目录

一、数学

1、线性同余方程

```
/*poj1061**/
#include<cstdio>
#include<iostream>
using namespace std;
typedef long long LL;
/*
计算 gcd(a,b),
同时计算方程 a*x+b*y=gcd(a,b)的一组特解
**/
template<class T> T exgcd(T a, T b, T& x, T& y) {
    if(b == 0) {
        x = 1, y = 0;
        return a;
    }
    T res = exgcd(b, a % b, x, y);
    T t = x;
    x = y, y = t - a / b * y;
    return res;
}

/*
解线性同余方程：ax=b(mod m)
使用性质：
若 c!=0&&ac=bc(mod mc),则 a=b(mod m).
然后使用定理：
若 a 和 m 互素，则方程 ax=b(mod m)在同余意义下恰有一个解。
因为 a 和 m 互素，所以存在整数 s,t 使得 as+mt=1,于是 asb+mtb=b,从而
asb=b(mod m)
令 x=sb,则有 ax=b(mod m).
**/
LL f(LL a, LL b, LL m) {
    LL s, t;
    LL gcd = exgcd(a, m, s, t);
    if(b % gcd != 0)return -1;
```

```
    b /= gcd, m /= gcd;
    s = (s * b) % m;
    s = (s + m) % m;
    return s;
}

int main() {
    LL x, y, m, n, L;
    while(cin >> x >> y >> m >> n >> L) {
        LL res = f((m - n + L) % L, (y - x + L) % L, L);
        if(res == -1)puts("Impossible");
        else printf("%lld\n", res);
    }
    return 0;
}
```

2、中国剩余定理

（1）模数两两互质版本

```
/*poj1370*/
#include<bits/stdc++.h>
using namespace std;
typedef long long LL;

template<class T> void exgcd(T a, T b, T& d, T& x, T& y) {
    if(b) {
        exgcd(b, a % b, d, y, x);
        y -= x * (a / b);
    }
    else d = a, x = 1, y = 0;
}

/*
计算方程组 x%m[i]=r[i]的一个解。
其中，r[i]一定两两互质。
要注意 r[i]全为 0 的情况，输出是 0，但是题目要求可能必须是正数。
**/
LL china(int n, int a[], int m[]) {
```

```
    LL M = 1, d, y, x = 0;
    for(int i = 0; i < n; ++i)M *= m[i];
    for(int i = 0; i < n; ++i) {
        LL w = M / m[i];
        exgcd((LL)m[i], w, d, d, y);
        x = (x + y * w * a[i]) % M;
    }
    return (x + M) % M;
}

int main() {
    int p, e, i, d, meishayong;
    int n, a[5], m[5] = {23, 28, 33};
    scanf("%d", &meishayong);
    for(int T = 1; scanf("%d%d%d%d", &p, &e, &i, &d), (p & e
& i & d) != -1; T++) {
        p %= 23, e %= 28, i %= 33;
        n = 3, a[0] = p, a[1] = e, a[2] = i;
        int res = china(n, a, m);
        if(res == d)printf("Case %d: the next triple peak occurs
in %d days.\n", T, 21252);
        else printf("Case %d: the next triple peak occurs in %d
days.\n", T, (res - d + 21252) % 21252);
    }
    return 0;
}
```

（2）模数不一定两两互质版本

```
/*hdu3579**/
#include<bits/stdc++.h>
using namespace std;
#define MAXN 1010
typedef long long LL;
LL m[MAXN], r[MAXN];
template<class T> void exgcd(T a, T b, T& d, T& x, T& y) {
    if(b) {
        exgcd(b, a % b, d, y, x);
```

```
        y -= x * (a / b);
    }
    else d = a, x = 1, y = 0;
}

/*
计算方程组 x%m[i]=r[i]的一个解。
其中，r[i]不一定两两互质。
要注意 r[i]全为 0 的情况，输出是 0，但是题目要求可能必须是正数。
**/
LL solve(int n) {
    LL M = m[0], R = r[0], x, y, d;
    for(int i = 1; i < n; ++i) {
        exgcd(M, (LL)m[i], d, x, y);
        if((r[i] - R) % d)return -1;
        x = (r[i] - R) / d * x % (m[i] / d);
        R += x * M;
        M = M / d * m[i];
        R %= M;
    }
    if(R < 0)R += M;
    return R;
}
int main() {
    int T, n;
    scanf("%d", &T);
    for(int ca = 1; ca <= T; ca++) {
        scanf("%d", &n);
        for(int i = 0; i < n; ++i)scanf("%lld", m + i);
        for(int i = 0; i < n; ++i)scanf("%lld", r + i);
        LL res = solve(n);
        if(res == 0) {
            LL lcm = 1;
            for(int i = 0; i < n; ++i) {
                lcm = lcm * m[i] / __gcd(lcm, m[i]);
            }
            res = lcm;
        }
```

```
        printf("Case %d: %lld\n", ca, res);
    }
    return 0;
}
```

