数据结构

- 普通平衡树

luogu 普通平衡树

```

#include <bits/stdc++.h>
#define rep(i,a,b) for (int i=a;i<=b;++i)
#define per(i,a,b) for (int i=a;i>=b;--i)
#define mst(x,a) memset(x,a,sizeof(x))
using namespace std;

typedef long long ll;
typedef double db;

const int MAXN = 1e5+5;

int MAX(int a,int b) {return a>b?a:b;}
int MIN(int a,int b) {return a<b?a:b;}

int val[MAXN]; //编号i的值
int num[MAXN]; //值为val[i]的有num[i]个
int ch[MAXN][2],fa[MAXN],sz[MAXN];
int ROOT,tot;

void clear(int x) { ch[x][0] = ch[x][1] = fa[x] = sz[x] = val[x] = num[x] = 0; }
bool which(int x) { return x == ch[fa[x]][1]; }
void maintain(int x) { sz[x] = sz[ch[x][0]] + sz[ch[x][1]] + num[x]; }
void rotate(int x) {
    int p = fa[x], gp = fa[p], wc=which(x);
    ch[p][wc] = ch[x][wc^1]; fa[ch[x][wc^1]] = p;
    ch[x][wc^1] = p; fa[p] = x;
    fa[x] = gp; if (gp) ch[gp][p == ch[gp][1]] = x;
    maintain(x);maintain(p);
}
void splay(int x) {
    for (int p;p=fa[x];rotate(x))
        if (fa[p]) rotate((which(x) == which(p)?p:x));
    ROOT = x;
}
int getrank(int k) {
    int now = ROOT, ret = 0;
    while (true) {
        if (k < val[now]) now = ch[now][0];
        else {
            ret += sz[ch[now][0]];
            if (k == val[now]) {
                splay(now);
                return ret+1;
            }
            ret += num[now];
            now = ch[now][1];
        }
    }
}
int find(int k) {
    int now = ROOT;
    while (true) {
        if (ch[now][0] && k <= sz[ch[now][0]]) now = ch[now][0];
        else {
            k -= sz[ch[now][0]]+num[now];
            if (k <= 0) {
                splay(now);
                return val[now];
            }
            now = ch[now][1];
        }
    }
}
int prenxt(bool t) { // 0 pre,  1 nxt
    int now = ch[ROOT][t];
    while (ch[now][t^1]) now = ch[now][t^1];
    splay(now);
    return now;
}
void newnode(int v,int p) {
```

```
    val[++tot] = v; num[tot]++;
    maintain(tot);
    if (p) {
        fa[tot] = p;
        ch[p][v > val[p]] = tot;
        maintain(p);
    }
}
void Insert(int v) {
    if (!ROOT) {
        newnode(v,0);
        ROOT = tot;
        return;
    }
    int now = ROOT,p = 0;
    while (true) {
        if (val[now] == v) {
            num[now]++;
            maintain(now); maintain(p);
            splay(now);
            break;
        }
        p = now;
        now = ch[now][v > val[now]];
        if (!now) {
            newnode(v,p);
            splay(tot);
            break;
        }
    }
}
void Delete(int x) {  //删除值为x的
    getrank(x);
    if (num[ROOT] > 1) {
        num[ROOT]--;
        maintain(ROOT);
        return;
    }
    if (!ch[ROOT][0] && !ch[ROOT][1]) { clear(ROOT); ROOT = 0; return; }
    rep(i,0,1) if (!ch[ROOT][i]) {
        int tmp = ROOT;
        ROOT = ch[ROOT][i^1]; fa[ROOT] = 0;
        clear(tmp);
        return;
    }
    int now = ROOT, pre=prenxt(0); //pre时splay过，用now存之前的root
    fa[ch[now][1]] = pre;
    ch[pre][1] = ch[now][1];
    clear(now);
    maintain(ROOT);
}
int main() {
    tot = 0;
    int n; cin >> n;
    rep(i,1,n) {
        int type; scanf("%d",&type);
        int x; scanf("%d",&x);
        if (type == 1) Insert(x);
        if (type == 2) Delete(x);
        if (type == 3) printf("%d\n",getrank(x));
        if (type == 4) printf("%d\n",find(x));
        if (type == 5) {
            Insert(x);
            printf("%d\n",val[prenxt(0)]);
            Delete(x);
        }
        if (type == 6) {
            Insert(x);
            printf("%d\n",val[prenxt(1)]);
            Delete(x);
        }
    }
    return 0;
}
```

- LCT

```cpp
#include<cstdio>
#include<cstring>
#include<climits>
#include<algorithm>
using namespace std;
const int MAXN = 5 * 1e4 + 5, MAXM = 1e5 + 5, INF = INT_MAX;
int inline swp(int& a, int& b) {
    a ^= b ^= a ^= b;
}
int inline readint() {
    int Num; char ch;
    while ((ch = getchar()) < '0' || ch > '9'); Num = ch - '0';
    while ((ch = getchar()) >= '0' && ch <= '9') Num = Num * 10 + ch - '0';
    return Num;
}
void outint(int x) {
    if (x >= 10) outint(x / 10);
    putchar(x % 10 + '0');
}
struct EDGE {
    int u, v, a, b;
    bool operator < (const EDGE& _)const {
        return a < _.a;
    }
}e[MAXM];
int p[MAXN], lev[MAXN];
int inline getp(int x) {
    return x == p[x] ? x : p[x] = getp(p[x]);
}
void Union(int& x, int& y) {
    int px = getp(x), py = getp(y);
    if (lev[px] > lev[py]) p[py] = px;
    else {
        p[px] = py;
        if (lev[px] == lev[py]) lev[px]++;
    }
}
int fa[MAXN + MAXM], ch[MAXN + MAXM][2], mx[MAXN + MAXM], val[MAXN + MAXM], sta[MAXN + MAXM];
bool rev[MAXN + MAXM];
bool inline isroot(int& x) {
    return ch[fa[x]][0] != x && ch[fa[x]][1] != x;
}
void inline pushup(int& x) {
    mx[x] = x;
    if (val[mx[x]] < val[mx[ch[x][0]]]) mx[x] = mx[ch[x][0]];
    if (val[mx[x]] < val[mx[ch[x][1]]]) mx[x] = mx[ch[x][1]];
}
void inline pushdown(int& x) {
    if (rev[x]) {
        rev[x] ^= 1, rev[ch[x][0]] ^= 1, rev[ch[x][1]] ^= 1;
        swp(ch[x][0], ch[x][1]);
    }
}
void rot(int x, int p) {
    int y = fa[x], z = fa[y];
    fa[ch[x][!p]] = y, ch[y][p] = ch[x][!p];
    fa[x] = z; if (!isroot(y)) ch[z][ch[z][1] == y] = x;
    fa[y] = x, ch[x][!p] = y;
    pushup(y), pushup(x);
}
void splay(int x) {
    int top = 1; sta[top] = x;
    for (int i = x; !isroot(i); i = fa[i]) sta[++top] = fa[i];
    while (top) pushdown(sta[top--]);
    while (!isroot(x)) {
        if (isroot(fa[x])) rot(x, ch[fa[x]][1] == x);
        else {
            int y = fa[x], z = fa[y], p = ch[z][1] == y;
            if (ch[y][p] == x) rot(y, p), rot(x, p);
            else rot(x, !p), rot(x, p);
        }
    }
}
void access(int x) {
    for (int t = 0; x; t = x, x = fa[x]) {
        splay(x);
```

```
        ch[x][1] = t;
        pushup(x);
    }
}
void mkrt(int x) {
    access(x); splay(x); rev[x] ^= 1;
}
void link(int x, int y) {
    mkrt(x); fa[x] = y;
}
void cut(int x, int y) {
    mkrt(x); access(y); splay(y);
    fa[x] = ch[y][0] = 0; pushup(y);
}
int qry(int x, int y) {
    mkrt(x); access(y); splay(y);
    return mx[y];
}
int main() {
    memset(lev, 0, sizeof(lev));
    memset(rev, false, sizeof(rev));
    int n = readint(), m = readint(), ans = INF;
    for (int i = 1; i <= n; i++) p[i] = i;
    for (int i = 1; i <= m; i++) e[i].u = readint(), e[i].v = readint(), e[i].a = readint(), e[i].b =
readint();
    sort(e + 1, e + m + 1);
    for (int i = 1; i <= m; i++) {
        int u = e[i].u, v = e[i].v, a = e[i].a, b = e[i].b, pu = getp(u), pv = getp(v);
        if (pu == pv) {
            int t = qry(u, v);
            if (val[t] > b) cut(t, e[t - n].u), cut(t, e[t - n].v);
            else continue;
        }
        else Union(pu, pv);
        val[i + n] = b; mx[i + n] = i + n;
        link(u, i + n); link(v, i + n);
        if (getp(1) == getp(n)) ans = min(ans, a + val[qry(1, n)]);
    }
    if (ans == INF) puts("-1");
    else outint(ans);
    return 0;
}
```

- ST

```
#include <bits/stdc++.h>
using namespace std;
int n,a[100005],LOG[100005];
int st[100005][25];
int MAX(int a, int b) {return a>b?a:b;}
void getLOG(int a) {
    LOG[1] = 0;
    for (int i=2;i<=n;i++) LOG[i] = LOG[i>>1] + 1;
}
int query(int l, int r) {
    int k = LOG[r-l+1];
    return MAX(st[l][k],st[r-(1<<k)+1][k]);
}
int main() {
    cin>>n;
    getLOG(n);
    for (int i=1;i<=n;i++) {
        cin>>a[i];
        st[i][0] = a[i];
    }
    for (int j=1;j<=20;j++)
        for (int i=1;i+(1<<j)-1<=n;i++)
            st[i][j] = MAX(st[i][j-1],st[i+(1<<(j-1))][j-1]);
    int a,b,c;
    cin>>a>>b;
    c = query(a,b);
    return 0;
}
```

- 文艺平衡树

luogu 文艺平衡树

```cpp

#include <bits/stdc++.h>
#define rep(i,a,b) for (int i=a;i<=b;++i)
#define per(i,a,b) for (int i=a;i>=b;--i)
#define mst(x,a) memset(x,a,sizeof(x))
using namespace std;

typedef long long ll;
typedef double db;

const int MAXN = 1e5+5;
const int inf = 1e9;

int MAX(int a,int b) {return a>b?a:b;}
int MIN(int a,int b) {return a<b?a:b;}

int n,m;
int val[MAXN],num[MAXN];
int ch[MAXN][2],fa[MAXN],sz[MAXN];
bool tag[MAXN];
int ROOT,tot;

void clear(int x) { ch[x][0] = ch[x][1] = fa[x] = sz[x] = val[x] = num[x] = tag[x] = 0; }
bool which(int x) { return x == ch[fa[x]][1]; }
void maintain(int x) {
    if (!x) return;
    sz[x] = num[x];
    if (ch[x][0]) sz[x] += sz[ch[x][0]];
    if (ch[x][1]) sz[x] += sz[ch[x][1]];
}
void pushdown(int x) {
    if (tag[x]) {
        if (ch[x][0]) tag[ch[x][0]] ^= 1;
        if (ch[x][1]) tag[ch[x][1]] ^= 1;
        swap(ch[x][0],ch[x][1]);
        tag[x] = 0;
    }
}
int BuildTree(int p, int l, int r) {
    if (l > r) return 0;
    int now = ++tot;
    int mid = (l+r)/2;
    clear(now);
    fa[now] = p;
    val[now] = mid-1;
    if (mid == 1) val[now] = -inf;
    if (mid == n+2) val[now] = inf;
    num[now] = 1;
    ch[now][0] = BuildTree(now,l,mid-1);
    ch[now][1] = BuildTree(now,mid+1,r);
    maintain(now);
    return now;
}
void rotate(int x) {
    int p = fa[x], gp = fa[p];
    pushdown(p); pushdown(x);
    int wc=which(x);
    ch[p][wc] = ch[x][wc^1]; fa[ch[x][wc^1]] = p;
    ch[x][wc^1] = p; fa[p] = x;
    fa[x] = gp; if (gp) ch[gp][p == ch[gp][1]] = x;
    maintain(x);maintain(p);
}
void splay(int x,int to) {
    for (int p;(p=fa[x]) && fa[x]!=to;rotate(x))
        if (fa[p] != to) rotate((which(x) == which(p)?p:x));
    if (to == 0) ROOT = x;
}
int find(int k) {
    int now = ROOT;
    while (true) {
        pushdown(now);
        if (k <= sz[ch[now][0]]) now = ch[now][0];
        else if (k == sz[ch[now][0]]+1) return now;
        else {
            k -= sz[ch[now][0]]+1;
            now = ch[now][1];
        }
    }
```

```
}
void reverse(int l,int r) {
    l = find(l-1);
    r = find(r+1);
    splay(l,0); splay(r,l);
    tag[ch[ch[ROOT][1]][0]] ^= 1;
}
void output(int x) {
    pushdown(x);
    if (ch[x][0]) output(ch[x][0]);
    if (val[x]!=inf && val[x]!=-inf) cout<<val[x]<<' ';
    if (ch[x][1]) output(ch[x][1]);
}
int main() {
    tot = 0;
    cin >> n >> m;
    ROOT = BuildTree(0,1,n+2);
    rep(i,1,m) {
        int x, y;
        cin >> x >> y;
        reverse(x+1,y+1);
    }
    output(ROOT);
    cout<<endl;
    return 0;
}
```

- 无旋treap luogu 普通平衡树

```

#include <bits/stdc++.h>
#define rep(i,a,b) for (int i=a;i<=b;++i)
#define mst(a,b) memset(a,b,sizeof(a))
using namespace std;

const int MAXN = 4e5+5;

int n,m,tot;
int ch[MAXN][2],sz[MAXN];
int val[MAXN],pri[MAXN];

void clear(int x) { ch[x][0] = ch[x][1] = sz[x] = val[x] = 0; }
void maintain(int x) { sz[x] = sz[ch[x][0]] + sz[ch[x][1]]; }
int newnode(int v) {
    val[++tot] = v; sz[tot] = 1; pri[tot] = rand();
    return tot;
}
void split_rank(int now,int k,int &x,int &y) {
    if (!now) {
        x = 0, y = 0;
        return;
    }
    if (sz[ch[now][0]] < k) x = now, split_rank(ch[now][1],k-1-sz[ch[now][0]],ch[now][1],y);
    else y = now, split_rank(ch[now][0],k,x,ch[now][0]);
    maintain(now);
}
void split_val(int now,int k,int &x,int &y) {
    if (!now) {
        x = 0, y = 0;
        return;
    }
    if (val[now] <= k) x = now, split_val(ch[now][1],k,ch[now][1],y);
    else y = now, split_val(ch[now][0],k,x,ch[now][0]);
    maintain(now);
}
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);
        maintain(x);
        return x;
    }
    else {
        ch[y][0] = merge(x,ch[y][0]);
        maintain(y);
```

```
        return y;
    }
}

int t1,t2,t3,t4,root;
void Insert(int x) {
    split_val(root,x,t1,t2);
    root = merge(merge(t1,newnode(x)),t2);
}
void Delete(int x) {
    split_val(root,x-1,t1,t2);
    split_rank(t2,1,t3,t4);
    root = merge(t1,t4);
}
int getrank(int x) {
    split_val(root,x-1,t1,t2);
    int ret = sz[t1]+1;
    root = merge(t1,t2);
    return ret;
}
int find(int x) {
    split_rank(root,x-1,t1,t2);
    split_rank(t2,1,t3,t4);
    int ret = val[t3];
    root = merge(t1,merge(t3,t4));
    return ret;
}
int main() {
    srand((unsigned)time(NULL));
    tot = 0;
    root = 0;
    int n; cin >> n;
    rep(i,1,n) {
        int type; scanf("%d",&type);
        int x; scanf("%d",&x);
        if (type == 1) Insert(x);
        if (type == 2) Delete(x);
        if (type == 3) printf("%d\n",getrank(x));
        if (type == 4) printf("%d\n",find(x));
        if (type == 5) {
            split_val(root,x-1,t1,t2);
            split_rank(t1,sz[t1]-1,t3,t4);
            printf("%d\n",val[t4]);
            root = merge(merge(t3,t4),t2);
        }
        if (type == 6) {
            split_val(root,x,t1,t2);
            split_rank(t2,1,t3,t4);
            printf("%d\n",val[t3]);
            root = merge(t1,merge(t3,t4));
        }
    }
    return 0;
}
```

