e3 + 3;
113 const int MAXM = 365 + 3;
114 int G[MAXN], A[MAXN], B[MAXM];
115 int main(){
116     ios :: sync_with_stdio(false);
117     cin.tie(nullptr);
118     int n, m, o = 0;
119     while(cin >> n >> m){
120         int s = ++ o;
121         int t = ++ o;
```

```
        for(int i = 1;i ⩽ m;++ i){
            cin >> G[i];
            A[i] = ++ o;
            MCMF :: add(A[i], t, G[i], INF);
        }
        for(int i = 1;i ⩽ n;++ i){
            B[i] = ++ o;
            int c, d;
            cin >> c >> d;
            MCMF :: add(s, B[i], 0, d);
            for(int j = 1;j ⩽ c;++ j){
                int t, l, r;
                cin >> t >> l >> r;
                t ++;
                MCMF :: add(B[i], A[t], l, r);
            }
        }
        cout << MCMF :: solve(s, t) << "\n\n";
        MCMF :: clear();
    }
    return 0;
}
```

6 数学

6.1 线性代数

6.1.1 行列式

```
#include<bits/stdc++.h>
using namespace std;
using i64 = long long;
const int INF  =  1e9;
const i64 INFL = 1e18;
const int MAXN = 600 + 3;
int MOD;
struct Mat{
    int n, m;
    int W[MAXN][MAXN];
    Mat(int _n = 0, int _m = 0){
        n = _n;
        m = _m;
        for(int i = 1;i ⩽ n;++ i)
            for(int j = 1;j ⩽ m;++ j)
                W[i][j] = 0;
    }
};
int mat_det(Mat a){
    int ans = 1;
    const int &n = a.n;
    for(int i = 1;i ⩽ n;++ i){
        int f = -1;
        for(int j = i;j ⩽ n;++ j) if(a.W[j][i] ≠ 0){
            f = j;
            break;
        }
```

```
        if(f == -1){
            return 0;
        }
        if(f != i){
            for(int j = 1;j <= n;++ j)
                swap(a.W[i][j], a.W[f][j]);
            ans = MOD - ans;
        }
        for(int j = i + 1;j <= n;++ j) if(a.W[j][i]){
            while(a.W[j][i]){
                int u = a.W[i][i];
                int v = a.W[j][i];
                if(u > v){
                    for(int k = 1;k <= n;++ k)
                        swap(a.W[i][k], a.W[j][k]);
                    ans = MOD - ans;
                    swap(u, v);
                }
                int rate = v / u;
                for(int k = 1;k <= n;++ k){
                    a.W[j][k] = (a.W[j][k] - 1ll * rate * a.W[i][k] %
                       MOD + MOD) % MOD;
                }
            }
        }
    }
    for(int i = 1;i <= n;++ i)
        ans = 1ll * ans * a.W[i][i] % MOD;
    return ans;
}
int main(){
    int n;
    cin >> n >> MOD;
    Mat A(n, n);
    for(int i = 1;i <= n;++ i)
        for(int j = 1;j <= n;++ j)
            cin >> A.W[i][j], A.W[i][j] %= MOD;
    cout << mat_det(A) << endl;
    return 0;
}
```

6.1.2 高斯消元与求秩（实数）

```
#include<bits/stdc++.h>
using namespace std;
using i64 = long long;
const int INF  =  1e9;
const i64 INFL = 1e18;
const int MAXN = 100 + 3;
const double EPS = 1e-9;
struct Mat{
    int n, m;
    double W[MAXN][MAXN];
    Mat(int _n = 0, int _m = 0){
        n = _n;
        m = _m;
        for(int i = 1;i <= n;++ i)
```

```
            for(int j = 1;j ⩽ m;++ j)
                W[i][j] = 0;
    }
};
bool zero(double f){
    return fabs(f) < EPS;
}
int mat_rank(Mat &a){
    const int &n = a.n;
    const int &m = a.m;
    int cnt = 0;
    for(int i = 1;i ⩽ m;++ i){
        int p = cnt + 1;
        int f = -1;
        for(int j = p;j ⩽ n;++ j){
            if(!zero(a.W[j][i])){
                f = j;
                break;
            }
        }
        if(f == -1)
            continue;
        if(f ≠ p){
            for(int j = 1;j ⩽ m;++ j)
                swap(a.W[p][j], a.W[f][j]);
        }
        ++ cnt;
        for(int j = p + 1;j ⩽ n;++ j){
            double rate = a.W[j][i] / a.W[p][i];
            for(int k = 1;k ⩽ m;++ k){
                a.W[j][k] -= rate * a.W[p][k];
            }
        }
    }
    return cnt;
}
double X[MAXN];
int main(){
    int n;
    cin >> n;
    Mat A(n, n);
    Mat T(n, n + 1);
    for(int i = 1;i ⩽ n;++ i){
        for(int j = 1;j ⩽ n;++ j)
            cin >> A.W[i][j];
        for(int j = 1;j ⩽ n;++ j)
            T.W[i][j] = A.W[i][j];
        cin >> T.W[i][n + 1];
    }
    int res1 = mat_rank(A);
    int res2 = mat_rank(T);
    if(res1 ≠ res2)
        cout << -1 << endl;
    else
    if(res2 <  n)
        cout << 0 << endl;
    else {
```

```
        for(int i = n;i >= 1;-- i){
            X[i] = T.W[i][n + 1] / T.W[i][i];
            for(int j = i - 1;j >= 1;-- j){
                double rate = T.W[j][i] / T.W[i][i];
                T.W[j][    i] -= rate * T.W[i][    i];
                T.W[j][n + 1] -= rate * T.W[i][n + 1];
            }
        }
        for(int i = 1;i <= n;++ i)
            cout << "x" << i << "=" << fixed << setprecision(2) << X[i]
                << endl;
    }
    return 0;
}
```

6.1.3 高斯消元与求秩（整数）

```
#include<bits/stdc++.h>
using namespace std;
using i64 = long long;
const int INF  =  1e9;
const i64 INFL = 1e18;
const int MAXN = 100 + 3;
const int MOD  = 998244353;
struct Mat{
    int n, m;
    int W[MAXN][MAXN];
    Mat(int _n = 0, int _m = 0){
        n = _n;
        m = _m;
        for(int i = 1;i <= n;++ i)
            for(int j = 1;j <= m;++ j)
                W[i][j] = 0;
    }
};
int power(int a, int b){
    int r = 1;
    while(b){
        if(b & 1) r = 1ll * r * a % MOD;
        b >>= 1,  a = 1ll * a * a % MOD;
    }
    return r;
}
int inv(int x){
    return power(x, MOD - 2);
}
int mat_rank(Mat &a){
    const int &n = a.n;
    const int &m = a.m;
    int cnt = 0;
    for(int i = 1;i <= m;++ i){
        int p = cnt + 1;
        int f = -1;
        for(int j = p;j <= n;++ j){
            if(a.W[j][i] != 0){
                f = j;
                break;
```

```
41              }
42          }
43          if(f == -1)
44              continue;
45          if(f != p){
46              for(int j = 1;j <= m;++ j)
47                  swap(a.W[p][j], a.W[f][j]);
48          }
49          ++ cnt;
50          int invp = inv(a.W[p][i]);
51          for(int j = p + 1;j <= n;++ j){
52              int rate = 1ll * a.W[j][i] * invp % MOD;
53              for(int k = 1;k <= m;++ k){
54                  a.W[j][k] = (a.W[j][k] - 1ll * rate * a.W[p][k] % MOD +
                        MOD) % MOD;
55              }
56          }
57      }
58      return cnt;
59  }
60  int X[MAXN];
61  int main(){
62      int n;
63      cin >> n;
64      Mat A(n, n);
65      Mat T(n, n + 1);
66      for(int i = 1;i <= n;++ i){
67          for(int j = 1;j <= n;++ j)
68              cin >> A.W[i][j];
69          for(int j = 1;j <= n;++ j)
70              T.W[i][j] = A.W[i][j];
71          cin >> T.W[i][n + 1];
72      }
73      int res1 = mat_rank(A);
74      int res2 = mat_rank(T);
75      if(res1 != res2)
76          cout << -1 << endl;
77      else
78      if(res2 <  n)
79          cout << 0 << endl;
80      else {
81          for(int i = n;i >= 1;-- i){
82              int invp = inv(T.W[i][i]);
83              X[i] = 1ll * T.W[i][n + 1] * invp % MOD;
84              for(int j = i - 1;j >= 1;-- j){
85                  int rate = 1ll * T.W[j][i] * invp % MOD;
86                  T.W[j][    i] = (T.W[j][    i] - 1ll * rate * T.W[i][
                            i] % MOD + MOD) % MOD;
87                  T.W[j][n + 1] = (T.W[j][n + 1] - 1ll * rate * T.W[i][n
                        + 1] % MOD + MOD) % MOD;
88              }
89          }
90          for(int i = 1;i <= n;++ i)
91              cout << "x" << i << "=" << X[i] << endl;
92      }
93      return 0;
94  }
```

6.1.4 矩阵求逆

```cpp
#include<bits/stdc++.h>
using namespace std;
using i64 = long long;
const int INF  =  1e9;
const i64 INFL = 1e18;
const int MAXN = 400 + 3;
const int MOD  = 1e9 + 7;
struct Mat{
    int n, m;
    int W[MAXN][MAXN];
    Mat(int _n = 0, int _m = 0){
        n = _n;
        m = _m;
        for(int i = 1;i <= n;++ i)
            for(int j = 1;j <= m;++ j)
                W[i][j] = 0;
    }
};
int power(int a, int b){
    int r = 1;
    while(b){
        if(b & 1) r = 1ll * r * a % MOD;
        b >>= 1,  a = 1ll * a * a % MOD;
    }
    return r;
}
int inv(int x){
    return power(x, MOD - 2);
}
bool mat_inv(Mat &a){
    const int &n = a.n;
    Mat b(n, n);
    for(int i = 1;i <= n;++ i)
        b.W[i][i] = 1;
    for(int i = 1;i <= n;++ i){
        int f = -1;
        for(int j = i;j <= n;++ j) if(a.W[j][i] != 0){
            f = j;
            break;
        }
        if(f == -1){
            return false;
        }
        if(f != i){
            for(int j = 1;j <= n;++ j)
                swap(a.W[i][j], a.W[f][j]),
                swap(b.W[i][j], b.W[f][j]);
        }
        int invp = inv(a.W[i][i]);
        for(int j = i + 1;j <= n;++ j){
            int rate = 1ll * a.W[j][i] * invp % 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 = 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 $A = \{a_1, a_2, \cdots, a_n\}$，终点集合 $B = \{b_1, b_2, \cdots, b_n\}$。

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

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**不相交**的路径组 $p=(p_1,p_2,\cdots,p_n)$，其中从 a_i 到达 b_{π_i}，π 是一个排列。定义 $\sigma(\pi)$ 是 π 逆序对的数量。

给出 LGV 的叙述如下：

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

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 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：

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

```cpp
#include<bits/stdc++.h>
using namespace std;
using i64 = long long;
const int INF  =  1e9;
const i64 INFL = 1e18;
const int MAXN = 300 + 3;
const int MOD  = 1e9 + 7;
struct Mat{
    int n, m;
    int W[MAXN][MAXN];
    Mat(int _n = 0, int _m = 0){
        n = _n;
        m = _m;
        for(int i = 1;i <= n;++ i)
            for(int j = 1;j <= m;++ j)
```

```
16                  W[i][j] = 0;
17              }
18      };
19      int mat_det(Mat a){
20          int ans = 1;
21          const int &n = a.n;
22          for(int i = 1;i <= n;++ i){
23              int f = -1;
24              for(int j = i;j <= n;++ j) if(a.W[j][i] != 0){
25                  f = j;
26                  break;
27              }
28              if(f == -1){
29                  return 0;
30              }
31              if(f != i){
32                  for(int j = 1;j <= n;++ j)
33                      swap(a.W[i][j], a.W[f][j]);
34                  ans = MOD - ans;
35              }
36              for(int j = i + 1;j <= n;++ j) if(a.W[j][i]){
37                  while(a.W[j][i]){
38                      int u = a.W[i][i];
39                      int v = a.W[j][i];
40                      if(u > v){
41                          for(int k = 1;k <= n;++ k)
42                              swap(a.W[i][k], a.W[j][k]);
43                          ans = MOD - ans;
44                          swap(u, v);
45                      }
46                      int rate = v / u;
47                      for(int k = 1;k <= n;++ k){
48                          a.W[j][k] = (a.W[j][k] - 1ll * rate * a.W[i][k] %
                                MOD + MOD) % MOD;
49                      }
50                  }
51              }
52          }
53          for(int i = 1;i <= n;++ i)
54              ans = 1ll * ans * a.W[i][i] % MOD;
55          return ans;
56      }
57      int D[MAXN];
58      int W[MAXN][MAXN];
59      int main(){
60          int n, m, t;
61          cin >> n >> m >> t;
62          for(int i = 1;i <= m;++ i){
63              int u, v, w;
64              cin >> u >> v >> w;
65              if(u != v){
66                  if(t == 0){ // 无向图
67                      D[u] = (D[u] + w) % MOD;
68                      D[v] = (D[v] + w) % MOD;
69                      W[u][v] = (W[u][v] + w) % MOD;
70                      W[v][u] = (W[v][u] + w) % MOD;
71                  } else {    // 叶向树
```

```
                D[v] = (D[v] + w) % MOD;
                W[u][v] = (W[u][v] + w) % MOD;
            }
        }
    }
    Mat A(n - 1, n - 1);
    for(int i = 2;i <= n;++ i)
        for(int j = 2;j <= n;++ j)  // 以 1 为根的叶向树
            A.W[i - 1][j - 1] = MOD - W[i][j];
    for(int i = 2;i <= n;++ i)
        A.W[i - 1][i - 1] = (D[i] + A.W[i - 1][i - 1]) % MOD;
    cout << mat_det(A) << endl;
    return 0;
}
```

