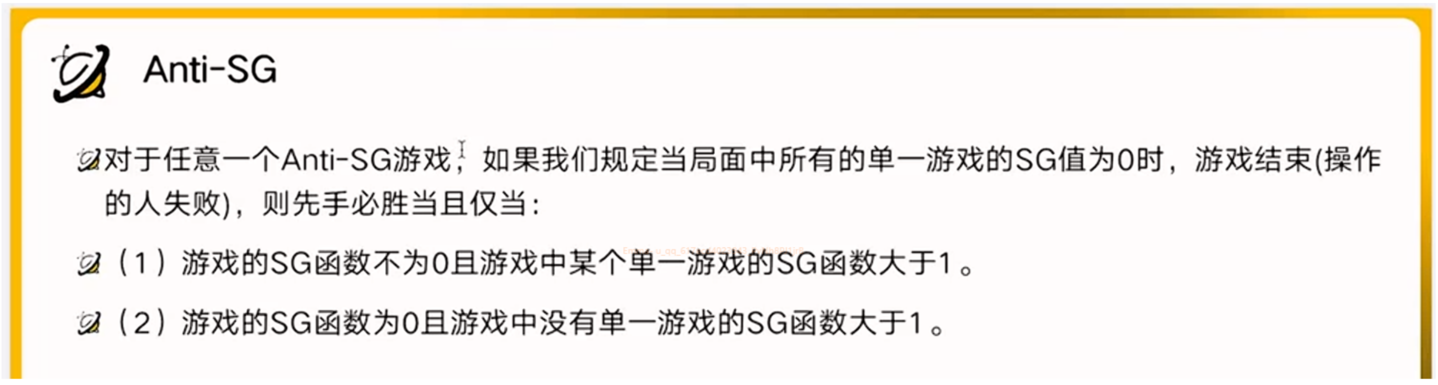
# 比赛板子

## aet重载运算符

```cpp
#include <bits/stdc++.h>

#define PII pair<int, int>
using namespace std;

const int N = 2e5 + 10;
int a[N], b[N], p[N], c[N], ans[N];
int n, m;
struct cmp
{
    bool operator()(int a, int b) const
    {
        return p[a] < p[b];
    }
};
set<int, cmp> s[4];
bool st[N];

int main()
{
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cin >> n;
    for (int i = 0; i < n; i++)
    {
        cin >> p[i];
    }
    for (int i = 0; i < n; i++)
    {
        cin >> a[i];
        s[a[i]].insert(i);
    }
    for (int i = 0; i < n; i++)
    {
        cin >> b[i];
        s[b[i]].insert(i);
    }

    cin >> m;
    for (int i = 0; i < m; i++)
        cin >> c[i];
    for (int i = 0; i < m; i++)
    {
        int x = c[i];
        bool flag = false;
        while (1)
        {
            if (s[x].empty())
                break;
            int res = *s[x].begin();
            s[x].erase(res);
            if (st[res])
                continue;
            else
            {
                st[res] = true;
                cout << p[res] << ' ';
                flag = true;
                break;
            }
        }
        if (!flag)
            cout << "-1 ";
    }
    return 0;
}

```

## anti-SG



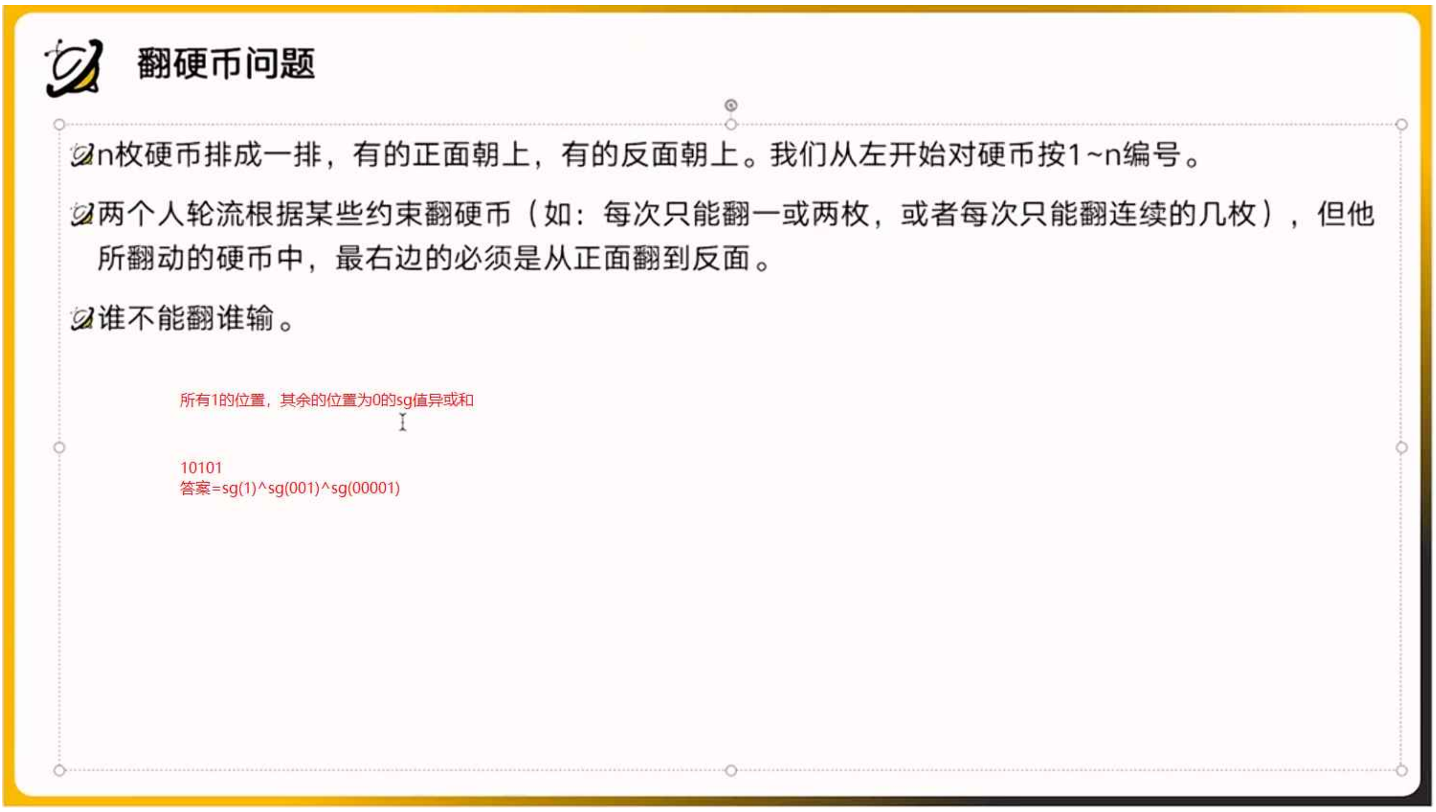
Anti-SG

对于任意一个Anti-SG游戏，如果我们规定当局面中所有的单一游戏的SG值为0时，游戏结束(操作的人失败)，则先手必胜当且仅当：

（1）游戏的SG函数不为0且游戏中某个单一游戏的SG函数大于1。

（2）游戏的SG函数为0且游戏中没有单一游戏的SG函数大于1。

## sg翻硬币模型



**翻硬币问题**

- n枚硬币排成一排，有的正面朝上，有的反面朝上。我们从左开始对硬币按1~n编号。
- 两个人轮流根据某些约束翻硬币（如：每次只能翻一或两枚，或者每次只能翻连续的几枚），但他所翻动的硬币中，最右边的必须是从正面翻到反面。
- 谁不能翻谁输。

