yizimi在ACM和复习用的板子

一两年前的代码，仅供参考，可能会有很多错误或者被卡常的情况（比如点分治部分）

特别感谢Rhein_E对模板的大量补充

更新日志（从2020.10.25 -- |)

```
2021.03.09 补充了Rhein_E的模板
2021.02.22 更新了数论->Miller-Rabin
2021.02.22 更新了数论->Pollard-Rho(未卡常版)
2021.02.22 补更了数据结构->珂朵莉树（ODT）
2021.02.06 更新了其他->网络流->网络最大流->预流推进HLPP
2021.02.04 更新了图论->割点（割顶）
2020.12.28 优化和完善目录
2020.12.28 优化了代码，删去多余头文件，简化长度
2020.12.28 更新了数论->高斯消元
```

未完待续

注意：

本板子库常见define和头文件：

```
// 循环
#define go(i, j, n, k) for(int i = j; i <= n; i += k)
#define fo(i, j, n, k) for(int i = j; i >= n; i -= k)
#define rep(i, x) for(int i = h[x]; i; i = e[i].nxt)
// #define rep(i, x) for(int i = h[x]; i != -1; i = e[i].nxt) 后来版本一般采用此种写法
// #define curep(i, x) for(int i = cur[x]; i != -1; i = e[i].nxt) 在Dinic中所用残留网络写法
#define mn // 数组
#define inf 1 << 30
#define llinf 1ll << 60
#define ll long long
// 线段树常用
#define lson l, m, rt << 1
#define rson m + 1, r, rt << 1 | 1
#define root 1, n, 1
#define bson l, r, rt
#define mod // 模数
inline int read() { // 快读，如需 long long 版会说明
    int x = 0, f = 1; char ch = getchar();
    while(ch > '9' || ch < '0') { if(ch == '-') f = -f; ch = getchar(); }
    while(ch >= '0' && ch <= '9') { x = x * 10 + ch - '0'; ch = getchar(); }
    return x * f;
}
```

一.数论

1.快速幂

```
inline ll mul(ll a,ll b,ll mod){
    ll ans = 1;
    while(b){
        if(b & 1) ans=ans*a%p;
        a = a * a % p; b >>= 1;
    }
    return ans % mod;
}
```

2.欧拉函数

```
inline ll phi(ll n){
    ll ans=n;
    for(int i = 2; i * i <= n; i++){
        if(n % i == 0) ans = ans / i * (i - 1);
        while(n % i == 0) n /= i;
    }
    if(n != 1) ans = ans / n * (n - 1);
    return ans;
}
```

3.乘法逆元（线性求逆）

Form 1:

```
ll inv[mn], n, p;
int main(){
    n = read(), p = read();
    inv[1] = 1;
    cout<< 1 <<"\n";
    go(i, 2, n, 1){
        inv[i] = (-1 * ((p / i) * inv[p % i]) % p + p) % p;
        printf("%d\n", inv[i]);
    }
    return 0;
}
```

Form 2:
by Rhein_E

```
LL qpow(LL x, LL y, LL p) {
    LL res = 1;
    while (y) {
        if (y & 1) (res *= x) %= p;
        y >>= 1;
        (x *= x) %= p;
    }
    return res;
}
LL inverse1(LL x, LL p) { return qpow(x, p - 2, p); }
void exGCD(LL a, LL b, LL &x, LL &y) {
    if (!b) x = 1, y = 0;
    else {
        exGCD(b, a % b, y, x);
        y -= a / b * x;
    }
}
LL inverse2(LL x, LL p) {
    LL xx, yy;
    exGCD(x, p, xx, yy);
    xx = (xx % p + p) % p;
    return xx;
}
//Rhein_E
```

4.线性筛素数

（1）埃式筛法

//实质上这个算法不是所谓欧拉的线性筛；而是一个O(nlglgn)的算法，但是在数据很大时，由于这个算法的常数小，且lglgn在数据很大的情况下基本不变，所以在某种意义上这个算法好写且更优秀。

```
bool prime[mn];
int n,m,a;
inline void primee(int nn){
    go(i, 0, nn, 1){
        prime[i] = true;
    }
    prime[0] = prime[1] = false;
    go(i, 2, nn, 1){
        if(!prime[i])
            continue;
        go(j, 2 * i, nn, i){
            prime[j] = false;
        }
    }
}
int main(){
    n=read();m=read();
    primee(n);
    go(i, 1, m, 1){
        a = read();
        if(prime[a])
            cout << "Yes" << endl;
        else
            cout << "No" << endl;
    }
    return 0;
}
```

（2）欧拉筛法

```
int v[mn], prime[mn], m;
inline void get_prime(int n) {
    m = 0;
    go(i, 2, n, 1) {
        if(v[i] == 0) {
            v[i] = i;
            prime[++m] = i;
        }
        go(j, 1, m, 1) {
            if(prime[j] > v[i] || prime[j] > n / i) break;
            v[prime[j] * i] = prime[j];
        }
    }
}
int n, k;
int main() {
    n = read(), k = read();
    get_prime(n);
    go(i, 1, k, 1) {
        int x = read();
        if(v[x] == x) puts("Yes");
        else puts("No");
    }
    return 0;
}
```

5.扩展欧几里得

Form 1:

```
void ex_gcd(int a, int b, int &x, int &y) {
    if(b) ex_gcd(b, a % b, y, x), y -= (a / b) * x;
    else x = 1, y = 0;
}
```

Form 2:
by Rhein_E

```
void ex_gcd(int a, int b, int &x, int &y, int &d) {
    if (!b) d = a, x = 1, y = 0;
    else {
        ex_gcd(b, a % b, y, x, d);
        y -= (a / b) * x;
    }
}
```

6.费马小定理

```
inline ll phi(ll n){
    ll ans = n;
    for(int i = 2; i * i <= n; i++){
        if(n % i == 0){
            ans = ans / i * (i - 1);
        }
        while(n % i == 0)
            n /= i;
    }
    if(n != 1){
        ans = ans / n * (n - 1);
    }
    return ans;
}
inline ll mul(ll a,ll b,ll mod){
    ll ans = 1;
    while (b) {
        if (b & 1) ans = ans * a % mod;
        a = a * a % mod; b >>= 1;
    }
    return ans % mod;
}
ll a, b;
int main(){
    a = read(), b = read();
    ll phin = phi(b);
    cout << mul(a, phin - 1, b);
    return 0;
}
```

7.矩阵加速

```
struct mat {
    ll a[4][4];
    mat() { go(i, 1, 3, 1) go(j, 1, 3, 1) a[i][j] = 0; }
    void init() { go(i, 1, 3, 1) a[i][i] = 1; }
};
inline mat mat_mul(mat a, mat b) {
    mat ans;
    go(k, 1, 3, 1)  go(i, 1, 3, 1)  go(j, 1, 3, 1)
        ans.a[i][j] += a.a[i][k] * b.a[k][j] % mod, ans.a[i][j] %= mod;
    return ans;
}
inline mat mat_pow(mat a, ll b) {
    mat ans; ans.init();
    for(; b; b >>= 1) {
        if(b & 1) ans = mat_mul(ans, a);
        a = mat_mul(a, a);
    }
    return ans;
}
inline void mat_put(mat a) {
    go(i, 1, 3, 1) go(j, 1, 3, 1) printf("%lld%c", a.a[i][j], (j == 3) ? '\n' :
' ');
}
```

8.整除分块

P2261 [CQOI2007]余数求和——AC代码

```
ll ans, n, k, sum;
int main(){
    n = read(), k = read();
    for (ll l = 1, r; l <= n; l = r + 1){
        if(k / l != 0)
            r = min(k / (k / l), n);
        else
            r = n;
        sum += (k / l) * (r - l + 1) * (l + r) / 2;
    }
    ans = n * k - sum;
    cout << ans;
    return 0;
}
```

9.博弈论

（1）nim游戏

```
int ans, x, n, T;
int main(){
    T = read();
    while(T--) {
        n = read();
        ans = 0;
        go(i, 1, n, 1) x = read(), ans ^= x;  // （）
        if(ans) puts("Yes");
        else puts("No");
    }
    return 0;
}
```

10.高斯消元

Form 1:

```
#define eps 1e-9
double A[mn][mn], b[mn], x[mn];
int n;
// 返回0为无解，1为有解，若判断非零解需特判，此处为n * n + 1矩阵
inline int Gauss() {
    for(int i = 1; i <= n; i++) {
        int r = i;
        for(int j = i + 1; j <= n; j++) {
            if(fabs(A[r][i]) < fabs(A[j][i]))
                r = j;
        }
        if(fabs(A[r][i]) < eps) return 0;
        if(i != r) swap(A[i], A[r]);
        double div = A[i][i];
        for(int j = i; j <= n + 1; j++) {
            A[i][j] /= div;
        }
        for(int j = i + 1; j <= n; j++) {
            div = A[j][i];
            for(int k = i; k <= n + 1; k++)
                A[j][k] -= A[i][k] * div;
        }
    }
    x[n] = A[n][n + 1];
    for(int i = n - 1; i >= 1; i--) {
        x[i] = A[i][n + 1];
        for(int j = i + 1; j <= n; j++) {
            x[i] -= (x[j] * A[i][j]);
        }
    }
    return 1;
}
```

Form 2:
by Rhein_E

```
const double eps = 1e-9;

struct Matrix {
    double data[105][105];
    int r, c;
    Matrix(int _r = 0, int _c = 0) : r(_r), c(_c) {}
    void eliminate();
    void print();
    void solve();
} m;

int main() {
    scanf("%d", &m.r);
    m.c = m.r + 1;
    for (int i = 0; i < m.r; ++i)
        for (int j = 0; j < m.c; ++j)
            std::cin >> m.data[i][j];
    m.eliminate();
    m.print();
    m.solve();

    return 0;
}

void Matrix::eliminate() {
    for (int i = 0; i < r; ++i) {
        if (data[i][i] > -eps && data[i][i] < eps)
            for (int j = i + 1; j < r; ++j)
                if (data[j][i] > eps || data[j][i] < -eps)
                    std::swap(data[i], data[j]);
        for (int j = i + 1; j < r; ++j)
            for (int k = r; k >= i; --k)
                if (data[j][i] > eps || data[j][i] < -eps)
                    data[j][k] = data[j][k] * data[i][i] / data[j][i] - data[i]
[k];
    }
}
void Matrix::print() {
    for (int i = 0; i < r; ++i) {
        for (int j = 0; j < c; ++j)
            printf("%10.5lf ", data[i][j]);
        puts("");
    }
}
void Matrix::solve() {
    for (int i = r - 1; i >= 0; --i) {
        data[i][r + 1] = data[i][r];
        for (int j = i + 1; j < r; ++j) data[i][r + 1] -= data[j][r + 1] *
data[i][j];
        data[i][r + 1] /= data[i][i];
    }
    for (int i = 0; i < r; ++i)
        printf("%10.5lf ", data[i][r + 1]);
    puts("");
}
//Rhein_E
```

11.Miller-Rabin

```
class Miller_Rabin {
    private:
        inline ll mul(ll a, ll b, ll mod) {
            ll ans = 1ll;
            while(b) {
                if(b & 1) ans = ans * a % mod;
                a = (a * a) % mod, b >>= 1;
            }
            return ans % mod;
        }
        inline bool miller_rabin(ll n, ll p) {
            ll r = 0, s = n - 1;
            if(!(n % p)) return 0;
            while(!(s & 1)) ++r, s >>= 1;
            ll k = mul(p, s, n);
            if(k == 1) return 1;
            for(ll j = 0; j < r; ++j, k = k * k % n)
                if(k == n - 1) return 1;
            return 0;
        }
    public:
        inline bool is_prime(ll x) {
            ll p[] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41};
            for(int i = 0; i < 13; ++i) {
                if(x == p[i]) return 1;
                if(!miller_rabin(x, p[i])) return 0;
            }
            return 1;
        }
} mr;
```

12.Pollard-Rho(未卡常版)

```
class Pollard_Rho {
    private:
        inline ll mul(ll a, ll b, ll mod) {
            ll tmp = (a * b - (ll)((long double)a / mod * b + 1.0e-8) * mod);
            return tmp < 0 ? tmp + mod : tmp;
        }
        inline ll pow(ll a, ll b, ll mod) {
            ll ans = 1ll; a %= mod;
            while(b) {
                if(b & 1) ans = mul(a, ans, mod);
                a = mul(a, a, mod), b >>= 1;
            }
            return ans % mod;
        }
        inline bool miller_rabin(ll n, ll p) {
            ll r = 0, s = n - 1;
            if(!(n % p)) return 0;
            while(!(s & 1)) ++r, s >>= 1;
            ll k = pow(p, s, n);
            if(k == 1) return 1;
            for(ll j = 0; j < r; ++j, k = mul(k, k, n))
                if(k == n - 1) return 1;
            return 0;
        }
        ll gcd(ll a, ll b) {
            if(b == 0) return a;
            return gcd(b, a % b);
        }
        inline ll pollard_rho(ll n, ll c) {
            ll x = rand() % (n - 2) + 1;
            ll y = x, i = 1, k = 2;
            while(1) {
                ++i;
                x = (mul(x, x, n) + c) % n;
                ll d = gcd(y - x, n);
                if(1 < d and d < n) return d;
                if(y == x) return n;
                if(i == k) y = x, k <<= 1;
            }
        }
    public:
        std::map<ll, int> mp;
        ll max_factor = 0;
        inline bool is_prime(ll x) {
            if(x == 2 || x == 3) return 1;
            if(x % 2 == 0 || x == 1) return 0;
            ll p[] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47,
53, 59, 61, 67, 71, 73, 79, 83, 89, 97};
            for(int i = 0; i < 25; ++i) {
                if(x == p[i]) return 1;
                if(!miller_rabin(x, p[i])) return 0;
            }
            return 1;
        }
        inline void find_factor(ll n, ll c) {
            if(n == 1) return;
            if(is_prime(n)) { ++mp[n], max_factor = std::max(max_factor, n);
 return; }
            ll p = n;
            while(p >= n)
                p = pollard_rho(p, c--);
            find_factor(p, c);
            find_factor(n / p, c);
        }
} pr;
```

13.FFT

(1)FFT-iterative

by Rhein_E

```
// Luogu P3803
#include <cstdio>
#include <cstring>
```

```cpp
#include <iostream>
#include <cmath>
typedef long long LL;
const double PI = 3.14159265;
#ifdef ONLINE_JUDGE
const int MAXN = 4e6 + 10;
#else
const int MAXN = 100;
#endif
struct Complex {
    double real, imag;
    Complex(double a = 0, double b = 0) : real(a), imag(b) {}
};
Complex operator+(const Complex &a, const Complex &b) { return Complex(a.real +
b.real, a.imag + b.imag); }
Complex operator-(const Complex &a, const Complex &b) { return Complex(a.real -
b.real, a.imag - b.imag); }
Complex operator-(const Complex &a) { return Complex(-a.real, -a.imag); }
Complex operator*(const Complex &a, const Complex &b) {
    return Complex(a.real * b.real - a.imag * b.imag, a.real * b.imag + a.imag *
b.real);
}
Complex a[MAXN], b[MAXN];
int n, m;

void FFT(Complex *arr, int sz, bool IFFT = 0) {
    static int rev[MAXN], bit;
    for (bit = 0; (1 << bit) < sz; ++bit)
        ;
    for (int i = 1; i < sz; ++i) {
        rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << bit - 1);
        if (i < rev[i]) std::swap(arr[i], arr[rev[i]]);
    }
    for (int half = 1; half < sz; half <<= 1) {
        Complex wn(cos(PI / half), (IFFT ? -1 : 1) * sin(PI / half));
        for (int len = half << 1, st = 0; st < sz; st += len) {
            Complex w(1, 0);
            for (int i = 0; i < half; ++i, w = w * wn) {
                Complex x = arr[st + i], y = w * arr[st + i + half];
                arr[st + i] = x + y, arr[st + i + half] = x - y;
            }
        }
    }
    if (IFFT) {
        for (int i = 0; i < sz; ++i)
            arr[i].real = (arr[i].real + 0.5) / sz;
    }
}
int main() {
#ifndef ONLINE_JUDGE
    freopen("C:\\Users\\Rhein_E\\Desktop\\code\\cpp\\templates\\in.txt", "r",
stdin);
    freopen("C:\\Users\\Rhein_E\\Desktop\\code\\cpp\\templates\\out.txt", "w",
stdout);
#endif
    std::cin >> n >> m;
    for (int i = 0; i <= n; ++i)
        std::cin >> a[i].real;
    for (int i = 0; i <= m; ++i)
        std::cin >> b[i].real;
    int N = 1, bit = 0;
    while (N <= n + m)
        N <<= 1;
    FFT(a, N);
    FFT(b, N);
    for (int i = 0; i < N; ++i)
        a[i] = a[i] * b[i];
    FFT(a, N, 1);
    for (int i = 0; i <= n + m; ++i)
        std::cout << (int)a[i].real << " ";
    std::cout << std::endl;

    return 0;
}
// Rhein_E
```