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。

```
#include<bits/stdc++.h>
using namespace std;
int power(int a, int b, int  p){
    int r = 1;
    while(b){
        if(b & 1) r = 1ll * r * a % p;
        b >>= 1,  a = 1ll * a * a % p;
    }
    return r;
}
namespace BSGS {
    unordered_map <int, int> M;
    int solve(int a, int y, int p){    // a ^ x = y (mod p)
        M.clear();
        int B = sqrt(p);
        int w1 = y, u1 = power(a, p - 2, p);
        int w2 = 1, u2 = power(a, B, p);
        for(int i = 0;i < B;++ i){
            M[w1] = i;
            w1 = 1ll * w1 * u1 % p;
        }
        for(int i = 0;i < p / B;++ i){
            if(M.count(w2)){
                return i * B + M[w2];
            }
            w2 = 1ll * w2 * u2 % p;
        }
        return -1;
    }
}
int main(){
    int p, b, n;
    cin >> p >> b >> n;
    int ans = BSGS :: solve(b, n, p);
    if(ans == -1){
        cout << "no solution\n";
    } else {
```

```
        cout << ans << "\n";
    }
    return 0;
}
```

6.3 中国剩余定理

6.3.1 定理

对于线性方程:

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

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:

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

```
#include<bits/stdc++.h>
using namespace std;
const int MAXN = 100 + 3;
long long A[MAXN], B[MAXN], M = 1;
long long exgcd(long long a, long long b, long long &x, long long &y){
    if(a == 0){
        x = 0, y = 1; return b;
    } else {
        long long x0 = 0, y0 = 0;
        long long d = exgcd(b % a, a, x0, y0);
        x = y0 - (b / a) * x0;
        y = x0;
        return d;
    }
}
int main(){
    int n;
    cin >> n;
    for(int i = 1;i <= n;++ i){
        cin >> B[i] >> A[i];
        M = M * B[i];
    }
    long long L = 0;
    for(int i = 1;i <= n;++ i){
        long long m = M / B[i], b, k;
        exgcd(m, B[i], b, k);
        L = (L + (__int128)A[i] * m * b) % M;
    }
    L = (L % M + M) % M;
    cout << L << endl;
    return 0;
}
```

6.4 狄利克雷前缀和

6.4.1 用法

计算:

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

```cpp
#include<bits/stdc++.h>
using namespace std;
const int MAXN = 2e7 + 3;
unsigned A[MAXN];
int p, P[MAXN]; bool V[MAXN];
void solve(int n){
    for(int i = 2;i <= n;++ i){
        if(!V[i]){
            P[++ p] = i;
            for(int j = 1;j <= n / i;++ j){ // 前缀和
                A[j * i] += A[j];
            }
        }
        for(int j = 1;j <= p && P[j] <= n / i;++ j){
            V[i * P[j]] = true;
            if(i % P[j] == 0)
                break;
        }
    }
}
unsigned seed;
inline unsigned read(){
    seed ^= seed << 13;
    seed ^= seed >> 17;
    seed ^= seed << 5;
    return seed;
}
int main(){
    int n;
    cin >> n >> seed;
    for(int i = 1;i <= n;++ i){
        A[i] = read();
    }
    solve(n);
    unsigned ans = 0;
    for(int i = 1;i <= n;++ i){
        ans ^= A[i];
    }
    cout << ans << endl;
    return 0;
}
```

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。

```
Matrix solve(ll n, ll A, ll B, ll C, Matrix R, Matrix U) {  // (0, 0)
    走到 (n, (An+B)/C)
    if (A >= C) return solve(n, A % C, B, C, U.qpow(A / C) * R, U);
    ll l = B / C, r = (A * n + B) / C;
    if (l == r) return R.qpow(n) * U.qpow(l);   // l = r -> l = r or A
        = 0 or n = 0.
    ll p = (C * r - B - 1) / A + 1;
    return R.qpow(n - p) * U * solve(r - l - 1, C, C - B % C + A - 1, A
        , U, R) * U.qpow(l);
}
```

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。

```
int exgcd(int a, int b, int &x, int &y){
    if(a == 0){
        x = 0, y = 1; return b;
    } else {
        int x0 = 0, y0 = 0;
        int d = exgcd(b % a, a, x0, y0);
        x = y0 - (b / a) * x0;
        y = x0;
        return d;
    }
}
```

6.7 快速离散对数

6.7.1 用法

给定原根 g 以及模数 mod，T 次询问 x 的离散对数。

复杂度 $\mathcal{O}(\mathrm{mod}^{2/3}+T\log\mathrm{mod})$。

```
#include<bits/stdc++.h>
using namespace std;
int power(int a, int b, int  p){
    int r = 1;
    while(b){
        if(b & 1) r = 1ll * r * a % p;
        b >>= 1,  a = 1ll * a * a % p;
    }
    return r;
}
namespace BSGS {
    unordered_map <int, int> M;
    int B, U, P, g;
```

```
    void init(int g, int P0, int B0){
        M.clear();
        B = B0;
        P = P0;
        U = power(power(g, B, P), P - 2, P);
        int w = 1;
        for(int i = 0;i < B;++ i){
            M[w] = i;
            w = 1ll * w * g % P;
        }
    }
    int solve(int y){
        int w = y;
        for(int i = 0;i < P / B;++ i){
            if(M.count(w)){
                return i * B + M[w];
            }
            w = 1ll * w * U % P;
        }
        return -1;
    }
}
const int MAXN = 1e5 + 3;
int H[MAXN], P[MAXN], H0, p, h, g, mod;
bool V[MAXN];
int solve(int x){
    if(x <= h){
        return H[x];
    }
    int v = mod / x, r = mod % x;
    if(r < x - r){
        return ((H0 + solve(r)) % (mod - 1) - H[v] + mod - 1) % (mod -
            1);
    } else {
        return (solve(x - r) - H[v + 1] + mod - 1) % (mod - 1);
    }
}
int main(){
    ios :: sync_with_stdio(false);
    cin.tie(nullptr);
    int T;
    cin >> g >> mod;
    h = sqrt(mod) + 1;
    BSGS :: init(g, mod, sqrt(1ll * mod * sqrt(mod) / log10(mod)));
    H0 = BSGS :: solve(mod - 1);
    H[1] = 0;
    for(int i = 2;i <= h;++ i){
        if(!V[i]){
            P[++ p] = i;
            H[i] = BSGS :: solve(i);
        }
        for(int j = 1;j <= p && P[j] <= h / i;++ j){
            int &p = P[j];
            H[i * p] = (H[i] + H[p]) % (mod - 1);
            V[i * p] = true;
            if(i % p == 0)
                break;
```

```
        }
    }
    cin >> T;
    while(T --){
        int x, tmp = 0;
        cin >> x;
        cout << solve(x) << "\n";
    }
    return 0;
}
```

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，$\mathcal{O}(m)$ 预处理，$\mathcal{O}(1)$ 单次查询 x, y 的最大公约数。

```
#include<bits/stdc++.h>
#define up(l, r, i) for(int i = l, END##i = r;i <= END##i;++ i)
#define dn(r, l, i) for(int i = r, END##i = l;i >= END##i;-- i)
using namespace std;
typedef long long i64;
const int INF = 2147483647;
const int MAXN= 5e3 + 3;
const int MAXT= 1e6 + 3;
const int MAXM= 1e3 + 3;
int G[MAXM][MAXM];
int T[MAXT][3];
int A[MAXN], B[MAXN], o = 1e6, h = 1e3, V[MAXT];
int tgcd(int a, int b){
    if(a <= h && b <= h) return G[a][b];
    return a == b ? a : 1;
}
int qgcd(int a, int b){
    int ans = 1;
    up(0, 2, i){
        if(T[b][i] > h){
            if(a % T[b][i] == 0) a /= T[b][i], ans *= T[b][i];
        } else {
            int d = G[a % T[b][i]][T[b][i]];
            a /= d, ans *= d;
        }
    }
    return ans;
}
const int MOD = 998244353;
int main(){
    ios :: sync_with_stdio(false);
    cin.tie(nullptr);
    up(1, h, i) G[0][i] = G[i][0] = i;
    up(1, h, i) up(1, h, j){
        if(i >= j) G[i][j] = G[i - j][j];
            else   G[i][j] = G[i][j - i];
    }
    up(2, o, i) if(!V[i]){
        V[i] = i;
        for(int j = 2;i * j <= o;++ j)
            if(!V[i * j]) V[i * j] = i;
```

```
    }
    T[1][0] = T[1][1] = T[1][2] = 1;
    up(2, o, i){
        int p = V[i];
        int a = T[i / p][0];
        int b = T[i / p][1];
        int c = T[i / p][2];
        int x, y, z;
        if(p ⩾ h){
            x = 1, y = i / p, z = p;
        } else {
            if(c * p ⩽ h){
                x = a, y = b, z = c * p;
            }
            else if(b * p ⩽ h){
                x = a, y = b * p, z = c;
                if(y > z) swap(y, z);
            }
            else if(a * p ⩽ h){
                x = a * p, y = b, z = c;
                if(x > y) swap(x, y);
                if(y > z) swap(y, z);
            } else {
                x = a * b, y = c, z = p;
                if(x > y) swap(x, y);
                if(y > z) swap(y, z);
                if(x > z) swap(x, z);
            }
        }
        T[i][0] = x;
        T[i][1] = y;
        T[i][2] = z;
    }
    int n;
    cin >> n;
    up(1, n, i) cin >> A[i];
    up(1, n, i) cin >> B[i];
    up(1, n, i){
        int s = 0, u = 1;
        up(1, n, j){
            int d = qgcd(A[i], B[j]);
            u = 1ll * u * i % MOD;
            s = (s + 1ll * d * u) % MOD;
        }
        printf("%d\n", s);
    }
    return 0;
}
```

6.9 原根

6.9.1 用法

计算 P 的最小原根。

6.9.2 附注

原根表，其中 $P = r \times 2^k$，对应原根为 g。

Prime	r	k	g	Prime	r	k	g
5	1	2	2	3221225473	3	30	5
17	1	4	3	75161927681	35	31	3
97	3	5	5	77309411329	9	33	7
193	3	6	5	206158430209	3	36	22
257	1	8	3	2061584302081	15	37	7
7681	15	9	17	2748779069441	5	39	3
12289	3	12	11	6597069766657	3	41	5
40961	5	13	3	39582418599937	9	42	5
65537	1	16	3	79164837199873	9	43	5
786433	3	18	10	263882790666241	15	44	7
5767169	11	19	3	1231453023109121	35	45	3
7340033	7	20	3	1337006139375617	19	46	3
23068673	11	21	3	3799912185593857	27	47	5
104857601	25	22	3	4222124650659841	15	48	19
167772161	5	25	3	7881299347898369	7	50	6
469762049	7	26	3	31525197391593473	7	52	3
1004535809	479	21	3	180143985094819841	5	55	6
2013265921	15	27	31	1945555039024054273	27	56	5
2281701377	17	27	3	4179340454199820289	29	57	3

```cpp
#include<bits/stdc++.h>
using namespace std;
int power(int a, int b, int p){
    int r = 1;
    while(b){
        if(b & 1) r = 1ll * r * a % p;
        b >>= 1,  a = 1ll * a * a % p;
    }
    return r;
}
int getphi(int x){
    int t = x, r = x;
    for(int i = 2;i <= x / i;++ i){
        if(t % i == 0){
            r = r / i * (i - 1);
            while(t % i == 0)
```

```
                t /= i;
            }
        }
        if(t != 1){
            r = r / t * (t - 1);
        }
        return r;
}
vector <int> getprime(int x){
        vector <int> p;
        int t = x;
        for(int i = 2;i <= x / i;++ i){
            if(t % i == 0){
                p.push_back(i);
                while(t % i == 0)
                    t /= i;
            }
        }
        if(t != 1)
            p.push_back(x);
        return p;
}
bool test(int g, int m, int mm, vector<int> &P){
        for(auto &p: P){
            if(power(g, mm / p, m) == 1)
                return false;
        }
        return true;
}
int get_genshin(int m){
        int mm = getphi(m);
        vector <int> P = getprime(mm);
        for(int i = 1;;++ i){
            if(test(i, m, mm, P))
                return i;
        }
}
int main(){
        cout << get_genshin(998244353) << endl;
        return 0;
}
```