所有1的位置，其余的位置为0的sg值异或和

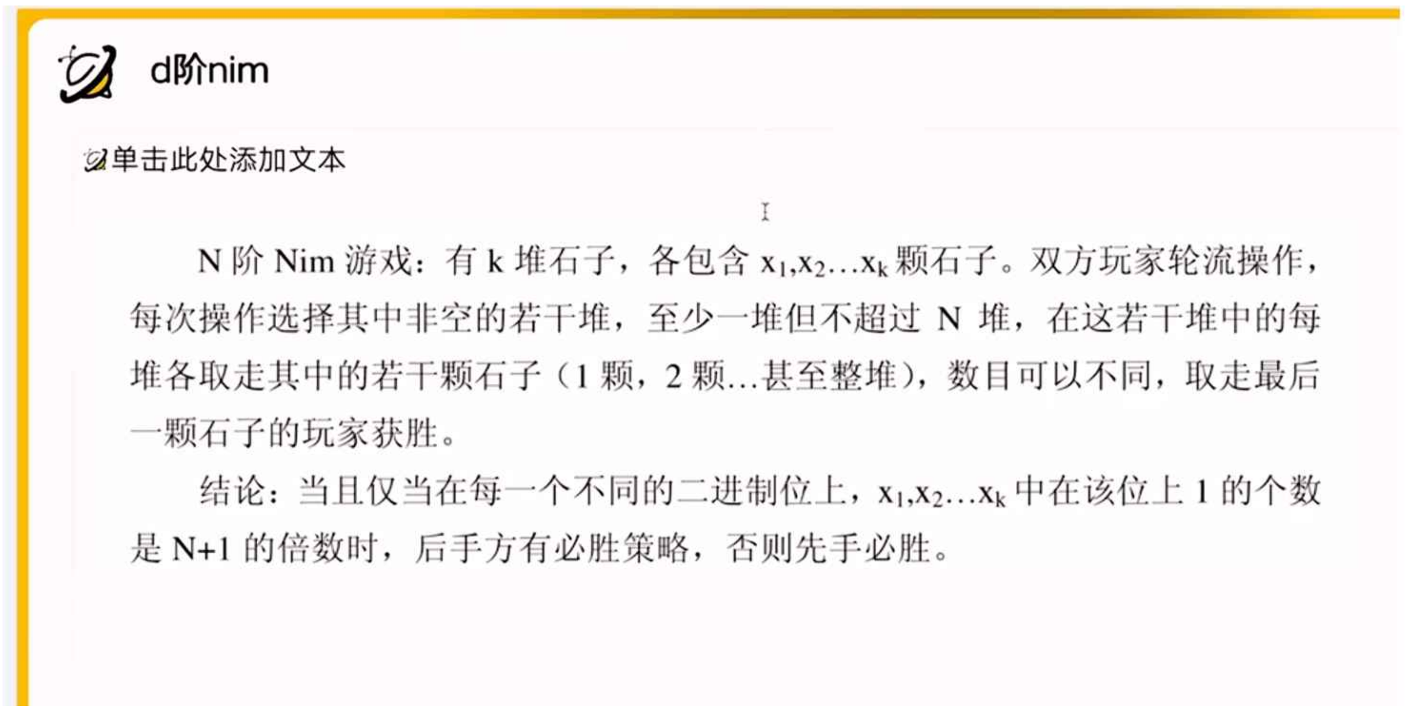
10101
答案=sg(1)^sg(001)^sg(00001)

## dinic(最大流)

```cpp
#include <bits/stdc++.h>
using namespace std;
int read()
{
    int num = 0;
    bool flag = 1;
    char c = getchar();
    for (; c < '0' || c > '9'; c = getchar())
        if (c == '-')
            flag = 0;
    for (; c >= '0' && c <= '9'; c = getchar())
        num = (num << 1) + (num << 3) + c - '0';
    return flag ? num : -num;
}
const int N = 110;
const int M = 5010;
template <typename T>
struct FlowGraph
{
    int s, t, vtot;
    int head[N], etot;
    int dis[N], cur[N];
    struct edge
    {
        int v, nxt;
        T f;
    } e[M << 1];
    void addedge(int u, int v, T f, T f2 = 0)
    {
        e[etot] = {v, head[u], f};
        head[u] = etot++;
        e[etot] = {u, head[v], f2};
        head[v] = etot++;
    }
    bool bfs()
    {
        for (int i = 1; i <= vtot; i++)
        {
            dis[i] = 0;
            cur[i] = head[i];
        }
        queue<int> q;
        q.push(s);
        dis[s] = 1;
        while (!q.empty())
        {
            int u = q.front();
            q.pop();
            for (int i = head[u]; ~i; i = e[i].nxt)
            {
                if (e[i].f && !dis[e[i].v])
                {
                    int v = e[i].v;
                    dis[v] = dis[u] + 1;
                    if (v == t)
                        return 1;
                    q.push(v);
                }
            }
        }
        return 0;
    }
    T dfs(int u, T m)
    {
        if (u == t)
            return m;
        T flow = 0;
        for (int i = cur[u]; ~i; cur[u] = i = e[i].nxt)
            if (e[i].f && dis[e[i].v] == dis[u] + 1)
            {
                T f = dfs(e[i].v, min(m, e[i].f));
                e[i].f -= f;
```

```cpp
                e[i ^ 1].f += f;
                m -= f;
                flow += f;
                if (!m)
                    break;
            }
        if (!flow)
            dis[u] = -1;
        return flow;
    }
    T dinic()
    {
        T flow = 0;
        while (bfs())
            flow += dfs(s, numeric_limits<T>::max());
        return flow;
    }
    void init(int s_, int t_, int vtot_)
    {
        s = s_;
        t = t_;
        vtot = vtot_;
        etot = 0;
        for (int i = 1; i <= vtot; i++)
            head[i] = -1;
    }
};
FlowGraph<long long> g;
int n, m, s, t;
int main()
{
    n = read();
    m = read();
    s = read();
    t = read();
    g.init(s, t, n);
    for (int i = 1; i <= m; i++)
    {
        int x = read(), y = read(), z = read();
        g.addedge(x, y, z);
    }
    printf("%lld\n", g.dinic());
    return 0;
}

```

## d阶nim



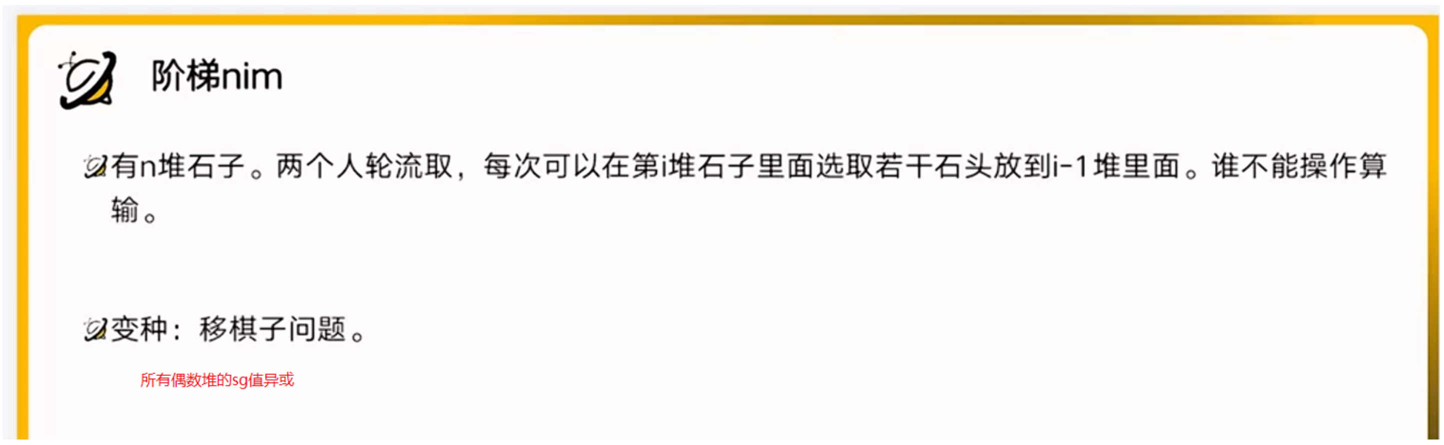
d阶nim

单击此处添加文本

N 阶 Nim 游戏：有 k 堆石子，各包含 $x_1,x_2 \ldots x_k$ 颗石子。双方玩家轮流操作，每次操作选择其中非空的若干堆，至少一堆但不超过 N 堆，在这若干堆中的每堆各取走其中的若干颗石子（1 颗，2 颗…甚至整堆），数目可以不同，取走最后一颗石子的玩家获胜。

结论：当且仅当在每一个不同的二进制位上，$x_1,x_2 \ldots x_k$ 中在该位上 1 的个数是 N+1 的倍数时，后手方有必胜策略，否则先手必胜。

## 阶梯nim



阶梯nim

有n堆石子。两个人轮流取，每次可以在第i堆石子里面选取若干石头放到i-1堆里面。谁不能操作算输。

变种：移棋子问题。

所有偶数堆的sg值异或

## ex_gcd

```cpp
#include <bits/stdc++.h>
using namespace std;
int ex_gcd(int a, int &x, int b, int &y, int c)
{
    if (b == 0)
    {
        x = c / a;
        y = 0;
        return a;
    }
    int d = ex_gcd(b, y, a % b, x, c);
    y = (y - (a / b) * x) % Mod;
    return d;
}
signed main()
{
    return 0;
}