- 笛卡尔树

(k,w)组成，k满足二叉搜索树，w满足堆 当k互不相同，w互不相同，则笛卡尔树结构唯一 建：按k排序（一般当作下标），每次插入到右链末端 找到x.w<u.w，将u接到x右子，x原右子树变为u左子树

```
int sta[N],top;
int ls[N],rs[N];
void build() { //O(n)
    rep(i,1,n) {
        int k = top;
        while (k > 0 && val[sta[k]] > val[i]) k--;
        if (k) rs[sta[k]] = i;
        if (k < top) ls[i] = sta[k+1];
        sta[++k] = i;
        top = k;
    }
}
```

- 普通莫队

```
#include <bits/stdc++.h>
#define rep(i,a,b) for (int i=a;i<=b;++i)
#define per(i,a,b) for (int i=a;i>=b;--i)

using namespace std;
const int N = 1e5+5;

typedef long long ll;
typedef double db;

int QBS,BS,maxx;
int a[N];
struct query {
    int l,r;
    bool operator < (const query &t) const {
        if (l/QBS == t.l/QBS) {
            if ((l/QBS)&1) return r < t.r;
            else return r > t.r;
        }
        return l < t.l
    }
}q[N];
void work(int id,int w) {

}
int get_ans(int x) {

}
int main() {
    scanf("%d%d",&n,&m);
    rep(i,1,n) {
        scanf("%d",&a[i]);
        maxx = max(maxx,a[i]);
    }
    rep(i,1,m) {
        scanf("%d%d",&q[i].l,&q[i].r);
        q[i].id = i;
    }
    QBS = (int)ceil(sqrt(n));
    BS  = (int)ceil(sqrt(maxx));
    sort(q+1,q+m+1);
    int l = 0,r = 0;
    rep(i,1,m) {
        query &cur = q[i];
        while (l < cur.l) work(l++,-1);
        while (l > cur.l) work(--l,1);
        while (r < cur.r) work(++r,1);
        while (r > cur.r) work(r--,-1);
        ans[cur.id] = get_ans(cur.y2)-get_ans(cur.y1-1);
    }
    rep(i,1,m) printf("%d\n",ans[i]);
    return 0;
}
```

可持久化

- 可持久化数组

```
#include <bits/stdc++.h>
using namespace std;
#define rep(i,a,b) for (int i=a;i<=b;++i)
const int MAXN = 1e6+5;
```

```
inline int read() {
    int f = 1,x = 0; char ch;
    do {ch = getchar(); if (ch == '-') f = -1;} while (ch<'0' || ch>'9');
    do {x = x*10+ch-'0'; ch = getchar();} while (ch>='0' && ch<='9');
    return f * x;
}
int a[MAXN];
int tcnt,version;
struct node {
    long long v;
    node *lson, *rson;
}pool[13*MAXN],*root[MAXN*2];

void BuildTree(node *p,int left,int right) {
    if (left == right) {
        p->v = a[left];
        return;
    }
    int mid = (left+right) >> 1;
    p->lson = &pool[++tcnt];
    p->rson = &pool[++tcnt];
    BuildTree(p->lson,left,mid);
    BuildTree(p->rson,mid+1,right);
}
void change(node *last,node *p,int loc,int left,int right,long long val) {
    if (left == right) {
        p->v = val;
        return;
    }
    int mid = (left+right) >> 1;
    if (loc <= mid) {
        p->lson = &pool[++tcnt];
        p->rson = last->rson;
        change(last->lson,p->lson,loc,left,mid,val);
    }
    else {
        p->rson = &pool[++tcnt];
        p->lson = last->lson;
        change(last->rson,p->rson,loc,mid+1,right,val);
    }
}
long long query(node *p,int loc,int left,int right) {
    if (left == right) return p->v;
    int mid = (left+right) >> 1;
    if (loc <= mid) return query(p->lson,loc,left,mid);
    else return query(p->rson,loc,mid+1,right);
}
int main() {
    int n,m;
    n = read(); m = read();
    rep(i,1,n) a[i] = read();
    root[0] = &pool[++tcnt];
    BuildTree(root[0],1,n);
    rep(i,1,m) {
        int ver = read(),tt = read(),loc = read(),val;
        root[++version] = &pool[++tcnt];
        if (tt == 1) {
            val = read();
            change(root[ver],root[version],loc,1,n,val);
        }
        else {
            *root[version] = *root[ver];
            printf("%lld\n",query(root[ver],loc,1,n));
        }
    }
    return 0;
}
```

- hdu1695 GCD

```
#include <iostream>
#include <cstdlib>
#include <cstdio>
#include <cstring>
#include <algorithm>
#define mst(a,b) memset(a,b,sizeof(a))
#define rep(i,a,b) for (int i=a;i<=b;++i)

```

```cpp
using namespace std;

const int MAXN = 100001;
int prime[MAXN],phi[MAXN];
bool vis[MAXN];

void euler() {

}
int main() {
    int t; cin >> t;
    euler();
    rep(_,1,t) {
        int a,b,c,d,k;
        scanf("%d%d%d%d%d",&a,&b,&c,&d,&k);
        if (k == 0 || k > b || k > d) {
            printf("Case %d: 0\n",_);
            continue;
        }
        b /= k;
        d /= k;
        if (b > d) swap(b,d);
        long long ans = 0;
        rep(i,1,b) {
            ans += phi[i];
        }
        rep(i,b+1,d) {
            ans += cal(i,b);
        }
    }
    return 0;
}
```

线段树

- Segment Tree

```cpp
#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <algorithm>
#include <cstring>
#define rep(i,a,b) for (int i=a;i<=b;++i)
#define per(i,a,b) for (int i=a;i>=b;--i)
#define mst(a,b) memset(a,b,sizeof(a))

typedef long long ll;
typedef double db;

using namespace std;

const int MAXN = 1e5+5;

int n,m;
ll a[MAXN];
struct segment_tree {
    #define ls id<<1
    #define rs id<<1|1
    struct node {
        ll sum, lazy;
    }tr[4*MAXN];

    void Update(int id) {
        tr[id].sum = tr[ls].sum + tr[rs].sum;
    }
    void pushdown(int id,int l,int r) {
        int mid = (l+r)/2;
        if (tr[id].lazy) {
            tr[ls].sum += tr[id].lazy*(mid-l+1); tr[ls].lazy += tr[id].lazy;
            tr[rs].sum += tr[id].lazy*(r-mid);   tr[rs].lazy += tr[id].lazy;
            tr[id].lazy = 0;
        }
    }
    void BuildTree(int id,int l,int r) {
        tr[id].lazy = 0;
        if (l == r) {
            tr[id].sum = a[l];
            return;
```

```
        }
        int mid = (l+r) >> 1;
        BuildTree(ls,l,mid); BuildTree(rs,mid+1,r);
        Update(id);
    }
    void Change(int id,int l,int r,int cl,int cr,int k) {
        if (cl <= l && r <= cr) {
            tr[id].lazy += k;
            tr[id].sum += (r-l+1)*k;
            return;
        }
        pushdown(id,l,r);
        int mid = (l+r) >> 1;
        if (cl <= mid) Change(ls,l,mid,cl,cr,k);
        if (cr > mid)  Change(rs,mid+1,r,cl,cr,k);
        Update(id);
    }
    ll Query(int id,int l,int r,int ql,int qr) {
        if (ql <= l && r <= qr) {
            return tr[id].sum;
        }
        pushdown(id,l,r);
        int mid = (l+r) >> 1;
        ll ret = 0;
        if (ql <= mid) ret += Query(ls,l,mid,ql,qr);
        if (qr > mid)  ret += Query(rs,mid+1,r,ql,qr);
        return ret;
    }
    #undef ls
    #undef rs
}sgt;
int main() {
    cin >> n >> m;
    rep(i,1,n) scanf("%lld",&a[i]);
    sgt.BuildTree(1,1,n);
    rep(i,1,m) {
        int ty; scanf("%d",&ty);
        int x,y;ll k;
        if (ty == 1) {
            scanf("%d%d%lld",&x,&y,&k);
            sgt.Change(1,1,n,x,y,k);
        }
        else {
            scanf("%d%d",&x,&y);
            printf("%lld\n",sgt.Query(1,1,n,x,y));
        }
    }
    return 0;
}
```

- luogu3224 并查集-合并

```
#include <bits/stdc++.h>
#define rep(i,a,b) for (int i=a;i<=b;++i)
using namespace std;

const int N = 1e5+5;
const int NN = 20*3*N;

int n,m,tot,sz,q;
int num[NN],lson[NN],rson[NN];
int p[N],fa[N],root[N];
void Update(int id) { num[id] = num[lson[id]] + num[rson[id]]; }
void Build(int &id,int l,int r,int x) {
    id = ++tot;
    if (l == r) {
        num[id]++; return;
    }
    int mid = (l+r) >> 1;
    if (x <= mid) Build(lson[id],l,mid,x);
    else Build(rson[id],mid+1,r,x);
    Update(id);
}
int Merge(int x,int y) {
    if (!x) return y; if (!y) return x;
    lson[x] = Merge(lson[x],lson[y]);
    rson[x] = Merge(rson[x],rson[y]);
    Update(x);
```

```
    return x;
}
int Find(int x) {
    if (x == fa[x]) return x;
    return fa[x] = Find(fa[x]);
}
void Union(int x,int y) {
    x = Find(x); y = Find(y);
    if (x != y) {
        fa[y] = x;
        root[x] = Merge(x,y);
    }
}
int Query(int id,int l,int r,int k) {
    if (!id) return -1;
    if (l == r) {
        if (num[id] >= k) return l;
        return -1;
    }
    int mid = (l + r) >> 1;
    if (k > num[mid]) return Query(rson[id],mid+1,r,k-num[lson[id]]);
    return Query(lson[id],l,mid,k);
}
int main() {
    scanf("%d%d",&n,&m);
    rep(i,1,n) scanf("%d",&p[i]),sz = max(sz,p[i]),fa[i] = i;
    rep(i,1,n) Build(root[i],1,sz,p[i]);
    rep(i,1,m) {
        int x,y; scanf("%d%d",&x,&y);
        Union(x,y);
    }
    scanf("%d",&q);
    rep(i,1,q) {
        int x,y; char ch[1]; scanf("%s%d%d",ch,&x,&y);
        if (ch[0] == 'Q') printf("%d\n",Query(root[Find(x)],1,sz,y));
        else Union(x,y);
    }
    return 0;
}
```

- 线段树合并

luogu3605

```

#include <bits/stdc++.h>
#define rep(i,a,b) for (int i=a;i<=b;++i)
using namespace std;

const int N = 1e5+5;
vector<int> g[N];
int p[N],tt[N],root[N],ans[N];
int lson[20*N],rson[20*N],num[20*N];
int n,tot;
void Update(int id) {
    num[id] = num[lson[id]] + num[rson[id]];
}
void Build(int &id,int l,int r,int x) {
    id = ++tot;
    if (l == r) {
        num[id]++; return;
    }
    int mid = (l+r) >> 1;
    if (x <= mid) Build(lson[id],l,mid,x);
    else Build(rson[id],mid+1,r,x);
    Update(id);
}
int Merge(int x,int y) {
    if (!x) return y; if (!y) return x;
    lson[x] = Merge(lson[x],lson[y]);
    rson[x] = Merge(rson[x],rson[y]);
    Update(x);
    return x;
}
int Query(int id,int l,int r,int x) {
    if (!id) return 0;
    if (l >= x) return num[id];
    int mid = (l + r) >> 1;
```

```
    if (x > mid) return Query(rson[id],mid+1,r,x);
    else return Query(lson[id],l,mid,x) + Query(rson[id],mid+1,r,x);
}
void dfs(int u,int fa) {
    for (auto v:g[u]) if (v!=fa) {
        dfs(v,u);
        root[u] = Merge(root[u],root[v]);
    }
    ans[u] = Query(root[u],1,n,p[u]+1);
}
int main() {
    scanf("%d",&n);
    rep(i,1,n) {
        scanf("%d",&p[i]);
        tt[i] = p[i];
    }
    sort(tt+1,tt+n+1);
    int sz = unique(tt+1,tt+n+1)-tt-1;
    rep(i,1,n) p[i] = lower_bound(tt+1,tt+sz+1,p[i])-tt;
    rep(i,2,n) {
        int x; scanf("%d",&x);
        g[x].push_back(i); g[i].push_back(x);
    }
    rep(i,1,n) Build(root[i],1,n,p[i]);
    dfs(1,0);
    rep(i,1,n) printf("%d\n",ans[i]);
    return 0;
}
```

计算几何

- 三角形面积并

```
#include<algorithm>
#include<iostream>
#include<cstring>
#include<cstdio>
#include<cmath>
using namespace std;
const int MAXN=105;
const double eps=1e-12;
const double INF=1e9;

int dcmp(double x)
{
    if(x<=eps&&x>=-eps) return 0;
    return (x>0)?1:-1;
}
struct Vector
{
    double x,y;
    Vector(double X=0,double Y=0)
    {
        x=X,y=Y;
    }
    bool operator < (const Vector &a)const
    {
        return x<a.x||(x==a.x&&y<a.y);
    }
    void read(){scanf("%lf%lf",&x,&y);}
};
typedef Vector Point;
struct Line
{
    Point p,q;
    Line(Point P=Point(0,0),Point Q=Point(0,0))
    {
        p=P,q=Q;
    }
};
Vector operator + (Vector a,Vector b) {return Vector(a.x+b.x,a.y+b.y);}
Vector operator - (Vector a,Vector b) {return Vector(a.x-b.x,a.y-b.y);}
Vector operator * (Vector a,double b) {return Vector(a.x*b,a.y*b);}

int n,LSH;
double ans;
double lsh[MAXN*MAXN*10];
Point seg[MAXN];
```

```
Line line[MAXN][4];

double Cross(Vector a,Vector b)
{
    return a.x*b.y-a.y*b.x;
}
bool ins(Point A,Point B,Point C,Point D)
{
    Vector v,w,u;
    v=A-C,w=C-D,u=B-D;
    if(dcmp(Cross(v,w))==dcmp(Cross(u,w))) return 0;
    v=C-A,w=B-A,u=D-A;
    if(dcmp(Cross(v,w))==dcmp(Cross(u,w))) return 0;
    return 1;
}
Point GLI(Point P,Vector v,Point Q,Vector w)
{
    Vector u=P-Q;
    double t=Cross(w,u)/Cross(v,w);
    return P+v*t;
}
double Plus(double x)
{
    int cnt=0;
    for(int i=1;i<=n;i++)
    {
        if (dcmp(line[i][1].p.x-line[i][1].q.x)==0&&dcmp(x==line[i][1].p.x))
            continue;
        double Min=INF,Max=-INF;
        for(int j=1;j<=3;j++)
        {
            if(x<line[i][j].p.x||x>line[i][j].q.x) continue;
            if(dcmp(line[i][j].p.x-line[i][j].q.x)==0) continue;
            Point P=GLI(line[i][j].p,line[i][j].q-line[i][j].p,Point(x,-INF),Vector(0,INF));
            Min=min(Min,P.y),Max=max(Max,P.y);
        }
        if(Max-Min>eps) seg[++cnt]=Point(Min,Max);
    }
    sort(seg+1,seg+cnt+1);
    if(!cnt) return 0.0;
    double l=seg[1].x,r=seg[1].y,sum=0.0;
    for(int i=2;i<=cnt;i++)
    {
        if(seg[i].x-r>eps) sum+=r-l,l=seg[i].x,r=seg[i].y;
        else r=max(r,seg[i].y);
    }
    sum+=r-l;
    return sum;
}
double Minus(double x)
{
    int cnt=0;
    for(int i=1;i<=n;i++)
    {
        if(dcmp(line[i][2].p.x-line[i][2].q.x)==0&&dcmp(x==line[i][2].p.x))
            continue;
        double Min=INF,Max=-INF;
        for(int j=1;j<=3;j++)
        {
            if(x<line[i][j].p.x||x>line[i][j].q.x) continue;
            if(dcmp(line[i][j].p.x-line[i][j].q.x)==0) continue;
            Point P=GLI(line[i][j].p,line[i][j].q-line[i][j].p,Point(x,-INF),Vector(0,INF));
            Min=min(Min,P.y),Max=max(Max,P.y);
        }
        if(Max-Min>eps) seg[++cnt]=Point(Min,Max);
    }
    sort(seg+1,seg+cnt+1);
    if(!cnt) return 0.0;
    double l=seg[1].x,r=seg[1].y,sum=0.0;
    for(int i=2;i<=cnt;i++)
    {
        if(seg[i].x-r>eps) sum+=r-l,l=seg[i].x,r=seg[i].y;
        else r=max(r,seg[i].y);
    }
    sum+=r-l;
    return sum;
}
int main()
{
```

```
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
    {
        Point A,B,C;A.read(),B.read(),C.read();
        if(A.x>B.x) swap(A.x,B.x),swap(A.y,B.y);
        if(B.x>C.x) swap(B.x,C.x),swap(B.y,C.y);
        if(A.x>B.x) swap(A.x,B.x),swap(A.y,B.y);
        lsh[++LSH]=A.x,lsh[++LSH]=B.x,lsh[++LSH]=C.x;
        line[i][1]=Line(A,B),line[i][2]=Line(B,C);line[i][3]=Line(A,C);
    }
    for(int i=1;i<=n;i++)
        for(int j=1;j<=3;j++)
            for(int k=i+1;k<=n;k++)
                for(int l=1;l<=3;l++)
                {
                    Point A=line[i][j].p,B=line[i][j].q,C=line[k][l].p,D=line[k][l].q;
                    if(ins(A,B,C,D))
                    {
                        Point q=GLI(A,B-A,C,D-C);
                        lsh[++LSH]=q.x;
                    }
                }
    sort(lsh+1,lsh+LSH+1);LSH=unique(lsh+1,lsh+LSH+1)-lsh-1;
    double last=0.0,now;
    for(int i=1;i<=LSH;i++)
    {
        now=Plus(lsh[i]);
        if(i>1) ans+=(now+last)*(lsh[i]-lsh[i-1])/2.0;
        last=Minus(lsh[i]);
    }
    printf("%.2lf\n",ans-eps);
}