(2)FFT-recursive

by Rhein_E

```
#include <cstdio>
#include <cstring>
#include <iostream>
#include <cmath>
typedef long long LL;
const double PI = 3.14159265;
#ifdef ONLINE_JUDGE
const int MAXN = 4e6 + 10;
#else
const int MAXN = 100;
#endif
struct Complex {
    double real, imag;
    Complex(double a = 0, double b = 0) : real(a), imag(b) {}
};
Complex operator+(const Complex &a, const Complex &b) { return Complex(a.real +
b.real, a.imag + b.imag); }
Complex operator-(const Complex &a, const Complex &b) { return Complex(a.real -
b.real, a.imag - b.imag); }
Complex operator*(const Complex &a, const Complex &b) {
    return Complex(a.real * b.real - a.imag * b.imag, a.real * b.imag + a.imag *
b.real);
}
Complex operator-(const Complex &a) { return Complex(-a.real, -a.imag); }
void FFT(Complex *arr, int n, bool IFFT = 0) {
    if (n == 1) return;
    Complex *a0 = new Complex[n >> 1], *a1 = new Complex[n >> 1];
    for (int i = 0; i < (n >> 1); ++i)
        a0[i] = arr[i << 1], a1[i] = arr[i << 1 | 1];
    FFT(a0, n >> 1, IFFT), FFT(a1, n >> 1, IFFT);
    Complex wn(cos(2 * PI / n), (IFFT ? -1 : 1) * sin(2 * PI / n)), w(1, 0);
    for (int i = 0; i < (n >> 1); ++i) {
        arr[i] = a0[i] + w * a1[i];
        arr[i + (n >> 1)] = a0[i] - w * a1[i];
        w = w * wn;
    }
}
Complex a[MAXN], b[MAXN];
int n, m;
int main() {
#ifndef ONLINE_JUDGE
    freopen("C:\\Users\\Rhein_E\\Desktop\\code\\cpp\\templates\\in.txt", "r",
stdin);
    freopen("C:\\Users\\Rhein_E\\Desktop\\code\\cpp\\templates\\out.txt", "w",
stdout);
#endif
    std::cin >> n >> m;
    for (int i = 0; i <= n; ++i)
        std::cin >> a[i].real;
    for (int i = 0; i <= m; ++i)
        std::cin >> b[i].real;
    int N = 1;
    while (N <= n + m)
        N <<= 1;
    FFT(a, N);
    FFT(b, N);
    for (int i = 0; i < N; ++i)
        a[i] = a[i] * b[i];
    FFT(a, N, 1);
    for (int i = 0; i <= n + m; ++i)
        std::cout << (int)(a[i].real / N + 0.5) << " ";
    std::cout << std::endl;
    return 0;
}
// Rhein_E
```

14.线性异或方程组（在线）

by Rhein_E

```
#include <cstdio>
#include <cstring>
#include <iostream>

inline char gc() {
    static char buf[1000000], *p1, *p2;
    if (p1 == p2) p1 = (p2 = buf) + fread(buf, 1, 1000000, stdin);
    return p1 == p2 ? EOF : *p2++;
}
inline int read() {
    int res = 0; char ch = gc(), sg = 0;
    while (ch != '-' && (ch < '0' || ch > '9')) ch = gc();
    if (ch == '-') ch = gc(), sg = 1;
    while (ch >= '0' && ch <= '9') res = (res << 1) + (res << 3) + ch - '0', ch
= gc();
    return sg ? -res : res;
}

const int MAXN = 32;
const int MAXM = 110;
int a[MAXM]; //记录方程
int c[MAXN]; //记录关键元为xi的方程的编号
int main() {
    //init
    memset(c, -1, sizeof(c));

    //input & solve
    //读入m个方程的系数及常数项
    bool flag = 1;
    int n = read(), m = read();
    for (int i = 0; i < m; ++i) {
        for (int j = 0; j < n; ++j)
            if (read()) a[i] |= (1 << j);
        if (read()) a[i] |= (1 << n);
        for (int j = 0; j < n; ++j)
            if (a[i] & (1 << j))
                if (~c[j]) a[i] ^= a[c[j]]; //关键元被占用，消去当前方程中的xj
                else {
                    c[j] = i; //把当前方程的关键元设为xj
                    break;
                }
        if (a[i] == (1 << n)) flag = 0; //消元后得到0=1的形式，无解
    }
    if (!flag) puts("No Solution.\n");
    else {
        for (int i = n - 1; i >= 0; --i)
            if (~c[i])
                for (int j = i - 1; j >= 0; --j)
                    if ((a[c[j]] >> i) & 1)
                        a[c[j]] ^= a[c[i]];
        for (int i = 0; i < n; ++i)
            if (~c[i]) printf("%d ", (a[c[i]] >> n) & 1);
            else printf("0 "); //xi可以为1或0，如果为1，需要向上消元
        puts("");
    }

    return 0;
}
```

15.线性递归

by Rhein_E

```
//Fibonacci
#include <cstdio>
#include <iostream>
#include <cstring>

typedef long long LL;
const int MAXN = 105;

struct Matrix {
    int n;
    LL data[MAXN][MAXN];
    Matrix(int _n = 0) : n(_n) { memset(data, 0, sizeof(data)); }
    Matrix operator*(Matrix a) {
        Matrix res(n);
        for (int i = 0; i < n; ++i)
```

```
            for (int j = 0; j < n; ++j)
                for (int k = 0; k < n; ++k)
                    res.data[i][j] += data[i][k] * a.data[k][j];
        return res;
    }
    static Matrix identity(int x) {
        Matrix res(x);
        for (int i = 0; i < x; ++i) res.data[i][i] = 1;
        return res;
    }
} base(2), f(2);
int n;

Matrix pow(Matrix x, int y) {
    Matrix res = Matrix::identity(x.n);
    while (y) {
        if (y & 1) res = res * x;
        x = x * x;
        y >>= 1;
    }
    return res;
}

int main() {
    while (~scanf("%d", &n)) {
        base = Matrix(2), f = Matrix(2);
        base.data[0][0] = base.data[0][1] = base.data[1][0] = 1;
        f.data[0][0] = f.data[1][0] = 1;
        f = pow(base, n - 1) * f;
        printf("%lld\n", f.data[0][0]);
    }
    return 0;
}
// Rhein_E
```

16.NTT

(1)NTT-iterative

by Rhein_E

```
// Luogu P3803
#include <cstdio>
#include <cstring>
#include <iostream>

typedef long long LL;
const LL G = 3;
const LL MOD = 998244353;
const int MAXN = 5e6;

LL qpow(LL x, LL y) {
    LL res = 1;
    while (y) {
        if (y & 1) res = (res * x) % MOD;
        y >>= 1;
        x = (x * x) % MOD;
```

```cpp
    }
    return res;
}

void NTT(LL *arr, int n, bool inv = 0) {
    static int rev[MAXN], bit;
    for (bit = 0; (1 << bit) < n; ++bit)
        ;
    for (int i = 1; i < n; ++i) {
        rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << bit - 1);
        if (i < rev[i]) std::swap(arr[i], arr[rev[i]]);
    }
    for (int len = 2; len <= n; len <<= 1) {
        LL wn = qpow(G, (MOD - 1) / len);
        if (inv) wn = qpow(wn, MOD - 2);
        for (int i = 0, half = len >> 1; i < n; i += len) {
            LL w = 1;
            for (int j = 0; j < half; ++j, w = w * wn % MOD) {
                LL x = arr[i + j], y = w * arr[i + j + half] % MOD;
                arr[i + j] = (x + y) % MOD, arr[i + j + half] = (x - y + MOD) %
MOD;
            }
        }
    }
    if (inv) {
        LL tmp = qpow(n, MOD - 2);
        for (int i = 0; i < n; ++i)
            arr[i] = (arr[i] * tmp) % MOD;
    }
}
LL a[MAXN], b[MAXN];
int N, M;

int main() {
#ifndef ONLINE_JUDGE
    freopen("C:\\Users\\Rhein_E\\Desktop\\code\\cpp\\templates\\in.txt", "r",
stdin);
    freopen("C:\\Users\\Rhein_E\\Desktop\\code\\cpp\\templates\\out.txt", "w",
stdout);
#endif
    scanf("%d%d", &N, &M);
    for (int i = 0; i <= N; ++i)
        scanf("%lld", a + i);
    for (int i = 0; i <= M; ++i)
        scanf("%lld", b + i);
    int lim = 1;
    while (lim <= M + N)
        lim <<= 1;
    NTT(a, lim);
    NTT(b, lim);
    for (int i = 0; i < lim; ++i)
        a[i] = a[i] * b[i] % MOD;
    NTT(a, lim, 1);
    for (int i = 0; i <= N + M; ++i)
        printf("%lld ", a[i]);

    return 0;
}
```

```
// Rhein_E
```

二.图论

1.并查集

```
int fa[mn], n, m;
inline int findx(int x) {
    if(x == fa[x]) return x;
    else return fa[x] = findx(fa[x]);
}
inline void mergex(int x, int y) {
    int xx = findx(x), yy = findx(y);
    if(xx == yy) return;
    if(rand() % 2) fa[xx] = yy;
    else fa[yy] = xx;
}
int main() {
    n = read(), m = read();
    go(i, 1, n, 1) fa[i] = i;
    go(i ,1, m, 1) {
        int s = read(), x = read(), y = read();
        if(s == 1) mergex(x, y);
        else {
            int xx = findx(x), yy = findx(y);
            if(xx == yy) puts("Y");
            else puts("N");
        }
    }
    return 0;
}
```

2.Kruskal算法

```
struct edge{
    int x, y, w;
} e[mm];
int n, m, b[mn], f[mn], sum = 0, num = 0;
inline int findx(int x) {
    return f[x] == x ? x : f[x] = findx(f[x]);
}
inline bool cmp(edge a,edge b) {
    return a.w < b.w;
}
inline void Kru() {
    go(i, 1, n, 1) {
        f[i] = i;
    }
    sort(e + 1, e + m + 1, cmp);
    go(i, 1, m, 1) {
        int u = findx(e[i].x);
        int v = findx(e[i].y);
        if(u != v) {
            f[u] = v;
            sum += e[i].w;
            num++;
            if(num == n - 1)
                return;
        }
    }
}
```

3.Dijkstra算法（优化版）

```
struct edge {
    int v, nxt, w;
} e[mn << 1];
int h[mn], p;
inline void add(int a, int b, int c) {
    e[++p].nxt = h[a], h[a] = p;
    e[p].v = b, e[p].w = c;
}
struct node{
    int x, d;
    bool operator < (const node &b) const { return d > b.d; }
};
priority_queue<node> q;
int n, m, dis[mn];
inline void Dij(int s) {
    go(i, 1, n, 1) dis[i] = inf;
    node o; o.x = s, o.d = 0;
    q.push(o), dis[s] = 0;
    while(!q.empty()) {
        node o = q.top(); q.pop();
        if(o.d != dis[o.x]) continue;
        int x = o.x, d = o.d;
        rep(i, x) {
            int v = e[i].v, w = e[i].w;
            if(dis[v] > dis[x] + w) {
                dis[v] = dis[x] + w;
                node oo; oo.x = v, oo.d = dis[v];
                q.push(oo);
            }
        }
    }
}
```

4.SPFA算法

```
struct edge{
    int v, nxt, w;
} e[mn << 1];
int h[mn], p;
inline void add(int a,int b,int c){
    e[++p].nxt = h[a], h[a] = p, e[p].v = b, e[p].w = c;
}
int dis[mn], vis[mn], n, m, s, t;
inline void SPFA(int s){
    go(i, 1, n, 1) dis[i] = inf;
    queue<int> q;
    q.push(s), dis[s] = 0, vis[s] = 1;
    while(!q.empty()){
        int x = q.front();
        q.pop(); vis[x] = 0;
        rep(i, x){
            int v = e[i].v, w = e[i].w;
            if(dis[v] > dis[x] + w){
                dis[v] = dis[x] + w;
                if(!vis[v]){
                    q.push(v), vis[v] = 1;
                }
            }
        }
    }
}
```

5.span id = "2.05">Floyd（已优化）<

```
int a[mn][mn], s, m, n, d, b, c;
inline void floyd(){
    go(k, 1, n, 1){
        go(i, 1, n, 1){
            if(i == k || a[i][k] == inf)
                continue;
            go(j, 1, n, 1){
                if(i == j || j == k || a[k][j] == inf)
                    continue;
                if(a[i][k] + a[k][j] < a[i][j]){
                    a[i][j] = a[i][k] + a[k][j];
                }
            }
        }
    }
}

```

6.二分图染色

```
bool vis[mk];
int c[mk];
vector<int> G[mk];
inline bool color(int u){
    vis[u] = true;
    go(i, 0, G[u].size() - 1, 1){
        int v = G[u][i];
        if(vis[v]){
            if(c[v] != c[u])
                return false;
        } else {
            c[v] = !c[u];
            if(!color(v))
                return false;
        }
    }
    return true;
}
```

7.拓扑排序

```
struct edge{ int v, nxt; } e[mm << 1]; int h[mn], p;
inline void add(int a, int b) { e[++p].nxt = h[a], h[a] = p, e[p].v = b; }
int n, m, du[mn];
vector<int> sorted;
queue<int> q;
inline void topsort() {
    go(i, 1, n, 1) if(!du[i]) q.push(i);
    while(!q.empty()) {
        int x = q.front(); q.pop();
        sorted.push_back(x);
        rep(i, x) {
            int v = e[i].v;
            if(!--du[v]) q.push(v);
        }
    }
}
int main() {
    n = read(), m = read();
    go(i, 1, m, 1) {
        int a = read(), b = read();
        add(a, b); du[b]++;
    }
    topsort();
    int sze = sorted.size();
    go(i, 0, sze - 1, 1)
        printf("%d ", sorted[i]);
    return 0;
}
```

8.缩点（tarjan求强联通分量）

```
struct edge{ int v, nxt; } e[mn << 1], ee[mn << 1]; int h[mn], p, hh[mn], pp;
inline void add(int a, int b) { e[++p].nxt = h[a], h[a] = p, e[p].v = b; }
inline void aadd(int a, int b) { ee[++pp].nxt = hh[a], hh[a] = pp, ee[pp].v = b;
}
int dfn[mn], low[mn], co[mn], st[mn], top, cnt, ans, col, n, m, w[mn], b[mn],
du[mn], dp[mn];
void tarjan(int x) {
    dfn[x] = low[x] = ++cnt;
    st[++top] = x;
    rep(i, x) {
        int v = e[i].v;
        if(!dfn[v]) {
            tarjan(v);
            low[x] = min(low[x], low[v]);
        } else if(!co[v]) {
            low[x] = min(low[x], dfn[v]);
        }
    }
    if(dfn[x] == low[x]) {
        ++col;
        while(st[top + 1] != x) {
            co[st[top]] = col;
            b[col] += w[st[top]];
            top--;
        }
    }
}
void dfs(int x, int f) {
    if(dp[x]) return;
    dp[x] = b[x];
    int maxx = 0;
    for(int i = hh[x]; i; i = ee[i].nxt) {
        int v = ee[i].v;
        if(v == f) continue;
        dfs(v, x);
        maxx = max(maxx, dp[v]);
    }
    dp[x] += maxx;
}
inline void debug() {
    go(i, 1, col, 1) printf("%d%c", b[i], (i == col) ? '\n' : ' ');
}
int main() {
    n = read(), m = read();
    go(i, 1, n, 1) w[i] = read();
    go(i, 1, m, 1) {
        int a = read(), b = read();
        add(a, b);
    }
    go(i, 1, n, 1) if(!dfn[i]) tarjan(i);
    go(x, 1, n, 1) {
        rep(i, x) {
            int v = e[i].v;
            if(co[x] != co[v])
                aadd(co[x], co[v]), du[co[v]]++;
        }
    }
    // debug();
    go(i, 1, col, 1) {
        if(!dp[i]) dfs(i, 0), ans = max(ans, dp[i]);
    }
    cout << ans << "\n";
    return 0;
}
```