```

## KM算法(完美最大权匹配n^3)

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
const int N = 1010;
const int inf = 1e18;
struct _KM
{
    int n, Map[N][N], matched[N];
    int slack[N], pre[N], ex[N], ey[N];
    bool visx[N], visy[N];
    void match(int u)
    {
        int x, y = 0, yy = 0, delta;
        memset(pre, 0, sizeof(pre));
        for (int i = 1; i <= n; i++)
            slack[i] = inf;
        matched[y] = u;
        while (1)
        {
            x = matched[y];
            delta = inf;
            visy[y] = 1;
            for (int i = 1; i <= n; i++)
            {
                if (visy[i])
                    continue;
                if (slack[i] > ex[x] + ey[i] - Map[x][i])
                {
                    slack[i] = ex[x] + ey[i] - Map[x][i];
                    pre[i] = y;
                }
                if (slack[i] < delta)
                {
                    delta = slack[i];
                    yy = i;
                }
            }
            for (int i = 0; i <= n; i++)
            {
                if (visy[i])
                    ex[matched[i]] -= delta, ey[i] += delta;
                else
                    slack[i] -= delta;
            }
            y = yy;
            if (matched[y] == -1)
                break;
        }
        while (y)
        {
            matched[y] = matched[pre[y]];
            y = pre[y];
        }
    }
    int KM()
    {
        memset(matched, -1, sizeof(matched));
        memset(ex, 0, sizeof(ex));
        memset(ey, 0, sizeof(ey));
        for (int i = 1; i <= n; i++)
        {
            memset(visy, 0, sizeof(visy));
            match(i);
        }
        int res = 0;
        for (int i = 1; i <= n; i++)
            if (matched[i] != -1)
                res += Map[i][matched[i]];
        return res / 2;
    }
    void init(int _n)
    {
        n = _n;
        for (int i = 1; i <= n; i++)
            for (int j = 1; j <= n; j++)
                Map[i][j] = -inf;
    }
} g;
int n, a[N], p[N], b[N], c[N];
signed main()
{
    scanf("%lld", &n);
    for (int i = 1; i <= n; i++)
        scanf("%lld", &a[i]);
    for (int i = 1; i <= n; i++)
        scanf("%lld", &p[i]);
    for (int i = 1; i <= n; i++)
        scanf("%lld", &b[i]);
    for (int i = 1; i <= n; i++)
        scanf("%lld", &c[i]);
    g.init(n * 2);
    for (int i = 1; i <= n; i++)
    {
        for (int j = 1; j <= n; j++)
        {
            int now = b[i] + c[j], w = 0;
            for (int k = 1; k <= n; k++)
                if (a[k] < now)
                    w += p[k];
            g.Map[i][j + n] = g.Map[j + n][i] = w;
        }
    }
    printf("%lld", g.KM());
    return 0;
}

```

## Manacher

```
#include <bits/stdc++.h>
using namespace std;
const int N = 11e6 + 10;
char s[N], b[N << 1];
int p[N << 1];
int Manacher()
{
    int n = strlen(s + 1);
    for (int i = 1; i <= n; i++)
        b[i * 2] = s[i], b[i * 2 - 1] = '$';
    int len = n << 1;
    b[0] = '%';
    b[++len] = '$';
    int pos, R = 0, Max = 0;
    for (int i = 1; i <= len; i++)
    {
        if (i < R)
            p[i] = min(R - i + 1, p[2 * pos - i]);
```

```cpp
        else
            p[i] = 1;
        while (b[i + p[i]] == b[i - p[i]])
            p[i]++;
        if (i + p[i] - 1 > R)
        {
            R = i + p[i] - 1;
            pos = i;
        }
        Max = max(Max, p[i] - 1);
    }
    return Max;
}
signed main()
{
    scanf("%s", s + 1);
    printf("%d", Manacher());
    return 0;
}

```

## 差分约束

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 5e3 + 10;
int read()
{
    int num = 0;
    bool flag = 1;
    char c = getchar();
    for (; c < '0' || c > '9'; c = getchar())
        if (c == '-')
            flag = 0;
    for (; c >= '0' && c <= '9'; c = getchar())
        num = (num << 1) + (num << 3) + c - '0';
    return flag ? num : -num;
}
int n, m, d[N];
vector<array<int, 3>> e;
signed main()
{
    n = read();
    m = read();
    for (int i = 1; i <= n; i++)
    {
        int x = read(), y = read(), z = read();
        //      e.push_back({y,x,z});
        // d[x]-d[y]<=z -> 答案的最大值
        e.push_back({x, y, z});
        // 设di'=-di(答案是di),求di'的最大值，就知道了di的最小值
        //(-dx')-(-dy')<=z -> dy'-dx'<=z 边反向，最后求答案
    }
    for (int i = 1; i <= n; i++)
        e.push_back({0, i, 0});
    // d[0]-d[i]<=0 -> d[0]<=d[i]，求的最小值，所以每个点其实是d0' di',
    //(-d0')<=(-di') di'-d0'<=0
    for (int i = 0; i <= n; i++)
        for (auto k : e)
            d[k[1]] = min(d[k[1]], d[k[0]] + k[2]);
    for (auto k : e)
        if (d[k[1]] > d[k[0]] + k[2])
        {
            printf("-1");
            return 0;
        }
    for (int i = 1; i <= n; i++)
        printf("%d ", -d[i] + d[0]);
    // 求最小值，每个取负就是这个答案
    return 0;
}

```

## 笛卡尔树

```
// 一个区间取出最小值为根
// 分成左右区间在左右儿子
// 两边分别重复这些操作
#include <bits/stdc++.h>
using namespace std;
// 性质1：区间最小值为两个端点的lca
// 性质2：中序遍历就是原数组
/*性质3：一个点的祖先节点，如果这个祖先节点到这个点的路径是第一个向 左/右 的路径
，那么这个祖先节点就是 左/右 边第一个小于等于它的数*/
const int N = 1e5 + 10;
int n, a[N], l[N], r[N];
void build()
{
  stack<int> st;
  int root = 0;
  for (int i = 1; i <= n; i++)
  {
    int last = 0;
    while (!st.empty() && a[st.top()] > a[i])
    {
      last = st.top();
      st.pop();
    }
    if (!st.empty())
      r[st.top()] = i;
    else
      root = i;
    l[i] = last;
    st.push(i);
  }
}
signed main()
{
  scanf("%d", &n);
  for (int i = 1; i <= n; i++)
    scanf("%d", &a[i]);
  return 0;
}

```

## 点, 线段, 极角排序

```
#include <bits/stdc++.h>
using namespace std;
#define ubit(a) (64 - __builtin_clzll(a))
// 获得最高的1的位置
#define popcount(a) __builtin_popcountll(a)
// 第一个1的个数
typedef double db;
const db EPS = 1e-9;
// 点(x,y)之间的操作 }

// TOP1
// 点
inline int sign(db a) { return a < -EPS ? -1 : a > EPS; }
// 1)a<0 -> -1  2)a>0 ->1  3)a=0->0
inline int cmp(db a, db b) { return sign(a - b); }
struct P
{
    db x, y;
    P() {}
    P(db _x, db _y) : x(_x), y(_y) {}
    P operator+(P p) { return {x + p.x, y + p.y}; }
    P operator-(P p) { return {x - p.x, y - p.y}; }
    P operator*(db d) { return {x * d, y * d}; }
    P operator/(db d) { return {x / d, y / d}; }
    // 点的加减乘除基本运算
    bool operator<(P p) const
    { // 两者谁更小
        int c = cmp(x, p.x);
        if (c)
            return c == -1;
        return cmp(y, p.y) == -1;
    }
    bool operator==(P o) const
    { // 是否相等
        return cmp(x, o.x) == 0 && cmp(y, o.y) == 0;
    }
    db dot(P p) { return x * p.x + y * p.y; }    // 点积
    db det(P p) { return x * p.y - y * p.x; }    // 叉积 =|a||b|sin(o) o为a到b逆时针的那个夹角
    db distTo(P p) { return (*this - p).abs(); } // 求点到点的距离
    db alpha() { return atan2(y, x); }           // 求极角(-pai,pai)(x负半轴为-pai逆时针到pai)
    db abs() { return sqrt(abs2()); }            // 求两点距离
    db abs2() { return x * x + y * y; }          // 两点距离平方
    P rot90() { return P(-y, x); }               // 逆时针旋转90度
    P unit() { return *this / abs(); }
    int quad() const { return sign(y) == 1 || (sign(y) == 0 && sign(x) >= 0); }
    P rot(db an) { return {x * cos(an) - y * sin(an), x * sin(an) + y * cos(an)}; } // 逆时针转an的角度(an弧度制)
};

// Top2
// 线，随便两个点表示一段线 ，要用点的包装
#define cross(p1, p2, p3) ((p2.x - p1.x) * (p3.y - p1.y) - (p3.x - p1.x) * (p2.y - p1.y)) //(p1p2 叉乘 p1p3)
#define crossOp(p1, p2, p3) sign(cross(p1, p2, p3))           // 0是三点共线 1是p1->p2->p3 是逆时针 ，-1则是顺时针

// 直线 p1p2, q1q2 是否恰有一个交点  1表示有交点 0表示无交点
bool chkLL(P p1, P p2, P q1, P q2)
{
    db a1 = cross(q1, q2, p1), a2 = -cross(q1, q2, p2);
    return sign(a1 + a2) != 0;
}