```

- 计算几何

二维几何：点与向量

```
#define y1 yy1
#define nxt(i) ((i + 1) % s.size())
typedef double LD;
const LD PI = 3.14159265358979323846;
const LD eps = 1E-10;
int sgn(LD x) { return fabs(x) < eps ? 0 : (x > 0 ? 1 : -1); }
struct L;
struct P;
typedef P V;
struct P {
    LD x, y;
    explicit P(LD x = 0, LD y = 0): x(x), y(y) {}
    explicit P(const L& l);
};
struct L {
    P s, t;
    L() {}
    L(P s, P t): s(s), t(t) {}
};

P operator + (const P& a, const P& b) { return P(a.x + b.x, a.y + b.y); }
P operator - (const P& a, const P& b) { return P(a.x - b.x, a.y - b.y); }
P operator * (const P& a, LD k) { return P(a.x * k, a.y * k); }
P operator / (const P& a, LD k) { return P(a.x / k, a.y / k); }
inline bool operator < (const P& a, const P& b) {
    return sgn(a.x - b.x) < 0 || (sgn(a.x - b.x) == 0 && sgn(a.y - b.y) < 0);
}
bool operator == (const P& a, const P& b) { return !sgn(a.x - b.x) && !sgn(a.y - b.y); }
P::P(const L& l) { *this = l.t - l.s; }
ostream &operator << (ostream &os, const P &p) {
    return (os << "(" << p.x << "," << p.y << ")");
}
istream &operator >> (istream &is, P &p) {
    return (is >> p.x >> p.y);
}

LD dist(const P& p) { return sqrt(p.x * p.x + p.y * p.y); }
LD dot(const V& a, const V& b) { return a.x * b.x + a.y * b.y; }
LD det(const V& a, const V& b) { return a.x * b.y - a.y * b.x; }
LD cross(const P& s, const P& t, const P& o = P()) { return det(s - o, t - o); }
```

```
// ---------------------------------------------
```

象限

```
// 象限
int quad(P p) {
    int x = sgn(p.x), y = sgn(p.y);
    if (x > 0 && y >= 0) return 1;
    if (x <= 0 && y > 0) return 2;
    if (x < 0 && y <= 0) return 3;
    if (x >= 0 && y < 0) return 4;
    assert(0);
}

// 仅适用于参照点在所有点一侧的情况
struct cmp_angle {
    P p;
    bool operator () (const P& a, const P& b) {
//        int qa = quad(a - p), qb = quad(b - p);
//        if (qa != qb) return qa < qb;
        int d = sgn(cross(a, b, p));
        if (d) return d > 0;
        return dist(a - p) < dist(b - p);
    }
};
```

线

```
// 是否平行
bool parallel(const L& a, const L& b) {
    return !sgn(det(P(a), P(b)));
}
// 直线是否相等
bool l_eq(const L& a, const L& b) {
    return parallel(a, b) && parallel(L(a.s, b.t), L(b.s, a.t));
}
// 逆时针旋转 r 弧度
P rotation(const P& p, const LD& r) { return P(p.x * cos(r) - p.y * sin(r), p.x * sin(r) + p.y * cos(r)); }
P RotateCCW90(const P& p) { return P(-p.y, p.x); }
P RotateCW90(const P& p) { return P(p.y, -p.x); }
// 单位法向量
V normal(const V& v) { return V(-v.y, v.x) / dist(v); }
```

点与线

```
// 点在线段上  <= 0包含端点 < 0 则不包含
bool p_on_seg(const P& p, const L& seg) {
    P a = seg.s, b = seg.t;
    return !sgn(det(p - a, b - a)) && sgn(dot(p - a, p - b)) <= 0;
}
// 点到直线距离
LD dist_to_line(const P& p, const L& l) {
    return fabs(cross(l.s, l.t, p)) / dist(l);
}
// 点到线段距离
LD dist_to_seg(const P& p, const L& l) {
    if (l.s == l.t) return dist(p - l);
    V vs = p - l.s, vt = p - l.t;
    if (sgn(dot(l, vs)) < 0) return dist(vs);
    else if (sgn(dot(l, vt)) > 0) return dist(vt);
    else return dist_to_line(p, l);
}
```

线与线

```
// 求直线交 需要事先保证有界
P l_intersection(const L& a, const L& b) {
    LD s1 = det(P(a), b.s - a.s), s2 = det(P(a), b.t - a.s);
    return (b.s * s2 - b.t * s1) / (s2 - s1);
}
// 向量夹角的弧度
LD angle(const V& a, const V& b) {
    LD r = asin(fabs(det(a, b)) / dist(a) / dist(b));
    if (sgn(dot(a, b)) < 0) r = PI - r;
    return r;
```

```
}
// 线段和直线是否有交   1 = 规范, 2 = 不规范
int s_l_cross(const L& seg, const L& line) {
    int d1 = sgn(cross(line.s, line.t, seg.s));
    int d2 = sgn(cross(line.s, line.t, seg.t));
    if ((d1 ^ d2) == -2) return 1; // proper
    if (d1 == 0 || d2 == 0) return 2;
    return 0;
}
// 线段的交   1 = 规范, 2 = 不规范
int s_cross(const L& a, const L& b, P& p) {
    int d1 = sgn(cross(a.t, b.s, a.s)), d2 = sgn(cross(a.t, b.t, a.s));
    int d3 = sgn(cross(b.t, a.s, b.s)), d4 = sgn(cross(b.t, a.t, b.s));
    if ((d1 ^ d2) == -2 && (d3 ^ d4) == -2) { p = l_intersection(a, b); return 1; }
    if (!d1 && p_on_seg(b.s, a)) { p = b.s; return 2; }
    if (!d2 && p_on_seg(b.t, a)) { p = b.t; return 2; }
    if (!d3 && p_on_seg(a.s, b)) { p = a.s; return 2; }
    if (!d4 && p_on_seg(a.t, b)) { p = a.t; return 2; }
    return 0;
}
```

多边形

面积、凸包

```
typedef vector<P> S;

// 点是否在多边形中 0 = 在外部 1 = 在内部 -1 = 在边界上
int inside(const S& s, const P& p) {
    int cnt = 0;
    FOR (i, 0, s.size()) {
        P a = s[i], b = s[nxt(i)];
        if (p_on_seg(p, L(a, b))) return -1;
        if (sgn(a.y - b.y) <= 0) swap(a, b);
        if (sgn(p.y - a.y) > 0) continue;
        if (sgn(p.y - b.y) <= 0) continue;
        cnt += sgn(cross(b, a, p)) > 0;
    }
    return bool(cnt & 1);
}
// 多边形面积, 有向面积可能为负
LD polygon_area(const S& s) {
    LD ret = 0;
    FOR (i, 1, (LL)s.size() - 1)
        ret += cross(s[i], s[i + 1], s[0]);
    return ret / 2;
}
// 构建凸包 点不可以重复 < 0 边上可以有点,  <= 0 则不能
// 会改变输入点的顺序
const int MAX_N = 1000;
S convex_hull(S& s) {
//    assert(s.size() >= 3);
    sort(s.begin(), s.end());
    S ret(MAX_N * 2);
    int sz = 0;
    FOR (i, 0, s.size()) {
        while (sz > 1 && sgn(cross(ret[sz - 1], s[i], ret[sz - 2])) < 0) --sz;
        ret[sz++] = s[i];
    }
    int k = sz;
    FORD (i, (LL)s.size() - 2, -1) {
        while (sz > k && sgn(cross(ret[sz - 1], s[i], ret[sz - 2])) < 0) --sz;
        ret[sz++] = s[i];
    }
    ret.resize(sz - (s.size() > 1));
    return ret;
}

P ComputeCentroid(const vector<P> &p) {
    P c(0, 0);
    LD scale = 6.0 * polygon_area(p);
    for (unsigned i = 0; i < p.size(); i++) {
        unsigned j = (i + 1) % p.size();
        c = c + (p[i] + p[j]) * (p[i].x * p[j].y - p[j].x * p[i].y);
    }
    return c / scale;
}
```

旋转卡壳

```
LD rotatingCalipers(vector<P>& qs) {
    int n = qs.size();
    if (n == 2)
        return dist(qs[0] - qs[1]);
    int i = 0, j = 0;
    FOR (k, 0, n) {
        if (!(qs[i] < qs[k])) i = k;
        if (qs[j] < qs[k]) j = k;
    }
    LD res = 0;
    int si = i, sj = j;
    while (i != sj || j != si) {
        res = max(res, dist(qs[i] - qs[j]));
        if (sgn(cross(qs[(i+1)%n] - qs[i], qs[(j+1)%n] - qs[j])) < 0)
            i = (i + 1) % n;
        else j = (j + 1) % n;
    }
    return res;
}

int main() {
    int n;
    while (cin >> n) {
        S v(n);
        FOR (i, 0, n) cin >> v[i].x >> v[i].y;
        convex_hull(v);
        printf("%.0f\n", rotatingCalipers(v));
    }
}
```

半平面交

```
struct LV {
    P p, v; LD ang;
    LV() {}
    LV(P s, P t): p(s), v(t - s) { ang = atan2(v.y, v.x); }
};  // 另一种向量表示

bool operator < (const LV &a, const LV& b) { return a.ang < b.ang; }
bool on_left(const LV& l, const P& p) { return sgn(cross(l.v, p - l.p)) >= 0; }
P l_intersection(const LV& a, const LV& b) {
    P u = a.p - b.p; LD t = cross(b.v, u) / cross(a.v, b.v);
    return a.p + a.v * t;
}

S half_plane_intersection(vector<LV>& L) {
    int n = L.size(), fi, la;
    sort(L.begin(), L.end());
    vector<P> p(n); vector<LV> q(n);
    q[fi = la = 0] = L[0];
    FOR (i, 1, n) {
        while (fi < la && !on_left(L[i], p[la - 1])) la--;
        while (fi < la && !on_left(L[i], p[fi])) fi++;
        q[++la] = L[i];
        if (sgn(cross(q[la].v, q[la - 1].v)) == 0) {
            la--;
            if (on_left(q[la], L[i].p)) q[la] = L[i];
        }
        if (fi < la) p[la - 1] = l_intersection(q[la - 1], q[la]);
    }
    while (fi < la && !on_left(q[fi], p[la - 1])) la--;
    if (la - fi <= 1) return vector<P>();
    p[la] = l_intersection(q[la], q[fi]);
    return vector<P>(p.begin() + fi, p.begin() + la + 1);
}

S convex_intersection(const vector<P> &v1, const vector<P> &v2) {
    vector<LV> h; int n = v1.size(), m = v2.size();
    FOR (i, 0, n) h.push_back(LV(v1[i], v1[(i + 1) % n]));
    FOR (i, 0, m) h.push_back(LV(v2[i], v2[(i + 1) % m]));
    return half_plane_intersection(h);
}
```

圆

```
struct C {
    P p; LD r;
    C(LD x = 0, LD y = 0, LD r = 0): p(x, y), r(r) {}
    C(P p, LD r): p(p), r(r) {}
};
```

三点求圆心

```
P compute_circle_center(P a, P b, P c) {
    b = (a + b) / 2;
    c = (a + c) / 2;
    return l_intersection({b, b + RotateCW90(a - b)}, {c , c + RotateCW90(a - c)});
}
```

圆线交点、圆圆交点

- 圆和线的交点关于圆心是顺时针的

```
vector<P> c_l_intersection(const L& l, const C& c) {
    vector<P> ret;
    P b(l), a = l.s - c.p;
    LD x = dot(b, b), y = dot(a, b), z = dot(a, a) - c.r * c.r;
    LD D = y * y - x * z;
    if (sgn(D) < 0) return ret;
    ret.push_back(c.p + a + b * (-y + sqrt(D + eps)) / x);
    if (sgn(D) > 0) ret.push_back(c.p + a + b * (-y - sqrt(D)) / x);
    return ret;
}

vector<P> c_c_intersection(C a, C b) {
    vector<P> ret;
    LD d = dist(a.p - b.p);
    if (sgn(d) == 0 || sgn(d - (a.r + b.r)) > 0 || sgn(d + min(a.r, b.r) - max(a.r, b.r)) < 0)
        return ret;
    LD x = (d * d - b.r * b.r + a.r * a.r) / (2 * d);
    LD y = sqrt(a.r * a.r - x * x);
    P v = (b.p - a.p) / d;
    ret.push_back(a.p + v * x + RotateCCW90(v) * y);
    if (sgn(y) > 0) ret.push_back(a.p + v * x - RotateCCW90(v) * y);
    return ret;
}
```

圆圆位置关系

```
// 1:内含 2:内切 3:相交 4:外切 5:相离
int c_c_relation(const C& a, const C& v) {
    LD d = dist(a.p - v.p);
    if (sgn(d - a.r - v.r) > 0) return 5;
    if (sgn(d - a.r - v.r) == 0) return 4;
    LD l = fabs(a.r - v.r);
    if (sgn(d - l) > 0) return 3;
    if (sgn(d - l) == 0) return 2;
    if (sgn(d - l) < 0) return 1;
}
```

圆与多边形交

- HDU 5130
- 注意顺时针逆时针（可能要取绝对值）

```
LD sector_area(const P& a, const P& b, LD r) {
    LD th = atan2(a.y, a.x) - atan2(b.y, b.x);
    while (th <= 0) th += 2 * PI;
    while (th > 2 * PI) th -= 2 * PI;
    th = min(th, 2 * PI - th);
    return r * r * th / 2;
}

LD c_tri_area(P a, P b, P center, LD r) {
    a = a - center; b = b - center;
    int ina = sgn(dist(a) - r) < 0, inb = sgn(dist(b) - r) < 0;
    // dbg(a, b, ina, inb);
    if (ina && inb) {
        return fabs(cross(a, b)) / 2;
    } else {
```

```
        auto p = c_l_intersection(L(a, b), C(0, 0, r));
        if (ina ^ inb) {
            auto cr = p_on_seg(p[0], L(a, b)) ? p[0] : p[1];
            if (ina) return sector_area(b, cr, r) + fabs(cross(a, cr)) / 2;
            else return sector_area(a, cr, r) + fabs(cross(b, cr)) / 2;
        } else {
            if ((int) p.size() == 2 && p_on_seg(p[0], L(a, b))) {
                if (dist(p[0] - a) > dist(p[1] - a)) swap(p[0], p[1]);
                return sector_area(a, p[0], r) + sector_area(p[1], b, r)
                    + fabs(cross(p[0], p[1])) / 2;
            } else return sector_area(a, b, r);
        }
    }
}

typedef vector<P> S;
LD c_poly_area(S poly, const C& c) {
    LD ret = 0; int n = poly.size();
    FOR (i, 0, n) {
        int t = sgn(cross(poly[i] - c.p, poly[(i + 1) % n] - c.p));
        if (t) ret += t * c_tri_area(poly[i], poly[(i + 1) % n], c.p, c.r);
    }
    return ret;
}
```