3、类欧几里得算法

(1) F、G 和 H 的推导代码模板

```
#include<bits/stdc++.h>
using namespace std;
typedef long long LL;
const LL mod = 1e9 + 7;
LL inv2, inv6;

LL qpow(LL a, LL x) {
    LL res = 1;
    for(; x > 0; x >>= 1) {
        if(x & 1)res = (res * a) % mod;
        a = (a * a) % mod;
    }
    return res;
}

namespace semi_eclid {
    struct data {
        LL f, g, h;
    };
    data calc(LL a, LL b, LL c, LL n) {
        data res;
        LL adc = a / c, bdc = b / c;
        if(a == 0) {
            res.f = bdc * (n + 1) % mod;
            res.g = bdc * (n + 1) % mod * n % mod * inv2 % mod;
            res.h = bdc * bdc % mod * (n + 1) % mod * n % mod
* inv2 % mod;
            return res;
        }
        if(a >= c || b >= c) {
```

```
            data s = calc(a % c, b % c, c, n);
            res.f = adc % mod * (n + 1) % mod * n % mod * inv2 %
mod + bdc * (n + 1) % mod + s.f;
            res.f %= mod;
            res.g = adc % mod * (2 * n + 1) % mod * (n + 1) %
mod * n % mod * inv6 % mod
                    + bdc % mod * (n + 1) % mod * n % mod * inv2 %
mod + s.g;
            res.g %= mod;
            res.h = adc * adc % mod * (2 * n + 1) % mod * (n +
1) % mod * n % mod * inv6 % mod
                    + bdc * bdc % mod * (n + 1) % mod + 2 * bdc %
mod * s.f % mod + 2 * adc % mod * s.g % mod
                    + adc * bdc % mod * (n + 1) % mod * n % mod
+ s.h;
            res.h %= mod;
            return res;
        }
        LL m = ((a * n + b) / c) % mod;
        data s = calc(c, c - b - 1, a, m - 1);
        res.f = (n * m % mod - s.f + mod) % mod;
        res.g = (n * m % mod * (n + 1) % mod - s.f - s.h + mod
+ mod) * inv2 % mod;
        res.h = (n * m % mod * (m + 1) % mod - 2 * s.g - 2 * s.f
- res.f) % mod;
        res.h = (res.h + mod) % mod;
        return res;
    }
    LL f(LL a, LL b, LL c, LL n) {
        return calc(a, b, c, n).f;
    }
    LL g(LL a, LL b, LL c, LL n) {
        return calc(a, b, c, n).g;
    }
    LL h(LL a, LL b, LL c, LL n) {
        return calc(a, b, c, n).h;
    }
}
```

```
LL f(LL a, LL b, LL c, LL n) {
    LL res = 0;
    for(int i = 0; i <= n; ++i)res += (a * i + b) / c;
    return res;
}

LL g(LL a, LL b, LL c, LL n) {
    LL res = 0;
    for(int i = 0; i <= n; ++i)res += (a * i + b) / c * i % mod;
    return res;
}

LL h(LL a, LL b, LL c, LL n) {
    LL res = 0;
    for(int i = 0; i <= n; ++i) {
        LL tmp = (a * i + b) / c;
        res += tmp * tmp % mod;
    }
    return res;
}

int random(int m) {
    return (LL)rand() * rand() % m + 1;
}

int main() {
    srand(time(NULL));
    inv2 = qpow(2, mod - 2), inv6 = qpow(6, mod - 2);
    for(int i = 0; i < 500; ++i) {
        int a = random(1e6), b = random(1e6), c = random(1e6),
n = random(1e6);
        LL res1 = semi_eclid::g(a, b, c, n);
        LL res2 = g(a, b, c, n) % mod;
        if(res1 != res2) {
            printf("%d %d %d %d\n", a, b, c, n);
            printf("%lld %lld\n", res1, res2);
            goto flag;
```

```
        }
        //assert(res1==res2);
    }
flag:
    return 0;
}
```

(2) bzoj1938

```
/*
有一个长度为 N 的序列 P 和两种操作，共 Q 个：
1. 给定 L, R, A, B,将第 L 到第 R 个之间的每个元素 Px 变成((X-L+1)×A) mod B。
2. 给定 L, R，询问第 L 到第 R 个元素的和。 数据规模：N≤10^9 ，Q≤50000 ， A, B≤10^6
方法：线段树离散化+类欧几里得算法
*/
#include<bits/stdc++.h>
using namespace std;
typedef long long LL;
#define MAXN 100010
#define lindex(x) (buf[x])
#define rindex(x) (buf[x+1]-1)
int n, m, q;
struct que {
    int t, l, r, a, b;
} qq[MAXN];
int buf[MAXN], bcnt;

struct node {
    LL sum;
    int l, r, a, b;
    node(int l = 0, int r = 0, int a = 0, int b = 0): l(l), r(r), a(a), b(b) {}
    LL f(LL a, LL b, LL c, LL n) {
        if(a == 0)return (b / c) * (n + 1);
        if(a >= c || b >= c)return (a / c) * (n + 1) * n / 2 + (b / c) * (n + 1) + f(a % c, b % c, c, n);
        LL m = (a * n + b) / c;
```

```
        return n * m - f(c, c - b - 1, a, m - 1);
    }
    inline LL sol() {
        LL t1 = f(a, 0, b, r), t2 = (l > 0 ? f(a, 0, b, l - 1) :
0);
        return 1LL * b * (t1 - t2);
    }
    void calc() {
        sum = 1LL * (r + l) * (r - l + 1) / 2 * a - sol();
    }
} ns[MAXN * 4];

inline void pushdown(int rt, int l, int r) {
    if(ns[rt].a != -1) {
        int lch = rt << 1, rch = rt << 1 | 1, mid = (l + r) >>
1;
        ns[lch] = node(ns[rt].l, rindex(mid) + ns[rt].l -
lindex(l), ns[rt].a, ns[rt].b), ns[lch].calc();
        ns[rch] = node(rindex(mid) + 1 + ns[rt].l - lindex(l),
ns[rt].r, ns[rt].a, ns[rt].b), ns[rch].calc();
        ns[rt].a = -1;
    }
}

void update(int ul, int ur, int a, int b, int rt = 1, int l =
1, int r = m) {
    if(l > r || l > ur || r < ul)return;
    if(l >= ul && r <= ur) {
        ns[rt] = node(lindex(l) - lindex(ul) + 1, rindex(r) -
lindex(ul) + 1, a, b);
        ns[rt].calc();
        return;
    }
    pushdown(rt, l, r);
    int mid = (l + r) >> 1;
    if(ul <= mid)update(ul, ur, a, b, rt << 1, l, mid);
    if(ur > mid)update(ul, ur, a, b, rt << 1 | 1, mid + 1, r);
    ns[rt].sum = ns[rt << 1].sum + ns[rt << 1 | 1].sum;
```

```
}

LL query(int ql, int qr, int rt = 1, int l = 1, int r = m) {
    if(l >= ql && r <= qr)return ns[rt].sum;
    pushdown(rt, l, r);
    int mid = (l + r) >> 1;
    LL res = 0;
    if(ql <= mid)res += query(ql, qr, rt << 1, l, mid);
    if(qr > mid)res += query(ql, qr, rt << 1 | 1, mid + 1, r);
    return res;
}

int main() {
    scanf("%d%d", &n, &q);
    for(register int i = 1; i <= q; ++i) {
        scanf("%d%d%d", &qq[i].t, &qq[i].l, &qq[i].r);
        if(qq[i].t == 1) {
            scanf("%d%d", &qq[i].a, &qq[i].b);
            qq[i].a %= qq[i].b;
        }
        buf[++bcnt] = qq[i].l, buf[++bcnt] = qq[i].r + 1;
    }
    buf[++bcnt] = n + 1;
    sort(buf + 1, buf + 1 + bcnt);
    m = unique(buf + 1, buf + 1 + bcnt) - buf - 1;
    for(register int i = 1, ti = m << 2; i < ti; ++i)ns[i].a
= -1;
    for(register int i = 1; i <= q; ++i) {
        int L = lower_bound(buf + 1, buf + 1 + m, qq[i].l) - buf;
        int R = upper_bound(buf + 1, buf + 1 + m, qq[i].r) - buf
- 1;
        if(qq[i].t == 1)update(L, R, qq[i].a, qq[i].b);
        else printf("%lld\n", query(L, R));
    }
    return 0;
}
```