9.割点（割顶）

```
inline void tarjan(int x, int rt) { // rt 是根节点QwQ
    int rtc = 0;
    dfn[x] = low[x] = ++cnt;
    rep(i, x) {
        int v = e[i].v;
        if(!dfn[v]) {
            tarjan(v, rt);
            low[x] = std::min(low[x], low[v]);
            if(low[v] >= dfn[x] && x != rt) cut[x] = 1;
            if(x == rt) rtc++;
        }
        low[x] = std::min(low[x], dfn[v]);
    }
    if(x == rt && rtc >= 2) cut[rt] = 1;
}

inline void solve() {
    n = read(), m = read();
    go(i, 1, m, 1) {
        int a = read(), b = read();
        add(a, b), add(b, a);
    }
    go(i, 1, n, 1) if(!dfn[i]) tarjan(i, i);
    go(i, 1, n, 1) ans += cut[i];
    std::printf("%d\n", ans);
    go(i, 1, n, 1) if(cut[i]) printf("%d\n", i);
}
inline void init() {
    memset(h, -1, sizeof h);
    p = -1;
}
```

10.树分治

（1）点分治（包括求树的重心）

```
struct edge { int v, nxt, w; } e[mn << 1]; int h[mn], p = -1;
inline void add(int a, int b, int c) { e[++p].nxt = h[a], h[a] = p, e[p].v = b,
e[p].w = c; }
int maxp[mn], sze[mn], dis[mn], rem[mn], vis[mn], test[mn], judge[mn], q[mn *
100], query[mn];
int n, m, sum, rot, ans;
void get_root(int x, int f) {
    sze[x] = 1, maxp[x] = 0;
    rep(i, x) {
        int v = e[i].v;
        if(v == f || vis[v]) continue;
        get_root(v, x);
        sze[x] += sze[v];
        maxp[x] = max(maxp[x], sze[v]);
    }
    maxp[x] = max(maxp[x], sum - sze[x]);
    if(maxp[x] < maxp[rot]) rot = x;
}
void get_dis(int x, int f) {
    rem[++rem[0]] = dis[x];
    rep(i, x) {
        int v = e[i].v, w = e[i].w;
        if(v == f || vis[v]) continue;
        dis[v] = dis[x] + w;
        get_dis(v, x);
    }
}
inline void calc(int x) {
    int p = 0;
    rep(i, x) {
        int v = e[i].v, w = e[i].w;
        if(vis[v]) continue;
        rem[0] = 0, dis[v] = w;
        get_dis(v, x);
        fo(j, rem[0], 1, 1)
            go(k, 1, m, 1)
                if(query[k] >= rem[j])
                    test[k] |= judge[query[k] - rem[j]];
        fo(j, rem[0], 1, 1)
            q[++p] = rem[j], judge[rem[j]] = 1;
    }
    go(i, 1, p, 1)
        judge[q[i]] = 0;
}
void pit_div(int x) { // point divide
    vis[x] = judge[0] = 1;
    calc(x);
    rep(i, x) {
        int v = e[i].v;
        if(vis[v]) continue;
        sum = sze[v], maxp[rot = 0] = inf;
        get_root(v, 0);
        pit_div(v);
    }
}
inline void solve() {
    n = read(), m = read();
    go(i, 1, n - 1, 1) {
        int x = read(), y = read(), z = read();
        add(x, y, z), add(y, x, z);
    }
    go(i, 1, m, 1) query[i] = read();
    maxp[rot] = sum = n;
    get_root(1, -1);
    pit_div(rot);
    go(i, 1, m, 1)
        if(test[i]) puts("AYE");
        else puts("NAY");
}
inline void init() {
    memset(h, -1, sizeof h);
    p = -1;
}
int main() {
    init();
    solve();
    return 0;
}
```

11.网络流

（一）网络最大流

（1）EK算法

P3376 【模板】网络最大流——AC代码

P.S.注释的部分为邻接矩阵的操作

```
bool vis[mn];
int n, m;

struct edge { int v, nxt, w; } e[mn << 1]; int h[mn], p = -1;
inline void add(int a, int b, int c) { e[++p].nxt = h[a], h[a] = p, e[p].v = b,
e[p].w = c; }

struct node { int v, id; } pre[mn];

inline bool bfs(int s,int t) {
    memset(pre, -1, sizeof pre);
    memset(vis, 0, sizeof vis);
    vis[s] = true;
    queue<int> q;
    q.push(s);
    while(!q.empty()) {
        int x = q.front();
        q.pop();
//      go(i, 1, n, 1)
//          if(!vis[i] && g[x][i] > 0) {
//              vis[i] = true;
//              pre[i] = x;
//              if(i == t) return true;
//              q.push(i);
//          }
        rep(i, x) {
            int v = e[i].v, w = e[i].w;
            if(!vis[v] && w > 0) {
                vis[v] = true;
                pre[v].v = x;
                pre[v].id = i;
                if(v == t) return true;
                q.push(v);
            }
        }
    }
    return false;
}

inline int EK(int s, int t) {
    int d, maxflow;
    maxflow = 0;
    while(bfs(s, t)) {
        d = inf;
        for(int i = t; i != s; i = pre[i].v)
            d = min(d, e[pre[i].id].w);
        for(int i = t; i != s; i = pre[i].v) {
            e[pre[i].id].w -= d;
            e[pre[i].id ^ 1].w += d;
        }
        maxflow += d;
    }
    return maxflow;
}
inline void solve() {
    n = read(), m = read();
    int s = read(), t = read();
    go(i, 1, m, 1) {
        int x = read(), v = read(), w = read();
//      g[x][v] += w;
        add(x, v, w);
        add(v, x, 0);
    }
    cout << EK(s, t) << "\n";
}
inline void init() {
    memset(h, -1, sizeof h);
}

int main() {
    init();
    solve();
    return 0;
}
```

（2）Dinic算法

P3376 【模板】网络最大流——AC代码

```
int n, m;
struct edge {
    int v, nxt, w;
} e[mn << 1];
int h[mn], p = -1;
inline void add(int a, int b, int c) {
    e[++p].nxt = h[a], h[a] = p, e[p].v = b, e[p].w = c;
}
inline void add_flow(int a, int b, int c) { add(a, b, c), add(b, a, 0); }
int dep[mn], cur[mn];

inline bool bfs(int s, int t) {
    go(i, 1, n, 1) dep[i] = inf, cur[i] = h[i];
    queue<int> q; dep[s] = 0, q.push(s);
    while(!q.empty()) {
        int x = q.front(); q.pop();
        rep(i, x) {
            int v = e[i].v, w = e[i].w;
            if(dep[v] >= inf && w) {
                dep[v] = dep[x] + 1;
                q.push(v);
            }
        }
    }
    if(dep[t] < inf) return true;
    return false;
}
inline int dfs(int x, int t, int lim) {
    if(x == t || !lim) return lim;
    int flow = 0, f = 0;
    curep(i, x) {
        int v = e[i].v, w = e[i].w;
        cur[x] = i;
        if(dep[v] == dep[x] + 1 && (f = dfs(v, t, min(lim, w)))) {
            flow += f, lim -= f;
            e[i].w -= f, e[i ^ 1].w += f;
            if(!lim) break;
        }
    }
    return flow;
}
inline int Dinic(int s, int t) {
    int maxflow = 0;
    while(bfs(s, t)) maxflow += dfs(s, t, inf);
    return maxflow;
}

inline void solve() {
    n = read(), m = read();
    int s = read(), t = read(), x, y, z;
    go(i, 1, m, 1) x = read(), y = read(), z = read(), add_flow(x, y, z);
    cout << Dinic(s, t) << "\n";
}
inline void init() {
    memset(h, -1, sizeof h);
    p = -1;
}

int main () {
    init();
    solve();
    return 0;
}
```

(3) 预流推进HLPP

```
int n, m, dep[mn];
struct cmp {
    inline bool operator() (int a, int b) const { return dep[a] < dep[b]; }
};

class Preflow_propulsion {
    private:
        struct edge {
            int v, nxt; ll w;
        } e[mm << 1];
        int h[mn], p = -1;
        inline void add(int a, int b, ll c) {
            e[++p].nxt = h[a], h[a] = p, e[p].v = b, e[p].w = c;
        }
        // int G[mn][mn];
        ll flow[mn];
        // int dep[mn];
        int gap[mn << 1];
        int vis[mn];
        const int infx = 0x3f3f3f3f;
        inline bool bfs(int s, int t) {
            std::memset(dep, 0x3f, sizeof dep);
            std::queue<int> qq;
            qq.push(t), dep[t] = 0;
            while(!qq.empty()) {
                int x = qq.front(); qq.pop();
                rep(i, x) {
                    int v = e[i].v;
                    if(dep[v] > dep[x] + 1 && e[i ^ 1].w)
                        qq.push(v), dep[v] = dep[x] + 1;
                }
            }
            return dep[s] != infx;
        }
        std::priority_queue<int, std::vector<int>, cmp> q;
        inline void push(int x, int s, int t) {
            rep(i, x) {
                int v = e[i].v; ll w = e[i].w;
                if(w && dep[v] + 1 == dep[x]) {
                    ll fl = std::min(flow[x], e[i].w);
                    e[i].w -= fl, e[i ^ 1].w += fl;
                    flow[x] -= fl, flow[v] += fl;
                    if(v != s and v != t and !vis[v])
                        q.push(v), vis[v] = 1;
                    if(!flow[x]) break;
                }
            }
        }
        inline void relabel(int x) {
            dep[x] = infx;
            rep(i, x) {
                int v = e[i].v; ll w = e[i].w;
                if(w && dep[x] > dep[v] + 1)
                    dep[x] = dep[v] + 1;
            }
        }
        inline ll hlpp(int s, int t) {
            if(!bfs(s, t)) return 0;
            dep[s] = n;
            std::memset(gap, 0, sizeof gap);

            go(i, 1, n, 1) if(dep[i] < infx) ++gap[dep[i]];

            rep(i, s) {
                int v = e[i].v; ll w = e[i].w;
                if(w) {
                    e[i].w -= w, e[i ^ 1].w += w;
                    flow[s] -= w, flow[v] += w;
                    if(!vis[v] and v != s and v != t)
                        q.push(v), vis[v] = 1;
                }
            }
            while(!q.empty()) {
                int x = q.top(); q.pop(), vis[x] = 0;
                push(x, s, t);
                if(flow[x]) {
                    if(!--gap[dep[x]])
                        go(i, 1, n, 1)
                            if(i != s && i != t && dep[i] > dep[x] && dep[i] < n
 + 1)
```

```
                        dep[i] = n + 1;
                    relabel(x);
                    ++gap[dep[x]];
                    q.push(x), vis[x] = 1;
                }
            }
            return flow[t];
        }
    public:
        inline void init() {
            std::memset(h, -1, sizeof h);
            p = -1;
        }
        inline void get_maxf(int s, int t, ll &maxflow) { maxflow = hlpp(s, t);
}
        inline void add_flow(int a, int b, int c) { add(a, b, c), add(b, a, 0);
}
} hlpp;
```

2.最小费用最大流

（1）SPFA版

P3381 【模板】最小费用最大流——AC代码

注释部分为EK的代码（对比修改

```

int n, m;
struct edge{
    int v, nxt, w, d;
} e[mn << 1];
int h[mn], p = -1;
inline void add(int a, int b, int c, int d) {
    e[++p].nxt = h[a], h[a] = p, e[p].v = b, e[p].w = c, e[p].d = d;
}
inline void add_flow(int a, int b, int c, int d) { add(a, b, c, d), add(b, a, 0,
-d); }
struct node {
    int v, id;
} pre[mn];
bool vis[mn];
inline bool bfs(int s, int t) {
    memset(pre, -1, sizeof pre);
    memset(vis, 0, sizeof vis);
    vis[s] = 1;
    queue<int> q; q.push(s);
    while(!q.empty()) {
        int x = q.front(); q.pop();
        rep(i, x) {
            int v = e[i].v, w = e[i].w;
            if(!vis[v] && w){
                vis[v] = 1;
                pre[v].v = x, pre[v].id = i;
                if(v == t) return true;
                q.push(v);
            }
        }
    }
    return false;
}
int dis[mn], flow[mn];
inline bool SPFA(int s, int t) {
    memset(dis, 0x7f, sizeof dis);
    memset(flow, 0x7f, sizeof flow);
    memset(vis, 0, sizeof vis);
    memset(pre, -1, sizeof pre);
    queue<int> q; q.push(s);
    dis[s] = 0, vis[s] = 1;
    while(!q.empty()) {
        int x = q.front(); q.pop();
        vis[x] = 0;
        rep(i, x) {
            int v = e[i].v, w = e[i].w, d = e[i].d;
            if(w && dis[v] > dis[x] + d) {
                dis[v] = dis[x] + d;
                pre[v].v = x;
                pre[v].id = i;
                flow[v] = min(flow[x], w);
                if(!vis[v]) q.push(v), vis[v] = 1;
            }
        }
    }
    if(pre[t].v != -1) return true;
    return false;
}
inline int EK(int s, int t) {
    int d, maxflow = 0;
    while(bfs(s, t)) {
        d = inf;
        for(int i = t; i != s; i = pre[i].v)
            d = min(d, e[pre[i].id].w);
        for(int i = t; i != s; i = pre[i].v)
            e[pre[i].id].w -= d,
            e[pre[i].id ^ 1].w += d;
        maxflow += d;
    }
    return maxflow;
}
int maxflow, mincost;
inline void MCMF(int s, int t) {
    while(SPFA(s, t)) {
        maxflow += flow[t];
        mincost += flow[t] * dis[t];
        for(int i = t; i != s; i = pre[i].v) {
            e[pre[i].id].w -= flow[t];
            e[pre[i].id ^ 1].w += flow[t];
        }
    }
}

inline void solve() {
    n = read(), m = read();
    int s = read(), t = read(), x, y, z, w;
    go(i, 1, m, 1)
        x = read(), y = read(), z = read(), w = read(),
        add_flow(x, y, z, w);
//  cout << EK(s, t);
    MCMF(s, t);
    cout << maxflow << " " << mincost << "\n";
}
inline void init() {
    memset(h, -1, sizeof h);
    p = -1;
}

int main () {
    init();
    solve();
    return 0;
}
```