6.10 快速乘法逆元（离线）

6.10.1 用法

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

```
#include<bits/stdc++.h>
using namespace std;
int power(int a, int b, int p){
        int r = 1;
        while(b){
            if(b & 1) r = 1ll * r * a % p;
            b >>= 1,  a = 1ll * a * a % p;
        }
        return r;
}
```

```cpp
const int MAXN = 5e6 + 3;
int A[MAXN], B[MAXN];
int P[MAXN], Q[MAXN];
int main(){
    ios :: sync_with_stdio(false);
    cin.tie(nullptr);
    int n, p, K, S = 1;
    cin >> n >> p >> K;
    P[0] = 1;
    for(int i = 1;i <= n;++ i){
        cin >> A[i];
        P[i] = 1ll * P[i - 1] * A[i] % p;
    }
    Q[n] = power(P[n], p - 2, p);
    for(int i = n;i >= 1;-- i){
        Q[i - 1] = 1ll * Q[i] * A[i] % p;
        B[i] = 1ll * Q[i] * P[i - 1] % p;
    }
    int ans = 0;
    for(int i = 1;i <= n;++ i){
        S = 1ll * S * K % p;
        ans = (ans + 1ll * S * B[i]) % p;
    }
    cout << ans << "\n";
    return 0;
}
```

6.11 快速乘法逆元（在线）

6.11.1 用法

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

```cpp
#include<bits/stdc++.h>
using namespace std;
const int MAXN = 1e7 + 3;
pair<int, int> F[MAXN], G[MAXN];
int I[MAXN];
using u32 = uint32_t;
u32 read(u32 &seed){
    seed ^= seed << 13;
    seed ^= seed >> 17;
    seed ^= seed << 5;
    return seed;
}
int main(){
    ios :: sync_with_stdio(false);
    cin.tie(nullptr);
    u32 seed;
    int n, p;
    cin >> n >> p >> seed;
    int m = pow(p, 1.0 / 3.0);
    I[1] = 1;
    for(int i = 2;i <= p / m;++ i){
        I[i] = 1ll * (p / i) * (p - I[p % i]) % p;
    }
    for(int i = 1;i < m;++ i){
        for(int j = i + 1;j <= m;++ j){
```

```
            if(!F[i * m * m / j].second){
                F[i * m * m / j] = { i, j };
                G[i * m * m / j] = { i, j };
            }
        }
    }
    F[    0] = G[    0] = { 0, 1 };
    F[m * m] = G[m * m] = { 1, 1 };
    for(int i = 1;i <      m * m;++ i) if(!F[i].second)
        F[i] = F[i - 1];
    for(int i = m * m - 1;i >= 1;-- i) if(!G[i].second)
        G[i] = G[i + 1];
    int lastans = 0;
    for(int i = 1;i <= n;++ i){
        int a, inv;
        a = (read(seed) ^ lastans) % (p - 1) + 1;
        int w = 1ll * a * m * m / p;
        auto &yy1 = F[w].second;    // *avoid y1 in <cmath>
        if(1ll * a * yy1 % p <= p / m){
            inv = 1ll * I[1ll * a * yy1 % p] * yy1 % p;
        } else {
            auto &yy2 = G[w].second;
            inv = 1ll * I[1ll * a * (p - yy2) % p] * (p - yy2) % p;
        }
        lastans = inv;
    }
    cout << lastans << "\n";
    return 0;
}
```

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：

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

下面代码先求出了多项式再计算 $f(k)$，也可以直接带入计算。

```
#include<bits/stdc++.h>
using namespace std;
const int MAXN = 2e3 + 3;
const int MOD  = 998244353;
int X[MAXN], Y[MAXN], F[MAXN], G[MAXN], H[MAXN], A[MAXN];
int power(int a, int b){
    int r = 1;
    while(b){
        if(b & 1) r = 1ll * r * a % MOD;
        b >>= 1,  a = 1ll * a * a % MOD;
    }
    return r;
}
int main(){
    int n, k;
    cin >> n >> k;
    for(int i = 1;i <= n;++ i){
```

```
        cin >> X[i] >> Y[i];
    }
    F[0] = 1;
    for(int i = 1;i <= n;++ i){      // 计算 prod(x - x_i)
        for(int j = 0;j <= n;++ j){
            G[j] = ((j == 0 ? 0 : F[j - 1]) - 1ll * F[j] * X[i] % MOD +
                MOD) % MOD;
        }
        for(int j = 0;j <= n;++ j){
            F[j] = G[j];
        }
    }
    for(int i = 1;i <= n;++ i){
        for(int j = 0;j <= n;++ j){
            G[j] = F[j];
        }
        for(int j = n;j >= 0;-- j){ // 计算 prod(x - x_j) / (x - x_i)
            H[j] = G[j + 1];
            G[j] = (G[j] + 1ll * H[j] * X[i]) % MOD;
        }
        int w = 1;                  // 计算 inv(prod(x_i - x_j))
        for(int j = 1;j <= n;++ j) if(j != i)
            w = 1ll * w * (X[i] - X[j] + MOD) % MOD;
        w = 1ll * power(w, MOD - 2) * Y[i] % MOD;
        for(int j = 0;j <= n;++ j)
            A[j] = (A[j] + 1ll * w * H[j]) % MOD;
    }
    int t = 1, ans = 0;
    for(int i = 0;i <= n - 1;++ i){
        ans = (ans + 1ll * A[i] * t) % MOD;
        t = 1ll * t * k % MOD;
    }
    cout << ans << endl;
}
```

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：

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

此外在数论上可以得到：

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

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 $x \bmod P = (x\times X)/2^{60}+S$。

```cpp
#include<bits/stdc++.h>
using namespace std;
long long S = 0, X = 0;
void init(int MOD){
    while((1 << (S + 1)) < MOD)
        S ++;
    X = ((__int128)1 << 60 + S) / MOD + !!(((__int128)1 << 60 + S) %
       MOD);
    cerr << S << " " << X << endl;
}
int power(long long x, int y, int MOD){
    long long r = 1;
    while(y){
        if(y & 1){
            r = r * x;
            r = r - MOD * ((__int128)r * X >> 60 + S);
        }
        x = x * x;
        x = x - MOD * ((__int128)x * X >> 60 + S);
        y >>= 1;
    }
    return r;
}
int main(){
    init(998244353);
    cout << power(2, 10, 998244353) << endl;
    cout << power(2, 20, 998244353) << endl;
    cout << power(2, 30, 998244353) << endl;
    cout << power(2, 40, 998244353) << endl;
    return 0;
}
```

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。

```cpp
#include<bits/stdc++.h>
using namespace std;
using i64 = long long;
const int INF  =  1e9;
const i64 INFL = 1e18;
i64 step(i64 a, i64 c, i64 m){
    return ((__int128)a * a + c) % m;
}
i64 multi(i64 a, i64 b, i64 m){
    return (__int128) a * b % m;
}
i64 power(i64 a, i64 b, i64 m){
    i64 r = 1;
    while(b){
        if(b & 1) r = multi(r, a, m);
        b >>= 1,  a = multi(a, a, m);
    }
    return r;
}
mt19937_64 MT;
bool test(i64 n){
    if(n < 3 || n % 2 == 0)
        return n == 2;
    i64 u = n - 1, t = 0;
    while(u % 2 == 0)
        u /= 2,
        t += 1;
    int test_time = 20;
    for(int i = 1; i <= test_time;++ i){
        i64 a = MT() % (n - 2) + 2;
        i64 v = power(a, u, n);
        if(v == 1){
            continue;
        }
        int s;
        for(s = 0;s < t;++ s){
            if(v == n - 1)
                break;
            v = multi(v, v, n);
        }
        if(s == t)
            return false;
    }
    return true;
}
basic_string<i64> rho(i64 n){
    if(n == 1)
        return {};
    if(test(n)){
        return {n};
    }
    i64 a  = MT() % (n - 1) + 1;
    i64 x1 = MT() % (n - 1);
    i64 x2 = x1;
    for(int i = 1;;i <<= 1){
        i64 tot = 1;
        for(int j = 1;j <= i;++ j){
```

```
            x2 = step(x2, a, n);
            tot = multi(tot, llabs(x1 - x2), n);
            if(j % 127 == 0){
                i64 d = __gcd(tot, n);
                if(d > 1)
                    return rho(d) + rho(n / d);
            }
        }
        i64 d = __gcd(tot, n);
        if(d > 1)
            return rho(d) + rho(n / d);
        x1 = x2;
    }
}
// ==== TEST ====
int main(){
    int T;
    cin >> T;
    for(int _ = 1;_ <= T;++ _){
        i64 n, p = 0;
        cin >> n;
        auto res = rho(n);
        for(auto &u : res)
            p = max(p, u);
        if(res.size() == 1)
            cout << "Prime" << endl;
        else
            cout << p << endl;
    }
    return 0;
}
```

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。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：

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

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：

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

```cpp
#include<bits/stdc++.h>
using namespace std;
const int MOD = 1e9 + 7;
int power(int a, int b){
    int r = 1;
    while(b){
        if(b & 1) r = 1ll * r * a % MOD;
        b >>= 1,  a = 1ll * a * a % MOD;
    }
    return r;
}
vector <tuple<int, int> > P;
void solve(int step, int n, int d, int f, int &ans){
    if(step == P.size()){
        ans = (ans + 1ll * power(n, n / d) * f) % MOD;
    } else {
        auto [w, c] = P[step];
        int dd = 1, ff = 1;
        for(int i = 0;i <= c;++ i){
            solve(step + 1, n, d * dd, f * ff, ans);
            ff = ff * (w - (i == 0));
            dd = dd * w;
        }
    }
}
int main(){
    int T;
    cin >> T;
    while(T --){
        int n, t;
        cin >> n;
        t = n;
        for(int i = 2;i * i <= n;++ i) if(n % i == 0){
            int w = i, c = 0;
            while(t % i == 0){
                t /= i, c ++;
```

```
            }
            P.push_back({ w, c });
        }
        if(t != 1){
            P.push_back({ t, 1 });
        }
        int ans = 0;
        solve(0, n, 1, 1, ans);
        ans = 1ll * ans * power(n, MOD - 2) % MOD;
        cout << ans << endl;
        P.clear();
    }
    return 0;
}
```

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：$G_k(n)=\sum_{i=1}^{n}[\text{mindiv}(i)>p_k \text{ or } \text{isprime}(i)]g(i)$ ($p_0=1$)，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 $f(1)$。

```
// 预处理，$1$ 所在的块也算进去了
namespace init {
    ll init_n, sqrt_n;
    vector<ll> np, p, id1, id2, val;
    ll cnt;
    void main(ll n) {
        init_n = n, sqrt_n = sqrt(n);
        ll M = sqrt_n * 2; // 筛出一个 > floor(sqrt(n)) 的质数，避免后续讨论边界
        np.resize(M + 1), p.resize(M + 1);
        for (ll i = 2; i <= M; ++i) {
            if (!np[i]) p[++p[0]] = i;
            for (ll j = 1; j <= p[0]; ++j) {
                if (i * p[j] > M) break;
                np[i * p[j]] = 1;
                if (i % p[j] == 0) break;
            }
        }
        p[0] = 1;
        id1.resize(sqrt_n + 1), id2.resize(sqrt_n + 1);
        val.resize(1);
        for (ll l = 1, r, v; l <= n; l = r + 1) {
```

```
            v = n / l, r = n / v;
            if (v ⩽ sqrt_n) id1[v] = ++cnt;
            else id2[init_n / v] = ++cnt;
            val.emplace_back(v);
        }
    }
    ll id(ll n) {
        if (n ⩽ sqrt_n) return id1[n];
        else return id2[init_n / n];
    }
}
using namespace init;
// 计算 $G_k$，两个参数分别是 $g$ 从 $2$ 开始的前缀和和 $g$
auto calcG = [&] (auto&& sum, auto&& g) → vector<ll> {
    vector<ll> G(cnt + 1);
    for (int i = 1; i ⩽ cnt; ++i) G[i] = sum(val[i]);
    ll pre = 0;
    for (int i = 1; p[i] * p[i] ⩽ n; ++i) {
        for (int j = 1; j ⩽ cnt; ++j) {
            if (p[i] * p[i] > val[j]) break;
            ll tmp = id(val[j] / p[i]);
            G[j] = (G[j] - g(p[i]) * (G[tmp] - pre)) % MD;
        }
        pre = (pre + g(p[i])) % MD;
    }
    for (int i = 1; i ⩽ cnt; ++i) G[i] = (G[i] % MD + MD) % MD;
    return G;
};
// 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。
function<ll(ll, int)> calcF = [&] (ll m, int k) {
    if (p[k] > m) return 0;
    ll ans = (fp[id(m)] - fp[id(p[k - 1])]) % MD;
    for (int h = k; p[h] * p[h] ⩽ m; ++h) {
        ll pc = p[h], c = 1;
        while (pc * p[h] ⩽ m) {
            ans = (ans + calcF(m / pc, h + 1) * f(p[h] ^ c)) % MD;
            ++c, pc = pc * p[h], ans = (ans + f(p[h] ^ c)) % MD;
        }
    }
    return ans;
};
```

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 $S(n) = \sum_{i=1}^{n} f(i)$：