二、数据结构

1、树分治

(1) 例题 1

```
/*统计树上距离不超过 k 的点对，(x,y)和(y,x)只记 1 次。**/
#include<cstdio>
#include<cstring>
#include<cstdlib>
#include<cassert>
#include<iostream>
#include<vector>
#include<algorithm>
using namespace std;
#define MAXN 100010
typedef long long LL;
struct edge {
    int to, next, wt;
    edge(int t = 0, int n = 0, int w = 0): to(t), next(n), wt(w)
{}
} es[MAXN * 2];
int head[MAXN], ecnt;
int n, k;
int siz[MAXN], maxsonsiz[MAXN], G, subn;
vector<int> dis;
bool vis[MAXN];
LL ans;

void add(int from, int to, int wt) {
    es[++ecnt] = edge(to, head[from], wt), head[from] = ecnt;
}

void init() {
    memset(head, 0, sizeof(head[0]) * (n + 3));
    memset(vis, 0, sizeof(vis[0]) * (n + 3));
    ecnt = 0;
    G = 0, subn = n, maxsonsiz[G] = subn;
```

```
}

void dfs4siz(int root, int par) {
    siz[root] = 1, maxsonsiz[root] = 0;
    for(int i = head[root]; i; i = es[i].next) {
        int to = es[i].to;
        if(to != par && !vis[to]) {
            dfs4siz(to, root);
            siz[root] += siz[to];
            maxsonsiz[root] = max(maxsonsiz[root], siz[to]);
        }
    }
    maxsonsiz[root] = max(maxsonsiz[root], subn - siz[root]);
    if(maxsonsiz[root] < maxsonsiz[G])G = root;
}

void dfs4dis(int root, int par, int dd) {
    dis.push_back(dd);
    for(int i = head[root]; i; i = es[i].next) {
        int to = es[i].to;
        if(to != par && !vis[to]) {
            dfs4dis(to, root, dd + es[i].wt);
        }
    }
}

LL calc(int root, int dd) {
    dis.clear();
    dfs4dis(root, 0, dd);
    sort(dis.begin(), dis.end());
    LL res = 0;
    for(int i = 0, j = (int)dis.size() - 1; i < j;) {
        if(dis[i] + dis[j] > k) {
            j--;
        }
        else {
            res += j - i;
            i++;
```

```
        }
    }
    return res;
}

void dfs(int root, int par) {
    dfs4siz(root, par);
    assert(G != 0);
    vis[G] = 1;
    ans += calc(G, 0);
    for(int i = head[G]; i; i = es[i].next) {
        int to = es[i].to;
        if(!vis[to]) {
            ans -= calc(to, es[i].wt);
            G = 0, subn = siz[to], maxsonsiz[G] = subn;
            dfs(to, G);
        }
    }
}

int main() {
    int x, y, z;
    while(scanf("%d%d", &n, &k), n || k) {
        init();
        for(int i = 1; i < n; ++i) {
            scanf("%d%d%d", &x, &y, &z);
            add(x, y, z), add(y, x, z);
        }
        ans = 0;
        dfs(1, 0);
        printf("%lld\n", ans);
    }
    return 0;
}
```

(2) 例题 2

```
/*寻找一对数对，使得路径上点权之积%(1e6+3)等于 k**/
#pragma comment(linker,"/STACK:102400000,102400000")
```

```cpp
#include<bits/stdc++.h>
using namespace std;
#define MAXN 100010
typedef long long LL;
const LL mod = 1e6 + 3;
struct edge {
    int to, next;
    edge(int to = 0, int next = 0): to(to), next(next) {}
} es[MAXN * 2];
int head[MAXN * 2], ecnt;
int n, k, v[MAXN];
namespace math {
#define MAXK 1000003
    LL fact[MAXK], inv[MAXK], finv[MAXK];
    LL qpow(LL a, LL x) {
        LL res = 1;
        for(; x > 0; x >>= 1) {
            if(x & 1)res = (res * a) % mod;
            a = (a * a) % mod;
        }
        return res;
    }
    void init() {
        fact[0] = fact[1] = 1;
        inv[0] = inv[1] = 1;
        for(int i = 2; i < MAXK; i++) {
            fact[i] = fact[i - 1] * i % mod;
            inv[i] = (mod - mod / i) * inv[mod % i] % mod;;
        }
        finv[MAXK - 1] = qpow(fact[MAXK - 1], mod - 2);
        for(int i = MAXK - 2; i >= 0; --i) finv[i] = finv[i +
1] * (i + 1) % mod;
    }
}


void add(int from, int to) {
    es[++ecnt] = edge(to, head[from]), head[from] = ecnt;
```

```
}
void init() {
    memset(head, 0, sizeof(head[0]) * (n + 5));
    ecnt = 0;

}

/*******Divide and conquer***********/
namespace dc {
    int siz[MAXN], maxsonsiz[MAXN], G, subn;
    bool vis[MAXN], ans_flag, min_flag;
    int ansx, ansy;
    struct node {
        int id, group;
        LL prod;
        node(LL prod = 0, int id = 0, int group = 0): prod(prod),
id(id), group(group) {}
        bool operator <(node nd)const {
            if(prod == nd.prod)return id < nd.id;
            else return prod < nd.prod;
        }
    };
    node pro[MAXN];
    int pcnt;
    void init2() {
        G = 0, subn = n, maxsonsiz[G] = subn;
        memset(vis, 0, sizeof(vis[0]) * (n + 3));
        ans_flag = 0, min_flag = 0, ansx = n + 1, ansy = n + 1;
    }
    /*[l,r]*/
    int lb(int l, int r, LL v) {
        int low = l - 1, high = r + 1, mid;
        while(low < high - 1) {
            mid = (low + high) / 2;
            if(pro[mid].prod >= v)high = mid;
            else low = mid;
        }
        return high;
```

```
    }
    int ub(int l, int r, LL v) {
        int low = l - 1, high = r + 1, mid;
        while(low < high - 1) {
            mid = (low + high) / 2;
            if(pro[mid].prod > v)high = mid;
            else low = mid;
        }
        return high;
    }
    void find_center(int root, int par) {
        siz[root] = 1, maxsonsiz[root] = 0;
        for(int i = head[root]; i; i = es[i].next) {
            int to = es[i].to;
            if(to != par && !vis[to]) {
                find_center(to, root);
                siz[root] += siz[to];
                maxsonsiz[root] = max(maxsonsiz[root],
siz[to]);
            }
        }
        maxsonsiz[root] = max(maxsonsiz[root], subn -
siz[root]);
        if(maxsonsiz[root] < maxsonsiz[G])G = root;
    }
    void calc_prod(int root, int par, int gp, node nd) {
        pro[pcnt++] = nd;
        for(int i = head[root]; i; i = es[i].next) {
            int to = es[i].to;
            if(to != par && !vis[to]) {
                if(root == G)gp++;
                calc_prod(to, root, gp, node(nd.prod * v[to] %
mod, to, gp));
            }
        }
    }
    bool cmp4unique(node a, node b) {
        return a.prod == b.prod && a.group == b.group;
```

```
    }
    void find_ans(int center, node nd) {
        pcnt = 0;
        calc_prod(center, 0, 1, nd);
        sort(pro, pro + pcnt);
        pcnt = unique(pro, pro + pcnt, cmp4unique) - pro;
        for(int i = 0; i < pcnt; ++i) {
            node& a = pro[i];
            assert(a.group != 0);
            assert(a.prod < mod);
            LL inv = 1LL * k * (math::inv[a.prod]) % mod *
v[center] % mod;
            int idx1 = lb(i, pcnt - 1, inv);
            int idx2 = ub(i, pcnt - 1, inv);
            for(int it3 = idx1; it3 < idx2; ++it3) {
                node& b = pro[it3];
                if(a.id != b.id && a.group != b.group) {
                    int nx = a.id, ny = b.id;
                    if(nx > ny)swap(nx, ny);
                    ans_flag = 1;
                    if(nx > ansx)continue;
                    else if(nx == ansx && ny < ansy)ansy = ny;
                    else if(nx < ansx) ansx = nx, ansy = ny;
                    if(ansx == 1 && ansy == 2) {
                        min_flag = 1;
                        return;
                    }
                }
            }
        }
    }
    void solve(int root, int par) {
        find_center(root, par);
        assert(G != 0);
        vis[G] = 1;
        find_ans(G, node(v[G], G, 1));
        if(min_flag)return;
        for(int i = head[G]; i; i = es[i].next) {
```

```
            int to = es[i].to;
            if(!vis[to]) {
                G = 0, subn = siz[to], maxsonsiz[G] = subn;
                solve(to, G);
            }
        }
    }
}

/*******Divide and conquer***********/
int main() {
//    freopen("e.in","r",stdin);
    math::init();
    while(~scanf("%d%d", &n, &k)) {
        init();
        for(int i = 1; i <= n; ++i)scanf("%d", v + i);
        for(int i = 1, a, b; i < n; ++i) {
            scanf("%d%d", &a, &b);
            add(a, b), add(b, a);
        }
        dc::init2();
        dc::solve(1, 0);
        if(dc::ans_flag)printf("%d %d\n", dc::ansx,
dc::ansy);
        else puts("No solution");
    }
    return 0;
}
```

