【金牌训练】部分题目完整代码

目　录

第五部分　树上问题

第六部分　技巧与思想

第七部分　计算几何

第八部分　数　论

第一部分　动态规划

第 1 章　概率期望 DP

453.树上移动(C,1s,256MB)

【参考程序】

```
#include <bits/stdc++.h>
template <class T>
inline void read(T &res)
{
    char ch;
    while (ch = getchar(), !isdigit(ch));
    res = ch ^ 48;
    while (ch = getchar(), isdigit(ch))
        res = res * 10 + ch - 48;
}

using std::vector;
const int mod = 1e9 + 7;
const int N = 1e5 + 5;
int inv[N], f[N], sze[N], g[N];
int n, cnt, ans; char s[N];
vector<int> e[N];

inline int quick_pow(int x, int k)
{
    int res = 1;
    while (k)
    {
        if (k & 1)
            res = 1ll * res * x % mod;
        x = 1ll * x * x % mod;
        k >>= 1;
    }
    return res;
}

inline void add(int &x, int y)
{
    x += y;
    x >= mod ? x -= mod : 0;
}
```

```
inline void dec(int &x, int y)
{
    x -= y;
    x < 0 ? x += mod : 0;
}

struct point
{ //定义三元组 (a,b,c) 表示 af[i][0] + bf[i][1] + c，方便运算
    int a, b, c;

    point() {}
    point(int A, int B, int C):
        a(A), b(B), c(C) {}

    inline void operator += (const point &x)
    {
        add(a, x.a);
        add(b, x.b);
        add(c, x.c);
    }

    inline void operator -= (const point &x)
    {
        dec(a, x.a);
        dec(b, x.b);
        dec(c, x.c);
    }

    inline point operator * (const int &x) const
    {
        return point(1ll * a * x % mod,
                     1ll * b * x % mod,
                     1ll * c * x % mod);
    }
}val[N][2];

inline void Dfs1(int x)
{  //通过树形 DP 求出每个点到所有点的距离之和
    sze[x] = 1;
    for (int i = 0, im = e[x].size(); i < im; ++i)
    {
        int y = e[x][i];
        Dfs1(y);
        sze[x] += sze[y];
```

```
        add(f[x], f[y]);
        add(f[x], sze[y]);
    }
}

inline void Dfs2(int x)
{
    int res = 0;
    for (int i = 0, im = e[x].size(); i < im; ++i)
    {
        int y = e[x][i];
        g[y] = f[x];
        dec(g[y], sze[y]);
        dec(g[y], f[y]);
        add(g[y], g[x]);
        add(g[y], n - sze[y]);
        Dfs2(y);
    }
    add(f[x], g[x]);
}

int main()
{
    freopen("C.in", "r", stdin);
    freopen("C.out", "w", stdout);
    read(n);
    scanf("%s", s + 1);
    for (int i = 1; i <= n; ++i)
        cnt += s[i] == '1';
    for (int i = 2, fa; i <= n; ++i)
    {
        read(fa);
        e[fa].push_back(i);
    }
    inv[1] = 1;
    for (int i = 2; i <= n; ++i)
        inv[i] = 1ll * (mod - mod / i) * inv[mod % i] % mod;
    Dfs1(1);
    Dfs2(1);

    val[0][0] = val[0][1] = point(0, 0, 0);
    val[1][0] = point(1, 0, 0);
    val[1][1] = point(0, 1, 0);
    for (int i = 1; i < n; ++i)
```

```
    {
        point &tmp1 = val[i + 1][1];
        tmp1 = val[i][1];
        tmp1 -= val[i - 1][1] * (1ll * (i - 1) * inv[n] % mod);
        tmp1 -= val[i - 1][0] * inv[n];
        if (i > 1)
            dec(tmp1.c, inv[n]);
        tmp1 = tmp1 * (1ll * n * inv[n - i] % mod);

        point &tmp2 = val[i + 1][0];
        tmp2 = val[i][0];
        tmp2 -= val[i - 1][0] * (1ll * i * inv[n] % mod);
        tmp2 -= val[i + 1][1] * inv[n];
        if (i < n - 1)
        {
            dec(tmp2.c, inv[n]);
            tmp2 = tmp2 * (1ll * n * inv[n - i - 1] % mod);
        }
    }
    point tmp1 = val[n][0],
          tmp2 = val[n][1];
    tmp1.c = 1ll * (mod - 1) * tmp1.c % mod;
    tmp2.c = 1ll * (mod - 1) * tmp2.c % mod;
    tmp1 = tmp1 * (1ll * tmp2.a * quick_pow(tmp1.a, mod - 2) % mod);

    int y = 1ll * (tmp2.c + mod - tmp1.c) * quick_pow(tmp2.b + mod -
tmp1.b, mod - 2) % mod;
    int x = 1ll * (tmp2.c + mod - 1ll * tmp2.b * y % mod) *
quick_pow(tmp2.a, mod - 2) % mod;

    tmp1 = val[cnt][0];
    tmp2 = val[cnt][1];
    int v0 = (1ll * tmp1.a * x + 1ll * tmp1.b * y + tmp1.c) % mod,
        v1 = (1ll*tmp2.a * x + 1ll * tmp2.b * y + tmp2.c) % mod;
    add(v0, inv[n]);
    add(v1, inv[n]);
    for (int i = 1; i <= n; ++i)
        ans = (1ll*f[i]*inv[n]%mod*(s[i] == '1' ? v1 : v0) + ans) % mod;
    printf("%d\n", ans);
    fclose(stdin); fclose(stdout);
    return 0;
}
```

第 2 章　数据结构优化 DP

461.区间覆盖(xmasinterval,3s,512MB)

【参考程序】

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

using namespace std;

#define p2 p << 1
#define p3 p << 1 | 1
#define ll long long

template <class t>
inline void read(t & res)
{
    char ch;
    while (ch = getchar(), !isdigit(ch));
    res = ch ^ 48;
    while (ch = getchar(), isdigit(ch))
    res = res * 10 + (ch ^ 48);
}

const int e = 5e5 + 5, mod = 1e9 + 9;
struct point
{
    int l, r;
}a[e];
int n, m, b[e], ans, mul[e * 4], add[e * 4], c[e];

inline int plu(int x, int y)
{
    x += y;
    if (x >= mod) x -= mod;
    return x;
}

inline void pushdown(int l, int r, int p)
{
    if (mul[p]) //mul[p]表示区间[l,r]的每个位置都乘上 2^mul[p]
    {
        mul[p2] += mul[p];
        mul[p3] += mul[p];
        int x = c[mul[p]];
        add[p2] = (ll)add[p2] * x % mod;
```

```
        add[p3] = (ll)add[p3] * x % mod;
        mul[p] = 0;
    }
    if (add[p]) //add[p]表示区间[l,r]的每个位置都加上 add[p]
    {
        int mid = l + r >> 1;
        add[p2] = plu(add[p2], add[p]);
        add[p3] = plu(add[p3], add[p]);
        add[p] = 0;
    }
}

inline void build(int l, int r, int p)
{
    if (l == r)
    {
        if (!l) add[p] = 1; //初始位置 0 的值为 1
        return;
    }
    int mid = l + r >> 1;
    build(l, mid, p2);
    build(mid + 1, r, p3);
}

inline void update(int l, int r, int s, int t, int p) //区间乘 2
{
    if (l == s && r == t)
    {
        mul[p]++;
        add[p] = plu(add[p], add[p]);
        return;
    }
    pushdown(l, r, p);
    int mid = l + r >> 1;
    if (t <= mid) update(l, mid, s, t, p2);
    else if (s > mid) update(mid + 1, r, s, t, p3);
    else
    {
        update(l, mid, s, mid, p2);
        update(mid + 1, r, mid + 1, t, p3);
    }
}

inline void modify(int l, int r, int s, int t, int v, int p)//区间加
```

```
{
    if (l == s && r == t)
    {
        add[p] = plu(add[p], v);
        return;
    }
    pushdown(l, r, p);
    int mid = l + r >> 1;
    if (t <= mid) modify(l, mid, s, t, v, p2);
    else if (s > mid) modify(mid + 1, r, s, t, v, p3);
    else
    {
        modify(l, mid, s, mid, v, p2);
        modify(mid + 1, r, mid + 1, t, v, p3);
    }
}

inline int query(int l, int r, int s, int p) //查询位置 s 的值
{
    if (l == r) return add[p];
    int mid = l + r >> 1;
    pushdown(l, r, p);
    if (s <= mid) return query(l, mid, s, p2);
    else return query(mid + 1, r, s, p3);
}

inline bool cmp(const point &a, const point &b)
{
    return a.l < b.l;
}

int main()
{
    freopen("xmasinterval.in", "r", stdin);
    freopen("xmasinterval.out", "w", stdout);
    int i;
    read(n); read(m);
    for (i = 1; i <= n; i++) read(a[i].l), read(a[i].r);
    for (i = 1; i <= m; i++) read(b[i]);
    c[0] = 1;
    for (i = 1; i <= m; i++) c[i] = 2ll * c[i - 1] % mod; //预处理 2 的幂
    sort(b + 1, b + m + 1);
    m = unique(b + 1, b + m + 1) - b - 1;
    for (i = 1; i <= n; i++)
```

```
    {
        if (a[i].l <= b[m]) a[i].l = lower_bound(b+1, b+m+1, a[i].l)-b;
        else a[i].l = m + 1;
        if (a[i].r >= b[1]) a[i].r = upper_bound(b+1, b+m+1, a[i].r)-b-1;
        else a[i].r = 0; //二分查找
    }
    sort(a + 1, a + n + 1, cmp);
    build(0, m, 1);
    for (i = 1; i <= n; i++)
    {
        int l = a[i].l, r = a[i].r;
        if (l > r) ans++;           //跳过覆盖不到任何关键点的区间
        int s = query(0, m, l - 1, 1);
        modify(0, m, 0, r, s, 1);  //位置 0~r 的值都加 s
        if (r != m) update(0, m, r + 1, m, 1); //位置 r+1~m 的值都乘 2
    }
    ans = (ll)c[ans] * query(0, m, m, 1) % mod;
    cout << ans << endl;
    fclose(stdin); fclose(stdout);
    return 0;
}
```

462.区间计数(interval,1s,25MB)

【参考程序】

```
#include <bits/stdc++.h>
#define For(i, a, b) for (int i = a, bb = b; i <= bb; ++i)
#define Rof(i, a, b) for (int i = a, bb = b; i >= bb; --i)
#define s64 long long

template <class T>
inline void get(T &res)
{
    char ch;
    bool bo = false;
    while ((ch = getchar()) < '0' || ch > '9')
        if (ch == '-') bo = true;

    res = ch - '0';
    while ((ch = getchar()) >= '0' && ch <= '9')
        res = (res << 1) + (res << 3) + ch - '0';

    if (bo) res = ~ res + 1;
    return;
}
```

```
template <class T>
inline void _put(T x)
{
    if (x > 9) _put(x / 10);
    putchar(x % 10 + '0');
    return;
}

template <class T>
inline void put(T x, char ch)
{
    if (x < 0)
    {
        putchar('-');
        x = ~ x + 1;
    }
    _put(x);
    putchar(ch);

    return;
}

#define k2 k << 1
#define k3 k << 1 | 1

const int MaxN = 3e5 + 5, INF = 0x3f3f3f3f;
int a[MaxN], pos[MaxN], tag[MaxN << 2], n;
s64 ans;
struct tree //线段树记录区间最小值、区间次小值以及出现次数
{
    int val1, cnt1, val2, cnt2;

    inline tree operator + (tree rhs)
    {
        tree lhs = *this;
        if (lhs.val1 > rhs.val1) lhs = rhs, rhs = *this;
        if (lhs.val2 < rhs.val1) return lhs;
        if (rhs.val1 == lhs.val2)
        {
            lhs.cnt2 += rhs.cnt1;
            return lhs;
        }
        else if (rhs.val1 == lhs.val1)
        {
```

```
            lhs.cnt1 += rhs.cnt1;
            if (rhs.val2 < lhs.val2)
                return tree{lhs.val1, lhs.cnt1, rhs.val2, rhs.cnt2};
            else
            {
                if (rhs.val2 == lhs.val2)
                    lhs.cnt2 += rhs.cnt2;
                return lhs;
            }
        }
        else
            return tree {lhs.val1, lhs.cnt1, rhs.val1, rhs.cnt1};
    }
} tr[MaxN << 2];

inline void build(int k, int l, int r)
{
    if (l == r)
    {
        tr[k] = tree {0, 1, INF, 1};
        return;
    }

    int mid = l + r >> 1;
    build(k2, l, mid);
    build(k3, mid + 1, r);
    tr[k] = tr[k2] + tr[k3];
    return;
}

inline void marktag(int k, int v)
{
    tag[k] += v;
    tr[k].val1 += v, tr[k].val2 += v;
    return;
}

inline void downtag(int k)
{
    if (!tag[k]) return;
    marktag(k2, tag[k]);
    marktag(k3, tag[k]);
    tag[k] = 0;
    return;
```

```
}

inline void modify(int k, int l, int r, int x, int y, int v)
{
    if (x <= l && r <= y)
        return marktag(k, v);

    downtag(k);
    int mid = l + r >> 1;
    if (x <= mid) modify(k2, l, mid, x, y, v);
    if (y > mid) modify(k3, mid + 1, r, x, y, v);
    tr[k] = tr[k2] + tr[k3];
    return;
}

inline tree query(int k, int l, int r, int x) //询问区间[x,n]的答案
{
    if (x <= l)
        return tr[k];

    downtag(k);
    int mid = l + r >> 1;
    return x <= mid ? query(k2, l, mid, x) + tr[k3] : query(k3, mid + 1,
r, x);
}

int main()
{
    freopen("interval.in", "r", stdin);
    freopen("interval.out", "w", stdout);

    get(n);
    For(i, 1, n)
        get(a[i]);
    For(i, 1, n)
        pos[a[i]] = i;
    build(1, 1, n);

    Rof(now, n, 1)
    {
        int L = a[pos[now] - 1], R = a[pos[now] + 1];
        if (L > R) std::swap(L, R);
        //L、R 分别表示在 P 中位置与 now 相邻的两个数
```

```
        if (R < now) //分类讨论加入数字 now 对连通块个数的贡献
            modify(1, 1, n, now, n, 1);
        else if (L < now)
            modify(1, 1, n, now, R - 1, 1);
        else
            modify(1, 1, n, now, L - 1, 1),
            modify(1, 1, n, R, n, -1);

        if (now == n)
            continue;
        tree x = query(1, 1, n, now + 1);
        if (x.val1 <= 2) //组成区间[now,now+1...n]的方案数计入答案
        {
            ans += x.cnt1;
            if (x.val2 == 2) ans += x.cnt2;
        }
    }
    put(ans, '\n');

    fclose(stdin), fclose(stdout);
    return 0;
}
```

463.序列划分（sequence,3s,256MB）

【参考程序】

```
#include <set>
#include <cmath>
#include <cstdio>
#include <cctype>
#include <cstdlib>
#include <cstring>
#include <iostream>
#include <algorithm>

typedef long long s64;

template <class T>
inline void read(T &x)
{
    static char ch;
    static bool opt;
    while (!isdigit(ch = getchar()) && ch != '-');
    x = (opt = ch == '-') ? 0 : ch - '0';
    while (isdigit(ch = getchar()))
        x = x * 10 + ch - '0';
```

```
    if (opt) x = ~x + 1;
}

const int MaxN = 1e5 + 5;
const int MaxLog = 18;
const int INF = 0x3f3f3f3f;

std::multiset<s64> min_set;

int n, tot;
int logval[MaxN], maxa[MaxN], minb[MaxLog + 1][MaxN];
//maxa[i]记录的是 a 的后缀最大值的序号
//minb[k][i]记录的是[i,i+2^k-1]b 的最大值
s64 m, a[MaxN], b[MaxN], s[MaxN], f[MaxN];

inline int maxpos(const int &x, const int &y)
{
    return a[x] > a[y] ? x : y;
}

inline int getminb(const int &l, const int &r)
{
    int k = logval[r - l + 1];
    return std::min(minb[k][l], minb[k][r - (1 << k) + 1]);
}

inline bool check(const s64 &mid)
{
    min_set.clear();
    memset(f, 0x3f, sizeof(f));

    static int q[MaxN], head, tail;
    q[head = tail = 1] = 0;
    f[0] = 0;

    int le = 0;
    //le 指向第一个满足 s[i]-s[j]<=mid 的 j
    for (int i = 1; i <= n; ++i)
    {
        while (head <= tail && s[i] - s[q[head]] > mid)
        {   //弹出队首不满足 s[i]-s[j]<=mid 的决策
            if (head < tail)
                min_set.erase(min_set.find(f[q[head]] + a[q[head + 1]]));
            ++head;
```

```
        }
        while (head <= tail && a[q[tail]] < a[i])//维护队列 a 的不增性
        {
            if (head < tail)
               min_set.erase(min_set.find(f[q[tail-1]]+a[q[tail]]));
            --tail;
        }
        if (head <= tail)
            min_set.insert(f[q[tail]]+a[i]);//维护原队尾的决策转移值
        q[++tail] = i;

        while (le < i && s[i] - s[le] > mid)
            ++le;    //维护 le 指针，用满足条件二的决策更新 f[i]
        if (le == q[head])    //若 le 指向队首元素
            f[i] = f[le] + a[q[head+1]];//则最大值应取队首后一个元素
        else
            f[i] = f[le] + a[q[head]]; //否则最大值取队首元素
        if (head < tail)  //查询满足条件一的最优决策更新 f[i]
            f[i] = std::min(f[i], *min_set.begin());
    }
    return f[n] <= m;
}

int main()
{
    freopen("sequence.in", "r", stdin);
    freopen("sequence.out", "w", stdout);

    read(n), read(m);
    logval[0] = -1;
    for (int i = 1; i <= n; ++i)
    {
        read(a[i]), read(b[i]);
        logval[i] = logval[i >> 1] + 1;
        minb[0][i] = b[i];
    }
    maxa[n] = n;
    for (int i = n - 1; i >= 0; --i)
        maxa[i] = maxpos(maxa[i + 1], i);
    //a 的后缀最大值预处理
    for (int j = 1; j <= MaxLog; ++j)
        for (int i = 1; i + (1 << j) - 1 <= n; ++i)
            minb[j][i] = std::min(minb[j-1][i],minb[j-1][i + (1 << j-1)]);
    //b 的 RMQ 预处理
```

```
    for (int i = 1; i <= n;)
    {
        int j;
        for (j = i; j < n && a[maxa[j+1]] >= getminb(i,j); j = maxa[j+1]);

        a[++tot] = a[i];
        b[tot] = b[i];
        for (i = i + 1; i <= j; ++i)
        {
            a[tot] = std::max(a[tot], a[i]);
            b[tot] += b[i];
        }
    }
    n = tot;
    //将必定在同一段的每个区间缩成数对
    s64 l = 0, r = 0, mid, res = 0;
    for (int i = 1; i <= n; ++i)
    {
        s[i] = s[i - 1] + b[i];
        l = std::max(l, b[i]);
        r += b[i];
    }
    while (l <= r)
    {
        mid = l + r >> 1;
        if (check(mid))
            r = mid - 1, res = mid;
        else
            l = mid + 1;
    }
    //二分答案
    std::cout << res << std::endl;

    fclose(stdin); fclose(stdout);
    return 0;
}
```

第 5 章　斜率优化 DP

491.太空飞船(spaceship,1s,256MB)

【参考程序】

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

template <class T>
inline void read(T &res)
{
    char ch; bool flag = false; res = 0;
    while (ch = getchar(), !isdigit(ch) && ch != '-');
    ch == '-' ? flag = true : res = ch ^ 48;
    while (ch = getchar(), isdigit(ch))
        res = res * 10 + ch - 48;
    flag ? res = -res : 0;
}

typedef long long ll;
const ll Maxn = 1ll << 62;
const int N = 3e5 + 5;
int a[N], n, K; ll s[N], ans = Maxn;

template <class T>
inline void CkMin(T &x, T y) {if (x > y) x = y;}
template <class T>
inline T Max(T x, T y) {return x > y ? x : y;}
template <class T>
inline T Min(T x, T y) {return x < y ? x : y;}
template <class T>
inline T Sqr(T x) {return x * x;}

inline void work1()
{
    for (int i = 1; i <= n; ++i)
        a[i + n] = a[i];
    for (int i = 1, im = n << 1; i <= im; ++i)
        s[i] = s[i - 1] + a[i];
    int j = 1; ll tmp = s[n] >> 1;
    while (s[j + 1] - s[1] <= tmp) ++j;
    CkMin(ans, Sqr(s[j] - s[1]) + Sqr(s[n] - s[j] + s[1]));
    for (int i = 2; i <= n; ++i)
    { //考虑分成 j+1 到 i，i+1 到 j 这 2 段
        while (s[j + 1] - s[i] <= tmp) ++j;
        CkMin(ans, Sqr(s[j] - s[i]) + Sqr(s[n] - s[j] + s[i]));
        CkMin(ans,Sqr(s[j-1] - s[i]) + Sqr(s[n] - s[j-1] + s[i]));
        //边界判断
    }
```

```
    cout << 1ll * K * K * ans - 1ll * K * s[n] * s[n] << endl;
}

ll f[25][805]; int h[805];

inline bool Slope1(int i, int j, int k, ll si)
{
   return (f[i][j]+s[j]*s[j]-f[i][k]-s[k]*s[k])>=2ll*si*(s[j]-s[k]);
}

inline bool Slope2(int i, int j, int k, int l) //维护凸壳斜率递增
{ //比较斜率时把除法转成乘法，避免精度误差
    return (f[i][j]+s[j]*s[j]-f[i][k]-s[k]*s[k]) * (s[k]-s[l])
         > (f[i][k]+s[k]*s[k]-f[i][l]-s[l]*s[l]) * (s[j]-s[k]);
}

inline void work3()
{
    ll ans = Maxn;
    for (int i = 1; i <= n; ++i)
        a[i + n] = a[i];
    for (int i = 1, im = n << 1; i <= im; ++i)
        s[i] = s[i - 1] + a[i];
    for (int st = 1; st <= n; ++st)
    {
        for (int i = 0; i <= K; ++i)
            for (int j = 0, jm = n << 1; j <= jm; ++j)
                f[i][j] = Maxn;
        f[0][st - 1] = 0;
        for (int i = 1; i <= K; ++i) //斜率优化 DP
        {
            int t = 1, w = 0;
            if (f[i - 1][st + i - 2] != Maxn)
                h[++w] = st + i - 2;
            for (int j = i-1; j < n; ++j)  //注意要保证转移合法
            {
                 while(t<w && Slope1(i-1,h[t],h[t+1],s[st+j]))
                    ++t;
                if (t <= w)
                   f[i][st + j] = f[i-1][h[t]]
                      +(s[st+j]-s[h[t]])*(s[st+j]-s[h[t]]);
                if (f[i-1][st+j] != Maxn)
                {
                   while(t<w && Slope2(i-1,h[w-1],h[w], st+j)) --w;
                   h[++w] = st + j;
                }
            }
        }
        CkMin(ans, f[K][st + n - 1]);
```

```
    }
    cout << 1ll * K * K * ans - 1ll * K * s[n] * s[n] << endl;
}

inline ll calc(int j, int i, int k)
{   //计算分成 i+1 到 j，j+1 到 k，k+1 到 i 这 3 段的结果
    ll tx = s[j] + s[n] - s[k],
       ty = s[i] - s[j],
       tz = s[k] - s[i];
    return tx * tx + ty * ty + tz * tz;
}

inline void work2()
{
    int j = 0, k = 2;
    for (int i = 1; i <= n; ++i)
        s[i] = s[i - 1] + a[i];
    ll tmp = s[n] / 3;
    for (int i = 1; i <= n; ++i)
    {
        while (j < i && s[i] - s[j] > tmp) ++j;
        while (k <= n && s[k] - s[i] <= tmp) ++k;
        for (int a = Max(j-1, 0), am = Min(j+1, i); a <= am; ++a)
            for (int b = Max(k-1,i), bm=Min(k+1,n); b<=bm; ++b)
               CkMin(ans, calc(a, i, b));  //边界判断
    }
    cout << 1ll * K * K * ans - 1ll * K * s[n] * s[n] << endl;
}

int main()
{
    read(n); read(K);
    for (int i = 1; i <= n; ++i) read(a[i]);

    if (K == 2)
        work1();
    else if (n <= 400)
        work3();
    else if (K == 3)
        work2();
    return 0;
}
```

第 6 章　插头 DP

502.标识设计(logo,2s,524MB)

【参考程序】

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

const int N = 32;
const int M = 4530;
typedef long long ll;
char p[N][N]; bool apr[M][N];
int cnt[M], g[N][N][N], add[M][N], del[M][N];
int a[4], b[4];
ll f[N][M][4][4], h[M][4][3], ans;
int n, m, G, bm, am;

int main()
{
    freopen("logo.in", "r", stdin);
    freopen("logo.out", "w", stdout);

    scanf("%d%d", &n, &m);
    for (int i = 1; i <= n; ++i)
        scanf("%s", p[i] + 1);

    g[0][0][0] = ++G; //给所有状态编号
    for (int i = 1; i <= m; ++i)
    {
        g[i][0][0] = ++G;
        for (int j = i + 1; j <= m; ++j)
        {
            g[i][j][0] = ++G;
            for (int k = j + 1; k <= m; ++k)
                g[i][j][k] = ++G;
        }
    }

    //预处理出每个状态在某个位置填上“1”以及删去原来的某个“1”得到的状态编号
    for (int i = 0; i <= m; ++i)
        for (int j = 0; j <= m; ++j)
            for (int k = 0; k <= m; ++k)
                if (g[i][j][k])
                {
                    int x = g[i][j][k];
                    am = 0;
```

```
                a[1] = a[2] = a[3] = 0;
                if (i)
                    a[++am] = i, apr[x][i] = true;
                if (j)
                    a[++am] = j, apr[x][j] = true;
                if (k)
                    a[++am] = k, apr[x][k] = true;
                cnt[x] = am;
                if (am < 3)
                {
                    for (int l = 1; l <= m; ++l)
                        if (!apr[x][l])
                        {
                            bm = am;
                            b[1] = b[2] = b[3] = 0;
                            for (int r = 1; r <= am; ++r)
                                b[r] = a[r];
                            b[++bm] = l;
                            std::sort(b + 1, b + bm + 1);
                            add[x][l] = g[b[1]][b[2]][b[3]];
                        }
                }
                for (int l = 1; l <= am; ++l)
                {
                    bm = 0;
                    b[1] = b[2] = b[3] = 0;
                    for (int r = 1; r <= am; ++r)
                        if (r != l)
                            b[++bm] = a[r];
                    std::sort(b + 1, b + bm + 1);
                    del[x][a[l]] = g[b[1]][b[2]][b[3]];
                }
            }
f[1][g[0][0][0]][0][0] = 1;
if (p[1][1] != '#')
    f[1][g[1][0][0]][0][1] = 1;
for (int i = 1; i <= n; ++i)
{
    for (int j = 1; j < m; ++j)
        for (int k = 0; k <= 3; ++k)
            for (int s = 1, sm = k == 3 ? 1 : G; s <= sm; ++s)
            {
                if (k + cnt[s] > 3)
                    continue;
```

```
            for (int c = 0; c <= 3; ++c)
                if (f[j][s][k][c])
                {
                    ll tmp = f[j][s][k][c];
                    if (apr[s][j + 1])
                    {
                        if (c == 2 || p[i][j + 1] == '#')
                            continue;
                        int nk = k + (c == 3);
                        if (nk > 3)
                            continue;
                        f[j + 1][s][nk][1] += tmp;
                        f[j + 1][del[s][j + 1]][nk][2] += tmp;
                    }
                    else if (c != 2)
                    {
                        if (c == 3 && p[i][j + 1] != '#')
                            f[j + 1][s][k][3] += tmp;
                        int nk = k + (c == 3);
                        if (nk > 3)
                            continue;
                        f[j + 1][s][nk][0] += tmp;
                        if (cnt[s] < 3 && p[i][j + 1] != '#')
                            f[j + 1][add[s][j + 1]][nk][1] += tmp;
                    }
                    else if (p[i][j + 1] != '#')
                        f[j + 1][s][k][3] += tmp;
                }
        }
for (int k = 0; k <= 2; ++k)
    for (int s = 1; s <= G; ++s)
        if (f[m][s][k][3])
        { //特别处理每一行最后一个方格状态为"3"的情况
            f[m][s][k + 1][0] += f[m][s][k][3];
            f[m][s][k][3] = 0;
        }

if (i == n)
    ans = f[m][1][3][0];
else
{ //特别处理每一行第一个方格的转移
    memset(h, 0, sizeof(h));
    for (int k = 0; k <= 3; ++k)
        for (int s = 1, sm = k == 3 ? 1 : G; s <= sm; ++s)
```

```
                    if (f[m][s][k][0])
                    {
                        ll tmp = f[m][s][k][0];
                        if (apr[s][1])
                        {
                            if (p[i + 1][1] == '#')
                                continue;
                            h[del[s][1]][k][2] += tmp;
                            h[s][k][1] += tmp;
                        }
                        else
                        {
                            h[s][k][0] += tmp;
                            if (p[i + 1][1] != '#')
                                h[add[s][1]][k][1] += tmp;
                        }
                    }
            memset(f, 0, sizeof(f));
            for (int k = 0; k <= 3; ++k)
                for (int s = 1; s <= G; ++s)
                    for (int c = 0; c <= 2; ++c)
                        f[1][s][k][c] = h[s][k][c];
        }
    }
    std::cout << ans << std::endl;


    fclose(stdin); fclose(stdout);
    return 0;
}
```

503.管道连接(game,2s,512MB)

【参考程序】

```
#include <bits/stdc++.h>
using namespace std;
int BURN = (1 << 12) - 1;
int BASE = 0;
long long pw[55], tw[55];
short a[55][55][5];
int n, m;

struct hashi {
    int hd[10007], nxt[1000005], cnt, now;
    long long st[1000005];
```

```
    inline void clr() {
        cnt = 0;
        memset(hd, 0, sizeof hd);
    }

  inline void push(long long sta) { //在哈希表中插入状态 sta
        now = sta % 10007;
        for (int i = hd[now]; i; i = nxt[i])
            if (st[i] == sta) return;
        cnt++;
        st[cnt] = sta;
        nxt[cnt] = hd[now];
        hd[now] = cnt;
    }

  inline void roll() {    //转移到下一列
        long long burning, status;
        for (int i = 1; i <= cnt; i++) {
            burning = st[i] & BURN;
            status = st[i] >> (m + 1);
            burning = (burning << 1) & BURN;
            (status *= BASE) %= pw[m + 1];
            st[i] = (status << (m + 1)) | burning;
        }
    }
} h[2];

void init() {
    m -= 2;
    swap(n, m);
    BASE = m + 1;
    BURN = (1 << (m + 1)) - 1;
    pw[0] = 1, tw[0] = 1;
    for (int i = 1; i <= 15; i++)
        pw[i] = pw[i - 1] * BASE, tw[i] = tw[i - 1] * 2;
    int ll, rr, dd, uu, tmp;
    for (int j = m; j >= 1; j--) //预处理每个格子的所有旋转方式
        for (int i = 1; i <= n; i++) {
            scanf("%d", &tmp);
            uu = (tmp & 8) > 0, rr = (tmp & 1) > 0;
            dd = (tmp & 2) > 0, ll = (tmp & 4) > 0;
            tmp = uu + rr + dd + ll;
            if (tmp <= 1) { // 优化 6
                a[i][j][0] = 1;
```

```
                a[i][j][1] = 0;
            } else if (tmp == 2) {
                if ((uu && dd) || (ll && rr)) {
                    a[i][j][0] = 2;
                    a[i][j][1] = 1 | 4, a[i][j][2] = 2 | 8;
                } else {
                    a[i][j][0] = 4;
                    a[i][j][1] = 1 | 2, a[i][j][2] = 2 | 4;
                    a[i][j][3] = 4 | 8, a[i][j][4] = 8 | 1;
                }
            } else if (tmp == 3) {
                a[i][j][0] = 4;
                a[i][j][1] = 2 | 4 | 8, a[i][j][2] = 1 | 4 | 8;
                a[i][j][3] = 1 | 2 | 8, a[i][j][4] = 1 | 2 | 4;
            } else {
                a[i][j][0] = 1, a[i][j][1] = 1 | 2 | 4 | 8;
            }
        }
}

int L, R;
long long adq[1000005];
int b[55], hshs[55], c[55], cc;

void decode(long long x) { //把状态 x 的每一位拆出来
    int len = 0;
    memset(hshs, 0, sizeof hshs);
    memset(b, 0, sizeof b);
    for (; x; x /= BASE) {
        b[len++] = x % BASE;
        hshs[x % BASE] = 1;
    }
}

long long encode() { //把状态压缩成一个数
    long long ret = 0;
    for (int i = m; i >= 0; i--) {
        ret *= BASE;
        ret += b[i];
    }
    return ret;
}
```

```
int hamming_weight(int x) {
    int ret = 0;
    for (; x; x -= x & -x) ret++;
    return ret;
}
int nxt = 1, cur = 0;

void dp() {
    long long burning = 0, status = 0, mxburning, mxstatus;
    int mxplug = 0, plug = 0, best = 0;
    for (int i = 1; i <= m; i++) burning += tw[i];
    for (int i = 1; i <= m; i++) status += pw[i];
    h[cur].push((status << (m + 1)) + burning);
    int x, y, z;
    int ll, rr, dd, uu;
    bool connected, burned;
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= m; j++) {
            h[nxt].clr();
            mxstatus = mxplug = plug = mxburning = best = 0;
            L = R = 0;
            for (int k = 1; k <= h[cur].cnt; k++)
                for (int aaa = 1; aaa <= a[i][j][0]; aaa++) {
                    plug = 0;
                    burning = h[cur].st[k] & BURN;
                    if (!burning) continue;
                    status = h[cur].st[k] >> (m + 1);
                    decode(status);
                    x = b[j - 1];
                    y = b[j];
                    for (z = 1; hshs[z]; z++);
                    uu = a[i][j][aaa] & 1, rr = a[i][j][aaa] & 2;
                    dd = a[i][j][aaa] & 4, ll = a[i][j][aaa] & 8;
                    connected = (y && uu) || (x && ll);
                    burned = false;
                    if (y && uu && (burning & tw[j])) burned |= true;
                    if (x && ll && (burning & tw[j-1])) burned |= true;
                    if (connected) {
                        if (x && ll) z = x;
                        else z = y;
                    }
                    if (x && y && uu && ll) {
                        int nmn = min(x, y), nmx = max(x, y);
                        for (int i = 0; i <= m; i++)
```

```
                if (b[i] == nmx) b[i] = nmn;
            for (int i = 0; i <= m; i++)
                if (b[i] > nmx) b[i]--;
            if (burned)
                for (int i = 0; i <= m; i++)
                    if (b[i]==nmn) burning|=tw[i];
            z = nmn;
        }
        b[j - 1] = b[j] = 0;
        if (burning & tw[j - 1]) burning ^= tw[j - 1];
        if (burning & tw[j]) burning ^= tw[j];
        if (dd) {
            b[j - 1] = z;
            if (burned) burning |= tw[j - 1];
        }
        if (rr) {
            b[j] = z;
            if (burned) burning |= tw[j];
        }
        if (!burning) continue;
        memset(hshs, 0, sizeof hshs);
        for (int i = 0; i <= m; i++) {
            hshs[b[i]]++;
            if (b[i]) plug |= tw[i];
        }
        for (int i = 0; i <= m; i++)
            if (b[i] && hshs[b[i]]==1 && !(burning&tw[i])){
                b[i] = 0;
                hshs[b[i]] = 0;
            }
        c[0] = 0;
        for (int i=1; i<=m; i++) c[i] = m;
        for (int i=1; i<=m; i++) c[b[i]] = min(c[b[i]],i);
        for (int i = 0; i <= m; i++) b[i] = c[b[i]];
        status = encode();
        adq[R++] = (status << (m + 1)) + burning;
        int hmw = hamming_weight(burning);
        if (hmw > best) {
            best = hmw;
            mxburning = burning;
            mxstatus = status;
            mxplug = plug;
        }
    }
```

```
            if (L != R) h[nxt].push((mxstatus << (m + 1)) + mxburning);
            while (L != R) {
                burning = adq[L] & BURN;
                status = adq[L] >> (1 + m);
                decode(status);
                plug = 0;
                for (int i = 0; i <= m; i++)
                    if (b[i])
                        plug |= tw[i];
                if (!(((burning & mxburning) == burning) &&
                     ((plug & mxplug) == plug))) h[nxt].push(adq[L]);
                L++;
            }
            cur ^= 1;
            nxt ^= 1;
        }
        h[cur].roll();
    }
    best = 0;
    memset(b, 0, sizeof b);
    for (int i = 1; i <= h[cur].cnt; i++) {
        burning = h[cur].st[i] & BURN;
        best = max(best, hamming_weight(burning));
    }
    printf("%d\n", best);
}

int main() {
    freopen("game.in","r",stdin);
    freopen("game.out","w",stdout);
    while (~scanf("%d%d", &n, &m)) {
        init();
        dp();
    }
    return 0;
}
```

第二部分　字符串算法

第 1 章　Manacher

527.回文子串(palindrome,4s,256MB)

【参考程序】

```
#include <cmath>
#include <cstdio>
#include <cstring>
#include <iostream>
#include <algorithm>
#define p2 p << 1
#define p3 p << 1 | 1

inline int read()
{
    int res = 0; bool bo = 0; char c;
    while (((c = getchar()) < '0' || c > '9') && c != '-');
    if (c == '-') bo = 1; else res = c - 48;
    while ((c = getchar()) >= '0' && c <= '9')
        res = (res << 3) + (res << 1) + (c - 48);
    return bo ? ~res + 1 : res;
}

inline char get()
{
    char c;
    while ((c = getchar()) < 'a' || c > 'z');
    return c;
}

template <class T>
inline T Min(const T &a, const T &b) {return a < b ? a : b;}

template <class T>
inline T Max(const T &a, const T &b) {return a > b ? a : b;}

const int N = 5e4 + 5, M = N << 2, E = 55, W = E << 1, Z = W << 1;

char s[N], tag[M], t[W], qt[Z];
int n, k, q, R[Z];

struct node
{
```

```
char st[E], ed[E];
intlen, res;

friend inline node operator + (node a, node b) // 合并信息
{
    int m = Min(k, a.len) + Min(k, b.len), n = 0, pos = 0, id = 0;
    for (inti = 1; i<= Min(k, a.len); i++)
        t[Min(k, a.len) - i + 1] = a.ed[i];
    for (inti = 1; i<= Min(k, b.len); i++)
        t[Min(k, a.len) + i] = b.st[i];
    qt[0] = '!';
    for (inti = 1; i<= m; i++)
        qt[++n] = '%', qt[++n] = t[i];
    qt[++n] = '%'; qt[n + 1] = '@';
    for (inti = 1; i<= n; i++)
    {
        R[i] = pos>i ?Min(pos - i, R[(id << 1) - i]) : 1;
        while (qt[i - R[i]] == qt[i + R[i]]) R[i]++;
        if (i + R[i] >pos) pos = i + R[i], id = i;
    } //把左子区间的 ed 和右子区间的 st 连接起来做 Manacher
    if (k & 1)
    {
        for (inti = 1; i<= n; i += 2)
            R[i] = Min(R[i], k);
        for (inti = 2; i<= n; i += 2)
            R[i] = Min(R[i], k + 1);
    }
    else
    {
        for (inti = 1; i<= n; i += 2)
            R[i] = Min(R[i], k + 1);
        for (inti = 2; i<= n; i += 2)
            R[i] = Min(R[i], k);
    }  //处理掉长度超过 K 的回文子串
    node res; res.len = a.len + b.len;
    if (a.len>= k) for (inti = 1; i<= k; i++)
        res.st[i] = a.st[i];
    else
    {
        for (inti = 1; i<= a.len; i++)
            res.st[i] = a.st[i];
        for (inti = 1; i<= Min(k - a.len, b.len); i++)
            res.st[a.len + i] = b.ed[Min(k, b.len) - i + 1];
    }
    if (b.len>= k) for (inti = 1; i<= k; i++)
        res.ed[i] = b.ed[i];
    else
    {
        for (inti = 1; i<= b.len; i++)
```

```
            res.ed[i] = b.ed[i];
        for (inti = 1; i<= Min(k - b.len, a.len); i++)
            res.ed[b.len + i] = a.st[Min(k, a.len) - i + 1];
    }  //计算 st 和 ed
    res.res = a.res + b.res + (R[(Min(k, a.len) << 1) + 1] >> 1);
    for (inti = 1; i<= (Min(k, a.len) << 1); i++)
        res.res += Max(0, i + R[i] - 2 - (Min(k, a.len) << 1) >> 1);
    for (inti = (Min(k, a.len) + Min(k, b.len) << 1) + 1;
        i>= (Min(k, a.len) + 1 << 1); i--)
            res.res += Max(0, (Min(k, a.len) << 1) - i + R[i] >> 1);
    return res;
  }
} T[M];

node orz_dyf(int l, int r, char c)  //处理标记的影响
{
  node res; res.len = r - l + 1;
  for (inti = 1; i<= Min(k, r - l + 1); i++)
    res.st[i] = res.ed[i] = c;
  res.res = 0;
  for (int i = 1; i<= Min(k, r - l + 1); i++)
    res.res += r - l + 2 - i;
  return res;
}

void build(int l, int r, int p)      //建树
{
  if (l == r)
  {
    T[p].len = T[p].res = 1;
    T[p].st[1] = T[p].ed[1] = s[l];
    return;
  }
  int mid = l + r >> 1;
  build(l, mid, p2); build(mid + 1, r, p3);
  T[p] = T[p2] + T[p3];
}

void down(int p)
{
  if (tag[p]) tag[p2] = tag[p], tag[p3] = tag[p], tag[p] = 0;
}

void upt(int l, int r, int p)
{
  int mid = l + r >> 1;
  T[p] = (tag[p2] ?orz_dyf(l, mid, tag[p2]) : T[p2])
    + (tag[p3] ?orz_dyf(mid + 1, r, tag[p3]) : T[p3]);
}
```