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

6.18.2 例题

- 对于 $f = \varphi$，寻找 $g = 1, h = \mathrm{id}$；

- 对于 $f=\mu$，寻找 $g=1,h=\varepsilon$。

```cpp
#include<bits/stdc++.h>
using namespace std;
const int MAXN = 1e7 + 3;
const int H = 1e7;
int P[MAXN], p; bool V[MAXN];
long long ph[MAXN], sph[MAXN];
long long mu[MAXN], smu[MAXN];
long long tp[MAXN];
long long solve_ph(long long N){
    for(int d = N / H;d >= 1;-- d){
        long long n = N / d;
        long long wh = 1ll * n * (n + 1) / 2;
        tp[d] = wh;
        for(long long l = 2, r;l <= n;l = r + 1){
            r = n / (n / l);
            long long wg = r - l + 1;
            long long ws = n / l <= H ? sph[n / l] : tp[N / (n / l)];
            tp[d] -= wg * ws;
        }
    }
    return N <= H ? sph[N] : tp[1];
}
long long solve_mu(long long N){
    for(int d = N / H;d >= 1;-- d){
        long long n = N / d;
        long long wh = 1;
        tp[d] = wh;
        for(long long l = 2, r;l <= n;l = r + 1){
            r = n / (n / l);
            long long wg = r - l + 1;
            long long ws = n / l <= H ? smu[n / l] : tp[N / (n / l)];
            tp[d] -= wg * ws;
        }
    }
    return N <= H ? smu[N] : tp[1];
}
int main(){
    ios :: sync_with_stdio(false);
    cin.tie(nullptr);
    ph[1] = 1;
    mu[1] = 1;
    for(int i = 2;i <= H;++ i){
        if(!V[i]){
            P[++ p] = i;
            ph[i] = i - 1;
            mu[i] = -1;
        }
        for(int j = 1;j <= p && P[j] <= H / i;++ j){
            int &p = P[j];
            V[i * p] = true;
            if(i % p == 0){
                ph[i * p] = ph[i] * p;
                mu[i * p] = 0;
                break;
            } else {
                ph[i * p] = ph[i] * (p - 1);
```

```
                mu[i * p] = -mu[i];
            }
        }
    }
    for(int i = 1;i ⩽ H;++ i){
        sph[i] = sph[i - 1] + ph[i];
        smu[i] = smu[i - 1] + mu[i];
    }
    int T;
    cin >> T;
    while(T ⟶ 0){
        int n;
        cin >> n;
        cout << solve_ph(n) << " " << solve_mu(n) << "\n";
    }
    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 $g(p)=\mathrm{id}(p)\varphi(p)=f(p)$，根据 $g*\mathrm{id}=\mathrm{id}_2$ 利用杜教筛求解。$h(p^c)$ 的值利用递推式进行计算。

```
#include<bits/stdc++.h>
using namespace std;
const int MAXN = 1e7 + 3;
const int MAXM = 1e5 + 3;
const int H = 1e7;
const int MOD = 1e9 + 7;
const int DIV2 = 500000004;
const int DIV6 = 166666668;
int P[MAXN], p; bool V[MAXN];
int g[MAXN], le[MAXN], ge[MAXN];
int s1(long long n){    // 1^1 + 2^1 + ... + n^1
    n %= MOD;
    return 1ll * n * (n + 1) % MOD * DIV2 % MOD;
}
int s2(long long n){    // 1^2 + 2^2 + ... + n^2
    n %= MOD;
    return 1ll * n * (n + 1) % MOD * (2 * n + 1) % MOD * DIV6 % MOD;
}
int sg(long long n, long long N){
```

```
    return n ⩽ H ? le[n] : ge[N / n];
}
int sieve_du(long long N){
    for(int d = N / H;d ⩾ 1;-- d){
        long long n = N / d;
        int wh = s2(n);
        for(long long l = 2, r;l ⩽ n;l = r + 1){
            r = n / (n / l);
            int wg = (s1(r) - s1(l - 1) + MOD) % MOD;
            int ws = sg(n / l, N);
            ge[d] = (ge[d] + 1ll * wg * ws) % MOD;
        }
        ge[d] = (wh - ge[d] + MOD) % MOD;
    }
    return N ⩽ H ? le[N] : ge[1];
}
vector <int> hc[MAXM], gc[MAXM];
int ANS;
void sieve_pn(int last, long long x, int h, long long N){
    ANS = (ANS + 1ll * h * sg(N / x, N)) % MOD;
    for(long long i = last + 1;x ⩽ N / P[i] / P[i];++ i){
        int c = 2;
        for(long long t = x * P[i] * P[i];t ⩽ N;t *= P[i], c ++){
            int hh = 1ll * h * hc[i][c] % MOD;
            sieve_pn(i, t, hh, N);
        }
    }
}
int main(){
    ios :: sync_with_stdio(false);
    cin.tie(nullptr);
    g[1] = 1;
    for(int i = 2;i ⩽ H;++ i){
        if(!V[i]){
            P[++ p] = i, g[i] = 1ll * i * (i - 1) % MOD;
        }
        for(int j = 1;j ⩽ p && P[j] ⩽ H / i;++ j){
            int &p = P[j];
            V[i * p] = true;
            if(i % p == 0){
                g[i * p] = 1ll * g[i] * p % MOD * p % MOD;
                break;
            } else {
                g[i * p] = 1ll * g[i] * p % MOD * (p - 1) % MOD;
            }
        }
    }
    for(int i = 1;i ⩽ H;++ i){
        le[i] = (le[i - 1] + g[i]) % MOD;
    }
    long long N;
    cin >> N;
    for(int i = 1;i ⩽ p && 1ll * P[i] * P[i] ⩽ N;i ++){
        int &p = P[i];
        hc[i].push_back(1);
        gc[i].push_back(1);
        for(long long c = 1, t = p;t ⩽ N;t = t * p, ++ c){
```

```
            if(c == 1){
                gc[i].push_back(1ll * p * (p - 1) % MOD);
            } else {
                gc[i].push_back(1ll * gc[i].back() * p % MOD * p % MOD)
                    ;
            }
            int w = 1ll * (t % MOD) * ((t - 1) % MOD) % MOD;
            int s = 0;
            for(int j = 1;j <= c;++ j){
                s = (s + 1ll * gc[i][j] * hc[i][c - j]) % MOD;
            }
            hc[i].push_back((w - s + MOD) % MOD);
        }
    }
    sieve_du(N);
    sieve_pn(0, 1, 1, N);
    cout << ANS << "\n";
    return 0;
}
```

6.20 常用数表

6.20.1 分拆数表

n	10	20	30	40	50	60	70	80	90	100
$p(n)$	42	627	5604	37338	204226	966467	4087968	15796476	56634173	190569292

6.20.2 因数个数表

N	10^1	10^2	10^3	10^4	10^5	10^6	10^7	10^8	10^9
$\max d(n)$	4	12	32	64	128	240	448	768	1344
$\max \omega(n)$	2	3	4	5	6	7	8	8	9
N	10^{10}	10^{11}	10^{12}	10^{13}	10^{14}	10^{15}	10^{16}	10^{17}	10^{18}
$\max d(n)$	2304	4032	6720	10752	17280	26880	41472	64512	103680
$\max \omega(n)$	10	10	11	12	12	13	13	14	15

6.20.3 大质数

10^{18} 级别:

- $P = 10^{18} + 3$，好记。
- $P = 2924438830427668481$，可以进行 NTT，$P = 174310137655 \times 2^24 + 1$，原根为 3。

6.21 二次剩余

6.21.1 用法

多次询问，每次询问给定奇素数 p 以及 y，在 $\mathcal{O}(\log p)$ 复杂度计算 x 使得 $x^2 \equiv 0 \pmod p$ 或者无解。

```
#include<bits/stdc++.h>
using namespace std;
int power(int a, int b, int p){
    int r = 1;
    while(b){
        if(b & 1) r = 1ll * r * a % p;
        b >>= 1,  a = 1ll * a * a % p;
    }
    return r;
}
bool check(int x, int p){
    return power(x, (p - 1) / 2, p) == 1;
}
struct Node {
    int real, imag;
};
Node mul(const Node a, const Node b, int p, int v){
    int nreal = (1ll * a.real * b.real + 1ll * a.imag * b.imag % p * v)
        % p;
    int nimag = (1ll * a.real * b.imag + 1ll * a.imag * b.real) % p;
    return { (nreal), nimag };
}
Node power(Node a, int b, int p, int v){
    Node r = { 1, 0 };
    while(b){
        if(b & 1) r = mul(r, a, p, v);
        b >>= 1,  a = mul(a, a, p, v);
    }
    return r;
}
mt19937 MT;
void solve(int n, int p, int &x1, int &x2){
    if(n == 0){
        x1 = x2 = 0;
        return;
    }
    if(!check(n, p)){
        x1 = x2 = -1;
        return;
    }
    int a, t;
    do {
        a = MT() % p;
    }while(check(t = (1ll * a * a - n + p) % p, p));
    Node u = { a, 1 };
    x1 = power(u, (p + 1) / 2, p, t).real;
    x2 = (p - x1) % p;
    if(x1 > x2)
        swap(x1, x2);
}
int main(){
```

```
    ios :: sync_with_stdio(false);
    cin.tie(nullptr);
    int T;
    cin >> T;
    while(T --){
        int n, p, x1, x2;
        cin >> n >> p;
        solve(n, p, x1, x2);
        if(x1 == -1){
            cout << "Hola!\n";
        } else {
            if(x1 == x2){
                cout << x1 << "\n";
            } else {
                cout << x1 << " " << x2 << "\n";
            }
        }
    }
    return 0;
}
```

6.22 单位根反演

6.22.1 定理

给出单位根反演如下:

$$[d \mid n] = \frac{1}{d}\sum_{i=0}^{d-1}\omega_d^{ni}$$

因为题太难了不会做，所以没有例题，咕着。

7 多项式

7.1 NTT 全家桶

7.1.1 用法

多项式全家桶。

- 包含基础多项式算法：快速傅里叶变换（FFT）及其逆变换（IFFT）、快速数论变换（NTT）及其逆变换（INTT）；
- 包含基于 NTT 的扩展多项式算法：多项式乘法（MUL）、多项式乘法逆元（INV）、多项式微分（DIF）、多项式积分（INT）、多项式对数（LN）、多项式指数（EXP）、多项式开根（SQT）、多项式平移（即计算 $G(x) = F(x + c)$，SHF）。

```
#include<bits/stdc++.h>
#define up(l, r, i) for(int i = l, END##i = r;i <= END##i;++ i)
#define dn(r, l, i) for(int i = r, END##i = l;i >= END##i;-- i)
using namespace std;
using i64 = long long;
const int INF  =  1e9;
const i64 INFL = 1e18;
const int MOD = 998244353;
```

```
int power(int a, int b){
    int r = 1;
    while(b){
        if(b & 1) r = 1ll * r * a % MOD;
        b >>= 1,  a = 1ll * a * a % MOD;
    }
    return r;
}
int inv(int x){
    return power(x, MOD - 2);
}
const int MAX_ = (1 << 19) + 3;
struct cplx{
    double a, b; cplx(double _a = 0, double _b = 0) :a(_a), b(_b){}
    cplx operator +(cplx t){ return cplx(a + t.a, b + t.b); }
    cplx operator -(cplx t){ return cplx(a - t.a, b - t.b); }
    cplx operator *(cplx t){ return cplx(a * t.a - b * t.b, a * t.b + b
        * t.a); }
    cplx operator *(int t) { return cplx(a * t, b * t); }
};
const long double pi = acos(-1);
namespace Poly{
    void FFT(int n, cplx Z[]){
        static int W[MAX_];
        int l = 1; W[0] = 0;
        while (n >>= 1)
            up(0, l - 1, i)
                W[l++] = W[i] << 1 | 1, W[i] <<= 1;
        up(0, l - 1, i)
            if(W[i] > i) swap(Z[i], Z[W[i]]);
        for (n = l >> 1, l = 1;n;n >>= 1, l <<= 1){
            cplx* S = Z, o(cos(pi / l), sin(pi / l));
            up(0, n - 1, i){
                cplx s(1, 0);
                up(0, l - 1, j){
                    cplx x = S[j] + s * S[j + l];
                    cplx y = S[j] - s * S[j + l];
                    S[j] = x, S[j + l] = y, s = s * o;
                }
                S += l << 1;
            }
        }
    }
    void IFFT(int n, cplx Z[]){
        FFT(n, Z); reverse(Z + 1, Z + n);
        up(0, n - 1, i)
            Z[i].a /= 1.0 * n, Z[i].b /= 1.0 * n;
    }
    void NTT(int n, int Z[]){
        static int W[MAX_];
        int g = 3, l = 1;
        W[0] = 0;
        while (n >>= 1)
            up(0, l - 1, i)
                W[l++] = W[i] << 1 | 1, W[i] <<= 1;
        up(0, l - 1, i)
            if (W[i] > i)swap(Z[i], Z[W[i]]);
```