圆的离散化、面积并

SPOJ: CIRU, EOJ: 284

- 版本 1：复杂度 $O(n^3 \log n)$。虽然常数小，但还是难以接受。
- 优点？想不出来。
- 原理上是用竖线进行切分，然后对每一个切片分别计算。
- 扫描线部分可以魔改，求各种东西。

```
inline LD rt(LD x) { return sgn(x) == 0 ? 0 : sqrt(x); }
inline LD sq(LD x) { return x * x; }

// 圆弧
// 如果按照 x 离散化，圆弧是 "横着的"
// 记录圆弧的左端点、右端点、中点的坐标，和圆弧所在的圆
// 调用构造要保证 c.x - x.r <= xl < xr <= c.y + x.r
// t = 1 下圆弧 t = -1 上圆弧
struct CV {
    LD yl, yr, ym; C o; int type;
    CV() {}
    CV(LD yl, LD yr, LD ym, C c, int t)
        : yl(yl), yr(yr), ym(ym), type(t), o(c) {}
};

// 辅助函数 求圆上纵坐标
pair<LD, LD> c_point_eval(const C& c, LD x) {
    LD d = fabs(c.p.x - x), h = rt(sq(c.r) - sq(d));
    return {c.p.y - h, c.p.y + h};
}
// 构造上下圆弧
pair<CV, CV> pairwise_curves(const C& c, LD xl, LD xr) {
    LD yl1, yl2, yr1, yr2, ym1, ym2;
    tie(yl1, yl2) = c_point_eval(c, xl);
    tie(ym1, ym2) = c_point_eval(c, (xl + xr) / 2);
    tie(yr1, yr2) = c_point_eval(c, xr);
    return {CV(yl1, yr1, ym1, c, 1), CV(yl2, yr2, ym2, c, -1)};
}

// 离散化之后同一切片内的圆弧应该是不相交的
bool operator < (const CV& a, const CV& b) { return a.ym < b.ym; }
// 计算圆弧和连接圆弧端点的线段构成的封闭图形的面积
LD cv_area(const CV& v, LD xl, LD xr) {
    LD l = rt(sq(xr - xl) + sq(v.yr - v.yl));
    LD d = rt(sq(v.o.r) - sq(l / 2));
    LD ang = atan(l / d / 2);
    return ang * sq(v.o.r) - d * l / 2;
}

LD circle_union(const vector<C>& cs) {
    int n = cs.size();
    vector<LD> xs;
    FOR (i, 0, n) {
        xs.push_back(cs[i].p.x - cs[i].r);
```

```
        xs.push_back(cs[i].p.x);
        xs.push_back(cs[i].p.x + cs[i].r);
        FOR (j, i + 1, n) {
            auto pts = c_c_intersection(cs[i], cs[j]);
            for (auto& p: pts) xs.push_back(p.x);
        }
    }
    sort(xs.begin(), xs.end());
    xs.erase(unique(xs.begin(), xs.end(), [](LD x, LD y) { return sgn(x - y) == 0; }), xs.end());
    LD ans = 0;
    FOR (i, 0, (int) xs.size() - 1) {
        LD xl = xs[i], xr = xs[i + 1];
        vector<CV> intv;
        FOR (k, 0, n) {
            auto& c = cs[k];
            if (sgn(c.p.x - c.r - xl) <= 0 && sgn(c.p.x + c.r - xr) >= 0) {
                auto t = pairwise_curves(c, xl, xr);
                intv.push_back(t.first); intv.push_back(t.second);
            }
        }
        sort(intv.begin(), intv.end());

        vector<LD> areas(intv.size());
        FOR (i, 0, intv.size()) areas[i] = cv_area(intv[i], xl, xr);

        int cc = 0;
        FOR (i, 0, intv.size()) {
            if (cc > 0) {
                ans += (intv[i].yl - intv[i - 1].yl + intv[i].yr - intv[i - 1].yr) * (xr - xl) / 2;
                ans += intv[i - 1].type * areas[i - 1];
                ans -= intv[i].type * areas[i];
            }
            cc += intv[i].type;
        }
    }
    return ans;
}
```

- 版本 2：复杂度 $O(n^2 \log n)$。
- 原理是：认为所求部分是一个奇怪的多边形 + 若干弓形。然后对于每个圆分别求贡献的弓形，并累加多边形有向面积。
- 同样可以魔改扫描线的部分，用于求周长、至少覆盖 k 次等等。
- 内含、内切、同一个圆的情况，通常需要特殊处理。
- 下面的代码是 k 圆覆盖。

```
inline LD angle(const P& p) { return atan2(p.y, p.x); }

// 圆弧上的点
// p 是相对于圆心的坐标
// a 是在圆上的 atan2 [-PI, PI]
struct CP {
    P p; LD a; int t;
    CP() {}
    CP(P p, LD a, int t): p(p), a(a), t(t) {}
};
bool operator < (const CP& u, const CP& v) { return u.a < v.a; }
LD cv_area(LD r, const CP& q1, const CP& q2) {
    return (r * r * (q2.a - q1.a) - cross(q1.p, q2.p)) / 2;
}

LD ans[N];
void circle_union(const vector<C>& cs) {
    int n = cs.size();
    FOR (i, 0, n) {
        // 有相同的圆的话只考虑第一次出现
        bool ok = true;
        FOR (j, 0, i)
            if (sgn(cs[i].r - cs[j].r) == 0 && cs[i].p == cs[j].p) {
                ok = false;
                break;
            }
        if (!ok) continue;
        auto& c = cs[i];
        vector<CP> ev;
        int belong_to = 0;
        P bound = c.p + P(-c.r, 0);
        ev.emplace_back(bound, -PI, 0);
        ev.emplace_back(bound, PI, 0);
        FOR (j, 0, n) {
```

```
            if (i == j) continue;
            if (c_c_relation(c, cs[j]) <= 2) {
                if (sgn(cs[j].r - c.r) >= 0) // 完全被另一个圆包含，等于说叠了一层
                    belong_to++;
                continue;
            }
            auto its = c_c_intersection(c, cs[j]);
            if (its.size() == 2) {
                P p = its[1] - c.p, q = its[0] - c.p;
                LD a = angle(p), b = angle(q);
                if (sgn(a - b) > 0) {
                    ev.emplace_back(p, a, 1);
                    ev.emplace_back(bound, PI, -1);
                    ev.emplace_back(bound, -PI, 1);
                    ev.emplace_back(q, b, -1);
                } else {
                    ev.emplace_back(p, a, 1);
                    ev.emplace_back(q, b, -1);
                }
            }
        }
        sort(ev.begin(), ev.end());
        int cc = ev[0].t;
        FOR (j, 1, ev.size()) {
            int t = cc + belong_to;
            ans[t] += cross(ev[j - 1].p + c.p, ev[j].p + c.p) / 2;
            ans[t] += cv_area(c.r, ev[j - 1], ev[j]);
            cc += ev[j].t;
        }
    }
}
```

最小圆覆盖

- 随机增量。期望复杂度 $O(n)$。

```
P compute_circle_center(P a, P b) { return (a + b) / 2; }
bool p_in_circle(const P& p, const C& c) {
    return sgn(dist(p - c.p) - c.r) <= 0;
}
C min_circle_cover(const vector<P> &in) {
    vector<P> a(in.begin(), in.end());
    dbg(a.size());
    random_shuffle(a.begin(), a.end());
    P c = a[0]; LD r = 0; int n = a.size();
    FOR (i, 1, n) if (!p_in_circle(a[i], {c, r})) {
        c = a[i]; r = 0;
        FOR (j, 0, i) if (!p_in_circle(a[j], {c, r})) {
            c = compute_circle_center(a[i], a[j]);
            r = dist(a[j] - c);
            FOR (k, 0, j) if (!p_in_circle(a[k], {c, r})) {
                c = compute_circle_center(a[i], a[j], a[k]);
                r = dist(a[k] - c);
            }
        }
    }
    return {c, r};
}
```

圆的反演

```
C inv(C c, const P& o) {
    LD d = dist(c.p - o);
    assert(sgn(d) != 0);
    LD a = 1 / (d - c.r);
    LD b = 1 / (d + c.r);
    c.r = (a - b) / 2 * R2;
    c.p = o + (c.p - o) * ((a + b) * R2 / 2 / d);
    return c;
}
```

三维计算几何

```
struct P;
struct L;
typedef P V;
```

```

struct P {
    LD x, y, z;
    explicit P(LD x = 0, LD y = 0, LD z = 0): x(x), y(y), z(z) {}
    explicit P(const L& l);
};

struct L {
    P s, t;
    L() {}
    L(P s, P t): s(s), t(t) {}
};

struct F {
    P a, b, c;
    F() {}
    F(P a, P b, P c): a(a), b(b), c(c) {}
};

P operator + (const P& a, const P& b) { return P(a.x + b.x, a.y + b.y, a.z + b.z); }
P operator - (const P& a, const P& b) { return P(a.x - b.x, a.y - b.y, a.z - b.z); }
P operator * (const P& a, LD k) { return P(a.x * k, a.y * k, a.z * k); }
P operator / (const P& a, LD k) { return P(a.x / k, a.y / k, a.z / k); }
inline int operator < (const P& a, const P& b) {
    return sgn(a.x - b.x) < 0 || (sgn(a.x - b.x) == 0 && (sgn(a.y - b.y) < 0 ||
                                  (sgn(a.y - b.y) == 0 && sgn(a.z - b.z) < 0)));
}
bool operator == (const P& a, const P& b) { return !sgn(a.x - b.x) && !sgn(a.y - b.y) && !sgn(a.z - b.z); }
P::P(const L& l) { *this = l.t - l.s; }
ostream &operator << (ostream &os, const P &p) {
    return (os << "(" << p.x << "," << p.y << "," << p.z << ")");
}
istream &operator >> (istream &is, P &p) {
    return (is >> p.x >> p.y >> p.z);
}

// ---------------------------------------------
LD dist2(const P& p) { return p.x * p.x + p.y * p.y + p.z * p.z; }
LD dist(const P& p) { return sqrt(dist2(p)); }
LD dot(const V& a, const V& b) { return a.x * b.x + a.y * b.y + a.z * b.z; }
P cross(const P& v, const P& w) {
    return P(v.y * w.z - v.z * w.y, v.z * w.x - v.x * w.z, v.x * w.y - v.y * w.x);
}
LD mix(const V& a, const V& b, const V& c) { return dot(a, cross(b, c)); }
```

旋转

```
// 逆时针旋转 r 弧度
// axis = 0 绕 x 轴
// axis = 1 绕 y 轴
// axis = 2 绕 z 轴
P rotation(const P& p, const LD& r, int axis = 0) {
    if (axis == 0)
        return P(p.x, p.y * cos(r) - p.z * sin(r), p.y * sin(r) + p.z * cos(r));
    else if (axis == 1)
        return P(p.z * cos(r) - p.x * sin(r), p.y, p.z * sin(r) + p.x * cos(r));
    else if (axis == 2)
        return P(p.x * cos(r) - p.y * sin(r), p.x * sin(r) + p.y * cos(r), p.z);
}
// n 是单位向量 表示旋转轴
// 模板是顺时针的
P rotation(const P& p, const LD& r, const P& n) {
    LD c = cos(r), s = sin(r), x = n.x, y = n.y, z = n.z;
    // dbg(c, s);
    return P((x * x * (1 - c) + c) * p.x + (x * y * (1 - c) + z * s) * p.y + (x * z * (1 - c) - y * s) * p.z,
             (x * y * (1 - c) - z * s) * p.x + (y * y * (1 - c) + c) * p.y + (y * z * (1 - c) + x * s) * p.z,
             (x * z * (1 - c) + y * s) * p.x + (y * z * (1 - c) - x * s) * p.y + (z * z * (1 - c) + c) * p.z);
}
```

线、面

函数相互依赖，所以交织在一起了。

```
// 点在线段上  <= 0包含端点 < 0 则不包含
bool p_on_seg(const P& p, const L& seg) {
    P a = seg.s, b = seg.t;
    return !sgn(dist2(cross(p - a, b - a))) && sgn(dot(p - a, p - b)) <= 0;
```

```
}
// 点到直线距离
LD dist_to_line(const P& p, const L& l) {
    return dist(cross(l.s - p, l.t - p)) / dist(l);
}
// 点到线段距离
LD dist_to_seg(const P& p, const L& l) {
    if (l.s == l.t) return dist(p - l.s);
    V vs = p - l.s, vt = p - l.t;
    if (sgn(dot(l, vs)) < 0) return dist(vs);
    else if (sgn(dot(l, vt)) > 0) return dist(vt);
    else return dist_to_line(p, l);
}

P norm(const F& f) { return cross(f.a - f.b, f.b - f.c); }
int p_on_plane(const F& f, const P& p) { return sgn(dot(norm(f), p - f.a)) == 0; }

// 判两点在线段异侧 点在线段上返回 0 不共面无意义
int opposite_side(const P& u, const P& v, const L& l) {
    return sgn(dot(cross(P(l), u - l.s), cross(P(l), v - l.s))) < 0;
}

bool parallel(const L& a, const L& b) { return !sgn(dist2(cross(P(a), P(b)))); }
// 线段相交
int s_intersect(const L& u, const L& v) {
    return p_on_plane(F(u.s, u.t, v.s), v.t) &&
           opposite_side(u.s, u.t, v) &&
           opposite_side(v.s, v.t, u);
}
```

凸包

增量法。先将所有的点打乱顺序，然后选择四个不共面的点组成一个四面体，如果找不到说明凸包不存在。然后遍历剩余的点，不断更新凸包。对遍历到的点做如下处理。

1. 如果点在凸包内，则不更新。
2. 如果点在凸包外，那么找到所有原凸包上所有分隔了对于这个点可见面和不可见面的边，以这样的边的两个点和新的点创建新的面加入凸包中。

```

struct FT {
    int a, b, c;
    FT() { }
    FT(int a, int b, int c) : a(a), b(b), c(c) { }
};

bool p_on_line(const P& p, const L& l) {
    return !sgn(dist2(cross(p - l.s, P(l))));
}

vector<F> convex_hull(vector<P> &p) {
    sort(p.begin(), p.end());
    p.erase(unique(p.begin(), p.end()), p.end());
    random_shuffle(p.begin(), p.end());
    vector<FT> face;
    FOR (i, 2, p.size()) {
        if (p_on_line(p[i], L(p[0], p[1]))) continue;
        swap(p[i], p[2]);
        FOR (j, i + 1, p.size())
            if (sgn(mix(p[1] - p[0], p[2] - p[1], p[j] - p[0]))) {
                swap(p[j], p[3]);
                face.emplace_back(0, 1, 2);
                face.emplace_back(0, 2, 1);
                goto found;
            }
    }
found:
    vector<vector<int>> mk(p.size(), vector<int>(p.size()));
    FOR (v, 3, p.size()) {
        vector<FT> tmp;
        FOR (i, 0, face.size()) {
            int a = face[i].a, b = face[i].b, c = face[i].c;
            if (sgn(mix(p[a] - p[v], p[b] - p[v], p[c] - p[v])) < 0) {
                mk[a][b] = mk[b][a] = v;
                mk[b][c] = mk[c][b] = v;
                mk[c][a] = mk[a][c] = v;
            } else tmp.push_back(face[i]);
        }
        face = tmp;
```

```
        FOR (i, 0, tmp.size()) {
            int a = face[i].a, b = face[i].b, c = face[i].c;
            if (mk[a][b] == v) face.emplace_back(b, a, v);
            if (mk[b][c] == v) face.emplace_back(c, b, v);
            if (mk[c][a] == v) face.emplace_back(a, c, v);
        }
    }
    vector<F> out;
    FOR (i, 0, face.size())
        out.emplace_back(p[face[i].a], p[face[i].b], p[face[i].c]);
    return out;
}
```

数学

- simpson

```
double simpson(double a,double b)
{
    double c=a+(b-a)/2;
    return (F(a)+4*F(c)+F(b))*(b-a)/6;
}
double asr(double a,double b,double eps,double A)
{
    double c=a+(b-a)/2;
    double L=simpson(a,c),R=simpson(c,b);
    if(fabs(L+R-A)<=15*eps) return L+R+(L+R-A)/15.0;
    return asr(a,c,eps/2,L)+asr(c,b,eps/2,R);
}
double asr(double a,double b,double eps)
{
    return asr(a,b,eps,simpson(a,b));
}