```
void change(int l, int r, int s, int e, char c, int p) //修改
{
    if (l == s && r == e) return (void) (tag[p] = c);
    int mid = l + r >> 1; down(p);
    if (e <= mid) change(l, mid, s, e, c, p2);
    else if (s >= mid + 1) change(mid + 1, r, s, e, c, p3);
    else change(l, mid, s, mid, c, p2),
        change(mid + 1, r, mid + 1, e, c, p3);
    upt(l, r, p);
}

node ask(int l, int r, int s, int e, int p) //查询
{
    if (l == s && r == e) return tag[p] ?orz_dyf(l, r, tag[p]) : T[p];
    int mid = l + r >> 1; node res; down(p);
    if (e <= mid) res = ask(l, mid, s, e, p2);
    else if (s >= mid + 1) res = ask(mid + 1, r, s, e, p3);
    else res = ask(l, mid, s, mid, p2)
        + ask(mid + 1, r, mid + 1, e, p3);
    return upt(l, r, p), res;
}

int main()
{
    freopen("palindrome.in", "r", stdin);
    freopen("palindrome.out", "w", stdout);

    int op, l, r; char c;
    scanf("%s", s + 1);
    n = strlen(s + 1);
    k = read(); q = read();
    build(1, n, 1);
    while (q--)
    {
        op = read(); l = read(); r = read();
        if (op == 1)
        {
            c = get();
            change(1, n, l, r, c, 1);
        }
        else printf("%d\n", ask(1, n, l, r, 1).res);
    }
    return 0;
}
```

第 2 章　后缀数组

532.往事之树(recollection,2s,512MB)

【参考程序】

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

using namespace std;
#define ll long long

const int e = 2e5 + 5, inf = 0x3f3f3f3f;
int f[e][18], logn, nxt[e], go[e], adj[e], len[e], n, num, dep[e], knar
[e][18], pool;
int ans,fav[e],sum[e],tmp[e],sa[e],nowt,rt[e],val[e][18],h[e],rk[e];
int lo[e], nowu;
struct point {
    int l, r, ld, rd;
} c[e * 19];

inline void add(int x, int y, int z) {
    nxt[++num] = adj[x], adj[x] = num, go[num] = y, len[num] = z;
}

inline void dfs1(int u) {
    int i;
    for (i = 0; i < logn; i++) f[u][i + 1] = f[f[u][i]][i];
    for (i = adj[u]; i; i = nxt[i]) {
        int v = go[i];
        f[v][0] = u;
        fav[v] = len[i];
        dep[v] = dep[u] + 1;
        dfs1(v);
    }
}

inline void init_sa() { //树上 sa
    int i, j, r = 1;
    for (i = 1; i <= n; i++) sum[fav[i]]++;
    for (i = 1; i <= 301; i++) sum[i] += sum[i-1];
    for (i = 1; i <= n; i++) sa[sum[fav[i]]--] = i;
    knar[sa[1]][0] = 1;
    for (i = 2; i <= n; i++)
        knar[sa[i]][0] = (fav[sa[i]] == fav[sa[i-1]] ? r : ++r);
    for (j = 0; j < logn; j++) {
        for (i = 0; i <= r; i++) sum[i] = 0;
```

```
        for (i = 1; i <= n; i++) sum[knar[f[i][j]][j]]++;
        for (i = 1; i <= r; i++) sum[i] += sum[i - 1];
        for (i = 1; i <= n; i++)
            tmp[sum[knar[f[i][j]][j]]--] = i;
        for (i = 0; i <= r; i++) sum[i] = 0;
        for (i = 1; i <= n; i++) sum[knar[i][j]]++;
        for (i = 1; i <= r; i++) sum[i] += sum[i - 1];
        for (i = n; i >= 1; i--)
            sa[sum[knar[tmp[i]][j]]--] = tmp[i];
        r = 1;
        knar[sa[1]][j + 1] = 1;
        for (i = 2; i <= n; i++)
          knar[sa[i]][j+1] = (knar[sa[i-1]][j] != knar[sa[i]][j] ||
             knar[f[sa[i-1]][j]][j]!=knar[f[sa[i]][j]][j] ? ++r : r);
    }
    for (i = 1; i <= n; i++) rk[sa[i]] = i;
}

inline int jump(int x, int y) { //倍增求 height
    int i, res = 0;
    for (i = logn; i >= 0; i--)
        if (f[x][i] && f[y][i] && knar[x][i] == knar[y][i]) {
            x = f[x][i];
            y = f[y][i];
            res += 1 << i;
        }
    return res;
}

inline void init_rmq() { //预处理 height 区间最小值
    int i, j;
    lo[0] = -1;
    for (i = 1; i < n; i++) lo[i] = lo[i >> 1] + 1;
    for (i = 1; i < n; i++) val[i][0] = jump(sa[i], sa[i + 1]);
    for (j = 1; j <= logn; j++)
        for (i = 1; i + (1 << j) <= n; i++)
            val[i][j] = min(val[i][j-1], val[i + (1 << j-1)][j-1]);
}

inline int query(int x, int y) { //查 lcp(sa[x],sa[y])
    if (x > y) swap(x, y);
    y--;
    int k = lo[y-x+1], res = min(val[x][k], val[y-(1 << k)+1][k]);
    return res;
```

```
}

inline void collect(int x) {
    int l = c[x].l, r = c[x].r;
    c[x].ld = inf; //ld,rd 分别代表区间排名最小、最大的后缀
    c[x].rd = 0;
    if (l) {
        c[x].ld = min(c[x].ld, c[l].ld);
        c[x].rd = max(c[x].rd, c[l].rd);
    }
    if (r) {
        c[x].ld = min(c[x].ld, c[r].ld);
        c[x].rd = max(c[x].rd, c[r].rd);
    }
}

inline void update(int l, int r, int &x, int s) {
    if (!x) x = ++pool;
    if (l == r) {
        c[x].ld = c[x].rd = s;
        return;
    }
    int mid = l + r >> 1;
    if (s <= mid) update(l, mid, c[x].l, s);
    else update(mid + 1, r, c[x].r, s);
    collect(x);
}

inline int merge(int x, int y) {
    if (!x || !y) return x ^ y;
    c[x].ld = min(c[x].ld, c[y].ld),c[x].rd = max(c[x].rd,c[y].rd);
    c[x].l = merge(c[x].l, c[y].l),c[x].r = merge(c[x].r,c[y].r);
    collect(x);
    ans = max(ans,nowt+query(c[c[x].l].rd,c[c[x].r].ld));//合并时更新答案
    return x;
}

inline void dfs2(int u) {
    if (u != 1) update(1, n, rt[u], rk[u]);
    int i;
    for (i = adj[u]; i; i = nxt[i]) {
        int v = go[i];
        dfs2(v);
        nowu = u, nowt = dep[u];
```

```
        rt[u] = merge(rt[u], rt[v]);
    }
}

int main() {
    freopen("recollection.in", "r", stdin);
    freopen("recollection.out", "w", stdout);
    scanf("%d", &n), logn = log(n) / log(2.0);
    int i, x, y, j;
    for (i = 2; i <= n; i++) {
        scanf("%d%d", &x, &y), y++;
        add(x, i, y);
    }
    dfs1(1), init_sa(), init_rmq(), dfs2(1);
    cout << ans << endl;
    return 0;
}
```

第 3 章　后缀自动机

541.循环同构(tle,1s,256MB)

【参考程序】

```
#include <bits/stdc++.h>
#define For(i, a, b) for (int i = a, bb = b; i<= bb; ++i)
#define Rof(i, a, b) for (int i = a, bb = b; i>= bb; --i)
#define s64 long long

template <class T>
inline void get(T &res)
{
    char ch;
    bool bo = false;
    while ((ch = getchar()) < '0' || ch> '9')
        if (ch == '-') bo = true;

    res = ch - '0';
    while ((ch = getchar()) >= '0' &&ch<= '9')
        res = (res << 1) + (res << 3) + ch - '0';

    if (bo) res = ~ res + 1;
    return;
}

template <class T>
inline void _put(T x)
{
    if (x > 9) _put(x / 10);
    putchar(x % 10 + '0');
    return;
}

template <class T>
inline void put(T x, char ch)
{
    if (x < 0)
    {
        putchar('-');
        x = ~ x + 1;
    }
    _put(x);
    putchar(ch);

    return;
}

const int MaxS = 3e5 + 5;
int a[MaxS<< 1], fail[MaxS], Q, num, lst, ans;
char s[MaxS], t[MaxS<< 2];
struct SAM
{
    int g[26], pa, mx, cnt;

    #define g(x, y) sam[x].g[y]
```

```
    #define pa(x) sam[x].pa
    #define mx(x) sam[x].mx
    #define c(x) sam[x].cnt

    inline void clear()
    {
        For(i, 0, 25)
            g[i] = 0;
        pa = mx = cnt = 0;

        return;
    }
} sam[MaxS<< 1];

inline void build_sam(int c)
{
    int now;
    sam[now = ++num].clear();
    mx(now) = mx(lst) + 1;
    c(now) = 1;
    while (lst&& !g(lst, c))
        g(lst, c) = now, lst = pa(lst);

    if (!lst)
        pa(now) = 1;
    else
    {
        int x = g(lst, c);
        if (mx(x) == mx(lst) + 1)
            pa(now) = x;
        else
        {
            int y;
            sam[y = ++num] = sam[x];
            c(y) = 0;
            mx(y) = mx(lst) + 1;
            pa(x) = pa(now) = y;
            while (lst&&g(lst, c) == x)
                g(lst, c) = y, lst = pa(lst);
        }
    }
    lst = now;

    return;
}

inline bool cmp(const int &a, const int &b)
{
    return mx(a) < mx(b);
}

inline void init_build() //建出 s 的 SAM
{
    sam[num = lst = 1].clear();
    For(i, 1, strlen(s + 1))
        build_sam(s[i] - 'a');

    For(i, 1, num)
```

```
        a[i] = i;
    std::sort(a + 1, a + num + 1, cmp);
    Rof(i, num, 2)
        c(pa(a[i])) += c(a[i]);

    return;
}

inline void build_kmp(int n) //求出 t 倍长后的 KMP
{
    int lst = 0;
    For(i, 2, n)
    {
        while (lst&&t[lst + 1] != t[i])
            lst = fail[lst];
        if (t[lst + 1] == t[i]) lst++;
        fail[i] = lst;
    }

    return;
}

inline void solve()
{
    scanf("%s", t + 1);

    int n = strlen(t + 1);
    For(i, n + 1, n << 1)
        t[i] = t[i - n];
    build_kmp(n << 1);

    int now = 1, c, d = 0, ans = 0;
    For(i, 1, (n << 1) - 1)
    {
        c = t[i] - 'a';
        if (g(now, c)) //若通过转移边 c 有状态，则直接转移
            now = g(now, c), d++;
        else
        { //否则通过 parent 树跳到第一个有转移边的祖先上
            while (!g(now, c) && pa(now))
                now = pa(now);
            d = mx(now);
            if (g(now, c))
                now = g(now, c), d++;
        }
        while (mx(pa(now)) >= n) //串 t 的循环同构串的长度必须是|t|
            now = pa(now), d = mx(now);
        if (d >= n && fail[i] < n)
               //为避免重复，每个循环同构串都只在第一次出现时被统计
            ans += c(now);
    }

    put(ans, '\n');
    return;
}

int main()
{
```

```
    freopen("tle.in", "r", stdin), freopen("tle.out", "w", stdout);

    scanf("%s", s + 1);
    init_build();

    get(Q);
    while (Q--)
        solve();

    fclose(stdin), fclose(stdout);
    return 0;
}
```

543.基因窃取(dna,8s,2GB)

【参考程序】

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

inline int readString(char *s) {
    //由于输入量过大，使用快速读入读取字符串
    char ch;
    int len;
    while (ch = getchar(), !isupper(ch));
    s[len = 1] = ch;
    while (ch = getchar(), isupper(ch))
        s[++len] = ch;
    return len;
}

const int N = 3e6 + 5;
const int M = 6e6 + 5;
const int L = 3e7 + 5;
const int mod = 1e9 + 7;
int n, m, ans;
int cnt[M], lef[M], rig[M];
char a[L];

inline void add(int &x, int y) {
    x += y;
    x >= mod ? x -= mod : 0;
}

inline int reg(char ch) {
    if (ch == 'A') return 0;
    else if (ch == 'T') return 1;
    else if (ch == 'C') return 2;
    else return 3;
}

struct node {
    int mxl, par, ch[4];

    #define mx(x) tr[x].mxl
    #define par(x) tr[x].par

    inline void Clear() {
        memset(ch, 0, sizeof(ch));
        mxl = par = 0;
```

```
    }
};

struct sam {
    //因为要建两个 SAM，我们对涉及到的 SAM 操作进行封装
    node tr[M];
    int T, now;
    int g[M][4], pos[M], tag[M], rig[M], idx[M];
    char s[M];

    inline void extend(int id, char ch) {
        int c = reg(ch), i = now, j, p;
        tr[now = ++T].Clear();
        mx(now) = mx(i) + 1;
        pos[id] = now;
        rig[now] = id;
        for (; i && !tr[i].ch[c]; i = par(i))
            tr[i].ch[c] = now;

        if (!i)
            par(now) = 1;
        else {
            j = tr[i].ch[c];
            if (mx(i) + 1 == mx(j))
                par(now) = j;
            else {
                tr[p = ++T] = tr[j];
                rig[T] = 0;
                mx(p) = mx(i) + 1;
                par(now) = par(j) = p;
                for (; i && tr[i].ch[c]==j; i=par(i)) tr[i].ch[c] = p;
            }
        }
    }

    inline void Build() {
        tr[now = T = 1].Clear();
        rig[1] = 0;
        for (int i = 1; i <= n; ++i) pos[i] = 0;
        for (int i = 1; i <= n; ++i) extend(i, s[i]);
        for (int i = 1; i <= T; ++i) tag[i] = 0;
        for (int i = 1; i <= n; ++i) cnt[i] = 0;
        for (int i = 1; i <= T; ++i) ++cnt[mx(i)];
        for (int i = 1; i <= n; ++i) cnt[i] += cnt[i - 1];
        for (int i = T; i >= 1; --i) idx[cnt[mx(i)]--] = i;
        for (int i = T; i >= 2; --i) {
            int x = idx[i], y = par(x);
            rig[y] = rig[x];
            g[y][reg(s[rig[x] - mx(y)])] = x;
        }
    }

    inline void Print() {
        //把每个点的标记往下推
        for (int i = 1; i <= T; ++i) {
            int x = idx[i];
            for (int j = 0; j < 4; ++j)
                if (g[x][j])
```

```
                    add(tag[g[x][j]], tag[x]);
        }
    }
} t1, t2;

int main() {
    freopen("dna.in", "r", stdin);
    freopen("dna.out", "w", stdout);

    n = readString(t1.s);
    for (int i = 1; i <= n; ++i)
        t2.s[i] = t1.s[n - i + 1];
    t1.Build();
    t2.Build();
    scanf("%d", &m);
    while (m--) {
        int x1 = 1, x2 = 1, len;
        len = readString(a);
        for (int i = 1; i <= len; ++i) {
            if (t1.mx(x1) < i)
                x1 = t1.g[x1][reg(a[i])];
            else if (t1.s[t1.rig[x1] - i + 1] != a[i])
                break;
            if (x1)
                add(t1.tag[x1], 1);
            else
                break;
        }
        for (int i = 1; i <= len; ++i) {
            if (t2.mx(x2) < i)
                x2 = t2.g[x2][reg(a[i])];
            else if (t2.s[t2.rig[x2] - i + 1] != a[i])
                break;
            if (x2)
                add(t2.tag[x2], 1);
            else
                break;
        }
    }

    t1.Print();
    t2.Print();
    for (int i = 1; i <= n; ++i)
        rig[i] = t1.tag[t1.pos[i]];
    for (int i = 1; i <= n; ++i)
        lef[i] = t2.tag[t2.pos[n - i + 1]];
    for (int i = 1; i < n; ++i)
        ans = (1ll * rig[i] * lef[i + 1] + ans) % mod;
    printf("%d\n", ans);
}
```

第三部分　图论

第 1 章　二分图匹配

572.不同缩写(diff,1s,256MB)

【参考程序】

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

template <class T>
inline void read(T &res)
{
    char ch;
    while (ch = getchar(), !isdigit(ch));
    res = ch ^ 48;
    while (ch = getchar(), isdigit(ch))
        res = res * 10 + ch - 48;
}

using std::vector;
using std::string;
const int N = 305;
const int M = 1e5 + 5;

int n, len, lim, E, T, bm, qr, src, tis;
int G[M][26], tag[M], sl[N], vis[M], mth[M];
char s[N];
string sa[N][N], b[M], sans[N];
vector<int> v[M];

inline bool Find(int x) //匈牙利算法求最大匹配
{
    for (int i = 0, y, im = v[x].size(); i < im; ++i)
        if (vis[y = v[x][i]] != tis)
        {
            vis[y] = tis;
            if (!mth[y] || Find(mth[y]))
                return mth[x] = y, mth[y] = x, true;
        }
    return false;
}
```

```
struct node
{
    int ch[26], par;
    inline void Clear()
    {
        memset(ch, 0, sizeof(ch));
        par = 0;
    }
};

struct SAM //序列自动机
{
    node tr[N];
    int T, lst[26];

    inline void Init()
    {
        tr[T = 1].Clear();
        for (int i = 0; i < 26; ++i)
            lst[i] = 1;
    }

    inline void insert(int c) //插入字符
    {
        tr[++T].Clear();
        tr[T].par = lst[c];
        for (int i = 0; i < 26; ++i)
            for (int j=lst[i]; j && !tr[j].ch[c]; j=tr[j].par)
                tr[j].ch[c] = T;
        lst[c] = T;
    }
}p[N];

inline void Dfs(int id, int x, int k, string a) //搜索串的子序列
{
    if (sl[id] >= n)
        return ;
    if (k > 1)
        sa[id][++sl[id]] = a;
    if (k > lim)
        return ;
    for (int i = 0; i < 26; ++i)
        if (p[id].tr[x].ch[i])
            Dfs(id, p[id].tr[x].ch[i],k+1,a+(char)(i+'a'));
```

```
}

inline int insert_Trie(const string &a)
{ //需要对搜索出的子序列进行去重，这里用 Trie 来实现
    int x = 1, y;
    for (int i = 0, im = a.size(); i < im; ++i)
    {
        y = a[i] - 'a';
        if (!G[x][y])
        {
            G[x][y] = ++E;
            tag[E] = 0;
            memset(G[E], 0, sizeof(G[E]));
        }
        x = G[x][y];
    }
    if (!tag[x])
        b[tag[x] = ++T] = a;
    return tag[x];
}

inline bool check(int mid)
{
    lim = mid;
    for (int i = 1; i <= n; ++i)
        sl[i] = 0;
    string a = "";
    for (int i = 1; i <= n; ++i)
        Dfs(i, 1, 1, a);
    memset(G[E = 1], 0, sizeof(G[1]));
    T = n;
    for (int i = 1; i <= n; ++i)
        for (int j = 1; j <= sl[i]; ++j)
            v[i].push_back(insert_Trie(sa[i][j]));
    for (int i = 1; i <= T; ++i)
        mth[i] = 0;
    int cnt = 0;
    for (int i = 1; i <= n; ++i)
    {
        ++tis;
        Find(i) ? ++cnt : 0;
    }
    for (int i = 1; i <= n; ++i)
        v[i].clear();
```

```
    if (cnt == n)
    {
        for (int i = 1; i <= n; ++i)
            sans[i] = b[mth[i]];
        return true;
    }
    else
        return false;
}

int main()
{
    freopen("diff.in", "r", stdin);
    freopen("diff.out", "w", stdout);
    read(n);
    int l = 1, r = 0, ans = 0;
    for (int i = 1; i <= n; ++i)
    {
        scanf("%s", s + 1);
        len = strlen(s + 1);
        len > r ? r = len : 0;
        p[i].Init();
        for (int j = 1; j <= len; ++j)
            p[i].insert(s[j] - 'a');
    }
    while (l <= r)
    {
        int mid = l + r >> 1;
        if (check(mid))
            ans = mid, r = mid - 1;
        else
            l = mid + 1;
    }
    if (ans)
    {
        printf("%d\n", ans);
        for (int i = 1; i <= n; ++i)
            std::cout << sans[i] << std::endl;
    }
    else
        puts("-1");
    fclose(stdin); fclose(stdout);
    return 0;
}
```

573.后缀表达（expr,1s,128MB）

【参考程序】

```
#include <bits/stdc++.h>
typedef std::vector<int> vi;
const int MaxN = 6e3 + 5;

char s[MaxN];
vi son[MaxN];
int n, fa[MaxN], top, stk[MaxN], tag[MaxN], tag_mark;
ans[MaxN], y[MaxN], ty[MaxN];
bool invalid[MaxN], leaf[MaxN][26], avl[MaxN][26];

inline bool match(int u, int from) {
    for (auto v : son[from])
        if (tag[v] != tag_mark&&avl[v][u]) {
            tag[v] = tag_mark;
            if (y[v] == -1 || match(y[v], from)) {
                return y[v] = u, true;
            }
        }
    return false;
}

inline bool tmp_match(int u, int from) { //判断能否找到增广路
    for (auto v : son[from])
        if (tag[v] != tag_mark&&avl[v][u]) {
            tag[v] = tag_mark;
            if (y[v] == -1 || tmp_match(y[v], from))
                return true;
        }
    return false;
}

inline void dfs(int u) {
    ans[u] = 1;
    for (auto v : son[u]) {
        dfs(v), ans[u] += ans[v];
    }
    for (int i = 0; i< 26; ++i)
          ans[u] += leaf[u][i]; //先计算ans[u]的初值
    tag_mark = 0;
    for (int i = 0; i< 26; ++i)
       if (leaf[u][i]) {
```

```
            ++tag_mark, avl[u][i] = true;
            ans[u] -= match(i, u); //将ans[u]减去最大匹配数
        }
    for (auto v : son[u]) {
        tag[v] = ++tag_mark; //找v的匹配点的增广路时，先标记v不能走
        if (y[v] == -1||tmp_match(y[v],u)) {
             //判断删去v后最大匹配数是否减小
             for (int i = 0; i< 26; ++i) avl[u][i] |= avl[v][i];
        }
    }
}

int main() {
    scanf("%s", s + 1), n = strlen(s + 1);
    for (int i = 1; i<= n; ++i) {
        y[i] = -1; //因为我们将字母a转化成序号0，所以这里初值设为-1
        if (isalpha(s[i]))
            stk[++top] = i;
        else {
            fa[stk[top]]=i,fa[stk[top-1]] = i;//利用栈建立表达式树
            stk[--top] = i;
        }
    }
    for (int i = 1; i< n; ++i) {
        while (s[fa[i]] == s[fa[fa[i]]]) {//合并相同的相邻操作符结点
            invalid[fa[i]] = true; //并且标记原图中无用的点
            fa[i] = fa[fa[i]];
        }
    }
    for (int i = 1; i<= n; ++i) {
        if (isalpha(s[i]))
          leaf[fa[i]][s[i]-'a'] = true;//标记每个点的叶子结点的集合
        else if (!invalid[i])
          son[fa[i]].push_back(i);     //son[u]储存u的儿子集合
    }
    dfs(n), printf("%d\n", ans[n]);
    return 0;
}
```

第 2 章　网络流

582.大收藏家（collection，1s，256MB）

【参考程序】

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

template <class T>
inline void read(T &x)
{
    static char ch;
    while (!isdigit(ch = getchar()));
    x = ch - '0';
    while (isdigit(ch = getchar()))
        x = x * 10 + ch - '0';
}

template <class T>
inline void relax(T &x, const T &y)
{
    if (x < y) x = y;
}

template <class T>
inline void tense(T &x, const T &y)
{
    if (x > y) x = y;
}

const int MaxN = 3e3 + 5;
struct edge
{
    int u, v;
    inline void scan()
    {
        read(u), read(v);
    }
}e[MaxN];

int n, m, a[MaxN];
std::map<int, int> pos[MaxN];

typedef std::map<int, int>::iterator map_it;
```

```
namespace Dinic
{
    const int MaxNV = 1e4 + 5;
    const int MaxNE = MaxNV << 4;

    int tot, ect = 1, src, des;
    int cur[MaxNV], lev[MaxNV];
    int adj[MaxNV], to[MaxNE], nxt[MaxNE], cap[MaxNE];

    inline void init()
    {
        for (int i = 1; i <= n; ++i)
            pos[i].clear();
        for (int i = 0; i <= tot; ++i)
            adj[i] = 0;
        tot = 0, ect = 1;
    }

    inline void addEdge(int u, int v, int w)
    {
        //printf("(%d->%d)cap = %d\n", u, v, w);
        to[++ect]=v, nxt[ect]=adj[u], adj[u]=ect, cap[ect]=w;
        to[++ect]=u, nxt[ect]=adj[v], adj[v]=ect, cap[ect]=0;
    }

    inline bool bfs()
    {
        static int que[MaxNV], qr;
        for (int i = src; i <= des; ++i)
            lev[i] = -1, cur[i] = adj[i];
        lev[que[qr = 1] = src] = 1;
        for (int ql = 1; ql <= qr; ++ql)
        {
            int u = que[ql];
            for (int e = adj[u], v; v = to[e], e; e = nxt[e])
                if (lev[v] == -1 && cap[e] > 0)
                {
                    lev[v] = lev[u] + 1;
                    if (v == des) return true;
                    que[++qr] = v;
                }
        }
        return false;
    }
```

```
inline int dfs(int u, int flow)
{
    if (u == des) return flow;

    int res = 0;
    for (int &e = cur[u], v; v = to[e], e; e = nxt[e])
        if (lev[v] > lev[u] && cap[e] > 0)
        {
              int delta=dfs(v,std::min(cap[e],flow-res));
            if (delta)
            {
                res += delta, cap[e] -= delta;
                          cap[e ^ 1] += delta;
                if (res == flow) return flow;
            }
        }
    if (res != flow) lev[u] = -1;
    return res;
}

inline int max_flow()
{
    static const int INF = 0x3f3f3f3f;

    int res = 0;
    while (bfs()) res += dfs(src, INF);

    return res;
}

inline void build_graph()
{
    init();

    src = 0;
    for (int i = 1; i <= n; ++i)
        pos[i][1] = ++tot;
    addEdge(pos[e[1].u][1], pos[e[1].v][1], 1);
    addEdge(pos[e[1].v][1], pos[e[1].u][1], 1);

    for (int i = 2; i <= m; ++i)
    {
        int u = e[i].u, v = e[i].v;
```

```
            pos[u][i] = ++tot, pos[v][i] = ++tot;
            addEdge(tot, tot - 1, 1);
            addEdge(tot - 1, tot, 1);
        }

        for (int i = 1; i <= n; ++i)
        {
            int lst = src;

            std::map<int, int> &S = pos[i];
            for (map_it it = S.begin(); it != S.end(); ++it)
            {
                addEdge(lst, it->second, lst ? a[i] : 1);
                lst = it->second;
            }
        }

        des = tot + 1;
        addEdge((--pos[1].end())->second, des, a[1]);
    }
}

int main()
{
    freopen("collection.in", "r", stdin);
    freopen("collection.out", "w", stdout);
    int T;
    read(T);
    while (T--)
    {
        read(n), read(m);
        for (int i = 1; i <= n; ++i)
            read(a[i]);
        for (int i = 1; i <= m; ++i)
            e[i].scan();

        Dinic::build_graph();
        printf("%d\n", Dinic::max_flow());
    }
    fclose(stdin); fclose(stdout);
    return 0;
}
```

583.排列(permutation,1s,256MB)

【参考程序】

```
#include <bits/stdc++.h>
using namespace std;
template <class t>
inline void read(t & res)
{
    bool fl = 0;
    char ch;
    while (ch = getchar(), ch != '-' && !isdigit(ch));
    if (ch == '-') fl = 1, res = 0;
    else res = ch ^ 48;
    while (ch = getchar(), isdigit(ch))
    res = res * 10 + (ch ^ 48);
    if (fl) res = ~res + 1;
}
const int e = 1e5 + 5, inf = 0x3f3f3f3f;
int adj[e],nxt[e],go[e],d[e],c[e],f[e],num=1,n,a[e],cur[e],s,t,m,ans;
bool vis[e];

inline void add(int x, int y, int v)
{
    nxt[++num] = adj[x];
    adj[x] = num;
    go[num] = y;
    c[num] = v;
    nxt[++num] = adj[y];
    adj[y] = num;
    go[num] = x;
}

inline bool bfs()
{
    int i;
    for (i = 1; i <= t; i++) cur[i] = adj[i], vis[i] = 0;
    vis[s] = 1;
    d[s] = 0;
    queue<int>q;
    q.push(s);
    while (!q.empty())
    {
        int u = q.front();
        q.pop();
        for (i = adj[u]; i; i = nxt[i])
```

```
        {
            int v = go[i];
            if (!vis[v] && c[i] > f[i])
            {
                vis[v] = 1;
                q.push(v);
                d[v] = d[u] + 1;
            }
        }
    }
    return vis[t];
}

inline int dfs(int u, int a)
{
    if (u == t || a == 0) return a;
    int flow = 0, f1;
    for (int &i = cur[u]; i; i = nxt[i])
    {
        int v = go[i];
        if (d[v] == d[u] + 1 && (f1 = dfs(v, min(c[i] - f[i], a))) > 0)
        {
            f[i] += f1;
            f[i ^ 1] -= f1;
            flow += f1;
            a -= f1;
            if (a == 0) break;
        }
    }
    return flow;
}

inline int maxflow()
{
    int res = 0;
    while (bfs()) res += dfs(s, inf);
    return res;
}

int main()
{
    freopen("permutation.in", "r", stdin);
    freopen("permutation.out", "w", stdout);
    int i, x, y;
```

```
    read(n); read(m);
    s = 2 * n + 1; t = 2 * n + 2;
    for (i = 1; i <= n; i++)  // 建边，容量为 0 的边可以不连
    {
        read(a[i]);
        if (a[i] >= 0)
        {
            ans += a[i];
            add(s, i, a[i]);         // s->i1，容量 a[i]
            add(i + n, t, a[i]);    // i2->t，容量 a[i]
        }
        else add(i, i + n, -a[i]); // i1->i2，容量-a[i]
    }
    while (m--)
    {
        read(x);
        read(y);
        add(x, y, inf);          // x1->y1，容量正无穷
        add(x + n, y + n, inf); // x2->y2，容量正无穷
    }
    ans -= maxflow();           // 用非负数之和减去最小割
    cout << ans << endl;
    fclose(stdin); fclose(stdout);
    return 0;
}
```

第 3 章　费用流

592.红树蓝树(w,1s,256MB)

【参考程序】

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

template <class T>
inline void read(T &res)
{
    char ch; bool flag = false; res = 0;
    while (ch = getchar(), !isdigit(ch) && ch != '-');
    ch == '-' ? flag = true : res = ch ^ 48;
    while (ch = getchar(), isdigit(ch))
        res = res * 10 + ch - 48;
    flag ? res = -res : 0;
}

using std::vector;
const int N = 1005;
const int M = 1e6 + 5;
const int Maxn = 0x3f3f3f3f;
int T = 1, n, src, des, ans, Q1, Q0, rtx, rty; bool vis[N], walk[N];
int fa1[N], fa2[N], w[N], dis[N];
int h[M], lim1[N], lim2[N];
int tag1[N], tag2[N], lst[N], nxt[M], cst[M], to[M], flw[M];
vector<int> v1[N], v2[N];

inline void Link(int x, int y, int z, int w)
{
    nxt[++T] = lst[x];lst[x] = T;to[T] = y;flw[T] = z;cst[T] = w;
    nxt[++T] = lst[y];lst[y] = T;to[T] = x;flw[T] = 0;cst[T] = -w;
}

inline bool SPFA()
{
    for (int i = 1; i <= des; ++i)
        dis[i] = Maxn, walk[i] = false;
    dis[h[1] = src] = 0;
    int t = 0, w = 1, x, y;
    while (t < w)
    {
        vis[x = h[++t]] = false;
        for (int e = lst[x]; e; e = nxt[e])
            if (y=to[e], flw[e]>0 && dis[y]>dis[x] + cst[e])
```

```
            {
                dis[y] = dis[x] + cst[e];
                if (!vis[y])
                    vis[h[++w] = y] = true;
            }
    }
    return dis[des] < Maxn;
}

template <class T>
inline T Min(T x, T y) {return x < y ? x : y;}

inline int Dinic(int x, int flow)
{
    if (x == des)
    {
        ans += flow * dis[des];
        return flow;
    }

    walk[x] = true;
    int y, del, res = 0;
    for (int e = lst[x]; e; e = nxt[e])
        if (y=to[e],!walk[y]&&flw[e]>0 && dis[y]==dis[x]+cst[e])
        {
            del = Dinic(y, Min(flow - res, flw[e]));
            if (del)
            {
                flw[e] -= del;
                flw[e ^ 1] += del;
                res += del;
                if (flow == res)
                    break;
            }
        }
    return res;
}

inline void MCMF()
{
    int res = 0;
    while (SPFA())
        res += Dinic(src, Maxn);
}
```

```
inline void Dfs1(int x)
{
    int tot = 0;
    for (int i = 0, im = v1[x].size(); i < im; ++i)
    {
        int y = v1[x][i];
        if (y == fa1[x]) continue;
        fa1[y] = x;
        Dfs1(y);
    }
}

inline void Dfs2(int x)
{
    for (int i = 0, im = v2[x].size(); i < im; ++i)
    {
        int y = v2[x][i];
        if (y == fa2[x])
            continue;
        fa2[y] = x;
        Dfs2(y);
    }
}

inline void Dfs3(int x, int &v) //通过 DFS 得到新的 bi
{
    for (int i = 0, im = v1[x].size(); i < im; ++i)
    {
        int y = v1[x][i];
        if (y == fa1[x])
            continue;
        if (tag1[y] != Maxn)
        {
            v -= tag1[y];
            continue;
        }
        Dfs3(y, v);
    }
}

inline void Dfs4(int x, int &v)
{
    for (int i = 0, im = v2[x].size(); i < im; ++i)
```

```
    {
        int y = v2[x][i];
        if (y == fa2[x])
            continue;
        if (tag2[y] != Maxn)
        {
            v -= tag2[y];
            continue;
        }
        Dfs4(y, v);
    }
}

int main()
{
    freopen("w.in", "r", stdin);
    freopen("w.out", "w", stdout);
    read(n); read(rtx); read(rty);
    src = n << 1 | 1;
    des = src + 1;
    for (int i = 1; i <= n; ++i)
        read(w[i]);
    for (int i = 1, x, y; i < n; ++i)
    {
        read(x); read(y);
        v1[x].push_back(y);
        v1[y].push_back(x);
    }
    for (int i = 1, x, y; i < n; ++i)
    {
        read(x); read(y);
        v2[x].push_back(y);
        v2[y].push_back(x);
    }

    for (int i = 1; i <= n; ++i)
        tag1[i] = tag2[i] = Maxn;
    read(Q0);
    for (int i = 1, x, y; i <= Q0; ++i)
    {
        read(x); read(y);
        tag1[x] = y;
    }
    read(Q1);
```

```
for (int i = 1, x, y; i <= Q1; ++i)
{
    read(x); read(y);
    tag2[x] = y;
}
Dfs1(rtx);
Dfs2(rty);
for (int i = 1; i <= n; ++i)
    lim1[i] = tag1[i], lim2[i] = tag2[i];
for (int i = 1; i <= n; ++i)
    if (lim1[i] != Maxn)
        Dfs3(i, lim1[i]);
for (int i = 1; i <= n; ++i)
    if (lim2[i] != Maxn)
        Dfs4(i, lim2[i]);
for (int i = 1; i <= n; ++i)
    if (lim1[i] < 0 || lim2[i] < 0)
        return puts("-1"), 0;
for (int i = 1; i <= n; ++i)
{
    int u, v;
    for (u = i; tag1[u] == Maxn; u = fa1[u]);
    for (v = i; tag2[v] == Maxn; v = fa2[v]);
    Link(u, v + n, 1, -w[i]);
}
for (int i = 1; i <= n; ++i)
    if (lim1[i] != Maxn)
        Link(src, i, lim1[i], 0);
for (int i = 1; i <= n; ++i)
    if (lim2[i] != Maxn)
        Link(i + n, des, lim2[i], 0);
MCMF();
bool flag = false; //判断无解情况
for (int e = lst[src]; e; e = nxt[e])
    if (flw[e] > 0)
    {
        flag = true;
        break;
    }
for (int e = lst[des]; e; e = nxt[e])
    if (flw[e ^ 1] > 0)
    {
        flag = true;
        break;
```

```
        }
    printf("%d\n", flag ? -1 : -ans);
    fclose(stdin); fclose(stdout);
    return 0;
}
```

593.木棍问题（trouble，1.5s，512MB）

【参考程序】

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

template <class T>
inline void read(T &x)
{
    static char ch;
    while (!isdigit(ch = getchar()));
    x = ch - '0';
    while (isdigit(ch = getchar()))
        x = x * 10 + ch - '0';
}

template <class T>
inline bool tense(T &x, const T &y)
{
    return x > y ? x = y, true : false;
}

#define trav(u) for (halfEdge * e = adj[u]; e; e = e->next)
#define flow(u) for (halfEdge *&e = cur[u]; e; e = e->next)

const int INF = 0x3f3f3f3f;
const int MaxN = 45;
const int MaxNV = MaxN * MaxN * 4;
const int MaxNE = MaxNV * 10;

struct halfEdge
{
    int v, w, c;
    halfEdge *next;
};

halfEdge *adj[MaxNV], *cur[MaxNV];
halfEdge adj_pool[MaxNE], *adj_tail = adj_pool;

int nV, src, des;
```

```
int dis[MaxNV];
bool walk[MaxNV];
int n, m, typ, A, B;
int pos[MaxN][MaxN][3];
bool bo[MaxN][MaxN];
char s[MaxN];
int res_cost;

inline void addEdge(int u, int v, int w, int c)
{
    adj_tail->v = v, adj_tail->w = w, adj_tail->c = +c, adj_tail->next = adj[u], adj[u] = adj_tail++;
    adj_tail->v = u, adj_tail->w = 0, adj_tail->c = -c, adj_tail->next = adj[v], adj[v] = adj_tail++;
}

inline void insEdge(int u, int v, int w, int c, bool opt)
{
    if (opt)
        addEdge(u, v, w, c);
    else
        addEdge(v, u, w, c);
}

inline halfEdge *oppo(halfEdge *e)
{
    return adj_pool + ((e - adj_pool) ^ 1);
}

inline bool SPFA()
{
    static bool inq[MaxNV];
    static int que[6000010], qr;
    for (int i = 1; i <= nV; ++i)
    {
        cur[i] = adj[i];
        dis[i] = INF;
        walk[i] = false;
    }
    inq[src] = true;
    dis[que[qr = 1] = src] = 0;
    for (int ql = 1; ql <= qr; ++ql)
    {
        int u = que[ql];
```

```
        inq[u] = false;
        trav(u)
            if (e->w && tense(dis[e->v], dis[u] + e->c) && !inq[e->v])
                inq[que[++qr] = e->v] = true;
    }
    return dis[des] < INF;
}

inline int Dinic(int u, int flw)
{
    if (u == des)
    {
        res_cost += dis[des];
        printf("%d\n", typ >= 8 && typ <= 12 ? (bool)res_cost : res_cost);
        return flw;
    }
    walk[u] = true;
    int res = 0;
    flow(u)
        if (e->w && dis[e->v] == dis[u] + e->c && !walk[e->v])
        {
            int delta = Dinic(e->v, std::min(e->w, flw - res));
            if (delta)
            {
                e->w -= delta;
                oppo(e)->w += delta;
                res += delta;
                if (res == flw)
                    break;
            }
        }
    if (res == flw)
        walk[u] = false;
    return res;
}

inline void MCMF()
{
    while (SPFA())
        Dinic(src, INF);
}

int main()
{
```

```
    freopen("trouble.in", "r", stdin);
    freopen("trouble.out", "w", stdout);
    read(typ), read(n), read(m), read(A), read(B);
    src = ++nV, des = ++nV;
    for (int i = 1; i <= n; ++i)
    {
       scanf("%s", s + 1);
       for (int j = 1; j <= m; ++j)
       {
        bo[i][j] = s[j] == '0';
        if (bo[i][j])
        {
           pos[i][j][0] = ++nV;
           pos[i][j][1] = ++nV;
           pos[i][j][2] = ++nV;
           for (int k = 0; k < 4; ++k)
              if (i + j & 1)
                addEdge(src, pos[i][j][0], 1, A * k);
            else
                addEdge(pos[i][j][0], des, 1, A * k);
           for (int k = 1; k <= 2; ++k)
           {
            insEdge(pos[i][j][0],pos[i][j][k],1,0,i+j & 1);
             insEdge(pos[i][j][0],pos[i][j][k],1,B-A,i+j & 1);
           }
        }
       }
    }
    for (int i = 1; i <= n; ++i)
       for (int j = 1; j <= m; ++j)
       {
        if (!bo[i][j])
            continue;
        if (i != n && bo[i + 1][j])
            insEdge(pos[i][j][1],pos[i+1][j][1],1,0,i+j & 1);
        if (j != m && bo[i][j + 1])
            insEdge(pos[i][j][2],pos[i][j+1][2],1,0,i+j & 1);
       }
    MCMF();
    return 0;
}
```

第 4 章　强连通分量

602.最大的流(flow,3s,256MB)

【参考程序】

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

const int MaxN = 3005, MaxM = 9005, mod = 13;
int f[MaxN], adj[MaxN], nxt[MaxM], go[MaxM];
int dfn[MaxN], low[MaxN], frm[MaxN], stk[MaxN], bel[MaxN], val[MaxN];
//dfn[u]表示搜索到 u 的次序编号（时间戳）
//low[u]表示 u 的子树经过最多一条后向边或栈中横叉边能回溯到的最早栈中结点编号
//frm[u]表示 u 是从哪条边推来的，即 u 和 fa(u)之间的连边的编号
//bel[u]表示 u 所属的边双连通分量的编号
//val[j]用 hash，表示删掉第 i 条边(2≤i≤tot)之后，j 所属的边双连通分量的编号
int n, m, s, t, x, y, fx, fy, tot = 1, tim, top, num, ans;
bool sgn[MaxM], flag; //sgn[i]表示第 i 条边是否被删除

inline int get() {
    char ch;
    while ((ch = getchar()) < '0' || ch > '9');
    int res = ch - '0';
    while ((ch = getchar()) >= '0' && ch <= '9')
        res = (res << 1) + (res << 3) + ch - '0';
    return res;
}

inline void add(int x, int y) {
    nxt[++tot] = adj[x];
    adj[x] = tot;
    go[tot] = y;
    return;
}

inline int find(int x) {
    return f[x] == x ? x : f[x] = find(f[x]);
}

inline void init() {
    top = tim = num = 0;
    For(i, 1, n) dfn[i] = low[i] = frm[i] = bel[i] = 0;
    return;
}

inline void ckmin(int &x, const int &y) {
    if (x > y) x = y;
    return;
}

inline void Tarjan(int u) {
    dfn[u] = low[u] = ++tim;
    stk[++top] = u;
    int v;
    for (int e = adj[u]; e; e = nxt[e]) {
```

```
        if (sgn[e] || e == (frm[u] ^ 1))
            continue;  //如果边 e 被删除或者 e 是反向边，则跳过
        if (!dfn[v = go[e]]) {
            frm[v] = e;
            Tarjan(v);
            ckmin(low[u], low[v]);
        } else ckmin(low[u], dfn[v]);
    }
    if (dfn[u] == low[u]) {
       //此时 frm[u]是桥，栈中在 u 之后的所有点与 u 构成一个边双连通分量
        num++;
        do
            bel[stk[top]] = num;
        while (stk[top--] != u);
    }
    return;
}

int main() {
    n = get(), m = get();
    For(i, 1, n) f[i] = i;
    while (m--) {
        x = get(), y = get();
        add(x, y);
        add(y, x);
        fx = find(x), fy = find(y);
        f[fy] = fx;
    }
    For(i, 1, n) find(i);
    for (int i = 2; i <= tot; i += 2) {
        //枚举被删除的边 i、i+1，这两条边互为反向边
        sgn[i] = sgn[i + 1] = true;
        init();
        For(j, 1, n)
        if (!dfn[j]) Tarjan(j);
        For(j, 1, n) val[j] = val[j] * mod + bel[j];
        //记下删除每条边后，每个点所属的边双连通分量
        //取模运算有可能会超时，考虑直接用自然溢出
        sgn[i] = sgn[i + 1] = false;
    }
    init();
    For(j, 1, n)
    if (!dfn[j]) Tarjan(j);
    //在不删除任何边的情况下再做一次 Tarjan
    //使得 bel[i]中储存的是原图中点 i 所在的边双连通分量的编号
    For(i, 1, n - 1)
    For(j, i + 1, n) { //枚举点对(i,j)
       if (f[i] != f[j]) continue;
       //利用并查集判断(i,j)是否联通，不连通则最大流为 0
       if (bel[i] != bel[j]) ans++;
       //如果原图中(i,j)所在的边双连通分量不同，则最大流为 1
       else if (val[i] != val[j]) ans += 2;
       //如果删掉某条边之后，(i,j)所在的边双连通分量不同，则最大流为 2
          else ans += 3; //由于入度最大为 3，所以最大流最大为 3
    }
    printf("%d\n", ans);
    return 0;
}
```