(2)Dijkstra版

by Rhein_E

```
// Luogu P1251
#include <cstdio>
#include <cstring>
#include <iostream>
#include <queue>
typedef long long LL;
typedef std::pair<LL, int> pli;
const int MAXN = 2010, MAXM = MAXN * 7;
const LL INF = 0x3f3f3f3f3f3f3f3fll;
struct Graph {
    struct Edge {
        int v, cap, cost, next;
        Edge(int _v = 0, int _c = 0, int _co = 0, int _n = 0) : v(_v), cap(_c),
cost(_co), next(_n) {}
    } E[MAXM << 1];
    int n, head[MAXN << 1], cnt, pre[MAXN << 1];
    LL h[MAXN << 1], dis[MAXN << 1];
    int s, t;
    void init(int _s, int _t, int _n) {
        s = _s, t = _t, n = _n, cnt = 0;
        memset(head, -1, sizeof(head));
    }
    void addEdge(int u, int v, int cap, int cost) {
        E[cnt] = Edge(v, cap, cost, head[u]);
        head[u] = cnt++;
        E[cnt] = Edge(u, 0, -cost, head[v]);
        head[v] = cnt++;
    }
    bool Dijkstra() {
        static std::priority_queue<pli, std::vector<pli>, std::greater<pli>> q;
        static bool vis[MAXN << 1];
        memset(vis, 0, sizeof(vis));
        memset(dis, 0x3f, sizeof(dis));
        memset(pre, -1, sizeof(pre));
        dis[s] = 0;
        q.push(std::make_pair(0ll, s));
        while (!q.empty()) {
            int u = q.top().second;
            q.pop();
            if (vis[u])
                continue;
            vis[u] = 1;
            for (int i = head[u]; ~i; i = E[i].next)
                if (E[i].cap && dis[E[i].v] > dis[u] + E[i].cost + h[u] -
h[E[i].v]) {
                    dis[E[i].v] = dis[u] + E[i].cost + h[u] - h[E[i].v];
                    pre[E[i].v] = i;
                    q.push(std::make_pair(dis[E[i].v], E[i].v));
                }
        }
        return (dis[t] ^ INF);
    }
    LL MCMF() {
        LL res = 0;
        while (Dijkstra()) {
            int max_flow = (int)INF;
            for (int i = 0; i < n; ++i)
                h[i] += dis[i];
            for (int p = pre[t]; ~p; p = pre[E[p ^ 1].v])
                max_flow = std::min(max_flow, E[p].cap);
            res += max_flow * h[t];
            for (int p = pre[t]; ~p; p = pre[E[p ^ 1].v])
                E[p].cap -= max_flow, E[p ^ 1].cap += max_flow;
        }
        return res;
    }
} G;
int main() {
#ifndef ONLINE_JUDGE
    freopen("in.txt", "r", stdin);
    freopen("out.txt", "w", stdout);
#endif
    int N, n, m, s, f, p;
    std::cin >> N;
    G.init(0, N << 1 | 1, (N << 1) + 2);
    for (int i = 1; i <= N; ++i) {
        int t;
        std::cin >> t;
        G.addEdge(0, i, t, 0);
        G.addEdge(i + N, N << 1 | 1, t, 0);
    }
    std::cin >> p >> m >> f >> n >> s;
    for (int i = 1; i <= N; ++i)
        G.addEdge(0, i + N, INF, p);
    for (int i = 1; i + m <= N; ++i)
        G.addEdge(i, i + m + N, INF, f);
    for (int i = 1; i + n <= N; ++i)
        G.addEdge(i, i + n + N, INF, s);
    for (int i = 1; i < N; ++i)
        G.addEdge(i, i + 1, INF, 0);

    std::cout << G.MCMF() << std::endl;

    return 0;
}
```

```
// Rhein_E
```

12.树链剖分

P3384 【模板】树链剖分——AC代码

```
int mod;
int n, m, r;
int b[mn];
struct segmenttree{
    int z[mn << 2], col[mn << 2];
    inline void update(int rt){
        z[rt] = (z[rt << 1] + z[rt << 1 | 1]) % mod;
    }
    inline int operation(int a,int b){
        return (a + b) % mod;
    }
    inline void color(int l,int r,int rt,int v){
        z[rt] += (r - l + 1) * v;
        col[rt] += v;
    }
    inline void push_col(int l,int r,int rt){
        if(col[rt]){
            int m = (l + r) >> 1;
            color(lson, col[rt]);
            color(rson, col[rt]);
            col[rt] = 0;
        }
    }
    inline void build(int l,int r,int rt){
        if(l==r){
            z[rt] = b[l] % mod;
            return;
        }
        int m = (l + r) >> 1;
        build(lson);
        build(rson);
        update(rt);
    }
    inline void modify(int l,int r,int rt,int nowl,int nowr,int v){
        if(nowl<=l && r<=nowr){
            color(bson, v);
            return;
        }
        int m = (l + r) >> 1;
        push_col(bson);
        if(nowl<=m)
            modify(lson, nowl, nowr, v);
        if(m<nowr)
            modify(rson, nowl, nowr, v);
        update(rt);
    }
    inline int query(int l,int r,int rt,int nowl,int nowr){
        if(nowl<=l && r<=nowr){
            return z[rt] % mod;
        }
        int m = (l + r) >> 1;
        push_col(bson);
        if(nowl<=m){
            if(m<nowr)
                return operation(query(lson, nowl, nowr), query(rson, nowl,
nowr));
            else
                return query(lson, nowl, nowr);
        }else{
            return query(rson, nowl, nowr);
        }
    }
} tr;
//Line Segment Tree ----------------------------------------
struct edge{
    int v,nxt;
} e[mn<<1];
int h[mn],p;
int w[mn];
inline void add(int a,int b){
    p++;
    e[p].nxt=h[a];
    h[a]=p;
    e[p].v=b;
}
//adjacency list ---------------------------------------------
int dep[mn], fa[mn], son[mn], id[mn], sze[mn], top[mn];
int cnt;
//arrs -------------------------------------------------------
void dfs1(int x,int f,int deep){
    dep[x] = deep;
    fa[x] = f;
    sze[x] = 1;
    int maxson = -1;
    rep(i,x){
        int v = e[i].v;
        if(v==f)
            continue;
        dfs1(v, x, deep + 1);
        sze[x] += sze[v];
        if(sze[v] > maxson)
            maxson = sze[v], son[x] = v;
    }
}
void dfs2(int x,int topf){
    id[x] = ++cnt;
    b[id[x]] = w[x];
    top[x] = topf;
    if(!son[x])
        return;
    dfs2(son[x], topf);
    rep(i,x){
        int v = e[i].v;
        if (v == son[x] || v == fa[x])
            continue;
        dfs2(v, v);
    }
}
//DFS -----------------------------------------------------
```

```
inline int query1(int x,int y){
    int ans = 0;
    while(top[x] != top[y]){
        if(dep[top[x]]<dep[top[y]])
            swap(x, y);
        ans += tr.query(root, id[top[x]], id[x]);
        ans %= mod;
        x = fa[top[x]];
    }
    if(dep[x]>dep[y])
        swap(x, y);
    ans += tr.query(root, id[x], id[y]);
    ans %= mod;
    return ans;
}
inline void modify1(int x,int y,int v){
    v %= mod;
    while(top[x] != top[y]){
        if(dep[top[x]]<dep[top[y]])
            swap(x, y);
        tr.modify(root, id[top[x]], id[x], v);
        x = fa[top[x]];
    }
    if(dep[x]>dep[y])
        swap(x, y);
    tr.modify(root, id[x], id[y], v);
}
inline int query2(int x){
    return tr.query(root, id[x], id[x] + sze[x] - 1);
}
inline void modify2(int x,int v){
    v %= mod;
    tr.modify(root, id[x], id[x] + sze[x] - 1, v);
}
//Change and Query ----------------------------------------
int main(){
    n = read(), m = read(), r = read(), mod = read();
    go(i,1,n,1){
        w[i] = read();
    }
    go(i,1,n-1,1){
        int x = read(), y = read();
        add(x, y), add(y, x);
    }
    dfs1(r, 0, 1);
    dfs2(r, r);
    tr.build(root);
    go(i,1,m,1){
        int s = read();
        if(s==1){
            int x = read(), y = read(), z = read();
            modify1(x, y, z);
        }else if(s==2){
            int x = read(), y = read();
            cout << query1(x, y) << "\n";
        }else if(s==3){
            int x = read(), z = read();
            modify2(x, z);
        }else if(s==4){
            int x = read();
            cout << query2(x) << "\n";
        }
    }
    return 0;
}
```

13.Prim

by Rhein_E

```
//Luogu P3366
#include <cstdio>
#include <iostream>
#include <cstring>
#include <queue>

typedef long long LL;
typedef std::pair<int, int> pii;
const int MAXN = 5005;
const int MAXM = 400005;

struct Graph {
    struct Edge {
        int v, next, len;
    } edge[MAXM];
    int n, m, cnt, head[MAXN];
    void init(int, int);
    void buildEdge(int, int, int);
    int Prim(int start = 1);
} G;

int main() {
    int N, M;
    while (~scanf("%d%d", &N, &M)) {
        G.init(N, M);
        while (M--) {
            int a, b, c;
            scanf("%d%d%d", &a, &b, &c);
            G.buildEdge(a, b, c);
            G.buildEdge(b, a, c);
        }
        int ans = G.Prim();
        if (~ans) printf("%d\n", ans);
        else puts("orz");
    }
    return 0;
}
void Graph::init(int _n = 0, int _m = 0) {
    cnt = 0;
    memset(head, -1, sizeof(head));
}
void Graph::buildEdge(int x, int y, int d) {
    edge[cnt].v = y, edge[cnt].len = d;
    edge[cnt].next = head[x];
    head[x] = cnt++;
}
int Graph::Prim(int start) {
    int res = 0;
    static std::priority_queue<pii, std::vector<pii>, std::greater<pii> > q;
    static bool vis[MAXN];
    vis[start] = 1;
    for (int i = head[start]; ~i; i = edge[i].next)
        q.push(std::make_pair(edge[i].len, edge[i].v));
    while (!q.empty()) {
        pii p = q.top();
        q.pop();
        if (vis[p.second]) continue;
        vis[p.second] = 1;
        res += p.first;
        for (int i = head[p.second]; ~i; i = edge[i].next)
            if (!vis[edge[i].v]) q.push(std::make_pair(edge[i].len, edge[i].v));
    }
    for (int i = 1; i <= n; ++i) if (!vis[i]) return -1;
    return res;
}
```

三.数据结构

1.堆

```
inline void swap(int &a, int &b) {
    int t = a; a = b; b = t;
}
int size, n, heap[mn];
inline void puth(int x) {
    int now, next;
    heap[++size] = x;
    now = size;
    while (now > 1) {
        next = now >> 1;
        if (heap[now] >= heap[next]) return;
        swap(heap[now], heap[next]);
        now = next;
    }
}
inline void geth() {
    cout << heap[1] << "\n";
    return;
}
inline int delh()
{
    int now, next, res;
    res = heap[1];
    heap[1] = heap[size--];
    now = 1;
    while (now * 2 <= size) {
        next = now * 2;
        if (next < size && heap[next + 1] < heap[next]) next++;
        if (heap[now] <= heap[next]) return res;
        swap(heap[now], heap[next]);
        now = next;
    }
}
```

2.线段树

```
ll z[mn << 2], col[mn << 2];
inline void update(int rt) {
    z[rt] = z[rt << 1] + z[rt << 1 | 1];
}
inline void color(int l, int r, int rt, ll v) {
    col[rt] += v;
    z[rt] += (r - l + 1) * v;
}
inline void push_col(int l, int r, int rt) {
    if(col[rt]) {
        int m = (l + r) >> 1;
        color(lson, col[rt]);
        color(rson, col[rt]);
        col[rt] = 0;
    }
}
inline void build(int l, int r, int rt) {
    if(l == r) { z[rt] = read(); return; }
    int m = (l + r) >> 1;
    build(lson), build(rson);
    update(rt);
}
inline void modify(int l, int r, int rt, int nowl, int nowr, ll v) {
    if(nowl <= l && r <= nowr) { color(bson, v); return; }
    int m = (l + r) >> 1;
    push_col(bson);
    if(nowl <= m) modify(lson, nowl, nowr, v);
    if(m < nowr) modify(rson, nowl, nowr, v);
    update(rt);
}
inline ll query(int l, int r, int rt, int nowl, int nowr) {
    if(nowl <= l && r <= nowr) return z[rt];
    int m = (l + r) >> 1;
    push_col(bson);
    if(nowl <= m)
        if(m < nowr)
            return query(lson, nowl, nowr) + query(rson, nowl, nowr);
        else return query(lson, nowl, nowr);
    else return query(rson, nowl, nowr);
}
```

万能模板（结构体）

```
struct tree{
    ll x;
};
struct SegmentTree{
    tree z[mn << 2];
    ll col[mn << 2];
    inline void update(int rt){
        z[rt].x = z[rt << 1].x + z[rt << 1 | 1].x;
    }
    inline tree operation(tree a,tree b){
        return (tree){a.x + b.x};
    }
    inline void color(int l,int r,int rt,ll v){
        z[rt].x += (r - l + 1) * v;
        col[rt] += v;
    }
    inline void push_col(int l,int r,int rt){
        if(col[rt]){
            int m = (l + r) >> 1;
            color(lson, col[rt]);color(rson, col[rt]);
            col[rt] = 0;
        }
    }
    inline void build(int l,int r,int rt){
        if(l==r){z[rt].x = read();return;}
        int m = (l + r) >> 1;build(lson);build(rson); update(rt);
    }
    inline void modify(int l,int r,int rt,int nowl,int nowr,ll v){
        if(nowl<=l && r<=nowr){color(bson, v); return;}
        int m = (l + r) >> 1; push_col(bson);
        if(nowl<=m) modify(lson, nowl, nowr, v);
        if(m<nowr)  modify(rson, nowl, nowr, v);
        update(rt);
    }
    inline tree query(int l,int r,int rt,int nowl,int nowr){
        if(nowl<=l && r<=nowr) return z[rt];
        int m = (l + r) >> 1; push_col(bson);
        if(nowl<=m){
            if(m<nowr) return operation(query(lson, nowl, nowr), query(rson,
nowl, nowr));
            else       return query(lson, nowl, nowr);
        }else          return query(rson, nowl, nowr);
    }
} tr;
```

3.(ex_线段树)zkw线段树

learn from dalao : GKxx

```
ll z[mn << 2];
int n, m, M;
inline void update(int rt) {
    z[rt] = z[rt << 1] + z[rt << 1 | 1];
}
inline void build() {
    for (M = 1; M < n; M <<= 1) ;
    for (int i = M + 1; i <= M + n; i++) z[i] = read();
    for (int i = M - 1; i; i--) update(i);
}
inline ll query(int l, int r) {
    ll ans = 0;
    for (--l += M, ++r += M; l ^ r ^ 1; l >>= 1, r >>= 1) {
        if (~l & 1) ans += z[l ^ 1];
        if (r & 1) ans += z[r ^ 1];
    }
    return ans;
}
inline void modify(int rt, ll v) {
    for (z[rt += M] += v, rt >>= 1; rt; rt >>= 1) update(rt);
}
```

ex：线段树优化Dijkstra

learn from dalao: Gkxx

```
int minn[mn << 2], pos[mn << 2], M, n, m;
inline void update(int rt) {
    minn[rt] = min(minn[rt << 1], minn[rt << 1 | 1]);
    pos[rt] = (minn[rt << 1] < minn[rt << 1 | 1]) ? pos[rt << 1] : pos[rt << 1 |
1];
}
inline void build() {
    for(M = 1; M < n + 2; M <<= 1) ;
    go(i, M, (M << 1) - 1, 1) minn[i] = inf, pos[i] = i - M;
    fo(i, M, 1, 1) update(i);
}
inline void modify(int rt, int v) {
    for(minn[rt += M] = v, rt >>= 1; rt; rt >>= 1) update(rt);
}
struct edge{ int v, nxt, w; } e[mm << 1]; int h[mn], p;
inline void add(int a, int b, int c) { e[++p].nxt = h[a], h[a] = p, e[p].v = b,
e[p].w = c; }
int dis[mn];
inline void Dij(int s) {
    go(i, 1, n, 1) dis[i] = inf;
    dis[s] = 0, build(), modify(s, 0);
    while(minn[1] < inf) {
        int x = pos[1], d = minn[1]; modify(x, inf);
        if(d != dis[x]) continue;
        rep(i, x) {
            int v = e[i].v, w = e[i].w;
            if(dis[v] > dis[x] + w)
                dis[v] = dis[x] + w, modify(v, dis[v]);
        }
    }
}
```

4.树状数组

```
ll y[mn];
int n, m;
inline ll lb(int x) { return x & -x; }
inline void modify(int p, ll v) {
    for (; p <= n; p += lb(p)) y[p] += v;
}
inline ll query(int p) {
    int ans = 0;
    for (; p; p -= lb(p)) ans += y[p];
    return ans;
}
```