```

- 矩阵

```
struct Matrix
{
    int n;
    ll a[MAXN][MAXN];
    Matrix(int _n=MAXN)
    {
        n=_n;
        for(int i=0;i<n;i++)
            for(int j=0;j<n;j++)
                a[i][j]=0;
    }
    Matrix operator * (const Matrix &B)const
    {
        Matrix C;C.n=n;
        for(int i=0;i<n;i++)
            for(int k=0;k<n;k++)
                for(int j=0;j<n;j++)
                    C.a[i][j]=(C.a[i][j]+(a[i][k]*B.a[k][j])%MOD)%MOD;
        return C;
    }
    Matrix operator + (const Matrix &B)const
    {
        Matrix C;C.n=n;
        for(int i=0;i<n;i++)
            for(int j=0;j<n;j++)
                C.a[i][j]=(a[i][j]+B.a[i][j])%MOD;
        return C;
    }
    Matrix operator % (const ll &t)const
    {
        Matrix A=(*this);
        for(int i=0;i<n;i++)
        {
            for(int j=0;j<n;j++)
            {
                A.a[i][j]%=MOD;
            }
        }
        return A;
    }
```

```
    Matrix operator ^ (const ll &t)const
    {
        Matrix A=(*this),res;
        ll p=t;
        res.n=n;
        for(int i=0;i<n;i++)
            res.a[i][i]=1;
        while(p)
        {
            if(p&1)res=res*A;
            A=A*A;
            p>>=1;
        }
        return res;
    }
    void debug()
    {
        for(int i=0;i<n;i++,printf("\n"))
            for(int j=0;j<n;j++)
                printf("%lld ",a[i][j]);
    }
    void reset()
    {
        for(int i=0;i<n;i++)
            a[i][i]=0;
    }
    bool judge()
    {
        for(int i=0;i<n;i++)
        {
            for(int j=0;j<n;j++)
            {
                if((a[i][j]&&(i!=j))||((!a[i][j])&&(i==j)))
                    return false;
            }
        }
        return true;
    }
};

```

- 大数区间素数筛

```
const int MAXN=100010;
int prime[MAXN+1];
void getPrime() {
    memset(prime,0,sizeof(prime));
    for(int i=2;i<=MAXN;i++) {
        if(!prime[i])prime[++prime[0]]=i;
        for(int j=1;j<=prime[0]&&prime[j]<=MAXN/i;j++) {
            prime[prime[j]*i]=1;
            if(i%prime[j]==0)break;
        }
    }
}
bool notprime[1000010];
int prime2[1000010];
void getPrime2(int L,int R) {
    memset(notprime,false,sizeof(notprime));
    if(L<2)L=2;
    for(int i=1;i<=prime[0]&&(long long)prime[i]*prime[i]<=R;i++)
    {
        int s=L/prime[i]+(L%prime[i]>0);
        if(s==1)s=2;
        for(int j=s;(long long)j*prime[i]<=R;j++)
            if((long long)j*prime[i]>=L)
                notprime[j*prime[i]-L]=true;
    }
    prime2[0]=0;
    for(int i=0;i<=R-L;i++)
        if(!notprime[i])
            prime2[++prime2[0]]=i+L;
}
int main() {
    getPrime();
    int L,U; // R-L <= 1e6
    while(scanf("%d%d",&L,&U)==2) {
        getPrime2(L,U);
```

```
        if(prime2[0]<2)printf("There are no adjacent primes.\n");
        else {
            int x1=0,x2=100000000,y1=0,y2=0;
            for(int i=1;i<prime2[0];i++) {
                if(prime2[i+1]-prime2[i]<x2-x1)
                {
                    x1=prime2[i];
                    x2=prime2[i+1];
                }
                if(prime2[i+1]-prime2[i]>y2-y1)
                {
                    y1=prime2[i];
                    y2=prime2[i+1];
                }
            }
            printf("%d,%d are closest, %d,%d are most distant.\n",x1,x2,y1,y2);
        }
    }
}

```

- 单纯形

- 要求有基本解，也就是 x 为零向量可行
- v 要初始化为 0，n 表示向量长度，m 表示约束个数

```
// min{ b x } / max { c x }
// A x >= c   / A x <= b
// x >= 0
namespace lp {
    int n, m;
    double a[M][N], b[M], c[N], v;

    void pivot(int l, int e) {
        b[l] /= a[l][e];
        FOR (j, 0, n) if (j != e) a[l][j] /= a[l][e];
        a[l][e] = 1 / a[l][e];

        FOR (i, 0, m)
            if (i != l && fabs(a[i][e]) > 0) {
                b[i] -= a[i][e] * b[l];
                FOR (j, 0, n)
                    if (j != e) a[i][j] -= a[i][e] * a[l][j];
                a[i][e] = -a[i][e] * a[l][e];
            }
        v += c[e] * b[l];
        FOR (j, 0, n) if (j != e) c[j] -= c[e] * a[l][j];
        c[e] = -c[e] * a[l][e];
    }
    double simplex() {
        while (1) {
            v = 0;
            int e = -1, l = -1;
            FOR (i, 0, n) if (c[i] > eps) { e = i; break; }
            if (e == -1) return v;
            double t = INF;
            FOR (i, 0, m)
                if (a[i][e] > eps && t > b[i] / a[i][e]) {
                    t = b[i] / a[i][e];
                    l = i;
                }
            if (l == -1) return INF;
            pivot(l, e);
        }
    }
}
```

- 类欧几里得
- $m = \lfloor \frac{an+b}{c} \rfloor$.
- $f(a,b,c,n) = \sum_{i=0}^{n} \lfloor \frac{ai+b}{c} \rfloor$: 当 $a \geq c$ or $b \geq c$ 时，$f(a,b,c,n) = (\frac{a}{c})n(n+1)/2 + (\frac{b}{c})(n+1) + f(a \bmod c, b \bmod c, c, n)$；否则 $f(a,b,c,n) = nm - f(c, c-b-1, a, m-1)$。
- $g(a,b,c,n) = \sum_{i=0}^{n} i\lfloor \frac{ai+b}{c} \rfloor$: 当 $a \geq c$ or $b \geq c$ 时，$g(a,b,c,n) = (\frac{a}{c})n(n+1)(2n+1)/6 + (\frac{b}{c})n(n+1)/2 + g(a \bmod c, b \bmod c, c, n)$；否则 $g(a,b,c,n) = \frac{1}{2}(n(n+1)m - f(c, c-b-1, a, m-1) - h(c, c-b-1, a, m-1))$。
- $h(a,b,c,n) = \sum_{i=0}^{n} \lfloor \frac{ai+b}{c} \rfloor^2$: 当 $a \geq c$ or $b \geq c$ 时，$h(a,b,c,n) = (\frac{a}{c})^2 n(n+1)(2n+1)/6 + (\frac{b}{c})^2(n+1) + (\frac{a}{c})(\frac{b}{c})n(n+1) + h(a \bmod c, b \bmod c, c, n) + 2(\frac{a}{c})g(a \bmod c, b \bmod c, c, n) + 2(\frac{b}{c}$；否则 $h(a,b,c,n) = nm(m+1) - 2g(c, c-b-1, a, m-1) - 2f(c, c-b-1, a, m-1) - f(a,b,c,n)$

- 特殊性质

一些数论公式

- 当 $x \geq \phi(p)$ 时有 $a^x \equiv a^{x \bmod \phi(p)+\phi(p)} \pmod p$
- $\mu^2(n)=\sum_{d^2|n}\mu(d)$
- $\sum_{d|n}\varphi(d)=n$
- $\sum_{d|n}2^{\omega(d)}=\sigma_0(n^2)$，其中 ω 是不同素因子个数
- $\sum_{d|n}\mu^2(d)=2^{\omega(d)}$

一些数论函数求和的例子

- $\sum_{i=1}^n i[gcd(i,n)=1]=\frac{n\varphi(n)+[n=1]}{2}$
- $\sum_{i=1}^n\sum_{j=1}^m[gcd(i,j)=x]=\sum_d\mu(d)\lfloor\frac{n}{dx}\rfloor\lfloor\frac{m}{dx}\rfloor$
- $\sum_{i=1}^n\sum_{j=1}^m gcd(i,j)=\sum_{i=1}^n\sum_{j=1}^m\sum_{d|gcd(i,j)}\varphi(d)=\sum_d\varphi(d)\lfloor\frac{n}{d}\rfloor\lfloor\frac{m}{d}\rfloor$
- $S(n)=\sum_{i=1}^n\mu(i)=1-\sum_{i=1}^n\sum_{d|i,d<i}\mu(d)\overset{t=\frac{i}{d}}{=}1-\sum_{t=2}^n S(\lfloor\frac{n}{t}\rfloor)$
  - 利用 $[n=1]=\sum_{d|n}\mu(d)$
- $S(n)=\sum_{i=1}^n\varphi(i)=\sum_{i=1}^n i-\sum_{i=1}^n\sum_{d|i,d<i}\varphi(i)\overset{t=\frac{i}{d}}{=}\frac{i(i+1)}{2}-\sum_{t=2}^n S(\frac{n}{t})$
  - 利用 $n=\sum_{d|n}\varphi(d)$
- $\sum_{i=1}^n\mu^2(i)=\sum_{i=1}^n\sum_{d^2|n}\mu(d)=\sum_{d=1}^{\lfloor\sqrt{n}\rfloor}\mu(d)\lfloor\frac{n}{d^2}\rfloor$
- $$\begin{aligned}\sum_{i=1}^n\sum_{j=1}^n gcd^2(i,j)&=\sum_d d^2\sum_t\mu(t)\lfloor\frac{n}{dt}\rfloor^2\\&\overset{x=dt}{=}\sum_x\lfloor\frac{n}{x}\rfloor^2\sum_{d|x}d^2\mu(\frac{x}{d})\end{aligned}$$
- $\sum_{i=1}^n\varphi(i)=\frac{1}{2}\sum_{i=1}^n\sum_{j=1}^n[i\perp j]-1=\frac{1}{2}\sum_{i=1}^n\mu(i)\cdot\lfloor\frac{n}{i}\rfloor^2-1$

斐波那契数列性质

- $F_{a+b}=F_{a-1}\cdot F_b+F_a\cdot F_{b+1}$
- $F_1+F_3+\cdots+F_{2n-1}=F_{2n},F_2+F_4+\cdots+F_{2n}=F_{2n+1}-1$
- $\sum_{i=1}^n F_i=F_{n+2}-1$
- $\sum_{i=1}^n F_i^2=F_n\cdot F_{n+1}$
- $F_n^2=(-1)^{n-1}+F_{n-1}\cdot F_{n+1}$
- $gcd(F_a,F_b)=F_{gcd(a,b)}$
- 模 n 周期（皮萨诺周期）
  - $\pi(p^k)=p^{k-1}\pi(p)$
  - $\pi(nm)=lcm(\pi(n),\pi(m)),\forall n\perp m$
  - $\pi(2)=3,\pi(5)=20$
  - $\forall p\equiv\pm1\pmod{10},\pi(p)|p-1$
  - $\forall p\equiv\pm2\pmod 5,\pi(p)|2p+2$

常见生成函数

- $(1+ax)^n=\sum_{k=0}^n\binom{n}{k}a^kx^k$
- $\frac{1-x^{r+1}}{1-x}=\sum_{k=0}^n x^k$
- $\frac{1}{1-ax}=\sum_{k=0}^\infty a^kx^k$
- $\frac{1}{(1-x)^2}=\sum_{k=0}^\infty(k+1)x^k$
- $\frac{1}{(1-x)^n}=\sum_{k=0}^\infty\binom{n+k-1}{k}x^k$
- $e^x=\sum_{k=0}^\infty\frac{x^k}{k!}$
- $\ln(1+x)=\sum_{k=0}^\infty\frac{(-1)^{k+1}}{k}x^k$

组合数学

- Stirling数

```
//第一类Stirling数 表示将n个不同元素构成m个圆排列的
const int MOD=1e9+7
int s[MAXN][MAXN];

void init()
{
    memset(s,0,sizeof(s));
    s[0][0]=1;
    for(int i=1;i<MAXN;i++)
        for(int j=1;j<=i;j++)
```

```
            s[i][j]=(s[i-1][j-1]+1LL*(i-1)*s[i-1][j]%MOD)%MOD;
}

//第二类Stirling数 表示将n个不同的元素拆分成m个无序集合的方案数
//可以通过fft快速求解
const int MOD=1e9+7;
int s[MAXN][MAXN];//存放要求的

void init()
{
    memset(s,0,sizeof(s));
    s[0][0]=1;
    for(int i=1;i<MAXN;i++)
        for(int j=1;j<=i;j++)
            s[i][j]=(s[i-1][j-1]+1LL*j*s[i-1][j]%MOD)%MOD;
}

```

- 组合数阶乘逆元

```
ll fac[N],invf[N];
ll fp(ll a, ll x) {
    ll ret = 1;
    for (;x;x>>=1,a=a*a%mod) if (x&1) ret=ret*a%mod;
    return ret;
}
ll C(ll a, ll b) {
    if (b == 0) return 1;
    return fac[a]*invf[b]%mod*invf[a-b]%mod;
}
void init() {
    fac[1] = 1;
    rep(i,2,2e5) fac[i] = fac[i-1]*i%mod;
    invf[200000] = fp(fac[200000],mod-2);
    per(i,199999,0) invf[i] = invf[i+1]*(i+1)%mod;
}
```

数论

- 扩展卢卡斯定理

```
// luogu 4720
// n,m <= 1e18, mod <= 1e6

#include <cstdio>
using namespace std;
typedef long long ll;
inline ll fp(ll a,ll x,ll p) {
    ll ret = 1; a%=p;
    for (;x;x>>=1,a=a*a%p) if (x&1) ret=ret*a%p;
    return ret;
}
inline ll gcd(ll a,ll b) {
    if (b == 0) return a;
    return gcd(b,a%b);
}
inline void exgcd(ll a,ll b,ll &x,ll &y) {
    if (!b) {x=1,y=0; return;}
    exgcd(b,a%b,x,y);
    ll tmp = x; x = y; y = tmp-a/b*y;
}
inline ll INV(ll a,ll p) {
    ll x,y; exgcd(a,p,x,y);
    return (x+p)%p;
}
inline ll FAC(ll n,ll p,ll pk) {  // (n!/p^all) mod(p^k)
    ll ret = 1;
    for (ll x=n;x;x/=p) {
        ll t1=1, t2=1;
        for (ll i=1;i<=pk;++i) if (i%p) t1 = t1*i%pk;
        t1 = fp(t1,x/pk,pk);
        for (ll i=pk*(x/pk);i<=x;++i) if (i%p) t2 = i%pk*t2%pk;
        ret = ret*t1%pk*t2%pk;
    }
    return ret;
}
inline ll PX(ll x,ll p) {
```

```
    ll ret = 0;
    for (ll i=x;i;i/=p) ret += i/p;
    return ret;
}
inline ll C_PK(ll n,ll m,ll p,ll pk) {
    ll ret = 1;
    ret = ret * FAC(n,p,pk) * INV(FAC(m,p,pk),pk) % pk * INV(FAC(n-m,p,pk),pk) % pk;
    ret = ret * fp(p,PX(n,p)-PX(m,p)-PX(n-m,p),pk) % pk;
    return ret;
}
ll exLucas(ll n,ll m,ll mod) {
    ll md[15]={},xx[15]={},tmp=mod; int ptot=0;
    for (int i=2;i*i<=mod;++i) {
        if (tmp%i == 0) {
            ll pk = 1;
            while (tmp%i == 0) {
                tmp /= i;
                pk *= i;
            }
            md[++ptot] = pk;
            xx[ptot] = C_PK(n,m,i,pk);
        }
    }
    if (tmp > 1) {
        md[++ptot] = tmp;
        xx[ptot] = C_PK(n,m,tmp,tmp);
    }
    //CRT
    ll ret = 0;
    for (int i=1;i<=ptot;++i) {
        ll M = mod/md[i], V = INV(M,md[i]);
        ret = (ret+xx[i]*M%mod*V%mod)%mod;
    }
    return ret;
}

int main() {
    ll n,m,mod;
    scanf("%lld%lld%lld",&n,&m,&mod);
    printf("%lld\n",exLucas(n,m,mod));
    return 0;
}
```