603.情报传递(gplt，2s，256MB)

【参考程序】

```
#include <bits/stdc++.h>
typedef long long s64;

template <class T>
inline void tense(T &x, const T &y) {
    if (x > y)
        x = y;
}

const int MaxNV = 1e5 + 5;
const int MaxNE = 1e5 + 5;
const int K = 20000;
std::bitset<K + 5> f[MaxNV];

int n, m, T, ind1[MaxNV], ind[MaxNV];
int px[MaxNE], py[MaxNE], qx[MaxNE], qy[MaxNE];
int dfn[MaxNV], low[MaxNV], dfs_clock;
int stk[MaxNV], col[MaxNV], top, totcol;

struct halfEdge {
    int v;
    halfEdge *nxt;
};

struct graph {
    halfEdge *adj[MaxNV], pool[MaxNE], *tail;
    inline void init() {
        tail = pool;
        for (int i = 1; i <= n; ++i)
            adj[i] = 0;
    }
    inline void addEdge(const int &u, const int &v) {
        tail->v = v, tail->nxt = adj[u], adj[u] = tail++;
    }
} g, dag; //g 表示原图,dag 表示缩点后的反图

#define trav(u, g) for (halfEdge *e = g.adj[u]; e; e = e->nxt)

inline void Tarjan(const int &u) {
    //Tarjan 算法求强连通分量
    dfn[u] = low[u] = ++dfs_clock;
```

```
    stk[++top] = u;
    trav(u, g)
    if (!dfn[e->v]) {
        Tarjan(e->v);
        tense(low[u], low[e->v]);
    } else if (!col[e->v])
        tense(low[u], dfn[e->v]);
    if (low[u] == dfn[u]) {
        ++totcol;
        do
            col[stk[top--]] = totcol;
        while (stk[top + 1] != u);
    }
}

inline bool check(const int &l, const int &r) {
    //检查 b 在[l,r]内的要求 2 是否满足
    for (int i = 1; i <= totcol; ++i) {
        ind[i] = ind1[i];
        f[i].reset();
        if (i >= l && i <= r)
            f[i][i - l] = 1;
    }
    static int q[MaxNV], tail;
    tail = 0;
    for (int i = 1; i <= totcol; ++i)
        if (!ind[i])
            q[++tail] = i;
    for (int head = 1, u; head <= tail; ++head) {
        u = q[head];
        trav(u, dag) {
            f[e->v] = f[e->v] | f[u]; //建的是反图，u 能到达的点 v 也能到达

            if (!--ind[e->v])
                q[++tail] = e->v;
        }
    }
    for (int i=1; i <= T; ++i) //如果区间内有一个要求 2 没有满足，则返回 false
       if (col[qy[i]]>=l && col[qy[i]]<=r && f[col[qx[i]]][col[qy[i]]-l])
            return false;
    return true;
}
```

```
int main() {
    g.init();
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= m; ++i) {
        scanf("%d%d", px + i, py + i);
        g.addEdge(px[i], py[i]);
    }
    for (int i = 1; i <= n; ++i)
        if (!dfn[i])
            Tarjan(i);
    dag.init();
    for (int u = 1; u <= n; ++u) //建出缩点后的反图
        trav(u, g)
        if (col[u] != col[e->v]) {
            ++ind1[col[u]];
            dag.addEdge(col[e->v], col[u]);
        }
    scanf("%d", &T);
    for (int i = 1; i <= T; ++i)
        scanf("%d%d", qx + i, qy + i);
    for (int i = 1, l, r; i <= totcol; i += K) {
        l = i, r = std::min(i + K - 1, totcol);
        if (!check(l, r)) {
            puts("NO");
            return 0;
        }
    }
    puts("YES");
    printf("%d\n", m);
    for (int i = 1; i <= m; ++i)
        printf("%d %d\n",px[i],py[i]);//若都能得到满足，则输出要求 1 的边即可
    return 0;
}
```

第 5 章　2-SAT

607.洗牌还原(wash,1s,128MB)

【参考程序】

```
#include <bits/stdc++.h>
#define For(i, a, b) for (int i = a, bb = b; i <= bb; ++i)
#define Rof(i, a, b) for (int i = a, bb = b; i >= bb; --i)
#define s64 long long
#define oppo(x) (x <= m ? x + m : x - m)

using std :: vector;
using std :: swap;

template <class T>
inline void get(T &res)
{
    char ch;
    bool bo = false;
    while (ch = getchar(), !isdigit(ch))
        if (ch == '-') bo = true;
    res = ch ^ 48;
    while (ch = getchar(), isdigit(ch))
        res = (res << 1) + (res << 3) + (ch ^ 48);
    if (bo) res = ~ res + 1;
    return;
}

template <class T>
inline void _put(T x)
{
    if (x > 9) _put(x / 10);
    putchar(x % 10 + '0');
    return;
}

template <class T>
inline void put(T x, char ch)
{
    if (x < 0)
    {
        putchar('-');
        x = ~ x + 1;
    }
    _put(x);
    putchar(ch);
    return;
}

const int MaxN = 2005, MaxM = 5e5 + 5;
int adj[MaxN], nxt[MaxM], go[MaxM], cap[MaxM], P;
int dfn[MaxN], low[MaxN], stk[MaxN], bel[MaxN], tim, top, num;
```

```
int apr[MaxN], l1[MaxN], r1[MaxN], l2[MaxN], r2[MaxN], n, m;
bool ans[MaxN];
vector <int> d[MaxN >> 1];

inline void Link(int x, int y)
{
    nxt[++P] = adj[x], adj[x] = P, go[P] = y;
    return;
}

inline bool cross(int x1, int x2, int y1, int y2)
{
    return x1 < y1 && y2 < x2 || y1 < x1 && x2 < y2;
}

inline bool check(int a, int b) //判断 a,b 两种分配是否矛盾
{
    return cross(l1[a], r1[a], l1[b], r1[b])
        || cross(l1[a], r1[a], l2[b], r2[b])
        || cross(l2[a], r2[a], l1[b], r1[b])
        || cross(l2[a], r2[a], l2[b], r2[b]);
}

inline void ckmin(int &x, const int &y)
{
    if (x > y) x = y;
    return;
}

inline void Tarjan(int u)
{
    dfn[u] = low[u] = ++tim;
    stk[++top] = u;

    int v;
    for (int e = adj[u]; e; e = nxt[e])
    if (!dfn[v = go[e]])
    {
        Tarjan(v);
        ckmin(low[u], low[v]);
    }
    else if (!bel[v])
        ckmin(low[u], dfn[v]);
    if (dfn[u] == low[u])
    {
        num++;
        do
            bel[stk[top]] = num;
        while (stk[top--] ^ u);
    }
    return;
}
```

```
int main()
{
    freopen("wash.in", "r", stdin);
      freopen("wash.out", "w", stdout);
    get(n);
    int x;
    For(i, 1, n)
    {
        get(x);
        if (!apr[x]) apr[x] = ++m;
        d[apr[x]].push_back(i);
    }
    For(i, 1, m)
    if (d[i].size() == 2) //只出现 2 次的字符只有一种分配方法
    {
        l1[i] = l1[i + m] = l2[i] = l2[i + m] = d[i][0];
        r1[i] = r1[i + m] = r2[i] = r2[i + m] = d[i][1];
    }
    else
    {
        l1[i] = l1[i + m] = d[i][0];
        l2[i] = r1[i + m] = d[i][1];
        r1[i] = l2[i + m] = d[i][2];
        r2[i] = r2[i + m] = d[i][3];
    }
    For(i, 1, (m << 1) - 1)
    For(j, i + 1, m << 1)
    if (i + m ^ j && check(i, j))
    {
        Link(i, oppo(j));
        Link(j, oppo(i));
    }
    For(i, 1, m << 1)
    if (!dfn[i]) Tarjan(i);
    For(i, 1, m)  //Tarjan 给连通块的编号恰为拓扑序逆序
        ans[r1[bel[i] < bel[i+m] ? i : i+m]] = ans[r2[i]] = true;
    For(i, 1, n)
        putchar(ans[i] ? '1' : '0');
    putchar('\n');
    fclose(stdin), fclose(stdout);
    return 0;
}
```

第四部分　高级数据结构

第 1 章　左偏树

631.次短路径(pal,1s,256MB)

【参考程序】

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

template <class T>
inline void read(T &res)
{
    char ch;
    while (ch = getchar(), !isdigit(ch));
    res = ch ^ 48;
    while (ch = getchar(), isdigit(ch))
        res = res * 10 + ch - 48;
}

template <class T>
inline void put(T x)
{
    if (x > 9) put(x / 10);
    putchar(x % 10 + 48);
}

using std::vector;
using std::priority_queue;
const int N = 1e5 + 5;
const int M = 4e5 + 5;
const int Maxn = 0x3f3f3f3f;
int n, m, E;
int dep[N], anc[N][20], rt[N], ans[N];
int key[M], dist[M], dis[N];
int lc[M], rc[M], pa[M], pb[M], pc[M];
bool vis[M];
vector<int> v[N], add[N], del[N];
```

```
struct point
{
    int s, t;

    point() {}
    point(int S, int T):
        s(S), t(T) {}

    inline bool operator < (const point &a) const
    {
        return s > a.s;
    }
};
priority_queue<point> que;

struct Edge
{
    int to, cst; Edge *nxt;
}p[M], *lst[N], *P = p;

inline void Link(int x, int y, int z)
{
    (++P)->nxt = lst[x]; lst[x] = P; P->to = y; P->cst = z;
    (++P)->nxt = lst[y]; lst[y] = P; P->to = x; P->cst = z;
}

inline int Merge(int x, int y)
{
    if (!x || !y)
        return x + y;
    if (key[x] > key[y])
        std::swap(x, y);
    rc[x] = Merge(rc[x], y);
    if (dist[lc[x]] < dist[rc[x]])
        std::swap(lc[x], rc[x]);
    dist[x] = dist[rc[x]] + 1;
    return x;
}

inline void Dijkstra() //堆优化 Dijkstra 求最短路
{
    for (int i = 1; i <= n; ++i)
        dis[i] = Maxn;
    que.push(point(dis[1] = 0, 1));
```

```
    for (int i = 1, x, y; i <= n; ++i)
    {
        while (!que.empty() && vis[que.top().t])
            que.pop();
        if (que.empty())
            break;
        else
        {
            vis[x = que.top().t] = true;
            que.pop();
        }
        for (Edge *e = lst[x]; e; e = e->nxt)
            if (y = e->to, dis[y] > dis[x] + e->cst)
                que.push(point(dis[y] = dis[x] + e->cst, y));
    }
}

inline int query_LCA(int x, int y) //倍增求 LCA
{
    if (x == y)
        return x;
    if (dep[x] < dep[y])
        std::swap(x, y);
    for (int i = 17; i >= 0; --i)
    {
        if (dep[anc[x][i]] >= dep[y])
            x = anc[x][i];
        if (x == y)
            return x;
    }
    for (int i = 17; i >= 0; --i)
        if (anc[x][i] != anc[y][i])
            x = anc[x][i], y = anc[y][i];
    return anc[x][0];
}

inline void init_LCA(int x)
{
    dep[x] = dep[anc[x][0]] + 1;
    for (int i = 0; anc[x][i]; ++i)
        anc[x][i + 1] = anc[anc[x][i]][i];
    for (int i = 0, im = v[x].size(); i < im; ++i)
    {
        int y = v[x][i];
```

```
        if (y == anc[x][0])
            continue;
        anc[y][0] = x;
        init_LCA(y);
    }
}

inline void Dfs(int x)
{
    int u = 0;
    for (int i = 0, im = v[x].size(); i < im; ++i)
    {
        int y = v[x][i];
        if (y == anc[x][0])
            continue;
        Dfs(y);
        u = Merge(rt[y], u);
    }
    for (int i = 0, im = add[x].size(); i < im; ++i)
        u = Merge(u, add[x][i]);
    for (int i = 0, im = del[x].size(); i < im; ++i)
        vis[del[x][i]] = true;
    while (u && vis[u])
        u = Merge(lc[u], rc[u]);
    rt[x] = u;
    ans[x] = !u ? -1 : key[u] - dis[x];
}

int main()
{
    freopen("pal.in", "r", stdin);
    freopen("pal.out", "w", stdout);
    read(n); read(m);
    dist[0] = -1;
    for (int i = 1; i <= m; ++i)
    {
        read(pa[i]);
        read(pb[i]);
        read(pc[i]);
        Link(pa[i], pb[i], pc[i]);
    }

    Dijkstra();
```

```
    for (int i = 1; i <= m; ++i) //建最短路径树
    {
        if (dis[pa[i]] == Maxn && dis[pb[i]] == Maxn)
            continue;
        if (dis[pa[i]] + pc[i] == dis[pb[i]])
            v[pa[i]].push_back(pb[i]);
        else if (dis[pb[i]] + pc[i] == dis[pa[i]])
            v[pb[i]].push_back(pa[i]);
    }

    memset(vis, false, sizeof(vis));
    memset(ans, 255, sizeof(ans));
    init_LCA(1);

    for (int i = 1; i <= m; ++i)
    {
        if (dis[pa[i]] == Maxn && dis[pb[i]] == Maxn)
            continue;
        if (dis[pa[i]] + pc[i] != dis[pb[i]]
        && dis[pb[i]] + pc[i] != dis[pa[i]])
        {
            int x = pa[i],
                y = pb[i],
                z = query_LCA(x, y);

            add[x].push_back(++E);
            key[E] = dis[x] + dis[y] + pc[i];
            del[z].push_back(E);

            add[y].push_back(++E);
            key[E] = dis[x] + dis[y] + pc[i];
            del[z].push_back(E);
        }
    }

    Dfs(1);
    for (int i = 2; i <= n; ++i)
        if (ans[i] == -1)
            puts("-1");
        else
            put(ans[i]), putchar('\n');
    fclose(stdin); fclose(stdout);
    return 0;
}
```

第 2 章　平衡树 splay

641.神秘物质 (atom,2s,256MB)

【参考程序】

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

template <class T>
inline void read(T &x)
{
    static char ch;
    while (!isdigit(ch = getchar()));
    x = ch - '0';
    while (isdigit(ch = getchar()))
        x = x * 10 + ch - '0';
}

template <class T>
inline void putint(T x)
{
    static char buf[15], *tail = buf;
    if (!x) putchar('0');
    else
    {
        if (x < 0) x = ~x + 1, putchar('-');
        for (; x; x /= 10) *++tail = x % 10 + '0';
        for (; tail != buf; --tail) putchar(*tail);
    }
}

template <class T>
inline void tense(T &x, const T &y)
{
    if (x > y)
        x = y;
}

template <class T>
inline void relax(T &x, const T &y)
{
    if (x < y)
        x = y;
}

constintMaxN = 2e5 + 5;
```

```
constint INF = 0x3f3f3f3f;

struct node
{
    int fa, lc, rc, val, dlt, sze;
    int mind, minv, maxv;

    #define fa(x) tr[x].fa
    #define lc(x) tr[x].lc
    #define rc(x) tr[x].rc
    #define val(x) tr[x].val //val 表示结点 x 的权值
    #define dlt(x) tr[x].dlt //dlt 表示结点 x 和它在序列中下一个元素的差值
    #define sze(x) tr[x].sze//sze 表示子树大小
    #define mind(x) tr[x].mind //mind 表示子树中最小的相邻元素差值
    #define minv(x) tr[x].minv //minv 表示子树中最小的权值
    #define maxv(x) tr[x].maxv //maxv 表示子树中最大的权值
}tr[MaxN];

intrt, nT;
char s[10];
int n, m;
int a[MaxN];

inline bool which(int x)
{
    return rc(fa(x)) == x;
}

inline void upt(int x)  //用 x 的左右儿子更新结点 x 的信息
{
    sze(x) = 1;
    mind(x) = dlt(x);
    minv(x) = maxv(x) = val(x);
    if (lc(x))
    {
        sze(x) += sze(lc(x));
        tense(mind(x), mind(lc(x)));
        tense(minv(x), minv(lc(x)));
        relax(maxv(x), maxv(lc(x)));
    }
    if (rc(x))
    {
        sze(x) += sze(rc(x));
        tense(mind(x), mind(rc(x)));
```

```
        tense(minv(x), minv(rc(x)));
        relax(maxv(x), maxv(rc(x)));
    }
}

inline void rotate(int x)
{
    int y = fa(x), z = fa(fa(x));
    int b = lc(y) == x ?rc(x) : lc(x);
    fa(x) = z, fa(y) = x;
    if (b) fa(b) = y;
    if (lc(z) == y)
        lc(z) = x;
    else
        rc(z) = x;
    if (lc(y) == x)
        rc(x) = y, lc(y) = b;
    else
        lc(x) = y, rc(y) = b;
    upt(y);
}

inline void Splay(int x, int tar) //将 x 旋转到 tar 下面
{
    while (fa(x) != tar)
    {
        if (fa(fa(x)) != tar)
            rotate(which(x) == which(fa(x)) ? fa(x) : x);
        rotate(x);
    }
    upt(x);
    if (!tar)
        rt = x;
}

inline void build(int&x,int l,int r)//刚开始的序列我们可以直接分治建树
{
    if (l > r) return;
    int mid = l + r >> 1;
    x = ++nT;
    val(x) = a[mid];
    dlt(x) = abs(a[mid] - a[mid + 1]);
    build(lc(x), l, mid - 1);
    build(rc(x), mid + 1, r);
```

```
    if (lc(x)) fa(lc(x)) = x;
    if (rc(x)) fa(rc(x)) = x;
    upt(x);
}

inline int query_kth(int k)//查询目前在序列中编号为 k 的结点
{
    int x = rt;
    while (k)
    {
        if (sze(lc(x)) + 1 == k)
            return x;
        else if (sze(lc(x)) + 1 > k)
            x = lc(x);
        else
        {
            k -= sze(lc(x)) + 1;
            x = rc(x);
        }
    }
}

inline int split(int l, int r) //提取出区间[l,r]对应的子树
{
    int t1 = query_kth(l), t2 = query_kth(r + 2);
    Splay(t1, 0), Splay(t2, t1);

    return lc(t2);
}

int main()
{
    freopen("atom.in", "r", stdin);
    freopen("atom.out", "w", stdout);

    mind(0) = minv(0) = INF;
    read(n), read(m);
    for (inti = 1; i<= n; ++i)
        read(a[i]);
    build(rt, 0, n + 1);

    for (inti = 1; i<= m; ++i)
    {
        scanf("%s", s + 1);
```

```
        int x, y;
        read(x), read(y);

        if (s[2] == 'e') //合并两个相邻结点
        {
            int u = split(x, x + 1), v = fa(u);
            val(u) = y;
            dlt(fa(v)) = abs(y - val(fa(v)));
            dlt(u) = abs(y - val(v));
        lc(u) = rc(u) = 0;

            upt(u);
            Splay(u, 0);
        }
        else if (s[1] == 'i')//插入结点
        {
            int u = split(x, x), v = fa(u);
            val(rc(u) = ++nT) = y;
            fa(nT) = u;
            dlt(u) = abs(y - val(u));
            dlt(nT) = abs(y - val(v));
            upt(nT);
            Splay(nT, 0);
        }
        else if (s[2] == 'a')//询问最大极差
        {
            int u = split(x, y);
            putint(maxv(u) - minv(u)), putchar('\n');
        }
        else //询问最小极差
        {
            int u = split(x, y - 1);
            putint(mind(u)), putchar('\n');
        }
    }
    return 0;
}
```

642.生产工作(train,4s,1GB)

【参考程序】

```
#include <bits/stdc++.h>
#define rep(i,n) for (int i=0;i<n;i++)
#define For(i,n) for (int i=1;i<=n;i++)
#define FOR(i,a,b) for (int i=a;i<=b;i++)
#define FORD(i,a,b) for (int i=a;i>=b;i--)
#define MOD(x) if (x >= mo) x %= mo;
#define DELMOD(x) if (x >= mo) x -= mo;

using namespace std;
typedef long long LL;
const int mo = 1000000007;
const int TREE = 100000;
const int SZ = 50000;

int sz, root, now;
int fa[SZ], c[SZ][2];
short h[SZ], lmost[SZ], rmost[SZ], high[SZ];
LL s[SZ], cnt[SZ], lans[SZ], rans[SZ], lcnt[SZ], rcnt[SZ];
bool rev[SZ];

inline void Update(int x)
{
    int &l = c[x][0], &r = c[x][1];
    s[x] = s[l] + s[r] + 1;
    lmost[x] = rmost[x] = high[x] = h[x];
    cnt[x] = 1;
    lans[x] = rans[x] = lcnt[x] = rcnt[x] = 0;
    if (l)
    {
        lmost[x] = lmost[l];
        if (high[l] > high[x])
            high[x] = high[l], cnt[x] = cnt[l];
        else if (high[l] == high[x])
            cnt[x] += cnt[l];

        lans[x] = (lans[x]+lans[l]+(s[r]+1)%mo*(lcnt[l]%mo))%mo;
        rans[x] += rans[l];
        DELMOD(rans[x]);

        lcnt[x] += lcnt[l];
        rcnt[x] += rcnt[l];
```

```
        if (rmost[l] > h[x])
        {
            lans[x] += s[r] + 1;
            MOD(lans[x]);
            lcnt[x]++;
        }

        else if (rmost[l] < h[x])
        {
            rans[x] += s[l];
            MOD(rans[x]);
            rcnt[x]++;
        }
    }

    if (r)
    {
        rmost[x] = rmost[r];
        if (high[r] > high[x])
            high[x] = high[r], cnt[x] = cnt[r];
        else if (high[r] == high[x])
            cnt[x] += cnt[r];

        lans[x] += lans[r];
        DELMOD(lans[x]);
        rans[x] = (rans[x]+rans[r]+rcnt[r]%mo*((s[l]+1)%mo))%mo;

        lcnt[x] += lcnt[r];
        rcnt[x] += rcnt[r];

        if (lmost[r] > h[x])
        {
            rans[x] += s[l] + 1;
            MOD(rans[x]);
            rcnt[x]++;
        }
          else if (lmost[r] < h[x])
        {
            lans[x] += s[r];
            MOD(lans[x]);
            lcnt[x]++;
        }
    }
}
```

```
inline void Reverse(int x)
{
    rev[x] = !rev[x];
    swap(c[x][0], c[x][1]);
    swap(lmost[x], rmost[x]);
    swap(lcnt[x], rcnt[x]);
    swap(lans[x], rans[x]);
}

inline void Down(int x)
{
    if (!rev[x]) return;
    rev[x] = false;
    if (c[x][0])
        Reverse(c[x][0]);
    if (c[x][1])
        Reverse(c[x][1]);
}

int FindMax(int x, int m)
{
    Down(x);
    int &l = c[x][0], &r = c[x][1];
    if (high[l] == high[now] && cnt[l] >= m)
        return FindMax(l, m);
    if (high[l] == high[now])
    {
        if (cnt[l] >= m)
            return FindMax(l, m);
        m -= cnt[l];
    }
    if (h[x] == high[now])
        m--;
    if (!m)
        return x;
    return FindMax(r, m);
}

inline int kth(int rk, int node = -1)
{
    int x;
    if (node == -1)
        x = root;
```

```
    else
        x = node;
    while (1) {
        Down(x);
        if (s[c[x][0]]+1==rk) return x;
        if (s[c[x][0]]+1>rk) x=c[x][0];
        else{
            rk-=s[c[x][0]]+1;
            x=c[x][1];
        }
    }
}

inline void Rotate(int &root,int x)
{
    int y,z,p,q;
    y=fa[x];z=fa[y];
    if (c[y][0]==x) p=0; else p=1;
    q=p^1;
    if (y==root) root=x;
    else if (c[z][0]==y) c[z][0]=x;
                    else c[z][1]=x;
    fa[x]=z;fa[y]=x;fa[c[x][q]]=y;
    c[y][p]=c[x][q];c[x][q]=y;
    Update(y);Update(x);
}

inline void Splay(int &root,int x)
{
    int y,z;
    while (x!=root){
        y=fa[x];z=fa[y];
        if (y!=root)
          if ((c[y][0]==x)^(c[z][0]==y)) Rotate(root,x);
                                    else Rotate(root,y);
        Rotate(root,x);
    }
}

int main()
{
    freopen("train.in", "r", stdin);
    freopen("train.out", "w", stdout);
    int hh, l, r, x, pos, u, v, q;
```

```
    char cmd;
    sz = 2;
    c[1][1] = 2; fa[2] = 1;
    s[1] = 2; s[2] = 1;
    root = 1;
    scanf("%d", &q);
    rep(qq, q)
    {
        scanf(" %c", &cmd);
        switch(cmd)
        {
        case 'I':
            scanf("%d%d", &hh, &x);
            u = kth(x+1);
            Splay(root,u);
            sz++; h[sz] = hh;
            fa[sz] = u; fa[c[u][1]] = sz;
            c[sz][1] = c[u][1]; c[u][1] = sz;
            Update(sz); Update(u);
            break;
        case 'Q':
            scanf("%d%d", &l, &r);
            u=kth(l);v=kth(r+2);
            Splay(root,u);Splay(c[u][1],v);
            now = c[v][0];
            u = FindMax(now, (cnt[now] + 1) / 2);
            Splay(c[v][0], u);
            l = c[u][0], r = c[u][1];
            printf("%I64d\n", (lans[l]+lcnt[l]+rans[r]+rcnt[r])%mo);
            break;
        case 'R':
            scanf("%d%d", &l, &r);
            u=kth(l);v=kth(r+2);
            Splay(root,u);Splay(c[u][1],v);
            Reverse(c[v][0]);
            break;
        }
    }
    return 0;
}
```

643.机器决斗（fittest，2s，512MB）

【参考程序】

```
#include <cstdio>
#include <algorithm>
typedef long long  ll;
const  int maxn = 330000;
int ld[maxn], ra[maxn];
ll  s[maxn];
ll  ans, res;
int n, ATK;

struct  qs
{
    int a, p;
}   q[maxn];

bool   operator<(qs x, qs y) {return x.p * y.a < x.a * y.p;}
inline void   renew(ll &x, ll y) {if (x < y) x = y;}

struct  cljsplay
{
    struct  node
    {
        ll  x, y, lx, ly;
        node   *p, *s[2];
        node() {p = s[0] = s[1] = 0;}
        bool   getlr() {return p->s[1] == this;}
        node   *link(int w, node *p) {s[w]=p;if(p)p->p=this;return this;}
    }   *root;

    void   rot(node *p)
    {
        node *q = p->p->p;
        p->getlr()  ?  p->link(0,p->p->link(1,  p->s[0]))  :  p->link(1,
p->p->link(0, p->s[1]));
        if (q) q->link(q->s[1]==p->p,p); else (root=p)->p=0;
    }

    void   splay(node *p)//将结点 p 旋到根
    {
        while (p->p && p->p->p)
            p->getlr()==p->p->getlr()?(rot(p->p),rot(p)):(rot(p),rot(p));
        if (p->p) rot(p);
    }
```

```
void   splay(node *p, node *tar)//将结点 p 旋到 tar 的下面
{
    while (p->p != tar && p->p->p != tar)
        p->getlr()==p->p->getlr()?(rot(p->p),rot(p)):(rot(p),rot(p));
    if (p->p != tar) rot(p);
}

void   preset()
{
    root=new node; root->x=0; root->y=0; root->lx=0; root->ly=-1;
}

node*  prev(node *p)
{
    if (!p) return 0;
    splay(p);
    if (!p->s[0]) return 0;
    node *q = p->s[0];
    for (; q->s[1]; q=q->s[1]);
    splay(q);
    return q;
}

node*  succ(node *p)
{
    if (!p) return 0;
    splay(p);
    if (!p->s[1]) return 0;
    node *q = p->s[1];
    for (; q->s[0]; q=q->s[0]);
    splay(q);
    return q;
}

void   insert(ll x, ll y)//插入一个点(x,y)
{
    node *p = root, *p1, *p2;
    while (p)
    {
        p1 = p;
        p = p->s[p->x < x];
    }
    p = new node; p->x = x; p->y = y;
```

```
    p1->link(p1->x < x, p);
    splay(p);
    if ((p1 = prev(p)) && (p2 = succ(p)))
    {
        if ((p2->x-p->x)*(p1->y-p->y)>=(p2->y-p->y)*(p1->x-p->x))
        {
            splay(p1);
            splay(p2, p1);
            p2->s[0] = 0;
            return;
        }
    }
    while (p2 = prev(p1 = prev(p)))
    {
        if ((p->x-p1->x)*(p2->y-p1->y)>=(p->y-p1->y)*(p2->x-p1->x))
        {
            splay(p2);
            splay(p, p2);
            p->s[0] = 0;
        }   else   break;
    }
    if (p1 = prev(p)) p->lx = p1->x, p->ly = p1->y;
    while (p2 = succ(p1 = succ(p)))
    {
        if ((p->x-p1->x)*(p2->y-p1->y)>=(p->y-p1->y)*(p2->x-p1->x))
        {
            splay(p2);
            splay(p, p2);
            p->s[0] = 0;
        }   else   break;
    }
    if (p1 = succ(p)) p1->lx = p->x, p1->ly = p->y;
}

ll  get(int k) //二分斜率
{
    ll  res = 0;
    node *p = root;
    while (p)
    {
        renew(res, p->y - k * p->x);
        p = p->s[p->y - k * p->x >= p->ly - k * p->lx];
    }
    return res;
```

```
    }
}tool1;

int main()
{
    freopen("fittest.in", "r", stdin);
    freopen("fittest.out", "w", stdout);
    scanf("%d%d", &n, &ATK);
    for (int i=1; i<=n; i++)
    {
        scanf("%d%d", &q[i].a, &q[i].p);
        q[i].p = (q[i].p + ATK - 1) / ATK;
    }
    std::sort(q + 1, q + n + 1);//按 q[i].p/q[i].a 排序
    ld[0] = 0;
    for (int i=1; i<=n; i++) ld[i] = ld[i-1] + q[i].p;
    ra[n+1] = 0;
    for (int i=n; i; i--) ra[i] = ra[i+1] + q[i].a;
    for (int i=1; i<=n; i++)
        s[i] = (ld[i]-1) * (ll)q[i].a + ra[i+1] * (ll)q[i].p;
    ans = 0;
    for (int i=1; i<=n; i++) ans += (ld[i]-1) * (ll)q[i].a;
    res = 0;
    tool1.preset();
    for (int i=n-1; i; i--)
    {
        tool1.insert(q[i+1].a, s[i+1]);//插入决策点
        renew(res, s[i] + tool1.get(q[i].p));//在凸壳上二分
    }
    printf("%I64d\n", ans - res);
    return 0;
}
```

第 3 章　重量平衡树

651.车位选择（park，1s，256MB）

【参考程序】

```
#include <bits/stdc++.h>
#define For(i, a, b) for (int i = a, bb = b; i <= bb; ++i)
#define Rof(i, a, b) for (int i = a, bb = b; i >= bb; --i)
#define s64 long long

template <class T>
inline void get(T &res)
{
    char ch;
    bool bo = false;
    while (ch = getchar(), !isdigit(ch))
        if (ch == '-') bo = true;
    res = ch ^ 48;
    while (ch = getchar(), isdigit(ch))
        res = (res << 1) + (res << 3) + (ch ^ 48);
    if (bo) res = ~ res + 1;
    return;
}

template <class T>
inline void _put(T x)
{
    if (x > 9) _put(x / 10);
    putchar(x % 10 + '0');
    return;
}

template <class T>
inline void put(T x, char ch) {
    if (x < 0) {
        putchar('-');
        x = ~ x + 1;
    }
    _put(x);
    putchar(ch);
    return;
}

const int MaxN = 1e6 + 5;
int n, m, rt;
struct point
{
    int lc, rc, sze, key, pri, val, dis, mx;
    #define lc(x) tr[x].lc
    #define rc(x) tr[x].rc
    #define s(x) tr[x].sze
    #define k(x) tr[x].key    //权值
    #define p(x) tr[x].pri    //优先级
    #define v(x) tr[x].val    //在该节点与其前驱间插入新节点的距离最小值
    #define d(x) tr[x].dis    //该节点与其前驱的距离
    #define mx(x) tr[x].mx    //该节点的子树中 val 最大的节点编号
```

```
} tr[MaxN];

inline void upt(int x)
{
    s(x) = s(lc(x)) + s(rc(x)) + 1;
    mx(x) = x;
    if (lc(x) && v(mx(x)) <= v(mx(lc(x))))
        mx(x) = mx(lc(x));
    if (rc(x) && v(mx(x)) < v(mx(rc(x))))
        mx(x) = mx(rc(x));
    return;
}

inline void zig(int &x) //右旋
{
    int y = lc(x);
    lc(x) = rc(y);
    rc(y) = x;
    upt(x);
    upt(y);
    x = y;
    return;
}


inline void zag(int &x) //左旋
{
    int y = rc(x);
    rc(x) = lc(y);
    lc(y) = x;
    upt(x);
    upt(y);
    x = y;
    return;
}

inline void insert(int &x,int now,int key,int val,int dis)
{  //插入权值为 key 的新节点 now
    if (!x)
    {
        x = now;
        lc(x) = rc(x) = 0;
        s(x) = 1;
        k(x) = key;
        v(x) = val;
        d(x) = dis;
        p(x) = rand();
        mx(x) = x;
        return;
    }
    if (key < k(x))
    {
        insert(lc(x), now, key, val, dis);
        upt(x);
        if (p(lc(x)) < p(x)) zig(x);
    }
    else
    {
```

```
        insert(rc(x), now, key, val, dis);
        upt(x);
        if (p(rc(x)) < p(x)) zag(x);
    }
    return;
}

inline void change(int x, int key, int val, int dis)
{  //更改权值为 key 的节点信息
    if (key == k(x))
    {
        v(x) = val;
        d(x) = dis;
        upt(x);
        return;
    }
    if (key < k(x))
        change(lc(x), key, val, dis);
    else
        change(rc(x), key, val, dis);
    upt(x);
    return;
}

inline int find_first() //查询权值最小的点
{
    int x = rt;
    while (lc(x)) x = lc(x);
    return x;
}

inline int find_last() //查询权值最大的点
{
    int x = rt;
    while (rc(x)) x = rc(x);
    return x;
}

inline int find_suf(int key) //查询权值为 key 的节点的后继
{
    int x = rt, res = 0;
    while (x)
        if (key < k(x)) res = x, x = lc(x);
        else x = rc(x);
    return res;
}

inline void Delete(int &x, int key) //删除权值为 key 的节点
{
    if (key == k(x))
    {
        if (!lc(x) || !rc(x)) x = lc(x) + rc(x);
        else
        {
            if (p(lc(x)) < p(rc(x))) zig(x);
            else zag(x);
            Delete(x, key);
        }
```

```
        return;
    }
    if (key < k(x)) Delete(lc(x), key);
    else Delete(rc(x), key);
    upt(x);
    return;
}

int main()
{
    freopen("park.in", "r", stdin);
    freopen("park.out", "w", stdout);

    srand(666);
    get(n), get(m);
    int opt, now;
    while (m--)
    {
        get(opt), get(now);
        if (opt & 1)
        {
            if (s(rt))
            {
                int lst = find_last();
                if (v(mx(rt)) >= n - k(lst))
                {
                    int x = mx(rt);
                    if (find_first() == x) //处理新节点权值为 1 的情况
                    {
                        insert(rt, now, 1, 0, 0);
                        change(rt, k(x), v(x) >> 1, d(x));
                    }
                    else
                    {
                        insert(rt, now, k(x)-d(x)+v(x),v(x)>>1, v(x));
                        change(rt, k(x), d(x)-v(x) >> 1, d(x)-v(x));
                    }
                }
                else //处理新节点权值为 n 的情况
                    insert(rt, now, n, n - k(lst) >> 1, n-k(lst));
            }
            else      //处理树为空的情况
                insert(rt, now, 1, 0, 0);
            put(k(now), '\n');
        }
        else
        {
            int suf = find_suf(k(now)), dis = d(now) + d(suf);
            Delete(rt, k(now));
            if (suf)  //若该节点有后继，则更改后继节点信息
                change(rt,k(suf),find_first()==suf ? dis : dis>>1,dis);
        }
    }
        fclose(stdin), fclose(stdout);
        return 0;
}
```

652.集合比较(comparison,1s,512MB)

【参考程序】

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

template <class T>
inline void read(T &res)
{
    char ch; bool flag = false; res = 0;
    while (ch = getchar(), !isdigit(ch) && ch != '-');
    ch == '-' ? flag = true : res = ch ^ 48;
    while (ch = getchar(), isdigit(ch))
        res = res * 10 + ch - 48;
    flag ? res = -res : 0;
}

template <class T>
inline void put(T x) {
    if (x > 9) put(x / 10);
    putchar(x % 10 + 48);
}

const double Maxn = 1ll << 60;
const double alpha = 0.7;
const int N = 5e4 + 5;
double lef[N], rig[N], val[N];
int cur[N], fac[N], sze[N], cnt[N], id[N];
int lc[N], rc[N], fa[N], sub1[N], sub2[N];
int n, T, cm, rt;

inline void Update(int x)
{
    fac[x] = fac[lc[x]] + fac[rc[x]] + cnt[x];
    sze[x] = sze[lc[x]] + sze[rc[x]] + 1;
}

template <class T>
inline T Max(T x, T y) {return x > y ? x : y;}

inline bool isBalance(int x)
{
    return alpha * sze[x] > (double)Max(sze[lc[x]], sze[rc[x]]);
}

inline void Travel(int x)
{
    if (!x) return;
    Travel(lc[x]);
    cur[++cm] = x;
    Travel(rc[x]);
}

inline void lift_up(int &x, int l, int r, double u, double v)
```

```
{
    if (l > r) return ;
    int mid = l + r >> 1;
    x = cur[mid];
    lef[x] = u; rig[x] = v;
    val[x] = (lef[x] + rig[x]) / 2.0;
    lift_up(lc[x], l, mid - 1, lef[x], val[x]);
    lift_up(rc[x], mid + 1, r, val[x], rig[x]);
    fa[lc[x]] = fa[rc[x]] = x;
    Update(x);
}

inline void reBuild(int &x, double u, double v)
{ //拍扁重建法，重构结点 x 的整个子树
    cm = 0;
    Travel(x);
    for (int i = 1, w; w = cur[i], i <= cm; ++i)
        sze[w] = 1, fac[w] = cnt[w], lc[w] = rc[w] = fa[w] = 0;
    lift_up(x, 1, cm, u, v);
}

inline int Insert(int v1, int v2, int t)
{ //插入新的集合
    int x = rt, y = 0, dir, res = 0;
    while (x)
    {
        ++fac[y = x];
        if (val[sub1[x]] == val[v1] && val[sub2[x]] == val[v2])
        {
            ++cnt[x];
            id[t] = x;
            return res + fac[lc[x]] + cnt[x];
        }
        if (val[sub1[x]] < val[v1] ||
            val[sub1[x]] == val[v1] && val[sub2[x]] < val[v2])
            res += fac[lc[x]] + cnt[x], x = rc[x], dir = 1;
        else
            x = lc[x], dir = 0;
    }
    fa[id[t] = x = ++T] = y;
    sub1[x] = v1; sub2[x] = v2;
    cnt[x] = sze[x] = fac[x] = 1;
    if (y)
    {
        if (dir == 0)
        {
            lc[y] = x;
            lef[x] = lef[y];
            rig[x] = val[y];
            val[x] = (lef[x] + rig[x]) / 2.0;
        }
        else
        {
```