```
65          for (n = l >> 1, l = 1;n;n >>= 1, l <<= 1){
66              int* S = Z, o = power(g, (MOD - 1) / l / 2);
67              up(0, n - 1, i){
68                  int s = 1;
69                  up(0, l - 1, j){
70                      int x = (S[j] + 1ll * s * S[j + l] % MOD      ) %
                            MOD;
71                      int y = (S[j] - 1ll * s * S[j + l] % MOD + MOD) %
                            MOD;
72                      S[j] = x, S[j + l] = y;
73                      s = 1ll * s * o % MOD;
74                  }
75                  S += l << 1;
76              }
77          }
78      }
79      void INTT(int n, int Z[]){
80          NTT(n, Z); reverse(Z + 1, Z + n);
81          int o = inv(n);
82          up(0, n - 1, i)
83              Z[i] = 1ll * Z[i] * o % MOD;
84      }
85      void MUL(int n, int A[], int B[]){          // 乘法
86          NTT(n, A), NTT(n, B);
87          up(0, n - 1, i)
88              A[i] = 1ll * A[i] * B[i] % MOD;
89          INTT(n, A);
90      }
91      void INV(int n, int Z[], int T[]){          // 乘法逆
92          static int A[MAX_];
93          up(0, n - 1, i)
94              T[i] = 0;
95          T[0] = power(Z[0], MOD - 2);
96          for (int l = 1;l < n;l <<= 1){
97              up(    0, 2 * l - 1, i) A[i] = Z[i];
98              up(2 * l, 4 * l - 1, i) A[i] = 0;
99              NTT(4 * l, A), NTT(4 * l, T);
100             up(0, 4 * l - 1, i)
101                 T[i] = (2ll * T[i] - 1ll * A[i] * T[i] % MOD * T[i] %
                        MOD + MOD) % MOD;
102             INTT(4 * l, T);
103             up(2 * l, 4 * l - 1, i)
104                 T[i] = 0;
105         }
106     }
107     void DIF(int n, int Z[], int T[]){          // 微分
108         up(0, n - 2, i)
109             T[i] = 1ll * Z[i + 1] * (i + 1) % MOD;
110         T[n - 1] = 0;
111     }
112     void INT(int n, int c, int Z[], int T[]){   // 积分
113         up(1, n - 1, i)
114             T[i] = 1ll * Z[i - 1] * inv(i) % MOD;
115         T[0] = c;
116     }
117     void LN(int n, int* Z, int* T){             // 求对数
118         static int A[MAX_];
```

```
119         static int B[MAX_];
120         up(0, 2 * n - 1, i)
121             A[i] = B[i] = 0;
122         DIF(n, Z, A);
123         INV(n, Z, B);
124         MUL(2 * n, A, B);
125         INT(n, 0, A, T);
126     }
127     void EXP(int n, int* Z, int* T){          // 求指数
128         static int A[MAX_];
129         static int B[MAX_];
130         up(1, 2 * n - 1, i) T[i] = 0;
131         T[0] = 1;
132         for (int l = 1;l < n;l <<= 1){
133             LN (2 * l, T, A);
134             up(    0, 2 * l - 1, i)
135                 B[i] = (-A[i] + Z[i] + MOD) % MOD;
136             B[0] = (B[0] + 1) % MOD;
137             up(2 * l, 4 * l - 1, i)
138                 T[i] = B[i] = 0;
139             MUL(4 * l, T, B);
140         }
141     }
142     void SQT(int n, int* Z, int* T){          // 开根
143         static int A[MAX_];
144         static int B[MAX_];
145         up(1, 2 * n - 1, i) T[i] = 0;
146         T[0] = 1;
147         int o = inv(2);
148         for (int l = 1;l < n;l <<= 1){
149             INV(2 * l, T, A);
150             up(0, 2 * l - 1, i)
151                 B[i] = Z[i];
152             up(2 * l, 4 * l - 1, i)
153                 A[i] = B[i] = 0;
154             MUL(4 * l, A, B);
155             up(0, 2 * l - 1, i)
156                 T[i] = 1ll * (T[i] + A[i]) * o % MOD;
157         }
158     }
159     void SHF(int n, int c, int* Z, int* T){   // 平移
160         static int A[MAX_];
161         static int B[MAX_];
162         static int F[MAX_];
163         static int G[MAX_];
164         int o = 1;
165         up(1, n - 1, i)
166             F[i] = 1ll * F[i - 1] *     i  % MOD,
167             G[i] = 1ll * G[i - 1] * inv(i) % MOD;
168         up(0, n - 1, i)
169             A[i] = 1ll * Z[n - 1 - i] * F[n - 1 - i] % MOD;
170         up(0, n - 1, i){
171             B[i] = 1ll * G[i] * o % MOD;
172             o = 1ll * o * c % MOD;
173         }
174         int l = 1; while (l < 2 * n - 1) l <<= 1;
175         up(n, l - 1, i)
```

```
            A[i] = B[i] = 0;
        MUL(l, A, B);
        up(0, n - 1, i)
            T[n - 1 - i] = 1ll * G[n - 1 - i] * A[i] % MOD;
    }
}
```

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、异或带入下式：

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

```
#include<bits/stdc++.h>
#define up(l, r, i) for(int i = l, END##i = r;i <= END##i;++ i)
#define dn(r, l, i) for(int i = r, END##i = l;i >= END##i;-- i)
using namespace std;
using i64 = long long;
const int INF  =  1e9;
const i64 INFL = 1e18;
const int MOD = 998244353;
namespace Solve1{   // and 卷积
    void FWT(int n, int *A){
        for(int l = 1 << n, u = 2, v = 1;u <= l;u <<= 1, v <<= 1)
            for(int j = 0;j < l;j += u)
                for(int k = 0;k < v;++ k)
                    A[j + v + k] = (A[j + v + k] + A[j + k]) % MOD;
    }
    void IFWT(int n, int *A){
        for(int l = 1 << n, u = l, v = l / 2;u > 1;u >>= 1, v >>= 1)
            for(int j = 0;j < l;j += u)
                for(int k = 0;k < v;++ k)
                    A[j + v + k] = (A[j + v + k] - A[j + k] + MOD) %
                       MOD;
    }
}
namespace Solve2{   // or  卷积
    void FWT(int n, int *A){
        for(int l = 1 << n, u = 2, v = 1;u <= l;u <<= 1, v <<= 1)
            for(int j = 0;j < l;j += u)
                for(int k = 0;k < v;++ k)
                    A[j + k] = (A[j + k] + A[j + v + k]) % MOD;
    }
    void IFWT(int n, int *A){
        for(int l = 1 << n, u = l, v = l / 2;u > 1;u >>= 1, v >>= 1)
            for(int j = 0;j < l;j += u)
                for(int k = 0;k < v;++ k)
                    A[j + k] = (A[j + k] - A[j + v + k] + MOD) % MOD;
    }
}
namespace Solve3{   // xor 卷积
    void FWT(int n, int *A){
        for(int l = 1 << n, u = 2, v = 1;u <= l;u <<= 1, v <<= 1)
```

```
            for(int j = 0;j < l;j += u)
                for(int k = 0;k < v;++ k){
                    int a = A[j + k];
                    int b = A[j + v + k];
                    A[j + k    ] = (a + b + MOD) % MOD;
                    A[j + v + k] = (a - b + MOD) % MOD;
                }
    }
    void IFWT(int n, int *A){
        int div2 = (MOD + 1) / 2;
        for(int l = 1 << n, u = l, v = l / 2;u > 1;u >>= 1, v >>= 1)
            for(int j = 0;j < l;j += u)
                for(int k = 0;k < v;++ k){
                    int a = A[j + k];
                    int b = A[j + v + k];
                    A[j + k    ] = 1ll * (a + b + MOD) * div2 % MOD;
                    A[j + v + k] = 1ll * (a - b + MOD) * div2 % MOD;
                }
    }
}
```

7.3 任意模数 NTT

```
#include<bits/stdc++.h>
#define up(l, r, i) for(int i = l, END##i = r;i <= END##i;++ i)
#define dn(r, l, i) for(int i = r, END##i = l;i >= END##i;-- i)
using namespace std;
using i64 = long long;
const int INF  =  1e9;
const i64 INFL = 1e18;
const int MAX_ = (1 << 19) + 3;
template <typename T>
struct cplx0{
    T a, b; cplx0(T _a = 0, T _b = 0) :a(_a), b(_b){}
    cplx0 operator +(cplx0 t){ return cplx0(a + t.a, b + t.b); }
    cplx0 operator -(cplx0 t){ return cplx0(a - t.a, b - t.b); }
    cplx0 operator *(cplx0 t){ return cplx0(a * t.a - b * t.b, a * t.b
       + b * t.a); }
    cplx0 operator *(int t) { return cplx0(a * t, b * t); }
};
using cplx = cplx0<double>;
const long double pi = acos(-1);
namespace Poly{
    void FFT(int n, cplx Z[]){
        static int W[MAX_];
        int l = 1; W[0] = 0;
        while (n >>= 1)
            up(0, l - 1, i)
                W[l++] = W[i] << 1 | 1, W[i] <<= 1;
        up(0, l - 1, i)
            if(W[i] > i) swap(Z[i], Z[W[i]]);
        for (n = l >> 1, l = 1;n;n >>= 1, l <<= 1){
            cplx* S = Z;
            cplx0<long double> o(cosl(pi / l), sinl(pi / l));
            up(0, n - 1, i){
                cplx0<long double> s(1, 0);
                up(0, l - 1, j){
```

```
                    cplx x = S[j] + cplx(s.a, s.b) * S[j + l];
                    cplx y = S[j] - cplx(s.a, s.b) * S[j + l];
                    S[j] = x, S[j + l] = y, s = s * o;
                }
                S += l << 1;
            }
        }
    }
    void IFFT(int n, cplx Z[]){
        FFT(n, Z); reverse(Z + 1, Z + n);
        up(0, n - 1, i)
            Z[i].a /= 1.0 * n, Z[i].b /= 1.0 * n;
    }
}
const int MAXN  = (1 << 19) + 3;
const int BLOCK = 32768;
cplx A1[MAXN], A2[MAXN], B1[MAXN], B2[MAXN];
int n, m, L, mod;
cplx P[MAXN], Q[MAXN];
void FFTFFT(int L, cplx X[], cplx Y[]){
    for(int i = 0;i < L;++ i){
        P[i].a = X[i].a;
        P[i].b = Y[i].a;
    }
    Poly :: FFT(L, P);
    for(int i = 0;i < L;++ i){
        Q[i] = (i == 0 ? P[0] : P[L - i]);
        Q[i].b = -Q[i].b;
    }
    for(int i = 0;i < L;++ i){
        X[i] = (P[i] + Q[i]);
        Y[i] = (Q[i] - P[i]) * cplx(0, 1);
        X[i].a /= 2.0, X[i].b /= 2.0;
        Y[i].a /= 2.0, Y[i].b /= 2.0;
    }
}
int main(){
    ios :: sync_with_stdio(false);
    cin.tie(nullptr);
    cin >> n >> m >> mod;
    for(int i = 0;i <= n;++ i){
        int a; cin >> a;
        a %= mod;
        A1[i].a = a / BLOCK;
        A2[i].a = a % BLOCK;
    }
    for(int i = 0;i <= m;++ i){
        int a; cin >> a;
        a %= mod;
        B1[i].a = a / BLOCK;
        B2[i].a = a % BLOCK;
    }
    for(L = 1;L <= n + m;L <<= 1);
    FFTFFT(L, A1, A2);
    FFTFFT(L, B1, B2);
    for(int i = 0;i < L;++ i){
        P[i] = A1[i] * B1[i] + cplx(0, 1) * A2[i] * B1[i];
```

```
91          Q[i] = A1[i] * B2[i] + cplx(0, 1) * A2[i] * B2[i];
92      }
93      Poly :: IFFT(L, P);
94      Poly :: IFFT(L, Q);
95      for(int i = 0;i < L;++ i){
96          long long a1b1 = P[i].a + 0.5;
97          long long a2b1 = P[i].b + 0.5;
98          long long a1b2 = Q[i].a + 0.5;
99          long long a2b2 = Q[i].b + 0.5;
100         long long w = ((a1b1 % mod * (BLOCK * BLOCK % mod)) + ((a2b1 +
                a1b2) % mod) * BLOCK + a2b2) % mod;
101         if(i <= n + m)
102             cout << w << " ";
103     }
104     return 0;
105 }
```

8 字符串

8.1 AC 自动机

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  using i64 = long long;
4  const int MOD = 1e9 + 7;
5  namespace ACAM{
6      const int MAXN =1e6 + 3;
7      const int MAXM = 26 + 3;
8      int C[MAXN][MAXM], o;
9      void insert(char *S){
10         int p = 0, len = 0;
11         for(int i = 0;S[i];++ i){
12             int e = S[i] - 'a';
13             if(C[p][e]){
14                 p = C[p][e];
15             } else {
16                 p = C[p][e] = ++ o;
17             }
18             ++ len;
19         }
20     }
21     int F[MAXN];
22     void build(){
23         queue <int> Q; Q.push(0);
24         while(!Q.empty()){
25             int u = Q.front(); Q.pop();
26             for(int i = 0;i < 26;++ i){
27                 int v = C[u][i];
28                 if(v == 0)
29                     continue;
30                 int p = F[u];
31                 while(!C[p][i] && p != 0)
32                     p = F[p];
33                 if(C[p][i] && C[p][i] != v)
34                     F[v] = C[p][i];
35                 Q.push(v);
```

```
            }
        }
    }
}
```