5.LCA（最近公共祖先）

（1）倍增版

```
struct edge {
    int v, nxt;
} e[mn << 1];
int h[mn], p;
inline void add(int a, int b) {
    e[++p].nxt = h[a], h[a] = p, e[p].v = b;
}
int dep[mn], fa[mn][mk], n, m, s;
void dfs(int x, int f) {
    // printf(" -- %d -- \n", x);
    dep[x] = dep[f] + 1;
    fa[x][0] = f;
    for(int i = 1; (1 << i) <= dep[x]; ++i)
        fa[x][i] = fa[fa[x][i - 1]][i - 1];
    rep(i, x) {
        int v = e[i].v;
        if(v == f) continue;
        dfs(v, x);
    }
}
inline int LCA(int a, int b) {
    if(dep[a] > dep[b]) swap(a, b);
    fo(i, mk - 1, 0, 1)
        if(dep[a] <= dep[b] - (1 << i))
            b = fa[b][i];
    if(a == b) return a;
    fo(i, mk - 1, 0, 1) {
        if(fa[a][i] == fa[b][i]) continue;
        else a = fa[a][i], b = fa[b][i];
    }
    return fa[a][0];
}
```

（2）树链剖分版

```
struct edge{
    int v, nxt;
}e[mn << 1];
int h[mn], p;
inline void add(int a, int b) {
    e[++p].nxt = h[a], h[a] = p, e[p].v = b;
}
int dep[mn], top[mn], sze[mn], fa[mn], son[mn], n, m;
void dfs1(int x, int f, int deep) {
    dep[x] = deep;
    sze[x] = 1;
    fa[x] = f;
    int maxson = -1;
    rep(i, x) {
        int v = e[i].v;
        if(v == f) continue;
        dfs1(v, x, deep + 1);
        sze[x] += sze[v];
        if(maxson < sze[v]) maxson = sze[v], son[x] = v;
    }
}
void dfs2(int x, int topf) {
    top[x] = topf;
    if(!son[x]) return;
    dfs2(son[x], topf);
    rep(i, x) {
        int v = e[i].v;
        if(v == fa[x] || v == son[x]) continue;
        dfs2(v, v);
    }
}
inline int LCA(int x, int y) {
    while(top[x] != top[y]) {
        if(dep[top[x]] < dep[top[y]]) swap(x, y);
        x = fa[top[x]];
    }
    return dep[x] < dep[y] ? x : y;
}
int main() {
    n = read(), m = read();
    int rot = read();
    go(i, 1, n - 1, 1) {
        int a = read(), b = read();
        add(a, b), add(b, a);
    }
    dfs1(rot, 0, 1);
    dfs2(rot, rot);
    go(i, 1, m, 1) {
        int x = read(), y = read();
        printf("%d\n", LCA(x, y));
    }
    return 0;
}
```

6.平衡树

(1) Treap

```
int rt, tot;
struct tree{
    int size, ch[2], w, pos;
};
struct Treap{
    tree z[mn];
    inline void update(int rt){
        z[rt].size = z[z[rt].ch[0]].size + z[z[rt].ch[1]].size + 1;
    }
    inline void rotate(int &rt,int p){
        int t = z[rt].ch[p];
        z[rt].ch[p] = z[t].ch[p ^ 1], z[t].ch[p ^ 1] = rt;
        update(rt), update(t);
        rt = t;
    }
    inline void add(int x,int &rt){
        if(!rt){
            rt = ++tot, z[rt].size = 1, z[rt].w = x, z[rt].pos = rand();
            return;
        }
        z[rt].size++;
        int nxt = x < z[rt].w ? 0 : 1;
        add(x, z[rt].ch[nxt]);
        if(z[z[rt].ch[nxt]].pos<z[rt].pos)
            rotate(rt, nxt);
    }
    inline void del(int x,int &rt){
        if(z[rt].w==x){
            if(z[rt].ch[0]*z[rt].ch[1]==0){
                rt = z[rt].ch[0] + z[rt].ch[1];
                return;
            }
            if(z[z[rt].ch[0]].pos>z[z[rt].ch[1]].pos){
                rotate(rt, 1);
                del(x, z[rt].ch[0]);
            }else{
                rotate(rt, 0);
                del(x, z[rt].ch[1]);
            }
        }else{
            int nxt = x < z[rt].w ? 0 : 1;
            del(x, z[rt].ch[nxt]);
        }
        update(rt);
    }
    inline int find(int x,int rt){
        if(!rt)
            return 1;
        if(z[rt].w>=x)
            return find(x, z[rt].ch[0]);
        else
            return find(x, z[rt].ch[1]) + z[z[rt].ch[0]].size + 1;
    }
    inline int ask(int x,int rt){
        if(z[z[rt].ch[0]].size==x-1)
            return z[rt].w;
        if(z[z[rt].ch[0]].size>=x)
            return ask(x, z[rt].ch[0]);
        else
            return ask(x - z[z[rt].ch[0]].size - 1, z[rt].ch[1]);
    }
    inline int pre(int x,int rt){
        if(!rt)
            return -inf;
        if(z[rt].w<x)
            return max(z[rt].w, pre(x, z[rt].ch[1]));
        return pre(x, z[rt].ch[0]);
    }
    inline int nxt(int x,int rt){
        if(!rt)
            return inf;
        if(z[rt].w>x)
            return min(z[rt].w, nxt(x, z[rt].ch[0]));
        return nxt(x, z[rt].ch[1]);
    }
} tr;
```

(2) Splay

Form 1:
by yizimi

```
int rot, tot, n;
struct tree{
    int ch[2], fa, cnt, w, size;
};
struct Splay{
    tree z[mn];
    void update(int rt){
        z[rt].size = z[z[rt].ch[0]].size + z[z[rt].ch[1]].size + z[rt].cnt;
    }
    void rotate(int a){
        int b = z[a].fa;
        int c = z[b].fa;
        int k = z[b].ch[1] == a;
        z[c].ch[z[c].ch[1] == b] = a;
        z[a].fa = c;
        z[b].ch[k] = z[a].ch[k ^ 1];
        z[z[a].ch[k ^ 1]].fa = b;
        z[a].ch[k ^ 1] = b;
        z[b].fa = a;
        update(b), update(a);
    }
    void splay(int a,int goal){
        while(z[a].fa!=goal){
            int b = z[a].fa;
            int c = z[b].fa;
            if(c!=goal)
                (z[b].ch[0] == a) ^ (z[c].ch[0] == b) ? rotate(a) : rotate(b);
            rotate(a);
        }
        if (goal == 0)
            rot = a;
    }
    void insert(int x){
        int fa = 0, rt = rot;
        while(rt && z[rt].w!=x){
            fa = rt;
            int nxt = x < z[rt].w ? 0 : 1;
            rt = z[rt].ch[nxt];
        }
        if(rt)
            z[rt].cnt++;
        else{
            rt = ++tot;
            int nxt = x < z[fa].w ? 0 : 1;
            if(fa)
                z[fa].ch[nxt] = rt;
            z[tot].ch[0] = 0, z[tot].ch[1] = 0, z[tot].fa = fa;
            z[tot].w = x, z[tot].size = 1, z[tot].cnt = 1;
        }
        splay(rt, 0);
    }
    void find(int x){
        int rt = rot;
        if(!rt)
            return;
        //int nxt = x < z[rt].w ? 0 : 1;
        //while(z[rt].ch[nxt] && x!=z[rt].w)
            //nxt = x < z[rt].w ? 0 : 1, rt = z[rt].ch[nxt];
        while (z[rt].ch[x > z[rt].w] && x != z[rt].w)
            rt = z[rt].ch[x > z[rt].w];
        splay(rt, 0);
    }
    int nxt(int x,int f){
        find(x);
        int rt = rot;
        if((z[rt].w>x && f) || (z[rt].w<x && !f))
            return rt;
        rt = z[rt].ch[f];
        while(z[rt].ch[f^1])
            rt = z[rt].ch[f ^ 1];
        return rt;
    }
    void del(int x){
        int last = nxt(x, 0);
        int next = nxt(x, 1);
        splay(last, 0);
        splay(next, last);
        int t = z[next].ch[0];
        if(z[t].cnt>1){
            z[t].cnt--;
            splay(t, 0);
        }else{
            z[next].ch[0] = 0;
        }
    }
    int ask(int x){
        int rt = rot;
        if(z[rt].size<x)
            return 0;
        while(1119){
            int b = z[rt].ch[0];
            if(x>z[b].size+z[rt].cnt){
                x -= z[b].size + z[rt].cnt;
                rt = z[rt].ch[1];
            }else if(z[b].size>=x)
                rt = b;
            else
                return z[rt].w;
        }
    }
    int findx(int x){
        find(x);
        return z[z[rot].ch[0]].size;
    }
    int query(int x,int f){
        return z[nxt(x, f)].w;
    }
```

```
} tr;
int main(){
    tr.insert(-2147483647);
    tr.insert(+2147483647);
    n = read();
    go(i,1,n,1){
        int s = read(), x = read();
        int ans;
        if(s == 1) tr.insert(x);
        if(s == 2) tr.del(x);
        if(s == 3) ans = tr.findx(x);
        if(s == 4) ans = tr.ask(x+1);
        if(s == 5) ans = tr.query(x, 0);
        if(s == 6) ans = tr.query(x, 1);
        if(s > 2) cout << ans << "\n";
    }
    return 0;
}
```

Form 2:(pointer)
by Rhein_E

```
// POJ Buy Tickets TLE
#include <cstdio>
#include <iostream>
#include <cstring>

struct Splay {
    struct Node {
        int val, size;
        Node *child[2], *fa;
        Node() { memset(this, 0, sizeof(Node)); }
    } * root;
    inline int getSize(Node *p) { return (p ? p->size : 0); }
    void update(Node *rt) {
        rt->size = 1;
        rt->size += getSize(rt->child[0]);
        rt->size += getSize(rt->child[1]);
    }
    void rotate(Node *rt) {
        int d = (rt->fa->child[1] == rt);
        Node *f = rt->fa;
        f->child[d] = rt->child[d ^ 1];
        if (rt->child[d ^ 1])
            rt->child[d ^ 1]->fa = f;
        rt->child[d ^ 1] = f;
        if (f->fa)
            f->fa->child[f->fa->child[1] == f] = rt;
        rt->fa = f->fa;
        f->fa = rt;

        update(f);
        update(rt);
    }
    void splay(Node *rt) {
        while (rt->fa) {
            if (!rt->fa->fa)
                rotate(rt);
            else if ((rt->fa->child[1] == rt) == (rt->fa->fa->child[1] == rt-
>fa)) {
                rotate(rt->fa);
                rotate(rt);
            } else {
                rotate(rt);
                rotate(rt);
            }
        }
        root = rt;
    }
    Node *query(int pos) {
        Node *p = root;
        if (pos > getSize(root))
            return 0;
        while (1) {
            if (getSize(p->child[0]) + 1 == pos) {
                splay(p);
                return p;
            } else if (getSize(p->child[0]) >= pos)
                p = p->child[0];
            else
                pos -= getSize(p->child[0]) + 1, p = p->child[1];
        }
    }
    Node *insert(int pos, int val) {
        if (!root) {
            root = new Node;
            root->size = 1, root->val = val;
        } else {
            Node *p = query(pos), *np = new Node;
            np->val = val, np->size = 1;
            if (!p->child[1])
                p->child[1] = np, np->fa = p;
            else {
                p = p->child[1];
                while (p->child[0])
                    p = p->child[0];
                p->child[0] = np, np->fa = p;
            }
            splay(np);
        }
        return root;
    }
    void clear() {
        clear(root);
        root = NULL;
    }
    void clear(Node *p) {
        if (p->child[0])
            clear(p->child[0]);
        if (p->child[1])
            clear(p->child[1]);
        delete p;
    }
} tree;
```

```
int N;
int main() {
#ifndef ONLINE_JUDGE
    freopen("C:\\Users\\Rhein_E\\Desktop\\code\\cpp\\templates\\in.txt", "r",
stdin);
    freopen("C:\\Users\\Rhein_E\\Desktop\\code\\cpp\\templates\\out.txt", "w",
stdout);
#endif
    while (~scanf("%d", &N)) {
        tree.insert(0, 0);
        for (int i = 0; i < N; ++i) {
            int a, b;
            scanf("%d%d", &a, &b);
            tree.insert(a + 1, b);
        }
        for (int i = 1; i <= N; ++i)
            printf("%d ", tree.query(i + 1)->val);
        puts("");
        tree.clear();
    }
    return 0;
}
```

文艺平衡树（Splay）

```
int n, m, rot, tot = 0, a[mn];
struct tree{
    int ch[2], w, sze, fa, col;
    tree(int _w = 0, int _sze = 0, int _fa = 0, int _col = 0)
    : w(_w), sze(_sze), fa(_fa), col(_col) { ch[0] = ch[1] = 0; }
};
struct Splay{
    tree z[mn];
    inline void update(int rt){
        z[rt].sze = z[z[rt].ch[0]].sze + z[z[rt].ch[1]].sze + 1;
    }
    inline void change(int x){ //jiao huan zuo you zi shu
        swap(z[x].ch[0], z[x].ch[1]);
    }
    inline void push_col(int rt){
        if(z[rt].col){
            z[z[rt].ch[0]].col ^= 1;
            z[z[rt].ch[1]].col ^= 1;
            z[rt].col = 0;
            swap(z[rt].ch[0], z[rt].ch[1]);
        }
    }
    inline int iden(int rt){
        return z[z[rt].fa].ch[0] == rt ? 0 : 1;
    }
    inline void connect(int x,int y,int son){
        z[x].fa = y;
        z[y].ch[son] = x;
    }
    inline void rotate(int x){
        int y = z[x].fa;
        int moot = z[y].fa;
```

```cpp
        int yson = iden(x);
        int mootson = iden(y);
        int B = z[x].ch[yson ^ 1];
        connect(B, y, yson), connect(y, x, yson ^ 1), connect(x, moot, mootson);
        update(y), update(x);
    }
    inline void splay(int x,int &k){
        if(x==k)
            return;
        int p = z[k].fa;
        while(z[x].fa != p){
            push_col(x);      //warning
            int y = z[x].fa;
            if(z[y].fa != p)  //warning
                rotate(iden(x) ^ iden(y) ? x : y);
            rotate(x);
        }
        k = x;
    }
    inline int findkth(int rt,int k){
        while(1119){
            push_col(rt);
            if(z[rt].ch[0] && k <= z[z[rt].ch[0]].sze){
                rt = z[rt].ch[0]/*,puts("okok")*/;
            }else {
                if(z[rt].ch[0])
                    k -= z[z[rt].ch[0]].sze;
                if(!--k)
                    return rt;
                rt = z[rt].ch[1];
            }
        }
    }
    inline int getRange(int l,int r,int &rt){
        int x = findkth(rt, l);
        //puts("getxok");
        splay(x, rt);
        //cout << rot << "\n";
        int y = findkth(rt, r + 2);
        int ooo = z[rt].ch[1];
        splay(y, ooo);
        return z[y].ch[0];
    }
    inline void modify(int &rt,int nowl,int nowr){
        int x = getRange(nowl, nowr, rt);
        z[x].col ^= 1;
        update(z[rt].ch[1]), update(rt);
    }
    inline void build(int l,int r,int rt){
        int m = (l + r) >> 1;
        z[rt].w = a[m];
        if(l <= m - 1) {
            z[z[rt].ch[0] = ++tot].fa = rt;
            build(l, m - 1, z[rt].ch[0]);
        }
        if(m + 1 <= r) {
            z[z[rt].ch[1] = ++tot].fa = rt;
            build(m + 1, r, z[rt].ch[1]);
        }
        update(rt);
    }
    inline void dfs(int rt){
        if(rt == 0)
            return;
        push_col(rt);
        //puts("push_colok");
        dfs(z[rt].ch[0]);
        if(z[rt].w > 0)
            printf("%d ", z[rt].w);
        dfs(z[rt].ch[1]);
    }
} tr;
int main(){
    n = read(), m = read();
    go(i, 1, n, 1) a[i] = i;
    rot = ++tot;
    tr.build(0, n + 1, rot);
    go(i,1,m,1){
        int x = read(), y = read();
        tr.modify(rot, x, y);
    }
    tr.dfs(rot);
    return 0;
}
```