```
            rc[y] = x;
            lef[x] = val[y];
            rig[x] = rig[y];
            val[x] = (lef[x] + rig[x]) / 2.0;
        }
    }

    int w = 0;
    while (fa[x])
    {
        x = fa[x];
        ++sze[x];
        !isBalance(x) ? w = x : 0;
    }
    if (w)
    {
        int fw = fa[w];
        if (fw)
        {
            if (val[w] < val[fw])
            {
                reBuild(w, lef[fw], val[fw]);
                fa[w] = fw;
                lc[fw] = w;
            }
            else
            {
                reBuild(w, val[fw], rig[fw]);
                fa[w] = fw;
                rc[fw] = w;
            }
        }
        else
        {
            if (val[w] < val[fw])
                reBuild(w, 0, Maxn / 2.0);
            else
                reBuild(w, Maxn / 2.0, Maxn);
            rt = w;
        }
    }
    return res + 1;
}

int main()
{
    freopen("comparison.in", "r", stdin);
    freopen("comparison.out", "w", stdout);
    read(n);
    rt = 1;
    rc[1] = 2; T = 2;
    fa[2] = 1;
    fac[2] = sze[2] = cnt[2] = 1;
```

```
    fac[1] = sze[1] = 2; cnt[1] = 1;
    lef[1] = 0; rig[1] = Maxn;
    sub1[1] = sub2[1] = 1;
    val[1] = (lef[1] + rig[1]) / 2.0;
    lef[2] = val[1]; rig[2] = Maxn;
    val[2] = (lef[2] + rig[2]) / 2.0;
    sub1[2] = sub2[2] = 2;

    id[0] = 1;
    id[n + 1] = 2;
    for (int i = 1, x, y; i <= n; ++i)
    {
        read(x); read(y);
        put(Insert(id[x], id[y], i)), putchar('\n');
    }
    fclose(stdin); fclose(stdout);
    return 0;
}
```

653.冒泡排序(fable,2s,256MB)

【参考程序】

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

using namespace std;

template <class t>
inline void read(t & res)
{
    char ch;
    while (ch = getchar(), !isdigit(ch));
    res = ch ^ 48;
    while (ch = getchar(), isdigit(ch))
    res = res * 10 + (ch ^ 48);
}

const int e = 2e5 + 5;
struct point
{
    int id, v;
}a[e];
int n, lc[e], rc[e], p[e], sze[e], b[e], k, rt, pos[e], c[e];

inline bool cmp(const point &a, const point &b)
{  //从大到小排序，相同元素排在后面的较大
    return a.v != b.v ? a.v>b.v : a.id > b.id;
}

inline void upt(int x)
{
    sze[x] = 1;
    if (lc[x]) sze[x] += sze[lc[x]];
    if (rc[x]) sze[x] += sze[rc[x]];
}
```

```
inline void zig(int &x)
{
    int y = lc[x];
    lc[x] = rc[y];
    rc[y] = x;
    upt(x);
    upt(y);
    x = y;
}

inline void zag(int &x)
{
    int y = rc[x];
    rc[x] = lc[y];
    lc[y] = x;
    upt(x);
    upt(y);
    x = y;
}

inline void insert(int &x, int id) // 插入初始位置为 id 的元素
{
    if (!x)
    {
        x = id; // 点编号即初始位置
        p[x] = rand();
        sze[x] = 1;
        return;
    }
    if (id < x)
    {
        insert(lc[x], id);
        if (p[lc[x]] < p[x]) zig(x);
    }
    else
    {
        insert(rc[x], id);
        if (p[rc[x]] < p[x]) zag(x);
    }
    upt(x);
}

inline int find(int rk) // 查找排名为 k 的元素
{
    int x = rt;
    while (x) {
        if (sze[lc[x]] + 1 == rk) return x;
        if (sze[lc[x]] + 1 >rk) x = lc[x];
        else rk -= sze[lc[x]] + 1, x = rc[x];
    }
}
```

```
inline int ask(int v) //二分求出目前 treap 上有几个数初始排在 v 前面
{
    int x = rt, res = 0;
    while (x) {
        if (x < v) res += sze[lc[x]] + 1, x = rc[x];
        else x = lc[x];
    }
    return res;
}

int main()
{
    freopen("fable.in", "r", stdin);
    freopen("fable.out", "w", stdout);
    srand(19260817);
    int i;
    read(n); read(k);
    for (i = 1; i<= n; i++) a[i].id = i, read(a[i].v);
    sort(a + 1, a + n + 1, cmp);
    for (i = 1; i<= n; i++)
    {
        int id = a[i].id, v = a[i].v;
        c[i] = ask(id); //求出 a[i]前面比 a[i]大的数的个数
        if (k <= c[i]) b[id - k] = v;
        else
        {
            if (sze[rt] < k) pos[i] = n + 1;
            else pos[i] = find(k);
            //求出从左往右第 k 个比 a[i]大的数 pos[i]，若不存在 pos[i]，
pos[i]=n+1
        }
        insert(rt, id); //先查询后插入
    }
    rt = 0;
    for (i = 1; i<= n; i++) lc[i] = rc[i] = p[i] = sze[i] = 0;
    for (i = n; i>= 1; i--)
    {
        int id = a[i].id;
        if (pos[i])
        {
            int s2 = ask(pos[i]), s1 = ask(id+1), res = s2-s1;
            // 求出初始排在 a[i]前面且比 a[i]小的数的个数 s1
            // 以及初始排在 pos[i]前面且比 a[i]小的数的个数 s2
            // 减得到 res 即 a[i]和 pos[i]之间比 a[i]小的数的个数
            b[id - c[i] + res] = a[i].v;
        }
        insert(rt, id);
    }
    for (i = 1; i<= n; i++) printf("%d\n", b[i]);
    fclose(stdin); fclose(stdout);
    return 0;
}
```

第 4 章　可持久化数据结构

661.前 K 大串(seq,2s,1024MB)

【参考程序】

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

using namespace std;
#define ll long long
template <class t>

inline void read(t & res)
{
    res = 0;
    char ch; int f = 1;
    while (ch = getchar(), (ch < '0' || ch > '9') && ch != '-');
    if (ch == '-') f = -1;
    else res = ch - 48;
    while (ch = getchar(), ch >= '0' && ch <= '9')
    res = res * 10 + ch - 48;
    if (f == -1) res = ~res + 1;
}

const int e = 2e5 + 5, o = 2e7 + 5;
const ll inf = 1e16;

struct point
{
    int x, y, id, r;
    ll sum;
};

struct node
{
    int l, r, id;
    ll val, add;
}c[o];

struct work
{
    ll val;
    int id;
};
```

```
int n,k,tot,h[e],b[e],cnt,id[e],nxt[e],pos[e],rt[e],pool,lst[e],a[e],rti;
ll z[e];
inline bool operator < (point a, point b)
{
    return a.sum < b.sum;
}
priority_queue<point>q;

inline void build(int l, int r, int &x) //建树
{
    x = ++pool;
    c[x].val=-inf; //初始化成最小值，因为a[i]有负数，避免是负数时把答案算成0
    if (l == r)
    {
        c[x].id = l;     //c[x].id为节点x里最大值对应左端点
        return;
    }
    int mid = l + r >> 1;
    build(l, mid, c[x].l);
    build(mid + 1, r, c[x].r);
}

inline int _find(int x)  //查询x离散化后的值
{
    int l = 1, r = tot;
    while (l <= r)
    {
        int mid = l + r >> 1;
        if (h[mid] == x) return mid;
        if (x < h[mid]) r = mid - 1;
        else l = mid + 1;
    }
}

inline void pushdown(int l, int r, int x)
{
    if (!c[x].add) return;
    int x1, x2;  //注意标记下传要新建节点
    c[x1 = ++pool] = c[c[x].l];
    c[x2 = ++pool] = c[c[x].r];
    c[x1].val += c[x].add;
    c[x2].val += c[x].add;
    c[x1].add += c[x].add;
```

```
    c[x2].add += c[x].add;
    c[x].add = 0;
    c[x].l = x1;
    c[x].r = x2;
}

inline void collect(int x) //记录最大值及其编号
{
    if (!c[c[x].r].id || (c[c[x].l].id && c[c[x].l].val > c[c[x].r].val))
    {
        c[x].id = c[c[x].l].id;
        c[x].val = c[c[x].l].val;
    }
    else
    {
        c[x].id = c[c[x].r].id;
        c[x].val = c[c[x].r].val;
    }
}

inline void update(int l, int r,int s,int t,ll v,int y,int &x) //[s,t]+=v
{
    c[x = ++pool] = c[y];
    if (l == s && r == t)
    {
        c[x].add += v;
        c[x].val += v;
        if (!c[x].id) c[x].id = l;
        return;
    }
    pushdown(l, r, y); // 注意节点 x,y 都要标记下传
    pushdown(l, r, x);
    int mid = l + r >> 1;
    if (t <= mid) update(l, mid, s, t, v, c[y].l, c[x].l);
    else if (s > mid) update(mid + 1, r, s, t, v, c[y].r, c[x].r);
    else
    {
        update(l, mid, s, mid, v, c[y].l, c[x].l);
        update(mid + 1, r, mid + 1, t, v, c[y].r, c[x].r);
    }
    collect(x);
}

inline work ckmax(work a, work b)
```

```
{
    if (a.val > b.val) return a;
    return b;
}

inline work query(int l, int r, int s, int t, int x)
{  //固定右端点，查找左端点在[s,t]中的最大子串和
    if (l == s && r == t) return (work){c[x].val, c[x].id};
    int mid = l + r >> 1;
    pushdown(l, r, x);
    if (t <= mid) return query(l, mid, s, t, c[x].l);
    else if (s > mid) return query(mid + 1, r, s, t, c[x].r);
    return ckmax(query(l,mid,s,mid,c[x].l), query(mid+1,r,mid+1,t,
    c[x].r));
}

int main()
{
    freopen("seq.in", "r", stdin);
    freopen("seq.out", "w", stdout);
    int i, j;
    read(n); read(k);
    for (i = 1; i <= n; i++)
    {
        read(b[i]);
        a[i] = b[i];
    }
    sort(a + 1, a + n + 1);
    for (i = 1; i <= n; i++)
    if (i == 1 || a[i] != a[i - 1]) h[++tot] = a[i];
    for (i = 1; i <= n; i++) id[i] = _find(b[i]); //离散化
    for (i = 1; i <= n; i++)
    {
        lst[i] = pos[id[i]];
        pos[id[i]] = i;
    }
    build(1, n, rt[0]);
    for (i = 1; i <= n; i++)
    {
        int x = lst[i] + 1; //上一次 b[i]出现在 lst[i]
        update(1, n, i, i, inf, rt[i-1], rt[i]); //和-inf 抵消
        update(1, n, x, i, b[i], rt[i], rt[i]); //[x,i]+=b[i]
    }
    for (i = 1; i <= n; i++)
```

```
    {
        work u=query(1,n,1,i,rt[i]); //求右端点为i，左端点在[1,i]最大子串和
        q.push((point){1, i, u.id, i, u.val});
        //将左端点范围，左、右端点编号，子串和都放入堆中
    }
    while (k--)
    {
        point u = q.top(); //取出最优值
        q.pop();
        if (k == 0)
        {
            cout << u.sum << endl;
            fclose(stdin);
            fclose(stdout);
            return 0;
        }
        if (u.x < u.id) //去掉最优值后，左端点分成两个区间
        {
            work v = query(1, n, u.x, u.id - 1, rt[u.r]);
            q.push((point){u.x, u.id - 1, v.id, u.r, v.val});
        }
        if (u.id < u.y)
        {
            work v = query(1, n, u.id + 1, u.y, rt[u.r]);
            q.push((point){u.id + 1, u.y, v.id, u.r, v.val});
        }
    }
}
```

662.交通运输（traffic，1s，256MB）

【参考程序】

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

template <class T>
inline void read(T &x)
{
    static char ch;
    static bool opt;
    while (!isdigit(ch = getchar()) && ch != '-');
    x = (opt = ch == '-') ? 0 : ch - '0';
    while (isdigit(ch = getchar()))
        x = x * 10 + ch - '0';
    if (opt) x = ~x + 1;
}
```

```
template <class T>
inline void putint(T x)
{
    static char buf[15], *tail = buf;
    if (!x) putchar('0');
    else
    {
        if (x < 0) putchar('-'), x = ~x + 1;
        for (; x; x /= 10) *++tail = x % 10 + '0';
        for (; tail != buf; --tail) putchar(*tail);
    }
}

template <class T>
inline bool relax(T &x, const T &y)
{
    return x < y ? x = y, true : false;
}

template <class T>
inline bool tense(T &x, const T &y)
{
    return x > y ? x = y, true : false;
}

#define trav(u) for (halfEdge *e = adj[u]; e; e = e->next)

const int MaxNV = 1e5 + 5;
const int MaxNE = MaxNV;
const int MaxS = MaxNV * 200;
const int INF = 0x3f3f3f3f;
int n;

namespace SEG
{
    int tot, lc[MaxS], rc[MaxS], sze[MaxS];

    inline void insert(int lst,int &x,int l, int r, int pos)
    {
        lc[x = ++tot] = lc[lst]; //将插入可持久化
          rc[x] = rc[lst];
          sze[x] = sze[lst] + 1;
        if (l == r) return;
```

```
        int mid = l + r >> 1;
        if (pos <= mid)
             insert(lc[lst], lc[x], l, mid, pos);
        else
             insert(rc[lst], rc[x], mid + 1, r, pos);
    }

    inline int merge(int x,int y,int l,int r)//可持久化线段树合并
    {
        if (!x || !y) return x + y;

        int res = ++tot;
        sze[res] = sze[x] + sze[y];
        if (l == r) return res;
        int mid = l + r >> 1;
        lc[res] = merge(lc[x], lc[y], l, mid);
        rc[res] = merge(rc[x], rc[y], mid + 1, r);
        return res;
    }

    inline int query_sum(int x, int l, int r, int u, int v)
    { //查询区间和
        if (!x) return 0;
        if (u > r) return 0;
        if (u <= l && r <= v)
            return sze[x];
        int mid = l + r >> 1, res = 0;
        if (u <= mid) res += query_sum(lc[x], l, mid, u, v);
        if (v > mid) res += query_sum(rc[x], mid + 1, r, u, v);
        return res;
    }
}

namespace BIT
{
    int c[MaxNV << 1];
    inline int sum(int x)//查询区间和
    {
        int res = 0;
        if (x > n) x = n;
        for (; x; x ^= x & -x)
            res += c[x];
        return res;
    }
```

```
    inline void add(int x, int opt)//单点修改
    {
        for (; x <= n; x += x & -x)
            c[x] += opt;
    }
    inline int query(int l, int r)//区间查询
    {
        return sum(r) - sum(l - 1);
    }
}

using SEG::merge;
using SEG::insert;
using SEG::query_sum;

struct halfEdge
{
    int v;
    halfEdge *next;
}adj_pool[MaxNE], *adj_tail = adj_pool, *adj[MaxNV];

int fa[MaxNV], sze[MaxNV], ans[MaxNV];
int nrt[MaxNV], rt[MaxNV];

inline void addEdge(int u, int v)
{
    adj_tail->v = v;
    adj_tail->next = adj[u];
    adj[u] = adj_tail++;
}

inline void dfs_init(int u)
{
    sze[u] = 1;
    trav(u)
    {
        dfs_init(e->v);
        sze[u] += sze[e->v];
        nrt[u] = merge(rt[e->v], nrt[u], 1, n); //nrt 为不包含根的子树信息
    }
    insert(nrt[u], rt[u], 1, n, sze[u]); //rt 为包含根的子树信息
}

int maxsze, minsze, posa, posb;
```

```
inline bool check(int x, int mid)
{
    int l = maxsze - mid;
    int r = std::min(n, mid - minsze); //计算在最大连通块选出的子树大小范围
    if (l <= 0) return true;
    if (l > r) return false;
    if (posa == fa[x]) //处理最大连通块为子树外的情况
        return query_sum(nrt[1], 1, n, l, r)-query_sum(rt[x], 1, n, l, r)
             - BIT::query(l, r) + BIT::query(l + sze[x], r + sze[x]) > 0;
    else //处理最大连通块为某儿子的子树的情况
        return query_sum(nrt[posa], 1, n, l, r) > 0;
}

inline void dfs_ans(int u)
{
    int maxv = 0;
    posa = 0, posb = 0, maxsze = 0, minsze = INF;
    trav(u)
    {
        if (relax(maxsze, sze[e->v])) posa = e->v;
        if (tense(minsze, sze[e->v])) posb = e->v;
    }
    if (relax(maxsze, n - sze[u])) posa = fa[u];
    if (u != 1 && tense(minsze, n - sze[u])) posb = fa[u];

    trav(u) if (e->v != posa) relax(maxv, sze[e->v]);
    if (fa[u] != posa) relax(maxv, n - sze[u]);

    int l = maxv, r = maxsze;
    while (l <= r)
    {
        int mid = l + r >> 1;
        if (check(u, mid)) r = mid - 1, ans[u] = mid;
        else l = mid + 1;
    }
    BIT::add(sze[u], 1); //维护 1->fa(u)的 sze 信息
    trav(u) dfs_ans(e->v);
    BIT::add(sze[u], -1);//回溯的时候删去
}

int main()
{
    freopen("traffic.in", "r", stdin);
```

```
    freopen("traffic.out", "w", stdout);
    read(n);
    for (int i = 2; i <= n; ++i)
    {
        read(fa[i]);
        addEdge(fa[i], i);
    }

    dfs_init(1);
    dfs_ans(1);
    for (int i = 1; i <= n; ++i)
        putint(ans[i]), putchar('\n');
    return 0;
}
```

663.回忆之树(memory,1s,256MB)

【参考程序】

```
#include <bits/stdc++.h>
#define For(i, a, b) for (int i = a, bb = b; i <= bb; ++i)
#define Rof(i, a, b) for (int i = a, bb = b; i >= bb; --i)
#define Swap(x, y) (x ^= y ^= x ^= y)

const int MaxN = 1e5 + 5, MaxL = 3e5 + 5;
int adj[MaxN], nxt[MaxN << 1], go[MaxN << 1], tot;
int f[MaxN][17], Nxt[MaxL], rev[MaxN], dep[MaxN], ans[MaxN];
int rt[MaxL], st[MaxL], ed[MaxL], tim, num_T;
int n, m, x, y, num = 1;
char cap[MaxN << 1], s[MaxN], t[MaxL], c;

std :: queue <int> Q;
std :: vector <int> son[MaxL];
struct point
{
    int id, pos, opt;

    point (){}
    point (int _id, int _pos, int _opt):
        id(_id), pos(_pos), opt(_opt) {}
} ;

std :: vector <point> d[MaxN];
struct trie
{
    int g[26], fail, cnt;

    inline void clear()
    {
        For(i, 0, 25) g[i] = 0;
        fail = cnt = 0;
        return;
    }
} tr[MaxL];
```

```
struct tree
{
    int sum, lc, rc;

    #define s(x) T[x].sum
    #define lc(x) T[x].lc
    #define rc(x) T[x].rc

    inline void clear()
    {
        sum = lc = rc = 0;
        return;
    }
} T[MaxL * 50];

template <class T>
inline void get(T &res)
{
    char ch;
    bool bo = false;
    while ((ch = getchar()) < '0' || ch > '9')
        if (ch == '-') bo = true;
    res = ch - '0';
    while ((ch = getchar()) >= '0' && ch <= '9')
        res = (res << 1) + (res << 3) + ch - '0';
    if (bo) res = ~ res + 1;
    return;
}

template <class T>
inline void _put(T x)
{
    if (x > 9) _put(x / 10);
    putchar(x % 10 + '0');
    return;
}

template <class T>
inline void put(T x)
{
    if (x < 0)
    {
        putchar('-');
        x = ~ x + 1;
    }
    return _put(x);
}

inline void add(int x, int y, char c)
{
    nxt[++tot] = adj[x];
    adj[x] = tot;
    go[tot] = y;
    cap[tot] = c;
    return;
}

inline void dfs(int u) //遍历原树，建立父子关系
```

```
{
    dep[u] = dep[f[u][0]] + 1;
    For(i, 0, 15) f[u][i + 1] = f[f[u][i]][i];
    int v;
    for (int e = adj[u]; e; e = nxt[e])
    if ((v = go[e]) != f[u][0])
    {
        f[v][0] = u;
        rev[v] = cap[e];
        dfs(v);
    }
    return;
}

inline int lca(int x, int y) //求原树上点 x,y 的 lca
{
    if (x == y) return x;
    if (dep[x] < dep[y]) Swap(x, y);
    Rof(i, 16, 0)
    if (dep[f[x][i]] >= dep[y])
    {
        x = f[x][i];
        if (x == y) return x;
    }
    Rof(i, 16, 0)
    if (f[x][i] != f[y][i])
    {
        x = f[x][i];
        y = f[y][i];
    }
    return f[x][0];
}

inline int build(int len, char *s) //将字符串 s 插入 AC 自动机
{
    int x = 1, opt;
    For(i, 1, len)
    {
        opt = s[i] - 'a';
        if (!tr[x].g[opt]) tr[tr[x].g[opt] = ++num].clear();
        x = tr[x].g[opt];
    }
    return x;
}

inline void change(int len, char *s) //将字符串 s 反向
{
    For(i, 1, len / 2)
    {
        char c = s[i];
        s[i] = s[len - i + 1];
        s[len - i + 1] = c;
    }
    return;
}

inline int kmp(int len_t, int len_s)
{  //kmp 求题目给定字符串在拐弯处出现的次数
```

```
    if (len_t > len_s) return 0;
    Nxt[1] = 0;
    int j = 0;
    For(i, 2, len_t)
    {
        while (j && t[j + 1] != t[i]) j = Nxt[j];
        if (t[j + 1] == t[i]) j++;
        Nxt[i] = j;
    }
    j = 0;
    int res = 0;
    For(i, 1, len_s)
    {
        while (j && t[j + 1] != s[i]) j = Nxt[j];
        if (t[j + 1] == s[i]) j++;
        if (j == len_t)
        {
            res++;
            j = Nxt[j];
        }
    }
    return res;
}

inline void init(int x, int y, int id)
{ //将第 id 个询问的字符串插入 AC 自动机
    int len_t = strlen(t + 1);
    if (len_t >= n) return;
    int z = lca(x, y), pos; //字符串 t 在[z,y]中出现的次数可拆成 t 在 rt 到 y、
rt 到 z 的路径上的出现次数之差
    pos = build(len_t, t);
    d[y].push_back(point(id, pos, 1)); //将询问挂在母串的末尾
    Rof(i, 16, 0) //注意留出拐弯处的字符串长度
        if (dep[f[y][i]] - dep[z] + 1 >= len_t) y = f[y][i];
    d[y].push_back(point(id, pos, -1));
    change(len_t, t); //反串如上处理
    pos = build(len_t, t);
    d[x].push_back(point(id, pos, 1));
    Rof(i, 16, 0)
        if (dep[f[x][i]] - dep[z] + 1 >= len_t) x = f[x][i];
    d[x].push_back(point(id, pos, -1));
    int len_s = dep[x] + dep[y] - dep[z] * 2; //将拐弯处的字符串提取出来用
kmp 暴力求解
    memset(s + 1, 0, sizeof(s + 1));
    For(i, 1, dep[x] - dep[z])
    {
        s[len_s - i + 1] = rev[x];
        x = f[x][0];
    }
    For(i, 1, dep[y] - dep[z])
    {
        s[i] = rev[y];
        y = f[y][0];
    }
    ans[id] += kmp(len_t, len_s);
    return;
}
```

```
inline void dfs_fail(int u) //求 fail 树的 dfs 序
{
    st[u] = ++tim;
    For(i, 0, son[u].size() - 1)
        dfs_fail(son[u][i]);
    ed[u] = tim;
    return;
}

inline void build_fail() //建立 fail 树
{
    For(i, 0, 25) tr[0].g[i] = 1;
    int u, v;
    Q.push(1);
    while (!Q.empty())
    {
        u = Q.front();
        Q.pop();
        For(i, 0, 25)
        {
            v = tr[u].g[i];
            if (!v) tr[u].g[i] = tr[tr[u].fail].g[i];
            else
            {
                tr[v].fail = tr[tr[u].fail].g[i];
                if (tr[v].fail) son[tr[v].fail].push_back(v);
                Q.push(v);
            }
        }
    }
    dfs_fail(1);
    return;
}

inline void modify(int &x, int y, int l, int r, int pos)
{
    T[x = ++num_T] = T[y];
    if (l == r)
    {
        s(x)++;
        return;
    }
    int mid = l + r >> 1;
    if (pos <= mid) modify(lc(x), lc(y), l, mid, pos);
    else modify(rc(x), rc(y), mid + 1, r, pos);
    s(x) = s(lc(x)) + s(rc(x));
    return;
}

inline int query(int x, int l, int r, int p, int q)
{
    if (!x) return 0;
    if (p <= l && r <= q) return s(x);
    int mid = l + r >> 1, res = 0;
    if (p <= mid) res += query(lc(x), l, mid, p, q);
    if (q > mid) res += query(rc(x), mid + 1, r, p, q);
    return res;
}
```

```
inline void dfs(int u,int dad,int now)//遍历原树，同时在 AC 自动机上走
{
    modify(rt[u], rt[dad], 1, num, st[now]);
    For(i, 0, d[u].size() - 1) //对于挂在 u 上的询问，它在 fail 树上对应点的子
树和即为答案
       ans[d[u][i].id] += d[u][i].opt*query(rt[u],1,num, st[d[u][i].pos],
ed[d[u][i].pos]);
    for (int e = adj[u]; e; e = nxt[e])
    if (go[e] != f[u][0]) dfs(go[e], u, tr[now].g[cap[e] - 'a']);
    return;
}

int main()
{
    freopen("memory.in", "r", stdin);
    freopen("memory.out", "w", stdout);

    get(n), get(m);
    For(i, 1, n - 1)
    {
        get(x), get(y);
        while ((c = getchar()) < 'a' || c > 'z');
        add(x, y, c);
        add(y, x, c);
    }
    dfs(1);
    tr[0].clear(), tr[1].clear();
    For(i, 1, m)
    {
        int u, v;
        get(u), get(v);
        memset(t + 1, 0, sizeof(t + 1));
        scanf("%s", t + 1);
        init(u, v, i);
    }
    build_fail();
    T[0].clear();
    dfs(1, 0, 1);
    For(i, 1, m)
    {
        put(ans[i]);
        putchar('\n');
    }

    fclose(stdin), fclose(stdout);
    return 0;
}
```

第 5 章　树套树

673.序列修改(sequence,5s,512MB)

【参考程序】

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

template <class T>
inline void read(T &res)
{
    char ch;
    while (ch = getchar(), !isdigit(ch));
    res = ch ^ 48;
    while (ch = getchar(), isdigit(ch))
        res = res * 10 + ch - 48;
}

template <class T>
inline void put(T x)
{
    if (x > 9) put(x / 10);
    putchar(x % 10 + 48);
}

using std::map;
using std::set;
const int N = 2e5 + 5;
const int M = 1e7 + 5;
int top, T, E, n, m, am;
int stk[M], rt[N];
map<int, int> ma;

struct point
{
    int l, r, v;

    point() {}
    point(int L, int R, int V):
        l(L), r(R), v(V) {}

    inline bool operator < (const point &a) const
    {
        return l < a.l;
    }
```

```
    inline bool operator == (const point &a) const
    {
        return l == a.l && r == a.r;
    }
}a[N];

typedef set<point>::iterator it1;
typedef set<int>::iterator it2;
set<point> s;
set<int> p[N];

struct node
{
    int lc, rc, cnt;

    #define lc(x) tr[x].lc
    #define rc(x) tr[x].rc
    #define cnt(x) tr[x].cnt

    inline void Clear()
    {
        lc = rc = cnt = 0;
    }
}tr[M];

inline int new_node()
{
    return top ? stk[top--] : ++T;
}

inline void Modify(int &x, int l, int r, int v)
{
    if (!x)
        tr[x = new_node()].Clear();
    ++cnt(x);

    if (l == r)
        return ;
    int mid = l + r >> 1;
    v <= mid ? Modify(lc(x), l, mid, v) : Modify(rc(x), mid + 1, r, v);
}

inline int Query(int x, int l, int r, int v)
{
```

```
    if (!x)
        return 0;
    if (l == r)
        return cnt(x);
    int mid = l + r >> 1;
    return v <= mid ? Query(lc(x), l, mid, v) : cnt(lc(x)) + Query(rc(x),
mid + 1, r, v);
}

inline void Delete(int &x, int l, int r, int v)
{
    --cnt(x);
    if (l == r)
        return ;
    int mid = l + r >> 1;
    v <= mid ? Delete(lc(x), l, mid, v) : Delete(rc(x), mid + 1, r, v);

    if (!cnt(x))
    { //空节点直接删除，节省空间
        stk[++top] = x;
        tr[x].Clear();
        x = 0;
    }
}

inline void bitModify(int x, int v)
{
    for (int i = x; i <= n; i += i & -i)
        Modify(rt[i], 0, n, v);
}

inline void bitDelete(int x, int v)
{
    for (int i = x; i <= n; i += i & -i)
        Delete(rt[i], 0, n, v);
}

inline int bitQuery(int x, int v)
{
    int res = 0;
    for (int i = x; i; i ^= i & -i)
        res += Query(rt[i], 0, n, v);
    return res;
}
```

```
inline void Insert(int x, int v)
{ //处理在位置 x 插入数字 v 产生的影响
    if (!ma[v])
        ma[v] = ++E;

    int u = ma[v], lst;
    it2 tmp = p[u].lower_bound(x);
    if (tmp == p[u].begin())
        bitModify(x, lst = 0);
    else
    {
        --tmp;
        bitModify(x, lst = *tmp);
        ++tmp;
    }

    if (tmp != p[u].end())
    {
        bitDelete(*tmp, lst);
        bitModify(*tmp, x);
    }
    p[u].insert(x);
}

inline void Remove(int x, int v)
{ //消除原来位置 x 上数字 v 产生的影响
    int u = ma[v], lst;
    it2 tmp = p[u].lower_bound(x);
    if (tmp == p[u].begin())
        bitDelete(x, lst = 0);
    else
    {
        --tmp;
        bitDelete(x, lst = *tmp);
        ++tmp;
    }
    ++tmp;
    if (tmp != p[u].end())
    {
        bitDelete(*tmp, x);
        bitModify(*tmp, lst);
    }
    p[u].erase(x);
```

```
}

inline void solve(int l, int r)
{ //把跨过区间的段拆开，使得区间内都是完整的段
    it1 pre, suf;
    pre = s.lower_bound(point(l, l, 0));
    --pre;

    if (pre->r >= l)
    {
        point tmp = *pre;
        s.erase(pre);
        s.insert(point(tmp.l, l - 1, tmp.v));
        s.insert(point(l, tmp.r, tmp.v));
        Insert(l, tmp.v);
    }

    suf = s.lower_bound(point(r + 1, r + 1, 0));
    --suf;
    if (suf->r > r)
    {
        point tmp = *suf;
        s.erase(tmp);
        s.insert(point(tmp.l, r, tmp.v));
        s.insert(point(r + 1, tmp.r, tmp.v));
        Insert(r + 1, tmp.v);
    }
}

int main()
{
    freopen("sequence.in", "r", stdin);
    freopen("sequence.out", "w", stdout);

    read(n); read(m);
    s.insert(point(0, 0, 0));
    s.insert(point(n + 1, n + 1, 0));
    for (int i = 1, v; i <= n; ++i)
    {
        read(v);
        s.insert(point(i, i, v));
        if (!ma[v])
            ma[v] = ++E;
```

```
        int u = ma[v], lst = 0;
        if (p[u].empty())
            bitModify(i, 0);
        else
        {
            it2 tmp = p[u].end();
            --tmp;
            bitModify(i, *tmp);
        }
        p[u].insert(i);
    }
    for (int i = 1, t, l, r, x; i <= m; ++i)
    {
        read(t); read(l); read(r);
        if (t == 1)
        {
            read(x);
            solve(l, r);

            it1 now = s.lower_bound(point(l, l, 0));
            am = 0;
            for (; now->r <= r; ++now) //依次把区间内的段提取出来
                Remove(now->l, now->v), a[++am] = *now;
            for (int i = 1; i <= am; ++i)
                s.erase(a[i]);
            s.insert(point(l, r, x));
            Insert(l, x);
        }
        else
        {
            solve(l, r);
           put(bitQuery(r, l - 1) - bitQuery(l - 1, l - 1)), putchar('\n');
        }
    }

    fclose(stdin); fclose(stdout);
    return 0;
}
```

第五部分　树上问题

第 1 章　树链剖分

702.字符串值(string,2s,512MB)

【参考程序】

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

template <class T>
inline void read(T &res)
{
    char ch; bool flag = false; res = 0;
    while (ch = getchar(), !isdigit(ch) && ch != '-');
    ch == '-' ? flag = true : res = ch ^ 48;
    while (ch = getchar(), isdigit(ch))
        res = res * 10 + ch - 48;
    flag ? res = -res : 0;
}

template <class T>
inline void put(T x)
{
    if (x > 9) put(x / 10);
    putchar(x % 10 + 48);
}

const int N = 4e5 + 5;
const int L = 16e5 + 5;
const int mod = 1e9 + 7;
int tpos[N], sze[N], son[N], pos[N], idx[N], top[N], fa[N], dep[N];
int sum[L], tag[L], ans[L], fans[N];
int n, T, now, E; char s[N];

inline void add(int &x, int y)
{
    x += y;
    x >= mod ? x -= mod : 0;
}

struct Edge
{
    int to; Edge *nxt;
```

```
}p[N], *lst[N], *P = p;

inline void Link(int x, int y)
{
    (++P)->nxt = lst[x]; lst[x] = P; P->to = y;
}

struct SAM
{
    int ch[26], par, maxl;

    #define par(x) tr[x].par
    #define mx(x) tr[x].maxl

    inline void Clear()
    {
        memset(ch, 0, sizeof(ch));
        par = maxl = 0;
    }
}tr[N];

inline void insertSAM(char ch, int id)
{
    int c = ch - 'a', i = now, j, p;
    tr[now = ++T].Clear();
    tpos[id] = now;
    mx(now) = mx(i) + 1;
    for (; i && !tr[i].ch[c]; i = par(i))
        tr[i].ch[c] = now;

    if (!i)
        par(now) = 1;
    else
    {
        j = tr[i].ch[c];
        if (mx(i) + 1 == mx(j))
            par(now) = j;
        else
        {
            tr[p = ++T] = tr[j];
            mx(p) = mx(i) + 1;
            par(now) = par(j) = p;
            for (; i && tr[i].ch[c] == j; i = par(i))
                tr[i].ch[c] = p;
```

```
        }
    }
}

inline void Dfs1(int x)
{
    dep[x] = dep[fa[x]] + 1;
    sze[x] = 1;
    for (Edge *e = lst[x]; e; e = e->nxt)
    {
        int y = e->to;
        if (y == fa[x])
            continue;
        fa[y] = x;
        Dfs1(y);
        sze[x] += sze[y];
        sze[y] > sze[son[x]] ? son[x] = y : 0;
    }
}

inline void Dfs2(int x)
{
    if (son[x])
    {
        pos[son[x]] = ++E;
        idx[E] = son[x];
        top[son[x]] = top[x];
        Dfs2(son[x]);
    }

    int y;
    for (Edge *e = lst[x]; e; e = e->nxt)
        if (!top[y = e->to])
        {
            pos[y] = ++E;
            idx[E] = y;
            top[y] = y;
            Dfs2(y);
        }
}

inline void Init()
{
    Dfs1(1);
```

```
    pos[1] = top[1] = idx[1] = E = 1;
    Dfs2(1);
}

#define sL s << 1
#define sR s << 1 | 1

inline void addTag(int s, int v)
{
    add(tag[s], v);
    ans[s] = (ans[s] + 1ll * sum[s] * v) % mod;
}

inline void Update(int s)
{
    ans[s] = ans[sL];
    add(ans[s], ans[sR]);
}

inline void downDate(int s)
{
    if (tag[s])
    {
        addTag(sL, tag[s]);
        addTag(sR, tag[s]);
        tag[s] = 0;
    }
}

inline void Build(int s, int l, int r)
{
    if (l == r)
        return (void)(sum[s] = mx(idx[l]) - mx(par(idx[l])));
    int mid = l + r >> 1;
    Build(sL, l, mid); Build(sR, mid + 1, r);
    sum[s] = sum[sL];
    add(sum[s], sum[sR]);
}

inline int Query(int s, int l, int r, int x, int y)
{
    if (l == x && r == y)
        return ans[s];
    downDate(s);
```

```
    int mid = l + r >> 1;
    if (y <= mid)
        return Query(sL, l, mid, x, y);
    else if (x > mid)
        return Query(sR, mid + 1, r, x, y);
    else
    {
        int res = Query(sL, l, mid, x, mid);
        add(res, Query(sR, mid + 1, r, mid + 1, y));
        return res;
    }
}

inline void Modify(int s, int l, int r, int x, int y)
{
    if (l == x && r == y)
        return addTag(s, 1);
    downDate(s);

    int mid = l + r >> 1;
    if (y <= mid)
        Modify(sL, l, mid, x, y);
    else if (x > mid)
        Modify(sR, mid + 1, r, x, y);
    else
    {
        Modify(sL, l, mid, x, mid);
        Modify(sR, mid + 1, r, mid + 1, y);
    }
    Update(s);
}

inline void pathModify(int x)
{ //维护 x 到根路径上的 right 集合大小
    while (top[x] != 1)
        Modify(1, 1, E, pos[top[x]], pos[x]), x = fa[top[x]];
    Modify(1, 1, E, 1, pos[x]);
}

inline int pathQuery(int x)
{ //计算后缀产生的贡献
    int res = 0;
    while (top[x] != 1)
```

```
        add(res, Query(1, 1, E, pos[top[x]], pos[x])), x = fa[top[x]];
    add(res, Query(1, 1, E, 1, pos[x]));
    return res;
}

int main()
{
    freopen("string.in", "r", stdin);
    freopen("string.out", "w", stdout);

    scanf("%d%s", &n, s + 1);
    tr[now = T = 1].Clear();
    for (int i = 1; i <= n; ++i)
        insertSAM(s[i], i);

    for (int i = 2; i <= T; ++i)
        Link(par(i), i);
    Init();
    Build(1, 1, E);

    for (int i = 1; i <= n; ++i)
    {
        fans[i] = fans[i - 1];
        add(fans[i], pathQuery(tpos[i]));
        pathModify(tpos[i]);
    }
    for (int i = 1; i <= n; ++i)
        add(fans[i], fans[i - 1]);
    for (int i = 1; i <= n; ++i)
        put(fans[i]), putchar('\n');

    fclose(stdin); fclose(stdout);
    return 0;
}
```

703.仙人掌题(cac,2s,512MB)

【参考程序】

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

template <class T>
inline void read(T &res)
{
    char ch; bool flag = false; res = 0;
    while (ch = getchar(), !isdigit(ch) &&ch != '-');
```

```
    ch == '-' ? flag = true : res = ch ^ 48;
    while (ch = getchar(), isdigit(ch))
        res = res * 10 + ch - 48;
    flag ? res = -res : 0;
}

template <class T>
inline void put(T x)
{
    if (x > 9) put(x / 10);
    putchar(x % 10 + 48);
}

using std::vector;
const int mod = 998244353;
const int N = 1e6 + 5;
const int M = 4e6 + 5;
int dfn[N], low[N], stk[N];
int dep[N], pos[N], fa[N], son[N];
int sze[N], top[N], idx[N];
int val[N], sum1[M], sum2[M], tag1[M], tag2[M];
int n, m, Q, E, C, tis, top_s; bool col[M];
vector<int> v[N];

template <class T>
inline void CkMin(T &x, T y) {x > y ? x = y : 0;}

struct Edge
{
    int to; Edge *nxt;
}p[M], *lst[N], *P = p;

inline void Link(int x, int y)
{
    (++P)->nxt = lst[x]; lst[x] = P; P->to = y;
    (++P)->nxt = lst[y]; lst[y] = P; P->to = x;
}

inline void Tarjan(int x, int Fa)
{
    dfn[x] = low[x] = ++tis;
    stk[++top_s] = x; int y, w;
    for (int i = 0, im = v[x].size(); i<im; ++i)
    {
```

```
        y = v[x][i];
        if (y == Fa)
            continue;
        if (!dfn[y])
        {
            Tarjan(y, x);
            CkMin(low[x], low[y]);

            if (dfn[x] <= low[y])
            {
                ++C;
                Link(x, C);
                while (w = stk[top_s], stk[top_s + 1] != y)
                    Link(w, C), --top_s;
            }
        }
        else
            CkMin(low[x], dfn[y]);
    }
}

#define sL s << 1
#define sR s << 1 | 1

inline void add(int &x, int y)
{
    x += y;
    x >= mod ? x -= mod : 0;
}

inline void addTag1(int s, int v)
{
    if (col[s]) add(sum1[s], v);
    add(tag1[s], v);
}

inline void addTag2(int s, int v)
{
    if (!col[s]) add(sum2[s], v);
    add(tag2[s], v);
}

inline void downDate(int s)
{
```

```
    if (tag1[s])
    {
        addTag1(sL, tag1[s]);
        addTag1(sR, tag1[s]);
        tag1[s] = 0;
    }
    if (tag2[s])
    {
        addTag2(sL, tag2[s]);
        addTag2(sR, tag2[s]);
        tag2[s] = 0;
    }
}

inline void Build(int s, int l, int r)
{
    if (l == r)
        return (void)(col[s] = idx[l] > n);
    int mid = l + r >> 1;
    Build(sL, l, mid); Build(sR, mid + 1, r);
}

inline void Modify(int s, int l, int r, int x, int y, int v)
{
    if (l == x && r == y)
        return addTag1(s, v), addTag2(s, mod - v);
    downDate(s);

    int mid = l + r >> 1;
    if (y <= mid)
        Modify(sL, l, mid, x, y, v);
    else if (x > mid)
        Modify(sR, mid + 1, r, x, y, v);
    else
    {
        Modify(sL, l, mid, x, mid, v);
        Modify(sR, mid + 1, r, mid + 1, y, v);
    }
}

inline void Modify2(int s, int l, int r, int x, int v)
{
    if (l == r)
        return add(sum2[s], v);
```