```

```cpp
// 求直线 p1p2, q1q2 的交点  （要先用上面的判断是否有交点）
P isLL(P p1, P p2, P q1, P q2)
{
    db a1 = cross(q1, q2, p1), a2 = -cross(q1, q2, p2);
    return (p1 * a2 + p2 * a1) / (a1 + a2);
}

// 判断区间 [l1, r1], [l2, r2] 是否相交
bool intersect(db l1, db r1, db l2, db r2)
{
    if (l1 > r1)
        swap(l1, r1);
    if (l2 > r2)
        swap(l2, r2);
    return !(cmp(r1, l2) == -1 || cmp(r2, l1) == -1);
}

// 线段 p1p2, q1q2 是否相交
bool isSS(P p1, P p2, P q1, P q2)
{
    return intersect(p1.x, p2.x, q1.x, q2.x) && intersect(p1.y, p2.y, q1.y, q2.y) &&
           crossOp(p1, p2, q1) * crossOp(p1, p2, q2) <= 0 && crossOp(q1, q2, p1) * crossOp(q1, q2, p2) <= 0;
}

// 线段 p1p2, q1q2 严格相交  （不交在端点）
bool isSS_strict(P p1, P p2, P q1, P q2)
{
    return crossOp(p1, p2, q1) * crossOp(p1, p2, q2) < 0 && crossOp(q1, q2, p1) * crossOp(q1, q2, p2) < 0;
}

// m 是否在 a 和 b 线段之间
bool isMiddle(db a, db m, db b)
{
    return sign(a - m) == 0 || sign(b - m) == 0 || (a < m != b < m);
}
// 点m是否在这个ab这个矩形内
bool isMiddle(P a, P m, P b)
{
    return isMiddle(a.x, m.x, b.x) && isMiddle(a.y, m.y, b.y);
}

// 点 p 是否在线段 p1p2 上
bool onSeg(P p1, P p2, P q)
{ // 可能有精度问题
```

```cpp
    return crossOp(p1, p2, q) == 0 && isMiddle(p1, q, p2);
}
// q1q2 和 p1p2 的交点 在 p1p2 上? 确定的时候不需要crossOp(p1,p2,q) == 0

// 点 p 严格在 p1p2 上
bool onSeg_strict(P p1, P p2, P q)
{
    return crossOp(p1, p2, q) == 0 && sign((q - p1).dot(p1 - p2)) * sign((q -
p2).dot(p1 - p2)) < 0;
}

// 求 q 到 直线p1p2 的投影（垂足） p1 != p2
P proj(P p1, P p2, P q)
{
    P dir = p2 - p1;
    return p1 + dir * (dir.dot(q - p1) / dir.abs2());
}

// 求 q 以 直线p1p2 为轴的反射  p1 != p2
P reflect(P p1, P p2, P q)
{
    return proj(p1, p2, q) * 2 - q;
}

// 求 q 到 线段p1p2 的最小距离
db nearest(P p1, P p2, P q)
{
    if (p1 == p2)
        return p1.distTo(q);
    P h = proj(p1, p2, q);
    if (isMiddle(p1, h, p2))
        return q.distTo(h);
    return min(p1.distTo(q), p2.distTo(q));
}

// 求 线段p1p2 与 线段q1q2 的距离
db disSS(P p1, P p2, P q1, P q2)
{
    if (isSS(p1, p2, q1, q2))
        return 0;
    return min(min(nearest(p1, p2, q1), nearest(p1, p2, q2)), min(nearest(q1,
q2, p1), nearest(q1, q2, p2)));
}

// TOP3
// 极角排序（x负半轴到逆时针排序）
const int N = 1e5 + 10;
```

```
P p[N];
int n;
void Sort()
{
    sort(p + 1, p + n + 1, [&](P &a, P &b)
         {
        int qa=a.quad(),qb=b.quad();
        if(qa!=qb)return qa<qb;
          else return sign(a.det(b))>0; });
}
signed main()
{
    return 0;
}

```

## 动态开点线段树

```
#include <bits/stdc++.h>
using namespace std;
int read()
{
    int num = 0;
    bool flag = 1;
    char c = getchar();
    for (; c < '0' || c > '9'; c = getchar())
        if (c == '-')
            flag = 0;
    for (; c >= '0' && c <= '9'; c = getchar())
        num = (num << 1) + (num << 3) + c - '0';
    return flag ? num : -num;
}
const int N = 1e5 + 10;
int n, q;
#define ll long long
struct Tree
{
    struct cow
    {
        int ls, rs;
        ll sum, lazy;
    } tree[N << 2];
    int tot = 1;
#define ls(p) tree[p].ls
#define rs(p) tree[p].rs
```

```cpp
#define sum(p) tree[p].sum
#define lazy(p) tree[p].lazy
    void chck_new(int p)
    {
        if (!ls(p))
            ls(p) = ++tot;
        if (!rs(p))
            rs(p) = ++tot;
    }
    void lazytime(int p, int l, int r)
    {
        int mid = l + r - 1 >> 1;
        sum(ls(p)) += lazy(p) * (mid - l + 1);
        sum(rs(p)) += lazy(p) * (r - mid);
        lazy(ls(p)) += lazy(p);
        lazy(rs(p)) += lazy(p);
        lazy(p) = 0;
    }
    void add(int p, int l, int r, int L, int R, ll d)
    {
        if (l >= L && r <= R)
        {
            sum(p) += d * (r - l + 1);
            lazy(p) += d;
            return;
        }
        chck_new(p);
        lazytime(p, l, r);
        int mid = l + r - 1 >> 1;
        if (L <= mid)
            add(ls(p), l, mid, L, R, d);
        if (R > mid)
            add(rs(p), mid + 1, r, L, R, d);
        sum(p) = sum(ls(p)) + sum(rs(p));
    }
    ll ask(int p, int l, int r, int L, int R)
    {
        if (l >= L && r <= R)
            return sum(p);
        chck_new(p);
        lazytime(p, l, r);
        int mid = l + r - 1 >> 1;
        ll ans = 0;
        if (L <= mid)
            ans += ask(ls(p), l, mid, L, R);
        if (R > mid)
            ans += ask(rs(p), mid + 1, r, L, R);
        return ans;
    }
} T;
int main()
{
    n = read();
    q = read();
    for (int i = 1; i <= n; i++)
        T.add(1, 1, n, i, i, read());
    for (int i = 1; i <= q; i++)
    {
        int op = read();
        if (op == 1)
        {
            int x = read(), y = read(), k = read();
            T.add(1, 1, n, x, y, k);
        }
        else
        {
            int x = read(), y = read();
            printf("%lld\n", T.ask(1, 1, n, x, y));
        }
    }
    return 0;
}
```