(3) 例题 3

```
/*一个图上，源点为点 1 的最短路径树，请问，
在这棵最短路径树上，最长的包含 K 个点的简单路径长度为多长？
长度为该最长长度的不同路径有多少条？ **/
#pragma comment(linker,"/STACK:102400000,102400000")
#include<bits/stdc++.h>
using namespace std;
#define MAXN 30010
#define MAXM 60010
```

```
int n, m, k;
namespace gragh {
    struct edge {
        int from, to, wt, next, used;
        edge(int from = 0, int to = 0, int wt = 0, int next =
0, int used = 0): from(from), to(to), wt(wt), next(next),
used(used) {}
    } es[MAXM * 2];
    struct node {
        int p, d;
        node(int p = 0, int d = 0): p(p), d(d) {}
        bool operator > (node nd)const {
            return d > nd.d;
        }
    };
    int head[MAXN], ecnt;
    int dis[MAXN], par[MAXN];
    bool vis[MAXN];
    priority_queue<node, vector<node>, greater<node> > que;
    void init() {
        memset(head, -1, sizeof(head[0]) * (n + 3));
        ecnt = 0;
    }
    void add(int from, int to, int wt) {
        es[ecnt] = edge(from, to, wt, head[from], 0), head[from]
= ecnt;
        ecnt++;
    }
    void dijk() {
        memset(dis, 0x3f, sizeof(dis[0]) * (n + 3));
        memset(par, -1, sizeof(par[0]) * (n + 3));
        memset(vis, 0, sizeof(vis[0]) * (n + 3));
        while(!que.empty())que.pop();
        dis[1] = 0, que.push(node(1, 0));
        while(!que.empty()) {
            node nd = que.top(); que.pop();
            for(int i = head[nd.p]; ~i; i = es[i].next) {
                edge& e = es[i];
```

```
                int to = e.to;
                if(dis[e.to] > dis[nd.p] + e.wt || (dis[e.to] ==
dis[nd.p] + e.wt && es[par[e.to]].to > nd.p)) {
                    dis[e.to] = dis[nd.p] + e.wt;
                    que.push(node(e.to, dis[e.to]));
                    par[e.to] = i ^ 1;
                }
            }
        }
        for(int i = 1; i <= n; ++i) {
            int from = i;
            while(from != 1) {
                vis[from] = 1;
                es[par[from]].used = 1, es[par[from] ^ 1].used
= 1;
                from = es[par[from]].to;
                if(vis[from])break;
            }
        }

    }
}
namespace tree {
    struct edge {
        int to, wt, next;
        edge(int to = 0, int wt = 0, int next = 0): to(to), wt(wt),
next(next) {}
    } es[MAXN * 2];
    struct node {
        int num, dis, group;
        node(int dis = 0, int num = 0, int group = 0): dis(dis),
num(num), group(group) {}
        bool operator < (node nd)const {
            if(num == nd.num)return dis < nd.dis;
            else return num < nd.num;
        }
    };
    int head[MAXN], ecnt;
```

```
    int siz[MAXN], maxsonsiz[MAXN], G, subn;
    bool vis[MAXN];
    node ns[MAXN];
    int ncnt;
    int maxlen, anscnt;
    void init() {
        memset(head, 0, sizeof(head[0]) * (n + 3));
        memset(vis, 0, sizeof(vis[0]) * (n + 3));
        ecnt = 0;
        maxlen = -1, anscnt = -1;
        G = 0, subn = n, maxsonsiz[G] = subn;
    }
    void add(int from, int to, int wt) {
        es[++ecnt] = edge(to, wt, head[from]), head[from] =
ecnt;
    }
    /*[l,r]*/
    int lb(int l, int r, int num) {
        int low = l - 1, high = r + 1, mid;
        while(low < high - 1) {
            mid = (low + high) / 2;
            if(ns[mid].num >= num)high = mid;
            else low = mid;
        }
        return high;
    }
    int ub(int l, int r, int num) {
        int low = l - 1, high = r + 1, mid;
        while(low < high - 1) {
            mid = (low + high) / 2;
            if(ns[mid].num > num)high = mid;
            else low = mid;
        }
        return high;
    }
    void find_center(int root, int par) {
        siz[root] = 1, maxsonsiz[root] = 0;
        for(int i = head[root]; i; i = es[i].next) {
```

```
            int to = es[i].to;
            if(to != par && !vis[to]) {
                find_center(to, root);
                siz[root] += siz[to];
                maxsonsiz[root] = max(maxsonsiz[root],
siz[to]);
            }
        }
        maxsonsiz[root] = max(maxsonsiz[root], subn -
siz[root]);
        if(maxsonsiz[root] < maxsonsiz[G])G = root;
    }
    void calc_node(int root, int par, int dd, int nn, int gp)
{
        ns[++ncnt] = node(dd, nn, gp);
        for(int i = head[root]; i; i = es[i].next) {
            int to = es[i].to;
            if(to != par && !vis[to]) {
                if(root == G)gp++;
                calc_node(to, root, dd + es[i].wt, nn + 1, gp);
            }
        }
    }
    void find_maxlen(int root, int par) {
        ncnt = 0;
        calc_node(root, par, 0, 1, 1);
        sort(ns + 1, ns + ncnt + 1);
        for(int i = 1; i <= ncnt; ++i) {
            if(ns[i].num > k)break;
            node& a = ns[i];
            int idx1 = lb(i, ncnt, k - a.num + 1);
            int idx2 = ub(i, ncnt, k - a.num + 1);
            for(int j = idx1; j < idx2; ++j) {
                node& b = ns[j];
                if(a.group != b.group) {
                    if(maxlen < a.dis + b.dis) {
                        anscnt = 1;
                        maxlen = a.dis + b.dis;
```

```
                    }
                    else if(maxlen == a.dis + b.dis)anscnt++;
                }
            }
        }
    }
    void solve(int root, int par) {
        find_center(root, par);
        assert(G != 0);
        vis[G] = 1;
        find_maxlen(G, 0);
        for(int i = head[G]; i; i = es[i].next) {
            int to = es[i].to;
            if(!vis[to]) {
                G = 0, subn = siz[to], maxsonsiz[G] = subn;
                solve(to, G);
            }
        }
    }
}

int main() {
//    freopen("f.in","r",stdin);
    int a, b, c;
    while(~scanf("%d%d%d", &n, &m, &k)) {
        gragh::init();
        for(int i = 0; i < m; ++i) {
            scanf("%d%d%d", &a, &b, &c);
            gragh::add(a, b, c), gragh::add(b, a, c);
        }
        gragh::dijk();
        tree::init();
        for(int i = 0; i < gragh::ecnt; ++i) {
            gragh::edge& e = gragh::es[i];
            if(e.used)tree::add(e.from, e.to, e.wt);
        }
        tree::solve(1, 0);
        printf("%d %d\n", tree::maxlen, tree::anscnt);
```

```
    }
    return 0;
}
```