(3) FHQ Treap

```
struct tree {
    int ch[2], w, pri, sze;
} z[mn];
int cnt;
inline void update(int rt) {
    z[rt].sze = 1;
    if(z[rt].ch[0])
        z[rt].sze += z[z[rt].ch[0]].sze;
    if(z[rt].ch[1])
        z[rt].sze += z[z[rt].ch[1]].sze;
}
inline int newnode(int w = 0) {
    z[++cnt].sze = 1;
    z[cnt].w = w;
    z[cnt].pri = rand();
    return cnt;
}
inline int merge(int x, int y) {
    if(!x || !y) return x + y;
    if(z[x].pri < z[y].pri) {
        z[x].ch[1] = merge(z[x].ch[1], y);
        update(x);
        return x;
    } else {
        z[y].ch[0] = merge(x, z[y].ch[0]);
        update(y);
        return y;
    }
}
inline void split(int rt, int k, int &x, int &y) {
    if(!rt) x = y = 0;
    else  {
        if(z[rt].w <= k) {
            x = rt, split(z[rt].ch[1], k, z[rt].ch[1], y);
        } else {
            y = rt, split(z[rt].ch[0], k, x, z[rt].ch[0]);
        }
        update(rt);
    }
}
inline int findkth(int rt, int k) {
    while(1119) {
        if(k <= z[z[rt].ch[0]].sze)
            rt = z[rt].ch[0];
        else if(k == z[z[rt].ch[0]].sze + 1){
            return rt;
        } else {
            k -= z[z[rt].ch[0]].sze + 1, rt = z[rt].ch[1];
        }
    }
}
int n, rot, x, y;
inline void debug() {
    go(i, 1, cnt, 1) {
           printf("%d:%d %d %d\n", i, z[i].pri, z[i].sze, z[i].w);
    }
}
int main() {
    srand((unsigned)time(NULL));
    n = read();
    int zz;
    go(i, 1, n, 1) {
        int s = read(), a = read();
        if(s == 1) {
            split(rot, a, x, y);
            rot = merge(merge(x, newnode(a)), y);
        }
        if(s == 2) {
            split(rot, a, x, zz);
            split(x, a - 1, x, y);
            y = merge(z[y].ch[0], z[y].ch[1]);
            rot = merge(merge(x, y), zz);
        }
        if(s == 3) {
            split(rot, a - 1, x, y);
            printf("%d\n", z[x].sze + 1);
            rot = merge(x, y);
        }
        if(s == 4) {
            printf("%d\n", z[findkth(rot, a)].w);
        }
        if(s == 5) {
            split(rot, a - 1, x, y);
            printf("%d\n", z[findkth(x, z[x].sze)].w);
            rot = merge(x, y);
        }
        if(s == 6) {
            split(rot, a, x, y);
            printf("%d\n", z[findkth(y, 1)].w);
            rot = merge(x, y);
        }
//          cout << x << " " << y << " " << zz << " " << rot << "\n";
//          debug();
    }
    return 0;
}
```

维护区间操作

```
struct tree {
    int sze, ch[2], pri, w;
    ll x, sum, col;
} z[mn];
inline void update(int rt) {
    z[rt].sze = 1, z[rt].sum = z[rt].x;
    if(z[rt].ch[0])
        z[rt].sze += z[z[rt].ch[0]].sze, z[rt].sum += z[z[rt].ch[0]].sum;
    if(z[rt].ch[1])
        z[rt].sze += z[z[rt].ch[1]].sze, z[rt].sum += z[z[rt].ch[1]].sum;
}
inline void push_col(int rt) {
    if(z[rt].col) {
        z[z[rt].ch[0]].x += z[rt].col;
        z[z[rt].ch[1]].x += z[rt].col;
        z[z[rt].ch[0]].col += z[rt].col;
        z[z[rt].ch[1]].col += z[rt].col;
        z[z[rt].ch[0]].sum += z[rt].col * z[z[rt].ch[0]].sze;
        z[z[rt].ch[1]].sum += z[rt].col * z[z[rt].ch[1]].sze;
        z[rt].col = 0;
    }
}
int cnt;
inline int newnode(int w, int v = 0) {
    z[++cnt].x = v;
    z[cnt].w = w;
    z[cnt].sze = 1;
    z[cnt].sum = v;
    z[cnt].pri = rand();
    return cnt;
}
inline int merge(int x, int y) {
    if(!x || !y) return x + y;
    if(z[x].pri < z[y].pri) {
        push_col(x);
        z[x].ch[1] = merge(z[x].ch[1], y);
        update(x);
        return x;
    } else {
        push_col(y);
        z[y].ch[0] = merge(x, z[y].ch[0]);
        update(y);
        return y;
    }
}
inline void split(int rt, int k, int &x, int &y) {
    if(!rt) x = y = 0;
    else {
        push_col(rt);
        if(z[rt].w <= k) {
            x = rt, split(z[rt].ch[1], k, z[rt].ch[1], y);
        } else {
            y = rt, split(z[rt].ch[0], k, x, z[rt].ch[0]);
        }
        update(rt);
    }
}
int n, m;
int xx, yy, rot, zz;
int main(){
//  fre();
    n = read(), m = read();
    go(i, 1, n, 1) {
        int a = read();
        split(rot, i, xx, yy);
        rot = merge(merge(xx, newnode(i, a)), yy);
    }
    go(i, 1, m, 1) {
        int s = read(), x = read(), y = read();
        if(s == 1) {
            ll v = read();
            split(rot, y, xx, zz);
            split(xx, x - 1, xx, yy);
            z[yy].col += v;
            z[yy].x += v;
            z[yy].sum += z[yy].sze * v;
            rot = merge(merge(xx, yy), zz);
        } else {
            split(rot, y, xx, zz);
            split(xx, x - 1, xx, yy);
            printf("%lld\n", z[yy].sum);
            rot = merge(merge(xx, yy), zz);
        }
    }
    return 0;
}
```

(4) WBLT

learn from dalao: **codesonic**

```
int n, cnt, fa, rot;
int sze[mn << 2], ch[mn << 2][2], val[mn << 2];
inline void newnode(int &rt, int v) {
    rt = ++cnt;
    sze[rt] = 1, val[rt] = v;
}
inline void copynode(int x, int y) {
    sze[x] = sze[y], ch[x][0] = ch[y][0],
    ch[x][1] = ch[y][1], val[x] = val[y];
}
inline void merge(int l, int r) {
    sze[++cnt] = sze[l] + sze[r];
    val[cnt] = val[r];
    ch[cnt][0] = l, ch[cnt][1] = r;
}
inline void rotate(int rt, bool flag) {
    if(flag) {
        merge(ch[rt][0], ch[ch[rt][1]][0]);
        ch[rt][0] = cnt, ch[rt][1] = ch[ch[rt][1]][1];
    } else {
        merge(ch[ch[rt][0]][1], ch[rt][1]);
        ch[rt][1] = cnt, ch[rt][0] = ch[ch[rt][0]][0];
    }
}
inline void maintain(int rt) {
    if(sze[ch[rt][0]] > sze[ch[rt][1]] * ratio)
        rotate(rt, 0);
    else if(sze[ch[rt][1]] > sze[ch[rt][0]] * ratio)
        rotate(rt, 1);
}
inline void update(int rt) {
    if(!sze[ch[rt][0]]) return;
    sze[rt] = sze[ch[rt][0]] + sze[ch[rt][1]];
    val[rt] = val[ch[rt][1]];
}
inline void insert(int rt, int x) {
    if(sze[rt] == 1) {
        newnode(ch[rt][0], min(x, val[rt]));
        newnode(ch[rt][1], max(x, val[rt]));
        update(rt);
        return;
    }
    maintain(rt);
    insert(x > val[ch[rt][0]] ? ch[rt][1] : ch[rt][0], x);
    update(rt);
}
inline void erase(int rt, int x) {
    if(sze[rt] == 1) {
        rt = ch[fa][0] == rt ? ch[fa][1] : ch[fa][0];
        copynode(fa, rt);
        return;
    }
    maintain(rt);
    fa = rt;
    erase(x > val[ch[rt][0]] ? ch[rt][1] : ch[rt][0], x);
    update(rt);
}
inline int find(int rt, int x) {
    if(sze[rt] == x) return val[rt];
    maintain(rt);
    if(x > sze[ch[rt][0]])
        return find(ch[rt][1], x - sze[ch[rt][0]]);
    return find(ch[rt][0], x);
}
inline int rnk(int rt, int x) {
    if(sze[rt] == 1) return 1;
    maintain(rt);
    if(x > val[ch[rt][0]])
        return rnk(ch[rt][1], x) + sze[ch[rt][0]];
    return rnk(ch[rt][0], x);
}
inline void solve() {
    n = read();
    newnode(rot, (1 << 30));
    go(i, 1, n, 1) {
        int s = read(), x = read();
        if(s == 1) insert(rot, x);
        if(s == 2) erase(rot, x);
        if(s == 3) printf("%d\n", rnk(rot, x));
        if(s == 4) printf("%d\n", find(rot, x));
        if(s == 5) printf("%d\n", find(rot, rnk(rot, x) - 1));
        if(s == 6) printf("%d\n", find(rot, rnk(rot, x + 1)));
    }
}
int main(){
    solve();
    return 0;
}
```

7.权值线段树

```
ll z[mn << 2];
inline void update(int rt) {
    z[rt] = z[rt << 1] + z[rt << 1 | 1];
}
inline void build(int l, int r, int rt) {
    if (l == r) { z[rt] = 0; return; }
    int m = (l + r) >> 1;
    build(lson); build(rson); update(rt);
}
inline void modify(int l, int r, int rt, ll x) {
    if (l == r) { z[rt]++; return; }
    int m = (l + r) >> 1;
    if (x <= m) modify(lson, x);
    else modify(rson, x);
    update(rt);
}
inline ll query(int l, int r, int rt, int nowl, int nowr) {
    if (nowl <= l && r <= nowr) { return z[rt]; }
    int m = (l + r) >> 1;
    if (nowl <= m) {
        if (m < nowr) return query(lson, nowl, nowr) + query(rson, nowl, nowr);
        else return query(lson, nowl, nowr);
    } else {
        return query(rson, nowl, nowr);
    }
}
int a[mn], b[mn], n, m;
int main() {
    n = read();
    go(i, 1, n, 1) a[i] = b[i] = read();
    sort(b + 1, b + n + 1);
    int size = unique(b + 1, b + n + 1) - b - 1;
    go(i, 1, n, 1) a[i] = lower_bound(b + 1, b + n + 1, a[i]) - b;
    build(1, 500000, 1);
    ll ans = 0;
    go(i, 1, n, 1) {
        ans += query(root, a[i] + 1, 500000);
        modify(root, a[i]);
    }
    cout << ans << "\n";
    return 0;
}
```

8.主席树（可持久化（权值）线段树）

```
int n, q, m, cnt = 0;
int a[mn], b[mn];
int rot[mn];
struct node{
    int l, r, sum;
    explicit node(int _l = 0, int _r = 0, int _sum = 0)
        : l(_l), r(_r), sum(_sum) {}
} z[mn << 5];
inline int build(int l,int r){
    int rt = ++cnt;
    z[rt].sum = 0;
    int m = (l + r) >> 1;
    if (l < r)
        z[rt].l = build(l, m),
        z[rt].r = build(m + 1, r);
    return rt;
}
inline int modify(int l,int r,int pre,int x){
    int rt = ++cnt;
    z[rt].l = z[pre].l, z[rt].r = z[pre].r, z[rt].sum = z[pre].sum + 1;
    int m = (l + r) >> 1;
    if (l < r) {
        if (x <= m)
            z[rt].l = modify(l, m, z[pre].l, x);
        else
            z[rt].r = modify(m + 1, r, z[pre].r, x);
    }
    return rt;
}
inline int query(int l,int r,int k,int nowl,int nowr){
    if(l>=r) return l;
    int x = z[z[nowr].l].sum - z[z[nowl].l].sum;
    int m = (l + r) >> 1;
    if(x >= k) return query(l, m, k, z[nowl].l, z[nowr].l);
    else       return query(m + 1, r, k - x, z[nowl].r, z[nowr].r);
}
int main(){
    n = read(), q = read();
    go(i, 1, n, 1) a[i] = b[i] = read();
    sort(b + 1, b + n + 1);
    m = unique(b + 1, b + n + 1) - b - 1;
    rot[0] = build(1, m);
    go(i, 1, n, 1)
```

```
        rot[i] = modify(1, m, rot[i - 1], lower_bound(b + 1, b + m + 1, a[i]) -
b);
    go(i, 1, q, 1) {
        int x = read(), y = read(), z = read();
        cout << b[query(1, m, z, rot[x - 1], rot[y])] << "\n";
    }
    return 0;
}
```

9.可持久化数组（可持久化线段树）

```
struct tree{
    int l, r, w;
};
int a[mn], rot[mn];
struct PersistableSegmentTree{
    tree z[mn << 5];
    int cnt = 0;
    inline void build(int l,int r,int &rt){
        rt = ++cnt;
        if(l==r){
            z[rt].w = a[l];
            return;
        }
        int m = (l + r) >> 1;
        build(l, m, z[rt].l);
        build(m + 1, r, z[rt].r);
    }
    inline void modify(int l,int r,int &rt,int pre,int now,int v){
        rt = ++cnt;
        z[rt].l = z[pre].l, z[rt].r = z[pre].r, z[rt].w = z[pre].w;
        if(l==r){
            z[rt].w = v;
            return;
        }
        int m = (l + r) >> 1;
        if(now<=m)
            modify(l, m, z[rt].l, z[pre].l, now, v);
        else
            modify(m + 1, r, z[rt].r, z[pre].r, now, v);
    }
    inline int query(int l,int r,int rt,int now){
        if(l==r)
            return z[rt].w;
        int m = (l + r) >> 1;
        if(now<=m)
            return query(l, m, z[rt].l, now);
        else
            return query(m + 1, r, z[rt].r, now);
    }
} P_tr;
int n, m;
int main(){
    n = read(), m = read();
    go(i, 1, n, 1) a[i] = read();
    P_tr.build(1, n, rot[0]);
    go(i, 1, m, 1){
        int pre = read(), s = read(), x = read();
        if(s==1){
            int v = read();
            P_tr.modify(1, n, rot[i], rot[pre], x, v);
        }else{
            cout << P_tr.query(1, n, rot[pre], x) << "\n";
            rot[i] = rot[pre];
        }
    }
    return 0;
}
```

10.二维树状数组

（1）单点修改，区间求和

```
ll z[mn][mn], n, m, q;
inline int lb(int x) { return x & -x; }
inline void modify(int x, int y, ll v) {
    for(int i = x; i <= n; i += lb(i))
        for(int j = y; j <= m; j += lb(j))
            z[i][j] += v;
}
inline ll query(int x, int y) {
    ll ans = 0;
    for(int i = x; i; i -= lb(i))
        for(int j = y; j; j -= lb(j))
            ans += z[i][j];
    return ans;
}
int main() {
    n = read(), m = read(), q = read();
    go(i, 1, n, 1) go(j, 1, m, 1){
        ll x = read(); modify(i, j, x);
    }
    go(i, 1, q, 1) {
        int s = read(), x = read(), y = read();
        if(s == 1) {
            ll v = read();
            modify(x, y, v);
        } else if(s == 2){
            int xx = read(), yy = read();
            printf("%lld\n", query(xx, yy) - query(x - 1, yy) - query(xx, y - 1)
+ query(x - 1, y - 1));
        }
    }
    return 0;
}
```