- inv

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

void exgcd(int a,int b,int &x, int &y) {
    if (b == 0) {
        x = 1, y = 0;
        return;
    }
    exgcd(b,a%b,y,x);
    y -= (a / b) * x;
}
long long inv[200005];
void get_inv(long long p) {
    inv[1] = 1;
    for (int i=2;i<=n;++i) {
        inv[i] = (long long)(p - p/i) * inv[p%i] % p;
    }
}
int main() {

    return 0;
}
```

- euler

```
#include <bits/stdc++.h>
using namespace std;
const int MAXN = 1000000;

int prime[MAXN],phi[MAXN],mu[MAXN],cnt;
bool vis[MAXN];
void euler() {
    phi[1] = 1;
```

```
    mu[1] = 1;
    for (int i=2;i<MAXN;++i) {
        if (!vis[i]) {
            mu[i] = -1;
            phi[i] = i-1;
            prime[cnt++] = i;
        }
        for (int j=0;j<cnt && 1LL*i*prime[j]<MAXN;++j) {
            vis[i*prime[j]] = 1;
            if (i % prime[j]) {
                phi[i*prime[j]] = phi[i] * (prime[j]-1);
                mu[i*prime[j]] = -mu[i];
            }
            else {
                phi[i*prime[j]] = phi[i] * prime[j];
                mu[i*prime[j]] = 0;
                break;
            }
        }
    }
}
int main() {
    euler();
    int n;
    for (int i=1;i<=cnt;++i) cout << prime[i] << endl;
    return 0;
}
```

- BSGS

```
LL BSGS(LL a, LL b, LL p) { // a^x = b (mod p)
    a %= p;
    if (!a && !b) return 1;
    if (!a) return -1;
    static map<LL, LL> mp; mp.clear();
    LL m = sqrt(p + 1.5);
    LL v = 1;
    FOR (i, 1, m + 1) {
        v = v * a % p;
        mp[v * b % p] = i;
    }
    LL vv = v;
    FOR (i, 1, m + 1) {
        auto it = mp.find(vv);
        if (it != mp.end()) return i * m - it->second;
        vv = vv * v % p;
    }
    return -1;
}
LL exBSGS(LL a, LL b, LL p) { // a^x = b (mod p)
    a %= p; b %= p;
    if (a == 0) return b > 1 ? -1 : b == 0 && p != 1;
    LL c = 0, q = 1;
    while (1) {
        LL g = __gcd(a, p);
        if (g == 1) break;
        if (b == 1) return c;
        if (b % g) return -1;
        ++c; b /= g; p /= g; q = a / g * q % p;
    }
    static map<LL, LL> mp; mp.clear();
    LL m = sqrt(p + 1.5);
    LL v = 1;
    FOR (i, 1, m + 1) {
        v = v * a % p;
        mp[v * b % p] = i;
    }
    FOR (i, 1, m + 1) {
        q = q * v % p;
        auto it = mp.find(q);
        if (it != mp.end()) return i * m - it->second + c;
    }
    return -1;
}
```

字符串

- manacher

```cpp
#include <bits/stdc++.h>
#define RUSH ios_base::sync_with_stdio(0)
#define rep(i,a,b) for (int i=a;i<=b;++i)
#define mst(a,b) memset(a,b,sizeof(a))
using namespace std;
const int MAXN = 3334;
int MIN(int a,int b) {return a<b?a:b;}
int MAX(int a,int b) {return a>b?a:b;}
int n,m;
int len;
int p[MAXN],mapp[2][MAXN][MAXN];
char a[MAXN],str[MAXN],mater[MAXN][MAXN];
void manacher() { //p是半径，扩倍以后相当于原串直径
    len = strlen(str);

    int id,mx = 0;
    rep(i,1,len-1) {
        if (mx > i) p[i] = MIN(p[2*id-i],mx-i);
        else p[i] = 1;
        for (;str[i+p[i]]==str[i-p[i]];p[i]++);
        if (p[i]+i > mx) {
            mx = p[i] + i;
            id = i;
        }
    }
}
void work(char a[],int num,int t) {
    mst(str,0);
    str[0] = '$'; str[1] = '#';
    for (int i=0;a[i];++i) {
        str[2*i+2] = a[i];
        str[2*i+3] = '#';
    }
    len = strlen(str);
    str[len] = '*';
    manacher();
    int cnt = 0;
    rep(i,1,len-1) {
        if (str[i]>='a'&&str[i]<='z') mapp[t][num][++cnt] = p[i]-1;
    }
}
void init() {
    int m,n;
    scanf("%d%d",&m,&n);
    rep(i,0,m-1) scanf("%s",mater[i]);
    rep(i,0,m-1) {
        rep(j,0,n-1) a[j] = mater[i][j];
        work(a,i+1,0);
    }
    rep(i,0,n-1) {
        rep(j,0,m-1) a[j] = mater[j][i];
        work(a,i+1,1);
    }
    int ans = 0;
    rep(i,1,m) rep(j,1,n) ans = MAX(ans,mapp[0][i][j]*mapp[1][i][j]);
    cout<<ans<<endl;
}
int main() {
    init();
}

```

- 扩展KMP

```cpp
//next[]: x[i..m-1] x[0..m-1] lcp
void pre_EKMP(char x[],int m,int next[])
{
    next[0]=m;
    int j=0;
    while(j+1<m&&x[j]==x[j+1])j++;
    next[1]=j;
    int k=1;
    for(int i=2;i<m;i++)
    {
        int p=next[k]+k-1;
        int L=next[i-k];
```

```
        if(i+L<p+1)next[i]=L;
        else
        {
            j=max(0,p-i+1);
            while(i+j<m && x[i+j]==x[j])j++;
            next[i]=j;
            k=i;
        }
    }
}
//extend[]: y[i..n-1] x[0..m-1] lcp
void EKMP(char x[],int m,char y[],int n,int next[],int extend[])
{
    pre_EKMP(x,m,next);
    int j=0;
    while(j<n&&j<m&&x[j]==y[j])j++;
    extend[0]=j;
    int k=0;
    for(int i=1;i<n;i++)
    {
        int p=extend[k]+k-1;
        int L=next[i-k];
        if(i+L<p+1)extend[i]=L;
        else
        {
            j=max(0,p-i+1);
            while(i+j<n && j<m && y[i+j]==x[j])j++;
            extend[i]=j;
            k=i;
        }
    }
}

```

- AC自动机

```
//luogu ac自动机简单版模版

#include <bits/stdc++.h>
#define RUSH ios_base::sync_with_stdio(0)
using namespace std;
const int MAXN = 1000010;

int n,tcnt;
struct Ahoy {
    int c[30];
    int fail;
    int end;
}tr[MAXN];
void Insert(string s) {
    int now = 0;
    int len = s.length();
    for (int i=0;i<len;i++) {
        int ch = s[i]-'a'+1;
        if (!tr[now].c[ch]) tr[now].c[ch] = ++tcnt;
        now = tr[now].c[ch];
    }
    tr[now].end++;
}
void getfail() {
    queue<int> Q;
    for (int i=1;i<=26;i++) {
        if (tr[0].c[i]) {
            tr[tr[0].c[i]].fail = 0;
            Q.push(tr[0].c[i]);
        }
    }
    while (!Q.empty()) {
        int now = Q.front();
        Q.pop();
        for (int i=1;i<=26;i++) {
            if (tr[now].c[i]) {
                tr[tr[now].c[i]].fail = tr[tr[now].fail].c[i];
                Q.push(tr[now].c[i]);
            }
            else tr[now].c[i] = tr[tr[now].fail].c[i]; //字典图
        }
    }
```

```
}
int Query(string s) {
    int len = s.length();
    int now = 0, ans = 0;
    for (int i=0;i<len;i++) {
        int ch = s[i]-'a'+1;
        now = tr[now].c[ch];
        for (int j=now;j&&tr[j].end!=-1;j=tr[j].fail) {
            ans += tr[j].end;
            tr[j].end = -1;
        }
    }
    return ans;
}
int main() {
    RUSH;
    cin>>n;
    for (int i=1;i<=n;i++) {
        string s;
        cin>>s;
        Insert(s);
    }
    tr[0].fail = 0;
    getfail();
    string m;
    cin>>m;
    cout<<Query(m)<<endl;
    return 0;
}
```

- 回文树

```
const int MAXN=100005;
const int CHAR=26;

struct Palindromic_Tree
{
    int next[MAXN][CHAR];//next指针，next指针和字典树类似，指向的串为当前串两端加上同一个字符构成
    int fail[MAXN];//fail指针，失配后跳转到fail指针指向的节点
    int cnt[MAXN];//节点i表示的本质不同的串的个数（建树时求出的不是完全的，最后count()函数跑一遍以后才是正确的）
    int num[MAXN];//表示以节点i表示的最长回文串的最右端点为回文串结尾的回文串个数
    int len[MAXN];//len[i]表示节点i表示的回文串的长度
    int S[MAXN];//存放添加的字符
    int last;//指向上一个字符所在的节点，方便下一次add
    int n;//字符数组指针
    int p;//节点指针

    int newnode(int l)//新建节点
    {
        for(int i=0;i<CHAR;i++) next[p][i]=0;
        cnt[p]=0;num[p]=0;len[p]=l;
        return p++;
    }

    void init()//初始化
    {
        p=0;
        newnode(0);
        newnode(-1);
        last=0;n=0;
        S[n]=-1;//开头放一个字符集中没有的字符，减少特判
        fail[0]=1;
    }

    int get_fail(int x)//和KMP一样，失配后找一个尽量最长的
    {
        while(S[n-len[x]-1]!=S[n]) x=fail[x];
        return x;
    }

    void add(int c)
    {
        c-='a';
        S[++n]=c;
        int cur=get_fail(last);//通过上一个回文串找这个回文串的匹配位置
        if(!next[cur][c])//如果这个回文串没有出现过，说明出现了一个新的本质不同的回文串
        {
            int now=newnode(len[cur]+2);//新建节点
```

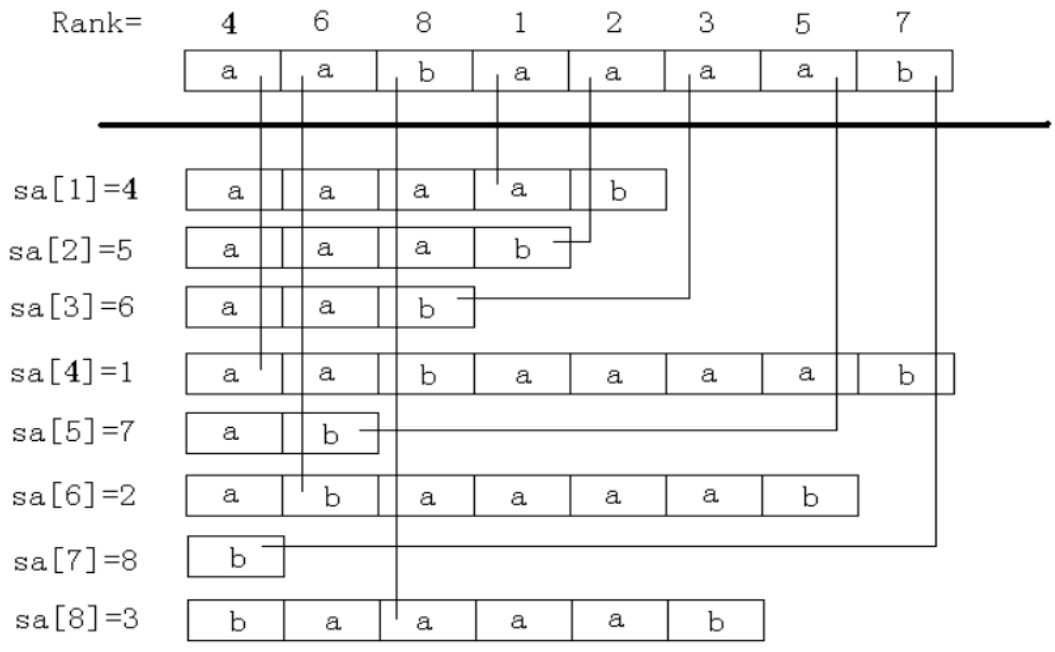
```
            fail[now]=next[get_fail(fail[cur])][c];//和AC自动机一样建立fail指针，以便失配后跳转
            next[cur][c]=now;
            num[now]=num[fail[now]]+1;
        }
        last=next[cur][c];
        cnt[last]++;
    }

    void count()
    {
        for(int i=p-1;i>=0;i--) cnt[fail[i]]+=cnt[i];
        //父亲累加儿子的cnt，因为如果fail[v]=u，则u一定是v的子回文串!
        cnt[0]=cnt[1]=0;//将两个根清零
    }
};
```

- 最小表示法

```
#include <iostream>
#include <cstring>
using namespace std;
char s[1000005];
int max(int x,int y) {return x<y?y:x;}
int min(int x,int y) {return x<y?x:y;}
int main()
{
    scanf("%s",s);
    int i = 0,j = 1,k = 0;
    int len = strlen(s);
    while (k<len && i<len && j<len) {
        if (s[(i+k)%len] == s[(j+k)%len]) k++;
        else {
            if (s[(i+k)%len] > s[(j+k)%len]) i = max(j+1,i+k+1);
            else j = max(i+1,j+k+1);
            k = 0;
        }
    }
    for (int t=0;t<len;t++) printf("%c",s[(min(i,j)+t)%len]); //取min:j后跳当且仅当开始的字符串更小，即此时i+k与j之间的字符串大于i串
    printf("\n");
    return 0;
}
```

- 后缀数组



- $sa[rk[i]] = rk[sa[i]] = i$
- $height[i] = lcp(sa[i], sa[i-1])$
- $height[rk[i]] >= height[rk[i-1]] - 1$
- $lcp(sa[i], sa[j]) = min(height[i+1...j])$

- 不同子串个数$\frac{n(n+1)}{2}-\sum_{i=2}^{n}height[i]$
- kmp

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int MAXN = 1e6+5;

string s;
int nxt[MAXN];
int main() {            //find s in t
    cin >> s;
    s += '#';
    int n = s.length();
    char ch = getchar();
    nxt[0] = 0;
    for (int i=1,j=0;i<=n;++i) {
        while (j > 0 && s[i] != s[j]) j = nxt[j-1];
        if (s[i] == s[j]) ++j;
        nxt[i] = j;
    }
    int j = 0,i = 0,cnt = 0;
    while ((ch = getchar())!='\n') {
        while (j > 0 && ch != s[j]) j = nxt[j-1];
        if (ch == s[j]) ++j;
        if (j == n-1) {
            cout << i << endl;
            cnt++;
            j = nxt[j-1];
        }
        i++;
    }
    cout << cnt << endl;
    return 0;

}

```

- lyndon分解

若字符串s的字典序严格小于s的所有后缀的字典序，称s是简单串 a,b,aab,abb,ababb 串s的Lyndon分解记为s=w1w2...wk,其中所有wi为简单串， 并且他们的字典序按照非严格单减排序，即w1>=w2>=wk。 可以发现，这样的分解存在且唯一。

```
vector<string> duval(string const& s) {
  int n = s.size(), i = 0;
  vector<string> factorization;
  while (i < n) {
    int j = i + 1, k = i;
    while (j < n && s[k] <= s[j]) {
      if (s[k] < s[j])
        k = i;
      else
        k++;
      j++;
    }
    while (i <= k) {
      factorization.push_back(s.substr(i, j - k));
      i += j - k;
    }
  }
  return factorization;
}
```