(4) 例题 4

```
/*
题意：给定一棵树，节点个数<=20000，每条边上有一个代价 c 和收益 b，找到一条路径 p 使得该路径上的代价和不超过 C 且收益和最大，输出最大收益。

总体思路：
对于这种树上找两点的，很显然可以用树分治做。
假设当前找出来的重心是 G，然后再算出其他点到 G 的代价和 c 以及相应的收益和 b。
然后，朴素的方法是，在上面 dfs 的时候对每个点打个标记，标记它是哪个子树上的。
这样，对所有点的代价和从小到大排序，枚举每个点 p，再二分找出代价和不超过 C-p.c 的数组下标 idx，
然后扫描 1->idx 之间的所有点 q，如果点 p 和点 q 不同组，那么，更新答案。
但是，这样的方法，最坏的情况，还是会退化成 O(N^2)。
解决办法就是，采用启发式合并(博客正文中已经详细说明)。
使用两个数组，dis 和 tdis。
dis 数组维护的是前 i-1 的子树所有节点的代价和以及在不超过该代价下最大的收益。
要注意看，看仔细，是不超过该代价下的最大的收益，也就是说，在保证 dis 按照代价从小到大有序之后，
还要对 dis 数组的收益 b 做一遍前缀最大值。
tdis 数组维护的是第 i 棵子树所有节点的代价和以及相应的收益。
注意看仔细，tdis 数组是第 i 棵子树中的信息，而且并没有刷过前缀最大值。
有了以上数组以后，每次计算出 tdis 数组以后，枚举 tdis 数组的每个元素 x，
由于 dis 数组已经刷过前缀最大值，所以，只需要二分找出代价和不超过 C-e.c 的元素 y，
然后用 x.b+y.b 更新答案即可。
另外，需要注意的是，如果答案只是一条从重心到子树中某个节点的链，还需要用 tdis 更新一遍答案。
```