## 对拍

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int T = 0;
    while (1)
    {
        T++;
        system("C:\\randsequence.exe"); // rand
        double stb = clock();
        system("C:\\1.exe"); // 暴力
        double edb = clock();
        double stz = clock();
        system("C:\\2.exe"); // 正解
        double edz = clock();
        if (system("fc C:\\1.out  C:\\2.out"))
        {
            cout << "WA";
            return 0;
        }
        else
        {
            printf("测试点: %d  \n 暴力: %.0lfms\n  正解: %.0lfms\n", T, edb - stb, edz - stz);
        }
    }
}

```

## 多边形

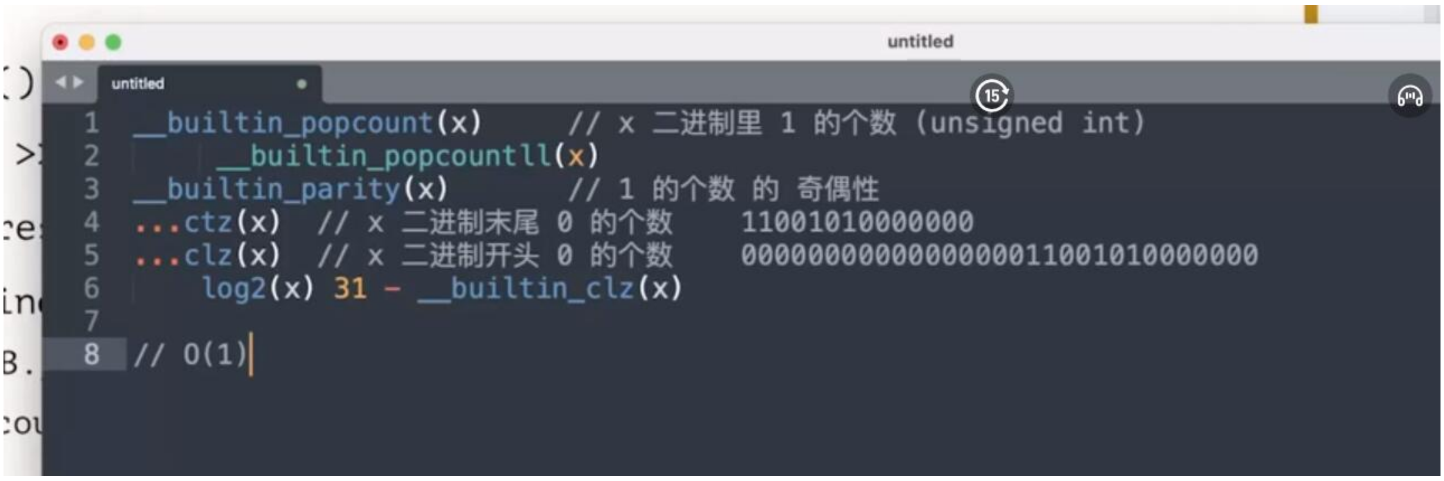
```
#include <bits/stdc++.h>
using namespace std;
typedef double db;
const db EPS = 1e-9;

inline int sign(db a) { return a < -EPS ? -1 : a > EPS; }

inline int cmp(db a, db b) { return sign(a - b); }

struct P
{
    db x, y;
    P() {}
    P(db _x, db _y) : x(_x), y(_y) {} // 点的加减乘除基本运算
    P operator+(P p) { return {x + p.x, y + p.y}; }
    P operator-(P p) { return {x - p.x, y - p.y}; }
    P operator*(db d) { return {x * d, y * d}; }
    P operator/(db d) { return {x / d, y / d}; }

    bool operator<(P p) const
    { // 先比较x大小，再比较y 返回小的
        int c = cmp(x, p.x);
        if (c)
            return c == -1;
        return cmp(y, p.y) == -1;
    }

    bool operator==(P o) const
    {
        return cmp(x, o.x) == 0 && cmp(y, o.y) == 0;
    }
```

```cpp

    db dot(P p) { return x * p.x + y * p.y; }
    db det(P p) { return x * p.y - y * p.x; } // 叉积 正为逆时针方向

    db distTo(P p) { return (*this - p).abs(); } // 求点到点的距离
    db alpha() { return atan2(y, x); }           // 求极角
    void read() { cin >> x >> y; }
    void write() { cout << "(" << x << "," << y << ")" << endl; }
    db abs() { return sqrt(abs2()); } // 求两点距离
    db abs2() { return x * x + y * y; }
    P rot90() { return P(-y, x); }     // 旋转 90度
    P unit() { return *this / abs(); } //
    int quad() const { return sign(y) == 1 || (sign(y) == 0 && sign(x) >= 0); }
    P rot(db an) { return {x * cos(an) - y * sin(an), x * sin(an) + y *
cos(an)}; }
};
#define cross(p1, p2, p3) ((p2.x - p1.x) * (p3.y - p1.y) - (p3.x - p1.x) *
(p2.y - p1.y))
#define crossOp(p1, p2, p3) sign(cross(p1, p2, p3))
bool isMiddle(db a, db m, db b)
{
    return sign(a - m) == 0 || sign(b - m) == 0 || (a < m != b < m);
}

bool isMiddle(P a, P m, P b)
{
    return isMiddle(a.x, m.x, b.x) && isMiddle(a.y, m.y, b.y);
}
// 点 p 在线段 p1p2 上
bool onSeg(P p1, P p2, P q)
{
    return crossOp(p1, p2, q) == 0 && isMiddle(p1, q, p2);
}
// 求直线 p1p2, q1q2 的交点
P isLL(P p1, P p2, P q1, P q2)
{
    db a1 = cross(q1, q2, p1), a2 = -cross(q1, q2, p2);
    return (p1 * a2 + p2 * a1) / (a1 + a2);
}
//-------------------这里开始是多边形，前面都是点线段的代码------------------

db area(vector<P> ps)
{
    db res = 0;
    for (int i = 0; i < ps.size(); i++)
        res += ps[i].det(ps[(i + 1) % ps.size()]);
    return abs(res) / 2;
```

```cpp
} // 面积，只有点按照逆时针或顺时针才能这么算

int contain(vector<P> ps, P p)
{ // 2:inside,1:on_seg,0:outside
    int n = ps.size(), ret = 0;
    for (int i = 0; i < n; i++)
    {
        P u = ps[i], v = ps[(i + 1) % n];
        if (onSeg(u, v, p))
            return 1;
        if (cmp(u.y, v.y) <= 0)
            swap(u, v);
        if (cmp(p.y, u.y) > 0 || cmp(p.y, v.y) <= 0)
            continue;
        ret ^= crossOp(p, u, v) > 0;
    }
    return ret * 2;
} // 点包含

// 凸包：包含所有点的一个凸多边形，且顶点都是给定点
vector<P> convexHull(vector<P> ps)
{
    int n = ps.size();
    if (n <= 1)
        return ps;
    sort(ps.begin(), ps.end());
    vector<P> qs(n * 2);
    int k = 0;
    for (int i = 0; i < n; qs[k++] = ps[i++])
        while (k > 1 && crossOp(qs[k - 2], qs[k - 1], ps[i]) <= 0)
            --k;
    for (int i = n - 2, t = k; i >= 0; qs[k++] = ps[i--])
        while (k > t && crossOp(qs[k - 2], qs[k - 1], ps[i]) <= 0)
            --k;
    qs.resize(k - 1);
    return qs;
} // 严格的凸包（不含180°角）
vector<P> convexHullNonStrict(vector<P> ps)
{
    int n = ps.size();
    if (n <= 1)
        return ps;
    sort(ps.begin(), ps.end());
    vector<P> qs(n * 2);
    int k = 0;
    for (int i = 0; i < n; qs[k++] = ps[i++])
        while (k > 1 && crossOp(qs[k - 2], qs[k - 1], ps[i]) < 0)
```

```cpp
            --k;
    for (int i = n - 2, t = k; i >= 0; qs[k++] = ps[i--])
        while (k > t && crossOp(qs[k - 2], qs[k - 1], ps[i]) < 0)
            --k;
    qs.resize(k - 1);
    return qs;
} // 不严格的凸包（含180°角）
// 注意这个一定要去重

vector<P> convexCut(const vector<P> &ps, P q1, P q2)
{
    vector<P> qs;
    int n = ps.size();
    for (int i = 0; i < n; i++)
    {
        P p1 = ps[i], p2 = ps[(i + 1) % n];
        int d1 = crossOp(q1, q2, p1), d2 = crossOp(q1, q2, p2);
        if (d1 >= 0)
            qs.push_back(p1);
        if (d1 * d2 < 0)
            qs.push_back(isLL(p1, p2, q1, q2));
    }
    return qs;
} // q1 q2分割凸多边形ps后变成的新凸多边形

db convexDiameter(vector<P> ps)
{
    int n = ps.size();
    if (n <= 1)
        return 0;
    int is = 0, js = 0;
    for (int k = 1; k < n; k++)
        is = ps[k] < ps[is] ? k : is, js = ps[js] < ps[k] ? k : js;
    int i = is, j = js;
    db ret = ps[i].distTo(ps[j]);
    do
    {
        if ((ps[(i + 1) % n] - ps[i]).det(ps[(j + 1) % n] - ps[j]) >= 0)
            (++j) %= n;
        else
            (++i) %= n;
        ret = max(ret, ps[i].distTo(ps[j]));
    } while (i != is || j != js);
    return ret;
} // 这些点中最远的两个点的点距离
signed main()
{
    return 0;
}

```

## 二进制



```
__builtin_popcount(x)      // x 二进制里 1 的个数 (unsigned int)
        __builtin_popcountll(x)
__builtin_parity(x)        // 1 的个数 的 奇偶性
...ctz(x)  // x 二进制末尾 0 的个数    11001010000000
...clz(x)  // x 二进制开头 0 的个数    0000000000000000011001010000000
    log2(x) 31 - __builtin_clz(x)