```
    downDate(s);

    int mid = l + r >> 1;
    x <= mid ? Modify2(sL, l, mid, x, v) : Modify2(sR, mid + 1, r, x, v);
}

inline int Query1(int s, int l, int r, int x)
{
    if (l == r)
        return sum1[s];
    downDate(s);

    int mid = l + r >> 1;
    return x <= mid ? Query1(sL, l, mid, x) : Query1(sR, mid + 1, r, x);
}

inline int Query2(int s, int l, int r, int x)
{
    if (l == r)
        return sum2[s];
    downDate(s);

    int mid = l + r >> 1;
    return x <= mid ? Query2(sL, l, mid, x) : Query2(sR, mid + 1, r, x);
}

inline void Dfs1(int x)
{
    sze[x] = 1;
    dep[x] = dep[fa[x]] + 1;
    for (Edge *e = lst[x]; e; e = e->nxt)
    {
        int y = e->to;
        if (y == fa[x])
            continue;
        fa[y] = x;
        Dfs1(y);
        sze[x] += sze[y];
        sze[y] >sze[son[x]] ? son[x] = y : 0;
    }
}

inline void Dfs2(int x)
{
```

```
    if (son[x])
    {
        pos[son[x]] = ++E;
        idx[E] = son[x];
        top[son[x]] = top[x];
        Dfs2(son[x]);
    }

    int y;
    for (Edge *e = lst[x]; e; e = e->nxt)
        if (!top[y = e->to])
        {
            pos[y] = ++E;
            idx[E] = y;
            top[y] = y;
            Dfs2(y);
        }
}

inline void Init()
{
    Dfs1(1);
    idx[1] = top[1] = pos[1] = E = 1;
    Dfs2(1);
}

inline void pathModify(int x, int y, int v)
{
    while (top[x] != top[y])
    {
        if (dep[top[x]] < dep[top[y]])
            std::swap(x, y);
        if (top[x] > n)
            add(val[fa[top[x]]], v);
        Modify(1, 1, C, pos[top[x]], pos[x], v);
        x = fa[top[x]];
    }
    if (dep[x] < dep[y])
        std::swap(x, y);
    if (y == top[y] && top[y] > n && fa[top[y]])
        add(val[fa[top[y]]], v);
    Modify(1, 1, C, pos[y], pos[x], v);
}
```

```
int main()
{
    freopen("cac.in", "r", stdin);
    freopen("cac.out", "w", stdout);
    read(n); read(m); read(Q);
    for (int i = 1, x, y; i<= m; ++i)
    {
        read(x); read(y);
        v[x].push_back(y);
        v[y].push_back(x);
    }

    C = n;
    Tarjan(1, 0);
    Init();
    Build(1, 1, C);

    int k, x, y, v;
    for (int i = 1; i<= Q; ++i)
    {
        read(k);
        if (!k)
        {
            read(x); read(y); read(v);
            pathModify(x, y, v);
            Modify2(1, 1, C, pos[x], v);
            Modify2(1, 1, C, pos[y], v);
        }
        else
        {
            read(x);
            int res = val[x];
            add(res, Query2(1, 1, C, pos[x]));
            if (fa[x])
                add(res, Query1(1, 1, C, pos[fa[x]]));
            if (son[x])
                add(res, Query1(1, 1, C, pos[son[x]]));
            put(res), putchar('\n');
        }
    }
    fclose(stdin); fclose(stdout);
    return 0;
}
```

第 2 章　点分治

713.随机访问(eat,2s,256MB)

【参考程序】

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

const int N = 1e5 + 5, M = N << 1, L = 1e6 + 5, ZZQ = 998244353;

int n, m, ecnt = 1, nxt[M], adj[N], go[M], cnt[N], inv[N], sze[N], maxs[N],
         ff, tot, rev[L], a[L], b[L], gt[L], ans, dep[N], nm, fa[N],
cir[N], ft[N], iscir[M];
bool vis[N];

void dfs_init(int u, int fe, int d) {
    dep[u] = d;
    vis[u] = 1;
    for (int e = adj[u], v; e; e = nxt[e]) {
        if (e == fe) continue;
        if (!vis[v = go[e]]) {
            fa[v] = u, ft[v] = e, dfs_init(v, e ^ 1, d + 1);
        } else if (dep[v] < dep[u]) {
            cir[nm = 1] = u;
            for (int x = u; x != v; x = fa[x]) {
                cir[++nm] = fa[x];
                iscir[ft[x]] = iscir[ft[x] ^ 1] = 1;
            }
            iscir[e] = iscir[e ^ 1] = 2;
        }
    }
}

int qpow(int a, int b) {
    int res = 1;
    while (b) {
        if (b & 1) res = 1ll * res * a % ZZQ;
        a = 1ll * a * a % ZZQ;
        b >>= 1;
    }
    return res;
}

void init(int n, int m, int *a) {
    ff = 1;
```

```
    tot = 0;
    while (ff <= n) ff <<= 1, tot++;
    for (int i = 0; i < ff; i++)
        rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << tot - 1);
    for (int i = m + 1; i < ff; i++) a[i] = 0;
}

void FFT(int n, int *a, int op) {
    for (int i = 0; i < n; i++) if (i < rev[i]) std::swap(a[i], a[rev[i]]);
    gt[n] = qpow(3, (ZZQ - 1) / n * ((n + op) % n));
    for (int i = n >> 1; i; i >>= 1)
        gt[i] = 1ll * gt[i << 1] * gt[i << 1] % ZZQ;
    for (int k = 1; k < n; k <<= 1) {
        int x = gt[k << 1];
        for (int i = 0; i < n; i += k << 1) {
            int w = 1;
            for (int j = 0; j < k; j++) {
                int u = a[i + j], v = 1ll * w * a[i + j + k] % ZZQ;
                a[i + j] = (u + v) % ZZQ;
                a[i + j + k] = (u - v + ZZQ) % ZZQ;
                w = 1ll * w * x % ZZQ;
            }
        }
    }
}

void add_edge(int u, int v) {
    nxt[++ecnt] = adj[u];
    adj[u] = ecnt;
    go[ecnt] = v;
    nxt[++ecnt] = adj[v];
    adj[v] = ecnt;
    go[ecnt] = u;
}

void dfs1(int u, int fu) {
    sze[u] = 1;
    maxs[u] = 0;
    for (int e = adj[u], v; e; e = nxt[e])
        if (!vis[v = go[e]] && iscir[e] != 2 && v != fu)
            dfs1(v, u), sze[u] += sze[v], maxs[u] = std::max(maxs[u],
sze[v]);
}
```

```
void dfs2(int u, int fu, int r, int &G) {
    if (std::max(sze[r] - sze[u], maxs[u]) < std::max(sze[r] - sze[G],
maxs[G]))
        G = u;
    for (int e = adj[u], v; e; e = nxt[e])
        if (!vis[v = go[e]] && iscir[e] != 2 && v != fu)
            dfs2(v, u, r, G);
}

int calcG(int u) {
    int G = u;
    dfs1(u, 0);
    dfs2(u, 0, u, G);
    return G;
}

void dfs_d(int u, int fu, int dep, bool tr, int &maxdep, int *a) {
    if (dep > maxdep) maxdep = dep, a[dep] = 0;
    a[dep]++;
    for (int e = adj[u], v; e; e = nxt[e])
       if (!vis[v = go[e]] && iscir[e] != 2 && v != fu && (tr || !iscir[e]))
            dfs_d(v, u, dep + 1, tr, maxdep, a);
}

void polySqr(int *a, int len) { //将 a 和自己做一次卷积
    init(len << 1, len, a); //FFT 预处理
    FFT(ff, a, 1);
    for (int i = 0; i < ff; i++) a[i] = 1ll * a[i] * a[i] % ZZQ;
    FFT(ff, a, -1);
    int gg = qpow(ff, ZZQ - 2);
    for (int i = 0; i <= (len << 1); i++)
        a[i] = 1ll * a[i] * gg % ZZQ;
}

void solveA(int u) {
    u = calcG(u); //找重心
    vis[u] = 1;

    int maxdep = -1;
    dfs_d(u, 0, 0, 1, maxdep, a); //加入当前以重心为根的子树内的所有深度信息
    polySqr(a, maxdep);
    for (int i = 0; i <= (maxdep << 1); i++)
        cnt[i]=(a[i]+cnt[i])%ZZQ;//加上两点都在当前以重心为根的子树内的情况
    for (int e = adj[u], v; e; e = nxt[e])
```

```
        if (!vis[v = go[e]] && iscir[e] != 2) {
            a[maxdep = 0] = 0;
            dfs_d(v, u, 1, 1, maxdep, a);
            polySqr(a, maxdep);
            for (int i = 0; i <= (maxdep << 1); i++)
                cnt[i] = (cnt[i] - a[i] + ZZQ) % ZZQ;//减去两点来自同一子树
        }
    for (int e = adj[u], v; e; e = nxt[e])
        if (!vis[v = go[e]] && iscir[e] != 2) solveA(v);
}

void solveB(int l, int r) { //分治计算 dis2 的贡献
    if (l == r) return;
    int mid = l + r >> 1;
    solveB(l, mid), solveB(mid + 1, r);

    int ma = -1, mb = -1;
    for (int i = l; i <= mid; i++) dfs_d(cir[i], 0, i - l, 0, ma, a);
    for (int i = r; i >= mid + 1; i--) dfs_d(cir[i], 0, r - i, 0, mb, b);

    ff = 1, tot = 0;
    while (ff <= ma + mb) ff <<= 1, tot++;
    for (int i = 0; i < ff; i++)
        rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << tot - 1);
    for (int i = ma + 1; i < ff; i++) a[i] = 0;
    for (int i = mb + 1; i < ff; i++) b[i] = 0;
    FFT(ff, a, 1), FFT(ff, b, 1);
    for (int i = 0; i < ff; i++) a[i] = 1ll * a[i] * b[i] % ZZQ;
    FFT(ff, a, -1);

    int gg = qpow(ff, ZZQ - 2);
    for (int i = 0; i <= ma + mb; i++)
        cnt[i + l + nm - r] = (1ll * a[i] * gg + cnt[i + l + nm - r]) %
ZZQ;
}

int main() {
    int x, y;
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= m; i++) scanf("%d%d", &x, &y), add_edge(x, y);
    inv[1] = 1;
    for (int i = 2; i <= n; i++)
        inv[i] = 1ll * (ZZQ - ZZQ / i) * inv[ZZQ % i] % ZZQ;
```

```
    if (m == n) {
        dfs_init(1, 0, 0);
        memset(vis, 0, sizeof(vis));
    }

    solveA(1); //处理 dis 和 dis1
    for (int i = 0; i < n; i++)
        ans = (1ll * cnt[i] * inv[i + 1] + ans) % ZZQ;

    if (m == n) {
        memset(cnt, 0, sizeof(cnt)), memset(vis, 0, sizeof(vis));

        int maxdep = -1;
        for (int i = 1; i <= nm; i++)
            dfs_d(cir[i], 0, 0, 0, maxdep,a); //处理 dep[u]+dep[v]=i 的贡献
        polySqr(a, maxdep);
        for (int i = 0; i <= (maxdep << 1); i++)
            cnt[i] = (a[i] + cnt[i]) % ZZQ; //加上没有限制的方案数

        for (int i = 1; i <= nm; i++) {
            maxdep = -1;
            dfs_d(cir[i], 0, 0, 0, maxdep, a);
            polySqr(a, maxdep);
            for (int j = 0; j <= (maxdep << 1); j++)
                cnt[j] = (cnt[j] - a[j] + ZZQ) % ZZQ;//减去来自同一个子树的
        }
        for (int i = 0; i <= n - nm; i++)
            ans = (ans - 1ll * cnt[i] * inv[i + nm] % ZZQ + ZZQ) % ZZQ;
        memset(cnt, 0, sizeof(cnt));

        solveB(1, nm); //处理 dis2
        for (int i = 0; i < n; i++)
            ans = (2ll * cnt[i] * inv[i + 1] + ans) % ZZQ;
    }
    std::cout << ans << std::endl;
    return 0;
}
```

第 3 章　虚树

721.森林旅行(travel,2s,256MB)

【参考程序】

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef pair <ll, int> pli;
typedef pair <int, int> pii;
#define mp make_pair
inline int Get() {
    char ch;
    while ((ch = getchar()) < '0' || ch > '9');
    int Num = ch - '0';
    while ((ch = getchar()) >= '0' && ch <= '9')
        Num = (Num << 3) + (Num << 1) + ch - '0';
    return Num;
}
const int N = 5e5 + 5;
const ll inf = 1e11;
int n, m, Q, que[N], qn, fa[N][19], dep[N], sze[N], dfn[N];
int tot, first[N], nxt[N << 1], ed[N << 1], idn, query[15][2];
map <pii, int> id;
vector <int> vec[N];
void AddEdge(int x, int y) {
    nxt[++tot] = first[x], first[x] = tot, ed[tot] = y;
    nxt[++tot] = first[y], first[y] = tot, ed[tot] = x;
}
priority_queue <pli, vector <pli>, greater <pli>> Que;
struct Graph {
    int tot, first[N], nxt[N << 1], ed[N << 1], cap[N << 1];
    ll dist[N];
    void AddEdge(int x, int y, int w) {
        nxt[++tot] = first[x], first[x] = tot, ed[tot] = y, cap[tot] = w;
        nxt[++tot] = first[y], first[y] = tot, ed[tot] = x, cap[tot] = w;
    }
    ll query(int x, int y) {
        for (int i = 1; i <= idn; ++i) dist[i] = inf;
        dist[x] = 0, Que.push(mp(0, x));
        while (!Que.empty()) {
            pli p = Que.top(); Que.pop();
            if (dist[p.second] < p.first) continue;
            for (int k = first[p.second], v; v = ed[k], k; k = nxt[k])
                if (dist[v] > p.first + cap[k]) {
```

```
                dist[v] = p.first + cap[k];
                Que.push(mp(dist[v], v));
            }
        }

        return dist[y] == inf ? -1 : dist[y];
    }
}G;
inline int LCA(int x, int y) {
    if (dep[x] < dep[y]) swap(x, y);
    int delta = dep[x] - dep[y], k = 0;
    for (; delta; delta >>= 1, ++k) if (delta & 1) x = fa[x][k];

    if (x == y) return x;
    for (int k = 18; k >= 0; --k)
        if (fa[x][k] != fa[y][k]) x = fa[x][k], y = fa[y][k];
    return fa[x][0];
}
int aux[N], auxfa[N], sta[N], top;
inline bool cmp(const int &x, const int &y) {
    return dfn[x] < dfn[y];
}
void build(int now) {
    int sze = vec[now].size(), cnt = 0;
    if (!sze) return;
    for (int i = 0; i < sze; ++i) aux[++cnt] = vec[now][i];
    if (id.find(mp(now, 1)) == id.end()) aux[++cnt] = 1, ++sze;
    sort(aux + 1, aux + cnt + 1, cmp);

    sta[top = 0] = 0;
    for (int i = 1; i <= sze; ++i) {
        int u = aux[i];
        if (!top) auxfa[u] = 0, sta[++top] = u;
        else {
            int x = LCA(u, sta[top]);
            while (dep[x] < dep[sta[top]]) {
                if (dep[sta[top - 1]] <= dep[x]) auxfa[sta[top]] = x;
                --top;
            }
            if (sta[top] != x) {
                aux[++cnt] = x;
                auxfa[x] = sta[top];
                sta[++top] = x;
            }
```

```
            auxfa[u] = x, sta[++top] = u;
        }
    }

    for (int i = 1, u; u = aux[i], i <= cnt; ++i)
        if (id.find(mp(now, u)) == id.end())
            id[mp(now, u)] = ++idn;

    for (int i = 1, u; u = aux[i], i <= cnt; ++i)
        if (auxfa[u])
            G.AddEdge(id[mp(now, u)], id[mp(now, auxfa[u])], dep[u] -
dep[auxfa[u]]);
}
int main() {
    freopen("travel.in", "r", stdin);
    freopen("travel.out", "w", stdout);
    n = Get(), m = Get(), Q = Get();
    for (int i = 1; i < n; ++i) AddEdge(Get(), Get());

    que[qn = 1] = 1, dep[1] = 1;
    for (int ql = 1, u; u = que[ql], ql <= qn; ++ql, sze[u] = 1)
        for (int k = first[u], v; v = ed[k], k; k = nxt[k])
            if (v != fa[u][0])
                que[++qn] = v, fa[v][0] = u, dep[v] = dep[u] + 1;
    for (int qr = qn, u; u = que[qr], qr > 1; --qr) sze[fa[u][0]] +=
sze[u];
    dfn[1] = 1;
    for (int ql = 1, u; u = que[ql], ql <= qn; ++ql) {
        int cur = dfn[u];
        for (int k = first[u], v; v = ed[k], k; k = nxt[k])
            if (v != fa[u][0]) dfn[v] = cur + 1, cur += sze[v];
    }
    for (int k = 1; k <= 18; ++k)
        for (int i = 1; i <= n; ++i) fa[i][k] = fa[fa[i][k - 1]][k - 1];

    for (int i = 1; i <= m; ++i) {
        int x1 = Get(), y1 = Get(), z1;
        if (id.find(mp(y1, x1)) != id.end()) z1 = id[mp(y1, x1)];
        else id[mp(y1, x1)] = z1 = ++idn, vec[y1].push_back(x1);

        int x2 = Get(), y2 = Get(), z2;
        if (id.find(mp(y2, x2)) != id.end()) z2 = id[mp(y2, x2)];
        else id[mp(y2, x2)] = z2 = ++idn, vec[y2].push_back(x2);
```

```
        G.AddEdge(z1, z2, 1);
    }
    for (int i = 1; i <= Q; ++i) {
        int x1 = Get(), y1 = Get(), z1;
        if (id.find(mp(y1, x1)) != id.end()) z1 = id[mp(y1, x1)];
        else id[mp(y1, x1)] = z1 = ++idn, vec[y1].push_back(x1);

        int x2 = Get(), y2 = Get(), z2;
        if (id.find(mp(y2, x2)) != id.end()) z2 = id[mp(y2, x2)];
        else id[mp(y2, x2)] = z2 = ++idn, vec[y2].push_back(x2);
        query[i][0] = z1, query[i][1] = z2;
    }

    for (int i = 1; i <= 2e5; ++i) build(i);
    for (int i = 1; i <= Q; ++i) {
        ll ret = G.query(query[i][0], query[i][1]);
        if (ret < 0) puts("impossible");
        else printf("%lld\n", ret);
    }
    fclose(stdin); fclose(stdout);
    return 0;
}
```

712.无心行挽(do,2.5s,512MB)

【参考程序】

```
#include <bits/stdc++.h>
using namespace std;
template <class t>
inline void read(t & res)
{
    char ch;
    while (ch = getchar(), !isdigit(ch));
    res = ch ^ 48;
    while (ch = getchar(), isdigit(ch))
    res = res * 10 + (ch ^ 48);
}
const int e = 1e5 + 5, inf = 0x3f3f3f3f;
struct point
{
    int x, v;
}del[e];
multiset<int, greater<int>> d[e];
int n, m, stk[e << 1], top, anc[e][17], f[e][17], a[e], b[e], g[e][17],
fa[e], dfn[e];
int num, nxt[e * 2], go[e * 2], adj[e], dep[e], dis[e], ans, cnt, p[e <<
```

```
1], in[e], tim;
int logn[e << 1], op[e], st[e << 1][18], id[e << 1], dc, to[e];
bool vis[e];
inline void add(int x, int y)
{
    nxt[++num] = adj[x];
    adj[x] = num;
    go[num] = y;
    nxt[++num] = adj[y];
    adj[y] = num;
    go[num] = x;
}
inline void prepare()
{
    int i, j;
    logn[0] = -1;
    for (i = 1; i <= tim; i++) logn[i] = logn[i >> 1] + 1, st[i][0] =
p[i];
    for (j = 1; (1 << j) <= tim; j++)
    for (i = 1; i + (1 << j) - 1 <= tim; i++)
    {
        int u = st[i][j - 1], v = st[i + (1 << j - 1)][j - 1];
        if (dep[u] < dep[v]) st[i][j] = u;
        else st[i][j] = v;
    }
}
inline int lca(int l, int r) // rmq 求 lca
{
    l = in[l]; r = in[r];
    if (l > r) swap(l, r);
    int k = logn[r - l + 1], u = st[l][k], v = st[r - (1 << k) + 1][k];
    return dep[u] < dep[v] ? u : v;
}
inline void dfs1(int u, int pa)
{
    int i;
    dep[u] = dep[pa] + 1;
    dfn[u] = ++dfn[0];
    p[++tim] = u;
    in[u] = tim;
    for (i = 0; i < 16; i++) anc[u][i + 1] = anc[anc[u][i]][i];
    for (i = adj[u]; i; i = nxt[i])
    {
        int v = go[i];
```

```
        if (v == pa) continue;
        anc[v][0] = u;
        dfs1(v, u);
        if (a[v] + 1 > a[u]) b[u] = a[u], a[u] = a[v] + 1;
        else b[u] = max(b[u], a[v] + 1);
        d[u].insert(a[v] + 1);
        p[++tim] = u;
    }
}
inline void dfs2(int u, int pa) // 预处理 f,g 数组
{
    int i;
    for (i = 0; i < 16; i++)
    {
        f[u][i + 1] = max(f[u][i], f[anc[u][i]][i] + (1 << i));
        g[u][i + 1] = max(g[u][i] + (1 << i), g[anc[u][i]][i]);
    }
    for (i = adj[u]; i; i = nxt[i])
    {
        int v = go[i];
        if (v == pa) continue;
        int l = a[u];
        if (a[v] + 1 == a[u]) l = b[u];
        f[v][0] = l + 1;
        g[v][0] = l;
        dfs2(v, u);
    }
}
inline int jump(int x, int d) // 查询 x 向上走 d 步到达的点
{
    for (int i = 16; i >= 0; i--)
    if (d & (1 << i)) x = anc[x][i];
    return x;
}
inline int askf(int x, int d)
{  // 从 x 开始走，不能走向 x 子树内的点且不能走出 x 的 d 级祖先的子树，能到达的最远距离。
    int res = 0, sum = 0, lx = x;
    for (int i = 16; i >= 0; i--)
    if (d & (1 << i))
    {
        res = max(res, sum + f[x][i]);
        sum |= 1 << i;
        x = anc[x][i];
```

```
    }
    return res;
}
inline int askg(int x, int d)
{ // 从 x 的 d 级祖先开始走，不能走向 x 子树内的点且不能走出 x 的 d 级祖先的子树，能到达的最远距离。
    int res = 0, sum = d;
    for (int i = 16; i >= 0; i--)
    if (d & (1 << i))
    {
        sum ^= 1 << i;
        res = max(res, sum + g[x][i]);
        x = anc[x][i];
    }
    return res;
}
inline bool cmp(int x, int y)
{
    return dfn[x] < dfn[y];
}
inline void build() // 建立虚树
{
    id[++cnt] = 1;
    sort(id + 1, id + cnt + 1, cmp);
    int i, lc = cnt;
    for (i = 1; i < lc; i++) id[++cnt] = lca(id[i], id[i + 1]);
    sort(id + 1, id + cnt + 1, cmp);
    cnt = unique(id + 1, id + cnt + 1) - id - 1;
    top = 0;
    for (i = 1; i <= cnt; i++)
    {
        while (top && lca(stk[top], id[i]) != stk[top]) top--;
        if (top) fa[id[i]] = stk[top];
        else fa[id[i]] = 0;
        stk[++top] = id[i];
    }
}
inline void solve()
{
    int i;
    for (i = 1; i <= cnt; i++)
    if (vis[id[i]]) dis[id[i]] = 0, op[id[i]] = 2;
    else dis[id[i]] = inf, op[id[i]] = 0, to[id[i]] = 0;
    for (i = 1; i <= dc; i++) d[del[i].x].insert(del[i].v);
```

```
for (i = cnt; i >= 1; i--)
{
    int v = id[i], u = fa[v], l = dis[v] + dep[v] - dep[u];
    if (!u) continue;
    if (l < dis[u]) dis[u] = l, to[u] = v;
}
for (i = 1; i <= cnt; i++)
{
    int v = id[i], u = fa[v];
    if (!u) continue;
    int l = dis[u] + dep[v] - dep[u];
    if (l < dis[v]) op[v] = 1, dis[v] = l;
}
dc = 0;
for (i = 1; i <= cnt; i++)
{
    int v = id[i], u = fa[v];
    if (!u) continue;
    int x = jump(v, dep[v] - dep[u] - 1);
    d[u].erase(a[x] + 1); // 在 u 的 multiset 中删除 a[x]+1
    del[++dc] = (point){u, a[x] + 1};
}
for (i = 1; i <= cnt; i++)
{
    int v = id[i], u = fa[v];
    int res = dis[v], l = dep[v] - dep[u];
    if (d[v].size()) res += *d[v].begin();
    ans = max(ans, res); // 第一类点
    if (!u) continue;
    if ((op[u] != 0 || (op[u] == 0 && to[u] != v)) && op[v] != 1)
    {   // 第二类点，pu != pv
        int w = (dis[u] - dis[v] + l) / 2, x = jump(v, w);
        if (w >= 1)
        {
            int w2 = w;
            if (w2 >= l) w2 = l - 1;
            ans = max(ans, dis[v] + askf(v, w2));
        }
        if (l - w - 1 >= 1)
        {
            int w2 = l - w - 1;
            if (w2 >= l) w2 = l - 1;
            ans = max(ans, dis[u] + askg(x, w2) + 1);
        }
```

```
        }
        else if (op[u] == 0 && to[u] == v) // pu,pv 都在 v 的子树里
        {
            if (l - 1 >= 1) ans = max(ans, dis[v] + askf(v, l - 1));
        }
        else
        {
            if (l - 1 >= 1) ans = max(ans, dis[u] + askg(v, l - 1) + 1);
        }
    }
    for (i = 1; i <= cnt; i++) vis[id[i]] = 0;
}
int main()
{
    freopen("do.in", "r", stdin);
    freopen("do.out", "w", stdout);
    int i, x, y;
    read(n); read(m);
    for (i = 1; i < n; i++) read(x), read(y), add(x, y);
    dfs1(1, 0);
    dfs2(1, 0);
    prepare();
    int tt = 0, res = 0;
    while (m--)
    {
        read(cnt);
        ans = 0;
        for (i = 1; i <= cnt; i++) read(id[i]);
        for (i = 1; i <= cnt; i++) vis[id[i]] = 1;
        build(), solve();
        printf("%d\n", ans);
    }
    fclose(stdin); fclose(stdout);
    return 0;
}
```

第 4 章　动态树 LCT

731.图上询问（graph，1.5s，512MB）

【参考程序】

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

typedef std::pair<int, int> pi;
#define mp(x, y) std::make_pair(x, y)

const int MaxN = 4e5 + 5;
const int INF = 0x3f3f3f3f;

struct edge {
    int u, v;
} e[MaxN];

int n, m, Q, ans[MaxN];
std::vector<pi> vQ[MaxN];

namespace BIT {
    int c[MaxN];
    inline void modify(int x, int del) { //在位置 x 加上 del
        for (; x <= m; x += x & -x)
            c[x] += del;
    }
    inline int query(int x) { //查询树状数组的前缀和 1~x
        int res = 0;
        for (; x; x ^= x & -x)
            res += c[x];
        return res;
    }
}

namespace LCT {
    struct node {
        int lc, rc, fa;
        int minp, val, rev;
        #define lc(x) tr[x].lc
        #define rc(x) tr[x].rc
        #define fa(x) tr[x].fa
        #define val(x) tr[x].val
        #define rev(x) tr[x].rev
        #define minp(x) tr[x].minp //minp 维护的是子树中权值最小的点的编号
    } tr[MaxN];
```

```
inline int minpos(int a, int b) { //返回权值较小的点的标号
    return val(a) < val(b) ? a : b;
}
inline void upt(int x) {
    minp(x) = x;
    if (lc(x))
        minp(x) = minpos(minp(x), minp(lc(x)));
    if (rc(x))
        minp(x) = minpos(minp(x), minp(rc(x)));
}
inline void rev_tag(int x) {
    std::swap(lc(x), rc(x));
    rev(x) ^= 1;
}
inline void dnt(int x) {
    if (rev(x)) {
        if (lc(x))
            rev_tag(lc(x));
        if (rc(x))
            rev_tag(rc(x));
        rev(x) = 0;
    }
}
inline bool which(int x) {
    return rc(fa(x)) == x;
}
inline bool is_root(int x) {
    return !fa(x) || lc(fa(x)) != x && rc(fa(x)) != x;
}

inline void Rotate(int x) {
    int y = fa(x), z = fa(y);
    if (!is_root(y)) (lc(z) == y ? lc(z) : rc(z)) = x;
    fa(x) = z, fa(y) = x;
    int b = lc(y) == x ? rc(x) : lc(x);
    if (b) fa(b) = y;
    if (lc(y) == x) rc(x) = y, lc(y) = b;
    else lc(x) = y, rc(y) = b;
    upt(y);
}

inline void Splay(int x) {
    static int que[MaxN], qr, y;
```

```
    for (y = x, qr = 0; !is_root(y); y = fa(y)) que[++qr] = y;
    que[++qr] = y;
    for (; qr; --qr) dnt(que[qr]);
    while (!is_root(x)) {
        if (!is_root(fa(x)))
            Rotate(which(fa(x)) == which(x) ? fa(x) : x);
        Rotate(x);
    }
    upt(x);
}

inline void access(int x) {
    for (int y = 0; x; y = x, x = fa(x)) {
        Splay(x), rc(x) = y;
        if (y) fa(y) = x;
        upt(x);
    }
}

inline void make_root(int x) { //将 x 置为 x 所在树的根
    access(x), Splay(x), rev_tag(x);
}

inline int find_root(int x) { //查询 x 所在树的根
    access(x), Splay(x);
    while (lc(x)) {
        dnt(x), x = lc(x);
    }
    return Splay(x), x;
}

inline void link(int x, int y) { //连接 x,y
    make_root(x), fa(x) = y;
}

inline void cut(int x, int y) { //删除连接 x,y 的边
    make_root(x), access(y), Splay(y);
    lc(y) = fa(x) = 0, upt(y);
}

inline void addEdge(int epos) {
    int u = e[epos].u, v = e[epos].v;
    minp(epos + n) = epos + n;
```

```
        if (u == v) return; //忽略自环
        if (find_root(u) != find_root(v)) { //如果 u,v 不连通，则直接加入边
            link(u, epos + n), link(v, epos + n);
            BIT::modify(epos, 1);
        } else {
            make_root(v), access(u), Splay(u);
            int pos = minp(u); //否则删去编号最小的边
            cut(e[val(pos)].u, pos), cut(e[val(pos)].v, pos);
            link(u, epos + n), link(v, epos + n);
            BIT::modify(val(pos), -1), BIT::modify(epos, 1);
        }
    }
}

int main() {
    freopen("graph.in", "r", stdin);
    freopen("graph.out", "w", stdout);
    scanf("%d%d%d", &n, &m, &Q);
    for (int i = 1; i <= n; ++i)
        LCT::val(i) = INF;
    for (int i = 1; i <= m; ++i) {
        LCT::val(i + n) = i;
        scanf("%d%d", &e[i].u, &e[i].v);
    }
    for (int i = 1; i <= Q; ++i) {
        int l, r;
        scanf("%d%d", &l, &r);
        vQ[r].push_back(mp(l, i)); //将询问按右端点分类
    }

    for (int r = 1; r <= m; ++r) {
        LCT::addEdge(r);

        for (int i = 0, im = vQ[r].size(); i < im; ++i) {
            int l = vQ[r][i].first, pos = vQ[r][i].second;
            ans[pos] = n - BIT::query(m) + BIT::query(l - 1);
        }
    }
    for (int i = 1; i <= Q; ++i) {
        printf("%d\n", ans[i]);
    }
    return 0;
}
```

732.斐波那契(fibonacci,2s,512MB)

【参考程序】

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

const int S = 1 << 20;
char frd[S], *ihead = frd + S;
const char *itail = ihead;

inline char nxtChar()
{
    if (ihead == itail)
        fread(frd, 1, S, stdin), ihead = frd;
    return *ihead++;
}

template <class T>
inline void read(T &res)
{
    char ch;
    while (ch = nxtChar(), !isdigit(ch));
    res = ch ^ 48;
    while (ch = nxtChar(), isdigit(ch))
        res = res * 10 + ch - 48;
}

char fwt[S], *ohead = fwt;
const char *otail = ohead + S;

inline void outChar(char ch)
{
    if (ohead == otail)
        fwrite(fwt, 1, S, stdin), ohead = fwt;
    *ohead++ = ch;
}

template <class T>
inline void put(T x)
{
    if (x > 9) put(x / 10);
    outChar(x % 10 + 48);
}

typedef long long ll;
```

```
const int mod = 998244353;
const int inv2 = 499122177;

const int N = 1e5 + 5;
int lc[N], rc[N], fa[N], tfa[N];
int rev[N], que[N], sze[N];
int n, m, qr;

inline void add(int &x, int y)
{
    x += y;
    x >= mod ? x -= mod : 0;
}

inline int quick_pow(int x, int k)
{
    int res = 1;
    while (k)
    {
        if (k & 1) res = 1ll * res * x % mod;
        x = 1ll * x * x % mod; k >>= 1;
    }
    return res;
}

struct Real
{ //定义类(a,b)表示 a+b*sqrt(5)
    int a, b;

    Real() {}
    Real(int A, int B):
        a(A), b(B) {}

    inline Real operator + (const Real &x) const
    {
        int tx = a; add(tx, x.a);
        int ty = b; add(ty, x.b);
        return Real(tx, ty);
    }

    inline Real operator - (const Real &x) const
    {
        int tx = a; add(tx, mod - x.a);
        int ty = b; add(ty, mod - x.b);
```

```
        return Real(tx, ty);
    }

    inline Real operator * (const Real &x) const
    {
        return Real((1ll * a * x.a + 5ll * b * x.b) % mod,
                    (1ll * a * x.b + 1ll * b * x.a) % mod);
    }

    inline Real operator / (const Real &x) const
    {
        int tx, ty, tz;
        tx = (1ll * a * x.a - 5ll * b * x.b) % mod; add(tx, mod);
        ty = (1ll * b * x.a - 1ll * a * x.b) % mod; add(ty, mod);
        tz = (1ll * x.a * x.a - 5ll * x.b * x.b) % mod; add(tz, mod);
        tz = quick_pow(tz, mod - 2);
        return Real(1ll * tx * tz % mod, 1ll * ty * tz % mod);
    }

    inline bool operator != (const Real &x) const
    {
        return a != x.a || b != x.b;
    }

    friend inline Real operator ^ (Real x, int k)
    {
        Real res = Real(1, 0);
        while (k)
        {
            if (k & 1) res = res * x;
            x = x * x; k >>= 1;
        }
        return res;
    }
};
Real pro_a[N], pro_b[N];
Real ans_a[N], ans_b[N];
Real pre_a[N], pre_b[N];
Real suf_a[N], suf_b[N];
Real tag_a[N], tag_b[N];
Real val_a[N], val_b[N];

const Real Zero = Real(0, 0);
const Real One = Real(1, 0);
```

```
const Real Pa = Real(inv2, inv2);
const Real Pb = Real(inv2, mod - inv2);

inline void Uptdate(int x)
{ //维护 pre,suf,pro,ans
    sze[x] = sze[lc[x]] + sze[rc[x]] + 1;

    pro_a[x] = pro_a[lc[x]] * val_a[x] * pro_a[rc[x]];
    pre_a[x] = pre_a[lc[x]] + (pre_a[rc[x]] + One) * val_a[x] *
pro_a[lc[x]];
    suf_a[x] = suf_a[rc[x]] + (suf_a[lc[x]] + One) * val_a[x] *
pro_a[rc[x]];
    ans_a[x] = ans_a[lc[x]] + ans_a[rc[x]] + (pre_a[rc[x]] + One) *
val_a[x] * (suf_a[lc[x]] + One);

    pro_b[x] = pro_b[lc[x]] * val_b[x] * pro_b[rc[x]];
    pre_b[x] = pre_b[lc[x]] + (pre_b[rc[x]] + One) * val_b[x] *
pro_b[lc[x]];
    suf_b[x] = suf_b[rc[x]] + (suf_b[lc[x]] + One) * val_b[x] *
pro_b[rc[x]];
    ans_b[x] = ans_b[lc[x]] + ans_b[rc[x]] + (pre_b[rc[x]] + One) *
val_b[x] * (suf_b[lc[x]] + One);
}

inline bool whichSide(int x)
{
    return lc[fa[x]] == x;
}

inline bool isRoot(int x)
{
    return lc[fa[x]] != x && rc[fa[x]] != x;
}

inline void addRev(int x)
{
    if (!x) return ;
    rev[x] ^= 1;
    std::swap(lc[x], rc[x]);
    std::swap(pre_a[x], suf_a[x]);
    std::swap(pre_b[x], suf_b[x]);
}

inline void addTag(int x, const Real &v1, const Real &v2)
```

```
{ //打标记
    if (!x) return ;

    Real tx, ty;
    tag_a[x] = val_a[x] = v1;
    tag_b[x] = val_b[x] = v2;

    pro_a[x] = v1 ^ sze[x];
    tx = pro_a[x] * v1;
    ty = One / (v1 - One);
    pre_a[x] = (tx - v1) * ty;
    suf_a[x] = pre_a[x];
    ans_a[x] = pre_a[x] * Real(sze[x] + 1, 0)
            - (Real(sze[x], 0) * tx - pre_a[x]) * ty;

    pro_b[x] = v2 ^ sze[x];
    tx = pro_b[x] * v2;
    ty = One / (v2 - One);
    pre_b[x] = (tx - v2) * ty;
    suf_b[x] = pre_b[x];
    ans_b[x] = pre_b[x] * Real(sze[x] + 1, 0)
            - (Real(sze[x], 0) * tx - pre_b[x]) * ty;
}

inline void downDate(int x)
{ //下传标记
    if (rev[x])
    {
        addRev(lc[x]);
        addRev(rc[x]);
        rev[x] = 0;
    }
    if (tag_a[x] != Zero)
    {
        addTag(lc[x], tag_a[x], tag_b[x]);
        addTag(rc[x], tag_a[x], tag_b[x]);
        tag_a[x] = tag_b[x] = Zero;
    }
}

inline void Rotate(int x)
{
    int y = fa[x], z = fa[y],
        b = lc[y] == x ? rc[x] : lc[x];
```

```
    if (!isRoot(y))
        (lc[z] == y ? lc[z] : rc[z]) = x;
    fa[x] = z; fa[y] = x;
    b ? fa[b] = y : 0;
    lc[y] == x ? (rc[x] = y, lc[y] = b) : (lc[x] = y, rc[y] = b);
    Uptdate(y);
}

inline void Splay(int x)
{
    que[qr = 1] = x;
    for (int y = x; !isRoot(y); y = fa[y])
        que[++qr] = fa[y];
    for (int i = qr; i >= 1; --i)
        downDate(que[i]);
    while (!isRoot(x))
    {
        if (!isRoot(fa[x]))
            Rotate(whichSide(fa[x]) == whichSide(x) ? fa[x] : x);
        Rotate(x);
    }
    Uptdate(x);
}

inline void Access(int x)
{
    for (int y = 0; x; y = x, x = fa[x])
    {
        Splay(x);
        rc[x] = y;
        Uptdate(x);
    }
}

inline void makeRoot(int x)
{
    Access(x); Splay(x); addRev(x);
}

inline void Link(int x, int y)
{
    makeRoot(x); fa[x] = y;
}
```

```
inline void Cut(int x, int y)
{
    makeRoot(x);
    Access(y); Splay(y);
    lc[y] = fa[x] = 0;
    Uptdate(y);
}

inline void Modify(int x, int y, const Real &v1, const Real &v2)
{
    makeRoot(x);
    Access(y);
    Splay(y);
    addTag(y, v1, v2);
}

int main()
{
    freopen("fibonacci.in", "r", stdin);
    freopen("fibonacci.out", "w", stdout);
    for (int i = 1; i <= n; ++i)
        tag_a[i] = tag_b[i] = Zero;
    pro_a[0] = pro_b[0] = One;
    pre_a[0] = pre_b[0] = suf_a[0] = suf_b[0] = ans_a[0] = ans_b[0] =
Zero;

    read(n); read(m);
    for (int i = 1, x; i <= n; ++i)
    {
        read(x);
        val_a[i] = Pa ^ x;
        val_b[i] = Pb ^ x;
    }
    for (int i = 2; i <= n; ++i)
        read(tfa[i]), fa[i] = tfa[i];
    int opt, u, v, x;
    for (int i = 1; i <= m; ++i)
    {
        read(opt);
        if (opt == 1)  {
            read(u); read(v);
            Cut(u, tfa[u]);
            Link(u, tfa[u] = v);
        }
```

```
        else if (opt == 2){
            read(u); read(v); read(x);
            Modify(u, v, Pa ^ x, Pb ^ x);
        }
        else if (opt == 3){
            read(u);
            Access(u); Splay(u);
            put((val_a[u] - val_b[u]).b), outChar('\n');
        }
        else       {
            read(u); read(v);
            makeRoot(u); Access(v); Splay(v);
            put((ans_a[v] - ans_b[v]).b), outChar('\n');
        }
    }
    fwrite(fwt, 1, ohead - fwt, stdout);
    fclose(stdin); fclose(stdout);
    return 0;
}
```

733.子树询问(lct,2s,256MB)

【参考程序】

```
#include <bits/stdc++.h>
typedef long long ll;
const int N = 2e5 + 5, M = N << 1;
int n,m,ecnt,nxt[M], adj[N], go[M], fa[N], lc[N], rc[N], len, que[N],
ft[N];
ll f[N], a[N], val[N], minv[N], add[N];
bool lea[N];
void add_edge(int u, int v) {
    nxt[++ecnt] = adj[u], adj[u] = ecnt, go[ecnt] = v;
    nxt[++ecnt] = adj[v], adj[v] = ecnt, go[ecnt] = u;
}
void dfs(int u, int fu) {
    val[u] = a[u], lea[u] = 1; ll s = 0;
    for (int e = adj[u], v; e; e = nxt[e]) {
        if ((v = go[e]) == fu) continue;
        dfs(v, u), s += f[v], lea[u] = 0;
    }
    val[u] -= s, f[u] = lea[u] ? a[u] : std::min(a[u], s);
    minv[u] = val[u], ft[u] = fa[u] = fu;
}
int which(int x) {
    return rc[fa[x]] == x;
```