最后，使用归并排序的思想归并 dis 数组和 tdis 数组即可，同时记得重新对 dis 数组刷一遍前缀最大值。

```
==========================================================
======================
在此题中，应用此方法的时间复杂度是:O(N*sqrt(N)*logN)
sqrt(N)的原因说简单点就是这种不去重方式的启发式合并，计算量是
O(N*sqrt(N))的，。
==========================================================
======================
**/
#pragma comment(linker,"/STACK:102400000,102400000")
#include<bits/stdc++.h>
using namespace std;
#define MAXN 20010
struct edge {
    int to, next;
    int c, b;
    edge(int to = 0, int next = 0, int c = 0, int b = 0): to(to),
next(next), c(c), b(b) {}
} es[MAXN * 2];
int head[MAXN], ecnt;
int n, C;

void add(int from, int to, int c, int d) {
    es[++ecnt] = edge(to, head[from], c, d), head[from] = ecnt;
}
void init() {
    memset(head, 0, sizeof(head[0]) * (n + 5));
    ecnt = 0;

}
int siz[MAXN], maxsonsiz[MAXN], G, subn;
bool vis[MAXN];
int ans;
/*存重心子节点的编号，重心到子节点的边的编号，以子节点为根的子树的
大小*/
struct node {
```

```
    int id, eid, siz;
    node(int id = 0, int eid = 0, int siz = 0): id(id), eid(eid),
siz(siz) {}
    bool operator <(node nd)const {
        return siz < nd.siz;
    }
} pro[MAXN];
/*存从重心往下搜索路径上消耗的代价和收益*/
struct record {
    int c, b;
    record(int c = 0, int b = 0): c(c), b(b) {}
} dis[MAXN], tdis[MAXN], tmp[MAXN];
int pcnt, dis_siz, tdis_siz, tmp_siz;
/*下面 main 函数中一定要记得调用这个函数初始化*/
void init2() {
    G = 0, subn = n, maxsonsiz[G] = subn;
    memset(vis, 0, sizeof(vis[0]) * (n + 3));
    ans = 0;
    dis_siz = tdis_siz = 0;
}
/*模板：找重心函数*/
void find_center(int root, int par) {
    siz[root] = 1, maxsonsiz[root] = 0;
    for(int i = head[root]; i; i = es[i].next) {
        int to = es[i].to;
        if(to != par && !vis[to]) {
            find_center(to, root);
            siz[root] += siz[to];
            maxsonsiz[root] = max(maxsonsiz[root], siz[to]);
        }
    }
    maxsonsiz[root] = max(maxsonsiz[root], subn - siz[root]);
    if(maxsonsiz[root] < maxsonsiz[G])G = root;
}
/*计算以 root 为根的子树的大小*/
int getsiz(int root, int par) {
    int res = 1;
    for(int i = head[root]; i; i = es[i].next) {
```

```
        int to = es[i].to;
        if(to != par && !vis[to]) {
            res += getsiz(to, root);
        }
    }
    return res;
}
/*
第一句，如果写了，相当剪了枝，那么之前算出了的子树的 siz 就会不对，
那么下面的 find_ans 函数中的 tdis 数组的长度就是 tdis_siz，
否则，tdis_siz=pro[i].siz，随便用哪个都可以。
因此，总的来说，还是写下剪枝好，下面 find_ans 函数中统一用 tdis_siz。
*/
void getdis(int root, int par, record r) {
    if(r.c > C)return;
    tdis[++tdis_siz] = r;
    for(int i = head[root]; i; i = es[i].next) {
        int to = es[i].to;
        if(to != par && !vis[to]) {
            getdis(to, root, record(r.c + es[i].c, r.b +
es[i].b));
        }
    }
}

bool cmp(record a, record b) {
    if(a.c == b.c)return a.b < b.b;
    else return a.c < b.c;
}

int bs(int l, int r, int v) {
    int low = l - 1, high = r + 1, mid;
    while(low < high - 1) {
        mid = (low + high) / 2;
        if(dis[mid].c <= v)low = mid;
        else high = mid;
    }
    return low;
```

```
}

void find_ans(int center, int par) {
    pcnt = 0;
    /*预处理每个儿子节点，然后再启发式合并。*/
    for(int i = head[center]; i; i = es[i].next) {
        int to = es[i].to;
        if(to != par && !vis[to]) {
            pro[++pcnt] = node(to, i, getsiz(to, center));
        }
    }
    sort(pro + 1, pro + pcnt + 1);
    for(int i = 1; i <= pcnt; ++i) {
        tdis_siz = 0;
        getdis(pro[i].id, center, record(es[pro[i].eid].c,
es[pro[i].eid].b));
        if(i > 1) {
            /*对于当前子树的所有 tdis 来说，二分查找出前 i-1 棵子树
的 dis*/
            for(int j = 1; j <= tdis_siz; ++j) {
                int pos = bs(1, dis_siz, C - tdis[j].c);
                if(pos != 0)ans = max(ans, tdis[j].b +
dis[pos].b);
            }
        }
        /*这里要非常注意，如果答案是从重心到子树中某个节点的链，就
用这条语句来更新答案。*/
        for(int j = 1; j <= tdis_siz; ++j)if(tdis[j].c <= C)ans
= max(ans, tdis[j].b);
        /*这里一定要记得排序，因为后面要做启发式合并。*/
        sort(tdis + 1, tdis + tdis_siz + 1, cmp);
        tmp_siz = 0;/*一下是归并排序思想做启发式合并，官方库的
merge 还是不太方便，自己实现最靠谱。*/
        for(int p1 = 1, p2 = 1; p1 <= dis_siz || p2 <= tdis_siz;)
{
            if(p1 <= dis_siz && p2 <= tdis_siz) {
                if(dis[p1].c < tdis[p2].c)tmp[++tmp_siz] =
dis[p1++];
```

```
                else if(dis[p1].c == tdis[p2].c) {
                    if(dis[p1].b < tdis[p2].b)tmp[++tmp_siz] =
dis[p1++];
                    else tmp[++tmp_siz] = tdis[p2++];
                }
                else tmp[++tmp_siz] = tdis[p2++];
            }
            else if(p1 <= dis_siz)tmp[++tmp_siz] = dis[p1++];
            else tmp[++tmp_siz] = tdis[p2++];
        }
        /*这里要非常注意,dis 数组收益 b 需要维护前 i 棵树的前缀和*/
        for(int j = 1; j <= tmp_siz; ++j)dis[j].c = tmp[j].c,
dis[j].b = max(dis[j - 1].b, tmp[j].b);
        dis_siz = tmp_siz;
    }
    dis_siz = 0;
}
/*模板：树分治*/
void solve(int root, int par) {
    find_center(root, par);
    assert(G != 0);
    vis[G] = 1;
    find_ans(G, 0);
    for(int i = head[G]; i; i = es[i].next) {
        int to = es[i].to;
        if(!vis[to]) {
            G = 0, subn = siz[to], maxsonsiz[G] = subn;
            solve(to, G);
        }
    }
}

int main() {
//    freopen("h.in", "r", stdin);
    int T;
    for(scanf("%d", &T); T--;) {
        scanf("%d", &n);
        init();
```

```
        for(int i = 1, a, b, c, d; i < n; ++i) {
            scanf("%d%d%d%d", &a, &b, &c, &d);
            add(a, b, c, d), add(b, a, c, d);
        }
        scanf("%d", &C);
        init2();
        solve(1, 0);
        printf("%d\n", ans);
    }
    return 0;
}
```