11.扫描线

POJ 1151 Atlantis——AC代码

```
struct tree{
    int mark; double sum;
} z[mn << 2];
struct seg{
    double l, r, h;
    int d;
    seg() {}
    seg(double _l, double _r, double _h, int _d) : l(_l), r(_r), h(_h), d(_d) {}
    bool operator < (const seg &b) const { return h < b.h; }
} s[mn];
int n, num, kkk;
double ha[mn];
double x, y, xx, yy;
#define root 0, m - 1, 1
#define lson l, m, rt << 1
#define rson m + 1, r, rt << 1 | 1
#define bson l, r, rt
inline void update(int l, int r, int rt) {
    if(z[rt].mark) z[rt].sum = ha[r + 1] - ha[l];
    else if(l == r) z[rt].sum = 0;
    else z[rt].sum = z[rt << 1].sum + z[rt << 1 | 1].sum;
}
inline void modify(int l, int r, int rt, int nowl, int nowr, int d) {
    if(nowl <= l && r <= nowr) {
        z[rt].mark += d;
        update(bson);
        return;
    }
    int m = (l + r) >> 1;
    if(nowl <= m) modify(lson, nowl, nowr, d);
    if(m < nowr)  modify(rson, nowl, nowr, d);
    update(bson);
}
inline int search(double key, double* x, int n) {
    int l = 0, r = n - 1;
    while(l <= r) {
        int m = (l + r) >> 1;
        if(x[m] == key) return m;
        if(x[m] > key) r = m - 1;
        else l = m + 1;
    }
    return -1;
}
int main() {
    while(cin >> n, n) {
        num = 0;
        go(i, 0, n - 1, 1) {
            cin >> x >> y >> xx >> yy;
            ha[num] = x;
            s[num++] = seg(x, xx, y, 1);
            ha[num] = xx;
            s[num++] = seg(x, xx, yy, -1);
        }
        sort(ha, ha + num);
        sort(s, s + num);
        int m = 1;
        go(i, 1, num - 1, 1)
            if(ha[i] != ha[i - 1]) ha[m++] = ha[i];
        double ans = 0;
        go(i, 0, num - 1, 1) {
            int L = search(s[i].l, ha, m);
            int R = search(s[i].r, ha, m) - 1;
            modify(root, L, R, s[i].d);
            ans += z[1].sum * (s[i + 1].h - s[i].h);
        }
        printf("Test case #%d\nTotal explored area: %.2lf\n", ++kkk, ans);
    }
    return 0;
}
```

12.可持久化平衡树

```
struct edge{
    int ch[2], sze, pri;
    ll w;
} z[mn * 50];
int rot[mn], xx, yy, zz, n, cnt;
inline void update(int rt) {
    z[rt].sze = 1;
    if(z[rt].ch[0]) z[rt].sze += z[z[rt].ch[0]].sze;
    if(z[rt].ch[1]) z[rt].sze += z[z[rt].ch[1]].sze;
}
inline int newnode(ll w = 0) {
    z[++cnt].w = w;
    z[cnt].sze = 1;
    z[cnt].pri = rand();
    return cnt;
}
inline int merge(int x, int y) {
    if(!x || !y) return x + y;
    if(z[x].pri < z[y].pri) {
        int rt = newnode();
        z[rt] = z[x];
        z[rt].ch[1] = merge(z[rt].ch[1], y);
        update(rt);
        return rt;
    } else {
        int rt = newnode();
        z[rt] = z[y];
        z[rt].ch[0] = merge(x, z[rt].ch[0]);
        update(rt);
        return rt;
    }
}
inline void split(int rt, ll k, int &x, int &y) {
    if(!rt) x = y = 0;
    else {
        if(z[rt].w <= k) {
            x = newnode();
            z[x] = z[rt];
            split(z[x].ch[1], k, z[x].ch[1], y);
            update(x);
        } else {
            y = newnode();
            z[y] = z[rt];
            split(z[y].ch[0], k, x, z[y].ch[0]);
            update(y);
        }
    }
}
inline int findkth(int rt, int k) {
    while(1119) {
        if(k <= z[z[rt].ch[0]].sze)
            rt = z[rt].ch[0];
        else {
            if(z[rt].ch[0]) k -= z[z[rt].ch[0]].sze;
            if(!--k) return rt;
            rt = z[rt].ch[1];
        }
    }
}
int main(){
    n = read();
    go(i, 1, n, 1) {
        xx = yy = zz = 0;
        int tmp = read(), s = read(); ll a = read();
        rot[i] = rot[tmp];
        if(s == 1) {
            split(rot[i], a, xx, yy);
            rot[i] = merge(merge(xx, newnode(a)), yy);
        } else if(s == 2) {
            split(rot[i], a, xx, zz);
            split(xx, a - 1, xx, yy);
            yy = merge(z[yy].ch[0], z[yy].ch[1]);
            rot[i] = merge(merge(xx, yy), zz);
        } else if(s == 3) {
            split(rot[i], a - 1, xx, yy);
            printf("%lld\n", z[xx].sze + 1);
            rot[i] = merge(xx, yy);
        } else if(s == 4) {
            printf("%lld\n", z[findkth(rot[i], a)].w);
        } else if(s == 5) {
            split(rot[i], a - 1, xx, yy);
            if(xx == 0) {
                printf("-2147483647\n");
                continue;
            }
            printf("%lld\n", z[findkth(xx, z[xx].sze)].w);
            rot[i] = merge(xx, yy);
        } else if(s == 6) {
            split(rot[i], a, xx, yy);
            if(yy == 0) {
                printf("2147483647\n");
                continue;
            }
            printf("%lld\n", z[findkth(yy, 1)].w);
            rot[i] = merge(xx, yy);
        }
    }
    return 0;
}
```

13.树套树

（1）线段树套FHQ Treap

```
int ch[mn * 40][2], pri[mn * 40], sze[mn * 40], w[mn * 40];
int cnt, xx, yy, zz, rot[mn << 2], n, q;
int a[mn];
inline void treap_update(int rt) {
    sze[rt] = 1;
    if(ch[rt][0]) sze[rt] += sze[ch[rt][0]];
    if(ch[rt][1]) sze[rt] += sze[ch[rt][1]];
}
inline int newnode(int ww = 0) {
    w[++cnt] = ww;
    sze[cnt] = 1;
    pri[cnt] = rand();
    return cnt;
}
inline int merge(int x, int y) {
    if(!x || !y) return x + y;
    if(pri[x] < pri[y]) {
        ch[x][1] = merge(ch[x][1], y);
        treap_update(x);
        return x;
    } else {
        ch[y][0] = merge(x, ch[y][0]);
        treap_update(y);
        return y;
    }
}
inline void split(int rt, int k, int &x, int &y) {
    if(!rt) x = y = 0;
    else {
        if(w[rt] <= k)
            x = rt, split(ch[rt][1], k, ch[rt][1], y);
        else
            y = rt, split(ch[rt][0], k, x, ch[rt][0]);
        treap_update(rt);
    }
}
inline int findkth(int rt, int k) {
    if(sze[rt] < k || sze[rt] == 0 || k == 0) return -1;
    while(1119) {
        if(k <= sze[ch[rt][0]])
            rt = ch[rt][0];
        else {
            if(ch[rt][0]) k -= sze[ch[rt][0]];
            if(!--k) return rt;
            rt = ch[rt][1];
        }
    }
}
```

```
inline void fhq_insert(int &rt, int a) {
    int xx, yy;
    split(rt, a, xx, yy);
    rt = merge(merge(xx, newnode(a)), yy);
}
inline void fhq_delete(int &rt, int a) {
    int xx, yy, zz;
    split(rt, a, xx, zz);
    split(xx, a - 1, xx, yy);
    yy = merge(ch[yy][0], ch[yy][1]);
    rt = merge(merge(xx, yy), zz);
}
inline int fhq_pre(int &rt, int a) {
    int xx, yy, ans;
    split(rt, a - 1, xx, yy);
    int tmp = findkth(xx, sze[xx]);
    if(tmp != -1)
        ans = w[tmp];
    else ans = -2147483647;
    rt = merge(xx, yy);
    return ans;
}
inline int fhq_suf(int &rt, int a) {
    int xx, yy, ans;
    split(rt, a, xx, yy);
    int tmp = findkth(yy, 1);
    if(tmp != -1)
        ans = w[tmp];
    else ans = 2147483647;
    rt = merge(xx, yy);
    return ans;
}
inline int fhq_find(int &rt, int a) {
    int xx, yy, ans;
    split(rt, a - 1, xx, yy);
    ans = sze[xx];
    rt = merge(xx, yy);
    return ans;
}
// FHQ Treap ---------------------------------------------

#define lson l, m, rt << 1
#define rson m + 1, r, rt << 1 | 1
#define root 1, n, 1
inline void build(int l, int r, int rt) {
    go(i, l, r, 1)
        fhq_insert(rot[rt], a[i]);
    if(l == r) return ;
    int m = (l + r) >> 1;
    build(lson), build(rson);
}
inline void modify(int l, int r, int rt, int now, int v) {
    fhq_delete(rot[rt], a[now]);
    fhq_insert(rot[rt], v);
    if(l == r) return ;
    int m = (l + r) >> 1;
    if(now <= m) modify(lson, now, v);
    else modify(rson, now, v);
```

```cpp
}
inline int query_rk(int l, int r, int rt, int nowl, int nowr, int v) {
    if(nowl <= l && r <= nowr) {
        return fhq_find(rot[rt], v);
    }
    int m = (l + r) >> 1;
    if(nowl <= m) {
        if(m < nowr) return query_rk(lson, nowl, nowr, v) + query_rk(rson, nowl,
nowr, v);
        else return query_rk(lson, nowl, nowr, v);
    } else return query_rk(rson, nowl, nowr, v);
}
inline int query_pre(int l, int r, int rt, int nowl, int nowr, int v) {
    if(nowl <= l && r <= nowr) {
        return fhq_pre(rot[rt], v);
    }
    int m = (l + r) >> 1;
    if(nowl <= m) {
        if(m < nowr) return max(query_pre(lson, nowl, nowr, v), query_pre(rson,
nowl, nowr, v));
        else return query_pre(lson, nowl, nowr, v);
    } else return query_pre(rson, nowl, nowr, v);
}
inline int query_suf(int l, int r, int rt, int nowl, int nowr, int v) {
    if(nowl <= l && r <= nowr) {
        return fhq_suf(rot[rt], v);
    }
    int m = (l + r) >> 1;
    if(nowl <= m) {
        if(m < nowr) return min(query_suf(lson, nowl, nowr, v), query_suf(rson,
nowl, nowr, v));
        else return query_suf(lson, nowl, nowr, v);
    } else return query_suf(rson, nowl, nowr, v);
}
// Segment tree ----------------------------------------

int main() {
    n = read(), q = read();
    go(i, 1, n, 1) {
        a[i] = read();
    }
    build(root);
    go(i, 1, q, 1) {
        int s = read(), x, y, v;
        if(s == 1) {
            x = read(), y = read(), v = read();
            printf("%d\n", query_rk(root, x, y, v) + 1);
        } else if(s == 2) {
            x = read(), y = read(), v = read();
            int l = 0, r = 1e8 + 5, ans = 0;
            while(l <= r) {
                int m = (l + r) >> 1;
                if(query_rk(root, x, y, m) + 1 <= v)
                    ans = m, l = m + 1;
                else r = m - 1;
            }
            printf("%d\n", ans);
        } else if(s == 3) {
            x = read(), v = read();
            modify(root, x, v);
            a[x] = v;
        } else if(s == 4) {
            x = read(), y = read(), v = read();
            printf("%d\n", query_pre(root, x, y, v));
        } else if(s == 5) {
            x = read(), y = read(), v = read();
            printf("%d\n", query_suf(root, x, y, v));
        }
    }
    return 0;
}
```

14.分块

```
int n, m, blo = 888;
ll z[mn], col[mn], sum[mn]; int bl[mn];
inline void modify(int l, int r, ll v) {
    go(i, l, min(bl[l] * blo, r), 1)
        z[i] += v, sum[bl[l]] += v;
    // sum[bl[l]] += (min(bl[l] * blo, r) - l + 1) * v;
    if(bl[l] != bl[r]) {
        go(i, (bl[r] - 1) * blo + 1, r, 1)
            z[i] += v, sum[bl[r]] += v;
        // sum[bl[r]] += (r - (bl[r] - 1) * blo) * v;
    }
    go(i, bl[l] + 1, bl[r] - 1, 1)
        col[i] += v;
}
inline ll query(int l, int r) {
    ll ans = 0;
    go(i, l, min(bl[l] * blo, r), 1)
        ans += (z[i] + col[bl[l]]);
    if(bl[l] != bl[r])
        go(i, (bl[r] - 1) * blo + 1, r, 1)
            ans += (z[i] + col[bl[r]]);
    go(i, bl[l] + 1, bl[r] - 1, 1)
        ans += sum[i] + col[i] * blo;
    // return ans % mod;
    return ans;
}

int main() {
    n = read(), m = read();
    go(i, 1, n, 1) z[i] = read();
    go(i, 1, n, 1) bl[i] = (i - 1) / blo + 1;
    go(i, 1, n, 1) sum[bl[i]] += z[i];
    go(i, 1, m, 1) {
        int s = read(), l = read(), r = read(), v;
        if(s == 1) v = read(), modify(l, r, v);
        if(s == 2) printf("%lld\n", query(l, r));
    }
    // getchar();
    return 0;
}
```

15.左偏树

```
int n, m, cnt;
int ch[mn][2], w[mn], fa[mn], dep[mn]; // 左偏树
bool vis[mn];
inline int findx(int x) { return x == fa[x] ? x : fa[x] = findx(fa[x]); }

inline int merge(int x, int y) {
    if(!x || !y) return x + y;
    if(w[x] > w[y] || (w[x] == w[y] && x > y)) swap(x, y);
    ch[x][1] = merge(ch[x][1], y);
    if(dep[ch[x][0]] < dep[ch[x][1]]) swap(ch[x][0], ch[x][1]);
    dep[x] = dep[ch[x][1]] + 1;
    fa[ch[x][0]] = fa[ch[x][1]] = x;
    return x;
}
inline int del(int x) {
    if(vis[x]) return -1;
    vis[x] = 1;
    int tmp = w[x];
    fa[ch[x][0]] = ch[x][0];
    fa[ch[x][1]] = ch[x][1];
    fa[x] = merge(ch[x][0], ch[x][1]);
    return tmp;
}
inline void solve() {
    n = read(), m = read();
    go(i, 1, n, 1) w[i] = read(), fa[i] = i;
    go(i, 1, m, 1) {
        int s = read(), x, y;
        if(s == 1) {
            x = read(), y = read();
            if(!vis[x] && !vis[y]) merge(findx(x), findx(y));
        } else {
            x = read();
            if(vis[x]) puts("-1");
            else printf("%d\n", del(findx(x)));
        }
    }
}
```

16.珂朵莉树(ODT)

```
// using ll read()
inline ll mul(ll a, ll b, ll mod) {
    a %= mod;
    ll ans = 1ll;
    for(; b; b >>= 1) {
        if(b & 1) ans = a * ans % mod;
        a = a * a % mod;
    }
    return ans % mod;
}
class ODT {
    public:
```

```cpp
        struct tree {
            int l, r;
            mutable ll v;
            tree(int _l, int _r = -1, ll _v = 0):
                l(_l), r(_r), v(_v) {}
            bool operator < (const tree &b) const {
                return l < b.l;
            }
        };
        #define IT std::set<tree>::iterator
        IT split(int pos) {
            IT it = s.lower_bound(tree(pos));
            if(it != s.end() && it->l == pos) return it;
            --it;
            int l = it->l, r = it->r;
            ll v = it->v;
            s.erase(it);
            s.insert(tree(l, pos - 1, v));
            return s.insert(tree(pos, r, v)).first;
        }
        std::set<tree> s;
        void assign(int l, int r, ll val = 0) {
            IT itl = split(l), itr = split(r + 1);
            s.erase(itl, itr);
            s.insert(tree(l, r, val));
        }
        void add(int l, int r, ll val) {
            IT itl = split(l), itr = split(r + 1);
            for(; itl != itr; ++itl) itl->v += val;
        }
        ll query_rank(int l, int r, int rk) {
            std::vector<std::pair<ll, int> > vp;
            IT itl = split(l), itr = split(r + 1);
            vp.clear();
            for(; itl != itr; ++itl)
                vp.push_back(std::pair<ll, int>(itl->v, itl->r - itl->l + 1));
            sort(vp.begin(), vp.end());
            for(std::vector<std::pair<ll, int> >::iterator it = vp.begin(); it
!= vp.end(); ++it) {
                rk -= it->second;
                if(rk <= 0) return it->first;
            }
            return -1ll;
        }
        ll query_sum(int l, int r, int ex, int mod) {
            IT itl = split(l), itr = split(r + 1);
            ll sum = 0ll;
            for(; itl != itr; ++itl)
                sum = (sum + (mul(itl->v, (ll)ex, (ll)mod) * (ll)(itl->r - itl-
>l + 1) % mod) % mod) % mod;
            return sum;
        }
} odt;

int n, m;
ll a[mn];
inline void solve() {
    n = read(), m = read(), seed = read(), vmax = read();
    go(i, 1, n, 1) {
        a[i] = (rnd() % vmax) + 1;
        odt.s.insert(ODT::tree(i, i, a[i]));
    }
    odt.s.insert(ODT::tree(n + 1, n + 1, 0ll));
    // options below
}
inline void init() {
    odt.s.clear();
}
```