// O(1)
```

## 割边

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5 + 10;
const int inf = 1e9;
int read()
{
    int num = 0;
    bool flag = 1;
    char c = getchar();
    for (; c < '0' || c > '9'; c = getchar())
        if (c == '-')
            flag = 0;
    for (; c >= '0' && c <= '9'; c = getchar())
        num = (num << 1) + (num << 3) + c - '0';
    return flag ? num : -num;
}
vector<pair<int, int>> e[N];
int n, m, dfn[N], low[N], idx;
vector<int> bridge;
void Tarjan(int x, int id)
{
    dfn[x] = low[x] = ++idx;
    for (auto it : e[x])
    {
        int y = it.first, id2 = it.second;
        if (!dfn[y])
        {
            Tarjan(y, id2);
            low[x] = min(low[x], low[y]);
        }
        else if (id != id2)
            low[x] = min(low[x], dfn[y]);
    }
    if (low[x] == dfn[x] && id)
        bridge.push_back(id);
}
signed main()
{
    n = read();
    m = read();
    for (int i = 1; i <= m; i++)
    {
        int x = read(), y = read();
        e[x].push_back({y, i});
        e[y].push_back({x, i});
    }
    for (int i = 1; i <= n; i++)
        if (!dfn[i])
            Tarjan(i, 0);
    sort(bridge.begin(), bridge.end());
    printf("%d\n", bridge.size());
    for (auto num : bridge)
        printf("%d ", num);
    return 0;
}

```

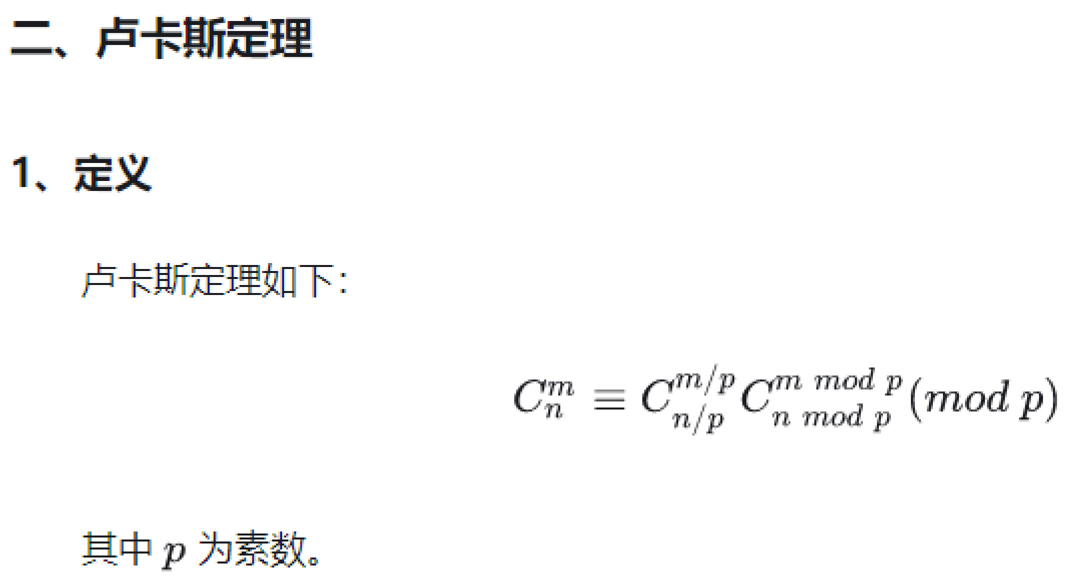
## 割点

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5 + 10;
const int inf = 1e9;
int read()
{
    int num = 0;
    bool flag = 1;
    char c = getchar();
    for (; c < '0' || c > '9'; c = getchar())
        if (c == '-')
```

```cpp
            flag = 0;
    for (; c >= '0' && c <= '9'; c = getchar())
        num = (num << 1) + (num << 3) + c - '0';
    return flag ? num : -num;
}
vector<int> e[N];
int n, m, dfn[N], low[N], idx, cut[N], rt;
void Tarjan(int x, int fa)
{
    dfn[x] = low[x] = ++idx;
    int ct = 0;
    for (auto y : e[x])
    {
        if (!dfn[y])
        {
            Tarjan(y, fa);
            ct++;
            if (low[y] >= dfn[x])
            {
                cut[x] = 1;
            }
            low[x] = min(low[x], low[y]);
        }
        else if (y != fa)
            low[x] = min(low[x], dfn[y]);
    }
    if (x == rt && ct <= 1)
        cut[x] = 0;
}
signed main()
{
    n = read();
    m = read();
    for (int i = 1; i <= m; i++)
    {
        int x = read(), y = read();
        e[x].push_back(y);
        e[y].push_back(x);
    }
    for (int i = 1; i <= n; i++)
        if (!dfn[i])
            rt = i, Tarjan(i, 0);
    int sum = 0;
    for (int i = 1; i <= n; i++)
        sum += cut[i];
    printf("%d\n", sum);
    for (int i = 1; i <= n; i++)
```

```
        if (cut[i])
            printf("%d ", i);
    return 0;
}

```

## 卢卡斯



### 二、卢卡斯定理

#### 1、定义

卢卡斯定理如下：

$$C_n^m \equiv C_{n/p}^{m/p} C_{n \bmod p}^{m \bmod p} (mod\ p)$$

其中 $p$ 为素数。

## 莫队(带时间修改)

```
#include <bits/stdc++.h>
using namespace std;
int read()
{
    int num = 0;
    bool flag = 1;
    char c = getchar();
    for (; c < '0' || c > '9'; c = getchar())
        ;
    for (; c >= '0' && c <= '9'; c = getchar())
        num = (num << 1) + (num << 3) + c - '0';
    return flag ? num : -num;
}
const int N = 1e5 + 10;
map<int, int> lsh;
struct Q
{
    int l, r, id;
} Qu[N], Ch[N];
```

```cpp
int v[N];
bool cmpQu(Q a, Q b)
{
    if (v[a.l] ^ v[b.l])
        return a.l < b.l;
    if (v[a.r] ^ v[b.r])
        if (v[a.l] & 1)
            return a.r < b.r;
        else
            return a.r > b.r;
    return a.id < b.id;
}
int cnt[N << 1], cnt_cnt[N], ans[N];
int n, q, a[N], tot, Qu_tot, Ch_tot;
void add(int x)
{
    cnt_cnt[cnt[a[x]]]--;
    cnt_cnt[++cnt[a[x]]]++;
}
void del(int x)
{
    cnt_cnt[cnt[a[x]]--]--;
    cnt_cnt[cnt[a[x]]]++;
}
void Time(int t, int i)
{
    int l = Qu[i].l, r = Qu[i].r;
    int pos = Ch[t].l, val = Ch[t].r;
    if (pos >= l && pos <= r)
    {
        cnt_cnt[cnt[a[pos]]--]--;
        cnt_cnt[cnt[a[pos]]]++;
        cnt_cnt[cnt[val]]--;
        cnt_cnt[++cnt[val]]++;
    }
    swap(Ch[t].r, a[pos]);
}
int main()
{
    n = read();
    q = read();
    for (int i = 1; i <= n; i++)
    {
        a[i] = read();
        if (!lsh[a[i]])
            lsh[a[i]] = ++tot;
        a[i] = lsh[a[i]];
    }
    for (int i = 1; i <= q; i++)
    {
        int t = read();
        if (t == 1)
        {
            Qu[++Qu_tot].l = read();
            Qu[Qu_tot].r = read();
            Qu[Qu_tot].id = i;
        }
        else
        {
            Ch[++Ch_tot].l = read();
            Ch[Ch_tot].r = read();
            Ch[Ch_tot].id = i;
            if (!lsh[Ch[Ch_tot].r])
                lsh[Ch[Ch_tot].r] = ++tot;
            Ch[Ch_tot].r = lsh[Ch[Ch_tot].r];
        }
    }
    int gh = pow(n, 2.0 / 3.0), l = 1, r = 0, t = 0;
    for (int i = 1; i <= n; i++)
        v[i] = (i - 1) / gh + 1;
    sort(Qu + 1, Qu + Qu_tot + 1, cmpQu);
    Ch[++Ch_tot].id = N;
    //
    for (int i = 1; i <= Qu_tot; i++)
    {
        int _l = Qu[i].l, _r = Qu[i].r,
            _t = Qu[i].id, Ans = 1;
        while (r < _r)
            add(++r);
        while (l > _l)
            add(--l);
        while (l < _l)
            del(l++);
        while (r > _r)
            del(r--);
        while (_t > Ch[t + 1].id)
            Time(++t, i);
        while (_t < Ch[t].id)
            Time(t--, i);
        while (cnt_cnt[Ans])
            Ans++;
        ans[Qu[i].id] = Ans;
    }
    for (int i = 1; i <= q; i++)
```