(5) 例题 5

```
/*在一颗节点个数为 N(<=200000)的树中，有 M 个拥挤节点，边上有边权，求一条路径 p 使得路径上经过的拥挤节点不超过 k 个，且边权累加和最大，输出最大的边权累加和。
思路：启发式合并。
*/
#pragma comment(linker,"/STACK:102400000,102400000")
#include<bits/stdc++.h>
using namespace std;
#define MAXN 200010
struct edge {
    int to, next, wt;
    edge(int to = 0, int next = 0, int wt = 0): to(to), next(next), wt(wt) {}
} es[MAXN * 2];
int head[MAXN], ecnt;
int n, m, k, crow[MAXN];

void add(int from, int to, int wt) {
    es[++ecnt] = edge(to, head[from], wt), head[from] = ecnt;
}
void init() {
    memset(head, 0, sizeof(head[0]) * (n + 5));
    memset(crow, 0, sizeof(crow[0]) * (n + 5));
    ecnt = 0;
}
```

```cpp
int siz[MAXN], maxsonsiz[MAXN], G, subn;
bool vis[MAXN];
int ans;
int dis[MAXN], tdis[MAXN];

struct node {
    int to, dep, wt;
    node(int to = 0, int dep = 0, int wt = 0): to(to), dep(dep),
wt(wt) {}
    bool operator <(node nd)const {
        return dep < nd.dep;
    }
};
node pro[MAXN];
int pcnt;

void init2() {
    G = 0, subn = n, maxsonsiz[G] = subn;
    memset(vis, 0, sizeof(vis[0]) * (n + 3));
    memset(dis, 0, sizeof(dis[0]) * (n + 3));
    memset(tdis, 0, sizeof(tdis[0]) * (n + 3));
    ans = 0;
}
void find_center(int root, int par) {
    siz[root] = 1, maxsonsiz[root] = 0;
    for(int i = head[root]; i; i = es[i].next) {
        int to = es[i].to;
        if(to != par && !vis[to]) {
            find_center(to, root);
            siz[root] += siz[to];
            maxsonsiz[root] = max(maxsonsiz[root], siz[to]);
        }
    }
    maxsonsiz[root] = max(maxsonsiz[root], subn - siz[root]);
    if(maxsonsiz[root] < maxsonsiz[G])G = root;
}
```

```
int getmaxdep(int root, int par, int dep) {
    if(dep >= k)return k;
    int res = dep;
    for(int i = head[root]; i; i = es[i].next) {
        int to = es[i].to;
        if(to != par && !vis[to]) {
            res = max(res, getmaxdep(to, root, dep + crow[to]));
        }
    }
    return res;
}

void getdis(int root, int par, int crow_num, int path_sum) {
    if(crow_num > k)return;
    tdis[crow_num] = max(tdis[crow_num], path_sum);
    for(int i = head[root]; i; i = es[i].next) {
        int to = es[i].to;
        if(to != par && !vis[to]) {
            getdis(to, root, crow_num + crow[to], path_sum +
es[i].wt);
        }
    }
}

void find_ans(int center, int par) {
    if(crow[center] > k)return;
    pcnt = 0;
    for(int i = head[center]; i; i = es[i].next) {
        int to = es[i].to;
        if(to != par && !vis[to]) {
            pro[++pcnt] = node(to, getmaxdep(to, center,
crow[to] + crow[center]), es[i].wt);
        }
    }
    sort(pro + 1, pro + pcnt + 1);
    for(int i = 1; i <= pcnt; ++i) {
        getdis(pro[i].to, center, crow[pro[i].to] +
crow[center], pro[i].wt);
```

```
        if(i > 1) {
            for(int j = 1; j <= pro[i].dep; ++j)tdis[j] =
max(tdis[j], tdis[j - 1]);
            for(int j = crow[center]; j <= pro[i - 1].dep;
++j)ans = max(ans, dis[j] + tdis[min(pro[i].dep, k - j +
crow[center])]);
        }
        for(int j = 0; j <= pro[i].dep; ++j) {
            dis[j] = max(dis[j], tdis[j]);
            if(j > 0)dis[j] = max(dis[j], dis[j - 1]);
            tdis[j] = 0;
        }
    }
    for(int j = 0; j <= pro[pcnt].dep; ++j)dis[j] = 0;

}

void solve(int root, int par) {
    find_center(root, par);
    assert(G != 0);
    vis[G] = 1;
    find_ans(G, 0);
    for(int i = head[G]; i; i = es[i].next) {
        int to = es[i].to;
        if(!vis[to]) {
            G = 0, subn = siz[to], maxsonsiz[G] = subn;
            solve(to, G);
        }
    }
}

int main() {
//    freopen("j3.in", "r", stdin);
    scanf("%d%d%d", &n, &k, &m);
    init();
    for(int i = 1, a; i <= m; ++i) {
        scanf("%d", &a);
        crow[a] = 1;
```

```
    }
    for(int i = 1, a, b, c; i < n; ++i) {
        scanf("%d%d%d", &a, &b, &c);
        add(a, b, c), add(b, a, c);
    }
    init2();
    solve(1, 0);
    printf("%d\n", ans);
    return 0;
}
```

2、树链剖分(边修改)

```
/*
树上操作:
链修改边权，权值+1
最后按输入顺序输出每条边的边权。
**/
#include<bits/stdc++.h>
using namespace std;
#define MAXN 100010
typedef long long LL;
namespace bit {
    LL n, a[MAXN];
    void init(int an) {
        n = an;
        memset(a, 0, sizeof(a[0]) * (n + 3));
    }
    void update0(int x, int val) {
        for(int i = x; i <= n; i += i & -i)a[i] += val;
    }
    void update(int x, int y) {
        if(x > y)return;
        update0(x, 1), update0(y + 1, -1);
    }
    LL query(int x) {
        LL res = 0;
        for(int i = x; i > 0; i -= i & -i)res += a[i];
```

```
        return res;
    }
}

struct edge {
    int from, to, next;
    edge(int from = 0, int to = 0, int next = 0): from(from),
to(to), next(next) {}
} es[MAXN << 1];
int head[MAXN], ecnt, n;
int f[MAXN], d[MAXN], siz[MAXN], zson[MAXN], dfscnt;
int o2n[MAXN], n2o[MAXN], top[MAXN];

void init() {
    memset(head, 0, sizeof(head[0]) * (n + 5));
    memset(zson, 0, sizeof(zson[0]) * (n + 5));
    ecnt = 0;
    dfscnt = 0;
    bit::init(n);
}

void add(int from, int to) {
    es[++ecnt] = edge(from, to, head[from]), head[from] = ecnt;
}

void dfs1(int root, int par) {
    f[root] = par, d[root] = d[par] + 1, siz[root] = 1;
    int msn = 0;
    for(int i = head[root]; i; i = es[i].next) {
        int to = es[i].to;
        if(to != par) {
            dfs1(to, root);
            siz[root] += siz[to];
            if(siz[to] > msn) {
                msn = siz[to], zson[root] = to;
            }
        }
    }
```

```
}

void dfs2(int root, int par, int tp) {
    o2n[root] = ++dfscnt, n2o[dfscnt] = root, top[root] = tp;
    if(!zson[root])return;
    dfs2(zson[root], root, tp);
    for(int i = head[root]; i; i = es[i].next) {
        int to = es[i].to;
        if(to != zson[root] && to != par) {
            dfs2(to, root, to);
        }
    }
}

void update(int x, int y) {
    int fx = top[x], fy = top[y];
    while(fx != fy) {
        if(d[fx] >= d[fy]) {
            bit::update(o2n[fx], o2n[x]);/*先不+1*/
            x = f[fx], fx = top[x];
        }
        else {
            bit::update(o2n[fy], o2n[y]);/*先不+1*/
            y = f[fy], fy = top[y];
        }
    }
    if(x == y)return;
    if(o2n[x] <= o2n[y])bit::update(o2n[x] + 1, o2n[y]); /*
最后再+1*/
    else bit::update(o2n[y] + 1, o2n[x]); /*最后再+1*/
}

int main() {
    scanf("%d", &n);
    init();
    for(int i = 1, u, v; i < n; ++i) {
        scanf("%d%d", &u, &v);
        add(u, v), add(v, u);
```

```
    }
    dfs1(1, 0);
    dfs2(1, 0, 1);
    int q, u, v;
    for(scanf("%d", &q); q--;) {
        scanf("%d%d", &u, &v);
        update(u, v);
    }
    for(int i = 1; i <= ecnt; i += 2) {
        u = es[i].from, v = es[i].to;
        if(d[u] > d[v])swap(u, v);
        printf("%lld ", bit::query(o2n[v]));
    }
    return 0;
}
```