8.2 扩展 KMP

8.2.1 定义

$$z_i^{(1)} = |\mathrm{lcp}(b, \mathrm{suffix}(b, i))|$$
$$z_i^{(2)} = |\mathrm{lcp}(b, \mathrm{suffix}(a, i))|$$

```cpp
#include<bits/stdc++.h>
using namespace std;
typedef long long i64;
const int MAXN = 2e7 + 3;
char A[MAXN], B[MAXN * 2];
int n, m, l, r, Z[MAXN * 2];
i64 ans1, ans2;
int main(){
    scanf("%s%s", A + 1, B + 1);
    n = strlen(A + 1);
    m = strlen(B + 1);
    l = 0, r = 0; Z[1] = 0, ans1 = m + 1;
    for(int i = 2;i <= m;++ i){
        if(i <= r) Z[i] = min(r - i + 1, Z[i - l + 1]);
        else       Z[i] = 0;
        while(B[Z[i] + 1] == B[i + Z[i]])
            ++ Z[i];
        if(i + Z[i] - 1 > r)
            r = i + Z[i] - 1, l = i;
        ans1 ^= 1ll * i * (Z[i] + 1);
    }
    l = 0, r = 0;
    Z[1] = 0, B[m + 1] = '#', strcat(B + 1, A + 1);
    for(int i = 2;i <= n + m + 1;++ i){
        if(i <= r) Z[i] = min(r - i + 1, Z[i - l + 1]);
        else       Z[i] = 0;
        while(B[Z[i] + 1] == B[i + Z[i]])
            ++ Z[i];
        if(i + Z[i] - 1 > r)
            r = i + Z[i] - 1, l = i;
    }
    for(int i = m + 2;i <= n + m + 1;++ i){
        ans2 ^= 1ll * (i - m - 1) * (Z[i] + 1);
    }
    printf("%lld\n%lld\n", ans1, ans2);
    return 0;
}
```

8.3 Manacher

```cpp
#include<bits/stdc++.h>
using namespace std;
using i64 = long long;
```

```
const int INF  =  1e9;
const i64 INFL = 1e18;
const int MAXN= 2.2e7 + 11;
char S[MAXN], T[MAXN]; int n, R[MAXN];
int main(){
    scanf("%s", S + 1);
    n = strlen(S + 1);
    for(int i = 1;i ⩽ n;++ i){
        T[2 * i - 1] = S[i];
        T[2 * i    ] =  '#';
    }
    T[0] = '#';
    n = 2 * n;
    int p = 0, x = 0, ans = 0;
    for(int i = 1;i ⩽ n;++ i){
        if(i ⩽ p) R[i] = min(R[2 * x - i], p - i);
        while(i - R[i] - 1 ⩾ 0 && T[i + R[i] + 1] == T[i - R[i] - 1])
            ++ R[i];
        if(i + R[i] > p){
            p = i + R[i];
            x = i;
        }
        ans = max(ans, R[i]);
    }
    printf("%d\n", ans);
    return 0;
}
```

8.4 回文自动机

```
#include<bits/stdc++.h>
using namespace std;
const int MAXM = 26 + 3;
namespace PAM{
    const int SIZ = 5e5 + 3;
    int n, s, F[SIZ], L[SIZ], D[SIZ];
    int M[SIZ][MAXM];
    char S[SIZ];
    void init(){
        S[0] = '$', n = 1;
        F[s = 0] = -1, L[0] = -1, D[0] = 0;
        F[s = 1] =  0, L[1] =  0, D[1] = 0;
    }
    void extend(int &last, char c){
        S[++ n] = c;
        int e = c - 'a';
        int a = last;
        while(c ≠ S[n - 1 - L[a]])
            a = F[a];
        if(M[a][e]){
            last = M[a][e];
        } else {
            int cur = M[a][e] = ++ s;
            L[cur] = L[a] + 2;
            if(a == 0){
                F[cur] = 1;
            } else {
```

```
                int b = F[a];
                while(c != S[n - 1 - L[b]])
                    b = F[b];
                F[cur] = M[b][e];
            }
            D[cur] = D[F[cur]] + 1;
            last = cur;
        }
    }
}
const int MAXN = 5e5 + 3;
char T[MAXN];
int main(){
    PAM :: init();
    int m = 0, last = 0, lastans = 0;
    for(char c = getchar();isalpha(c);c = getchar()){
        char d = (c - 97 + lastans) % 26 + 97;
        PAM :: extend(last, d);
        cout << (lastans = PAM :: D[last]) << " ";
    }
    return 0;
}
/*
azzzyyzyyx
1 2 1 2 3 2 2 2 3 3
*/
```

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）

```
#include<bits/stdc++.h>
using namespace std;
using i64 = long long;
const int INF  =  1e9;
const i64 INFL = 1e18;
const int MAXN = 1e6 + 3;
int n, m;
int A[MAXN], B[MAXN];
int C[MAXN], R[MAXN], P[MAXN], Q[MAXN];
char S[MAXN];
int main(){
    scanf("%s", S), n = strlen(S), m = 256;
    for(int i = 0;i < n;++ i) R[i] = S[i];
    for (int k = 1;k <= n;k <<= 1){
        for(int i = 0;i < n;++ i){
            Q[i] = ((i + k > n - 1) ? 0 : R[i + k]);
            P[i] = R[i];
            m = max(m, R[i]);
        }
#define fun(a, b, c) \
        memset(C, 0, sizeof(int) * (m + 1));            \
        for(int i = 0;i <  n;++ i) C[a] +=       1;     \
```

```
        for(int i = 1;i <= m;++ i) C[i] += C[i - 1];    \
        for(int i = n - 1;i >= 0;-- i) c[-- C[a]] = b;
        fun(Q[  i ],   i , B)
        fun(P[B[i]], B[i], A)
#undef fun
        int p = 1; R[A[0]] = 1;
        for(int i = 1;i <= n - 1;++ i){
            bool f1 = P[A[i]] == P[A[i - 1]];
            bool f2 = Q[A[i]] == Q[A[i - 1]];
            R[A[i]] = f1 && f2 ? R[A[i - 1]] : ++ p;
        }
        if (m == n) break;
    }
    for(int i = 0;i < n;++ i)
        printf("%u ", A[i] + 1);
    return 0;
}
```

8.7 后缀数组（SAIS）

```
#include<bits/stdc++.h>
using namespace std;
using i64 = long long;
const int INF  =  1e9;
const i64 INFL = 1e18;
const int MAXN = 1e6 + 3;
const int MAXM = 256 + 3;
#define LTYPE 0
#define STYPE 1
void induce_sort(int n, int S[], int T[], int m, int LM[], int SA[],
   int C[]){
    vector <int> BL(n);
    vector <int> BS(n);
    vector <int> BM(n);
    fill(SA, SA + n, -1);
    for(int i = 0;i < n;++ i){      // 预处理桶
        BM[i] = BS[i] = C[i] - 1;
        BL[i] = i == 0 ? 0 : C[i - 1];
    }
    for(int i = m - 1;i >= 0;-- i)  // 放置 LMS 后缀
        SA[BM[S[LM[i]]] --] = LM[i];
    for(int i =     0, p;i  < n;++ i)   // 计算 L 类型后缀的位置
        if(SA[i] > 0 && T[p = SA[i] - 1] == LTYPE)
            SA[BL[S[p]] ++] = p;
    for(int i = n - 1, p;i >= 0;-- i)   // 计算 S 类型后缀的位置
        if(SA[i] > 0 && T[p = SA[i] - 1] == STYPE)
            SA[BS[S[p]] --] = p;
}
// 长度 n，字符集 [0, n)，要求最后一个元素为 0
// 例如输入 ababa 传入 n = 6, S = [1 2 1 2 1 0]
void sais(int n, int S[], int SA[]){
    vector <int> T(n);
    vector <int> C(n);
    vector <int> I(n, -1);
    T[n - 1] = STYPE;
    for(int i = n - 2;i >= 0;-- i){     // 递推类型
```

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

```
    }
    return 0;
}
```

8.8 广义后缀自动机（离线）

```
#include<bits/stdc++.h>
using namespace std;
using i64 = long long;
const int INF  =  1e9;
const i64 INFL = 1e18;
const int MAXM= 26 + 3;
namespace SAM{
    const int SIZ = 2e6 + 3;
    int M[SIZ][MAXM];
    int L[SIZ], F[SIZ], S[SIZ];
    int s = 0, h = 25;
    void init(){
        F[0] = -1, s = 0;
    }
    void extend(int &last, char c){
        int e = c - 'a';
        int cur = ++ s;
        L[cur] = L[last] + 1;
        int p = last;
        while(p != -1 && !M[p][e])
            M[p][e] = cur, p = F[p];
        if(p == -1){
            F[cur] = 0;
        } else {
            int q = M[p][e];
            if(L[p] + 1 == L[q]){
                F[cur] = q;
            } else {
                int clone = ++ s;
                L[clone] = L[p] + 1;
                F[clone] = F[q];
                for(int i = 0;i <= h;++ i)
                    M[clone][i] = M[q][i];
                while(p != -1 && M[p][e] == q)
                    M[p][e] = clone, p = F[p];
                F[cur] = F[q] = clone;
            }
        }
        last = cur;
    }
    void solve(){
        i64 ans = 0;
        for(int i = 1;i <= s;++ i)
            ans += L[i] - L[F[i]];
        cout << ans << endl;
    }
}
namespace Trie{
    const int SIZ = 1e6 + 3;
    int M[SIZ][MAXM], s, h = 25;
    void insert(char *S){
```

```
        int p = 0;
        for(int i = 0;S[i];++ i){
            int e = S[i] - 'a';
            if(M[p][e]){
                p = M[p][e];
            } else
                p = M[p][e] = ++ s;
        }
    }
    int O[SIZ];
    void build_sam(){
        queue <int> Q;
        Q.push(0);
        while(!Q.empty()){
            int u = Q.front(); Q.pop();
            for(int i = 0;i <= h;++ i){
                char c = i + 'a';
                if(M[u][i]){
                    int v = M[u][i];
                    O[v] = O[u];
                    SAM :: extend(O[v], c);
                    Q.push(v);
                }
            }
        }
    }
}
const int MAXN = 1e6 + 3;
char S[MAXN];
int main(){
    SAM :: init();
    int n, last = 0;
    cin >> n;
    for(int i = 1;i <= n;++ i){
        scanf("%s", S);
        Trie :: insert(S);
    }
    Trie :: build_sam();
    SAM :: solve();
    cout << SAM :: s + 1 << endl;
    return 0;
}
```

8.9 广义后缀自动机（在线）

```
#include<bits/stdc++.h>
using namespace std;
using i64 = long long;
const int INF  =  1e9;
const i64 INFL = 1e18;
const int MAXM= 26 + 3;
namespace SAM{
    const int SIZ = 2e6 + 3;
    int M[SIZ][MAXM];
    int L[SIZ], F[SIZ], S[SIZ];
    int s = 0, h = 25;
    void init(){
```

```
13          F[0] = -1, s = 0;
14      }
15      void extend(int &last, char c){
16          int e = c - 'a';
17          if(M[last][e]){
18              int p = last;
19              int q = M[last][e];
20              if(L[q] == L[last] + 1){
21                  last = q;
22              } else {
23                  int clone = ++ s;
24                  L[clone] = L[p] + 1;
25                  F[clone] = F[q];
26                  for(int i = 0;i <= h;++ i)
27                      M[clone][i] = M[q][i];
28                  while(p != -1 && M[p][e] == q)
29                      M[p][e] = clone, p = F[p];
30                  F[q] = clone;
31                  last = clone;
32              }
33          } else {
34              int cur = ++ s;
35              L[cur] = L[last] + 1;
36              int p = last;
37              while(p != -1 && !M[p][e])
38                  M[p][e] = cur, p = F[p];
39              if(p == -1){
40                  F[cur] = 0;
41              } else {
42                  int q = M[p][e];
43                  if(L[p] + 1 == L[q]){
44                      F[cur] = q;
45                  } else {
46                      int clone = ++ s;
47                      L[clone] = L[p] + 1;
48                      F[clone] = F[q];
49                      for(int i = 0;i <= h;++ i)
50                          M[clone][i] = M[q][i];
51                      while(p != -1 && M[p][e] == q)
52                          M[p][e] = clone, p = F[p];
53                      F[cur] = F[q] = clone;
54                  }
55              }
56              last = cur;
57          }
58      }
59      void solve(){
60          i64 ans = 0;
61          for(int i = 1;i <= s;++ i)
62              ans += L[i] - L[F[i]];
63          cout << ans << endl;
64      }
65  }
66  const int MAXN = 1e6 + 3;
67  char S[MAXN];
68  int main(){
69      SAM :: init();
```

```
    int n, last = 0;
    cin >> n;
    for(int i = 1;i <= n;++ i){
        scanf("%s", S);
        int m = strlen(S);
        last = 0;
        for(int j = 0;j < m;++ j){
            SAM :: extend(last, S[j]);
        }
    }
    SAM :: solve();
    cout << SAM :: s + 1 << endl;
    return 0;
}
```

8.10 后缀自动机

```
#include<bits/stdc++.h>
using namespace std;
using i64 = long long;
const int INF  =  1e9;
const i64 INFL = 1e18;
const int MAXM= 26 + 3;
namespace SAM{
    const int SIZ = 2e6 + 3;
    int M[SIZ][MAXM];
    int L[SIZ], F[SIZ], S[SIZ];
    int last = 0, s = 0, h = 25;
    void init(){
        F[0] = -1, last = s = 0;
    }
    void extend(char c){
        int cur = ++ s, e = c - 'a';
        L[cur] = L[last] + 1;
        S[cur] = 1;
        int p = last;
        while(p != -1 && !M[p][e])
            M[p][e] = cur, p = F[p];
        if(p == -1){
            F[cur] = 0;
        } else {
            int q = M[p][e];
            if(L[p] + 1 == L[q]){
                F[cur] = q;
            } else {
                int clone = ++ s;
                L[clone] = L[p] + 1;
                F[clone] = F[q];
                S[clone] = 0;
                for(int i = 0;i <= h;++ i)
                    M[clone][i] = M[q][i];
                while(p != -1 && M[p][e] == q)
                    M[p][e] = clone, p = F[p];
                F[cur] = F[q] = clone;
            }
        }
        last = cur;
```

```
    }
    vector <int> E[SIZ];
    void build(){
        for(int i = 1;i <= s;++ i){
            E[F[i]].push_back(i);
        }
    }
    i64 ans = 0;
    void dfs(int u){
        for(auto &v : E[u]){
            dfs(v), S[u] += S[v];
        }
        if(S[u] > 1)
            ans = max(ans, 1ll * S[u] * L[u]);
    }
}
const int MAXN = 1e6 + 3;
char S[MAXN];
int main(){
    SAM :: init();
    scanf("%s", S); int n = strlen(S);
    for(int i = 0;i < n;++ i)
        SAM :: extend(S[i]);
    SAM :: build( );
    SAM :: dfs  (0);
    printf("%lld\n", SAM :: ans);
    return 0;
}
```