```cpp
        if (ans[i])
            printf("%d\n", ans[i]);
    return 0;
}

```

## 强连通分量

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5 + 10;
const int inf = 1e9;
int read()
{
    int num = 0;
    bool flag = 1;
    char c = getchar();
    for (; c < '0' || c > '9'; c = getchar())
        if (c == '-')
            flag = 0;
    for (; c >= '0' && c <= '9'; c = getchar())
        num = (num << 1) + (num << 3) + c - '0';
    return flag ? num : -num;
}
vector<int> e[N];
int n, m, dfn[N], low[N], v[N], idx;
vector<vector<int>> scc;
stack<int> st;
void Tarjan(int x)
{
    dfn[x] = low[x] = ++idx;
    st.push(x);
    v[x] = 1;
    for (auto y : e[x])
    {
        if (!dfn[y])
        {
            Tarjan(y);
            low[x] = min(low[x], low[y]);
        }
        if (v[y])
            low[x] = min(low[x], dfn[y]);
    }
    if (low[x] == dfn[x])
    {
        vector<int> c;
        while (1)
        {
            int now = st.top();
            st.pop();
            v[now] = 0;
            c.push_back(now);
            if (now == x)
                break;
        }
        sort(c.begin(), c.end());
        scc.push_back(c);
    }
}
signed main()
{
    n = read();
    m = read();
    for (int i = 1; i <= m; i++)
    {
        int x = read(), y = read();
        e[x].push_back(y);
    }
    for (int i = 1; i <= n; i++)
        if (!dfn[i])
            Tarjan(i);
    sort(scc.begin(), scc.end());
    for (auto now : scc)
    {
        for (auto x : now)
            printf("%d ", x);
        printf("\n");
    }
    return 0;
}

```

## 树链剖分

```
#include <bits/stdc++.h>
using namespace std;
int read()
{
    int num = 0;
    bool flag = 1;
```

```cpp
    char c = getchar();
    for (; c < '0' || c > '9'; c = getchar())
        if (c == '-')
            flag = 0;
    for (; c >= '0' && c <= '9'; c = getchar())
        num = (num << 1) + (num << 3) + c - '0';
    return flag ? num : -num;
}
const int N = 1e5 + 10;
int n;
struct Chain
{
    struct Tr
    {
        int Max, sum, l, r;
#define l(p) tree[p].l
#define r(p) tree[p].r
#define Max(p) tree[p].Max
#define sum(p) tree[p].sum
    } tree[N << 2];
    void build(int p, int l, int r)
    {
        l(p) = l;
        r(p) = r;
        if (l == r)
        {
            sum(p) = Max(p) = w[id[l]];
            return;
        }
        int mid = l + r >> 1;
        build(p << 1, l, mid);
        build(p << 1 | 1, mid + 1, r);
        sum(p) = sum(p << 1) + sum(p << 1 | 1);
        Max(p) = max(Max(p << 1), Max(p << 1 | 1));
    }
    void change(int p, int x, int z)
    {
        if (l(p) == r(p))
        {
            sum(p) = Max(p) = z;
            return;
        }
        int mid = l(p) + r(p) >> 1;
        if (x <= mid)
            change(p << 1, x, z);
        else
            change(p << 1 | 1, x, z);
        sum(p) = sum(p << 1) + sum(p << 1 | 1);
        Max(p) = max(Max(p << 1), Max(p << 1 | 1));
    }
    int ask_Max(int p, int l, int r)
    {
        if (l(p) >= l && r(p) <= r)
            return Max(p);
        int mid = l(p) + r(p) >> 1, Max = -INT_MAX;
        if (l <= mid)
            Max = max(Max, ask_Max(p << 1, l, r));
        if (r > mid)
            Max = max(Max, ask_Max(p << 1 | 1, l, r));
        return Max;
    }
    int ask_sum(int p, int l, int r)
    {
        if (l(p) >= l && r(p) <= r)
            return sum(p);
        int mid = l(p) + r(p) >> 1, sum = 0;
        if (l <= mid)
            sum += ask_sum(p << 1, l, r);
        if (r > mid)
            sum += ask_sum(p << 1 | 1, l, r);
        return sum;
    }

    struct cow
    {
        int x, y;
    } e[N << 1];
    int head[N], tot, w[N], id[N], dfn[N];
    void inse(int xxxx, int yyyy)
    {
        e[++tot].x = head[xxxx];
        head[xxxx] = tot;
        e[tot].y = yyyy;
    }
    int fa[N], dep[N], sz[N], hv[N];
    void dfs1(int x, int f)
    {
        dep[x] = dep[f] + 1;
        sz[x] = 1;
        fa[x] = f;
        for (int i = head[x]; i; i = e[i].x)
        {
            int y = e[i].y;
            if (y == f)
```

```cpp
                continue;
            dfs1(y, x);
            sz[x] += sz[y];
            if (sz[y] > sz[hv[x]])
                hv[x] = y;
        }
}
int top[N];
void dfs2(int x, int t)
{
    top[x] = t;
    dfn[x] = ++tot;
    id[tot] = x;
    if (hv[x] == 0)
        return;
    dfs2(hv[x], t);
    for (int i = head[x]; i; i = e[i].x)
    {
        int y = e[i].y;
        if (y == fa[x] || y == hv[x])
            continue;
        dfs2(y, y);
    }
}
void build_tree()
{
    for (int i = 1; i < n; i++)
    {
        int x = read(), y = read();
        inse(x, y);
        inse(y, x);
    }
    for (int i = 1; i <= n; i++)
        w[i] = read();
    dfs1(1, 0);
    tot = 0;
    dfs2(1, 1);
    build(1, 1, n);
}
int Ask_Max(int x, int y)
{
    int Max = -INT_MAX;
    while (top[x] != top[y])
    {
        if (dep[top[x]] < dep[top[y]])
            swap(x, y);
        Max = max(Max, ask_Max(1, dfn[top[x]], dfn[x]));
```

```
            x = fa[top[x]];
        }
        if (dep[x] < dep[y])
            swap(x, y);
        Max = max(Max, ask_Max(1, dfn[y], dfn[x]));
        return Max;
    }
    int Ask_sum(int x, int y)
    {
        int sum = 0;
        while (top[x] != top[y])
        {
            if (dep[top[x]] < dep[top[y]])
                swap(x, y);
            sum += ask_sum(1, dfn[top[x]], dfn[x]);
            x = fa[top[x]];
        }
        if (dep[x] < dep[y])
            swap(x, y);
        sum += ask_sum(1, dfn[y], dfn[x]);
        return sum;
    }
} T;
signed main()
{
    n = read();
    T.build_tree();
    int q = read();
    while (q--)
    {
        char op[10];
        scanf("%s", op + 1);
        if (op[1] == 'Q')
        {
            if (op[2] == 'M')
                printf("%d\n", T.Ask_Max(read(), read()));
            else
                printf("%d\n", T.Ask_sum(read(), read()));
        }
        else
        {
            int l = T.dfn[read()], t = read();
            T.change(1, l, t);
        }
    }
    return 0;
}
```

```

```

## 线性基

```cpp
#include <bits/stdc++.h>
using namespace std;
#define ll long long
ll read()
{
    ll num = 0;
    bool flag = 1;
    char c = getchar();
    for (; c < '0' || c > '9'; c = getchar())
        if (c == '-')
            flag = 0;
    for (; c >= '0' && c <= '9'; c = getchar())
        num = (num << 1) + (num << 3) + c - '0';
    return flag ? num : -num;
}
const int B = 60;
struct linear_basis
{
    ll num[B + 10];
    int hv;
    bool insert(ll x)
    {
        for (int i = B - 1; i >= 0; i--)
        {
            if (x >> i & 1)
            {
                if (num[i] == 0)
                {
                    num[i] = x;
                    hv++;
                    return 1;
                }
                x ^= num[i];
            }
        }
        return 0;
    }
    ll querymin(ll x)
    {
        for (int i = B - 1; i >= 0; i--)
            x = min(x, x ^ num[i]);
        return x;
    }
    ll querymax(ll x)
    {
        for (int i = B - 1; i >= 0; i--)
            x = max(x, x ^ num[i]);
        return x;
    }
    // x     ǰ   C/
} ba;
**  Ŀ  û n                 û                        0     **//
//         insert    false ǵ          0
int n;
ll k;
signed main()
{
    n = read();
    k = read();
    ll now = 1;
    for (int i = 1; i <= n; i++)
        if (!ba.insert(read()))
            now *= 2;
    k = k / now;
    ll ans = 0;
    for (int i = B - 1; i >= 0; i--)
    {
        if (ba.num[i] == 0)
            continue;
        if (k >= (1ll << ba.hv - 1))
        {
            k -= (1ll << ba.hv - 1);
            ans = max(ans, ans ^ ba.num[i]);
        }
        else
            ans = min(ans, ans ^ ba.num[i]);
        ba.hv--;
    }
    printf("%lld\n", ans);
    return 0;
}