四.其他

（一）字符串算法

1.manacher算法

P3805 【模板】manacher算法——AC代码

```
char s[mn], str[mn];
int f[mn], l;
inline void manacher() {
    int nowmid = 0, nowr;
    for (int i = l; str[i] != 0; i++) str[i] = 0;
    go(i, 1, l - 1, 1) {
        if (nowmid > i) {
            f[i] = min(f[nowr * 2 - i], f[nowr] + nowr - i);
        } else f[i] = 1;
        while (str[i + f[i]] == str[i - f[i]]) ++f[i];
        if (i + f[i] > nowmid) {
            nowmid = i + f[i];
            nowr = i;
        }
    }
}
inline void init(char a) {
    str[0] = a, str[1] = a;
    go(i, 0, l - 1, 1) {
        str[(i << 1) + 2] = s[i];
        str[(i << 1) + 3] = a;
    }
    l = (l << 1) + 2;
    str[l] = 0;
}
inline char huaji()
{
    srand((unsigned)time(NULL));
    int o = rand() % 120;
    while ((o >= 'a' && o <= 'z') || (o >= 7 && o <= 10))
        o = rand() % 120;
    return char(o);
}
int main()
{
    scanf("%s", s);
    l = strlen(s);
    char a = huaji();
    init(a); manacher();
    int ans = -1;
    go(i, 0, l - 1, 1) ans = max(f[i], ans);
    cout << ans - 1;
    return 0;
}
```

2.Trie树

```
struct node {
    int next[26];
    bool exist;
    node() {
        exist = false;
        memset(next, 0, sizeof(next));
    }
} z[233333];
int cnt = 1;
inline void insert(char *s) {
    int l = strlen(s + 1);
    int p = root;
    go(i, 1, l, 1) {
        if (z[p].next[s[i] - 'a'] == 0) {
            cnt++;
            z[p].next[s[i] - 'a'] = cnt;
        }
        p = z[p].next[s[i] - 'a'];
    }
    z[p].exist = true;
}
inline bool query(char *s) {
    int l = strlen(s + 1);
    int p = root;
    go(i, 1, l, 1) {
        if (z[p].next[s[i] - 'a'] == 0) return false;
        p = z[p].next[s[i] - 'a'];
    }
    return z[p].exist;
}
```

3.字符串hash

(1）自然溢出法：P3370 【模板】字符转哈希——AC代码

```
char s[ms];
int n, sum;
ull h[ms], bit[ms];
ull a[mn], t;
int main()
{
    n = read();
    bit[0] = 1;
    go(i, 1, ms - 30, 1)
        bit[i] = bit[i - 1] * base;
    //采用自然炸裂法（逃
    go(x, 1, n, 1) {
        scanf("%s", s);
        ull l = strlen(s);
        go(i, 0, l - 1, 1)
            h[i + 1] = h[i] * base + s[i];
        a[x] = h[l];
    }
    sort(a + 1, a + n + 1);
    go(i, 1, n, 1) {
        if (t != a[i]) sum++;
        t = a[i];
    }
    cout << sum << "\n";
    return 0;
}
```

(2) 单模哈希法：P3370 【模板】字符转哈希——80分代码

```
char s[ms];
const ull p = 19260817;
int n, sum;
ull h[ms], bit[ms];
ull a[mn], t;
int main() {
    n = read();
    bit[0] = 1;
    go(i, 1, ms - 30, 1)
        bit[i] = (bit[i - 1] * base) % p;
    //采用dan膜炸裂法（逃
    go(x, 1, n, 1) {
        scanf("%s", s);
        ull l = strlen(s);
        go(i, 0, l - 1, 1) {
            h[i + 1] = (h[i] * base + s[i]) % p;
        }
        a[x] = h[l];
    }
    sort(a + 1, a + n + 1);
    go(i, 1, n, 1) {
        if (t != a[i]) sum++;
        t = a[i];
    }
    cout << sum << "\n";
    return 0;
}
```

(3) 双模哈希法：P3370 【模板】字符转哈希——AC代码

```
struct node {
    ull x, y;
} a[mn];
char s[mn];
int n, ans = 1;
inline bool cmp(node a, node b) {
    return a.x < b.x;
}
inline ull hash1(char s[]) {
    int len = strlen(s);
    ull ans = 0;
    for (int i = 0; i < len; i++)
        ans = (ans * base + (ull)s[i]) % mod1;
    return ans;
}
inline ull hash2(char s[]) {
    int len = strlen(s);
    ull ans = 0;
    for (int i = 0; i < len; i++)
        ans = (ans * base + (ull)s[i]) % mod2;
    return ans;
}
int main() {
    n = read();
    for (int i = 1; i <= n; i++) {
        scanf("%s", s);
        a[i].x = hash1(s);
        a[i].y = hash2(s);
    }
    sort(a + 1, a + n + 1, cmp);
    for (int i = 2; i <= n; i++)
        if (a[i].x != a[i - 1].x || a[i - 1].y != a[i].y)
            ans++;
    cout << ans;
}
```

4.KMP字符串匹配

POJ 3461 乌力波————AC代码

```
struct KMP{
    int ne[mn], len;
    inline void get_ne(char ch[]){
        memset(ne, 0, sizeof 0);
        ne[0] = ne[1] = 0;
        len = strlen(ch);
        go(i,1,len-1,1){
            int x = ne[i];
            while(x && ch[i] != ch[x])
                x = ne[x];
            ne[i + 1] = ch[i] == ch[x] ? x + 1 : 0;
        }
    }
    inline int finds(char ch[], char s[]){
        int x = 0, ans = 0;
        for(int i = 0; s[i]; i++){
            while(x && ch[x] != s[i])
                x = ne[x];
            if(ch[x] == s[i])
                x++;
            if(x == len)
                ans++;
        }
        return ans;
    }
    inline void debug(){//附赠debug
        go(i, 1, len, 1)
```

```
            printf("ne[%d] = %d\n", i, ne[i]);
    }
} worker;
char ch[mn], s[mn];
int T;
inline void init() {
    memset(ch, 0, sizeof ch);
    memset(s, 0, sizeof s);
}
int main(){
    T = read();
    while(T--) {
        init();
        scanf("%s%s", ch, s);
        worker.get_ne(ch);
        printf("%d\n", worker.finds(ch, s));
    }
    return 0;
}
```

P3375 【模板】KMP字符串匹配 ————AC代码

```

struct KMP{
    int len, ne[mn];
    inline void get_ne(char ch[]) {
        memset(ne, 0, sizeof ne);
        ne[0] = ne[1] = 0;
        len = strlen(ch);
        go(i, 1, len - 1, 1) {
            int x = ne[i];
            while(x && ch[i] != ch[x])
                x = ne[x];
            ne[i + 1] = ch[i] == ch[x] ? x + 1 : 0;
        }
    }
    inline int finds(char ch[], char s[]) {
        int x = 0, ans = 0;
        for(int i = 0; s[i]; i++){
            while(x && ch[x] != s[i])
                x = ne[x];
            if(ch[x] == s[i])
                x++;
            if(x == len)
                printf("%d\n", i - len + 2), ans++;
        }
        return ans;
    }
    inline void output() {
        go(i, 1, len, 1)
            printf("%d ", ne[i]);
        puts("");
    }
} kmp;
char ch[mn], s[mn];
int main() {
    scanf("%s%s", s, ch);
    kmp.get_ne(ch);
    int ans = kmp.finds(ch, s);
    kmp.output();
    int _ = 0;
    return ~~(0^_^0);
}
```

5.AC自动机

by Rhein_E

```
struct AC_Automaton {
    struct Node {
        Node *next[26], *fail;
        int end;
        Node() {
            for (int i = 0; i < 26; ++i)
                next[i] = 0;
            end = 0, fail = 0;
        }
    } * root;
    Node *que[MAXN];
    int hd, tl;

    AC_Automaton() { root = new Node; }
    Node *insert(char *str) {
        int len = strlen(str);
        Node *p = root;
        for (int i = 0; i < len; ++i) {
            if (!p->next[str[i] - 'a']) p->next[str[i] - 'a'] = new Node;
            p = p->next[str[i] - 'a'];
        }
        ++p->end;
        return p;
    }
    void build_fail() {
        hd = tl = 0;
        for (int i = 0; i < 26; ++i)
            if (root->next[i]) root->next[i]->fail = root, que[tl++] = root-
>next[i];
        while (hd < tl) {
            Node *p = que[hd++];
            for (int i = 0; i < 26; ++i)
                if (p->next[i]) {
                    if (p->fail->next[i])
                        p->next[i]->fail = p->fail->next[i];
                    else
                        p->next[i]->fail = root;
                    que[tl++] = p->next[i];
                } else
                    p->next[i] = p->fail->next[i];
        }
    }
    int match(char *str) {
        int len = strlen(str), res = 0;
        Node *p = root;
        for (int i = 0; i < len; ++i) {
            if (p->next[str[i] - 'a'])
                p = p->next[str[i] - 'a'];
            else
                p = root;
            for (Node *q = p; q && ~q->end; q = q->fail)
                res += q->end, q->end = -1;
        }
        return res;
    }
} ac;
int n;
char s[MAXN], t[MAXN];
int main() {
    scanf("%d", &n);
    while (n--) {
        scanf("%s", s);
        ac.insert(s);
    }
    ac.build_fail();
    scanf("%s", t);
    printf("%d\n", ac.match(t));
    return 0;
}
```

五、其他

1.排序算法（暂且不在数论里）

(1)归并排序

```
int a[mn], by[mn];
inline void msort(int *A, int x, int y, int *T) {
    if (y - x > 1) {
        int m = x + (y - x) / 2;
        int p = x, q = m, i = x;
        msort(A, x, m, T);
        msort(A, m, y, T);
        while (p < m || q < y) {
            if (q >= y || (p < m && A[p] <= A[q]))
                T[i++] = A[p++];
            else
                T[i++] = A[q++];
        }
        for (i, x, y - 1, 1) {
            A[i] = T[i];
        }
    }
}
```

(2)快速排序

```
int n, a[100005];
int qsort(int l, int r)
{
    int i, j, mid, p;
    i = l; j = r;
    mid = a[(l + r) / 2];
    do {
        while (a[i] < mid) i++;
        while (a[j] > mid) j--;
        if (i <= j) {
            p = a[i]; a[i] = a[j]; a[j] = p;
            i++; j--;
        }
    } while (i <= j);
    if (l < j) qsort(l, j);
    if (i < r) qsort(i, r);
}
```

(3)堆排序

(见数据结构->堆)

(4)冒泡排序

```
    for(int i = 1; i <= n; i++)
        for(int j = 1; j <= n - i; j++)
            if(a[j] > a[j + 1])
                swap(a[j], a[j + 1]);
}
```

2.LCS（最长公共子序列）

```
int f[mn];
int a[mn], b[mn], c[mn];
int n, l;
int main() {
    n = read();
    go(i, 1, n, 1) a[i] = read(), c[a[i]] = i;
    go(i, 1, n, 1) b[i] = read(), f[i] = inf;
    f[0] = 0, l = 0;
    go(i, 1, n, 1) {
        int le = 0, ri = l, mid;
        if (c[b[i]] > f[l]) f[++l] = c[b[i]];
        else {
            while (le < ri) {
                mid = (le + ri) / 2;
                if (f[mid] > c[b[i]])
                    ri = mid;
                else
                    le = mid + 1;
            }
            f[le] = min(c[b[i]], f[le]);
        }
    }
    cout << l;
    return 0;
}
```

3.舞蹈链(Dancing_Links_X)

by Rhein_E

```
// luogu P4929
#include <cstdio>
#include <cstring>
#include <iostream>

const int MAXN = 1010;

struct Node {
    Node *left, *right, *up, *down, *head; //指向四个方向和列的表头
    int row, cnt; //记录行号(用于记录答案)及列元素个数
} *head, *cols[MAXN];

int mtx[MAXN][MAXN], n, m;
int ans[MAXN], top;

void init();
bool solve();

int main() {
    scanf("%d%d", &n, &m);
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < m; ++j)
            scanf("%d", &mtx[i][j]);
    init();
    if (solve()) {
        //puts("One solution found:");
        for (int i = 0; i < top; ++i) printf("%d ", ans[i] + 1);
        puts("");
    } else puts("No Solution!");


    return 0;
}
void init() {
    top = 0;
    head = new Node;
    for (int i = 0; i < m; ++i) {
        cols[i] = new Node;
        cols[i]->row = -1;
        cols[i]->head = cols[i];
    }
    for (int i = 1; i < m; ++i) cols[i]->left = cols[i - 1];
    for (int i = m - 2; i >= 0; --i) cols[i]->right = cols[i + 1];
    cols[0]->left = head, head->right = cols[0];
    cols[m - 1]->right = head, head->left = cols[m - 1];

    Node *p[MAXN]; //记录列尾
    for (int i = 0; i < m; ++i) p[i] = cols[i];
    for (int i = 0; i < n; ++i) {
        Node *nplast = NULL;
        for (int j = 0; j < m; ++j)
            if (mtx[i][j]) {
                Node *np = new Node;
                np->row = i;
                //插入列中
                np->up = p[j], p[j]->down = np;
                np->down = p[j]->head, p[j]->head->up = np;
                np->head = p[j]->head;

                //插入行中
                if (nplast) {
                    np->right = nplast->right;
                    nplast->right->left = np;
                    np->left = nplast, nplast->right = np;
                } else {
                    np->left = np->right = np;
                    nplast = np;
                }
                p[j] = np;
                ++np->head->cnt;
            }

    }
}
bool solve() {
    if (head->right == head) return 1; //还有列未被覆盖

    // 找到元素个数最少的列，能有效降低复杂度
    Node *c = head->right;
    for (Node *p = c->right; p != head; p = p->right)
        if (p->cnt < c->cnt) c = p;

    for (Node *p = c->down; p != c; p = p->down) {//枚举选择的行
        for (Node *pp = p->right; pp != p; pp = pp->right) { //相关的列
            for (Node *ppc = pp->down; ppc != pp; ppc = ppc->down) { //包含相关列的
行
                if (ppc != pp->head)
                    for (Node *ppr = ppc->right; ppr != ppc; ppr = ppr->right) {
                        ppr->up->down = ppr->down, ppr->down->up = ppr->up;
                        --ppr->head->cnt;
                    }
            }
            pp->head->left->right = pp->head->right;
            pp->head->right->left = pp->head->left;
        }
        for (Node *ppc = p->down; ppc != p; ppc = ppc->down) {
            if (ppc != p->head)
                for (Node *ppr = ppc->right; ppr != ppc; ppr = ppr->right) {
                    ppr->up->down = ppr->down, ppr->down->up = ppr->up;
                    --ppr->head->cnt;
                }
        }
        p->head->left->right = p->head->right;
        p->head->right->left = p->head->left;

        // 记录答案，继续搜索
        ans[top++] = p->row;
        if (solve()) return 1;
        --top;

        // 撤销选择行的操作
        for (Node *pp = p->right; pp != p; pp = pp->right) { //相关的列
            for (Node *ppc = pp->down; ppc != pp; ppc = ppc->down) { //包含相关列的行
                if (ppc != pp->head)
                    for (Node *ppr = ppc->right; ppr != ppc; ppr = ppr->right) {
                        ppr->up->down = ppr->down->up = ppr;
                        ++ppr->head->cnt;
                    }
            }
            pp->head->left->right = pp->head->right->left = pp->head;
        }
        for (Node *ppc = p->down; ppc != p; ppc = ppc->down) {
            if (ppc != p->head)
                for (Node *ppr = ppc->right; ppr != ppc; ppr = ppr->right) {
                    ppr->up->down = ppr->down->up = ppr;
                    ++ppr->head->cnt;
                }
        }
        p->head->left->right = p->head->right->left = p->head;
    }
    return 0;
}
//Rhein_E
```

希望可以有所帮助!