```
}
void addtag(int x, ll v) {
    add[x] += v, minv[x] += v, val[x] += v;
}
void down(int x) {
    if (lc[x]) addtag(lc[x], add[x]);
    if (rc[x]) addtag(rc[x], add[x]);
    add[x] = 0;
}
bool isRoot(int x) {
    return !fa[x] || (lc[fa[x]] != x && rc[fa[x]] != x);
}
void upt(int x) {
    minv[x] = val[x];
    if (lc[x]) minv[x] = std::min(minv[x], minv[lc[x]]);
    if (rc[x]) minv[x] = std::min(minv[x], minv[rc[x]]);
}
void rotate(int x) {
    int y = fa[x], z = fa[y], b = lc[y] == x ? rc[x] : lc[x];
    if (z && !isRoot(y)) (lc[z] == y ? lc[z] : rc[z]) = x;
    fa[x] = z, fa[y] = x; if (b) fa[b] = y;
    if (lc[y] == x) rc[x] = y, lc[y] = b;
    else lc[x] = y, rc[y] = b;
    upt(y), upt(x);
}
void splay(int x) {
    que[len = 1] = x;
    for (int y = x; !isRoot(y); y = fa[y]) que[++len] = fa[y];
    for (int i = len; i >= 1; i--) down(que[i]);
    while (!isRoot(x)) {
        if (!isRoot(fa[x])) {
            if (which(x) == which(fa[x])) rotate(fa[x]);
            else rotate(x);
        }
        rotate(x);
    }
}
void access(int x) {
    for (int y = 0; x; y = x, x = fa[x]) {
        splay(x), rc[x] = y;
        if (y) fa[y] = x;
        upt(x);
    }
}
```

```
ll query(int x, ll k) {
    if (!x) return 0;
    down(x);
    bool is = !rc[x] || minv[rc[x]] >= k;
    if (is && val[x] < k) return x;
    else if (is) return query(lc[x], k);
    else return query(rc[x], k);
}
void modify(int x, ll v) {
    access(x), splay(x), addtag(x, v);
}
void change(int x, ll y) {
    splay(x), a[x] += y;
    ll k = lea[x] ? y : (val[x] < -y ? y : -val[x]);
    val[x] += y, upt(x);
    if (!lea[x] && val[x] >= y) return;
    x = ft[x];
    while (x) {
        access(x), splay(x);
        int up = query(x, k); ll tmp = val[up];
        modify(x, -k); if (ft[up]) modify(ft[up], k);
        x = ft[up], k = tmp;
        if (k <= 0) return;
    }
}

int main() {
    int x, y; char c; ll z;
    scanf("%d", &n);
    for (int i = 1; i <= n; i++) scanf("%lld", a + i);
    for (int i = 1; i < n; i++) scanf("%d%d", &x, &y), add_edge(x, y);
    dfs(1, 0);
    scanf("%d", &m);
    while (m--) {
        while (c = getchar(), !isupper(c)); scanf("%d", &x);
        if (c == 'C') scanf("%lld", &z), change(x, z);
        else printf("%lld\n", lea[x]?a[x]:(splay(x),std::min(a[x]-val[x],
a[x])));
    }
    return 0;
}
```

第六部分　技巧与思想

第 1 章　分治

752.最优分组(schooldays,2s,1024MB)

【参考程序】

```
#include <iostream>
#include <cstdio>
#define For(i, a, b) for (int i = a, bb = b; i <= bb; ++i)
#define Rof(i, a, b) for (int i = a, bb = b; i >= bb; --i)
#define k2 k << 1
#define k3 k << 1 | 1

const int MaxN = 1e6 + 5, mod = 1e9 + 7;
int c[MaxN], d[MaxN], lft[MaxN], pos[MaxN << 2], n;

inline int Plus(int x, int y)
{
    int res = x + y;
    if (res >= mod) res -= mod;
    return res;
}

struct point
{
    int mx, cnt;
    inline point operator + (const point &rhs) const
    {
        if (mx != rhs.mx)
            return mx > rhs.mx ? *this : rhs;
        return point {mx, Plus(cnt, rhs.cnt)};
    }
} tr[MaxN << 2], tag[MaxN << 2], f[MaxN];

template <class T>
inline void get(T &res)
{
    char ch;
    bool bo = false;
    while (ch = getchar(), !isdigit(ch))
        if (ch == '-') bo = true;
    res = ch ^ 48;
```

```
    while (ch = getchar(), isdigit(ch))
        res = (res << 1) + (res << 3) + (ch ^ 48);
    if (bo) res = ~ res + 1;
    return;
}

template <class T>
inline void _put(T x)
{
    if (x > 9) _put(x / 10);
    putchar(x % 10 + '0');
    return;
}

template <class T>
inline void put(T x, char ch)
{
    if (x < 0) {
        putchar('-');
        x = ~ x + 1;
    }
    _put(x);
    putchar(ch);
    return;
}

inline void solve_lft() //求 lft 数组
{
    int q[MaxN], t = 1, w = 0, now = 0, lim;
    For(i, 1, n)
    {
        while (t <= w && d[q[w]] >= d[i])
            w--;
        q[++w] = i;
        now--;
        do {
            now++;
            while (t <= w && q[t] <= now) t++;
        } while (i - now > d[q[t]]);
        lft[i] = now;
    }
    return;
}
```

```
inline int max_pos(int x, int y)
{
    return c[x] > c[y] ? x : y;
}

inline void upt_seg(int k)
{
    tr[k] = tr[k2] + tr[k3];
    pos[k] = max_pos(pos[k2], pos[k3]);
    return;
}

inline void build_seg(int k, int l, int r)
{
    tag[k] = point {-1, 0};
    if (l == r)
    {
        tr[k] = point {-1, 0};
        pos[k] = l;
        return;
    }
    int mid = l + r >> 1;
    build_seg(k2, l, mid);
    build_seg(k3, mid + 1, r);
    upt_seg(k);
    return;
}

inline void add_seg(int k, point v)
{
    tr[k] = tr[k] + v;
    tag[k] = tag[k] + v;
    return;
}

inline void down_seg(int k)
{
    if (tag[k].mx == -1) return;
    add_seg(k2, tag[k]);
    add_seg(k3, tag[k]);
    tag[k] = point {-1, 0};
    return;
}
```

```
inline void insert_seg(int k, int l, int r, int x, int y, point v)
{
    if (x <= l && r <= y)
        return add_seg(k, v);
    down_seg(k);
    int mid = l + r >> 1;
    if (x <= mid)
        insert_seg(k2, l, mid, x, y, v);
    if (y > mid)
        insert_seg(k3, mid + 1, r, x, y, v);
    upt_seg(k);
    return;
}

inline point query_val_seg(int k, int l, int r, int x, int y)
{   //求[x,y]中的最大分组组数
    if (x > y)
        return point {-1, 0};
    if (x <= l && r <= y)
        return tr[k];
    down_seg(k);
    int mid = l + r >> 1;
    point res = point {-1, 0};
    if (x <= mid)
        res = res + query_val_seg(k2, l, mid, x, y);
    if (y > mid)
        res = res + query_val_seg(k3, mid + 1, r, x, y);
    return res;
}

inline int query_pos_seg(int k, int l, int r, int x, int y)
{ //求[x,y]中使 c[pos]最大的 pos
    if (x <= l && r <= y)
        return pos[k];
    down_seg(k);
    int mid = l + r >> 1;
    if (y <= mid)
        return query_pos_seg(k2, l, mid, x, y);
    else if (x > mid)
        return query_pos_seg(k3, mid + 1, r, x, y);
    else
        return max_pos(query_pos_seg(k2, l, mid, x, y), query_pos_seg(k3,
mid + 1, r, x, y));
}
```

```
inline int query_r(int l, int r, int v)
{  //求[l,r]中 lft[res]小于 v 的最大的 res
    int mid, res;
    while (l <= r)
    {
        mid = l + r >> 1;
        if (lft[mid] < v)
        {
            res = mid;
            l = mid + 1;
        }
        else r = mid - 1;
    }
    return res;
}

inline void solve_trans(int mid, int l, int r)
{   //将[l,mid-1]的状态转移到[mid,r]
    int p = std :: max(mid, l+c[mid]), q = std :: min(r, mid+c[mid]-1);
    int now = p;
    point cur = query_val_seg(1, 0, n, l, p-c[mid]-1);
    while (now <= q && lft[now] < l) //情况 1
    {
        cur = cur + f[now - c[mid]];
        if (cur.mx != -1)
            f[now] = f[now] + point {cur.mx+1, cur.cnt};
        now++;
    }
    if (now > r || lft[now] >= mid)
        return;
    if (lft[now] < l) //情况 2
    {
        int rr = query_r(now, r, l);
        if (cur.mx != -1)
            insert_seg(1, 0, n, now, rr, point {cur.mx + 1, cur.cnt});
        now = rr + 1;
    }
    For(i, now, r) //情况 3
    {
        if (lft[i] >= mid) return;
        cur = query_val_seg(1, 0, n, lft[i], std :: min(mid-1,i-c[mid]));
        if (cur.mx != -1)
            f[i] = f[i] + point {cur.mx + 1, cur.cnt};
```

```
    }
    return;
}

inline void solve(int l, int r) //分治求[l,r]的答案
{
    if (l > r)
        return;
    if (l == r)
    {
        insert_seg(1, 0, n, l, l, f[l]);
        f[l] = query_val_seg(1, 0, n, l, l);
        return;
    }
    int mid = query_pos_seg(1, 0, n, l + 1, r);
    solve(l, mid - 1);
    solve_trans(mid, l, r);
    solve(mid, r);
    return;
}

int main()
{
    freopen("schooldays.in", "r", stdin);
    freopen("schooldays.out", "w", stdout);

    get(n);
    For(i, 1, n)
        get(c[i]), get(d[i]);
    solve_lft();
    f[0] = point {0, 1};
    For(i, 1, n)
        f[i] = point {-1, 0};
    build_seg(1, 0, n);
    solve(0, n);
    if (f[n].mx == -1)
        puts("-1");
    else
        put(f[n].mx, ' '), put(f[n].cnt, '\n');

    fclose(stdin), fclose(stdout);
    return 0;
}
```

第 2 章　启发式合并

763.攻城略池(conquer,1s,512MB)

【参考程序】

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef unsigned int uint;
typedef unsigned long long ull;
typedef long double ld;
#define pair(x, y) make_pair(x, y)
#define runtime() ((double)clock() / CLOCKS_PER_SEC)

inline int read() {
    static int r, sign;
    static char c;
    r = 0, sign = 1;
    do c = getchar(); while (c != '-' && (c < '0' || c > '9'));
    if (c == '-') sign = -1, c = getchar();
    while (c >= '0' && c <= '9')
        r = r * 10 + (int)(c - '0'), c = getchar();
    return sign * r;
}

template <typename T>
inline void print(T *a, int n) {
    for (int i = 1; i < n; ++i) cout << a[i] <<"";
    cout << a[n] << endl;
}
#define PRINT(_l, _r, _s, _t) { cout << #_l #_s "~" #_t #_r ": "; for
(int _i = _s; _i != _t; ++_i) cout << _l _i _r <<""; cout << endl; }

#define N 100000
const int MAX_TIME = (int)3e8;
struct edge {
    int next, node, w;
} e[N << 1 | 1];
int head[N + 1], tot = 0;

inline void addedge(int a, int b, int w) {
    e[++tot].next = head[a];
    head[a] = tot, e[tot].node = b, e[tot].w = w;
}

int n, d[N + 1], f[N + 1], fa[N + 1], w[N + 1], seq[N + 1], cnt = 0;
```

```
void dfs(int x) {
    seq[++cnt] = x;
    for (int i = head[x]; i; i = e[i].next) {
        int node = e[i].node;
        if (node == fa[x]) continue;
        fa[node] = x, w[node] = e[i].w;
        dfs(node);
    }
}

class Node {
public:
    Node *l, *r, *f;
    int s, d, v, m;
    ll sum;
    Node() {}
    Node(int _v) {
        l = r = f = NULL;
        s = 1, d = 0, sum = m = v = _v;
    }
} *root[N + 1];

void update(Node *x) {
    x->s = (x->l ? x->l->s : 0) + (x->r ? x->r->s : 0) + 1;
    x->sum = (x->l ? x->l->sum : 0LL) + (x->r ? x->r->sum : 0LL) +
(ll)x->v;
    x->m = min(x->v, x->l ? x->l->m : MAX_TIME);
}

void apply(Node *x, int d) {
    x->m += d, x->d += d, x->v += d, x->sum += (ll)d * x->s;
}

void push(Node *x) {
    if (!x->d) return ;
    if (x->l) apply(x->l, x->d);
    if (x->r) apply(x->r, x->d);
    x->d = 0;
}

void zig(Node *x) {
    static Node *y, *z;
    y = x->f, z = y->f;
    push(x);
    if (z && z->l == y) z->l = x;
```

```
    else if (z && z->r == y) z->r = x;
    if (x->r) x->r->f = y;
    y->l = x->r;
    x->r = y, y->f = x, x->f = z;
    update(y);
}

void zag(Node *x) {
    static Node *y, *z;
    y = x->f, z = y->f;
    push(x);
    if (z && z->l == y) z->l = x;
    else if (z && z->r == y) z->r = x;
    if (x->l) x->l->f = y;
    y->r = x->l;
    x->l = y, y->f = x, x->f = z;
    update(y);
}

void splay(Node *x, Node **r) {
    static Node *y, *z, *f;
    f = (*r)->f;
    while (x->f != f) {
        y = x->f, z = y->f;
        if (z) push(z);
        push(y);
        if (z == f) {
            if (y->l == x) zig(x);
            else zag(x);
        } else {
            if (z->l == y) {
                if (y->l == x) zig(y), zig(x);
                else zag(x), zig(x);
            } else {
                if (y->l == x) zig(x), zag(x);
                else zag(y), zag(x);
            }
        }
    }
    push(x);
    update(*r = x);
}

void insert(Node **r, Node *x) { //在 Splay 中插入一个元素
    if (*r == NULL) {
        *r = x;
```

```
        return ;
    }
    Node *p = *r, *f = NULL;
    while (p != NULL) {
        f = p, push(p);
        if (p->v >= x->v) p = p->l;
        else p = p->r;
    }
    x->f = f, x->l = x->r = NULL;
    if (f->v < x->v) f->r = x;
    else f->l = x;
    splay(x, r);
}

Node* merge(Node *x, Node *y) { //合并两棵 Splay
    if (x == NULL) return y;
    if (y == NULL) return x;
    if (x->s > y->s) return merge(y, x);
    static queue <Node*> q;
    q.push(x);
    while (!q.empty()) {
        x = q.front();
        q.pop(), push(x);
        if (x->l) q.push(x->l);
        if (x->r) q.push(x->r);
        insert(&y, x);
    }
    return y;
}

Node* upper_bound(Node *x, int v) {
    Node *r = NULL;
    while (x != NULL) {
        push(x);
        if (x->v > v) r = x, x = x->l;
        else x = x->r;
    }
    return r;
}

ll check(Node **x, int t) {
    Node *p = upper_bound(*x, t);
    if (p == NULL) return (ll)t * (*x)->s - (*x)->sum;
    splay(p, x);
    return (*x)->l ? ((ll)t * (*x)->l->s - (*x)->l->sum) : 0LL;
}
```

```
int calc(Node *x, int v, int s, ll sum) { //在 Splay 上二分
    push(x);
    if (x->l) {
        if ((ll)x->v * (s + x->l->s) - (sum + x->l->sum) >= v)
            return calc(x->l, v, s, sum);
        s += x->l->s, sum += x->l->sum;
    }
    int r = x->r ? x->r->m : MAX_TIME;
    s += 1, sum += x->v;
    if ((ll)r * s - sum >= v)
        return (v + sum + s - 1) / s;
    return calc(x->r, v, s, sum);
}

int main(int argc, char *argv[]) {
    freopen("conquer.in", "r", stdin);
    freopen("conquer.out", "w", stdout);

    ios :: sync_with_stdio(false);
    cin >> n;
    for (int i = 1; i <= n; ++i) cin >> d[i];
    for (int i = 1; i < n; ++i) {
        int x, y, w;
        cin >> x >> y >> w;
        addedge(x, y, w), addedge(y, x, w);
    }

    dfs(1);

    for (int _i = n; _i > 0; --_i) {
        int i = seq[_i];
        if (d[i] > 0)
            f[i] = calc(root[i], d[i], 0, 0LL);
        insert(&root[i], new Node(f[i]));
        apply(root[i], w[i]);
        if (fa[i] != 0) root[fa[i]] = merge(root[i], root[fa[i]]);
    }

    int ans = *max_element(f + 1, f + n + 1);
    cout << ans << endl;

    fclose(stdin); fclose(stdout);
    return 0;
}
```

第 3 章　分块算法

771.历史序列 (seq,1s,32MB)

【参考程序】

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
const int e = 1e5 + 5, o = 400;
vector<int>op[o];
int pos[e], mod[e], q1[o], q2[o], bl[o], br[o], s, n, m, bel[e], cnt[o];
ll a[e], delta[o];
template <class t>
inline void read(t & res)
{
    res = 0;
    char ch; int f = 1;
    while (ch = getchar(), (ch < '0' || ch > '9') && ch != '-');
    if (ch == '-') f = -1;
    else res = ch - 48;
    while (ch = getchar(), ch >= '0' && ch <= '9')
    res = res * 10 + ch - 48;
    if (f == -1) res = ~res + 1;
}
template <class t>
inline void print(t x)
{
    if (x < 0)
    {
        putchar('-');
        print(~x + 1);
    }
    else
    {
        if (x > 9) print(x / 10);
        putchar(x % 10 + 48);
    }
}
inline bool cmp(int x, int y)
{
    return a[x] < a[y];
}
inline void init()
{
    s = sqrt(n);
```

```
    int i;
    for (i = 1; i <= n; i++)
    {
        bel[i] = (i - 1) / s + 1;
        pos[i] = i;
    }
    for (i = 1; i <= bel[n]; i++)
    {
        bl[i] = br[i - 1] + 1;
        br[i] = min(bl[i] + s - 1, n);
        sort(pos + bl[i], pos + br[i] + 1, cmp); //对每块分别从小到大排序
    }
}
inline void pushdown(int x)
{
    int l = bl[x],r = br[x],i,len = op[x].size(),cs = cnt[x]+len, j = 0;
    if (!len)
    {
        for (i = l; i <= r; i++)
        {
            a[i] += delta[x];
            mod[i] += cnt[x];
        }
        cnt[x] = delta[x] = 0;
        return;
    }
    ll lst = op[x][len - 1];
    for (i = l; i <= r; i++)
    {
        int k = pos[i];
        while (j < len && a[k] >= op[x][j]) // 此操作对 a[k]无影响，跳过
        {
            j++;
            cs--;//块内元素升序使得可以 O(块大小)求出每个元素的数值改变次数
        }
        mod[k] += cs;
        if (j != len) a[k] = lst; // 除非全部跳过，否则都会先变成 lst
    }
    op[x].clear();
    cnt[x] = 0;
    for (i = l; i <= r; i++)
    {
        a[i] += delta[x]; // op 里已经把前面无效的 delta 减掉了
        mod[i] += cnt[x];
```

```
    }
    delta[x] = 0;
}
inline void addsort(int l, int r, int d) // 不完整的块内每个元素+d
{
    int i, bk = bel[l], w1 = 0, w2 = 0;
    pushdown(bk); // 排序之前先下传标记
    for (i = bl[bk]; i <= br[bk]; i++)
    {
        int k = pos[i];
        if (l <= k && k <= r) q1[++w1] = k;
        else q2[++w2] = k;
    }
    int t1 = 1, t2 = 1;
    for (i = bl[bk]; i <= br[bk]; i++)
    if (t2 > w2 || t1 <= w1 && a[q1[t1]] + d <= a[q2[t2]]) // 归并排序
    {
        int k = q1[t1++];
        pos[i] = k;
        a[k] += d;
        mod[k]++; // mod[k]是 a[k]的数值改变次数
    }
    else pos[i] = q2[t2++];
}
inline void covsort(int l, int r, int d) // 不完整的块内每个元素和 d 取 max
{
    int i, bk = bel[l], w1 = 0, w2 = 0;
    pushdown(bk);
    for (i = bl[bk]; i <= br[bk]; i++)
    {
        int k = pos[i];
        if (l <= k && k <= r && a[k] < d) q1[++w1] = k;
        else q2[++w2] = k;
    }
    int t1 = 1, t2 = 1;
    for (i = bl[bk]; i <= br[bk]; i++)
    if (t2 > w2 || t1 <= w1 && d <= a[q2[t2]])
    {
        int k = q1[t1++];
        pos[i] = k;
        a[k] = d;
        mod[k]++;
    }
    else pos[i] = q2[t2++];
```

```
}
inline void update(int l, int r, int d) // 区间加 d
{
    int i, nl = bel[l], nr = bel[r];
    if (nl == nr)
    {
        addsort(l, r, d);
        return;
    }
    addsort(l, br[nl], d);
    addsort(bl[nr], r, d);
    for (i = nl + 1; i < nr; i++)
    {
        delta[i] += d; // 打标记，区间 i 数值集体+d
        cnt[i]++; // 打标记，区间 i 数值变化次数集体+1
    }
}
inline void change(int l, int r, int d) // 区间和 d 取 max
{
    int i, nl = bel[l], nr = bel[r];
    if (nl == nr)
    {
        covsort(l, r, d);
        return;
    }
    covsort(l, br[nl], d);
    covsort(bl[nr], r, d);
    for (i = nl + 1; i < nr; i++)
    {
        if (op[i].empty() || d - delta[i] > op[i].back())
        op[i].push_back(d - delta[i]); // 维护上升序列
    }
}
int main()
{
    freopen("seq.in", "r", stdin);
    freopen("seq.out", "w", stdout);
    char ch;
    int i, l, r, c;
    read(n);
    for (i = 1; i <= n; i++) read(a[i]);
    init();
    read(m);
    while (m--)
```

```
    {
        while (ch = getchar(), ch != 'A' && ch != 'M' && ch != 'Q');
        if (ch == 'A')
        {
            read(l);
            read(r);
            read(c);
            if (c == 0) continue;
            update(l, r, c);
        }
        else if (ch == 'M')
        {
            read(l);
            read(r);
            read(c);
            change(l, r, c);
        }
        else
        {
            read(l);
            pushdown(bel[l]);
            print(a[l]);
            putchar(' ');
            print(mod[l]);
            putchar('\n');
        }
    }
    fclose(stdin); fclose(stdout);
    return 0;
}
```

773.区间异或(xor,2s,512MB)

【参考程序】

```
#include <bits/stdc++.h>
const int N = 3e5 + 5, mod = (1 << 30) - 1;
int a[N], b[N], c[N], A[N], B[N], C[N], bel[N], val[N];
int bl[N], br[N], tag1[N], tag2[N], cnt1[N], cnt2[N], sum1[N], sum2[N];
int n, m, S, bm, cm, K;
bool vis[N];
std::vector<int> v[N];
inline void add(int &x, int y) {
    x = x + y & mod;
}
struct query {
```

```
    int st, ed, w;
    inline void scan() {
        scanf("%d%d%d", &st, &ed, &w);
    }
    inline bool operator < (const query &a) const {
        return ed < a.ed;
    }
} p[N];

inline void Modify(int x, int w) { //将[x,n]加上 w
    for (int i = x, im = br[bel[x]]; i <= im; ++i) add(tag1[i], w);
    for (int i = bel[x] + 1; i <= bm; ++i) add(tag2[i], w);
}

inline void solve(int v1, int v2) { //算法一
    for (int i = 0; i <= n + 1; ++i) sum1[i] = sum2[i] = 0;
    for (int i = 1; i <= m; ++i) {
        add(sum1[p[i].ed], p[i].w);
        add(sum1[p[i].st - 1], mod + 1 - p[i].w);
    }
    for (int i = n; i >= 1; --i) {
        add(sum1[i], sum1[i + 1]), add(sum2[i], sum2[i + 1]);
        if (c[B[i]] == v2) add(sum2[i], sum1[i]);
        if (c[A[i]] == v1) add(b[i], sum2[i]);
    }

    for (int i = 0; i <= n + 1; ++i) sum1[i] = sum2[i] = 0;
    for (int i = 1; i <= n; ++i) sum1[i] = sum1[i - 1] + (c[B[i]] == v2);
    for (int i = 1; i <= m; ++i)
        add(sum2[p[i].st - 1],
        1ll * (sum1[p[i].ed] - sum1[p[i].st - 1]) * p[i].w & mod);
    for (int i = n; i >= 1; --i) {
        add(sum2[i], sum2[i + 1]);
        if (c[A[i]] == v1) add(b[i], mod + 1 - sum2[i]);
    }

    for (int i = 0; i <= n + 1; ++i) sum1[i] = sum2[i] = 0;
    for (int i = 1; i <= m; ++i) add(sum1[p[i].ed], p[i].w);
    for (int i = n; i >= 1; --i) add(sum1[i], sum1[i + 1]);
    for (int i = 1, cnt = 0; i <= n; ++i) {
        cnt += c[A[i]] == v1;
        if (c[B[i]] == v2) add(b[i + 1], mod+1-(1ll*cnt*sum1[i] & mod));
    }
```

```
    for (int i = 0; i <= n + 1; ++i) sum1[i] = 0;
    for (int i = 1; i <= m; ++i) add(sum1[p[i].st - 1], p[i].w);
    for (int i = n; i >= 1; --i) add(sum1[i], sum1[i + 1]);
    for (int i = 1, cnt = 0; i <= n; ++i) {
        cnt += c[A[i]] == v1;
        if (c[B[i]] == v2)
            add(b[i + 1], 1ll * cnt * sum1[i] & mod);
    }

    for (int i = 0; i <= n + 1; ++i) sum1[i] = sum2[i] = 0;
    for (int i = 1; i <= n; ++i) sum1[i] = sum1[i - 1] + (c[A[i]] == v1);
    for (int i = 1; i <= m; ++i) {
        int tmp = 1ll * sum1[p[i].st - 1] * p[i].w & mod;
        add(sum2[p[i].st], tmp), add(sum2[p[i].ed + 1], mod + 1 - tmp);
    }
    for (int i = 1; i <= n; ++i) {
        add(sum2[i], sum2[i - 1]);
        if (c[B[i]] == v2) add(b[i + 1], sum2[i]);
    }
}

int main() {
    freopen("xor.in", "r", stdin);
    freopen("xor.out", "w", stdout);

    scanf("%d%d%d", &n, &m, &K);
    for (int i = 1; i <= n; ++i) {
        scanf("%d", val + i), val[i] ^= val[i - 1];
        A[i] = val[i - 1], B[i] = val[i];
    }

    for (int i = 0; i <= n; ++i)
        c[++cm] = val[i], c[++cm] = val[i] ^ K;
    std::sort(c + 1, c + cm + 1);
    cm = std::unique(c + 1, c + cm + 1) - c - 1;
    for (int i = 1; i <= n; ++i) {
        //离散化
        A[i] = std::lower_bound(c + 1, c + cm + 1, A[i]) - c;
        C[i] = std::lower_bound(c + 1, c + cm + 1, B[i] ^ K) - c;
        B[i] = std::lower_bound(c + 1, c + cm + 1, B[i]) - c;
        v[A[i]].push_back(i);
    }
    for (int i = 1; i <= n; ++i) ++cnt1[C[i]];
```

```
    S = ceil(sqrt(n));
    for (int i = 1; i <= n; ++i) bel[i] = (i - 1) / S + 1;
    bl[bm = 1] = 1;
    for (int i = 1; i <= n; ++i)
        if (bel[i] != bel[i + 1]) br[bm] = i, bl[++bm] = i + 1;
    --bm;

    for (int i = 1; i <= m; ++i) p[i].scan();
    std::sort(p + 1, p + m + 1);
    for (int i = n, r = m; i >= 1; --i) {
        while (r > 0 && p[r].ed >= i) Modify(p[r].st, p[r].w), --r;
        if (!vis[B[i]]) {
            if (cnt1[C[i]] - cnt2[C[i]] <= S) {
                //算法二
                for (int j = 0, jm = v[C[i]].size(); j < jm; ++j) {
                    int t = v[C[i]][j]; if (t > i) break;
                    int tmp = tag1[t] + tag2[bel[t]] & mod;
                    add(b[t], tmp), add(b[i + 1], mod + 1 - tmp);
                }
            } else {
                vis[B[i]] = true, solve(c[B[i]] ^ K, c[B[i]]);
            }
        }
        ++cnt2[C[i]];
    }

    for (int i = 1; i <= n; ++i)
        add(b[i], b[i - 1]);
    for (int i = 1; i <= n; ++i)
        printf("%d%c", b[i], " \n"[i == n]);

    return 0;
}
```

第 4 章　莫队算法

783.树上染色（cw，5s，512MB）

【参考程序】

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

template <class T>
inline void read(T &x)
{
    static char ch;
    while (!isdigit(ch = getchar()));
    x = ch - '0';
    while (isdigit(ch = getchar()))
        x = x * 10 + ch - '0';
}

template <class T>
inline void putint(T x)
{
    static char buf[15], *tail = buf;
    if (!x) putchar('0');
    else
    {
        for (; x; x /= 10) *++tail = x % 10 + '0';
        for (; tail != buf; --tail) putchar(*tail);
    }
}

template <class T>
inline void relax(T &x, const T &y)
{
    if (x < y)
        x = y;
}

#define lcc x << 1, l, mid
#define rcc x << 1 | 1, mid + 1, r
#define trav(u) for (int e = adj[u], v; v = to[e], e; e = nxt[e])

const int S = 3422;
const int MaxNV = 1e5 + 5;
const int MaxNE = MaxNV << 1;
```

```
const int MaxNS = MaxNV << 2;

struct request
{
    int lca, l, r, t, pos, ls, rs;

    request(){}
    request(int lc, int c, int d, int tim, int p):
        lca(lc), l(c), r(d), t(tim), pos(p)
    {
        ls = (l - 1) / S + 1;
        rs = (r - 1) / S + 1;
    }

    inline bool operator < (const request &rhs) const
    {
        if (ls != rhs.ls)
            return ls < rhs.ls;
        if (rs != rhs.rs)
            return rs < rhs.rs;
        return t < rhs.t;
    }
}req[MaxNV];

int n, m, mC, mQ, mR;
int a[MaxNV], b[MaxNV];

int nS, seq[MaxNE], st[MaxNV], ed[MaxNV];

int nE, to[MaxNE], nxt[MaxNE], adj[MaxNV];
int fa[MaxNV], sze[MaxNV], son[MaxNV], dep[MaxNV];
int nP, pos[MaxNV], idx[MaxNV], top[MaxNV];

int chg[MaxNV][3], ans[MaxNV];

inline void addEdge(int u, int v)
{
    nxt[++nE] = adj[u];
    adj[u] = nE;
    to[nE] = v;
}

namespace segment_tree
{
```

```
struct info
{
    int sze, lef, rit;
    info(){}
    info(int a, int b, int c):
        sze(a), lef(b), rit(c) {}
    inline info operator + (const info &rhs) const//合并信息
    {
        if (lef == -1)
            return rhs;
        if (rhs.lef == -1)
            return *this;
        return info(sze + rhs.sze - (rit == rhs.lef), lef, rhs.rit);
    }
    inline info swap() //将信息翻转
    {
        return info(sze, rit, lef);
    }
}seg[MaxNS];

inline void upt(int x)
{
    seg[x] = seg[x << 1] + seg[x << 1 | 1];
}

inline void build(int x, int l, int r) //初始化建线段树
{
    if (l == r)
    {
        seg[x] = info(1, a[idx[l]], a[idx[l]]);
        return;
    }

    int mid = l + r >> 1;
    build(lcc), build(rcc);
    upt(x);
}

inline info seg_query(int x, int l, int r, int u, int v)
{//查询[u,v]颜色段信息
    if (u <= l && r <= v)
        return seg[x];

    info res(0, -1, -1);
```

```
        int mid = l + r >> 1;
        if (u <= mid)
            res = res + seg_query(lcc, u, v);
        if (v > mid)
            res = res + seg_query(rcc, u, v);
        return res;
    }

    inline void seg_modify(int x, int l, int r, int pos, int val)
    {   //将线段树上位置为 pos 的颜色改为 val
        if (l == r)
        {
            seg[x] = info(1, val, val);
            return;
        }
        int mid = l + r >> 1;
        if (pos <= mid)
            seg_modify(lcc, pos, val);
        else
            seg_modify(rcc, pos, val);
        upt(x);
    }
}

namespace heavy_path
{
    using namespace segment_tree;

    inline void dfs_init(int u)
    {
        seq[st[u] = ++nS] = u; //记录结点的入栈出栈序

        sze[u] = 1;
        trav(u)
            if (v != fa[u])
            {
                fa[v] = u;
                dep[v] = dep[u] + 1;

                dfs_init(v);
                sze[u] += sze[v];
                if (sze[v] > sze[son[u]])
                    son[u] = v;
```

```
        }

    seq[ed[u] = ++nS] = u;
}

inline void dfs_heavy(int u) //重链剖分
{
    if (son[u])
    {
        top[son[u]] = top[u];
        idx[pos[son[u]] = ++nP] = son[u];
        dfs_heavy(son[u]);
    }

    trav(u)
        if (v != fa[u] && v != son[u])
        {
            top[v] = v;
            idx[pos[v] = ++nP] = v;

            dfs_heavy(v);
        }
}

inline void node_modify(int u, int val) //单点改颜色
{
    seg_modify(1, 1, n, pos[u], val);
}

inline int path_query(int u, int v)//查询路径颜色段数
{
    info resl(0, -1, -1), resr(0, -1, -1);

    while (top[u] != top[v])
    {
        if (dep[top[u]] < dep[top[v]])//注意 u,v 向上的两个路径分开考虑
        {
            resr = seg_query(1, 1, n, pos[top[v]], pos[v]) + resr;
            v = fa[top[v]];
        }
        else
        {
           resl =resl + seg_query(1, 1, n, pos[top[u]], pos[u]).swap();
            u = fa[top[u]];
```

```
            }
        }

        if (dep[u] < dep[v]) //最后也要注意合并信息的顺序
            return (resl+seg_query(1, 1, n, pos[u], pos[v]) + resr).sze;
        else
            return (resl+seg_query(1,1,n,pos[v],pos[u]).swap()+resr).sze;
    }

    inline int query_lca(int u, int v) //查询 u,v 的 lca
    {
        while (top[u] != top[v])
        {
            if (dep[top[u]] > dep[top[v]])
                u = fa[top[u]];
            else
                v = fa[top[v]];
        }
        return dep[u] < dep[v] ? u : v;
    }
}

bool exist[MaxNV];
int cnt[MaxNV], sum[MaxNV], maxcnt;

inline void add_x(int x) //颜色 x 的出现次数加一
{
    ++cnt[x];
    if (cnt[x])
        --sum[cnt[x] - 1];
    if (!(sum[cnt[x]]++))
        relax(maxcnt, cnt[x]);
}

inline void dec_x(int x) //颜色 x 的出现次数减一
{
    --cnt[x], ++sum[cnt[x]];
    if (!(--sum[cnt[x] + 1]) && maxcnt == cnt[x] + 1)
        --maxcnt;
}

inline void modify_x(int x) //将树上结点 x 的选取状态取反
{
    if (exist[x])
```

```
    {
        exist[x] = false;
        dec_x(a[x]);
    }
    else
    {
        exist[x] = true;
        add_x(a[x]);
    }
}

int main()
{
    freopen("cw.in", "r", stdin);
    freopen("cw.out", "w", stdout);

    read(n), read(m);
    for (int i = 1; i <= n; ++i)
    {
        read(a[i]);
        b[i] = a[i];
    }

    for (int i = 1; i < n; ++i)
    {
        int u, v;
        read(u), read(v);
        addEdge(u, v), addEdge(v, u);
    }

    using namespace heavy_path;

    dfs_init(1);
    nP = idx[1] = pos[1] = top[1] = 1;
    dfs_heavy(1);

    using namespace segment_tree;

    build(1, 1, n);

    for (int i = 1; i <= m; ++i)
    {
        int opt, u, v;
        read(opt), read(u), read(v);
```

```
    if (opt == 1)
    {
        ++mC;
        chg[mC][0] = u;
        chg[mC][1] = b[u];
        chg[mC][2] = b[u] = v;

        node_modify(u, v);
    }
    else if (opt == 2)
        ans[++mQ] = path_query(u, v);
    else
    {
        ++mQ;
        if (u == v)
        {
            ans[mQ] = 1;
            continue;
        }

        int lca = query_lca(u, v);
        if (u == lca || v == lca)
        {
            if (dep[u] > dep[v]) std::swap(u, v);
            req[++mR] = request(0, st[u], st[v], mC, mQ);
        }
        else
        {
            if (ed[u] > st[v]) std::swap(u, v);
            req[++mR] = request(lca, ed[u], st[v], mC, mQ);
        }
    }
}

std::sort(req + 1, req + mR + 1);
sum[0] = n;

int l = 1, r = 0, tim = 0;
for (int i = 1; i <= mR; ++i)
{
    int ql = req[i].l, qr = req[i].r, qt = req[i].t;

    while (tim < qt) //移动时间指针
    {
```

```
            ++tim;
            int x = chg[tim][0];
            a[x] = chg[tim][2];
            if (exist[x])
            {
                dec_x(chg[tim][1]);
                add_x(chg[tim][2]);
            }
        }
        while (tim > qt)
        {
            int x = chg[tim][0];
            a[x] = chg[tim][1];
            if (exist[x])
            {
                add_x(chg[tim][1]);
                dec_x(chg[tim][2]);
            }
            --tim;
        }

        while (l < ql) modify_x(seq[l++]);
        while (l > ql) modify_x(seq[--l]); //移动序列左指针
        while (r < qr) modify_x(seq[++r]);
        while (r > qr) modify_x(seq[r--]); //移动序列右指针

        if (req[i].lca)
        {
            modify_x(req[i].lca);
            ans[req[i].pos] = maxcnt;
            modify_x(req[i].lca);
        }
        else
            ans[req[i].pos] = maxcnt;
    }

    for (int i = 1; i <= mQ; ++i)
    {
        printf("%d", ans[i]);
        putchar('\n');
    }
    return 0;
}
```

第 5 章　CDQ&整体二分

792.区间 k 小（kth，3s，1024MB）

【参考程序】

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

template <class T>
inline void read(T &x) {
    static char ch;
    while (!isdigit(ch = getchar()));
    x = ch - '0';
    while (isdigit(ch = getchar()))
        x = x * 10 + ch - '0';
}

template <class T>
inline void putint(T x) {
    static char buf[15], *tail = buf;
    if (!x) putchar('0');
    else {
        for (; x; x /= 10) *++tail = x % 10 + '0';
        for (; tail != buf; --tail) putchar(*tail);
    }
}

typedef std::vector<int> vi;
typedef std::pair<int, int> pi;
typedef std::vector<pi> vpi;

#define mp(x, y) std::make_pair(x, y)

const int MaxN = 4e5 + 5;
const int INF = 0x3f3f3f3f;
const int MaxNode = 7e7 + 5;

struct node {
    int lc, rc, tag;
    #define lc(x) tr[x].lc
    #define rc(x) tr[x].rc
    #define tag(x) tr[x].tag
}tr[MaxNode];

struct rectangle {
    int lx, rx, ly, ry;
```

```
    rectangle(){}
    rectangle(int la, int ra, int lb, int rb):
        lx(la), rx(ra), ly(lb), ry(rb) {}
}rec1[MaxN], rec2[MaxN];

struct operate
{
    int x, l, r, w;
    operate(){}
    operate(int a, int b, int c, int d): x(a), l(b), r(c), w(d) {}
    inline bool operator < (const operate &rhs) const
    {
        return x <rhs.x;
    }
};

intnT;
int n, W, Q, type;

int a[MaxN];
vi pos[MaxN];
vpi rt[MaxN];

inline void seg_modify(int lst,int&x,int l,int r, int u, int v, int val)
{ //线段树区间修改
    tr[x = ++nT] = tr[lst];

    if (u <= l && r <= v)
    {
        tag(x) += val;//因为我们只需要单点查询，所以维护区间标记即可
        return;
    }

    int mid = l + r >> 1;
    if (u <= mid)
        seg_modify(lc(lst), lc(x), l, mid, u, v, val);
    if (v > mid)
        seg_modify(rc(lst), rc(x), mid + 1, r, u, v, val);
}

inline int seg_query(int x, int l, int r, int p, int lab = 0)
{
    if (l == r)
        return tag(x) + lab; //单点查询只要把走过结点的标记相加即为答案
```

```
    int mid = l + r >> 1;
    if (p <= mid)
        return seg_query(lc(x), l, mid, p, lab + tag(x));
    else
        return seg_query(rc(x), mid + 1, r, p, lab + tag(x));
}

inline void solve(int x, int l, int r)
{
    if (l == r) return;

    int mid = l + r >> 1;
    std::vector<operate>t_oper(0);

    for (inti = l; i<= mid; ++i)
    {
        intsze = pos[i].size();
        for (int j = 0; j <sze; ++j)
        {
            int p = pos[i][j];

       t_oper.push_back(operate(rec1[p].lx+0,rec1[p].ly,rec1[p].ry,+1));
       t_oper.push_back(operate(rec1[p].rx+1,rec1[p].ly,rec1[p].ry,-1));
            if (rec2[p].lx)
            {
                t_oper.push_back(operate(rec2[p].lx+0,rec2[p].ly,
rec2[p].ry,-W));
                t_oper.push_back(operate(rec2[p].rx+1,rec2[p].ly,
rec2[p].ry,+W)); //将矩形修改拆成两个部分
            }
        }
    }
    std::sort(t_oper.begin(), t_oper.end());//将区间内所有修改按横坐标排序

    intsze = t_oper.size(), lst_rt = 0;
    for (inti = 0, lst = 0; i<sze; i = lst + 1)
    {
        int now = t_oper[i].x; lst = i;
        while (lst<sze - 1 && now == t_oper[lst + 1].x)
            ++lst; //把横坐标相同的修改一起处理

        for (int j = i; j <= lst; ++j)
        {
            operate t = t_oper[j];
```

```
            seg_modify(lst_rt, lst_rt, 1, n, t.l, t.r, t.w);
        }
        rt[x].push_back(mp(now, lst_rt));
        //每个涉及到的横坐标记录一个可持久化线段树的根
    }

    solve(x << 1, l, mid);
    solve(x << 1 | 1, mid + 1, r);
}

inline int query_ans(int x, int l, int r, intnl, intnr, int k)
{
    if (l == r) return l;

    int mid = l + r >> 1;
    intrt_pos = std::upper_bound(rt[x].begin(), rt[x].end(), mp(nl,INF))
- rt[x].begin(); //二分出查询对应的可持久化线段树的根

    if (!rt_pos)
        return query_ans(x << 1 | 1, mid + 1, r, nl, nr, k);