8.11 字典树

```
#include<bits/stdc++.h>
using namespace std;
using i64 = long long;
const int INF  =  1e9;
const i64 INFL = 1e18;
const int MAXM= 10 + 3;
namespace Trie{
    const int SIZ = 1e6 + 3;
    int M[SIZ][MAXM], s, h = 10;
    void extend(int &last, char c){
        int e = c - 'a';
        if(M[last][e]){
            last = M[last][e];
        } else {
            last = M[last][e] = ++ s;
        }
    }
    void insert(char *S){
        int p = 0;
        for(int i = 0;S[i];++ i){
            int e = S[i] - 'a';
            if(M[p][e]){
                p = M[p][e];
            } else
                p = M[p][e] = ++ s;
        }
```

```
    }
}
```

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。

```
#include<bits/stdc++.h>
using namespace std;
using i64 = long long;
const int MAXN = 2e5 + 3;
int X[MAXN], Y[MAXN];
struct Frac {
    int a, b;
    Frac (int _a, int _b){
        if(_b < 0){
            a = -_a, b = -_b;
        } else {
            a =  _a, b =  _b;
        }
    }
};
struct Node {
    int x, y;
}P[MAXN];
bool operator < (const Frac A, const Frac B){
    return 1ll * A.a * B.b - 1ll * A.b * B.a < 0;
}
bool operator < (const Node A, const Node B){
    return A.x == B.x ? A.y > B.y : A.x < B.x;
}
const Frac intersect(Node A, Node B){
    int a = B.y - A.y;
    int b = A.x - B.x;
    assert(b != 0);
    if(b < 0){
        a = -a, b = -b;
    }
    return Frac(a, b);
}
bool F[MAXN];
int main(){
    int TT;
    cin >> TT;
    while(TT -- ){
        int n;
        cin >> n;
        int maxx = -1e9, minx = 1e9;
        for(int i = 1;i <= n;++ i){
            auto &[x, y] = P[i];
            cin >> x >> y;
            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.size() - 1]];
57                      auto &[x2, y2] = P[Q1[Q1.size() - 2]];
58                      long long cmp = 1ll * (y - y1) * (x1 - x2) - 1ll *
                            (x - x1) * (y1 - y2);
59                      if(cmp > 0){
60                          Q1.pop_back();
61                      } else break;
62                  }
63                  Q1.push_back(i);
64              }
65              if(Q2.size() ⩽ 1){
66                  Q2.push_back(i);
67              } else {
68                  while(Q2.size() ⩾ 2){
69                      auto &[x1, y1] = P[Q2[Q2.size() - 1]];
70                      auto &[x2, y2] = P[Q2[Q2.size() - 2]];
71                      long long cmp = 1ll * (y - y1) * (x1 - x2) - 1ll *
                            (x - x1) * (y1 - y2);
72                      if(cmp < 0){
73                          Q2.pop_back();
74                      } else break;
75                  }
76                  Q2.push_back(i);
77              }
78          }
79          Q = Q1;
80          for(int i = Q2.size();i ≠ 0;i --){
81              if(i ≠ Q2.size())
82                  Q.push_back(Q2[i - 1]);
83          }
84          long long area = 0;
85          int x0 = P[Q[0]].x;
86          int y0 = P[Q[0]].y;
87          for(int i = 1;i + 1 < Q.size();++ i){
88              auto &[x1, y1] = P[Q[    i]];
89              auto &[x2, y2] = P[Q[i + 1]];
90              area += 1ll * (x1 - x0) * (y2 - y0) - 1ll * (x2 - x0) * (y1
                    - y0);
91          }
92          area = -area;
93          for(auto &i: Q1) F[i] = true;
94          for(auto &i: Q2) F[i] = true;
95          bool ok = false;
96          for(int i = 1;i ⩽ n;++ i) if(!F[i]){
97              ok = true;
98              maxx = max(maxx, P[i].x);
99              minx = min(minx, P[i].x);
```

```
100         }
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()].x)
111                 continue;
112             while(L1.size() >= 2){
113                 auto &P1 = P[L1[L1.size() - 1]];
114                 auto &P2 = P[L1[L1.size() - 2]];
115                 Frac i1 = intersect(P1, P[i]);
116                 Frac i2 = intersect(P2, P[i]);
117                 if(i1 < i2){
118                     L1.pop_back();
119                 } else break;
120             }
121             L1.push_back(i);
122         }
123         // L2 插入 kx + b 维护上凸壳
124         for(int i = n;i >= 1;-- i) if(!F[i]){
125             auto &[k, b] = P[i];
126             if(!L2.empty() && k == P[L2.back()].x)
127                 continue;
128             while(L2.size() >= 2){
129                 auto &P1 = P[L2[L2.size() - 1]];
130                 auto &P2 = P[L2[L2.size() - 2]];
131                 Frac i1 = intersect(P1, P[i]);
132                 Frac i2 = intersect(P2, P[i]);
133                 if(i1 < i2){
134                     L2.pop_back();
135                 } else break;
136             }
137             L2.push_back(i);
138         }
139         vector <Frac> E1;
140         E1.push_back(Frac( -2e9, 1 ));
141         for(int i = 0;i + 1 < L1.size();++ i){
142             auto &P1 = P[L1[i    ]];
143             auto &P2 = P[L1[i + 1]];
144             E1.push_back(intersect(P1, P2));
145         }
146         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]];
```

```
            long long w = 1ll * x2 * y1 - 1ll * x1 * y2;
            int A = y2 - y1;
            int B = x1 - x2;
            int x = 0, y = 0;
            if(B == 0){
                if(A > 0){
                    x = minx, y = 0;
                } else {
                    x = maxx, y = 0;
                }
            } else
            if(B <  0){
                Frac K = Frac(-A, -B);
                int p = 0;
                for(int k = 20;k >= 0;-- k){
                    int pp = p | 1 << k;
                    if(pp < E1.size() && E1[pp] < K){
                        p = pp;
                    }
                }
                x = P[L1[p]].x;
                y = P[L1[p]].y;
            } else {
                Frac K = Frac( A,  B);
                int p = 0;
                for(int k = 20;k >= 0;-- k){
                    int pp = p | 1 << k;
                    if(pp < E2.size() && E2[pp] < K){
                        p = pp;
                    }
                }
                x = P[L2[p]].x;
                y = P[L2[p]].y;
            }
            ans = max(ans, area - (w + 1ll * A * x + 1ll * B * y));
        }
        // cerr << "ans = " << ans << endl;
        cout << ans << "\n";
    }
    return 0;
}
```

9.2 最小圆覆盖

```
#include "2d.cpp"
point geto(point a, point b, point c) {
    double a1, a2, b1, b2, c1, c2;
    point ans(0, 0);
    a1 = 2 * (b.x - a.x), b1 = 2 * (b.y - a.y),
    c1 = sqr(b.x) - sqr(a.x) + sqr(b.y) - sqr(a.y);
    a2 = 2 * (c.x - a.x), b2 = 2 * (c.y - a.y),
    c2 = sqr(c.x) - sqr(a.x) + sqr(c.y) - sqr(a.y);
    if (equal(a1, 0)) {
        ans.y = c1 / b1;
        ans.x = (c2 - ans.y * b2) / a2;
    } else if (equal(b1, 0)) {
        ans.x = c1 / a1;
```

```
        ans.y = (c2 - ans.x * a2) / b2;
    } else {
        ans.x = (c2 * b1 - c1 * b2) / (a2 * b1 - a1 * b2);
        ans.y = (c2 * a1 - c1 * a2) / (b2 * a1 - b1 * a2);
    }
    return ans;
}
mt19937 MT;
circ minimal(vector <point> V){
    shuffle(V.begin(), V.end(), MT);
    point  o = V[0];
    double r = 0;
    for(int i = 0;i < V.size();++ i) {
        if (sign(dis(o, V[i]) - r) == 1) continue;
        o.x = (V[i].x + V[0].x) / 2;
        o.y = (V[i].y + V[0].y) / 2;
        r = dis(V[i], V[0]) / 2;
        for(int j = 0;j < i;++ j) {
            if (sign(dis(o, V[j]) - r) == 1) continue;
            o.x = (V[i].x + V[j].x) / 2;
            o.y = (V[i].y + V[j].y) / 2;
            r = dis(V[i], V[j]) / 2;
            for(int k = 0;k < j;++ k) {
                if (sign(dis(o, V[k]) - r) == 1) continue;
                o = geto(V[i], V[j], V[k]);
                r = dis(o, V[i]);
            }
        }
    }
    circ res;
    res.o = o;
    res.r = r;
    return res;
}
```