图

- 欧拉回路

O(N+M) 回路： 无向：度数都是偶数；有向：入度=出度 路径： 无向：0/2度数奇数； 有向：一对出=入+1 入=出+1 无向图建边正数u到v，负数v到u

```
int n,m;
int d[N],h[N],to[M],w[M],nxt[M<<1],tot;
int ans[M],cnt;
bool vis[M<<1];
namespace UDG {
    void add(int u,int v,int l) {
        d[u]++;
        to[++tot]=v; w[tot]=l; nxt[tot]=h[u];h[u]=tot;
```

```
    }
    void dfs(int u) {
        for (int &i=h[u];i;) {
            if (vis[i]) {i=nxt[i]; continue;}
            vis[i] = vis[i^1] = 1;
            int j = w[i];
            dfs(to[i]);
            ans[++cnt] = j;
        }
    }
    void solve() {
        ed = 1;
        scanf("%d%d",&n,&m);
        rep(i,1,m) {
            int u,v; scanf("%d%d",&u,&v);
            add(u,v,i); add(v,u,-i);
        }
        rep(i,1,n) if (d[i]&1) {puts("NO"); return;}
        rep(i,1,n) if (h[i]) {dfs(i); break;}
        rep(i,1,n) if (h[i]) {puts("NO"); return;}
        puts("YES");
        per(i,m,1) printf("%d ",ans[i]);
    }
}
namespace DG {
    void add(int u,int v) {
        d[u]++; d[v]--;
        to[++tot]=v; nxt[tot]=h[u];h[u]=tot;
    }
        void dfs(int u) {
        for (int &i=h[u];i;) {
            if (vis[i]) {i=nxt[i]; continue;}
            vis[i] = 1;
            int j = i;
            dfs(to[i]);
            ans[++cnt] = j;
        }
    }
    void solve() {
        ed = 1;
        scanf("%d%d",&n,&m);
        rep(i,1,m) {
            int u,v; scanf("%d%d",&u,&v);
            add(u,v);
        }
        rep(i,1,n) if (d[i]) {puts("NO"); return;}
        rep(i,1,n) if (h[i]) {dfs(i); break;}
        rep(i,1,n) if (h[i]) {puts("NO"); return;}
        puts("YES");
        per(i,m,1) printf("%d ",ans[i]);
    }
}
```

- 最短路

O(mlogm)

```

#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <algorithm>
#include <cstring>
#include <queue>
using namespace std;

typedef long long ll;
typedef pair<int, int> pii;
const int MAXN = 100010;
const ll INF = 0x7fffffff;
const double eps = 1e-8;
const int mod = 1e7+7;

vector<pii> vec[MAXN];
priority_queue<pii, vector<pii>, greater<pii> > pq;

int dis[MAXN];
bool vis[MAXN];
```

```cpp
int n, m, S;

void dijk() {
    memset(dis,0x3f,sizeof(dis));
    memset(vis,0,sizeof(vis));
    pq.push(make_pair(0, S));
    dis[S] = 0;
    while (!pq.empty()) {
        int u = pq.top().second;
        pq.pop();
        if (vis[u]) continue;
        vis[u] = true;
        for (int i=0;i<vec[u].size();++i) {
            int v = vec[u][i].second, w = vec[u][i].first;
            if (dis[u] + w < dis[v]) {
                dis[v] = dis[u] + w;
                pq.push(make_pair(dis[v], v));
            }

        }
    }
    return;
}
int main() {
    scanf("%d%d", &n, &m);
    for (int i=1;i<=m;++i) {
        int u, v, w;
        scanf("%d%d%d", &u, &v, &w);
        vec[u].push_back(make_pair(w, v));
        vec[v].push_back(make_pair(w, u));
    }
    dijk();
    int ans = INT_MAX;
    for (int i=1;i<=n;i++) ans = min(ans, dis[i]);
    printf("%d\n",ans);
    return 0;
}

```

- tarjan

```cpp
#include <bits/stdc++.h>
using namespace std;
#define rep(a,b,c) for (int a=b;a<=c;++a)
#define pb push_back
const int MAXN = 10001;
int n,m,belong[MAXN],dfn[MAXN],low[MAXN],INDEX,bcnt;
bool in[MAXN];
vector<int> vec[MAXN];
stack<int> st;
int MIN(int a,int b) {return a>b?b:a;}
void tarjan_SCC(int u)
{
    int v;
    dfn[u] = low[u] = ++INDEX;
    st.push(u);
    in[u] = true;
    rep(i,0,(int)vec[u].size()-1) {
        v = vec[u][i];
        if(!dfn[v]) {
            tarjan(v);
            low[u] = MIN(low[u],low[v]);
        }
        else if(in[v]) {
            low[u] = MIN(low[u],dfn[v]);
        }
    }
    v= -1;
    if(dfn[u] == low[u]) {
        bcnt++;
        do {
            v = st.top(); st.pop();
            belong[v] = bcnt;
            in[v] = false;
        } while(v != u);
    }
}
void tarjan_EDGE(int u,int fa) {
```

```
    int v;
    dfn[u] = low[u] = ++INDEX;
    rep(i,0,(int)vec[u].size()-1) {
        v = vec[u][i];
        if(!dfn[v]) {
            tarjan_EDGE(v,u);
            low[u] = MIN(low[u],low[v]);
        }
        else if(v != fa) {
            low[u] = MIN(low[u],dfn[v]);
        }
    }
    //low相同的在一个边双中
}
int main()
{
    cin>>n>>m;
    rep(i,1,m) {
        int a,b;
        cin>>a>>b;
        vec[a].push_back(b);
    }
    rep(i,1,n) {
        //cout<<i<<endl;
        if (!dfn[i]) tarjan_SCC(i);
    }
    rep(i,1,n) cout<<belong[i]<<' ';
    cout<<endl;
    return 0;
}
/*
6 8
1 2
1 3
2 4
3 4
3 5
4 1
4 6
5 6
*/

```

- 最小环

必然是一个(u,v)边长+图删除(u,v)后的dis(u,v) dij同理，O(m(n+m)logm)

```
const int N = 405;
int dis[N][N],val[N][N];
int ans = INT_MAX;
rep(i,1,n) rep(j,1,n) dis[i][j] = val[i][j];
rep(k,1,n) {
    rep(i,1,k-1) rep(j,1,i-1) ans = min(ans,dis[i][j]+val[i][k]+val[k][j]);
    rep(i,1,n) rep(j,1,n) dis[i][j] = min(dis[i][j], dis[i][k]+dis[k][j]);
}
```

- 最大团

最大团 = 补图G的最大独立集数 ———>最大独立集数 = 补图G'最大团

```
#include<bits/stdc++.h>
const int MAXN=102;
int mx;//最大团数(要初始化为0)
int x[MAXN],tuan[MAXN];
int can[MAXN][MAXN];
//can[i]表示在已经确定了经选定的i个点必须在最大团内的前提下还有可能被加进最大团的结点集合
int num[MAXN];
//num[i]表示由结点i到结点n构成的最大团的结点数
bool g[MAXN][MAXN];//邻接矩阵(从1开始)
int n,m;
bool dfs(int tot,int cnt)
{
    if(tot==0) {
        if(cnt>mx) {
            mx=cnt;
            for(int i=0;i<mx;i++) tuan[i]=x[i];
            return true;
        }
```

```
        return false;
    }
    for(int i=0;i<tot;i++) {
        if(cnt+(tot-i)<=mx)return false;
        if(cnt+num[can[cnt][i]]<= mx)return false;
        int k=0;
        x[cnt]=can[cnt][i];
        for(int j=i+1;j<tot;j++) {
            if(g[can[cnt][i]][can[cnt][j]]) {
                can[cnt+1][k++]=can[cnt][j];
            }
        }
        if(dfs(k,cnt+1))return false;
    }
    return false;
}
void MaxTuan() {
    mx=1;
    for(int i=n;i>=1;i--) {
        int k=0;
        x[0]=i;
        for(int j=i+1;j<=n;j++) {
            if(g[i][j]) can[1][k++]=j;
        }
        dfs(k,1);
        num[i]=mx;
    }
}
int main() {
    while(scanf("%d",&n)&&n) {
        memset(g,0,sizeof(g));
        for(int i=1;i<=n;i++) {
            for(int j=1;j<=n;j++) {
                scanf("%d",&g[i][j]);
            }
        }
        mx=0;
        MaxTuan();
        printf("%d\n",mx);
    }
    return 0;
}

```

树

- 矩阵树定理 给定一个无向图G，求它生成树的个数t(G)

(1)G的度数矩阵D[G]是一个n*n的矩阵,并且满足:当i≠j时,dij=0;当i=j时,dij等于vi的度数; (2)G的邻接矩阵A[G]是一个n*n的矩阵,并且满足:如果vi,vj之间有边直接相连,则aij=1,否则为0; 定义图G的Kirchhoff矩阵C[G]为C[G]=D[G]-A[G]; Matrix-Tree定理:G的所有不同的生成树的个数=C[G]任何一个n-1阶主子式的行列式的绝对值； n-1阶主子式是对于r(1≤r≤n),将C[G]的第r行,第r列同时去掉后得到的新矩阵,用Cr[G]表示;

- 特殊性质： (1)对于任何一个图G,它的Kirchhoff矩阵C的行列式总是0,这是因为C每行每列所有元素的和均为0; (2)如果G是不连通的,则它的Kirchhoff矩阵C的任一个主子式的行列式均为0; (3)如果G是一颗树,那么它的Kirchhoff矩阵C的任一个n-1阶主子式的行列式均为1;
- 虚树

处理树上询问点数和n在同一个量级，每一个询问约等于一个树dp

```
#include <bits/stdc++.h>
#define rep(i,a,b) for (int i=a;i<=b;++i)
#define per(i,a,b) for (int i=a;i>=b;--i)
using namespace std;
typedef long long ll;
const int N = 5e5+5;
int n,m,k;
int h[N],mk[N]; //mk标记关键点
ll dp[N];
struct edge {
    int v,w;
};
vector<edge> g[N],vt[N];
int idx,id[N],fa[N][20],dep[N];
int me[N][20]; //向上最短边
void dfs(int u,int f) {
    id[u] = ++idx;
    dep[u] = dep[f]+1; fa[u][0] = f;
    for (auto e:g[u]) {
        int v = e.v,w = e.w;
        if (v == f) continue;
```

```
        dfs(v,u);
        me[v][0] = w;
    }
}
int lca(int a,int b) {
    if (dep[a] < dep[b]) swap(a,b);
    int dis = dep[a] - dep[b];
    per(i,18,0) if (dis>>i&1) a = fa[a][i];
    if (a == b) return a;
    per(i,18,0) if (fa[a][i]!=fa[b][i]) a=fa[a][i],b=fa[b][i];
    return fa[a][0];
}
int sta[N],top;
inline bool cmp(int a,int b) { return id[a] < id[b]; }
inline int getme(int u,int v) {
    if (dep[u] < dep[v]) swap(u,v);
    if (u == v) return 0;
    int ret = INT_MAX;
    int cur = u;
    per(i,18,0) if (dep[fa[cur][i]] <= v) {
        ret = min(ret,me[cur][i]);
        cur = fa[cur][i];
    }
    return ret;
}
inline void add(int u,int v) {
    int w = getme(u,v);
    vt[u].push_back((edge){v,w});
    vt[v].push_back((edge){u,w});
}
void build() {
    sort(h+1,h+k+1,cmp);
    sta[top=1] = 1; vt[1].clear();
    rep(i,1,k) {
        if (h[i] == 1) continue;
        int l = lca(h[i],sta[top]);
        if (l != sta[top]) {
            while (id[l] < id[sta[top-1]])
                add(sta[top-1],sta[top]), top--;
            if (id[l] > id[sta[top-1]])
                vt[l].clear(),add(l,sta[top]),sta[top] = l;
            else add(l,sta[top--]);
        }
        vt[h[i]].clear();
        sta[++top] = h[i];
    }
    rep(i,1,top-1) add(sta[i],sta[i+1]);
}
void solve(int u,int f) {
    dp[u] = 0;
    for (auto e:vt[u]) {
        int v = e.v,w = e.w;
        if (v == f) continue;
        solve(v,u);
        if (mk[v]) dp[u] += (ll)w;
        else dp[u] += min(dp[v],(ll)w);
    }
}
int main() {
    scanf("%d",&n);
    rep(i,0,n) rep(j,0,18) me[i][j] = INT_MAX;
    rep(i,1,n-1) {
        int u,v,w;
        scanf("%d%d%d",&u,&v,&w);
        g[u].push_back((edge){v,w});
        g[v].push_back((edge){u,w});
    }
    dfs(1,0);
    rep(i,1,18) rep(j,1,n) {
        fa[j][i] = fa[fa[j][i-1]][i-1];
        me[j][i] = min(me[fa[j][i-1]][i-1],me[j][i-1]);
    }
    scanf("%d",&m);
    rep(i,1,m) {
        scanf("%d",&k);
        rep(i,1,k) scanf("%d",&h[i]),mk[h[i]] = 1;
        build();
        solve(1,0);
        printf("%lld\n",dp[1]);
```

```
        rep(i,1,k) mk[h[i]] = 0;
    }
}
```

- 树网的核

```
#include <bits/stdc++.h>
#define rep(i,a,b) for (int i=a;i<=b;++i)
#define fi first
#define se second
using namespace std;
typedef long long ll;
const int N = 2e5+5;
const int inf = 2147483647;

int n,s,top;
int dis[305],fa[305];
bool vis[305];
vector<pair<int,int> > g[305];

void dfs(int u,int f) {
    fa[u] = f;
    if (dis[u] > dis[top]) top = u;
    for (auto e:g[u]) {
        int v = e.fi, w = e.se;
        if (v == f || vis[v]) continue;
        dis[v] = dis[u]+w;
        dfs(v,u);
    }
}
int main() {
    scanf("%d%d",&n,&s);
    rep(i,1,n-1) {
        int u,v,w;
        scanf("%d%d%d",&u,&v,&w);
        g[u].push_back({v,w});
        g[v].push_back({u,w});
    }
    dfs(1,0);
    int p = top; top = 0;
    rep(i,0,n) dis[i] = 0,fa[i] = 0;
    dfs(p,0);
    int l = top, r = top, ans = inf;
    while (r) {
        while (dis[l]-dis[r] > s) l = fa[l];
        //cout << l << ' ' << r << endl;
        ans = min(ans,max(dis[top]-dis[l],dis[r]));
        r = fa[r];
    }
    for (int i=top;i;i=fa[i]) vis[i] = 1;
    int tmp = 0;
    for (int i=top;i;i=fa[i]) {
        top = 0;
        dfs(i,fa[i]);
        tmp = max(tmp,dis[top]-dis[i]);
    }
    printf("%d\n",max(ans,tmp));
    return 0;
}
```

- Prufer序列 Prufer 序列可以将一个带标号n个结点的树用[1,n]中的n-2个整数表示。完全图的生成树与数列之间的双射。
- 树建立prufer序列

  每次选择一个编号最小的叶结点并删掉它，在序列中记录下它连接到的结点，n-2次后就只剩下两个结点。

  O(n)：删除一个叶子时，判断与它相连的点标号是否小于它，如果小于，直接删除

```
int fa[N],pru[N];
void fa_to_pru() {
    vector<int> deg(n+1); //后继数量
    int ptr = -1;
    rep(i,1,n) deg[fa[i]]++;
    rep(i,1,n) if (deg[i] == 0) { ptr = i;break; }
    int leaf = ptr;
    rep(i,1,n-2) {
        int f = fa[leaf];
        pru[i] = f;
        if (--deg[f] == 0 && f < ptr) leaf = f;
```

```
        else {
            ptr++;
            while (deg[ptr]) ++ptr;
            leaf = ptr;
        }
    }
}
```

- prufer序列建立树

```
void pru_to_fa() {
    vector<int> deg(n+1);
    rep(i,1,n-2) deg[pru[i]]++;
    int ptr = 1;
    while (deg[ptr]) ++ptr;
    int leaf = ptr;
    rep(i,1,n-2) {
        int f = pru[i]; fa[leaf] = f;
        if (--deg[f] == 0 && f < ptr) leaf = f;
        else {
            ++ptr;
            while (deg[ptr]) ++ptr;
            leaf = ptr;
        }
    }
    fa[leaf] = n;
}
```