    --rt_pos;
    int u = rt[x][rt_pos].second;
    intlsze = seg_query(u, 1, n, nr); //根据[l,mid]的信息二分答案

    if (lsze>= k)
        return query_ans(x << 1, l, mid, nl, nr, k);
    else
        return query_ans(x << 1 | 1, mid + 1, r, nl, nr, k-lsze);
}

intmain()
{
    freopen("kth.in", "r", stdin);
    freopen("kth.out", "w", stdout);

    intlast_ans = 0;
    read(n), read(W), read(Q), read(type);
    for (inti = 1; i<= n; ++i)
    {
        read(a[i]);
        pos[a[i]].push_back(i);
    }
```

```
    for (inti = 0; i< n; ++i)
        if (!pos[i].empty())
        {
            intsze = pos[i].size();
            for (int j = 0; j <sze; ++j)
            {
                int p = pos[i][j];
                if (j >= W) //预处理修改的矩形
                {
                    rec1[p] = rectangle(pos[i][j - W] + 1, p, p, n);
                    if (j > W)
                       rec2[p]=rectangle(pos[i][j-W-1]+1,pos[i][j-W],p,n);
                    else
                        rec2[p] = rectangle(1, pos[i][j - W], p, n);
                }
                else
                    rec1[p] = rectangle(1, p, p, n);
            }
        }

    solve(1, 0, n);

    while (Q--)
    {
        int l, r, k;
        read(l), read(r), read(k);
        if (type)
        {
            l ^= last_ans;
            r ^= last_ans;
            k ^= last_ans;
        }

        putint(last_ans = query_ans(1, 0, n, l, r, k));
        putchar('\n');
    }

    return 0;
}
```

第七部分　计算几何

第 1 章　计算几何初探

821.遮天蔽日(blot,1s,512MB)

【参考程序】

```
#include <bits/stdc++.h>
using namespace std;
#define ld long double

const int N = 105;
const ld pi = 3.141592653589, eps = 1e-11;

int n, m;
ld a1, a2, ans, t1, t2, T;

struct point {
    ld x, y;
    point() {}
    point(ld _x, ld _y) :
        x(_x), y(_y) {}

    inline void scan() {
        scanf("%Lf%Lf", &x, &y);
    }
    inline point rotate(ld a) { //求绕(0,0)逆时针旋转(弧度为 a)后的位置
        ld cosA = cos(a), sinA = sin(a);
        return point(x * cosA - y * sinA, x * sinA + y * cosA);
    }
    inline ld len() { // 求到原点的距离
        return sqrt(x * x + y * y);
    }
} S, G, b[N];

inline point operator + (point a, point b) {
    return point(a.x + b.x, a.y + b.y);
}
inline point operator - (point a, point b) {
    return point(a.x - b.x, a.y - b.y);
}
inline point operator * (point a, ld v) {
    return point(a.x * v, a.y * v);
}
inline point operator / (point a, ld v) {
```

```
    return point(a.x / v, a.y / v);
}
inline ld dot(point a, point b) { // 求 ab 的点积
    return a.x * b.x + a.y * b.y;
}
inline ld operator * (point a, point b) { // 求 ab 的叉积
    return a.x * b.y - b.x * a.y;
}

inline bool is_cross(point a, point b, point c, point d)
{ //判断线段 ab 和 cd 是否相交
    if (min(a.x,b.x)-max(c.x,d.x)>eps || min(c.x,d.x)-max(a.x,b.x)>eps)
        return 0;
    if (min(a.y,b.y)-max(c.y,d.y)>eps || min(c.y,d.y)-max(a.y,b.y)>eps)
        return 0;
    if (((d - a) * (b - a)) * ((c - a) * (b - a)) > eps) return 0;
    if (((d - c) * (a - c)) * ((d - c) * (b - c)) > eps) return 0;
    return 1;
}

inline int nxt(int x) {
    return x == n ? 1 : x + 1;
}

struct polygon {
    point p[N];

    inline ld area() { // 求多边形的面积
        ld res = 0;
        for (int i = 1; i <= n; i++) res += p[i] * p[nxt(i)];
        return res / 2.0;
    }

    inline void init() {
        for (int i = 1; i <= n; i++)
            p[i].scan();
        if (area() < 0)
            reverse(p + 1, p + n + 1);
    }

    inline void getG() { // 求多边形的重心
        ld sum = area(), gx = 0, gy = 0; //gx,gy 分别为 x,y 的带权平均数
        for (int i = 1; i <= n; i++) {
            point a = p[i], b = p[nxt(i)];
            ld s = a * b / 2.0;
            gx += s * (a.x + b.x) / 3.0;
```

```
            gy += s * (a.y + b.y) / 3.0;
        }
        gx /= sum;
        gy /= sum;
        G = point(gx, gy);
    }

    inline bool check(point a, point b) { //判断圆弧 ab 是否被照射到
        point o = point(0, 0);
        for (int i = 1; i <= n; i++) {
            point c = p[i], d = p[nxt(i)];
            if (is_cross(o, a, c, d) && is_cross(o, b, c, d))
                return 0;
            // 判断 ab 是否被线段 cd 挡住
        }
        return 1;
    }
} A;

struct circle {
    point O;
    ld R;

    inline void scan() {
        int x, y, r;
        scanf("%d%d%d", &x, &y, &r);
        O = point(x, y);
        R = r;
    }
    inline void getcut() { //将太阳放在原点，求过原点的切线和圆的交点
        ld d = O.len(), a = asin(R / d);
        b[1] = O.rotate(+a) * cos(a);
        b[2] = O.rotate(-a) * cos(a);
        m = 2;
    }
    inline point getcross(point A) { // 求直线 OA 和圆的交点
        ld d = fabs(O * A) / A.len(), a = acos(d / R);
        if (O * A >= 0)
            return (A.rotate(+pi / 2.0)).rotate(+a) * R / A.len() + O;
        else
            return (A.rotate(-pi / 2.0)).rotate(-a) * R / A.len() + O;
    }
    inline ld calc(point b, point c) { // 求圆弧 bc 的长度
        point ub = b - O, uc = c - O;
        ld a = dot(ub, uc) / ub.len() / uc.len();
        return fabs(acos(a) * R);
```

```
    }
} C;

inline bool cmp(const point &a, const point &b) { // 极角排序
    return a * b > 0;
}

int main() {
    freopen("blot.in", "r", stdin);
    freopen("blot.out", "w", stdout);
    S.scan(), C.scan();
    scanf("%d%Lf%Lf%Lf", &n, &t1, &t2, &T);
    A.init(), A.getG();

    int i;
    for (i = 1; i <= n; i++) // 把太阳放在原点
        A.p[i] = A.p[i] - S;
    C.O = C.O - S;
    G = G - S;
    a1 = T / t1 * 2.0 * pi;
    a2 = T / t2 * 2.0 * pi;

    for (i = 1; i <= n; i++) // 按题意旋转
        A.p[i] = A.p[i] - G, A.p[i] = A.p[i].rotate(a2);
    G = G - C.O, G = G.rotate(a1), G = G + C.O;
    for (i = 1; i <= n; i++)
        A.p[i] = A.p[i] + G;

    C.getcut();
    for (i = 1; i <= n; i++)
        if     ((A.p[i]*b[1])*(A.p[i]*b[2])<=eps     &&     A.p[i].len()-
b[1].len()<=eps)
            b[++m] = C.getcross(A.p[i]);
    //如果直线 OA.p[i]和圆有交点并且线段 OA.p[i]和圆无交点
    sort(b + 1, b + m + 1, cmp);

    for (i = 1; i < m; i++)
        if (A.check(b[i], b[i + 1]))
            ans += C.calc(b[i], b[i + 1]);

    double fans = ans;
    printf("%.2lf\n", fans);
    return 0;
}
```

第 2 章　凸包

833.构造线段(disanti,1s,512MB)

【参考程序】

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

template <class T>
inline void read(T & res) {
    char ch;
    while (ch = getchar(), !isdigit(ch));
    res = ch ^ 48;
    while (ch = getchar(), isdigit(ch))
    res = res * 10 + (ch ^ 48);
}

template <class T>
inline void print(T x) {
    if (x > 9) print(x / 10);
    putchar(x % 10 + 48);
}

const int N = 1e4 + 5;
int n, m, cnt, top, stk[N], b[N], id[N], src, c[N], flag, d[N], tot, mid;
bool in[N];

struct line {
    int x, y;
    line(){}
    line(int _x, int _y) :
        x(_x), y(_y) {}
}a[N];

struct point
{
    ll x, y;
    point(){}
    point(ll _x, ll _y) :
        x(_x), y(_y) {}
}p[N];
```

```
inline point operator - (point a, point b) {
    return point(a.x - b.x, a.y - b.y);
}

inline ll operator * (point a, point b) {
    return a.x * b.y - a.y * b.x;
}

inline bool operator < (point a, point b) {
    return a.x < b.x || (a.x == b.x && a.y < b.y);
}

inline bool check_in(point a, point b, point c, point u)
{ //判断 u 是否在三角形 abc 中
    if ((b - a) * (c - a) < 0) swap(b, c);
    return (b-a)*(u-a) > 0 && (u-a)*(c-a) > 0 && (c-b)*(u-b) > 0;
}

inline bool which(int x) // 判断 x 是 A 类点还是 B 类点
{
    return x > n;
}

inline bool cmp(int x, int y) {
    return (p[x] - p[src]) * (p[y] - p[src]) > 0;
}

inline point del(int x, int y) {
    return p[x] - p[y];
}

inline void build() // 求所有点的凸包
{
    int i; src = 1;
    for (i = 2; i <= n + m; i++)
        if (p[i] < p[src]) src = i;

    for (i = 1; i <= n + m; i++)
        if (src != i) id[++id[0]] = i;

    sort(id + 1, id + n + m, cmp);

    stk[top = 1] = src;
    for (i = 1; i < n + m; i++)
```

```
    {
       while  (top>=2&&del(stk[top],stk[top-1])*del(id[i],stk[top-1])<=0)
           top--;
        stk[++top] = id[i];
    }

    for (i = 1; i <= top; i++)
        b[++cnt] = stk[i], in[stk[i]] = 1;

    for (i = 1; i <= n + m; i++)
        if (!in[i]) d[++tot] = i;
}

inline void nie() {
    puts("GG!");
    fclose(stdin);
    fclose(stdout);
    exit(0);
}

inline int init() // 判断凸包上是否只有一种点
{
    int i, res = 1, t = 0, s, ed;
    for (i = 2; i <= cnt; i++)
        if (which(b[i]) != which(b[1])) res = 2;

    if (res == 1) return 1;

    for (i = 1; i < cnt; i++)
        if (which(b[i]) != which(b[i + 1])) t++;

    if (t >= 3) nie();
    if (t == 1)  {
        for (i = 1; i < cnt; i++)
            if (which(b[i]) != which(b[i + 1])) mid = i;
        return 2;
    }

    for (i = 1; i < cnt; i++)
        if (which(b[i]) != which(b[i + 1])) {
            s = i + 1;
            break;
        }
```

```
    ed = cnt;
    for (i = s; i <= cnt; i++)
        if (which(b[i]) == which(b[s])) c[++c[0]] = b[i];
        else {
            ed = i - 1;
            mid = ed - s + 1;
            break;
        }

    for (i = ed + 1; i <= cnt; i++)
        c[++c[0]] = b[i];

    for (i = 1; i < s; i++)
        c[++c[0]] = b[i];

    for (i = 1; i <= cnt; i++)
        b[i] = c[i];

    return 2;
}

inline void link(int x, int y) {
    a[++a[0].x] = line(x, y);
}

inline void solve(int x, int y, int z, vector<int>g)
{  // 三角形 xyz，g 中的点在三角形内部
    if (!g.size()) return;

    int num[2] = {0, 0}, len = g.size(), count[2] = {0, 0}, i;
    num[which(x)]++;
    num[which(y)]++;
    num[which(z)]++;

    if (num[0] == 2) //z 与 x,y 种类不同
    {
        if (which(x)) swap(x, z);
        if (which(y)) swap(y, z);
    }
    else
    {
        if (!which(x)) swap(x, z);
        if (!which(y)) swap(y, z);
    }
```

```
    for (i = 0; i < len; i++)
        count[which(g[i])]++;

    if (!count[0] || !count[1]) //g 中的点类别相同
    {
        int rt;
        if (which(x) == which(g[0])) rt = x;
        else rt = z;
        for (i = 0; i < len; i++)
            link(rt, g[i]);
        return;
    }

    int u;
    for (i = 0; i < len; i++)
        if (which(g[i]) == which(z))
        {
            u = g[i];
            break;
        }

    vector<int>g1(0), g2(0), g3(0);

    for (i = 0; i < len; i++) //分成 3 个三角形
        if (g[i] != u) {
            if (check_in(p[x], p[y], p[u], p[g[i]])) g1.push_back(g[i]);
           else if (check_in(p[x],p[z],p[u], p[g[i]])) g2.push_back(g[i]);
            else g3.push_back(g[i]);
        }

    link(u, z);
    solve(x, y, u, g1);
    solve(x, z, u, g2);
    solve(y, z, u, g3);
}

inline void solveA() //凸包上只有一种点
{
    int i, x = 0, j, k;
    for (i = 1; i < cnt; i++)
        link(b[i], b[i + 1]);

    if (!tot) return;
```

```
    for (i = 1; i <= tot; i++)
        if (which(d[i]) != which(b[1])) {
            x = d[i];
            break;
        }

    if (!x) {
        for (i = 1; i <= tot; i++)
            link(d[i], b[1]);
        return;
    }

    static vector<int>g[N];
    for (i = 1; i <= cnt; i++)
        g[i].clear();

    for (i = 1; i <= tot; i++)
        if (d[i] != x)
            for (j = 1; j <= cnt; j++) {
                int u = b[j], v = b[j + 1];
                if (j == cnt) v = b[1];
               if(check_in(p[u],p[v],p[x],p[d[i]])) g[j].push_back(d[i]);
            }

    for (i = 1; i <= cnt; i++) {
        int u = b[i], v = b[i + 1];
        if (i == cnt) v = b[1];
        solve(u, v, x, g[i]);
    }
}

inline void solveB() //凸包上有两种点
{
    int i, j, rt = b[mid + 1];
    for (i = 1; i < cnt; i++)
        if (i != mid) link(b[i], b[i + 1]);

    for (i = 1; i < mid; i++) {
        int u = b[i], v = b[i + 1];
        vector<int>g(0);
        for (j = 1; j <= tot; j++)
            if (check_in(p[u], p[v], p[rt], p[d[j]])) g.push_back(d[j]);
        solve(u, v, rt, g);
    }
```

```
    rt = b[1];
    for (i = mid + 1; i < cnt; i++)
    {
        int u = b[i], v = b[i + 1];
        vector<int>g(0);
        for (j = 1; j <= tot; j++)
            if (check_in(p[u], p[v], p[rt], p[d[j]])) g.push_back(d[j]);
        solve(u, v, rt, g);
    }
}

int main()
{
    freopen("disanti.in", "r", stdin);
    freopen("disanti.out", "w", stdout);

    read(n); read(m);
    int i, j;
    for (i = 1; i <= n + m; i++)
        read(p[i].x), read(p[i].y);

    build();
    flag = init();

    if (flag == 1) solveA();
    else solveB();

    for (i = 1; i <= n + m - 2; i++)
        if (!which(a[i].x))
        print(a[i].x), putchar(' '), print(a[i].y), putchar('\n');

    for (i = 1; i <= n + m - 2; i++)
        if (which(a[i].x))
        print(a[i].x-n), putchar(' '), print(a[i].y-n), putchar('\n');

    return 0;
}
```

第 3 章　旋转卡壳

842.星际迷航(twirling,0.2s,512MB)

【参考程序】

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

template <class T>
inline void read(T &res) {
    char ch; bool flag = false; res = 0;
    while (ch = getchar(), !isdigit(ch) && ch != '-');
    ch == '-' ? flag = true : res = ch ^ 48;
    while (ch = getchar(), isdigit(ch))
        res = res * 10 + ch - 48;
    flag ? res = -res : 0;
}

const int N = 5005;
const int Maxn = 0x3f3f3f3f;
const int Minn = -0x3f3f3f3f;
int n, top, m, m0, m1, m2, m3;
int nxt[N], cur[N], stk[N];

template <class T>
inline T Max(T x, T y) {return x > y ? x : y;}
template <class T>
inline void CkMax(T &x, T y) {x < y ? x = y : 0;}

struct point {
    int x, y;

    point() {}
    point(int X, int Y):
        x(X), y(Y) {}

    inline int norm() {
        return x * x + y * y;
    }

    inline point operator + (const point &a) const {
        return point(x + a.x, y + a.y);
    }
```

```
    inline point operator - (const point &a) const {
        return point(x - a.x, y - a.y);
    }

    inline int operator * (const point &a) const {
        return x * a.y - y * a.x;
    }

    inline int operator / (const point &a) const {
        return x * a.x + y * a.y;
    }

    inline bool operator < (const point &a) const {
        return x < a.x || x == a.x && y < a.y;
    }

    inline point Rotate()
    { //逆时针旋转 90 度
        return point(-y, x);
    }
}p[N], q[N];

inline bool cmp(const int &x, const int &y) {
    point ta = p[x] - p[1],
          tb = p[y] - p[1];
    int d = ta * tb;
    return !d ? (ta.norm() < tb.norm()) : (d < 0);
}

inline double calc(point a, point b, point c, point d)
{ //计算选定四个向量的答案
    a = a.Rotate();
    c = c.Rotate().Rotate();
    d = d.Rotate().Rotate().Rotate();
   double px=(double)m0*a.x+(double)m1*b.x+(double)m2*c.x+(double)m3*d.x,
          py=(double)m0*a.y+(double)m1*b.y+(double)m2*c.y+(double)m3*d.y;
    return sqrt(px * px + py * py);
}

inline double solve(int m) {
    m0 = m + 3 >> 2;
    m1 = m + 2 >> 2;
    m2 = m + 1 >> 2;
    m3 = m >> 2;
```

```
    if (n == 1)
        return calc(q[1], q[1], q[1], q[1]);

    int k = 1, j = 1, h = 1, res = Minn;
    for (int i = 2; i <= n; ++i) {
        int tmp = (q[i] - q[1]) * (q[nxt[1]] - q[1]);
        if (tmp > res)
            res = tmp, k = i;
    }
    res = Minn;
    for (int i = nxt[1]; i != k; i = nxt[i]) {
        int tmp = (q[i] - q[1]) / (q[nxt[1]] - q[1]);
        if (tmp > res)
            res = tmp, j = i;
    }
    int tmp = (q[k] - q[1]) / (q[nxt[1]] - q[1]);
    if (tmp > res)
        j = k;
    res = Maxn;
    for (int i = k; i != 1; i = nxt[i]) {
        int tmp = (q[i] - q[1]) / (q[nxt[1]] - q[1]);
        if (tmp < res)
            res = tmp, h = i;
    }
    if (0 < res)
        h = 1;

double ans=Max(calc(q[j],q[1],q[k],q[h]),calc(q[j],q[nxt[1]],q[k],q[h]));
    for (int i = nxt[1]; i != 1; i = nxt[i])
    { //旋转卡壳
        while (nxt[k] != i && (q[nxt[k]] - q[i]) * (q[nxt[i]] - q[i])
                           > (q[k] - q[i]) * (q[nxt[i]] - q[i]))
            k = nxt[k];
        while (nxt[j] != nxt[k] && (q[nxt[j]]-q[i])/(q[nxt[i]]-q[i])
                                > (q[j]-q[i])/(q[nxt[i]]-q[i]))
            j = nxt[j];
        while (nxt[h] != nxt[i] && (q[nxt[h]]-q[i])/(q[nxt[i]]-q[i])
                                < (q[h]-q[i])/(q[nxt[i]]-q[i]))
            h = nxt[h];
        CkMax(ans, Max(calc(q[j], q[i], q[k], q[h]), calc(q[j], q[nxt[i]],
q[k], q[h])));
    }
    return ans;
}
```

```
int main() {
    freopen("twirling.in", "r", stdin);
    freopen("twirling.out", "w", stdout);

    read(n); read(m);
    for (int i = 1, x, y; i <= n; ++i) {
        read(x);
        read(y);
        p[i] = point(y + x, y - x);
    }

    int st = 1;
    for (int i = 2; i <= n; ++i)
        if (p[i] < p[st])
            st = i;
    if (st != 1)
        std::swap(p[st], p[1]);
    for (int i = 2; i <= n; ++i)
        cur[i] = i;
    std::sort(cur + 2, cur + n + 1, cmp);
    stk[top = 1] = 1;
    for (int i = 2; i <= n; ++i)
    { //求所有星球的凸包
        int x = cur[i];
       while(top>1&&(p[x]-p[stk[top-1]])*(p[stk[top]]-p[stk[top-1]])<=0)
            --top;
        stk[++top] = x;
    }
    n = top;
    for (int i = 1; i <= n; ++i)
        q[i] = p[stk[i]];
    for (int i = 1; i <= n; ++i)
        nxt[i] = i == n ? 1 : i + 1;

    double fans = 0;
    for (int i = m, im = Max(m - 3, 1); i >= im; --i)
        CkMax(fans, solve(i));
    printf("%.10lf\n", fans);

    fclose(stdin); fclose(stdout);
    return 0;
}
```

第八部分　数论

第 1 章　线性基

883.最大的割(cut,1.5s,256MB)

【参考程序】

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

template <class T>
inline void read(T &res)
{
    char ch;
    while (ch = getchar(), !isdigit(ch));
    res = ch ^ 48;
    while (ch = getchar(), isdigit(ch))
        res = res * 10 + ch - 48;
}

using std::bitset;
const int N = 505;
const int L = 1005;
int n, m, lst, T, nxt[N + L]; char s[L];
bitset<L> base[L];
bitset<N> form[L], to[N + L];

inline void Insert(bitset<L> a, bitset<N> b)
{ //将一个行向量及其点权组成插入线性基
    for (int i = 999; i >= 0; --i)
        if (a[i]) {
            if (base[i].none()) {
                base[i] = a;
                form[i] = b;
                return ;
            }
            a ^= base[i];
            b ^= form[i];
        }
    nxt[++T] = lst;
    lst = T;
    to[T] = b; //为方便删去，零行用链表存储
}
```

```
inline void Modify(int x, bitset<L> v) {
    int st = 0;
    for (int e = lst; e; e = nxt[e])
        if (to[e][x]) //优先在零行中找包含 x 点权的行向量
        {
            st = e;
            break;
        }
    if (st) {
        if (lst == st)
            lst = nxt[lst];
        else
            for (int e = lst; nxt[e]; e = nxt[e])
                if (nxt[e] == st) {
                    nxt[e] = nxt[nxt[e]];
                    break;
                }
        for (int e = lst; e; e = nxt[e])
            if (to[e][x])
                to[e] ^= to[st];
        for (int i = 0; i <= 999; ++i)
            if (form[i][x])
                form[i] ^= to[st];
        Insert(v, to[st]);
    }
    else {
        for (int i = 0; i <= 999; ++i)
            if (form[i][x])
            {
                st = i;
                break;
            }
        for (int i = st + 1; i <= 999; ++i)
            if (form[i][x])
                base[i] ^= base[st], form[i] ^= form[st];
        bitset<L> a = base[st] ^ v;
        bitset<N> b = form[st];
        base[st].reset();
        form[st].reset();
        Insert(a, b);
    }
}
```

```
inline void Query()
{ //询问线性基中的最大点权
    bitset<L> ans;
    ans.reset();
    for (int i = 999; i >= 0; --i)
        if (!ans[i] && base[i].any())
            ans ^= base[i];
    int len = 999;
    while (len > 0 && !ans[len])
        --len;
    for (int i = len; i >= 0; --i)
        putchar(ans[i] ? '1' : '0');
    putchar('\n');
}

int main()  {
    freopen("cut.in", "r", stdin);
    freopen("cut.out", "w", stdout);
    read(n); read(m);
    for (int i = 1; i <= n; ++i)
    { //初始时所有点权为0，要把所有点权都加入零行
        nxt[++T] = lst;
        lst = T;
        bitset<N> a;
        a.reset();
        a[i] = 1;
        to[T] = a;
    }
    int u, v;
    while (m--)  {
        read(u); read(v);
        bitset<L> w;
        w.reset();
        scanf("%s", s);
        for (int i = 0, im = strlen(s); i < im; ++i)
            if (s[im - i - 1] == '1')
                w[i] = 1;
        Modify(u, w);
        Modify(v, w);
        Query();
    }
    fclose(stdin); fclose(stdout);
    return 0;
}
```

第 2 章　高斯消元

893.带权的图（graph,1s,512MB）

【参考程序】

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

template <class T>
inline void read(T &x) {
    static char ch;
    while (!isdigit(ch = getchar()));
    x = ch - '0';
    while (isdigit(ch = getchar()))
        x = x * 10 + ch - '0';
}

typedef long long s64;

const int MaxN = 110;
const int MaxM = 4020;

int n, m;
s64 mod;

int ect, adj[MaxN];
int nxt[MaxM], to[MaxM];
s64 weight_a[MaxM], weight_b[MaxM];
s64 weight_p[MaxM], ans[MaxM];

#define foredge(u) for (int e = adj[u], v; v = to[e], e; e = nxt[e])

inline void add(s64 &x, const s64 &y) {
    if (x += y, x >= mod) x -= mod;
}

inline void dec(s64 &x, const s64 &y) {
    if (x -= y, x < 0) x += mod;
}

inline s64 qmul(s64 x, s64 y) { //计算 x*y%mod
    s64 res = 0;
    for (; y; y >>= 1, add(x, x))
        if (y & 1) add(res, x);
    return res;
}
```

```
inline s64 qpow(s64 x, s64 y) { //计算 x^y%mod
    s64 res = 1;
    for (; y; y >>= 1, x = qmul(x, x))
        if (y & 1) res = qmul(res, x);
    return res;
}

inline void addEdge(int u, int v, s64 a, s64 b, s64 p) {
    nxt[++ect] = adj[u], adj[u] = ect, to[ect] = v;
    weight_a[ect] = a;
    weight_b[ect] = b;
    weight_p[ect] = p;
}

s64 mat[MaxN][MaxN];

inline void Gauss() { //高斯消元
    for (int i = 1; i <= n; ++i) {
        int p = 0;
        for (int j = i; j <= n; ++j)
            if (mat[j][i]) {
                p = j; //找到一行使得其第 i 个变量的系数不为 0
                break;
            }
        if (p != i) {
            for (int j = i; j <= n + 1; ++j)
                std::swap(mat[p][j], mat[i][j]);
        }
        s64 div = qpow(mat[i][i], mod - 2);
        for (int j = i; j <= n + 1; ++j)
            mat[i][j] = qmul(mat[i][j], div);
        for (int j = 1; j <= n; ++j)
            if (j != i && mat[j][i]) {
                s64 t = mat[j][i];
                for (int k = i; k <= n + 1; ++k)
                    dec(mat[j][k], qmul(mat[i][k], t));
                //用当前的行消去其他行的第 i 个变量
            }
    }
}
int main() {
    read(n), read(m), read(mod);
    for (int i = 1; i <= m; ++i) {
```

```
        int x, y;
        s64 a, b, p;
        read(x), read(y), read(a), read(b);
        b = qpow(b, mod - 2); //这里 b 提前取逆元，免去不必要的运算
        p = qmul(a, b); //p 表示题目中 a/b 的值
        addEdge(x, y, a, b, p);
        addEdge(y, x, (mod - a) % mod, b, (mod - p) % mod);
    }
    mat[1][1] = 1; //强制 ϕ(1)=0
    for (int u = 2; u <= n; ++u) {
        foredge(u) { //对于每个点 u，枚举它的出边，列方程
            dec(mat[u][u], weight_b[e]);
            add(mat[u][v], weight_b[e]);
            dec(mat[u][n + 1], weight_p[e]);
        }
    }
    Gauss();
    for (int u = 1; u <= n; ++u) {
        foredge(u)
        if (e & 1) {
            s64 cur = mat[v][n + 1];
            dec(cur, mat[u][n + 1]);
            add(cur, weight_a[e]);
            ans[e] = qmul(cur,weight_b[e]);//根据计算出 ϕ，来得出边权 C 的值
        }
    }
    for (int e = 1; e <= ect; e += 2) {
        printf("%d\n", ans[e]);
    }
    return 0;
}
```

第 3 章　拉格朗日插值

903.染色方案（color，3s，512MB）

【参考程序】

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

template <class T>
inline void read(T &x)
{
    static char ch;
    while (!isdigit(ch = getchar()));
    x = ch - '0';
    while (isdigit(ch = getchar()))
        x = x * 10 + ch - '0';
}

template <class T>
inline void putint(T x)
{
    static char buf[15], *tail = buf;
    if (!x)
        putchar('0');
    else
    {
        for (; x; x /= 10) *++tail = x % 10 + '0';
        for (; tail != buf; --tail) putchar(*tail);
    }
}

typedef std::vector<int> vi; //用 std::vector<int>表示多项式
#define mp(x, y) std::make_pair(x, y)

const int mod = 998244353;

inline int qpow(int x, int y) //快速幂求 x^y
{
    int res = 1;
    for (; y; y >>= 1, x = 1LL * x * x % mod)
        if (y & 1)
            res = 1LL * res * x % mod;
    return res;
}

inline void add(int &x, const int &y) //模意义下的 x+=y
```

```
{
    x += y;
    if (x >= mod)
        x -= mod;
}

inline void dec(int &x, const int &y)//模意义下的 x-=y
{
    x -= y;
    if (x < 0)
        x += mod;
}

const int MaxN = 1e6 + 5;

namespace polynomial
{
    int L, P;
    int rev[MaxN];

    inline void init(int n)
    {
        P = 0, L = 1;
        while (L < n)
            L <<= 1, ++P;

        for (int i = 1; i < L; ++i)
            rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (P - 1));
    }

    inline void DFT(vi &a, int n, int opt) //给 a 做 DFT/IDFT 变换
    {
        for (int i = 0; i < n; ++i)
            if (i < rev[i])
                std::swap(a[i], a[rev[i]]);

        int g = opt == 1 ? 3 : (mod + 1) / 3;
        for (int k = 1; k < n; k <<= 1)
        {
            int omega = qpow(g, (mod - 1) / (k << 1));
            for (int i = 0; i < n; i += k << 1)
            {
                int x = 1;
                for (int j = 0; j < k; ++j)
```

```
            {
                int u = a[i + j];
                int v = 1LL * a[i + j + k] * x % mod;
                add(a[i + j] = u, v);
                dec(a[i + j + k] = u, v);
                x = 1LL * x * omega % mod;
            }
        }
    }
    if (opt == -1)
    {
        int inv = qpow(n, mod - 2);
        for (int i = 0; i < n; ++i)
            a[i] = 1LL * a[i] * inv % mod;
    }
}

inline vi mul(vi a, vi b) //将 a,b 两个多项式相乘
{
    int na = a.size(), nb = b.size();
    int n = na + nb - 1;

    init(n);
    a.resize(L), b.resize(L);

    DFT(a, L, 1), DFT(b, L, 1);
    for (int i = 0; i < L; ++i)
        a[i] = 1LL * a[i] * b[i] % mod;
    DFT(a, L, -1);

    return a.resize(n), a;
}

inline vi plus(vi a, vi b) //将 a,b 两个多项式相加
{
    vi res = a;
    int n = std::max(a.size(), b.size());
    res.resize(n);

    for (int i = 0; i < n; ++i)
        add(res[i], b[i]);
    return res;
}
}
```

```
inline vi convert(int *a, int n) //将数组转换成 vector 的形式
{
    vi res(0);
    for (int i = 0; i < n; ++i)
        res.push_back(a[i]);
    return res;
}

int n, b[MaxN];
int fac[MaxN], fac_inv[MaxN];

vi val, ans;
int c[MaxN];

inline void c_init()
{
    for (int i = 0; i < n; ++i)
    {
        c[i] = 1LL * val[i] * fac_inv[i] % mod * fac_inv[n - i - 1] %
mod;
        if ((n - i - 1) & 1)
            c[i] = c[i] ? mod - c[i] : 0;
    }
}

inline void fac_init(int n)
{
    fac[0] = 1;
    for (int i = 1; i <= n; ++i)
        fac[i] = 1LL * fac[i - 1] * i % mod;
    fac_inv[n] = qpow(fac[n], mod - 2);
    for (int i = n - 1; i >= 0; --i)
        fac_inv[i] = 1LL * fac_inv[i + 1] * (i + 1) % mod;
}

inline std::pair<vi, vi> solve(int l, int r) //分治求 M 和 A
{
    if (l == r)
    {
        vi t(2);
        t[0] = l ? mod - l : 0;
        t[1] = 1;
        return mp(t, vi(1, c[l]));
```

```
    }
    int mid = (l + r) >> 1;
    std::pair<vi, vi> pl = solve(l, mid);
    std::pair<vi, vi> pr = solve(mid + 1, r);

    using namespace polynomial;
    return mp(mul(pl.first, pr.first), plus(mul(pl.first, pr.second),
mul(pl.second, pr.first)));
}

int main()
{
    freopen("color.in", "r", stdin);
    freopen("color.out", "w", stdout);

    read(n); ++n;
    for (int i = 0; i < n; ++i)
        read(b[i]);

    fac_init(n);
    for (int i = 0; i < n; ++i)
        b[i] = 1LL * b[i] * fac_inv[i] % mod;

    val = polynomial::mul(convert(b, n), convert(fac_inv, n));
    val.resize(n); //先用卷积求出 c

    for (int i = 0; i < n; ++i)
        val[i] = 1LL * val[i] * fac[i] % mod;
    c_init();

    ans = solve(0, n - 1).second;
    for (int i = 0; i < n; ++i)
    {
        putint(ans[i]);
        putchar(" \n"[i == n - 1]);
    }

    return 0;
}
```

第 4 章　欧拉函数

911.欧拉函数(phi,2s,512MB)

【参考程序】

```
#include <bits/stdc++.h>
#define p2 p << 1
#define p3 p << 1 | 1
template <class T>
inline void read(T &res)
{
    res = 0; bool bo = 0; char c;
    while (((c = getchar()) < '0' || c > '9') && c != '-');
    if (c == '-') bo = 1; else res = c - 48;
    while ((c = getchar()) >= '0' && c <= '9')
        res = (res << 3) + (res << 1) + (c - 48);
    if (bo) res = ~res + 1;
}
const int N = 5e4 + 5, M = 3e7 + 5, djq = 1e9 + 7;
int n,q,a[N],tot,pri[N],A[N],B[N],inv[N],tmp,p[N],rt[N<<2],ToT,pr=1;
bool mark[N];
std::set<int> az[N];
struct node
{
    int lc, rc, val;
} T[M];
int qpow(int a, int b)
{
    int res = 1;
    while (b)
    {
        if (b & 1) res = 1ll * res * a % djq;
        a = 1ll * a * a % djq;
        b >>= 1;
    }
    return res;
}
void modify(int l, int r, int pos, int v, int &p) //内层修改
{
    if (!p) T[p = ++ToT].val = 1;
    T[p].val = 1ll * T[p].val * v % djq;
    if (l == r) return;
    int mid = l + r >> 1;
    if (pos <= mid) modify(l, mid, pos, v, T[p].lc);
    else modify(mid + 1, r, pos, v, T[p].rc);
```

```
}
void change(int l, int r, int x, int y, int v, int p) //外层修改
{
    if (x > y) return;
    modify(1, n, y, v, rt[p]);
    if (l == r) return;
    int mid = l + r >> 1;
    if (x <= mid) change(l, mid, x, y, v, p2);
    else change(mid + 1, r, x, y, v, p3);
}
int query(int l, int r, int s, int e, int p) //内层查询
{
    if (e < l || s > r || !p) return 1;
    if (s <= l && r <= e) return T[p].val;
    int mid = l + r >> 1;
    return 1ll * query(l, mid, s, e, T[p].lc) *
        query(mid + 1, r, s, e, T[p].rc) % djq;
}
int ask(int l, int r, int xl, int xr, int yl, int yr, int p)//外层查询
{
    if (xr < l || xl > r) return 1;
    if (xl <= l && r <= xr) return query(1, n, yl, yr, rt[p]);
    int mid = l + r >> 1;
    return 1ll * ask(l, mid, xl, xr, yl, yr, p2) *
        ask(mid + 1, r, xl, xr, yl, yr, p3) % djq;
}
void wxh(int pos, int num, int op) //在 pos 位置加入或删除一个数 num
{
    int S = sqrt(num); tmp = 0;
    for (int i = 1; i <= tot; i++)
    {
        if (pri[i] > S) break;
        if (num % pri[i]) continue;
        p[++tmp] = pri[i];
        while (num % pri[i] == 0) num /= pri[i];
    }
    if (num > 1) p[++tmp] = num;
    for (int i = 1; i <= tmp; i++)
        if (op == -1)
        {
            std::set<int>::iterator it=az[p[i]].find(pos),ti=it,tj=it;
            ti--; tj++;
            change(1,n,*ti+1,pos-1,1ll*(p[i]-1)*inv[p[i]] % djq, 1);
            change(1,n,pos+1,*tj-1,1ll*(p[i]-1)*inv[p[i]] % djq, 1);
```

```
            change(1,n,*ti+1,*tj-1,1ll*p[i]*inv[p[i] - 1] % djq, 1);
            az[p[i]].erase(it);
        }
        else
        {
            std::set<int>::iterator tj = az[p[i]].lower_bound(pos), ti
 = tj; ti--;
            change(1,n,*ti+1,*tj-1,1ll*(p[i]-1)*inv[p[i]] % djq, 1);
            change(1,n,*ti+1,pos-1,1ll*p[i] * inv[p[i]-1] % djq, 1);
            change(1,n,pos+1,*tj-1,1ll*p[i] * inv[p[i]-1] % djq, 1);
            az[p[i]].insert(pos);
        }
}
void change(int x, int v)
{
    for (; x <= n; x += x & -x)
        A[x] += v;
}
int ask(int x)
{
    int res = 0;
    for (; x; x -= x & -x)
        res += A[x];
    return res;
}
void modify(int x, int v)
{
    for (; x <= n; x += x & -x)
        B[x] = 1ll * B[x] * v % djq;
}
int query(int x)
{
    int res = 1;
    for (; x; x -= x & -x)
        res = 1ll * res * B[x] % djq;
    return res;
}
int main()
{
    #ifndef nealchentxdy
        freopen("phi.in", "r", stdin);
        freopen("phi.out", "w", stdout);
    #endif
    int op, l, r;
```

```
read(n); read(q);
for (int i = 1; i <= n; i++) read(a[i]);
for (int i = 2; i <= 40000; i++) if (!mark[i])
{
    pri[++tot] = i;
    for (int j = i << 1; j <= 40000; j += i) mark[j] = 1;
}
inv[1] = 1;
for (int i = 2; i <= 40000; i++)
    inv[i] = 1ll * (djq - djq / i) * inv[djq % i] % djq;
for (int i = 0; i <= n; i++) B[i] = 1;
for (int i = 1; i <= tot; i++) pr = 1ll * pr * (pri[i]-1) % djq
  *inv[pri[i]] % djq,az[pri[i]].insert(0),az[pri[i]].insert(n+1);
for (int i = 1; i <= n; i++)
{
    change(i, a[i]); modify(i, a[i]);
    int num = a[i], S = sqrt(num);
    for (int j = 1; j <= tot; j++)
    {
        if (pri[j] > S) break;
        if (num % pri[j]) continue;
        az[pri[j]].insert(i);
        while (num % pri[j] == 0) num /= pri[j];
    }
    if (num > 1) az[num].insert(i);
}
for (int i = 1; i <= tot; i++)
{
    std::set<int>::iterator it = az[pri[i]].begin(), ti=it; ti++;
    for (; ti != az[pri[i]].end(); ti++, it++)
        change(1,n,*it+1,*ti-1,1ll*pri[i]*inv[pri[i]-1] % djq,1);
} // 预处理每个质因子出现的位置
while (q--)
{
    read(op); read(l); read(r);
    if (!op) //修改
    {
        change(l, r-a[l]); modify(l, 1ll * r * inv[a[l]] % djq);
        wxh(l, a[l], -1); a[l] = r; wxh(l, r, 1);
    }
    else if (op == 1) // 和的 phi
    {
        int num = ask(r) - ask(l - 1), S = sqrt(num), ans = num;
        for (int i = 1; i <= tot; i++)
```

```
            {
                if (pri[i] > S) break;
                if (num % pri[i]) continue;
                ans /= pri[i], ans *= pri[i] - 1;
                while (num % pri[i] == 0) num /= pri[i];
            }
            if (num > 1) ans /= num, ans *= num - 1;
            printf("%d\n", ans);
        }
        else printf("%d\n", 1ll*query(r)*qpow(query(l-1), djq-2)
            % djq * ask(1,n,1,l,r,n,1) % djq * pr % djq); //积的 phi
    }
    return 0;
}
```

912.神秘语言（language,1s,512MB）

【参考程序】

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

#define ll long long
#define s128 __int128
#define vll vector<ll>
#define vi vector<int>
#define pb emplace_back

const int e = 1e5 + 5, mod = 1e9 + 7;

vll phi[e], ord[e], c[e];
vi d[e];
ll n = 1e5, L, R = 1e10, val[e], lst_L, lst_R, now;
bool bo[e];
int ans, pri[e], cnt, lim;

inline void add(int &x, int y) {
    (x += y) >= mod && (x -= mod);
}

inline ll ksm(ll x, ll y, ll mo) {
    ll res = 1;
    while (y) {
        if (y & 1) res = (s128)res * x % mo;
        y >>= 1;
        x = (s128)x * x % mo;
```