9.3 最左转线

```
#include "2d.cpp"
namespace DSU{
    const int MAXN = 1e5 + 3;
    int F[MAXN];
    int getfa(int u){
        return u == F[u] ? u : F[u] = getfa(F[u]);
    }
}
namespace Dual{
    const int MAXN = 1e5 + 3;
    const int MAXM = 1e5 + 3;
    int A[MAXM], B[MAXM], W[MAXM], I[MAXM], n, m;
    int outer;
    bool cmp(int a, int b){
        return W[a] < W[b];
    }
    vector <pair<int, int> > E[MAXN];
    const int MAXT = 20 + 3;
    int F[MAXN][MAXT], G[MAXN][MAXT], D[MAXN], h = 20;
    void dfs(int u, int f){
```

```
21          D[u] = D[f] + 1;
22          for(int i = 1;i ⩽ h;++ i)
23              F[u][i] = F[F[u][i - 1]][i - 1],
24              G[u][i] = max(G[u][i - 1], G[F[u][i - 1]][i - 1]);
25          for(auto &[v, w] : E[u]) if(v ≠ f){
26              G[v][0] = w;
27              F[v][0] = u;
28              dfs(v, u);
29          }
30      }
31      void build(){
32          for(int i = 1;i ⩽ n;++ i)
33              DSU :: F[i] = i;
34          for(int i = 1;i ⩽ m;++ i)
35              I[i] = i;
36          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 - P[a].x) * (P[b].y - P[
                a].y);
83      }
84      double scanx;
85      struct Cmp1{
86          bool operator ()(const pair<edge, int> l1, const pair<edge, int
                > l2) const{
87              const edge &e1 = l1.first;
88              const edge &e2 = l2.first;
89              double h1 = gety(get<0>(e1), get<1>(e1), scanx);
90              double h2 = gety(get<0>(e2), get<1>(e2), scanx);
91              return h1 < h2;
92          };
93      };
94      struct Cmp2{
95          bool operator ()(const pair<edge, int> l1, const pair<edge, int
                > l2) const{
96              if(l1.second == l2.second)
97                  return false;
98              const edge &e1 = l1.first;
99              const edge &e2 = l2.first;
100             vec v1 = P[get<1>(e1)] - P[get<0>(e1)];
101             vec v2 = P[get<1>(e2)] - P[get<0>(e2)];
102             if(sign(v1.y) != sign(v2.y)){
103                 return v1.y > 0;
104             } else {
105                 return sign(mulx(v1, v2)) == 1;
106             }
107         };
108     };
109     vector <pair<edge, int> > E[MAXN];
110     vector <int> G[MAXG];
111     int L[MAXE], R[MAXE], W[MAXE], n, m, q, o;
112     double theta;
113     int outer;
114     void rotate(){
115         srand(time(0));
116         theta = PI * rand() / RAND_MAX;
117     }
118     int add(double x, double y){
119         srand(time(0));
120         P[++ n] = rotate(vec(x, y), theta);
121         return n;
122     }
123     int link(int u, int v, int w){
124         ++ m;
125         E[u].push_back({{u, v}, ++ o});
126         L[o] = u, R[o] = v, W[o] = w;
127         E[v].push_back({{v, u}, ++ o});
128         L[o] = v, R[o] = u, W[o] = w;
129         return m;
130     }
131     int I[MAXE];
```

```
132     int polys;
133     pair<edge, int> findleft(int l, int r){
134         auto it = lower_bound(E[r].begin(), E[r].end(), make_pair(edge(
                r, l), 0), Cmp2());
135         if(it == E[r].begin())
136             return E[r].back();
137         else
138             return *(it - 1);
139     }
140     void leftmost(){
141         for(int i = 1;i <= n;++ i){
142             sort(E[i].begin(), E[i].end(), Cmp2());
143         }
144         for(int p = 1;p <= n;++ p){
145             for(auto &[e1, id1] : E[p]){
146                 auto &[x, y] = e1;
147                 if(!I[id1]){
148                     int l = x;
149                     int r = y;
150                     I[id1] = ++ polys;
151                     G[polys].push_back(id1);
152                     while(r != p){
153                         auto [e2, id2] = findleft(l, r);
154                         auto [a, b] = e2;
155                         I[id2] = polys;
156                         G[polys].push_back(id2);
157                         l = r;
158                         r = b;
159                     }
160                 }
161             }
162         }
163         for(int i = 1;i <= polys;++ i){
164             double area = 0;
165             for(int j = 0;j < G[i].size();++ j){
166                 area += mulx(P[L[G[i][j]]], P[R[G[i][j]]]);
167             }
168             if(area < 0)
169                 outer = i;
170         }
171     }
172     void dual(){
173         Dual :: n = polys;
174         Dual :: m = 0;
175         for(int i = 1;i <= m;++ i){
176             int u = I[2 * i - 1], v = I[2 * i], w = W[2 * i];
177             if(u == outer || v == outer)
178                 w = 1e9L + 1;
179             ++ Dual :: m;
180             Dual :: A[Dual :: m] = u;
181             Dual :: B[Dual :: m] = v;
182             Dual :: W[Dual :: m] = w;
183         }
184         Dual :: build();
185         Dual :: outer = outer;
186     }
187     set <pair<edge, int>, Cmp1> S;
```

```
188     vector <pair<double, int> > T;
189     vector <pair<double, int> > Q;
190     double X[MAXQ], Y[MAXQ];
191     int    Z[MAXQ];
192     int ask(double x, double y){
193         ++ q;
194         point p = rotate(vec(x, y), theta);
195         X[q] = p.x;
196         Y[q] = p.y;
197         return q;
198     }
199     void locate(){
200         T.clear(), Q.clear(), S.clear();
201         for(int i = 1;i <= q;++ i){
202             Q.push_back(make_pair(X[i], i));
203         }
204         for(int i = 1;i <= polys;++ i){
205             for(auto &e : G[i]){
206                 int u = L[e];
207                 int v = R[e];
208                 if(P[u].x > P[v].x){
209                     T.push_back(make_pair(P[v].x + 1e-5,  e));
210                     T.push_back(make_pair(P[u].x - 1e-5, -e));
211                 }
212             }
213         }
214         sort(T.begin(), T.end());
215         sort(Q.begin(), Q.end());
216         int p1 = 0, p2 = 0;
217         scanx = -1e9;
218         Cmp1 CMP;
219         while(p1 < Q.size() || p2 < T.size()){
220             // for(auto it1 = S.begin(), it2 = next(S.begin()); it2 !=
                   S.end();++ it1, ++ it2)
221             //     assert(CMP(*it1, *it2));
222             double x1 = p1 < Q.size() ? Q[p1].first : 1e9;
223             double x2 = p2 < T.size() ? T[p2].first : 1e9;
224             scanx = min(x1, x2);
225             if(equal(scanx, x1)){
226                 auto &x = X[Q[p1].second];
227                 auto &y = Y[Q[p1].second];
228                 auto &z = Z[Q[p1].second];
229                 P[n + 1] = point(-1e9, y);
230                 P[n + 2] = point( 1e9, y);
231                 auto it = S.lower_bound({{n + 1, n + 2}, 0});
232                 if(it == S.end())
233                     z = outer;
234                 else
235                     z = it -> second;
236                 ++ p1;
237             }
238             if(equal(scanx, x2)){
239                 int g = T[p2].second;
240                 if(g > 0){
241                     assert(!S.count({{L[g], R[g]}, I[g]}));
242                     S.insert({{L[g], R[g]}, I[g]});
243                 } else {
```

```
                    g = -g;
                    assert( S.count({{L[g], R[g]}, I[g]}));
                    S.erase ({{L[g], R[g]}, I[g]});
                }
                ++ p2;
            }
        }
    }
}
const int MAXN = 1e5 + 3;
int A[MAXN], B[MAXN];
int main(){
#ifndef ONLINE_JUDGE
    freopen("test.in", "r", stdin);
    freopen("test.out", "w", stdout);
#endif
    int n, m, q;
    Planer :: rotate();
    cin >> n >> m;
    for(int i = 1;i <= n;++ i){
        double x, y;
        cin >> x >> y;
        Planer :: add(x, y);
    }
    for(int i = 1;i <= m;++ i){
        int u, v, w;
        cin >> u >> v >> w;
        Planer :: link(u, v, w);
    }
    Planer :: leftmost();
    Planer :: dual();
    cin >> q;
    for(int i = 1;i <= q;++ i){
        double a1, b1, a2, b2;
        cin >> a1 >> b1;
        A[i] = Planer :: ask(a1, b1);
        cin >> a2 >> b2;
        B[i] = Planer :: ask(a2, b2);
    }
    Planer :: locate();
    for(int i = 1;i <= q;++ i)
        A[i] = Planer :: Z[A[i]],
        B[i] = Planer :: Z[B[i]];
    for(int i = 1;i <= q;++ i){
        int ans = Dual :: solve(A[i], B[i]);
        cout << ans << endl;
    }
    return 0;
}
```

9.4 二维基础

```
#include <bits/stdc++.h>
using namespace std;
using i64 = long long;
const int INF  =  1e9;
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 + a.x * s);
49 }
50 bool cmpx(point a, point b){
51     return sign(a.x - b.x) == -1;
52 }
53 bool cmpy(point a, point b){
54     return sign(a.y - b.y) == -1;
55 }
56 struct line{    // 有向直线
57     point o;
58     vec p;
59     line(point _o, vec _p) : o(_o), p(_p){}
60 };
61 struct segm{    // 有向线段
62     point a, b;
```

```
63      segm(point _a, point _b) : a(_a), b(_b){}
64  };
65  int side(line l, point p){
66      return sign(mulx(l.p, p - l.o));
67  }
68  int side(segm s, point p){
69      return sign(mulx(s.b - s.a, p - s.a));
70  }
71  bool parallel(line a, line b){
72      return equal(0, mulx(a.p, b.p));
73  }
74  double abs(vec a){
75      return sqrt(a.x * a.x + a.y * a.y);
76  }
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) / mulx(a.p, b.p), b.p);
88  }
89  bool intersect(double l1, double r1, double l2, double r2){
90      if(l1 > r1) swap(l1, r1);
91      if(l2 > r2) swap(l2, r2);
92      if(equal(r1, l2) || equal(r2, l1))
93          return true;
94      return !equal(max(r1, r2) - min(l1, l2), r1 - l1 + r2 - l2);
95  }
96  bool intersect(segm s1, segm s2){
97      bool fx = intersect(s1.a.x, s1.b.x, s2.a.x, s2.b.x);
98      if(!fx) return false;
99      bool fy = intersect(s1.a.y, s1.b.y, s2.a.y, s2.b.y);
100     if(!fy) return false;
101     bool g1 = side(s1, s2.a) * side(s1, s2.b) == 1;
102     if(g1) return false;
103     bool g2 = side(s2, s1.a) * side(s2, s1.b) == 1;
104     if(g2) return false;
105     return true;
106 }
107 struct circ{  // 二维圆形
108     point o;
109     double r;
110 };
111 struct poly{  // 二维多边形
112     vector <point> P;
113 };
114 double area(point a, point b, point c){
115     return abs(mulx(b - a, c - a)) / 2;
116 }
117 double area(const poly &P){
118     double ans = 0;
119     for(int i = 0;i < P.P.size();++ i){
```

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

10 其他

10.1 笛卡尔树

```
#include <bits/stdc++.h>
using namespace std;
using i64 = long long;
const int INF  =  1e9;
const i64 INFL = 1e18;
int qread();
const int MAXN = 1e7 + 3;
int n, L[MAXN], R[MAXN], A[MAXN];
void build(){
    stack <int> S;
    A[n + 1] = -1e9;
    for(int i = 1;i ⩽ n + 1;++ i){
        int v = 0;
        while(!S.empty() && A[S.top()] > A[i]){
            auto u = S.top();
            R[u] = v;
            v    = u;
            S.pop();
        }
        L[i] = v;
        S.push(i);
    }
}
int main(){
    n = qread();
    for(int i = 1;i ⩽ n;++ i)
        A[i] = qread();
    build();
    long long ans1 = 0, ans2 = 0;
    for(int i = 1;i ⩽ n;++ i){
        // cout << L[i] << " " << R[i] << endl;
        ans1 ^= 1ll * i * (L[i] + 1);
        ans2 ^= 1ll * i * (R[i] + 1);
    }
    cout << ans1 << " " << ans2 << endl;
    return 0;
}
```

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 $f(i) = \sum_j [a_j \le a_i \wedge b_j \le b_i \wedge c_j \le c_i]$。

```cpp
#include<bits/stdc++.h>
#define up(l, r, i) for(int i = l, END##i = r;i <= END##i;++ i)
#define dn(r, l, i) for(int i = r, END##i = l;i >= END##i;-- i)
using namespace std;
typedef long long i64;
const int INF = 2147483647;
const int MAXN= 1e5 + 3;
const int MAXM= 2e5 + 3;
struct Node{
    int id, a, b, c;
}A[MAXN], B[MAXN];
bool cmp(Node a, Node b){
    if(a.a != b.a) return a.a < b.a;
    if(a.b != b.b) return a.b < b.b;
    if(a.c != b.c) return a.c < b.c;
    return a.id < b.id;
}
int K[MAXN], H[MAXN];
int qread();
int n, m, D[MAXM];
namespace BIT{
    void increase(int x, int w){
        while(x <= m) D[x] += w, x += x & -x;
    }
    void decrease(int x, int w){
        while(x <= m) D[x] -= w, x += x & -x;
    }
    void query(int x, int &r){
        while(x) r += D[x], x -= x & -x;
    }
}
void cdq(int l, int r){
    if(l != r){
        int t = l + r >> 1; cdq(l, t), cdq(t + 1, r);
        int p = l, q = t + 1, u = l;
        while(p <= t && q <= r){
            if(A[p].b <= A[q].b)
                BIT :: increase(A[p].c, 1),       B[u ++] = A[p ++];
             else
                BIT :: query(A[q].c, K[A[q].id]), B[u ++] = A[q ++];
        }
        while(p <= t) BIT :: increase(A[p].c, 1),       B[u ++] = A[p
           ++];
        while(q <= r) BIT :: query(A[q].c, K[A[q].id]), B[u ++] = A[q
           ++];
        up(l, t, i) BIT :: decrease(A[i].c, 1);
        up(l, r, i) A[i] = B[i];
    }
}
int main(){
    n = qread(), m = qread();
    up(1, n, i) A[i].id = i, A[i].a = qread(), A[i].b = qread(), A[i].c
        = qread();
    sort(A + 1, A + 1 + n, cmp), cdq(1, n);
    sort(A + 1, A + 1 + n, cmp);
    dn(n, 1, i){
        if(A[i].a == A[i + 1].a && A[i].b == A[i + 1].b && A[i].c == A[
```

```
            i + 1].c)
55              K[A[i].id] = K[A[i + 1].id];
56          H[K[A[i].id]] ++;
57      }
58      up(0, n - 1, i) printf("%d\n", H[i]);
59      return 0;
60  }
```

10.3 自适应辛普森

10.3.1 例题

计算

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

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

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。

```cpp
#include<bits/stdc++.h>
using namespace std;
const double T0 = 2e3, Tk = 1e-14, delta = 0.993, R = 1e-3;
mt19937 MT(114514);
double distance(double x, double y, double a, double b){
    return sqrt(pow(a - x, 2) + pow(b - y, 2));
}
const int MAXN = 1e3 + 3;
double X[MAXN], Y[MAXN], W[MAXN]; int n;
double calculate(double x, double y){
    double gx, gy, a;
    for(int i = 0;i < n; ++i){
        a = atan2(y - Y[i], x - X[i]);
        gx += cos(a) * W[i];
        gy += sin(a) * W[i];
    }
    return pow(gx, 2) + pow(gy, 2);
}
double ex, ey, eans = 1e18;
void SA(){
    double T = T0, x = 0, y = 0, ans = calculate(x, y);
    double ansx, ansy;
    uniform_real_distribution<double> U;
    while(T > Tk){
        double nx, ny, nans;
        nx = x + 2 * (U(MT) - .5) * T;
        ny = y + 2 * (U(MT) - .5) * T;
        if((nans = calculate(nx, ny)) < ans){
            ans = nans;
            ansx = x = nx;
            ansy = y = ny;
        } else if(exp(-distance(nx, ny, x, y) / T / R) > U(MT)){
            x = nx, y = ny;
        }
        T *= delta;
    }
    if(ans < eans) eans = ans, ex = ansx, ey = ansy;
}
int main(){
    cin >> n;
    for(int i = 0;i < n;++ i)
        cin >> X[i] >> Y[i] >> W[i];
    cout << fixed << setprecision(3);
    if(n == 1){
        cout << X[0] << " " << Y[0] << endl;
    } else {
        SA(), SA(), SA();
        cout << ex << " " << ey << endl;
    }
    return 0;
}
```

10.5 伪随机生成

```cpp
#include<bits/stdc++.h>
using namespace std;
using u32 = uint32_t;
using u64 = uint64_t;
u32 xorshift32(u32 &x){
    x ^= x << 13;
    x ^= x >> 17;
    x ^= x << 5;
    return x;
}
u64 xorshift64(u64 &x){
    x ^= x << 13;
    x ^= x >> 7;
    x ^= x << 17;
    return x;
}
```