```

## 行列式计算

```cpp
#include <bits/stdc++.h>
```

```cpp

#define IOS ios::sync_with_stdio(false), cin.tie(0)
#define ll long long
using namespace std;
const int N = 205;
ll g[N][N];

ll Rainbow(int n, int mod)
{
    ll ans = 1;
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= n; j++)
            g[i][j] %= mod;
    for (int i = 1; i <= n; i++)
    {
        for (int j = i + 1; j <= n; j++)
        {
            int x = i, y = j;
            while (g[x][i])
            {
                int t = g[y][i] / g[x][i];
                for (int k = i; k <= n; k++)
                    g[y][k] = (g[y][k] - t * g[x][k]) % mod;
                swap(x, y);
            }
            if (x == i)
            {
                for (int k = i; k <= n; k++)
                    swap(g[j][k], g[i][k]);
                ans = -ans;
            }
        }
        if (g[i][i] == 0)
            return 0;
        ans = ans * g[i][i] % mod;
    }
    if (ans < 0)
        ans += mod;
    return ans;
    // 近似于n^3
}

int main()
{
    int n, m, mod;
    cin >> n >> m >> mod;
    for (int i = 0; i < m; i++)
    {
        int u, v;
        cin >> u >> v;
        g[u][v]--, g[v][u]--;
        g[u][u]++, g[v][v]++;
    }
    cout << Rainbow(n - 1, mod) << "\n";
    return 0;
}
```

## 最小费用流

```
#include <bits/stdc++.h>
using namespace std;
int read()
{
    int num = 0;
    bool flag = 1;
    char c = getchar();
    for (; c < '0' || c > '9'; c = getchar())
        if (c == '-')
            flag = 0;
    for (; c >= '0' && c <= '9'; c = getchar())
        num = (num << 1) + (num << 3) + c - '0';
    return flag ? num : -num;
}
const int N = 20100;
const int M = 201000;
template <typename T>
struct MinCostGraph
{
    int s, t, vtot, head[N], etot;
    T dis[N], flow, cost;
    int pre[N];
    bool vis[N];
    struct edge
    {
        int v, nxt;
        T f, c;
    } e[M << 2];
    void addedge(int u, int v, T f, T c, T f2 = 0)
    {
        e[etot] = {v, head[u], f, c};
        head[u] = etot++;
        e[etot] = {u, head[v], f2, -c};
        head[v] = etot++;
    }
    bool spfa()
    {
        T inf = numeric_limits<T>::max() / 2;
        for (int i = 1; i <= vtot; ++i)
        {
            dis[i] = inf;
            vis[i] = 0;
            pre[i] = -1;
        }
        dis[s] = 0;
        vis[s] = 1;
        queue<int> q;
        q.push(s);
        while (!q.empty())
        {
            int u = q.front();
            for (int i = head[u]; ~i; i = e[i].nxt)
            {
                int v = e[i].v;
                if (e[i].f && dis[v] > dis[u] + e[i].c)
                {
                    dis[v] = dis[u] + e[i].c;
                    pre[v] = i;
                    if (!vis[v])
                    {
                        vis[v] = 1;
                        q.push(v);
                    }
                }
            }
            q.pop();
            vis[u] = 0;
        }
        return dis[t] != inf;
    }
    void augment()
    {
        int u = t;
        T f = numeric_limits<T>::max();
        while (~pre[u])
        {
            f = min(f, e[pre[u]].f);
            u = e[pre[u] ^ 1].v;
        }
        flow += f;
        cost += f * dis[t];
        u = t;
        while (~pre[u])
        {
            e[pre[u]].f -= f;
            e[pre[u] ^ 1].f += f;
            u = e[pre[u] ^ 1].v;
        }
    }
    array<int, 2> solve()
    {
        flow = 0;
        cost = 0;
        while (spfa())
            augment();
        return {flow, cost};
    }

    void init(int s_, int t_, int vtot_)
    {
        s = s_;
        t = t_;
        vtot = vtot_;
        etot = 0;
        for (int i = 1; i <= vtot; ++i)
            head[i] = -1;
    }
};
MinCostGraph<int> g;
int n, m;
int main()
{
    n = read();
    m = read();
    g.init(1, n, n);
    for (int i = 1; i <= m; i++)
    {
        int u = read(), v = read(), f = read(), c = read();
        g.addedge(u, v, f, c);
    }
    array<int, 2> it = g.solve();
    printf("%d %d", it[0], it[1]);
    return 0;
}
```

## kmp

写法一

```cpp
string s, t;
cin >> s >> t;
ll n = s.length(), m = t.length();
s = " " + s;
t = " " + t;
ll nxt[m + 1] = {0}, f[n + 1] = {0};
ll j = 0;
rep(i, 2, m)
{
    while (j > 0 && t[j + 1] != t[i])
        j = nxt[j];
    if (t[j + 1] == t[i])
        j++;
    nxt[i] = j;
}
j = 0;
rep(i, 1, n)
{
    while ((j == m) || (j > 0 && t[j + 1] != s[i]))
        j = nxt[j];
    if (t[j + 1] == s[i])
        j++;
    f[i] = j;
}
```

写法二

```cpp
string s, t;
cin >> s >> t;
ll n = s.length(), m = t.length();
t =  " " +  t + "#" + s;
ll nxt[n + m + 2] = {0};
ll j = 0;
rep(i, 2, n + m + 2)
{
    while (j > 0 && t[j + 1] != t[i])
        j = nxt[j];
    if (t[j + 1] == t[i])
        j++;
    nxt[i] = j; // 从 m + 1 到 n + m + 1 遍历寻找 nxt[i] == m
}
```

```
//哈希
//H(x) = x % mod;
const int mod = 11;//注意是常量,或者写成int mod = 11;
vector<int> HashTable[11];//vector<vector<int>> HashTable(mod);
//计算哈希值
int Hash(int x)
{
    return x % mod;
}
//插入新值
void Insert(int x)
{
    int add = Hash(x);
    HashTable[add].push_back(x);
}
//查找元素
bool isExist(int x)
{
    int add = Hash(x);
    int len = HashTable[add].size();
    for(int i = 0; i < len; i++)
    {
        if(HashTable[add][i] == x)
            return true;
    }
    return false;
}
```

```
//字符串哈希
int p = 26;
int base = 37;//大于字符数量的素数:37 101 137等
int Hash(string s)
{
    int res = 0;
    int len = s.length();
    for(int i = 0; i < len; i++)
        res = (res * base + s[i] - 'a' + 1) % p;
    return res;
}
//计算子串
int a[N + 1];//1-n下标依次
int SonHash(string s)
{
    int res = 0;
    int len = s.length();
    for(int i = 0; i < len; i++)
        res = (res * base + s[i] - 'a' + 1) % p, a[i + 1] = res;
    return res;
}
int CalcSubstringHash(int l, int r)//子串哈希值
{
    int t = a[l - 1] * (int)pow(base, r - l + 1) % p;//快速幂
    return (a[r] - t + p) % p;
}
```

```
//整数二分
bool check(int x) { /*Code*/ }

int calc(int l, int r)
{
    while(r - l > 1)
    {
        int mid = (l + r) / 2;
        if(check(mid))
            l = mid;
        else
            r = mid;
    }
    return l;
}
```

```
//浮点数二分
bool check(double x) { /*Code*/ }

double calc(double l, double r)
{
    const double eps = 1e-6;//题目给出的精度
    while(r - l > eps)
    {
        double mid = (l + r) / 2;
        if(check(mid))
            l = mid;
        else
            r = mid;
    }
    return l;
}
```

```
//逆元
const int mod = 1e9 + 7;
int inv(int a)
{
    int ret = 1;
    int b = mod - 2;
    while(b)
    {
        if(b & 1)
            ret = ret * a % mod;
        b /= 2;
        a = a * a % mod;
    }
    return ret;
}
```

```
//阶乘逆元
const int mod = 1e9 + 7;
const int N = 1e6 + 10;
int Fac[N];
int Inv[N];
int inv(int a){...}
void init(int n)
{
    Fac[0] = 1;
    for(int i = 1; i <= n; i++)
        Fac[i] = Fac[i - 1] * i % mod;
    Inv[n] = inv(Fac[n]);
    for(int i = n; i >= 0; i--)
        Inv[i] = Inv[i + 1] * (i + 1) % mod;
}
```