```
    }
    return res;
}

inline ll lcm(ll x, ll y) {
    ll g = __gcd(x, y);
    return x / g * y;
}

inline void factor(ll x, vll &a, vi &b) {
    a.clear(); b.clear();
    for (int i = 1; i <= cnt && (ll)pri[i] * pri[i] <= x; i++)
    if (x % pri[i] == 0) { //注意 pri²>x 的时候就可以退出循环了
        int p = pri[i], k = 1;
        a.pb(p); x /= p;
        while (x % p == 0) x /= p, k++;
        b.pb(k);
    }
    if (x != 1) a.pb(x), b.pb(1); //最后剩下一个最大质因子
}

inline ll calc_ord(ll mo, ll ph) //计算 ord(mo)
{
    vll a; vi b;
    factor(ph, a, b);
    int len = a.size(), i;
    ll res = ph;
    for (i = 0; i < len; i++) {
        ll p = a[i]; int k = b[i];
        while (k--) { //试除 phi(mo)的质因子
            if (ksm(2, res / p, mo) == 1) res /= p;
            else break;
        }
    }
    return res;
}

inline void init() {
    int i, j;
    for (i = 2; i <= n; i++) // 预处理 10^5 以内的质数
    if (!bo[i]) {
        pri[++cnt] = i;
        for (j = i << 1; j <= n; j += i) bo[j] = 1;
    }
```

```
}

inline void dfs(int k, int id, ll ph, ll od) // dfs 枚举因数
{
    if (k == lim) {
        if (od > 1) now += ph / od; // 注意 od=1 的不能算
        return;
    }
    int p = c[id][k], len = d[id][k];
    for (int i = 0; i <= len; i++)
    dfs(k + 1, id, ph * phi[p][i], lcm(od, ord[p][i]));
}

inline void work() {
    ans = 0;
    cin >> L >> R;
    L++; R++;
    lst_L = L; lst_R = R;
    if (~L & 1) L--; // 如果 L 是奇数，要通过 G(L-1)+1 得到 G(L)
    int i, len = R - L; ll j;
    for (i = 0; i <= len; i++) c[i].clear(), d[i].clear(), val[i] = L+i;
    for (i = 1; i <= n; i++)
    if (!bo[i]) {
        phi[i].clear(); ord[i].clear();
        phi[i].pb(1); ord[i].pb(1);
    }
    for (i = 1; i <= cnt; i++) // 对[L,R]分解质因数
    {
        int p = pri[i];
        ll s = (L - 1) / p + 1, t = R / p;
        for (j = s; j <= t; j++) {
            ll k = j * p;
            if (~k & 1) continue;
            int id = k - L, h = 0;
            ll ph = p - 1, t = p;
            c[id].pb(p);
            while (val[id] % p == 0) {
                val[id] /= p; h++;
                if (h >= phi[p].size()) {
                    phi[p].pb(ph);
                    ord[p].pb(calc_ord(t, ph));
                                // 计算 phi(p^k)和 ord(p^k)
                }
                ph *= p; t *= p;
```

```
            }
            d[id].pb(h);
        }
    }
    now = 0;
    for (i = 0; i <= len; i++)
    {
        ll ex_phi = 0, ex_ord;
        if (i + L & 1)
        {
            if (val[i] != 1)
            {
                ll p = val[i];
                ex_phi = p - 1; ex_ord = calc_ord(p, p - 1);
            }
            lim = c[i].size();
            now = 0;
            dfs(0, i, 1, 1);
            if (ex_phi) dfs(0, i, ex_phi, ex_ord);
                    // 注意 i 的最大质因子可能大于 10^5
        }
        else now++;
        if (lst_L <= i + L) add(ans, ksm(26, now % (mod - 1), mod));
    }
    if (lst_L == 1) add(ans, 1);
    cout << ans << endl;
}

int main()
{
    freopen("language.in", "r", stdin);
    freopen("language.out", "w", stdout);
    int T;
    init();
    cin >> T;
    while (T--) work();
    fclose(stdin);
    fclose(stdout);
    return 0;
}
```

第 5 章　Burnside 引理与 polya 定理

921.最小表示（string，2s，512MB）

【参考程序】

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

namespace ModuloCalculator
{
    const int mod = 1e9 + 7;

    inline void add(int &x, const int &y)
    {
        if (x += y, x >= mod)
            x -= mod;
    }
    inline void addprod(int &x, const int &y, const int &z)
    {
        x = (1LL * y * z + x) % mod;
    }
    inline int qpow(int x, int y) //快速幂求 x^y % mod
    {
        int res = 1;
        for (; y; y >>= 1, x = 1LL * x * x % mod)
            if (y & 1)
                res = 1LL * res * x % mod;
        return res;
    }
}

using std::string;
using namespace ModuloCalculator;

int gcd(int n, int m)
{
    return m ? gcd(m, n % m) : n;
}

const int MaxN = 1e3 + 5;
int pw[MaxN];
int ans[MaxN];

inline string get_min(string s) //找到环串 s 的最小表示串
{
    int len = s.length();
```

```
    s += s;

    int i = 0, j = 1;
    while (i < len && j < len)
    {
        int k = 0;
        while (k < len && s[i + k] == s[j + k])
            ++k;
        if (k == len)
            break;
        if (s[i + k] > s[j + k])
            i += k + 1;
        else
            j += k + 1;
        if (i == j)
            ++i;
    }
    return s.substr(std::min(i, j), len);
}

inline string find_less(string s) //找到字典序<=s 的最大的最小表示串
{
    if (s == get_min(s))
        return s;
      int len = s.length();
    for (int i = len - 1; i >= 0; --i)
    {
        if (s[i] == 'a')
        {
            s[i] = 'z';
            continue;
        }
        --s[i];
        if (s == get_min(s))
            return s;
        else
            s[i] = 'z';
    }
}

inline int solve_same(string s) //处理最小表示串和 s 字典序相同的情况
{
    int n = s.length();
    for (int i = 1; i <= n; ++i)
```

```
    {
        bool flg = true;
        for (int j = 0; i + j < n; ++j)
            if (s[j] != s[i + j])
            {
                flg = false;
                break;
            }
        if (flg)
            return i;
    }
    return 0;
}

inline int solve_less(string s) //处理最小表示串小于 s 字典序的情况
{
    static int f[MaxN], g[MaxN][MaxN];
    int res = 0, n = s.length();
    f[0] = 1;
    for (int i = 1; i <= n; ++i)
    {
        f[i] = 0;
        for (int j = 0; j < i; ++j)
            addprod(f[i], f[i - j - 1], 'z' - s[j]);
    }
    for (int i = 0; i < n; ++i)
        for (int j = i; j < n; ++j) //处理第一类
            addprod(res, f[i], 1LL*(s[j-i]-'a')*pw[n - j - 1]%mod);
    for (int i = 1; i <= n; ++i)
        for (int j = 0; i + j <= n; ++j)
            add(g[i][j] = 1LL*('z'-s[j])*f[i-1] % mod,g[i-1][j+1]);
    for (int i = 0; i < n - 1; ++i) //处理第二类
    {
        int cur = 0;
        for (int j = i + 1; j < n; ++j)
        {
            if (s[j] == s[cur])
                ++cur; //cur 表示当前匹配了 s[0]~s[cur-1]
            else
            {
                assert(s[j] > s[cur]);
                add(res, g[n - j - 1][cur + 1]);
                addprod(res, f[n - j - 1], s[j] - s[cur] - 1);
                cur = 0;
```

```
            }
        }
    }
    return res;
}

int n;
string str;

int calc(int d) //计算循环节长度为 d 的情况
{
    if (ans[d] != -1)
        return ans[d];
    bool can_equal = true;
    string s0 = str.substr(0, d);
    for (int i = d; i < n; i += d)
    {
        string tmp = str.substr(i, d);
        if (tmp > s0) break;
        if (tmp < s0)
        {
            can_equal = false;
            break;
        }
    }
    string less = find_less(s0);
    if (less != s0)
        can_equal = true;
    return ans[d] = solve_less(less) + (can_equal?solve_same(less):0);
}

void work()
{
    std::cin >> str;
    n = str.length();
    memset(ans, 255, sizeof(ans));

    int res = 0;
    for (int i = 1; i <= n; ++i)
    {
        int d = gcd(i, n);
        add(res, calc(d));
    }
```

```
    res = 1LL * res * qpow(n, mod - 2) % mod;
    std::cout << res << '\n';
}

int main()
{
    freopen("string.in", "r", stdin);
    freopen("string.out", "w", stdout);
    pw[0] = 1;
    for (int i = 1; i < MaxN; ++i)
        pw[i] = 26LL * pw[i - 1] % mod; //预处理 26 的次幂
      int T;
    std::cin >> T;
    while (T--)
        work();
    return 0;
}
```

923.烷烃计数（alkane,2s,512MB）

【参考程序】

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

#define ll long long
#define pb emplace_back
#define vi vector<int>

template <class t>
inline void read(t & res)
{
    char ch;
    while (ch = getchar(), !isdigit(ch));
    res = ch ^ 48;
    while (ch = getchar(), isdigit(ch))
    res = res * 10 + (ch ^ 48);
}

template <class t>
inline void print(t x)
{
    if (x > 9) print(x / 10);
    putchar(x % 10 + 48);
}
```

```
const int e = 1e5 + 5, o = (1 << 20) | 5, mod = 998244353;
int lim, rev[o], m = 100000, T, tw[o], inv6, inv2, L;

inline void add(int &x, int y) {
    (x += y) >= mod && (x -= mod);
}

inline void del(int &x, int y) {
    (x -= y) < 0 && (x += mod);
}

inline int plu(int x, int y) {
    add(x, y);
    return x;
}

inline int sub(int x, int y) {
    del(x, y);
    return x;
}

inline int ksm(int x, int y)
{
    int res = 1;
    while (y)
    {
        if (y & 1) res = (ll)res * x % mod;
        y >>= 1;
        x = (ll)x * x % mod;
    }
    return res;
}

inline void ntt(int n, vi &a, int op)
{
    int i, k, r = (op == 1 ? 3 : (mod + 1) / 3);
    for (i = 1; i < n; i++)
    if (i < rev[i]) swap(a[i], a[rev[i]]);
    for (k = 1; k < n; k <<= 1)
    {
        int w0 = ksm(r, (mod - 1) / (k << 1));
        tw[0] = 1;
        for (i = 1; i < k; i++) tw[i] = (ll)tw[i - 1] * w0 % mod;
        for (i = 0; i < n; i += k << 1)
```

```
        for (int j = 0, *fw = tw; j < k; j++, fw++)
        {
            int b = a[i + j], c = (ll)(*fw) * a[i + j + k] % mod;
            a[i + j] = plu(b, c);
            a[i + j + k] = sub(b, c);
        }
    }
    if (op == -1)
    {
        int tot = ksm(n, mod - 2);
        for (i = 0; i < n; i++) a[i] = (ll)a[i] * tot % mod;
    }
}

inline void init_rev(int n)
{
    lim = 1;
    int i, k = 0;
    while (lim < n) lim <<= 1, k++;
    for (i = 0; i < lim; i++) rev[i] = (rev[i>>1]>>1) | ((i & 1)<<k-1);
}

inline vi operator * (vi a, vi b) // 多项式乘法
{
    int i, la = a.size(), lb = b.size();
    a.resize(min(L, la)); b.resize(min(L, lb));
    la = a.size(); lb = b.size();
    init_rev(la + lb - 1);
    a.resize(lim); b.resize(lim);
    ntt(lim, a, 1); ntt(lim, b, 1);
    for (i = 0; i < lim; i++) a[i] = (ll)a[i] * b[i] % mod;
    ntt(lim, a, -1);
    a.resize(la + lb - 1);
    return a;
}

inline vi operator + (vi a, vi b)
{
    int i, len = max(a.size(), b.size());
    a.resize(len); b.resize(len);
    for (i = 0; i < len; i++) add(a[i], b[i]);
    return a;
}
```

```
inline vi operator - (vi a, vi b)
{
    int i, len = max(a.size(), b.size());
    a.resize(len); b.resize(len);
    for (i = 0; i < len; i++) del(a[i], b[i]);
    return a;
}

inline vi operator * (int v, vi a)
{
    int i, la = a.size();
    for (i = 0; i < la; i++) a[i] = (ll)a[i] * v % mod;
    return a;
}

inline vi inverse(vi a, int m) // 多项式求逆
{
    vi b, p;
    b.pb(ksm(a[0], mod - 2));
    for (int n = 1; n < m; n <<= 1) {
        p = a; p.resize(n << 1);
        b = (2 * b) - (p * b * b);
        b.resize(n << 1);
    }
    b.resize(m);
    return b;
}

inline vi mulx(vi a) {
    a.pb(0);
    int i, la = a.size();
    for (i = la - 1; i >= 1; i--) a[i] = a[i - 1];
    a[0] = 0;
    return a;
}

inline vi init(vi a, int k, int m)
{
    int i, la = a.size(), mx = min(la - 1, (m - 1) / k);
    vi b(m);
    for (i = 0; i <= mx; i++) b[i * k] = a[i];
    return b;
}
```

```
int main()
{
    freopen("alkane.in", "r", stdin);
    freopen("alkane.out", "w", stdout);
    vi a, a2, a3, a4, a5, b, c, p, q; a.pb(1);
    int n, x, i;
    inv6 = ksm(6, mod - 2);
    inv2 = ksm(2, mod - 2);
    for (n = 1; n < m; n <<= 1) // 牛顿迭代求 A(x)
    {
        L = n << 1;
        a2 = init(a, 2, n << 1);
        a3 = init(a, 3, n << 1),
        b = (a * a * a) + (3 * a2 * a) + (2 * a3);
        b = mulx(inv6 * b);
        add(b[0], 1); b = b - a; b.resize(n << 1);
        c = (a * a) + a2;
        c = mulx(inv2 * c);
        del(c[0], 1); c.resize(n << 1);
        a = a - (b * inverse(c, n << 1));
        a.resize(n << 1);
    }
    c = a;
    a2 = a5 = init(a, 2, n);
    a3 = init(a, 3, n);
    a4 = init(a, 4, n);
    p = (a*a*a*a) + (6*a*a*a2) + (3*a2*a2) + (8*a*a3) + (6*a4);
    p = mulx(ksm(24, mod - 2) * p); //求 P(x)和 Q(x)
    del(c[0], 1);
    q = inv2 * (c * c + a2);
    del(a2[0], 1);
    b = p - q + a2;
    read(T);
    while (T--)
    {
        read(x);
        print(b[x]);
        putchar('\n');
    }
    fclose(stdin); fclose(stdout);
    return 0;
}
```

第 6 章　同余方程

933.城镇漫步（town,2s,512MB）

【参考程序】

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

#define ll long long

template <class t>
inline void read(t & res)
{
    char ch;
    while (ch = getchar(), !isdigit(ch));
    res = ch ^ 48;
    while (ch = getchar(), isdigit(ch))
    res = res * 10 + (ch ^ 48);
}

template <class t>
inline void print(t x)
{
    if (x < 0) {
        putchar('-');
        x = ~x + 1;
    }
    if (x > 9) print(x / 10);
    putchar(x % 10 + 48);
}

const int e = 2e5 + 5, o = e << 1;

int adj[e], nxt[o], go[o], num, n, q, st[o][19], logn[o], dep[e],
dfn[e], tim, pos[o], m;
ll lcm;

inline void tense(ll &x, ll y)
{
    if (y == -1) return;
    if (x == -1) x = y;
    x = min(x, y);
}

inline void add(int x, int y)
{
    nxt[++num] = adj[x]; adj[x] = num; go[num] = y;
    nxt[++num] = adj[y]; adj[y] = num; go[num] = x;
}

inline void dfs(int u, int pa)
{
    dep[u] = dep[pa] + 1;
    dfn[u] = ++tim;
    pos[tim] = u;
```

```
    for (int i = adj[u]; i; i = nxt[i])
    {
        int v = go[i];
        if (v == pa) continue;
        dfs(v, u);
        pos[++tim] = u;
    }
}

inline int ckmin(int x, int y) {
    return dep[x] < dep[y] ? x : y;
}

inline int ckmax(int x, int y) {
    return dep[x] > dep[y] ? x : y;
}

inline void init()
{
    int i, j;
    logn[0] = -1;
    for (i = 1; i <= tim; i++)
        logn[i] = logn[i >> 1] + 1, st[i][0] = pos[i];
    for (j = 1; (1 << j) <= tim; j++)
    for (i = 1; i + (1 << j) - 1 <= tim; i++)
    st[i][j] = ckmin(st[i][j - 1], st[i + (1 << j - 1)][j - 1]);
}

inline int lca(int l, int r) //RMQ 求 lca
{
    l = dfn[l]; r = dfn[r];
    if (l > r) swap(l, r);
    int k = logn[r - l + 1];
    return ckmin(st[l][k], st[r - (1 << k) + 1][k]);
}

inline int dist(int x, int y)
{
    int z = lca(x, y);
    return dep[x] + dep[y] - (dep[z] << 1);
}

inline int cross(int z, int x, int y) //求路径 x →y 上离 z 最近的点
{
    int l1 = lca(z, x), l2 = lca(z, y), l3 = lca(x, y);
    return ckmax(ckmax(l1, l2), l3);
}

inline ll f(ll A, ll B, ll L, ll R)
{
    if (!L) return 0;
    if (R / A > (L - 1) / A) return (L - 1) / A + 1;
    ll k = f(B % A, A, (A - R % A) % A, (A - L % A) % A);
    return (L + k * B - 1) / A + 1;
}
```

```
inline ll calc(ll A, ll B, ll L, ll R)
{
    if (!A) //注意特判 A,L 为 0 的情况
    {
        if (!L) return 0;
        return -1;
    }
    if (!L) return 0;
    ll d = __gcd(A, B);
    if (R / d == (L - 1) / d) return -1;
    return f(A, B, L, R);
}

inline ll solve(ll a, ll m1, ll l, ll r, ll m2) //求 g 函数
{
    ll k, s = ((l - a) % m2 + m2) % m2, t = ((r - a) % m2 + m2) % m2;
    if (s <= t) k = calc(m1 % m2, m2, s, t);
    else
    {
        k = calc(m1 % m2, m2, s, m2 - 1);
        tense(k, calc(m1 % m2, m2, 0, t));
    }
    if (k == -1) return -1;
    return (k * m1 + a) % lcm;
}

inline ll find(ll x, ll y)
{
    y %= 2 * x;
    return y <= x ? y : 2 * x - y;
}

inline ll work1(int a1, int a2, int S, int b1, int b2, int T) //情况一
{
    int La = dist(a1, S), Lb =dist(b1,S),Ra=dist(a2,T),Rb = dist(b2, T),
    Ca = dist(a1, a2), Cb = dist(b1, b2), Da = Ca << 1, Db = Cb << 1;
    ll ans = -1, res;
    lcm = (ll)Da / __gcd(Da, Db) * Db;
    res = solve(La, Da, Lb, Lb, Db); tense(ans, res);
    res = solve(Ca + Ra, Da, Cb + Rb, Cb + Rb, Db); tense(ans, res);

    res = solve(La, Da, Cb + Rb, Db - Lb, Db);
    if (~res) tense(ans, res + (find(Cb, res) - Lb) / 2);

    res = solve(Lb, Db, Ca + Ra, Da - La, Da);
    if (~res) tense(ans, res + (find(Ca, res) - La) / 2);

    res = solve(Ca + Ra, Da, Lb, Cb - Rb, Db);
    if (~res) tense(ans, res + (Cb - Rb - find(Cb, res)) / 2);

    res = solve(Cb + Rb, Db, La, Ca - Ra, Da);
    if (~res) tense(ans, res + (Ca - Ra - find(Ca, res)) / 2);
    return ans;
}
```

```
inline ll work2(int a1, int a2, int S, int b1, int b2, int T) //情况二
{
    int La=dist(a2,S), Lb=dist(b1,S), Ra = dist(a1,T), Rb = dist(b2,T),
        Ca = dist(a1,a2), Cb = dist(b1,b2), Da = Ca << 1, Db = Cb << 1;
    ll ans = -1, res;
    lcm = (ll)Da / __gcd(Da, Db) * Db;
    res = solve(Ca + La, Da, Lb, Lb, Db); tense(ans, res);
    res = solve(Ra, Da, Cb + Rb, Cb + Rb, Db); tense(ans, res);

    res = solve(Ca + La, Da, Cb + Rb, Db - Lb, Db);
    if (~res) tense(ans, res + (find(Cb, res) - Lb) / 2);

    res = solve(Lb, Db, Ra, Ca - La, Da);
    if (~res) tense(ans, res + (Ca - La - find(Ca, res)) / 2);

    res = solve(Ra, Da, Lb, Cb - Rb, Db);
    if (~res) tense(ans, res + (Cb - Rb - find(Cb, res)) / 2);

    res = solve(Cb + Rb, Db, Ca + La, Da - Ra, Da);
    if (~res) tense(ans, res + (find(Ca, res) - Ra) / 2);
    return ans;
}

inline ll work(int a1, int a2, int b1, int b2)
{
    if (dist(a1, b1) & 1) return -1;
    //永远不会相遇
    int S = cross(b1, a1, a2), T = cross(b2, a1, a2);
    if (dist(b1,S) + dist(S,T) + dist(T,b2) != dist(b1,b2)) return -1;
    //两条路径没有交，非法
    if (a1 == a2 || b1 == b2) return dist(a1, b1);
    //特判路径长度为 1 的情况
    if (dist(a1, S) <= dist(a1,T)) return work1(a1, a2, S, b1, b2, T);
    else return work2(a1, a2, S, b1, b2, T);
}

int main()
{
    freopen("town.in", "r", stdin);
    freopen("town.out", "w", stdout);
    read(n);
    int i, x, y, a1, a2, b1, b2;
    for (i = 1; i < n; i++) read(x), read(y), add(x, y);
    dfs(1, 0); init();
    read(m);
    while (m--)
    {
        read(a1); read(a2); read(b1); read(b2);
        print(work(a1, a2, b1, b2));
        putchar('\n');
    }
    fclose(stdin); fclose(stdout);
    return 0;
}
```

第 7 章　莫比乌斯反演

942.树上的数（tree,2s,256MB）

【参考程序】

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

#define ll long long

template <class t>
inline void read(t & res) {
    char ch;
    while (ch = getchar(), !isdigit(ch));
    res = ch ^ 48;
    while (ch = getchar(), isdigit(ch))
    res = res * 10 + (ch ^ 48);
}

template <class t>
inline void print(t x) {
    if (x > 9) print(x / 10);
    putchar(x % 10 + 48);
}

const int e = 2e5 + 5, o = 1e6 + 5;

struct point
{
    int id, t;

    point(){}
    point(int _id, int _t) :
        id(_id), t(_t) {}
};
struct line
{
    int x, y, z;
}b[e], c[e];
bool bo[o];
int fa[e], sze[e], n, q, miu[o], L = 1e6, tim[e], vis[e];
ll ans[e], sum;
vector<int>stk, a[o];
vector<point>g[o], h[e];
```

```
inline void init() //预处理 miu 函数以及每个数的约数
{
    int i, j;
    for (i = 1; i <= L; i++) miu[i] = 1;
    for (i = 2; i <= L; i++)
    if (!bo[i]) {
        miu[i] = -1;
        a[i].push_back(i);
        for (j = i << 1; j <= L; j += i) {
            bo[j] = 1;
            if (j / i % i == 0) miu[j] = 0;
            else miu[j] = -miu[j];
            a[j].push_back(i);
        }
    }
}

inline int find(int x) {
    while (fa[x]) x = fa[x];
    return x;
}

inline void merge(int x, int y) //将边(x,y)加入可持久化并查集
{
    int fx = find(x), fy = find(y);
    if (sze[fx] < sze[fy]) swap(fx, fy);
    fa[fy] = fx;
    sum += (ll)sze[fx] * sze[fy];
    sze[fx] += sze[fy];
    stk.emplace_back(fy);
}

inline void cut() //删边
{
    int fy = stk.back(), fx = fa[fy];
    sze[fx] -= sze[fy];
    sum -= (ll)sze[fx] * sze[fy];
    fa[fy] = 0;
    stk.pop_back();
}

inline void ins(int x, int id, int t) //拆边
{
    point u = point(id, t);
```

```
    int s, len = a[x].size(), i, all = 1 << len;
    for (s = 0; s < all; s++) {
        int y = 1;
        for (i = 0; i < len; i++)
        if (s & (1 << i)) y *= a[x][i];
        g[y].emplace_back(u);
    }
}

inline void solve(int x) //计算权值为 x 的边对每个询问的贡献
{
    int i, j, len = g[x].size(), k;
    sum = 0;
    ll tot = 0;
    for (i = 0; i < len; i = j + 1)  {
        j = i - 1;
        int now_t = g[x][i].t;
        while (j < len - 1 && g[x][j + 1].t == now_t) {
            j++;
            int pos = g[x][j].id;
            merge(b[pos].x, b[pos].y);
        }
        ll tmp = miu[x] * sum;
        if (now_t != -1) {
            ans[now_t] += tmp;
            for (k = i; k <= j; k++) cut();
        }
        else {
            for (k = j + 1; k < len; k++) vis[g[x][k].t] = x;
            for (k = 0; k <= q; k++)
            if (vis[k] != x) ans[k] += tmp;
        }
    }
    while (stk.size()) cut();
}

int main()
{
    freopen("tree.in", "r", stdin);
    freopen("tree.out", "w", stdout);
    init();
    read(n);
    int i, j;
    for (i = 1; i < n; i++) read(b[i].x), read(b[i].y), read(b[i].z);
```

```
    read(q);
    for (i = 1; i <= q; i++)
       read(c[i].x), read(c[i].y), tim[c[i].x] = i;
    for (i = 1; i < n; i++)
    if (!tim[i]) ins(b[i].z, i, -1); //始终不修改的边
    for (i = 1; i <= q; i++)
    {
        point u = point(c[i].x, c[i].y);
        for (j = i; j <= q; j++)
        if (j == i || c[j].x != c[i].x) h[j].emplace_back(u);
        else break;
        bool pd = 0;
        for (j = 1; j < i; j++)
        if (c[j].x == c[i].x)
        {
            pd = 1;
            break;
        }
        if (!pd)
        {
            point v = point(c[i].x, b[c[i].x].z);
            for (j = 0; j < i; j++) h[j].emplace_back(v);
            //在时刻 j 加入一条编号为 c[i].x，边权为 b[c[i].x].z 的边
        }
    }
    for (i = 0; i <= q; i++)
    {
        int len = h[i].size();
        for (j = 0; j < len; j++) ins(h[i][j].t, h[i][j].id, i);
    }
    for (i = 1; i <= n; i++) sze[i] = 1;
    for (i = 1; i <= L; i++)
    if (g[i].size()) solve(i);
    for (i = 0; i <= q; i++) print(ans[i]), putchar('\n');
    fclose(stdin); fclose(stdout);
    return 0;
}
```

943.平方约数（math，2s，512MB）

【参考程序】

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

template <class T>
inline void read(T &x)
{
    static char ch;
    while (!isdigit(ch = getchar()));
    x = ch - '0';
    while (isdigit(ch = getchar()))
        x = x * 10 + ch - '0';
}

template <class T>
inline void putint(T x)
{
    static char buf[15], *tail = buf;
    if (!x)
        putchar('0');
    else
    {
        for (; x; x /= 10) *++tail = x % 10 + '0';
        for (; tail != buf; --tail) putchar(*tail);
    }
}

typedef unsigned int u32;

const int MaxN = 2e5 + 5;
const u32 all_and = (1 << 30) - 1;

bool sie[MaxN];
int np, pri[MaxN];
int low[MaxN], low_all[MaxN];

int miu[MaxN];
u32 s[MaxN], s2[MaxN];

inline void sieve(int n) //经过改造的线性筛，直接枚举最小质因子的次幂
{
    low[1] = low_all[1] = 1;
    miu[1] = s[1] = s2[1] = 1;
    for (int i = 2; i <= n; ++i)
```

```
    {
        if (!sie[i])
        {
            pri[++np] = i;
            for (int j = 1, cur = 1; ; ++j) //先预处理 p^k 的函数值
            {
                cur *= i;
                low[cur] = i;
                low_all[cur] = cur;

                s[cur] = j + 1;
                s2[cur] = 2 * j + 1;

                miu[cur] = -(j == 1);
                sie[cur] = j != 1;

                if (cur > n / i)
                    break;
            }
        }
        for (int j = 1; j <= np && 1LL*i*pri[j]<=n && i % pri[j]; ++j)
        {
            int p = pri[j];
            for (int cur = p; ; cur *= p) //要求 i*cur 最小质因子次幂是 cur
            {
                s[cur * i] = s[cur] * s[i];
                s2[cur * i] = s2[cur] * s2[i];
                miu[cur * i] = miu[cur] * miu[i];

                low[cur * i] = p;
                low_all[cur * i] = cur;

                sie[cur * i] = true;

                if (cur * i > n / p)
                    break;
            }
        }
    }
}

const int MaxQ = 1e4 + 5;

int S;
```

```
struct request
{
    int n, m, pos;
    inline void scan(int i)
    {
        pos = i;
        read(n), read(m);
    }
    inline bool operator < (const request &rhs) const
    {
        if (n / S == rhs.n / S)
            return m < rhs.m;
        else
            return n / S < rhs.n / S;
    } //根据 n 所在的块为第一关键字，m 为第二关键字对询问排序
}req[MaxQ];

u32 ans[MaxN], cur_ans;
u32 val[MaxN][2];
int upp[MaxN][2];

inline void uptans(int d, int i, int type, int opt)
{
    cur_ans -= (u32)miu[d] * val[d][type] * val[d][type ^ 1];
    val[d][type] += (u32)opt * s[i] * s2[i * d];
    cur_ans += (u32)miu[d] * val[d][type] * val[d][type ^ 1];
}

inline void update(int d, int type, int opt)
{
    if (opt == 1)
        uptans(d, ++upp[d][type], type, +1);
    else
        uptans(d, upp[d][type]--, type, -1);
}

inline void modify(int n, int type, int opt)
{
    int p[20], cnt = 0;
    int cur = n;
    while (cur > 1) {
        p[cnt++] = low[cur];
        cur /= low_all[cur];
    }
```

```
    int all = (1 << cnt) - 1;
    for (int s = 0; s <= all; ++s)
    {
        int d = 1;
        for (int i = 0; i < cnt; ++i)
            if (s >> i & 1)
                d *= p[i]; //枚举 n 所有的不含平方因子的约数 d

        update(d, type, opt);
    }
}

int main()
{
    freopen("math.in", "r", stdin);
    freopen("math.out", "w", stdout);

    sieve(200000);
    S = 1000;
    int Q;
    read(Q);
    for (int i = 1; i <= Q; ++i)
        req[i].scan(i);
    std::sort(req + 1, req + Q + 1);

    int curn = 0, curm = 0;
    for (int i = 1; i <= Q; ++i)
    {
        int n = req[i].n, m = req[i].m;
        while (curn < n) modify(++curn, 0, +1);
        while (curn > n) modify(curn--, 0, -1);
        while (curm < m) modify(++curm, 1, +1);
        while (curm > m) modify(curm--, 1, -1);
        ans[req[i].pos] = cur_ans;
    }
    for (int i = 1; i <= Q; ++i)
    {
        putint(ans[i] & all_and);
        putchar('\n');
    }
    return 0;
}
```

第 8 章　快速傅里叶变换 FFT

962.子集计数(acp,1s,512MB)

【参考程序】

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

template <class T>
inline void read(T &res)
{
    char ch;
    while (ch = getchar(), !isdigit(ch));
    res = ch ^ 48;
    while (ch = getchar(), isdigit(ch))
        res = res * 10 + ch - 48;
}

template <class T>
inline void put(T x)
{
    if (x > 9) put(x / 10);
    putchar(x % 10 + 48);
}

using std::vector;
const int N = 1e5 + 5;
const int M = 2e5 + 5;
const int mod = 1051721729;
const int g = 6;
const int inv_g = 175286955;
int cnt[N], b[N], p[N], pos[N], ans[N], rev[M << 1];
int m, n, pm, Q, T = 1;

inline int quick_pow(int x, int k)
{
    int res = 1;
    while (k) {
        (k & 1) ? res = 1ll * res * x % mod : 0;
        x = 1ll * x * x % mod; k >>= 1;
    }
    return res;
}
```

```
inline void add_up(int &x, int y)
{
    x += y;
    x >= mod ? x -= mod : 0;
}

inline void add_down(int &x, int y)
{
    x -= y;
    x < 0 ? x += mod : 0;
}

struct point
{
    int k, t;

    point() {}
    point(int K, int T):
        k(K), t(T) {}
};
vector<point> v[N];

#define sL s << 1
#define sR s << 1 | 1

struct poly
{
    vector<int> f;
    int fn;

    inline void Init(int nn)
    {
        fn = nn;
        f.clear();
        for (int i = 0; i <= fn; ++i)
            f.push_back(1);
    }

    inline void NTT(char opt)
    {
        int now_g = opt == 1 ? g : inv_g;
        for (int i = 0; i < fn; ++i)
            if (i < rev[i])
                std::swap(f[i], f[rev[i]]);
```

```
        for (int k = 1, w, res, u, v; k < fn; k <<= 1)
        {
            w = quick_pow(now_g, (mod - 1) / (k << 1));
            for (int i = 0; i < fn; i += k << 1)
            {
                res = 1;
                for (int j = 0; j < k; ++j)
                {
                    u = f[i + j];
                    v = 1ll * res * f[i + j + k] % mod;
                    f[i + j] = f[i + j + k] = u;
                    add_up(f[i + j], v);
                    add_down(f[i + j + k], v);
                    res = 1ll * res * w % mod;
                }
            }
        }
    }

    inline void operator *= (poly a)
    {
        int tot = fn + a.fn, k = 0, gn;
        for (gn = 1; gn <= tot; gn <<= 1)
            ++k; --k;
        for (int i = gn - fn; i >= 1; --i)
            f.push_back(0);
        for (int i = gn - a.fn; i >= 1; --i)
            a.f.push_back(0);
        fn = a.fn = gn;

        for (int i = 1; i < gn; ++i)
            rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << k);
        NTT(1); a.NTT(1);
        for (int i = 0; i < gn; ++i)
            f[i] = 1ll * f[i] * a.f[i] % mod;
        NTT(-1);

        for (int i = 0, inv = quick_pow(gn, mod - 2); i <= tot; ++i)
            f[i] = 1ll * f[i] * inv % mod;
        fn = tot;
        for (int i = gn - tot; i >= 1; --i)
            f.pop_back();
    }
}tr[M], F, tF, G;
```

```
inline void solve(int s, int l, int r)
{ //分治 NTT
    if (l == r)
        return tr[s].Init(p[l]);
    int mid = l + r >> 1;
    solve(s, l, mid);
    int rc = ++T;
    solve(rc, mid + 1, r);
    tr[s] *= tr[rc];
}

int main()
{
    freopen("acp.in", "r", stdin);
    freopen("acp.out", "w", stdout);

    read(n);
    for (int i = 1, x; i <= n; ++i)
        read(x), ++cnt[x];
    for (int i = 1; i <= n; ++i)
    {
        b[++m] = cnt[i];
        if (cnt[i]) p[++pm] = cnt[i];
    }
    solve(1, 1, pm);
    F = tr[1];

    std::sort(b + 1, b + m + 1);
    m = std::unique(b + 1, b + m + 1) - b - 1;
    for (int i = 1; i <= n; ++i)
        pos[i] = std::lower_bound(b + 1, b + m + 1, cnt[i]) - b;

    read(Q);
    for (int i = 1, x, K; i <= Q; ++i)
    {
        read(x); read(K);
        v[pos[x]].push_back(point(K, i));
    }

    ++n;
    F.f.push_back(0);
    for (int i = 0; i <= n; ++i)
        tF.f.push_back(0);
```

```
    for (int i = 1; i <= m; ++i) //模拟乘除多项式
        if (!v[i].empty())
        {
            G = F;
            for (int j = 0; j <= n; ++j)
                if (j + b[i] + 2 <= n)
                    add_up(G.f[j + b[i] + 2], mod - F.f[j]);
            for (int j = 0; j <= n; ++j)
            {
                tF.f[j] = G.f[j];
                    if (j - b[i] - 1 >= 0)
                add_up(tF.f[j], tF.f[j - b[i] - 1]);
            }
            for (int j = 0, jm = v[i].size(); j < jm; ++j)
                ans[v[i][j].t] = tF.f[v[i][j].k];
        }
    for (int i = 1; i <= Q; ++i)
        put(ans[i]), putchar('\n');

    fclose(stdin); fclose(stdout);
    return 0;
}
```

963.花钟计数（a，4s，512MB）

【参考程序】

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

const int mod = 998244353;
const int inv3 = (mod + 1) / 3;
const int N = 5e4 + 5;
const int M = 2e5 + 5;
int ex[N], f0[N], g0[N], f1[N], g1[N];
int tw[M], rev[M], x[M], y[M], z[M], a0[M], b0[M], a1[M], b1[M];
int n, ans;

inline void add(int &x, int y) {
    x += y;
    x >= mod ? x -= mod : 0;
}

inline void dec(int &x, int y) {
    x -= y;
    x < 0 ? x += mod : 0;
}
```

```
inline int quick_pow(int x, int k)
{
    int res = 1;
    while (k)  {
        if (k & 1)
            res = 1ll * res * x % mod;
        x = 1ll * x * x % mod;
        k >>= 1;
    }
    return res;
}

inline void NTT(int *f, int fm, int opt)
{
    int g = opt == 1 ? 3 : inv3;
    for (int i = 0; i < fm; ++i)
        if (i < rev[i])
            std::swap(f[i], f[rev[i]]);
    for (int k = 1; k < fm; k <<= 1)
    {
        int w = quick_pow(g, (mod - 1) / (k << 1));
        tw[0] = 1;
        for (int j = 1; j < k; ++j)
            tw[j] = 1ll * tw[j - 1] * w % mod;
        for (int i = 0; i < fm; i += k << 1)
            for (int j=0, *f1=f+i, *f2=f+i+k; j < k; ++j, ++f1, ++f2)
            {
                int u = *f1,
                    v = 1ll * tw[j] * (*f2) % mod;
                *f1 = *f2 = u;
                add(*f1, v);
                dec(*f2, v);
            }
    }
    if (opt == -1) {
        for (int i = 0, inv = quick_pow(fm, mod - 2); i < fm; ++i)
            f[i] = 1ll * f[i] * inv % mod;
    }
}

inline void solve(int l, int r) {
    if (l == r)
        return ;
```

```
    int mid = l + r >> 1;
    solve(l, mid);
    for (int i = l; i <= r; ++i) {
        int t = i - l + 1, tmp = 1ll * (t - 1) * (t - 1) % mod;
        x[t] = 1ll * tmp * ex[t - 1] % mod;
        y[t] = t > 1 ? 1ll * tmp * ex[t - 2] % mod : 0;
        z[t] = t > 2 ? 1ll * tmp * ex[t - 3] % mod : 0;
    }
    for (int i = l; i <= mid; ++i) {
        a0[i - l] = f0[i];
        b0[i - l] = g0[i];
        a1[i - l] = f1[i];
        b1[i - l] = g1[i];
    }

    int tot = r - l + 1 + mid - l, fm = 1, k = -1;
    for (fm = 1; fm <= tot; fm <<= 1, ++k);
    for (int i = r - l + 2; i < fm; ++i)
        x[i] = y[i] = z[i] = 0;
    for (int i = mid - l + 1; i < fm; ++i)
        a0[i] = b0[i] = a1[i] = b1[i] = 0;
    for (int i = 1; i < fm; ++i)
        rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << k);

    NTT(x, fm, 1), NTT(y, fm, 1), NTT(z, fm, 1);
    NTT(a0, fm, 1), NTT(b0, fm, 1), NTT(a1, fm, 1), NTT(b1, fm, 1);
    for (int i = 0; i < fm; ++i)  {
        int ta0 = a0[i], ta1 = a1[i],
            tb0 = b0[i], tb1 = b1[i];
        a0[i] = (1ll * ta0 * x[i] + 1ll * tb0 * y[i]) % mod;
        b0[i] = (1ll * ta0 * y[i] + 1ll * tb0 * z[i]) % mod;
        a1[i] = (1ll * ta1 * x[i] + 1ll * tb1 * y[i]) % mod;
        b1[i] = (1ll * ta1 * y[i] + 1ll * tb1 * z[i]) % mod;
    }
    NTT(a0, fm, -1), NTT(b0, fm, -1), NTT(a1, fm, -1), NTT(b1, fm, -1);

    for (int i = mid + 1; i <= r; ++i)  {
        add(f0[i], a0[i - l]);
        add(g0[i], b0[i - l]);
        add(f1[i], a1[i - l]);
        add(g1[i], b1[i - l]);
    }
    solve(mid + 1, r);
}
```

```
int main()
{
    freopen("a.in", "r", stdin);
    freopen("a.out", "w", stdout);

    scanf("%d", &n);
    if (n <= 2)
        return puts("0"), 0;
    ex[0] = ex[2] = 1;
    for (int i = 4; i <= n; ++i)
    {
        ex[i] = ex[i - 2];
        add(ex[i], ex[i - 4]);
    }
    for (int i = 1; i < n; ++i)
    {
        f0[i + 1] = 1ll * i * i % mod * ex[i] % mod;
        g0[i + 1] = 1ll * i * i % mod * ex[i - 1] % mod;
        f1[i + 1] = 1ll * i * i % mod * ex[i - 1] % mod;
        if (i > 1)
            g1[i + 1] = 1ll * i * i % mod * ex[i - 2] % mod;
    }
    if (n > 4)
        solve(2, n - 2);
    for (int j = 1; j < n - 2; ++j)
    {
        int tmp = 1ll * j * j % mod * (j + 1) % mod;
        ans = (1ll * f0[n - j - 1] * tmp % mod * ex[j]
             + 1ll * g0[n - j - 1] * tmp % mod * ex[j - 1]
             + 1ll * f1[n - j - 1] * tmp % mod * ex[j - 1] + ans) % mod;
        if (j > 1)
            ans = (1ll * g1[n-j-1] * tmp % mod * ex[j - 2] + ans) % mod;
    }
    ans = (1ll*(n-1)*(n-1)%mod*(ex[n-1]+ex[n-3])%mod*n%mod + ans) % mod;

    printf("%d\n", ans);
    fclose(stdin); fclose(stdout);
    return 0;
}
```