三、算法

1、回文串的数位 dp(hdu6156)

```
#include<iostream>
#include<cstdio>
#include<cstring>
using namespace std;
typedef long long LL;
/*注意，因为最小进制为 2，所以，pos 最大可能是 30(因为 x 和 y 最大是
1e9)，要确保大小足够。 */
LL dp[66][66][66][2];
int digit[66], dn, bit[66];

template <class T> inline void read(T &x) {
    int t;
    bool flag = false;
    while((t = getchar()) != '-' && (t < '0' || t > '9')) ;
    if(t == '-') flag = true, t = getchar();
    x = t - '0';
    while((t = getchar()) >= '0' && t <= '9') x = x * 10 + t
- '0';
```

```
    if(flag) x = -x;
}

LL dfs(int pos, int base, int start, bool ispd, bool limit) {
    if(pos <= 0)return ispd;
    if(!limit && dp[pos][base][start][ispd] != -1)return
dp[pos][base][start][ispd];
    int top = limit ? digit[pos] : base - 1;
    LL ans = 0;
    for(int i = 0; i <= top; ++i) {
        bit[pos] = i;
        if(pos == start && i == 0)ans += dfs(pos - 1, base, start
- 1, ispd, limit && i == top);
        else {
            int mid = (start + 1) >> 1;
            if(!ispd || pos > mid)ans += dfs(pos - 1, base, start,
ispd, limit && i == top);
            else ans += dfs(pos - 1, base, start, i == bit[start
- pos + 1], limit && i == top);
            /**else 这里的第 4 个参数不需要 ispd && i == bit[start
- pos + 1]，因为只要某位和对应位不相等，
            那么之后的搜索都会从前面的 if 中进去。*/
        }
    }
    if(!limit)dp[pos][base][start][ispd] = ans;
    return ans;
}

LL solve(LL n, int b) {
    dn = 0;
    while(n > 0) {
        digit[++dn] = n % b;
        n /= b;
    }
    return dfs(dn, b, dn, true, true);
}

int main() {
```

```
#ifndef ONLINE_JUDGE
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);
#endif
    LL L, R, l, r, T, ans;
    memset(dp, -1, sizeof(dp));
    scanf("%lld", &T);
    for(LL t = 1; t <= T; ++t) {
        ans = 0;
        read(L), read(R), read(l), read(r);
        for(int b = l; b <= r; ++b) {
            LL ans1 = solve(R, b), ans2 = solve(L - 1, b);
            ans += (ans1 - ans2) * b + (R - L + 1 - (ans1 - ans2));
        }
        printf("Case #%lld: %lld\n", t, ans);
    }
    return 0;
}
```

四、其他

1、Json 解析(2014icpc 牡丹江 H 题)

```
/*
解析 Json 对象。
其中所有的键值对都只有字母和数字组成。
每次给定一个键，回答相应的值。
如果不存在，输出 Error!
**/
#include<bits/stdc++.h>
using namespace std;
#define MAXN 400010
#define mk(x,y) make_pair(x,y)
typedef pair<int, int> pii;
typedef unsigned long long ull;
const ull base = 19260817;
char text[MAXN], key[MAXN];
```

```
unordered_map<ull, int> mp;
unordered_map<ull, int>::iterator it;
pii val[MAXN];
int n, lr[MAXN], nxt[MAXN], vcnt;
stack<int> stk;

inline ull to(char ch) {
    if(isdigit(ch))return ch - '0';
    if(isupper(ch))return ch - 'A' + 10;
    if(islower(ch))return ch - 'a' + 36;
    if(ch == '.')return 62;
    return 63;
}

void dfs(int l, int r, ull initval) {
    ull newval = initval;
    for(int i = l + 1; i <= r; ++i) {
        if(text[i] == ':') {
            if(text[i + 1] == '{') {
                pii p = mk(i + 1, lr[i + 1]);
                mp[newval] = vcnt;
                val[vcnt++] = p;
                dfs(i + 1, lr[i + 1], newval * base + 62);
                i = lr[i + 1];
            }
            else {
                pii p = mk(i + 1, nxt[i] - 1);
                mp[newval] = vcnt;
                val[vcnt++] = p;
                newval = initval;
                i = nxt[i];
            }
        }
        else if(text[i] == ',')newval = initval;
        else if(text[i] == '{' || text[i] == '}')continue;
        else newval = newval * base + to(text[i]);
    }
}
```

```
void parse() {
    n = strlen(text);
    mp.clear();
    vcnt = 0;
    memset(lr, -1, sizeof(lr[0]) * (n + 5));
    memset(nxt, -1, sizeof(nxt[0]) * (n + 5));
    while(!stk.empty())stk.pop();
    for(int i = 0; i < n; ++i) {
        if(text[i] == '{')stk.push(i);
        else if(text[i] == '}') {
            int l = stk.top(); stk.pop();
            lr[l] = i;
        }
    }
    for(int i = n - 1, last = n; i > 0; --i) {
        char c = text[i];
        if(c == ':' || c == ',' || c == '{' || c == '}') {
            nxt[i] = last;
            last = i;
        }
    }
    dfs(0, n - 1, 0);
}

void sol() {
    int q;
    scanf("%s", text);
    parse();
    for(scanf("%d", &q); q--;) {
        scanf("%s", key);
        ull keyval = 0;
        for(int i = 0, len = strlen(key); i < len; ++i)keyval
= keyval * base + to(key[i]);
        it = mp.find(keyval);
        if(it == mp.end())puts("Error!");
        else {
            pii& pp = val[it->second];
```

```
            for(int i = pp.first; i <= pp.second;
++i)putchar(text[i]); putchar('\n');
        }
    }
}
int main() {
//    freopen("h.in","r",stdin);
    int T;
    for(scanf("%d", &T); T--;)sol();
    return 0;
}
```