- Cayley公式

  完全图Kn有n^{n-2}棵生成树

- 图连通方案数

  一个n个点m条边的带标号无向图有k个连通块。添加k-1条边使得整个图连通，求方案数。

  设每一个连通块大小为s_i，则方案数为$n^{k-2}\prod_{i=1}^{k}s_i$

- 倍增LCA

```
#include <bits/stdc++.h>
using namespace std;
#define pb push_back
#define mst(a,b) memset(a,b,sizeof(a))
const int MAXN = 100010;
struct node {
    int v;
};
vector<node> tr[MAXN];
int pnt[MAXN][20],dep[MAXN],root,n;
bool vis[MAXN];
int findroot(int u) {
    if (pnt[u][0]) return findroot(pnt[u][u]);
    return u;
}
void addedge(int u,int v) {
    node tmp; tmp.v=v;
    tr[u].pb(tmp);
    pnt[v][0] = u;
}
void dfs(int u) {
    vis[u] = 1;
    for (int i=0;i<tr[u].size();i++) {
        int v = tr[u][i].v;
        if (!vis[v]) {
            dep[v] = dep[u] + 1;
            dfs(v);
        }
    }
    return;
}
int lca(int a, int b) {
    if (dep[a]<dep[b]) swap(a,b);
    int dis = dep[a] - dep[b];
    for (int i=16;i>=0;i--)
        if ((1<<i)&dis) a = pnt[a][i];
    if (a == b) return a;
    for (int i=16;i>=0;i--) {
        if (pnt[a][i] != pnt[b][i]) a=pnt[a][i],b=pnt[b][i];
    }
```

```
    return pnt[a][0];
}
int main() {
    mst(pnt,0); mst(vis,0); mst(dep,0);
    root = findroot(1);
    dep[root] = 1;
    dfs(root);
    for (int i=1;i<=16;i++)
        for (int j=1;j<=n;j++)
            pnt[j][i] = pnt[pnt[j][i-1]][i-1];
    int a,b,c;
    c = lca(a,b);
    return 0;
}
```

- 次小生成树

```
const int MAXN=110;
const int INF=0x3f3f3f3f;
bool vis[MAXN];
int lowc[MAXN];
int pre[MAXN];
int cost[MAXN][MAXN];
int Max[MAXN][MAXN]; // longest edge from i to j on mst
bool used[MAXN][MAXN];
int Prim(int n)
{
    int ans=0;
    memset(vis,false,sizeof(vis));
    memset(Max,0,sizeof(Max));
    memset(used,false,sizeof(used));
    vis[0]=true;
    pre[0]=-1;
    for(int i=1;i<n;i++)
    {
        lowc[i]=cost[0][i];
        pre[i]=0;
    }
    lowc[0]=0;
    for(int i=1;i<n;i++)
    {
        int minc=INF;
        int p=-1;
        for(int j=0;j<n;j++)
        if(!vis[j]&&minc>lowc[j])
        {
            minc=lowc[j];
            p=j;
        }
        if(minc==INF)return -1;
        ans+=minc;
        vis[p]=true;
        used[p][pre[p]]=used[pre[p]][p]=true;
        for(int j=0;j<n;j++)
        {
            if(vis[j]&&j!=p)Max[j][p]=Max[p][j]=max(Max[j][pre[p]],lowc[p]);
            if(!vis[j]&&lowc[j]>cost[p][j])
            {
                lowc[j]=cost[p][j];
                pre[j]=p;
            }
        }
    }
    return ans;
}
int smst(int n,int ans) //point id start from 0
{
    int Min=INF;
    for(int i=0;i<n;i++)
        for(int j=i+1;j<n;j++)
            if(cost[i][j]!=INF && !used[i][j])
            {
                Min=min(Min,ans+cost[i][j]-Max[i][j]);
            }
    if(Min==INF)return -1;// no
    return Min;
}

```

树分治

- poj1741 点分治模版

```cpp
#include <iostream>
#include <cstdio>
#include <vector>
#include <cstring>
#include <algorithm>
#define rep(i,a,b) for (int i=a;i<=b;++i)
#define mst(a,b) memset(a,b,sizeof(a))
#define pii pair<int,int>
#define mp make_pair
#define pb push_back
#define fi first
#define se second
using namespace std;
int MAX(int a,int b) { return a>b?a:b; }
const int MAXN = 1e5+5;
const int inf = 2147483647;

vector<pii> vec[MAXN];
int n,m,G = 1,tot,ans;
int sz[MAXN],mxson[MAXN],dis[MAXN];
bool vis[MAXN];
void getG(int u,int fa) {
    sz[u] = 1;
    mxson[u] = 0;
    rep(i,0,(int)vec[u].size()-1) {
        int v = vec[u][i].fi;
        if (!vis[v] && v!=fa) {
            getG(v,u);
            sz[u] += sz[v];
            mxson[u] = MAX(mxson[u],sz[v]);
        }
    }
    mxson[u] = MAX(mxson[u],tot - sz[u]);
    G = mxson[G] > mxson[u]?u:G;
}
int dcnt = 0;
void getdis(int u,int fa,int dep) {
    dis[++dcnt] = dep;
    rep (i,0,(int)vec[u].size()-1) {
        int v = vec[u][i].fi;
        if (v != fa && !vis[v]) getdis(v,u,dep+vec[u][i].se);
    }
}
int calc(int u,int add) {
    int ret = 0;
    dcnt = 0;
    getdis(u,0,add);
    sort(dis+1,dis+dcnt+1);
    for (int l = 1,r = dcnt;l < r;) {
        if (dis[l] + dis[r] > m) r--;
        else {
            ret += r-l;
            l++;
        }
    }
    return ret;
}
void divide(int u) {
    vis[u] = 1;
    ans += calc(u,0);
    rep(i,0,(int)vec[u].size()-1) {
        int v = vec[u][i].fi;
        if (!vis[v]) {
            ans -= calc(v,vec[u][i].se); //减去两点在同一棵子树的情况
            G = 0; mxson[G] = inf; tot = sz[v];
            getG(v,u);
            divide(G);
        }
    }
}
int main() {
    while (scanf("%d%d",&n,&m)) {
        if (n == 0 && m == 0) break;
        rep(i,1,n) vec[i].clear();
```

```
        mst(vis,0);
        ans = 0;
        rep(i,1,n-1) {
            int x,y,z;
            scanf("%d%d%d",&x,&y,&z);
            vec[x].pb(mp(y,z)); vec[y].pb(mp(x,z));
        }
        G = 0; mxson[G] = inf; tot = n;
        getG(1,0);
        divide(G);
        cout<<ans<<endl;
    }
    return 0;
}
```

匹配_流

- 费用流

```
#include <bits/stdc++.h>
#define rep(i,a,b) for (int i=a;i<=b;++i)
#define mst(a,b) memset(a,b,sizeof(a))
#define mcp(a,b) memcpy(a,b,sizeof(b))
using namespace std;
const int MAXN = 5005;
const int inf = 0x3f3f3f3f;
int MIN(int a,int b) {return a<b?a:b;}
int n,m,S,T,COST;
int ccnt,lv[MAXN],dis[MAXN];
bool vis[MAXN];
struct Edge {
    int v,f,co;
    Edge *next,*rev;
}*h[MAXN],*cur[MAXN],pool[20*MAXN];
queue<int> Q;
void addedge(int u,int v,int cap,int cost) {
    Edge *p = &pool[++ccnt];
    Edge *q = &pool[++ccnt];
    p->v = v; p->f = cap; p->co = cost; p->next=h[u]; h[u] = p; p->rev = q;
    q->v = u; q->f = 0; q->co = -cost; q->next=h[v]; h[v] = q; q->rev = p;
}
bool makelevel() {
    mst(dis,0x3f);
    mcp(cur,h);
    while (!Q.empty()) Q.pop();
    Q.push(S);
    dis[S] = 0;
    vis[S] = 1;
    while (!Q.empty()) {
        int u = Q.front();
        Q.pop();
        vis[u] = 0;
        for (Edge *p=h[u];p;p=p->next) {
            int v = p->v;
            if (p->f && dis[u] + p->co < dis[v]) {
                dis[v] = dis[u] + p->co;
                if(!vis[v]) {    //如果不在队列内
                    Q.push(v);
                    vis[v] = true;
                }
            }
        }
    }
    return dis[T] != inf;
}
int findpath(int u,int minc) {
    if (u == T) return minc;
    vis[u] = 1;
    int sum = 0, flow;
    for (Edge *p=cur[u];p && sum <= minc;p=p->next) {
        cur[u] = p;
        if (p->f && !vis[p->v] && dis[p->v]==dis[u]+p->co) {
            flow = findpath(p->v,MIN(minc-sum,p->f));
            if (flow) {
                COST += flow * p->co;
                p->f -= flow;
                p->rev->f += flow;
                sum += flow;
```

```
            }
        }
    }
    vis[u] = 0;
    return sum;
}
int MCMF() {
    int ret = 0;
    while (makelevel()) {
        int tmp;
        while (tmp=findpath(S,inf)) {
            ret += tmp;
        }
    }
    return ret;
}
int main() {
    scanf("%d%d%d%d",&n,&m,&S,&T);
    mst(pool,0); mst(h,0);
    for (int i=1;i<=m;i++) {
        int x,y,ca,co;
        scanf("%d%d%d%d",&x,&y,&ca,&co);
        addedge(x,y,ca,co);
    }
    int ans=MCMF();
    cout<<ans<<' '<<COST<<endl;
}

```

- hungary

```
#include <bits/stdc++.h>
using namespace std;
#define mst(a,b) memset(a,b,sizeof(a))
#define rep(i,a,b) for (int i=a;i<=b;++i)
#define pb push_back
const int MAXN = 1005;
int n,m;
vector<int> vec[MAXN];
bool used[MAXN];
int pa[MAXN];
bool Hungary(int x) {
    rep(i,0,(int)vec[x].size()-1) {
        int v = vec[x][i];
        if (!used[v]) {
            used[v] = 1;
            if (pa[v] == 0 || Hungary(pa[v])) {
                pa[v] = x;
                return true;
            }
        }
    }
    return false;
}
int main() {
    cin>>n>>m;
    mst(pa,0); mst(used,0);
    rep(i,1,m) {
        int a,b; cin>>a>>b;
        vec[a].pb(b);
    }
    int ans = 0;
    rep(i,1,n) {
        mst(used,0);
        if (Hungary(i)) ans++;
    }
    cout<<ans<<endl;
    return 0;
}
```

- Dinic_最大流

poj1273

```

#include <bits/stdc++.h>
#define rep(i,a,b) for (int i=a;i<=b;++i)
#define mst(a,b) memset(a,b,sizeof(a))
```

```cpp
using namespace std;
const int MAXN = 5005;
int MIN(int a,int b) {return a<b?a:b;}
int n,m,S,T;
int ccnt,lv[MAXN];
struct Edge {
    int v,f;
    Edge *next,*rev;
}*h[MAXN],*cur[MAXN],pool[10*MAXN];
queue<int> Q;
void addedge(int u,int v,int c) {
    Edge *p = &pool[++ccnt];
    Edge *q = &pool[++ccnt];
    p->v = v; p->f = c; p->next=h[u]; h[u] = p; p->rev = q;
    q->v = u; q->f = 0; q->next=h[v]; h[v] = q; q->rev = p;
}
bool makelevel() {
    mst(lv,-1);
    while (!Q.empty()) Q.pop();
    rep(i,S,T) cur[i] = h[i];
    lv[S] = 0;
    Q.push(S);
    while (!Q.empty()) {
        int u = Q.front(); Q.pop();
        for (Edge *p=h[u];p;p=p->next) {
            int v = p->v;
            if (lv[v] == -1 && p->f) {
                Q.push(v);
                lv[v] = lv[u] + 1;
            }
        }
        if (lv[T] > 0) return true;
    }
    return false;
}
int findpath(int u,int minc) {
    if (u == T) return minc;
    int sum = 0, flow;
    for (Edge *p=cur[u];p && sum<=minc;p=p->next) {
        cur[u] = p;
        if (p->f && lv[p->v]==lv[u]+1) {
            flow = findpath(p->v,MIN(minc-sum,p->f));
            if (flow) {
                p->f -= flow;
                p->rev->f += flow;
                sum += flow;
            }
        }
    }
    if (sum == 0) lv[u] = -1;
    return sum;
}
int dinic() {
    int ret = 0;
    rep(i,S,T) cur[i] = h[i];
    while (makelevel()) {
        int tmp;
        while (tmp=findpath(S,INT_MAX)) {
            ret += tmp;
        }
    }
    return ret;
}
int main() {
    while(scanf("%d%d",&m,&n)!=EOF) {
        mst(pool,0); mst(h,0);
        for (int i=1;i<=m;i++) {
            int x,y,c;
            scanf("%d%d%d",&x,&y,&c);
            addedge(x,y,c);
        }
        S = 1; T = n;
        int ans=dinic();
        cout<<ans<<endl;
    }
}

```

- 带下界网络流
- 无源汇可行流：$u \to v$ 边容量为 $[l,r]$，连容量 $r-l$，虚拟源点到 v 连 l，u 到虚拟汇点连 l。

- 有源汇可行流：为了让流能循环使用，连 $T \to S$，容量 ∞。
- 有源汇最大流：跑完可行流后，加 $S' \to S$，$T \to T'$，最大流就是答案（$T \to S$ 的流量自动退回去了，这一部分就是下界部分的流量）。
- 有源汇最小流：T 到 S 的那条边的实际流量，减去删掉那条边后 T 到 S 的最大流。
- 网上说可能会减成负的，还要有限地供应 S 之后，再跑一遍 S 到 T 的。
- 费用流：必要的部分（下界以下的）不要钱，剩下的按照最大流。

小技巧

- 枚举子集

```
//真子集
for (int s = (S - 1) & S; s; s = (s - 1) & S)
//大小为k的子集
template<typename T>
void subset(int k, int n, T&& f) {
    int t = (1 << k) - 1;
    while (t < 1 << n) {
        f(t);
        int x = t & -t, y = t + x;
        t = ((t & ~y) / x >> 1) | y;
    }
}
```

- 快速乘

```
typedef long long ll;
inline ll mul(ll a,ll b,const ll &p) {
    ll lf = a*(b>>25LL)%p * (1LL<<25)%p;
    ll rg = a*(b & ((1LL<<25)-1))%p;
    return (lf+rg)%p;
}

```

- 双hash

```
struct myhash_pair {
    template<class T1,class T2> size_t operator()(const pair<T1,T2>&p) const {
        auto hash1 = hash<T1>{}(p.first);
        auto hash2 = hash<T2>{}(p.second);
        return hash1^hash2;
    }
};

unordered_map<pii,int> myhash;

pii duohash(int a[]) {
    ll h1=0,h2=0;
    rep(i,0,9) {
        h1 = (h1*31+a[i])%(1000000007);
        h2 = (h2*31+a[i])%(998244353);
    }
    return {(int)h1,(int)h2};
}

int main() {
    a[10] = {0,1,2,3,4,5,6,7,8,9,10};
    myhash[duohash(a)] = 1;
}

```

- 线性基

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
ll a[105],lb[62];
inline void insert(ll x) {
    for (int i=60;i+1;--i) {
        if (!(x>>i)) continue;
        if (!lb[i]) { lb[i] = x; break; }
        x = x^lb[i];
    }
    //if (x) return false; 不能插入,表示已经能被表示
}
```

```
ll get_max() {
    ll ret = 0;
    for (int i=60;i+1;--i)
        ret = ((ret^lb[i])>ret)?ret^lb[i]:ret;
    return ret;
}
void get_kth() { //第k小
    ll LB[62] = {};
    int cnt = 0;
    rep(i,0,60) if (lb[i]) LB[++cnt] = lb[i];
    int q; scanf("%d",&q);
    rep(i,1,q) {
        ll k,t = 0; scanf("%lld",&k);
        if (k > (1<<cnt)-1) { printf("-1\n"); continue;}
        ll ans = 0;
        while (k) {
            ans ^= (x&1)*LB[t++];
            k >>= 1;
        }
        printf("%lld\n",ans);
    }
}
void count() {
    //如果k能用线性基表示出来,线性基能表示出这个数的方案为2^(s - tot)
    //s为线性基插入过的次数,tot为线性基的大小
}
int main() {
    int n;
    cin >> n;
    for (int i=1;i<=n;++i) {
        scanf("%lld",&a[i]);
        insert(a[i]);
    }
    cout << get_max() << endl;
    get_kth();
    count_k();
    return 0;
}

```

- 位运算

```
#include<bits/stdc++.h>
#include <time.h>
using namespace std;
int Count_One(int x) {
    x = (x&0x55555555)+((x>>1)&0x55555555);
    x = (x&0x33333333)+((x>>2)&0x33333333);
    x = (x&0x0F0F0F0F)+((x>>4)&0x0F0F0F0F);
    x = (x&0x00FF00FF)+((x>>8)&0x00FF00FF);
    x = (x&0x0000FFFF)+((x>>16)&0x0000FFFF);
    return x;
}
int main() {
    clock_t _start,_end;
    _start = clock();
    for (int i=1;i<=44040192;++i) Count_One(i);
    _end = clock();
    cout << "time = " << double(_end-_start) / CLOCKS_PER_SEC << endl;
    return 0;
}
```